SELECCIÓN DE EMISORES DE STREAMING

Tesista: **Luciano Iglesias**

Directores: **Luis Marrone y Armando De Giusti**

Tesis presentada para obtener el grado de Magister en Redes de Datos

Facultad de Informática - Universidad Nacional de La Plata

Octubre 2013

Índice general

Página

Listado de los cuadros

Listado de Figuras

Capítulo 1

Introducción

1.1. Objetivos

1.1.1. Objetivos generales

- Investigar los aspectos vinculados a la obtención de servicios de streaming, el modelado de redes de datos y la caracterización del tráfico allí cursado.
- Determinar un mecanismo que permita establecer un orden entre los nodos que brindan un determinado servicio o recurso deseado en una red "best-effort" como es Internet.

1.1.2. Objetivos específicos

- Armar una taxonomía de los servicios que se pueden brindar en redes IP.
- Analizar diferentes formas de modelizar redes de datos, de manera tal que resulte lo más fiel posible a la realidad de una red como Internet, en cuanto a topología, velocidad de enlaces, agregación y caracterización del tráfico, congestión, etc.
- Diagramar y ejecutar simulaciones, en un modelo de red, que permitan cuantificar parámetros de comunicación (bandwidth, delay, jitter, packet loss) en diferentes tipos de servicios que se pueden brindar en dicha red.
- Establecer un criterio que permita ordenar los nodos que ofrecen el servicio deseado considerando los resultados de las simulaciones abordadas.

1.2. Alcance

El alcance de este trabajo está acotado a entornos de red IP donde no hay calidad de servicio (QoS), como es la Internet pública. Abarca servicios únicamente de streaming brindados por múltiples servidores (por lo menos más de uno) y es por eso que se puede enmarcar dentro de las redes P2P (peer to peer) o también se puede enmarcar en otros casos de múltiples servidores publicando el mismo contenido. En base a una revisión bibliográfica sobre el tema, se va a determinar la forma de modelar un escenario de simulación, la clase de tráfico que se va a cursar sobre el escenario para que este cobre sentido, la clase de tráfico que representa los flujos de streaming que se desean monitorizar y la información de dichas comunicaciones que se va a guardar para su posterior análisis.

El entorno de simulación permite diagramar la topología de la red, simular el envío de datos entre nodos, generar un archivo de traza con la información detallada de la conmutación de cada paquete por cada uno de los nodos entre el origen y el destino de la comunicación, filtrando solo aquellos flujos que son de interés para ser almacenados. A partir de esos archivos de traza se hace un análisis estadístico y de desempeño de cada uno de los flujos de streaming de interés para diagramar la estrategia que permita establecer el orden de conveniencia para la selección del nodo desde el que se va a solicitar el envío del recurso.

También son parte del alcance de este trabajo enmarcar la investigación en base a la revisión bibliográfica, resumiendo sus puntos más sobresalientes. Por otro lado, es también parte del trabajo, la selección del software que se utiliza para armar el modelo, realizar la simulación, hacer el análisis estadístico de los resultados, generar los gráficos y tablas necesarios para poder explicar dichos resultados, etc.

1.3. Motivación

La creciente demanda de servicios de streaming (audio y video bajo demanda) que utilizan como infraestructura para su distribución Internet está creciendo rápidamente, lo que despierta mucho interés. Cuando existe más de una fuente o servidor que aloja y tiene a disposición del cliente el recurso en cuestión es clave la elección del mejor o los mejores de ellos en términos de la conectividad ya que puede ser determinante para lograr o no su correcta reproducción. El tráfico, generado por el servidor elegido, debería llegar al cliente en tiempo y forma para su correcta reproducción. Las condiciones en las que se vinculan los potenciales servidores y el cliente pueden ser muy distintas entre unos y otros. Cuando existe una única alternativa no es posible tomar una decisión equivocada respecto del servidor a utilizar, ya que es el único que puede otorgar el servicio, pero si en cambio hay más de uno es interesante poder ordenarlos de acuerdo a la calidad del vínculo entre el cliente que quiere consumir ese recurso y todos los potenciales servidores que pueden brindarlo.

Es muy común encontrar servicios en Internet que nos den más de una alternativa para reproducir un archivo multimedia, y lo que hacen habitualmente los usuarios es elegir uno arbitrariamente sin tener en cuenta que la elección cuidadosa del servidor puede ser determinante en el funcionamiento correcto del servicio. Otro escenario habitual en el que pueden aparecer varias alternativas para reproducir un archivo multimedia es el de las redes P2P. Esto es así por la naturaleza misma de las redes P2P, que tienen como objetivo compartir recursos, entre los que se encuentran entre otras cosas, los archivos multimedia. Además, en las redes P2P los hosts pueden actuar tanto de servidor como de cliente, lo que va a generar muy probablemente interconexiones entre los peer (hosts miembros de la red P2P) con características muy diversas porque pueden ir desde host que están en la misma red (por ejemplo, conectados al mismo ISP) hasta hosts que requieren atravesar enlaces WAN intercontinentales.

1.4. Trabajos publicados relacionados con el tema

Se presentó un *artículo* vinculado a este trabajo de tesis en el "XVII CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN" [1]. El artículo se enfoca en el estudio e investigación de los temas vinculados al desempeño en la transferencia de información. Describe el proceso sobre cómo es dicha transferencia sobre una red simulada por software. Realiza una comparación de los desempeños alcanzados para distintas velocidades de transferencia de datos tipo streaming. Se analizan los resultados, en base a una serie de parámetros considerados como son el throughput, el delay, el packet loss y el jitter. Se concluye con una reflexión sobre qué parámetros afectan en mayor o menor medida las transferencias realizadas.

Una vez creada la red simulada (utilizando los mismos procedimientos que se explicarán en este trabajo más adelante) se procedió a iniciar en los nodos tráfico (cómo también se verá más adelante). Para poder analizar los factores involucrados en el tiempo de transferencia, en la investigación, se llevaron a cabo una serie de simulaciones. Se eligieron ciertos escenarios que a continuación se describen.

Para desarrollar un análisis empírico a través de la simulación se armaron tres categorías de tráfico, según la velocidad a la que se inyectaron los datos en la comunicación. Las tres categorías usaron un tráfico del tipo CBR (Constant Bit Rate), a diferentes velocidades. Una funcionó a 400 Kbps, otra a 1 Mbps y la otra a 1,5 Mbps. El propósito de tener las diferentes categorías es ver qué ocurre con aplicaciones que requieren esas velocidades de transferencia para funcionar de manera correcta. Dicha simulación se llevó a cabo para distintos volúmenes de datos comenzando con 1 KByte hasta 600 KBytes iterando con un incremento de 1 KByte por simulación, para cada una de las tres velocidades que fueron mencionadas antes. Por cada simulación se generó un archivo con la traza y se calcularon las siguientes métricas por cada categoría y por cada volumen de transferencia: el throughput promedio, el delay promedio, el jitter máximo y el porcentaje de packet loss.

Se generaron a partir del procesamiento de las trazas y posterior cálculo de las métricas

unos gráficos para poder analizar visualmente el desempeño en cada uno de los casos planteados. En el caso del throughput los resultados estuvieron muy por debajo del desempeño teórico para el enlace. Por ejemplo, el throughput para las transferencias más voluminosas (más de 300 KBytes), en los casos de 1 Mbps y 1,5 Mbps se concentró en torno a los 550 Kbps y 700 Kbps respectivamente, mientras que para el caso de 400 Kbps (teóricos) anduvo mejor cerca de los 350 Kbps. El delay se mantuvo relativamente constante siempre por debajo de los 80 ms para todos los casos. La estrepitosa caída del throughput, implica momentos de congestión que se pueden verificar mediante el análisis de packet loss que fue muy alto, llegando a niveles del 30 % o más para los casos de 1 Mbps y 1,5 Mbps y alrededor de un 10 % para el caso de 400 Kbps. Por otro lado el delay, da la pauta de que por momentos la red está descongestionada porque cuando los paquetes llegan a destino lo hacen con un delay razonable o relativamente estable. El jitter máximo que nos mide los peores casos de la variabilidad del delay alcanza un máximo de 35 ms.

Como conclusión se pudo observar empíricamente que el tráfico agregado desde muchas fuentes tiene momentos de rafagosidad intensa y otros momentos en que no. Hay como una oscilación entre momentos de tráfico intenso que provocan congestión y momentos de tráfico más relajados que pueden ser conmutados por la red sin problemas. Estas conclusiones se infieren a partir del análisis de un grupo de flujos, como reflejo de la forma en que el tráfico incide en ellos. Se trata de una observación indirecta, en la que se ve como es afectado un flujo de streaming por el resto del tráfico. A diferencia del tráfico TCP, que se autorregula gracias a sus mecanismos para evitar congestión, el rate de inserción del streaming es fijo (por lo menos en los casos de codecs con bitrate constantes, como se verá más adelante) lo que implica un patrón más rígido de tráfico del tipo UDP.

Este trabajo es el resultado de los primeros pasos en la investigación para desarrollar esta tesis y muestra un poco el camino recorrido hasta el primer trimestre del 2011.

1.5. Contenido del trabajo

En el capítulo 2 se clasifican las aplicaciones (software) según el uso que hacen de la red. Además, se hace hincapié en el concepto de streaming y en el tipo de aplicaciones. Se mencionan la métricas de red necesarias para la cuantificación de los resultados. También se analizan algunos de los posibles lugares de aplicación de la investigación, así como la influencia que pueden llegar a tener los codecs de audio y video en el tráfico de red. Se estudia la situación actual del tráfico en Internet y se analizan los posibles modelos de tráfico de datos que existen. Por último, se explica el funcionamiento de un simulador de red y se mencionan las ventajas y desventajas de los simuladores analizados para la realización del trabajo.

En el capítulo 3 se explica paso por paso cómo se llevó a cabo la parte experimental de la investigación. Se describe como se llevó a cabo la simulación para poder realizar el modelo planteado, tanto en la disposición de los nodos, la conectividad entre ellos y la generación de tráfico. Se describen los casos de streaming generados y se explican los motivos de la cantidad y duración de las muestras tomadas. Y por último, se explica cómo es el procesamiento de la salida del simulador.

En el capítulo 4 se calculan estadísticos a partir de los resultados y se hace un análisis comparativo a través de los histogramas del throughput y el packet loss. Se analiza el vínculo existente entre simulaciones de distinta duración. Se determina un mecanismo para establecer el orden, entre los servidores de streaming planteado en los objetivos. Y finalmente, se plantean posibles líneas de investigación futuras.

En el anexo A se describe el análisis de inferencia estadística realizado para tratar de determinar las distribuciones de probabilidad y sus parámetros a partir de las muestras.

Capítulo 2

Estado del arte

2.1. Taxonomía de las aplicaciones de red

Como en cualquier tipo de actividad de investigación es necesario poder clasificar los objetos de estudio con el fin de poder compartir datos o resultados con los pares. Tener un lenguaje en común, permite a otros, valerse de las investigaciones propias y viceversa. Las aplicaciones de software, y en particular las aplicaciones de software que hacen uso de las redes de datos, no son la excepción a esta regla. Por este motivo, y según el propósito de una aplicación de red, esta genera diferentes tipos de tráfico con patrones de envío, tamaños de paquete, y comportamiento diferentes. Para poder clasificar correctamente el tipo de tráfico que se va a analizar hay que ver las diferentes alternativas que existen. Según la perspectiva con la que se analice la aplicación estás pueden ser elásticas o inelásticas, tolerantes o intolerantes, adaptativas o no adaptativas. A continuación se explican cada una de estás clasificaciones y los requerimientos de transporte que tienen los flujos participantes. Al final se esta sección, se detalla el lugar que ocupa dentro de la clasificación el streming de audio y video.

2.1.1. Elásticas e inelásticas

Las aplicaciones elásticas pueden soportar significativas variaciones en el throughput y en el delay sin que se afecte de manera considerable su calidad. A medida que el desempeño de la red empeora, la utilización de la aplicación se degrada de una manera amigable. Estas son las tradicionales aplicaciones de transferencia de datos como, por ejemplo, son la trans-

ferencia de archivos, el correo electrónico y parte del tráfico web. Mientras que prolongados delays y fluctuaciones en el throughput pueden degradar el desempeño, el resultado efectivo de la transferencia de datos no es afectado por las condiciones desfavorables de la red. Sin embargo, ciertas restricciones pueden alcanzarse cuando estos servicios son considerados en el contexto de aplicaciones avanzadas. El tráfico elástico puede ser categorizado por los requerimientos de throughput y delay:

1. **Tráfico masivo asincrónico:** como por ejemplo el correo electrónico (smtp). Tanto delay cómo el throughput son muy relajados; por lo tanto, estas no constituyen aplicaciones avanzadas en el contexto de las aplicaciones elásticas.

2. **Tráfico masivo interactivo:** como por ejemplo las transferencias de archivos (ftp) o la web (http). Aquí el throughput no es tan relajado como en el caso anterior porque suele haber una persona a la espera de la transferencia, pero la tolerancia es alta.

3. **Tráfico interactivo a ráfagas:** como por ejemplo la administración remota (telnet, ssh) o el acceso a archivos de red (NFS). Estas aplicaciones interactivas requieren interacción humano-a-máquina casi de tiempo real e idealmente deberían tener delays de 200 ms o menos, aunque cabe la posibilidad de tolerar 300 ms o un poco más.

En general, las aplicaciones masivas interactivas también tienen requerimientos similares a los de las interactivas a ráfagas para el delay, pero es preferible o más determinante tener un throughput alto. En la mayoría de los casos, los usuarios están dispuestos a tolerar un delay que es aproximadamente proporcional al volumen de datos a ser transferido. El acceso a la web es un ejemplo de aplicación interactiva que no es de tiempo real. Sin embargo, la interacción es importante y un respuesta rápida es requerida. En el contexto de aplicaciones avanzadas, como por ejemplo, la exploración científica a gran escala, la minería y visualización masiva de datos o las aplicaciones de Grid. En estos casos, el volumen de datos y el requerimiento de datos en tiempo real (o casi en tiempo real) implica que un alto nivel de throughput y unas limitaciones más estrictas de delay son necesarias.

Las aplicaciones **inelásticas** (también llamadas aplicaciones de tiempo real) son comparativamente intolerantes al delay, al jitter (variabilidad del delay), variabilidad de throughput y errores, porque generalmente transportan algún tipo de multimedia sensible a QoS como por ejemplo voz o comandos a control remoto. Si ciertas provisiones necesarias para la QoS no son tenidas en cuenta, la calidad puede volverse inaceptable y la aplicación tal vez pierda su utilidad. Sin embargo, dependiendo de la tarea de la aplicación y los tipos de datos multimedia intervinientes, la aplicación puede operar exitosamente dentro de un rango de valores de QoS. Por ejemplo, las aplicaciones de streaming de audio y video, siendo aplicaciones interactivas, no son extremadamente sensibles al delay y el jitter, cosa que es de mucho interés para esta investigación.

2.1.2. Tolerantes e intolerantes

Dentro del mundo de las aplicaciones inelásticas, hay algunas que pueden tolerar ciertos niveles de degradación de QoS (las **tolerantes**) y pueden operar dentro de un rango suministrado de QoS con una calidad aceptable o satisfactoria. Una aplicación de video puede tolerar un cierto volumen de pérdida de paquetes sin que dichos impedimentos sean significativamente molestos al usuario. Consecuentemente, las aplicaciones tolerantes pueden ser:

1. **Adaptativas**: las aplicaciones adaptativas tolerantes pueden llegar a resistir hasta ciertos niveles de variación del delay construyendo un buffer para amortiguar el jitter, o adaptarse al bitrate disponible, la pérdida de paquetes y la congestión a través de reducciones en su forma de codificar los datos o en el bitrate de la transmisión (por ejemplo, un stream de video puede descartar algunos paquetes, cuadros o capas). Las aplicaciones adaptativas son en cierto grado capaces de ajustar su demanda de recursos con un rango de valores aceptables. La adaptación de las aplicaciones es accionada por los mecanismos apropiados que directa o indirectamente informan a la aplicación del desempeño actual de la red. La adaptación es también muy importante

en el contexto del control de congestión en Internet. Una aplicación puede adaptarse a la pérdida de paquetes, lo cual es un indicador de congestión en la red, mediante la reducción del rate de transmisión.

2. **No adaptativas**: las aplicaciones no adaptativas tolerantes no pueden adaptarse de la misma forma, pero pueden aún tolerar algunas variaciones en las prestaciones de la red. Por ejemplo, la calidad de un stream de audio o video pude degradarse por pérdida de paquetes, pero todavía ser inteligible al usuario, veedor o escuchador. Son aplicaciones que no se adaptan a los cambios de la red, pero pueden seguir funcionando en condiciones más adversas y todavía ser útiles a su función. La información de los cambios en la red no son utilizados por la aplicación para adaptarse, ya sea porque es una característica no implementada en el software o porque la naturaleza misma de la aplicación no lo permite.

Por otro lado, hay aplicaciones que fallan en lograr su tarea si sus demandas de QoS no son alcanzadas. Estas aplicaciones pueden ser llamadas **intolerantes**. Un ejemplo, de tal tipo de aplicación es el control remoto de un equipo para una misión crítica, tal como un brazo robot o instrumentos quirúrgicos. Algunas aplicaciones pueden adaptar su rate a cambios instantáneos en el throughput (adaptables al rate), mientras otros pueden ser totalmente no adaptativos.

Para poder comprender más cabalmente la clasificación de las aplicaciones de red se presenta el cuadro 2.1 que resume todas las posibilidades recién mencionadas.

En el caso particular de este trabajo, en el que se va a investigar tráfico tipo streaming, se puede ubicar dentro de la clasificación cómo aplicaciones inelásticas, tolerantes y no adaptativas. El hecho de que sea inelástica se debe a que no puede soportar variaciones muy amplias de throughput y packet loss ya que podría alcanzarse un nivel inaceptable en la prestación del servicio que está llevando a cabo. El hecho de que sea tolerante, se debe a que pese a que es una aplicación inelástica puede funcionar de manera aceptable mientras los parámetros de throughput y packet loss se mantengan dentro de un rango, o sea, no

Elásticas	Tráfico masivo asincrónico Tráfico masivo interactivo Tráfico interactivo a ráfagas	
Inelásticas	Tolerantes	Adaptativas No adaptativas
	Intolerantes	

Cuadro 2.1: Clasificación de las aplicaciones de red

es estrictamente necesario que mantengan su valor constante. Por último, el hecho de que sea una aplicación no adaptativa es más una restricción de la forma en que se llevan a cabo las simulaciones en este trabajo que una elección. Más adelante se verá que el tráfico de streaming simulado se va a generar a partir de lo que se conoce como CBR (Constant Bir Rate) y que esté va a permanecer constante durante la ejecución de la simulación independientemente del comportamiento que pueda presentar la red.

2.1.3. Streaming

Ya se mencionó al streming en la clasificación de aplicaciones, pero aún no se explicó en detalle qué es este tipo de tráfico. El streaming que pude ser tanto de voz, audio o video se refiere a la transmisión sobre una red de datos de un contenido multimedia previamente almacenado o en vivo, sin que este sea completamente descargado en el receptor antes de iniciar su reproducción. Hay una gran variedad de ejemplos en la actualidad, como los sitios web que almacenan video para ver embebidos en el navegador de Internet, lo que se conoce como video bajo demanda (VoD), los sistemas de televisión donde se puede demandar la reproducción de un determinado contenido, o la televisión digital. Los posibles escenarios de aplicación pueden variar ampliamente en términos de calidad y por ende en términos de las características de red requeridas.

La red Internet, tal como es hoy, es del tipo best-effort, lo que implica que no soporta ningún tipo de diferenciación de tráfico y por ende no puede garantizar QoS. Todo el trabajo

se basa en la suposición de que estamos en un entorno de este tipo.

Hoy en Internet existen a grandes rasgos dos tipos de servicios, por un lado aquellos que podíamos decir que están orientados a TCP, como son la web, la transferencia de archivos y el correo electrónico, enriquecido con imágenes, animaciones, etc., y además, en los últimos años ha aparecido también contenido audio-visual accesible a través de streming que comenzó siendo de una calidad moderada y que cada vez mejora más y más. También Internet comenzó a utilizarse como un medio de comunicación interactivo tanto de voz como de video de bajo costo. Siguiendo por este rumbo, los usuarios, de la mano de los desarrolladores de software y los fabricantes de hardware, están comenzando a aprovecharse del potencial ofrecido por Internet para potenciar estos nuevos usos. Se puede ver como está emergiendo una nueva generación de aplicaciones que pude revolucionar la forma en que los usuarios llevan a cabo investigaciones, trabajan en conjunto y se comunican. Podemos de alguna forma clasificar estas nuevas aplicaciones que funcionan sobre Internet como avanzadas. Estas aplicaciones de Internet avanzadas pueden ofrecer nuevas oportunidades para la comunicación y la colaboración, potenciando los mecanismos de enseñanza y aprendizaje, mejorando la forma en que los grupos de investigación interactúan para compartir datos e ideas, generando un nuevo mecanismos de distribución de contenidos audio-visuales, para entretenimiento, educación, etc.. El streaming comenzó a funcionar en Internet con una calidad moderada requiriendo un throughput de unas pocas decenas de Kbps y ha ido creciendo ofreciendo cada vez mejores calidades.

Una característica que tiene el tráfico TCP, es que hace control de congestión. Esta característica, que implementa modificando el tamaño de sus buffers de envío, según las confirmaciones que recibe del receptor de los datos que le permite detectar momentos de congestión, no están presentes de igual forma en el tráfico de streaming, que generalmente se basa en UDP como protocolo de transporte. En UDP no hay control de congestión, por lo que si se desea hacerlo, esto pasa a ser una responsabilidad de la capa de aplicación. Como se explicó antes las aplicaciones inelásticas, pueden ser adaptativas o no adaptativas. En el caso del streaming, cuando se refiere a las aplicaciones que sí son adaptativas, de alguna u

otra forma hacen algo similar a TCP ya que pueden variar el bit rate para seguir funcionando con menor throughput en un modo de calidad más bajo. En cambio, cuando el streaming está generado por aplicaciones que no son adaptativas, los momentos de congestión de la red van a ser atravesados con mayor dificultad. Es muy habitual, independientemente de si el cliente y el servidor de streaming son adaptativos o no, que se postergue el comienzo de la reproducción del streaming para poder construir un buffer que permita amortiguar las variaciones de desempeño de la red [2] [3].

2.2. Métricas de red

2.2.1. Unidades

Es habitual que exista confusión cuando se habla de unidades para cuantificar información digital. Las unidades para medir espacio de memoria, disco y/o archivos, que habitualmente utiliza la industria, es la siguiente: si se tiene por ejemplo, kilo significa 2^{10} (1024) y no 10^3 (1000) porque las memorias son siempre potencia de 2. Entonces 1 KB de memoria contiene 1024 bytes en lugar de 1000. De manera similar, 1 MB de memoria contiene 2^{20} (1048576) bytes, 1 GB contiene 2^{30} (1073741824) bytes y 1 TB contiene 2^{40} (1099511627776) bytes. Sin embargo, un enlace de comunicación de 1 Kbps transmite y/o recibe 1000 bits por segundo y un enlace de 10 Mbps transmite y recibe 10000000 bits por segundo porque estas unidades expresan potencias de 10, y no en potencias de 2. Desafortunadamente, hay una tendencia a mezclar estos dos sistemas, especialmente para los tamaños de disco. Para evitar la ambigüedad, usaremos los símbolos KB, MB, y GB para 2^{10} , 2^{20} y 2^{30} bytes respectivamente y los símbolos Kbps, Mbps y Gbps para 10^3, 10^6 y 10^9 bits por segundo respectivamente.

2.2.2. Throughput, Delay, Packet Loss y Jitter

Throughput

El throughput es la cantidad de datos transmitidos exitosamente sobre un canal de comunicaciones por unidad de tiempo. El throughput se mide habitualmente en bits por segundo, lo que sería unidad de información por bloque de tiempo. Los datos pueden ser enviados sobre un enlace físico, o también pueden atravesar varios nodos de red. Es importante diferenciar el throughput del bitrate y del goodput. El bitrate es la cantidad de datos, también medido por lo general en bits por segundo, que pueden ser transportados sobre un medio de red. La diferencia con el throughput es que este no toma en cuenta si los datos llegan o no exitosamente al destino, sino que cuantifica la capacidad que tiene el enlace para el transporte de datos. Por el otro lado, el goodput es la velocidad de despacho exitoso de datos sobre un canal de comunicaciones, al igual que el throughput, pero que además, agrega que el despacho sea a la capa de aplicación, dejando de lado todo el overhead de la pila de protocolos. Teniendo en cuenta estas consideraciones, el throughput es menor que el bitrate y su vez es mayor que goodput. En este trabajo de usa el throughput como métrica del traspaso de datos. El throughput se calcula así:

$$T = \frac{successful\_data}{transmision\_time} \tag{2.1}$$

Packet Loss

El packet loss se produce cuando un paquete que se envía por la red no llega a destino, por el motivo que sea (congestión, caída de enlaces, desaparición de rutas, errores de transmisión, etc.). El cálculo porcentual del packet loss se realiza de la siguiente manera.

$$L = [(\frac{drop\_packets}{sent\_packets})100]\,\% \tag{2.2}$$

Delay

El delay da cuenta de cuanto tiempo le lleva a un bit de datos atravesar la red desde un nodo o host hasta otro. Habitualmente se mide en segundos o fracciones de segundo. El delay se calcula restando el instante de arribo de un paquete o bit con el instante de partida de ese mismo paquete o bit:

$$D = (T_d - T_s) \tag{2.3}$$

donde, T_d es el instante de recepción del paquete en el destino y T_s es el instante de emisión del paquete en el origen. Si T_d no existe, porque el paquete no llega a destino, D no se puede calcular.

Jitter

El jitter es una medida de la variabilidad del delay que sufren los paquetes de datos de una red. La espera variable que sufren los paquetes en los routers debido a los tiempos de encolamiento, generan un delay que es también variable. Es habitual expresar el jitter como el promedio de una comunicación, así como también es de interés el jitter máximo para detectar el peor caso. El jitter se calcula así:

$$J = |D_{i+1} - D_i| \tag{2.4}$$

donde, D_i es el delay del i-ésimo paquete y $D_i + 1$ el delay del siguiente. Tanto D_i como $D_i + 1$ deben ser calculables para que J se pueda calcular.

Jitter promedio:

$$J_p = (1/packets) \sum_{i=1}^{packets} J_i \tag{2.5}$$

Jitter Máximo:

$$J_m = \max_{1 \leq i \leq packets} J_i \tag{2.6}$$

2.2.3. Métricas vinculadas al streaming

Las métricas, de las recién mencionadas, más vinculadas al streaming son el throughput y el packet loss. Esto son los dos parámetros sensibles que afectan una transmisión de streaming. Para realizar una transmisión de video con una calidad modesta, con baja resolución, es posible usar una conexión con bajo throughput y algo de packet loss. Pero a medida que los requerimientos aumentan, la necesidad de mayor throughput y bajo packet loss se incrementa. Los usuarios están acostumbrados a servicios que no están basados en IP, de los que obtienen muy buenas calidades de reproducción, y entonces esperan características similares en la calidad de estos otros servicios.

Las métricas de delay y jitter también afectan al streaming aunque valores relativamente altos son tolerados. El *buffering* en el receptor que genera una demora inicial en la reproducción permite que haya un desfasaje entre los datos del audio o video a reproducir y la reproducción en sí. En el momento en el que se está realizando la reproducción, también, se están recibiendo datos de aquello que se va a reproducir en el futuro inmediato. Si el delay se mantiene relativamente estable, aunque sea alto, el sistema puede continuar la reproducción sin problemas. De hecho, como no hay interactividad, salvo por las funciones de tipo DVD (play, stop, pause, fastfordward y rewind) que pueden tolerar uno o dos segundos para funcionar, mientras se mantenga el buffer con datos la reproducción puede hacerse sin problemas.

2.3. Escenarios de aplicación

2.3.1. Redes P2P

Una red P2P (peer to peer) es un sistema totalmente distribuido, en el cual todos los nodos son equivalentes en cuanto a su funcionalidad. Son sistemas distribuidos consistentes de nodos interconectados capaces de autoorganizarse con diferentes topologías de red con el propósito de compartir recursos tales como contenidos, ciclos de CPU, espacio de almace-

namiento y ancho de banda, capaces de adaptarse a las fallas y acomodarse a poblaciones transitorias de nodos mientras mantienen una aceptable conectividad y rendimiento, sin requerir intermediación o soporte de un servidor o autoridad global centralizada.

Aunque no existe un consenso absoluto sobre la definición de un sistema P2P, estos reúnen generalmente las siguientes características:

1. Comparten sus recursos de computación, en lugar de delegar esta tarea a un servidor centralizado. Cada nodo, al no poder descansar en la coordinación de un servidor central, para el intercambio de recursos tiene una participación activa para de manera independiente y unilateralmente desarrollar tareas para otros nodos como ubicar y almacenar contenido, rutear información, conectarse y desconectarse de nodos vecinos, encriptar y desencriptar contenido.

2. Tienen habilidad para autoorganizarse de manera colaborativa. La conexión y desconexión de los nodos a la red es frecuente. Además, son tolerantes a fallos y se adaptan a los cambios poblacionales de nodos mientras mantiene un desempeño aceptable.

Según el objetivo de la red P2P los sistemas pueden clasificarse en los siguientes grupos:

1. Comunicación y colaboración. Esta categoría incluye los sistemas que proveen una infraestructura para facilitar la comunicación directa entre los peers. Entre los ejemplos encontramos los sistemas de mensajería instantánea.

2. Computación distribuidas. Esta categoría incluye los sistemas que tienen por objetivo aprovechar los ciclos de CPU ociosos de los peers. Esto se logra dividiendo la tarea a realizar en subtareas más pequeñas para distribuir entre los peers, que estos las procesen y luego devuelvan el resultado para que este sea reunido. Una coordinación centralizada es requerida inevitablemente.

3. Distribución de contenidos. La mayoría de los sistemas P2P caen dentro de esta categoría, la cuál incluye el software para el intercambio de archivos multimedia. Estos

sistemas van desde los más básicos intercambiadores de archivos a otros más sofisticados que crean un medio de almacenamiento distribuido para asegurar y hacer eficiente la publicación de contenidos, organización, indexación, búsqueda, actualización y recuperación de los datos.

La operación de cualquier sistema P2P recae sobre una red de peers (nodos) y la interconexión entre ellos (aristas). Esta red lógica está generada por encima de la red física subyacente (habitualmente basada en IP), y generalmente se la refiere cómo la red overlay. La topología, estructura y grado de centralización de la red overlay, así cómo el ruteo y los mecanismos de localización que emplea para los mensajes y contenidos son cruciales para la operación del sistema, y afectan su tolerancia a fallos, capacidad de autoorganización, adaptación a los cambios, desempeño, escalabilidad y seguridad. Las redes overley pueden distinguirse en términos de su centralización y estructura.

A pesar de que en su forma más pura una red overlay P2P se supone totalmente descentralizada, en la práctica esto no es siempre así, por lo que existen varios grados de centralización. Existen tres categorías:

1. Arquitecturas puramente descentralizadas: todos los nodos en la red desarrollan exactamente la misma tarea, actuando tanto como clientes y como servidores, y no existe ningún tipo de coordinación central para sus actividades.

2. Arquitecturas parcialmente centralizadas: la base es la misma que en el caso de la puramente descentralizada. Algunos de los nodos, sin embargo, asumen un rol más importante, actuando como índices centrales locales para compartir archivos por los peers locales. La forma en que estos supernodos son asignados en su rol por la red varia entre los diferentes sistemas. Es importante notar que estos supernodos no constituyen un único punto de falla para la red P2P, ya que estos son asignados dinámicamente, y si fallan, la red tomará acciones para reemplazarlos por otros.

3. Arquitecturas descentralizadas híbridas: en estos sistemas, hay un servidor central que facilita la interacción entre los peers, manteniendo directorios con metadatos

que describen los archivos compartidos almacenados por los peers. Sin embargo, la interacción y el intercambio de archivos se lleva a cabo directamente entre dos peers, el servidor central facilita esta interacción realizando búsquedas e identificando los nodos que almacenan los archivos. Por supuesto que en esta arquitectura existe un único punto de falla, el servidor central, por lo que son propensos a fallas técnicas o ataques maliciosos, así como también son poco escalables.

Si la estructura de la red overlay fue creada de manera no determinística, agregando nodos y contenidos de cualquier manera, o si su creación estuvo basada en reglas específicas, entonces se pueden categorizar las redes P2P, en términos de su estructura como:

1. Desestructurada: la ubicación de los contenidos (archivos) no está relacionada a la topología subyacente. En una red de este tipo, el contenido necesita ser localizado. Los mecanismos de búsqueda van desde métodos de fuerza bruta, tales como flooding hasta que el contenido buscado sea localizado, a más sofisticados que utilizan técnicas de preservación de recursos que incluyen caminos aleatorios e índices de ruteo. Los mecanismos de búsqueda empleados en las redes desestructuradas tienen implicaciones en lo que concierne a cuestiones de disponibilidad, escalabilidad y persistencia. Los sistemas desestructurados son por lo general más apropiados para acomodarse a poblaciones transitorias de nodos.

2. Estructurados: estos surgieron principalmente en un intento por resolver los problemas de escalabilidad que los sistemas desestructurados tienen que enfrentar. En las redes estructuradas, la topología de la red overlay está perfectamente controlada y los archivos (o los apuntadores a ellos) son ubicados en lugares precisos. Estos sistemas esencialmente tienen una vinculación entre el contenido (por ejemplo, el identificador de un archivo) y su ubicación (por ejemplo, la dirección del nodo), en forma de una tabla de ruteo distribuida, de manera que las consultas puedan ser eficientemente ruteadas al nodo con el contenido deseado. Los sistemas estructurados ofrecen una solución escalable para consultas donde el identificador exacto del objeto es conocido.

Una desventaja de estos sistemas es que es difícil mantener la estructura requerida para lograr un ruteo eficiente de los mensajes en el contexto de una población de nodos muy volátil, en la cual los nodos están entrando y saliendo de manera frecuente.

Independientemente de la estructura que presente la red overlay, todos los sistemas P2P se separan en dos fases. Una primera fase donde se realiza la búsqueda del o los recursos deseados por parte de un peer. La fase de búsqueda va a estar ligada a las facilidades que pueden otorgar las diferentes implementaciones y al hecho de si son redes estructuradas o desestructuradas. La segunda fase va a ser la de consumir ese o esos recursos encontrados en la primera fase, y allí sólo van a participar los peers que van a actuar como cliente y servidor. Si el recurso en cuestión está a disposición del peer que va a actuar como cliente, en más de un peer que podría actuar como servidor entonces nos encontramos con el problema de decidir a cuál de todos los peers servidores le vamos a solicitar el recurso. Si además, la red P2P en cuestión tiene la capacidad de intercambiar tráfico de streaming entre los peers, estamos en un escenario de posible aplicación de este trabajo. Es un escenario en el que el recurso multimedia se puede enviar vía streaming y que además, hay más de una opción para elegir como servidor y se necesita una estrategia para poder decidir a cuál de todos seleccionar (u ordenar en función de la conectividad que los vincula) [4].

2.3.2. Audio y video bajo demanda

El concepto de audio bajo demanda (AoD) y de video bajo demanda (VoD) refieren a una tecnología que permite a los usuarios escuchar un audio y ver un video en el momento que lo deseen. La transferencia del audio o video se inicia a través de una acción del usuario y se transfiere a través de streaming al dispositivo que va a realizar la reproducción. Aunque la interactividad no es la característica principal de esta tecnología, es común ofrecer funciones de control como pause, stop, play, fast-forward, etc. similar a los VHS, DVD o Blu-ray.

Cuando el AoD o VoD, permiten elegir entre más de una fuente para realizar la emisión del contenido, estamos nuevamente en presencia de un posible escenario de aplicación de

este trabajo. La decisión del servidor a utilizar, hecha de una manera no arbitraria, puede resultar clave para la prestación del servicio.

2.4. Codecs de audio y video

Los codecs (abreviatura de codificación-decodificación) especifican los mecanismos para la implementación de un software, hardware o ambos, que permite transformar un flujo de datos o stream a otro y habitualmente almacenarlo en un archivo. Por ejemplo, un audio o video se puede codificar para ser almacenado en un archivo, enviado por la red, o cifrado y luego se pude decodificar para obtener el audio o video original. Cuando la decodificación de los datos devuelve una copia exacta de los datos antes de codificar se dice que el codec es sin pérdida; en cambio cuando los datos obtenidos de la decodificación no son exactamente los mismos que antes de la codificación se dice que el codec es con pérdida. Los codecs con pérdida son muy habituales ya que la información que descartan no es vital para la reproducción (obteniendo una calidad similar) y a cambio logran reducir ampliamente el tamaño del stream codificado.

Los codecs pueden ser de dos tipos: CBR (Constant Bit Rate) o VBR (Variable Bit Rate). El VBR varía la cantidad de datos de salida por segmento de tiempo, mientras que el CBR no. Con el modo de transmisión CBR, se consigue patrones de tráfico predecibles y se hace la administración de la red más sencilla. Sin embargo, cualquier tipo de ganancia lograda a través de la multiplexación estadística se pierde. Es más, debido a la naturaleza variable, del video principalmente, CBR no suele proveer buena calidad de video. Por el otro lado existen tres variantes para el VBR, sin restricciones, con restricciones y con retroalimentación. En el modo sin restricciones, la codificación opera sin tener en cuenta ninguna restricción externa, lo que provee una calidad de video casi constante. En el modo con restricciones, puede haber varias de ellas como, tamaños de buffer, tiempo de codificación, etc. El esfuerzo del codificador debe estar puesto en maximizar la calidad respetando las restricciones. Y por último, el VBR con retroalimentación, propone conocer el estado

de la red y ajustar los cambios debido a ella, como por ejemplo, regulando el bitrate de salida [2].

2.4.1. Técnicas de adaptación para audio y video

Los codecs que utilizan VBR con retroalimentación, son los más evolucionados debido a su capacidad de adaptarse a los cambios de la red. En esta sección se presenta un resumen de las técnicas más usadas para la adaptación de streams de audio y video.

Adaptación del bitrate

Adaptar el rate de transmisión de un flujo multimedia es una técnica que una aplicación puede usar para reaccionar a los cambios de disponibilidad del ancho de banda. Hay diferentes formas para hacer esta adaptación del rate. Algunas de ellas solo pueden ser aplicadas cuando los videos o audios están precodificados y almacenados para su posterior transmisión bajo demanda; otros, en cambio, pueden ser usados tanto para objetos almacenados previamente como para transmisiones en vivo.

Una forma habitual de cambiar el rate de transmisión para streams de audio es el de cambiar a otro codec. Esto se debe al hecho de que típicamente los codecs de audio producen streams CBR. Algunos codecs en capas también han aparecido.

Existen muchas formas de adaptar el bitrate de salida de un video:

- Negociación del ancho de banda: el cliente y el servidor hacen una estimación del ancho de banda disponible entre los dos antes de comenzar con la transmisión del video. Luego de la fase de negociación, el servidor transmite un stream que aproxime la capacidad estimada. Este método es inflexible pero simple, y es muy usado por productos de streaming.

- Múltiples versiones: muchos de los productos de streaming actuales soportan varias versiones codificadas del stream a diferentes bitrates. El servidor cambia al stream

que mejor aproxime las capacidades del cliente. Algunos productos soportan el cambio dinámico entre los múltiples streams. Para poder hacer esto es necesario tener puntos de sincronización en los streams. Como contra este método tiene un proceso de codificación complicado y la necesidad de espacio extra de almacenamiento para las múltiples versiones codificadas. Además, la granularidad de la adaptación del rate está limitada por el número de streams disponibles.

- Control de los parámetros de codificación: este método se puede aplicar tanto en a video almacenado como a una transmisión en vivo donde los parámetros de codificación son continuamente actualizados en base a estimaciones de un controlador de rate en el servidor. Se utilizan distintos parámetros para modificar el bitrate de salida como la resolución y la crominancia.

- Codificación con múltiples descripciones: la codificación del stream se hace en varios streams independientes, que usualmente son dos. Dichos streams pueden ser decodificados individualmente, lo que le da a esta técnica una buena protección contra las pérdidas. Si ambos streams son recibidos entonces la calidad del video decodificado es mayor.

- Codificación de video escalable jerárquicamente o por capas: este método de codificación divide el stream de video en un conjunto acumulativo de substreams (capas) en las cuales cada capa es una refinamiento de las anteriores. Los codecs más modernos (MPEG-2, MPEG-4, H.263+ y H.264) soportan la codificación en capas. Esta técnica es particularmente útil para la transferencia de video por una red ya que permite (i) adaptación del rate agregando o descartando capas de acuerdo a las condiciones de la red, (ii) permitiendo dar prioridad a las capas más bajas, que contienen las información más importante cuando hay disponible algún tipo de prioridad para los flujos (QoS) y (iii) una protección desigual puede ser empleada para proteger a las capas más bajas, y así mitigar el impacto del packet loss en las capas más importantes.

Adaptación al delay y jitter

El delay variable extremo-a-extremo puede ser enfrentado mediante la construcción de un buffer. El tipo de aplicación es un factor crucial en esto. Las aplicaciones interactivas solo pueden permitirse un buffer limitado, debido a las restricciones en los tiempos de respuesta. En cambio, y lo que es de interés para este trabajo, las aplicaciones de streaming pueden tolerar buffers de hasta varios segundos. Esta técnica puede ser usada tanto para transmisiones de audio como de video. La idea para eliminar el jitter es, en lugar de reproducir los datos apenas arriban, ir poniéndolos en el buffer durante un período de tiempo y luego sacarlo para su reproducción.

Adaptación y resistencia al packet loss

El packet loss es inevitable en las redes IP. Hacer uso de técnicas que incrementen la tolerancia de los flujos multimedia a la pérdida y aminorar los efectos en la calidad percibida es crucial para las aplicaciones. Existen numerosas técnicas para adaptarse a diferentes rates de packet loss y a diferentes patrones de packet loss en la capa de aplicación. Los métodos pueden complementarse entre ellos, utilizando diferentes técnicas y cumpliendo diferentes objetivos. La adaptación al packet loss puede ser alcanzada usando soluciones proactivas que tratan de proteger al flujo de un potencial packet loss; o usando soluciones reactivas que no protegen al flujo del packet loss en sí, sino que tratan de reducir los factores que lo causan o de reparar la parte de los datos que está dañada.

- Resistencia proactiva al packet loss
    - Hacer uso de una codificación robusta: con el objetivo de incrementar la eficacia de la compresión, los algoritmos de codificación tratan de remover cualquier redundancia en la señal explotando tanto lo espacial como lo temporal. Desafortunadamente, esta compresión del stream lo hace más vulnerable al packet loss. Una forma de hacer más robusta la compresión es permitir la existencia de redundancia en la señal de salida.

- Corrección de errores hacia adelante: es una técnica basada en el emisor de resistencia al error que transmite paquetes redundantes desde los cuales el contenido de los paquetes perdidos dentro de un bloque puede ser recuperado.
- Intercalación: es una técnica útil que se puede usar si la unidades de datos son más chicas que el tamaño de los paquetes, como es el caso de la transmisión de voz. Las unidades de datos son resecuenciadas antes de la transmisión, separadas a cierta distancia, luego, paquetizadas, transmitidas y luego reordenadas en el receptor. La pérdida de un paquete resulta en múltiples huecos en la reconstrucción de la señal lo que permite obtener algo más inteligible que si se tratara de un hueco largo, por parte del oído humano.

- Resistencia reactiva al packet loss

  - Control de congestión: generalmente el packet loss es una indicación de congestión, entonces si los flujos reaccionan reduciendo el volumen de datos de transmisión, es de esperar que el packet loss se reduzca.
  - Retransmisión: los paquetes perdidos pueden ser solicitados nuevamente al emisor. Debido a que la retransmisión involucra un delay muy alto, este método no es apropiado para las aplicaciones interactivas, pero si es una alternativa válida para el streaming de audio y video, que pude tolerar algunos segundos gracias al buffering.
  - Encubrimiento del error: es una técnica que aplica el receptor, para tratar de reducir el efecto del packet loss. Esta técnica intenta suplir los datos perdidos mediante técnicas como substitución por silencio, interpolación, repetición, etc. Esta técnica no puede ofrecer el mismo grado de resistencia que las opciones proactivas.

2.5. El tráfico en Internet

2.5.1. Volúmenes de tráfico

Se calcula que el crecimiento anual del ancho de banda en Internet es de entre un 50 % a un 60 % a nivel mundial. Los tipos de tráfico, sus características y sus distribuciones están siempre cambiando. Por ejemplo, en 2009 la mayor parte del tráfico se migró a un pequeño número de grandes proveedores de hosting, como aquellos que soportan computación en la nube. Existen además, numerosas predicciones que sostienen que en unos pocos años, la gran mayoría del tráfico de red será de streaming de audio y video. Y el crecimiento continuará debido al incremento del tamaño de las pantallas y el crecimiento de su resolución, lo que generará más tráfico aún.

Una expectativa que parece técnicamente lógica y natural es que el tráfico de streaming sea transmitido sobre UDP, probablemente usando RTP. Debido a la pérdida del control de congestión de TCP, si hay un basto incremento del streaming sobre UDP, esto podría cambiar los patrones históricos mediante los cuales el tráfico se beneficia mutuamente. Por este motivo, la evolución de la proporción observada de UDP sobre TCP en el tráfico actual de Internet es objeto de interés. En efecto, si la predicción en el crecimiento del tráfico de streaming va a eliminar de la mayoría de los flujos cualquier mecanismo de control de congestión, las consecuencias pueden ser serias. La proporción de UDP sobre TCP actual está cambiando debido al incremento de la demanda de IPTV y las aplicaciones de tiempo real basadas en UDP. Las aplicaciones de streaming de audio y video usan diversas tecnologías, por lo que no es fácil clasificarlas. En algunos casos, por ejemplo, hay aplicaciones de VoD que transmiten paquetes sobre TCP o inclusive sobre HTTP. Por último, y para complicarlo aún más, algunas aplicaciones escogen dinámicamente si usan UDP, TCP o HTTP.

De acuerdo a las expectativas de crecimiento del streaming, se espera haya un reflejo en el crecimiento de la proporción de UDP respecto de TCP. En base a datos disponibles de una variedad de mediciones, tanto en redes comerciales como académicas, se ve que

hubo un crecimiento de UDP sobre TCP, pero sin un patrón consistente de crecimiento. Se observó que tanto el tráfico TCP como UDP varió durante los años, tanto en el número de flujos, como en su duración y volumen. Pese a que hubo un crecimiento durante unos años, hoy no hay evidencia clara de que la proporción esté creciendo en favor de UDP ni tampoco de TCP. La proporción es dependiente de la popularidad de las aplicaciones y, por consecuencia, de las elecciones de los usuarios. El volumen mayoritario de tráfico está dominado por TCP, mientras que UDP tiene más flujos pero de menor volumen. En los últimos años las redes de los ISP aumentaron significativamente el tráfico P2P, mientras que las redes empresariales contienen mucho tráfico FTP. La elección de los usuarios en el trabajo es diferente al de la casa.

El surgimiento de aplicaciones que utilizan números de puerto arbitrarios, hizo que la identificación de los protocolos basadas en los números de puerto únicamente se vuelvan totalmente imprecisas. Una inspección completa del paquete es la única alternativa en la práctica para determinar de que tipo de tráfico de trata. Esta situación complica los mecanismos de medición de tráfico, ya que les incrementa la complejidad y el tiempo de computación necesarios para su ejecución.

Como se puede ver, en base a lo que pasó en los últimos años, es difícil hacer predicciones sobre lo que va a pasar. Una de las cosas que se pueden saber con claridad es que el tráfico va a aumentar en volumen debido al aumento de usuarios, y a los nuevos hábitos de consumo de tecnología de streaming, aplicaciones de tiempo real, el uso de las aplicaciones más clásicas pero con mayores volúmenes de datos, y además, la conectividad móvil, que amplia los puntos de acceso. En contraposición, es incierto cuáles van a ser los protocolos en que se va a basar ese crecimiento, por los motivos que se vieron anteriormente [5].

Cada año, Cisco realiza su Visual Networking Index (VNI) Forecast para predecir el volumen de tráfico de datos que se usarán en el mundo. La última predicción muestra que aparecerá un repunte masivo en el uso de datos, que irá desde los 369 Exabytes de tráfico IP usados en el mundo en 2011 a aproximadamente 1,3 Zettabytes en 2016. De acuerdo con Cisco, el rápido crecimiento en el tráfico de datos será producto de la proliferación de

los dispositivos conectados, la siempre creciente conectividad de banda ancha, y la gran adopción de video sobre IP a nivel mundial.

Para poner en perspectiva el crecimiento que esto representa, según cisco, el volumen de datos en 2016 será más que todo el tráfico transferido entre 1984 y 2012. Y el ritmo de incremento entre 2015 y 2016 solamente (330 Exabytes), es casi igual a todo lo transferido en 2011.

¿El motivo de esto? La enormidad, de dispositivos que se conectan a Internet. Se está siendo testigos de un fuerte rompimiento de paradigmas donde pasamos de un concepto de Internet asociada sólo a computadoras, a una misma red debiendo abastecer además a tablets, smartphones, televisores, automóviles, GPS, reproductores multimedia y casi cualquier cosa que consuma electricidad.

De hecho, de las 10,3 billones de conexiones detectadas en 2011 se espera que se pase a 18,9 billones de conexiones para 2016. Esto representa aproximadamente dos conexiones y media por cada persona del planeta. Estos nuevos dispositivos están desplazando el dominio de las PCs sobre el volumen de datos traficado. En 2011, las PCs se llevaron alrededor del 94 % de todo el tráfico consumido en Internet, cifra que se espera que caiga al 81 % para 2016.

El crecimiento masivo del video online implica que la influencia de los archivos compartidos vía P2P seguirá decreciendo a lo largo del tiempo. Mientras que los archivos P2P componen más del 77 % del consumo global de tráfico en Internet en 2011, la predicción de Cisco para el 2016 es que será del 54 %. Esto no significa que el intercambio de archivos decrecerá (de hecho, se espera que el tráfico crezca desde los 4,6 Exabytes por mes en 2011 a 10 Exabytes en 2016). Sin embargo, como porcentaje de los datos usados, el tráfico P2P crecerá mucho más lento que el streaming de video.

Las predicciones de los VNI de Cisco han sido bastante conservadas respecto al volumen de tráfico que eventualmente pueda crecer. El primer VNI de 2007, esperaba que el tráfico por mes en 2011 sea de 28,4 Exabytes, y esa predicción terminó siendo baja. El tráfico de datos fue un 7 % mayor, 30,7 Exabytes por mes [6].

Lo que se puede ver de los dos informes recién mencionados, es que el aumento del tráfico en Internet va a ser enorme y que, además, uno de los tráficos que más va a crecer es el del streaming. Sin embargo, no está claro sobre que protocolos ese tráfico en aumento va a funcionar, y eso va a depender y variar de acuerdo al éxito o fracaso de las aplicaciones de software y los proveedores de contenidos. En este trabajo, se van a hacer un grupo de suposiciones, como para fijar los criterios para hacer la definición del escenario y las simulaciones.

2.5.2. Modelos de tráfico

El tráfico de datos es la secuencia de elementos de datos en movimiento a través de un dispositivo físico. El típico elemento de datos es una secuencia continua de bits que conforman un paquete de datos. Cuando estos elementos pasan a través de un dispositivo físico, hay habitualmente una demora provocada por la recepción (generalmente acompañado del encolamiento del paquete), luego por el procesamiento (donde el paquete es analizado) y por último el despacho (donde finalmente el paquete abandona el dispositivo). Todo este procedimiento causa demoras. Este sistema físico es generalmente analizado mediante la creación de una abstracción conocida como modelo. Dicho modelo es una representación matemática precisa de la interacción de varias variables y funciones que gobiernan el sistema original. Es deseable que el modelo copie el funcionamiento del sistema original tan fielmente como sea posible en base al conocimiento que se tiene del sistema y a los conocimientos matemáticos con los que se cuenta. Al mismo tiempo, es deseable que la representación sea los más simple posible para poder analizarla. Para ello, en la mayoría de los casos, los modelos matemáticos aproximan las características reales de los modelos que están por detrás. Es común que existan varios modelos para un mismo sistema dónde los más simples son en general menos precisos que los más complejos.

Un modelo que resulta ser aceptable para la representación un router de red es el de usar una única cola FIFO. Este modelo cuanta con varias partes, por un lado el servidor que es la entidad que atiende los requerimientos, los clientes que son los que solicitan el

servicio y la cola que es el lugar donde los clientes esperan para ser atendidos. El servidor es necesario que esté todo el tiempo ocupado, sirviendo, si es que por lo menos hay un cliente en el sistema. Los clientes que arriban en un momento que el servidor está ocupado esperan en la cola para ser atendidos. El comportamiento matemático de los instantes de arribo y los tiempos de servicio para diferentes clientes son parte del modelo. Los instantes de arribo son una representación equivalente a los tiempos entre arribos y al instante del primer arribo. Generalmente, los arribos se encolan sucesivamente como entradas de los clientes y allí, dichos arribos, experimentan una posible espera para luego ser servidos antes de que se produzca su despacho exitoso por la salida. Para poder construir un modelo del arribo y posterior servicio se utilizan variables aleatorias. La naturaleza estadística de los arribos puede ser expresada de la siguiente forma, si sucesivos tiempos entre arribos (interarrival times -IATs) son independientes, una especificación de las condiciones iniciales que determine el instante de tiempo en el cual la operación de encolamiento comienza y la Función de Densidad de Probabilidad (probability density function – pdf) de los IATs son suficientes para describir completamente la naturaleza de los arribos. Usar una variable aleatoria con distribución de Pareto para los IATs es habitual para la construcción de este modelo. Esta variable aleatoria exhibe algunas importantes variaciones en sus características, según los valores de sus parámetros para su pdf. Su varianza puede ser finita o infinita. Las variables aleatorias con varianza infinita encuentran aplicación en la caracterización de tráfico de datos a ráfagas [7].

Históricamente el modelado de tráfico de red, tanto para paquetes como para arribo de conexiones se asumía como un proceso de Poisson debido a que dicho proceso posee propiedades que son teóricamente atractivas. Un número de estudios mostraron, sin embargo, que el tráfico de datos tanto para redes de área local como para redes de área amplia no siguen claramente una distribución exponencial en el interarribo de paquetes. Trabajos de principio de la década de los 90, mostraron convincentemente que el tráfico LAN se modela mejor usando un proceso autosimilar estadístico, que tiene varias diferencias con el proceso de Poisson. Para el tráfico autosimilar no hay un tamaño natural para una ráfaga, sino que

Proceso de Poisson	**Proceso autosimilar**
Arribo de conexiones TELNET	Arribo de paquetes de usuario de una sesión TELNET
Arribo de conexiones FTP	SMTP (generado por máquina)
	Canal de datos de FTP (una pequeña fracción de ráfagas largas llevan los datos)
	Arribo de solicitudes HTTP
	Tráfico Ethernet
	TCP sobre enlaces WAN

Cuadro 2.2: Clasificación de los modelos de tráfico según el protocolo

el tráfico a ráfagas aparece en un amplio rango de escalas de tiempo. En el cuadro 2.2 se muestran algunos ejemplos de tráfico que se ajustan mejor al modelo de Poisson y otros que se ajustan mejor al modelo autosimilar.

Lo que se puede observar es que existe una amplia variedad de situaciones relacionadas con redes de datos del mundo real, en los que se adecua mejor un modelo autosimilar que uno de Poisson. A partir de todos estos acontecimientos es que tiene sentido armar un escenario de simulación con tráfico autosimilar para trazar un paralelismo con lo que pasa hoy en Internet.

A modo de ejemplo veamos que pasa con el protocolo FTP. El protocolo FTP funciona de una forma que es denominada fuera de banda. Por un lado, utiliza una conexión persistente para el canal de control (envío de comandos, por el puerto 21 de TCP) y otra conexión para los datos en sí, los archivos y los listados de archivos (que se abre sólo cuando hay una transferencia real de información, por el puerto 20 de TCP). Al inicio de una sesión (el inicio del canal de control) lo llamaremos FTP, mientras que al inicio de una sesión de datos (el inicio de la transferencia de uno o varios archivos) lo llamaremos FTPDATA. Las sesiones FTP pueden ser modeladas como un proceso de Poisson, pero este no es el caso

de de las conexiones FTPDATA que llevan el mayor volumen de datos.

A diferencia de otros protocolos como TELNET o SSH, donde el que origina el proceso de arribo de paquetes está ampliamente determinado por el patrón de generación de paquetes del cliente, el proceso de arribo de paquetes para las conexiones FTPDATA está ampliamente determinado por factores de la red como el ancho de banda disponible, la congestión, y detalles del algoritmo de control de congestión de la capa de transporte. El arribo entre paquetes para tráfico FTPDATA está lejos de ser exponencial, y esto no sorprende, ya que los factores de red antes mencionados conducen a un proceso bastante diferente al de los arribos sin memoria (que es una de las condiciones necesarias para que se de un proceso de Poisson).

Cada sesión FTP genera un número arbitrario de conexiones FTPDATA. Una de las preguntas clave es cómo se distribuyen esas conexiones dentro de la sesión FTP. La distribución de los espacios (el tiempo que pasa entre que termina una conexión FTPDATA y empieza la siguiente dentro de una sesión FTP) muestra la repetida aparición de espacios cortos que pueden ser interpretados como pertenecientes a una única ráfaga de transferencia de archivos (vía los comando mget o mput). Supongamos que se considera una misma ráfaga cuando los intervalos entre transferencias no superan los 4 segundos. Utilizando esta definición de ráfaga de las conexiones FTPDATA, se observa que la distribución es heavy-tailed. Tan solo el 0,5 % de las ráfagas FTPDATA contienen entre el 30 % y el 60 % del total de bytes transferidos. En este caso hay generalmente más volumen de tráfico que se adecua mejor a un modelo autosimilar que a uno de Poisson.

El fenómeno de la autosimilaridad tiene un profundo impacto en el desempeño de las redes de datos. Estudios basados en la teoría tradicional de colas (proceso de Poisson) muestran que el crecimiento brusco del retardo en un sistema, comienza recién cuando se alcanza una utilización del 80 %, mientras que capturas de tráfico Ethernet e ISDN muestran que ese crecimiento brusco empieza antes, entre el 50 % y el 60 %. Esos problemas de desempeño se deben a la autosimilaridad estadística del tráfico. Uno de los descubrimientos más importantes que se realizó sobre Ethernet, es que a medida que la carga aumenta, el

grado de autosimilaridad aumenta. Este resultado es sin duda muy importante, ya que es justamente con mucha carga donde las cuestiones relacionadas al desempeño toman un papel preponderante. A partir de estos resultados, se empezó a abandonar la teoría de que la agregación de tráfico de datos generados a partir de la multiplexación un gran número de fuentes independientes, resulta en un proceso de Poisson.

La existencia de tráfico a ráfagas en los backbones (que se pueden modelar mediante un proceso autosimilar estadístico), producto de la agregación de flujos de múltiples fuentes, incide en el desempeño de los routers, ya que estos reciben muchos paquetes de golpe. Dichas ráfagas degradan el desempeño de los routers porque deben procesar mucha información en un período corto de tiempo, lo que agrega demoras en el encolamiento de los paquetes y termina incrementando la demora del sistema. Inclusive y debido a este incremento se produce el llenado de las colas en los routers lo que produce el descarte de paquetes y su consecuente degradación en los flujos individuales. Todo este fenómeno tiene un impacto directo en el tamaño de los buffers, con los que los routers deben contar, como para poder hacer frente a los requerimientos del tráfico en situaciones de alta carga [8].

2.6. Simuladores de red

2.6.1. Motivación del uso de simuladores

La simulación es un método muy útil para el desarrollo y puesta en marcha de una infraestructura de red. La simulación de redes de datos son hoy usadas habitualmente en todos los aspectos que tienen que ver con el diseño, configuración y análisis de una red.

En los últimos años, las redes de computadoras se han ido convirtiendo en sistemas complejos, haciendo que su análisis sea bastante difícil. Mientras que las técnicas analíticas (como por ejemplo el uso de la teoría de colas) han sido adecuadas en los primeros tiempos del surgimiento de las redes, la complejidad y velocidad del hardware de red y los protocolos cada vez más sofisticados hicieron necesaria la simulación para poder analizar

escenarios realistas. La simulación de redes es usada actualmente para varios aspectos de la investigación y desarrollo de redes, incluyendo el análisis de protocolos, verificación de los mismos, evaluación de tecnologías y arquitecturas de red nuevas, pruebas de módulos de software de red nuevos y análisis de configuraciones de escenarios desplegados.

Algunos de los usos que se le pueden dar a un software de simulación de redes de datos son:

1. Sustitución de redes reales: una red real puede ser difícil de poner en marcha (comprarla, instalarla y configurarla) de manera de poder experimentar con diferentes escenarios, especialmente para configuraciones en redes grandes. Además, es difícil recrear las condiciones de red deseadas en redes reales para poder llevar a cabo la experimentación requerida, por ejemplo, generar tráfico y patrones de congestión que sean de interés.

2. Prototipos de nuevas tecnologías de red: a menudo, nuevos protocolos de red son propuestos, tal como variantes de TCP o nuevos protocolos de multidifusión. Los diseñadores de protocolos no solo necesitan analizar las fortalezas y debilidades sino que también necesitan convencerse y convencer a otros que los nuevos protocolos propuestos funcionan tan bien o mejor que los ya existentes antes de aventurarse en realizar desarrollos reales.

3. Duplicación de redes existentes: en algunas situaciones, las simulaciones son útiles para recrear escenarios para verificar ciertas teorías o modelos o para entender mejor ciertos fenómenos. Por ejemplo, la simulación es útil para reconstruir ataques de DDoS (Distributed Denial of Service) para aprender más sobre su modo de operación y sus sensibilidades respecto de la topología de red y las condiciones del tráfico. Ya que no es posible reinyectar dicho tráfico malicioso en la red real, la simulación de la red sirve como duplicado basado en software de la red existente.

2.6.2. Arquitectura de los simuladores

Casi todos los ambientes de simulación de redes usan un método de simulación basado en eventos discretos para simular el comportamiento de una red. Esto simplemente quiere decir que se está interesado en el estado del sistema que está siendo simulado (la red) en puntos discretos en el tiempo y puede ser ignorado de manera segura los estados del sistema en instantes que se encuentran entre esos puntos discretos. Como un ejemplo, considere como modelar el acto de un router de enviar un paquete de red en un enlace punto a punto cableado para otro router conectado al mismo enlace. En el sistema físico, esto que parece un sencillo acto es de hecho bastante complicado, ya que incluye desde codificar los bits individuales en señales ópticas o eléctricas, calcular la detección de errores y la corrección de información, transmitir esas ondas en el enlace de comunicación, detectar esas ondas en el sistema final, verificar la exactitud de la señal recibida, y reconstruir el paquete en el router receptor. Sin embargo, para realizar una simulación, se pude modelar todo ese proceso simplemente señalando que el router comienza el envío del paquete en un instante T y el paquete es recibido completamente en un instante $T + \Delta T$. ΔT en este caso simplemente representa el tiempo que le lleva al emisor transmitir todos los bits individuales del paquete, más el retardo de la propagación de la señal a lo largo del enlace de transmisión hasta el receptor. El estado del sistema en cualquier momento durante el intervalo ΔT por supuesto que varía de acuerdo al medio físico, pero puede ser ignorado de manera segura dentro del simulador. Es fácil ver que, usando este esquema, lo único que se necesita cambiar es el estado del sistema, que en este ejemplo cambiaría dos veces: uno cuando el transmisor completó el envío del paquete y el otro cuando el destinatario recibió todo el paquete. Cada cambio de estado se realiza como un evento en el simulador de eventos discretos [9].

Usando un simulador de eventos discretos como se describió en el párrafo anterior, el diseño básico del loop principal del simulador de red resulta bastante simple. Las acciones necesarias y las estructuras de datos son las siguientes:

1. Una variable que contiene el valor actual de T, llamada instante de simulación. Esta variable representa el instante en el cual el estado actual del sistema es conocido y representado en el ambiente de simulación. Es inicializado con un valor arbitrario, habitualmente cero, y luego avanza de forma no decreciente a medida que el sistema progresa.

2. Una lista de eventos futuros pendientes. Esta lista contiene información sobre cambios de estado que todavía no han ocurrido, pero que son conocidos para haber sido programados para ocurrir en el futuro. Como ejemplo, está la actividad de transmisión del paquete discutida antes. Cuando un router comienza la transmisión de un paquete en el instante T, este programa dos eventos futuros en la lista de eventos pendientes. El primero es un evento en el instante $T + \Delta T_1$ para indicar que la transmisión ha sido completada, y todos los bits han sido enviados a través del enlace. Esto es una indicación de que el enlace está entonces libre para enviar otro paquete si es que hay alguno disponible. El segundo evento futuro en el instante $T + \Delta T_2$ indica que el paquete será recibido en el router receptor, lo cual necesariamente ocurrirá después de que la transmisión del paquete haya sido completada, debido al retardo de la propagación de la señal en el enlace. El instante en el futuro cuando el evento ocurrirá ($T + \Delta T$ en este caso) es conocido como en timestamp del evento. Es importante notar que los cambios de estado en la red debido a estos dos eventos futuros no son necesariamente conocidos, ni tampoco pueden ser determinados en el instante T, cuando la transmisión del paquete comienza. El estado del sistema en el instante T es por supuesto diferente que el que será en el instante $T + \Delta T_1$ cuando la transmisión es completa, o $T + \Delta T_2$ cuando el paquete es recibido. Por eficiencia, el listado de eventos futuros pendientes es mantenido en orden ascendente de acuerdo al timestamp.

3. Una vez que se tiene la variable del instante de simulación y la lista ordenada de eventos pendientes futuros, el loop principal del simulador de red realiza lo siguiente:

   a) Si no hay más eventos futuros pendientes termina.

   b) Remueve de la lista de eventos futuros pendientes al primero.

   c) Pone la variable de instante de simulación con el timestamp del elemento recién removido.

   d) Procesa los cambios de estado que ocurren debido a las acciones del evento. Estos cambios de estado pueden incluir la creación de uno o más eventos futuros pendientes adicionales. Por ejemplo, si el evento es la recepción de un paquete en un router, dicho router puede inmediatamente transmitir el paquete en un enlace diferente y programar el correspondiente evento de recepción del paquete en el próximo salto.

   e) Regresa al primer paso.

2.6.3. Simuladores

Generación del grafo

La topología física subyacente de Internet es una cuestión compleja que evoluciona con el tiempo. Desafortunadamente, desarrollar un entendimiento cabal resulta una tarea difícil. A pesar de esto existen un número de modelos de topologías, pero todavía es una pregunta abierta cuan representativas son estás topologías respecto de la actual Internet.

Para generar el grafo se utilizó el software BRITE [10] cuyo objetivo fue el de escribir una herramienta que mejore el estado del arte del momento y que permita a las topologías sintéticas reflejar con precisión varios aspectos de la topología actual de Internet (grados de distribución, estructura jerárquica, etc.). Otro de los objetivos fue el de poner en una única herramienta tantos modelos de generación como fuera posible y que además permita

interoperatividad con software como ns2 y otros. En general lo que se busco es tener una herramienta universal de generación de topologías.

Simuladores de eventos discretos

Hoy en día hay varias alternativas de simuladores de redes de datos de eventos discretos, por lo que se hizo un análisis de las características de algunos de ellos para ver cuál se adecuaba a las necesidades del desarrollo de este trabajo. Luego del análisis se eligió uno de ellos. Acá se listan los tres simuladores analizados y sus principales características:

1. GTNetS [11]
    a) Maneja un nivel de detalle por capa de red
    b) Escalabilidad limitada
    c) Requiere mucho código para hacer una simulación
    d) Está libre para su uso y descarga
2. VNUML [12]
    a) Utiliza virtualización
    b) Cada nodo es una máquina virtual
    c) Escalabilidad limitada
    d) Tiene licencia GNU GPL

3. Network simulation 2 (ns2) [13]

   a) Soporta muchos protocolos, que incluyen TCP, routing, multicast, wireless, etc.

   b) Programable vía scripts en OTcl

   c) Genera archivos de traza como salida

   d) Es muy usado en ambientes académicos

   e) Escala relativamente bien

   f) Tiene licencia GNU GPL

Finalmente, viendo las características se optó por ns2 principalmente porque escala bien para una cantidad de nodos relativamente alta, es muy usado en ambientes académicos y por lo tanto hay disponible buena cantidad de información y ayuda para su uso y, por último, porque es software libre.

En la actualidad también está disponible ns3 que es un proyecto que continúa y mejora algunos aspectos de ns2. En el momento del análisis y elección del simulador todavía no había una versión estable de ns3 por lo que no se tuvo en cuenta a la hora de probar y elegir. Al igual que ns2, ns3 es un simulador de redes de eventos discretos pero que no es una nueva versión de ns2 sino que es un nuevo proyecto, cuya API no es compatible con la de ns2. El simulador ns3 se programa en un lenguaje que no es OTcl, permite generar una salida que es compatible con el de una captura de red, permite manejar direccionamiento IP en los nodos, diseña mejores modelos para 802.11, entre muchas de la mejoras que tiene respecto de la versión 2 [14].

Capítulo 3

Creación del escenario y simulaciones

3.1. Generación del escenario

Luego de abordar el estudio teórico de la temática planteada, se diseñó un trabajo experimental con el fin de analizar los diferentes parámetros que intervienen en la transmisión de datos en una red, y de esta manera, generar una estrategia adecuada para alcanzar el objetivo que se planteó.

Para comenzar con la simulación es necesario definir una red sobre el cual llevar a cabo las simulaciones. El formato de creación de la topología de red necesita ser compatible con el simulador elegido. La generación del core de la red, que es un grafo de 200 nodos, se llevó a cabo usando el software BRITE que de manera aleatoria dispuso los nodos y los enlaces entre ellos, de forma que el grafo quede totalmente conexo. El software permite exportar el resultado a un formato compatible con la entrada de ns2.

Los routers (nodos) tienen todos un grado mayor o igual a uno lo que garantiza que todos los nodos estén conectados con al menos uno de los otros nodos. Los enlaces (aristas) tienen asignados su configuración. Son todos bidireccionales (full-duplex) y aleatoriamente se les asignó una velocidad (bitrate) entre 3 y 4 Mbps así como un delay menor a 3 milisegundos.

Hasta aquí lo que se tiene es un grafo con todos los nodos conectados al menos por una arista, y para cada una de esas aristas ya se les asignó una configuración a nivel físico y de enlace. Todo este grafo conforma el core de la red, o sea, todos los routers con sus enlaces que los interconectan a sus routers adyacentes. Con esto queda definido toda la interconexión física entre los routers y solo queda comenzar a cursar tráfico por dichos enlaces.

El mecanismo de ruteo que permite la conexión entre todos los nodos (routers o hosts) de la red, en otras palabras, que desde cualquier nodo se pueda alcanzar cualquier otro atravesando varios saltos es resuelto por el simulador ns2 al inicio de la simulación. El simulador cuando arranca establece las rutas óptimas para alcanzar todas las redes (y por ende todos los routers y hosts) y quedan establecidas de manera estática durante toda la ejecución de la simulación. No hay un protocolo de ruteo dinámico durante la ejecución de la simulación, por lo que no hay tráfico de ruteo, y todas la rutas quedan preestablecidas al inicio de la ejecución. Con esta propiedad que se configura en ns2, queda resuelta la capa de red y hace que todos sean alcanzables por todos.

Una vez programados, todos estos pasos previos a la ejecución de la simulación en sí, quedan sentadas las bases para el intercambio de tráfico entre los nodos. Las cuestiones relacionadas a las capas inferiores (física, enlace y red) ya están establecidas.

3.1.1. Generación sintética de tráfico autosimilar

Para poder completar el escenario una vez armada la red de nodos interconectados hay que cursar tráfico entre ellos. Este flujo de datos termina de completar el escenario sobre el que se van a inyectar los flujos de streaming a evaluar. Este tráfico que circula por la red se generó de manera sintética o artificial de forma tal que represente al conjunto de aplicaciones de software que interactúan con la red y envían o reciben datos de ella. Para emular todos estos flujos se usó un modelo de tráfico autosimilar. Se justificó su utilización debido a la gran cantidad de tráfico que existe en Internet que se ajusta a las características de este modelo, como ya se vio en el Capítulo 2.

Hay algunos métodos para producir tráfico con características de autosimilaridad estadística en redes de datos simuladas. El primero de estos métodos permite la generación de tráfico a partir de la multiplexación de fuentes ON/OFF, que transmiten datos a velocidad fija en los períodos de ON y dejan de transmitir en los períodos de OFF. La longitud de los períodos ON/OFF está determinada por variables aleatorias que presentan características heavy-tailed.

Un segundo método para generar tráfico autosimilar es el de utilizar un modelo de colas M/G/∞, donde los clientes llegan de acuerdo a un proceso de Poisson y tienen tiempos de servicio generados por una distribución heavy-tailed con varianza infinita. En este modelo, X_t es el número de clientes en el sistema en el instante t. El proceso de conteo $\{X_t\}_{t=0,1,2...}$ es asintóticamente autosimilar. El modelo M/G/∞ implica que multiplexar conexiones de velocidad constante que tienen instantes de arribo de conexión a partir de un proceso de Poisson y una distribución heavy-tailed para el tiempo de vida de las conexiones resulta en la generación de tráfico autosimilar.

En este trabajo se modeló el tráfico autosimilar usando el primer método mediante la agregación de fuentes ON/OFF en los nodos de la red que utilizan la distribución de Pareto para determinar la duración de los intervalos de ON y de OFF. Los paquetes son enviados a una velocidad fija durante los períodos ON, y no se envían paquetes durante los períodos de OFF. Ambos períodos son tomados de dos variables aleatorias independientes con distribución de Pareto. El tamaño de los paquetes que se envían durante el período de ON es fijo. Estas fuentes son usadas para generar tráfico agregado que exhiba dependencia de largo alcance (Long Range Dependency – LRD)[1]. Se optó por este método ya que está implementado en ns2.

En la simulación sobre ns2 es necesario establecer la parametrización inicial. Los valores que se utilizaron para el trabajo son: tiempo de ráfaga inicial (ON) de $\frac{1}{2}$ segundo, tiempo ocioso inicial (OFF) de $\frac{1}{2}$ segundo, una velocidad constante de transferencia de 200 Kbps, un tamaño de paquete fijo de 210 bytes, y como parámetro de forma de la distribución de Pareto el valor 1,2 , lo que hace que tenga una varianza infinita. [3]

El algoritmo que lleva adelante ns2 para la generación de tráfico basado en la distribución de Pareto con fuentes ON/OFF funciona de la siguiente forma (tomando como ejemplo los valores usados en este trabajo).

[1]Es un fenómeno que refleja persistencia en procesos autosimilares, dando lugar a la existencia de características de agrupamiento y ráfagas en todas las escalas temporales.

Inicialmente se ejecuta:

1. $interval = \frac{packet\_size \cdot 8}{rate} = \frac{210bytes \cdot 8bits/byte}{200Kbps} = 8,4 milisegundos$
2. $burstlen = \frac{burst\_time}{interval} = \frac{500}{8,4} = 59$

Luego, en cada ronda ON/OFF, se van computando las dos variables aleatorias independientes con distribución de Pareto:

1. *next_burstlen*: número de paquetes a ser transmitidos en el siguiente período de ráfaga
2. *next_idle_time*: siguiente período ocioso en segundos

Cada ronda del algoritmo de generación de tráfico de Pareto consiste en los siguientes pasos:

1. Se computa el *next_burstlen* de acuerdo al burstlen y al parámetro de forma de Pareto
2. Se envían *next_burstlen* paquetes (cada intervalo de transmisión de paquetes es de *interval* segundos)
3. Se calcula el *next_idle_time* usando *idle_time* y el parámetro de forma de Pareto
4. Se va a dormir por un período de *next_idle_time* y regresa luego al paso 1

En general, si una variable X tiene distribución de Pareto, y f es la pdf y E el valor esperado, entonces:

$$f(x) = \frac{a \cdot b^a}{x^{a+1}} \quad para \quad x \geq b \tag{3.1}$$

$$E(X) = \frac{b \cdot a}{a-1} \quad si \quad a > 1 \tag{3.2}$$

donde, a es llamado el parámetro de forma de Pareto y b es llamado el parámetro de escala. El generador de tráfico de Pareto se utiliza de la siguiente forma (a es el parámetro de forma en las siguientes ecuaciones):

$$burstlen = E(X) = \frac{b_1 \cdot a}{a-1} \tag{3.3}$$

$$idle\_time = E(Y) = \frac{b_2 \cdot a}{a-1} \tag{3.4}$$

Además:

$$b_1 = \frac{burstlen \cdot (a-1)}{a} \tag{3.5}$$

$$b_2 = \frac{idle\_time \cdot (a-1)}{a} \tag{3.6}$$

ns2 tiene un generador de números aleatorios de Pareto que recibe los parámetros de forma y escala:

```
double pareto(double scale, double shape);
```

Cuando el generador de tráfico de Pareto necesita computar next_burstlen, hace lo siguiente:

```
int next_burstlen = int(pareto(b1, a) + 0.5);
/* next_burstlen should be at least 1 packet */
if(next_burstlen == 0) next_burstlen = 1;
```

Cuando el generador de tráfico de Pareto necesita computar next_idle_time, hace lo siguiente:

```
double next_idle_time = pareto(b2, a);
```

De este algoritmo se ejecutan una o más instancias en cada uno de los 200 nodos que componen el core de la red. El tráfico de cada una de estas instancias se origina en el nodo y tiene como destino algún otro nodo cualquiera que se encuentra a uno o más saltos de distancia. La agregación de todos estos flujos en los enlaces de comunicación entre los nodos,

sumado al patrón de ráfagas de los períodos de ON y de OFF, cuya duración depende de dos variables aleatorias independientes con distribución de Pareto (con un parámetro de forma que hace el desvío estándar infinito) generan un escenario con tráfico autosimilar estadístico. Este modelo refleja la naturaleza de ráfagas de los paquetes de datos, identificado como Long Range Dependency.

3.2. Esquema de las simulaciones

3.2.1. Simulaciones

Una vez establecido el escenario, como se explicó recién, se anexaron a la nube de routers los hosts que intercambiaron los flujos de streaming. Estos dos hosts se conectaron, cada uno, vía un único vínculo a la nube de routers y fueron los encargados de realizar las transferencias de streaming, uno como emisor y el otro como receptor. Estos hosts se conectaron a través de un enlace con una velocidad de 2 Mbps full-duplex y un delay de 10 milisegundos. El tráfico entre el emisor y el receptor fue encaminado a través de varios saltos, de entre los routers de la red.

Se realizaron varias simulaciones conectando el servidor en tres lugares distintos de la nube de routers y el cliente siempre en el mismo lugar. La intención de tener el servidor conectado en los diferentes lugares es poder hacer la comparación de los flujos desde cada una de sus posiciones. Como se definió en el objetivo el lugar dónde está conectado el servidor es a lo que se le va a buscar un criterio de orden debido al desempeño de los flujos. A partir de ahora, nos vamos a referir a los tres servidores como diferentes, pero en realidad lo que cambia en cada uno de ellos es a qué nodo se conecta en la red, pero las características de los servidores son siempre las mismas, así como también el enlace que lo conecta a la red. Cada servidor entra en el core de la red por un nodo diferente en cada uno de los tres casos pero se conecta contra el mismo cliente.

Particularmente, se definieron dos velocidades de inyección de tráfico streaming. La in-

Sobre la misma topología de red	Audio (256 Kbps)	Video (1 Mbps)
Streaming	$OrigenS_1 \rightarrow DestinoC$	$OrigenS_1 \rightarrow DestinoC$
	$OrigenS_2 \rightarrow DestinoC$	$OrigenS_2 \rightarrow DestinoC$
	$OrigenS_3 \rightarrow DestinoC$	$OrigenS_3 \rightarrow DestinoC$

Cuadro 3.1: Estructura de las simulaciones

yección de tráfico se realizó usando un velocidad fija, lo que se conoces como Constant Bit Rate (CBR). Las velocidades de inyección fueron una de 256 Kbps y la otra de 1 Mbps. Se eligió tráfico de tipo CBR porque representa el patrón de tráfico característico del streaming. Las simulaciones se separaron en dos velocidades distintas para ver que diferencias se encuentran en el impacto que esto tiene en la red teniendo en cuenta que no se hace ningún tipo de control de congestión por parte del emisor en el flujo que está generando. Además, se eligieron estas velocidades porque son típicas en la transmisión de audio (música) para el caso de 256 Kbps y en la transmisión de video (películas) para el caso de 1 Mbps. En el cuadro 3.1 se realiza un esquema de la estructura de las simulaciones.

Cada una de las seis simulaciones se ejecutó durante un período de tiempo de 5 minutos durante el cual se generó el tráfico con características autosimilares y simultáneamente se cursó el tráfico de streaming del que se guardó la traza para su posterior análisis. Cada una de estas simulaciones se repitió 100 veces (más adelante se verá por qué se repitió ese número de veces), sumando un total de 600 simulaciones. Cien de cada tipo, por tres orígenes distintos, por dos velocidades de inyección de streaming diferentes son 100 x 3 x 2 = 600. Para poder entender mejor la estructura y variantes de las simulaciones a continuación hay

un algoritmo con el pseudo código de lo que se realizó:

```
N=100
rate=256Kbps
FOR (i in N) DO
    FOR EACH (server = S1, S2, S3) DO
        simular(rate, server);
        procesar_resultado();
    END FOR EACH
END FOR
rate=1Mbps
FOR (i in N) DO
    FOR EACH (server = S1, S2, S3) DO
        simular(rate, server);
        procesar_resultado();
    END FOR EACH
END FOR
```

Cada resultado de la simulación arroja dos resultados, el throughput y el packet loss, que se calculan de acuerdo a las fórmulas 2.1 y 2.2 respectivamente. La ejecución de cada simulación en sí, es un proceso estocástico (no determinístico), que arroja diferentes resultados.

Como resultado de todo el proceso de simulaciones se obtienen seis listados, dependientes de la velocidad (256 Kbps o 1 Mbps) y el servidor (S_1, S_2 o S_3). Cada listado consta de 100 valores de throughput y otros 100 de packet loss, como se puede observar en el ejemplo del cuadro 3.2.

Throughput (Kbps)	Packet Loss (%)
198	3,9
110	4,5
.	.
.	.
.	.
153	1,7

Cuadro 3.2: Ejemplo de resultados de las simulaciones

3.2.2. Tamaño de las muestras

La simulación de cada una de las seis pruebas se repitió un número de veces. El objetivo de cada prueba es el de establecer cuál es el throughput y el packet loss de cada comunicación entre el cliente y el servidor. Dicha comunicación es un proceso estocástico por lo que se hace imposible saber el verdadero valor del throughput y del packet loss. A través de métodos estadísticos de estimación puntual es posible intentar determinar cuales son los valores de estos dos parámetros, es decir, de dar un único valor, aproximado, para estos parámetros desconocidos. En lugar de dar un valor único y tratar de ver que error comenten los estimadores con respecto a los parámetros reales, se van a utilizar intervalos de confianza[2].

Tamaño de la muestra para el throughput

Para el caso del intervalo de confianza de la media μ del throughput, se analizará la distribución de la variable $\bar{X}$ (promedio aritmético) que es su estadístico natural.

Así entonces, es posible calcular los límites de confianza para μ, mediante la siguiente expresión:

[2]Los cálculos son en base a las simulaciones generadas para el streaming de 256 Kbps usando uno de los tres servidores con 100 ejecuciones o muestras.

$$P(\bar{X} - z_{1-\alpha/2} \cdot \sigma/\sqrt{n} < \mu < \bar{X} + z_{1-\alpha/2} \cdot \sigma/\sqrt{n}) = 1 - \alpha \tag{3.7}$$

pero para esto necesitamos conocer el desvío estándar (σ) de la población, lo que en principio parecería una limitación, ya que si no se conoce el valor de μ, menos aún se va a conocer el del desvío estándar. Sin embargo esto se resuelve reemplazándolo por su estimador.

A partir del intervalo de confianza se pude resolver un problema que aparece con frecuencia que es el de calcular el tamaño de la muestra para obtener un error ϵ estipulado, con un nivel de confianza 1–α. A partir de la semiamplitud del intervalo de confianza se obtiene:

$$\epsilon = z_{1-\alpha/2} \cdot \sigma/\sqrt{n} \tag{3.8}$$

por lo que n se puede calcular así:

$$n = (z_{1-\alpha/2} \cdot \sigma/\epsilon)^2 \tag{3.9}$$

valor que se recomienda redondear al entero superior. Se evidencia de la fórmula que cuanto más pequeño sea el valor del error ϵ, mayor debe ser el tamaño de la muestra a tomar.

El error ϵ suele especificarse como una fracción o porcentaje de la estimación $\bar{X}$, es decir $\epsilon = p \cdot \bar{X}$, por lo que se puede calcular n de la siguiente forma, teniendo en cuenta que se requiere de una muestra inicial grande desde la cual se van a estimar la media y el desvío estándar:

$$n = (z_{1-\alpha/2} \cdot \sigma/p \cdot \bar{X})^2 \tag{3.10}$$

donde

- $z_{1-\alpha/2}$ es el valor que tiene la distribución Normal Estándar en el punto $1 - \alpha/2$

- p es un valor de la proporción respecto del valor esperado
- σ se reemplaza por su estimador $\hat{S}$

Hay que tener en cuenta que $(p \cdot \mu)$ es un valor en términos de lo que se está midiendo y no una proporción o probabilidad. Es decir, que si se está midiendo throughput debería ser un througput pequeño. Por ejemplo, si el valor esperado es 1000 Kbps y p es 0,01 entonces tenemos el valor 10.

Procediendo con el cálculo suponiendo un nivel de riesgo, $\alpha = 0,05$ y, una proporción, $p = 0,01$ (el 1 %) se obtiene[3]:

$n = (z_{1-\alpha/2}\hat{S}/p\bar{X})^2$

$n = (z_{0,975}\hat{S}/0,01\bar{X})^2$

$n = (1,96 \cdot 3,837774/0,01 \cdot 107,1302)^2$

$n = (7,52203704/0,01 \cdot 107,1302)^2$

$n \approx 49,21$

En consecuencia, si el tamaño de la muestra es 50, se puede tener un 95 % de confianza en que μ difiere en menos de 1,07 Kbps de $\bar{X}$ [15] [16].

Tamaño de la muestra para el packet loss

En el caso del packet loss lo que hay que estimar es la proporción de éxitos, es decir, de los paquetes que no llegan a destino (lo que también podrían considerarse fracasos en lugar de éxitos), y para ello hay que calcular la probabilidad de ocurrencias, p, de eventos exitosos en una serie de n ensayos de Bernoulli. Sea k el número de eventos exitosos, entonces el estimador de p es:

$$\hat{p} = k/n \tag{3.11}$$

Ahora se necesita construir el intervalo de confianza para p, teniendo en cuenta que la

[3]Se usó el software libre R-project para el cálculo de $\hat{S}$ y $\bar{X}$

distribución del número de éxitos de una muestra es binomial. Ahora, es posible construir un intervalo que contenga a p con una probabilidad de $1 - \alpha$.

Para una muestra grande, se puede usar una aproximación normal a la distribución binomial. Entonces la distribución de la muestra de $\hat{p}$ se puede modelar como $N_{\mu,\sigma}$ siendo:

$\mu = p$

$\sigma = \sqrt{p \cdot q/n} \quad siendo \quad q = 1–p$

Por lo que el intervalo de confianza que es:

$$P(\hat{p} - z_{1-\alpha/2} \cdot \sigma < \mu < \hat{p} + z_{1-\alpha/2} \cdot \sigma) = 1 - \alpha \tag{3.12}$$

luego queda:

$$P(\hat{p} - z_{1-\alpha/2} \cdot \sqrt{p \cdot q/n} < \mu < \hat{p} + z_{1-\alpha/2} \cdot \sqrt{p \cdot q/n}) = 1 - \alpha \tag{3.13}$$

A partir de la semiamplitud del intervalo de confianza se obtiene que:

$$\epsilon = z_{1-\alpha/2} \cdot \sqrt{p \cdot q}/\sqrt{n} \tag{3.14}$$

por lo que n se puede calcular así:

$$n = (z_{1-\alpha/2} \cdot \sqrt{p \cdot q}/\epsilon)^2 \tag{3.15}$$

Procediendo con el cálculo suponiendo un nivel de riesgo, $\alpha = 0,05$ y, una proporción, $\epsilon = 0,07$ (el 7 %) se obtiene[4]:

$n = (z_{1-\alpha/2}\sqrt{p \cdot q}/\epsilon)^2$

$n = (z_{0,975}\sqrt{p \cdot q}/\epsilon)^2$

$n = (1,96 \cdot \sqrt{0,12817 \cdot 0,87183}/0,07)^2$

$n \approx 87,6$

En consecuencia, si el tamaño de la muestra es 88, se puede tener un 95 % de confianza en que p difiere en menos del 7 % del valor de $\hat{p}$.

[4]Se usó el software libre R-project para el cálculo de p y q

Como ya se mencionó en el punto 3.2.1 se hicieron simulaciones como para obtener 100 muestras y se mostró que esto es suficiente para tener un determinado nivel de confianza.

3.3. Duración de las simulaciones

Para poder obtener una estimación del estado estacionario del throughput y del packet loss, la simulación tiene que ser lo suficientemente larga. En este estudio el tiempo de simulación es de 5 minutos, un tiempo más que razonable en términos de red para tener un pantallazo del sistema funcionando a pleno, y poder así, hacer cálculos precisos de los distintos parámetros.

En términos prácticos, 5 minutos de prueba con cada uno de los servidores candidatos a servir el streaming es mucho tiempo, más, si se trata de una aplicación interactiva donde hay un usuario a la espera de la selección. Por este motivo, se realizó un segundo conjunto de simulaciones de menor duración individual para evaluar si los resultados obtenidos son equiparables con los que se obtuvieron de las simulaciones más largas. Se realizaron las 600 simulaciones tal como se explicó en 3.2.1 con la única diferencia de que cada simulación individual duró 30 segundos en lugar de 5 minutos. Se redujo el tiempo de la simulación a un 10 % (de 300 segundos - 5 minutos - a tan solo 30 segundos), lo que implica que se pasó de transmitir 37,5 MBytes o 3,75 MBytes.

Posiblemente 30 segundos para una aplicación real interactiva todavía sea un tiempo demasiado grande para tener a un usuario a la espera de que comience la reproducción de una audio o video vía streaming, además, del tiempo necesario para el buffering inicial y el tiempo propio de la ejecución de la aplicación. Por otro lado, no es objetivo de este trabajo determinar cuál es el tiempo mínimo de muestra que se debería utilizar para poder tomar una decisión correcta respecto del servidor a seleccionar.

3.4. Procesamiento de la traza

Como se mostró en el punto 3.2.1 después de la ejecución de cada simulación se procesa el resultado con el objetivo de calcular el throughput y el packet loss de la comunicación. El simulador ns2 escribe en un archivo de texto la traza de la simulación llevada a cabo. La generación de dicha traza es programable a través de los scripts con los que se programa la simulación en sí. Existen varios formatos de traza posible, que surgieron a partir de la incorporación al simulador de las comunicaciones wireless. En este trabajo se utilizó el primero de los formatos establecidos para la traza que registra la información de conexiones cableadas. Este formato genera una línea de texto por cada evento que le ocurre a cada paquete de datos. Esto quiere decir que, por cada paquete de cada flujo hay muchas de estas líneas describiendo que es lo que va ocurriendo con el paquete entre el origen y el destino final. Los datos que se proveen en cada línea de la traza son 12 y se muestran en el cuadro 3.3 con una breve explicación de cada uno de ellos.

Parámetro	**Descripción**
1. Event	Puede tomar los siguientes valores: ▪ "+": evento de encolamiento de un paquete ▪ "-": evento de desencolamiento de un paquete ▪ "r": evento de recepción de un paquete ▪ "d": evento de descarte de un paquete
2. Time	El instante de tiempo (en segundos) en que la traza fue creada (es como el timestamp del evento)
3-4. Source Node - Destination Node	Denota el identificador del nodo de origen y del nodo de destino de la trama generada de manera tal que se pueda identificar el enlace (capa 2) en el que ocurre el evento
5. Packet Type	Es el tipo de paquete del que se trata (Por ejemplo, CBR)
6. Packet Size	Es el tamaño del paquete (en bytes)

Continúa en la próxima página

Cuadro 3.3 – *Continúa de la página anterior*

Parámetro	**Descripción**
7. Flags	Es un string de 7 flags con los siguientes significados: ▪ "-": deshabilitado en todos los casos ▪ 1ro = "E": ECN (Explicit Congestion Notification) ▪ 2do = "P": la prioridad del encabezado IP está habilitada ▪ 3ro: No se usa ▪ 4to = "A": Acción de congestión ▪ 5to = "E": Ha ocurrido congestión ▪ 6to = "F": La fase de TCP "fast start" está en uso ▪ 7to = "N": Explicit Congestion Notification (ECN) está prendido
8. Flow ID	Es un identificador único para el flujo
9-10. Source Address - Destination Address	Es la dirección de red (capa 3) de origen y de destino que tiene el siguiente formato: son dos campos "a.b", donde "a" es la dirección "b" es el puerto
11. Sequence Number	Es el número de secuencia del paquete (capa 4). Aunque UDP no tiene número de secuencia en su encabezado, este es agregado igual con el propósito de análisis

Continúa en la próxima página

Cuadro 3.3 – *Continúa de la página anterior*

Parámetro	**Descripción**
12. Packet Unique ID	Es un identificador único del paquete

Cuadro 3.3: Formato de la traza

Una vez realizada la simulación y obtenido el archivo de traza recién descripto se procedió a procesarlo. El procesamiento consta de convertir toda la información del archivo de traza en datos comprensibles y necesarios para el análisis del trabajo. A partir de la programación de un conjunto de scripts (en shell script y awk) se obtuvo el throughput y el packet loss de los streamings de cada simulación. En líneas generales los scripts realizaron el parsing línea por línea de la traza y fueron acumulado los datos necesarios para el cálculo. El archivo de traza es un archivo muy grande (aproximadamente 4Gbytes para los cinco minutos de tiempo de simulación) ya que se almacena toda la información referida a todos los eventos de cada uno de los paquetes de la simulación tanto del streaming generado para la medición como del resto del tráfico que se usa para completar el escenario. Se presenta a continuación a modo de pseudocódigo qué realizaron los scripts tanto para el throughput como para el packet loss. En ambos scripts lo primero que se hace es filtrar solo las líneas que tienen información acerca de los flujos de streaming usando el campo "Packet Type".

```
/* Pseudocódigo para el cálculo del Throughput */
recv = 0
FOR EACH (event, node_id, pkt_id IN line) DO
// Por cada evento de la traza que corresponda a un paquete de streaming
    IF (event == "r" && node_id == dst) THEN
    //Si el evento es de recepción en el nodo de destino final
        recv += pkt_size - hdr_size //Acumula la carga útil del paquete
    END IF
END FOR EACH
```

```
PRINT (recv / simulation_time)*(8/1000) //Imprime el resultado en Kbps

/* Pseudocódigo para el cálculo del Packet Loss */
//Inicializa dos vectores send y recv con todos sus elementos en cero
tx = 0
drop = 0
pkt_loss = 0
FOR EACH (event, node_id, pkt_id IN line) DO
//Por cada evento de la traza que corresponda a un paquete de streaming
    IF (event == "+" && node_id == src) THEN
    //Si el evento es de encolado en el nodo de origen
        send[pkt_id] = 1 // Marca el paquete como enviado
    END IF
    IF (event == "r" && node_id == dst) THEN
    //Si el evento es de recepción en el nodo de destino final
        recv[pkt_id] = 1 // Marca el paquete como recibido
    END IF
END FOR EACH
FOR EACH (i IN send) DO
    IF (send[i]) THEN
        tx++ //Acumula el número de enviados
        IF (!recv[i]) THEN
            drop++ //Acumula el número de descartados
        END IF
    END IF
END FOR EACH
PRINT (drop / tx) * 100 //Imprime el porcentaje de pérdida
```

Como resultado cada uno de los dos scritps lo que se obtiene es un valor escalar que

representa el throughput expresado en Kbps y el packet loss expresado en porcentaje de pérdida sobre el total de paquetes enviados. Cada uno de estos valores se almacena en un archivo como se vio en el punto 3.2.1 para su posterior análisis y procesamiento.

Capítulo 4

Resultados

4.1. Análisis estadístico

Uno de los objetivos planteados en el trabajo es el de establecer un mecanismo que permita ordenar los servidores de streaming a seleccionar de acuerdo a lo conveniente que sea elegirlos para efectuar la transmisión. Para poder establecer dicho orden se procesaron los datos generados por el simulador, de manera tal, de poder cuantificar los resultados y de poder realizar comparaciones entre ellos.

Una primera aproximación para entender que es lo que pasó en cada uno de los casos, fue el cálculo de los estadísticos más comunes (media, desvío estándar, mínimo, máximo, mediana y cuartiles). En la sección 4.2 se procedió al agrupamiento de los datos y a la generación de los histogramas en cada uno de los casos.

Se pueden observar los estadísticos del throughput para 256 Kbps y 1 Mbps en los cuadros 4.1 y 4.3 respectivamente y los del packet loss para 256 Kbps y 1 Mbps en 4.2 y 4.4 respectivamente. S_1, S_2 y S_3 en la primer columna de los cuadros refiere a los servidores 1, 2 y 3 respectivamente.

	Mínimo	1^{er} **cuartil**	**Mediana**	**Media**	3^{er} **cuartil**	**Máximo**	**Desvío est.**
S_1	195,7	207,3	212,6	213,1	218,4	228,7	7,71
S_2	224,1	233	235,6	235,3	237,6	241,3	3,31
S_3	229,5	237,8	239,3	239,1	240,9	243,6	2,38

Cuadro 4.1: Estadísticos comunes del throughput para 256 Kbps

	Mínimo	1^{er} **cuartil**	**Mediana**	**Media**	3^{er} **cuartil**	**Máximo**	**Desvío est.**
S_1	6,9	11,1	13,47	13,29	15,64	20,35	3,13
S_2	1,82	3,32	4,15	4,27	5,2	8,8	1,34
S_3	0,89	1,98	2,64	2,69	3,22	6,63	0,97

Cuadro 4.2: Estadísticos comunes del packet loss en porcentaje para 256 Kbps

	Mínimo	1^{er} **cuartil**	**Mediana**	**Media**	3^{er} **cuartil**	**Máximo**	**Desvío est.**
S_1	604	633,5	648,4	650,6	666,3	729,3	24,08
S_2	701,8	747,3	764,5	761,7	774,4	810,5	20,95
S_3	751,3	776	793,9	792,6	806,1	838,6	19,91

Cuadro 4.3: Estadísticos comunes del throughput para 1 Mbps

	Mínimo	1^{er} **cuartil**	**Mediana**	**Media**	3^{er} **cuartil**	**Máximo**	**Desvío est.**
S_1	24,03	30,6	32,45	32,23	34,01	37,08	2,5
S_2	15,58	19,33	20,36	20,65	22,16	26,9	2,18
S_3	12,65	16,03	17,3	17,44	19,17	21,74	2,07

Cuadro 4.4: Estadísticos comunes del packet loss en porcentaje para 1 Mbps

4.2. Agrupamiento de datos: histogramas

4.2.1. Throughput

Teniendo en cuenta las dos velocidades de transferencia probadas, la de 256 Kbps y la de 1 Mbps, para cada uno de los tres servidores podemos ver (como se esperaba) que el throughput alcanzado en cada caso varía. En general, el desempeño de las conexiones a 256 Kbps tienen un mejor rendimiento en relación el throughput teórico alcanzable que las de 1 Mbps y esto se debe a que la inyección de los datos se hace más rápido en el último caso provocando o colaborando a un nivel mayor de congestión. Para el caso de los 256 Kbps, el throughput alcanzado se mueve desde un mínimo d 195,7 Kbps hasta un el máximo de 243,6 Kbps lo que representa del bitrate o ancho de banda teórico entre el 76,44 % y el 95,15 %. En cambio, para el caso de 1 Mbps, el mínimo throughput alcanzado es de 604 Kbps mientras que el máximo throughput alcanzado es de 838,6 Kbps lo que representa del bitrate entre el 60,4 % y el 83,86 %. En ambos casos se puede observar que el canal de comunicación que vincula al cliente con el servidor está con un grado de congestión elevado, existiendo descarte de paquetes en uno o varios de los routers intermedios del camino. Estas manifestaciones de pérdida de paquetes son coincidentes con el análisis que se hará más adelante acerca del packet loss.

Bitrate de 256 Kbps

El caso de 256 Kbps se pude observar en el histograma de la figura 4.1, donde el eje de las x muestra *intervalos de 5 Kbps*, mientras que el eje de las y muestra las *frecuencias relativas* con las que esos bitrates aparecen. Allí se observa que el servidor 1 se mueve por valores más bajos, y por ende peores, que los servidores 2 y 3. El histograma del servidor 1 muestra algo que es inusual con la existencia de dos picos. La interpretación que se le puede atribuir al fenómeno de los dos picos es que esté describiendo dos procesos diferentes, o si se quiere el tráfico atraviesa dos momentos distintos en la red que parecen alternarse. Mientras que el servidor 1 varía entre los 195,7 a 228,7 Kbps, los servidores 2 y 3 se aproximan más,

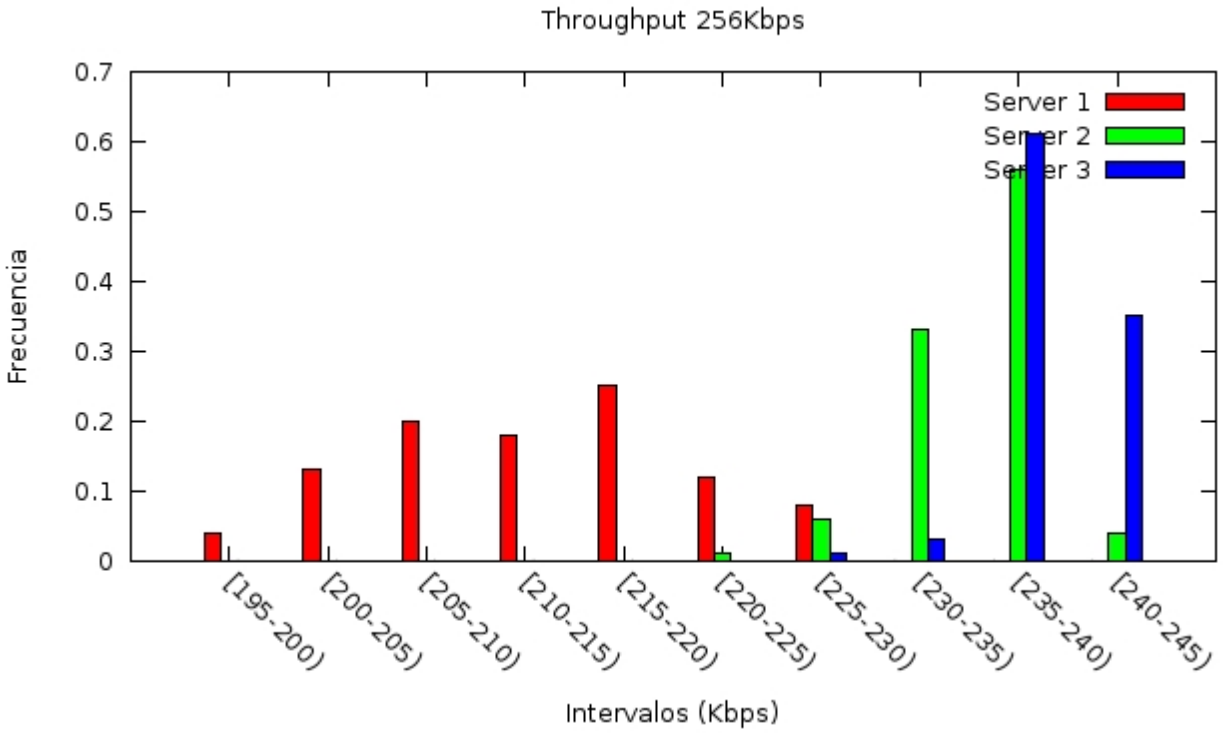


Figura 4.1: Histograma para el bitrate de 256 Kbps

variando entre 224,1 y 241,3 Kbps y 229,5 y 243,6 Kbps respectivamente. Los servidores 2 y 3 tienen un aspecto similar entre ellos, cosa que se nota en el análisis de las posibles distribuciones que se hace en el anexo A, ya que las distribuciones candidatas para las conexiones a los servidores 2 y 3 son las mismas. Particularmente, entre los servidores 2 y 3 se observa que el servidor 3 está un poco mejor que el 2 ya que se posiciona más a la derecha (throughputs más altos). Por otro lado el servidor 3 tiene una fuerte concentración de valores en el intervalo [235-245), mientras que el 2 tiene una fuerte concentración en el intervalo [230-240). Además, los valores del servidor 3 tienen menor dispersión que los demás, mostrando cierta estabilidad que los demás no tienen.

Bitrate de 1 Mbps

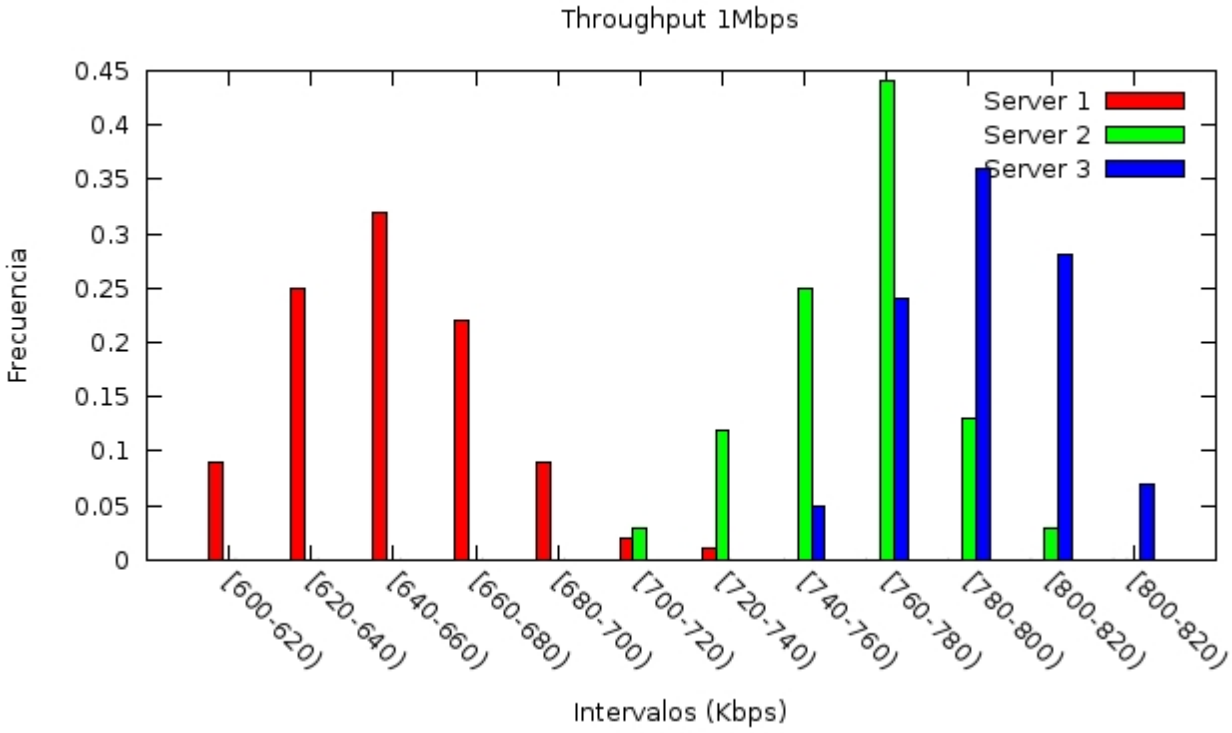


Figura 4.2: Histograma para el bitrate de 1 Mbps

Para el caso de 1 Mbps que se puede observar en la figura 4.2 que de manera similar a lo que paso para los 256 Kbps el servidor 1 está por debajo de los otros (a la izquierda del gráfico), mostrando un desempeño peor. Hay que recordar que lo que cambia entre las simulaciones de 256 Kbps y de 1 Mbps es la velocidad de inyección de los paquetes en la red pero que tanto el cliente como el servidor son los mismos. Por este motivo es esperable que aquellos servidores que se desempeñaban mejor en 256 Kbps lo sigan haciendo mejor con 1 Mbps ya que el camino a través de lo routers es el mismo que antes, entonces, los que atravesaban el camino más congestionado van a seguir siendo los más perjudicados. De manera análoga a lo que sucedió con 256 Kbps los servidores 2 y 3 se desempeñaron mejor que el 1, que varía entre 604 y 729,3 Kbps. A su vez, el 3 se comporta mejor que el 2. En el caso del 2 se concentró fuertemente el throughput entre 720 y 800 mientras que el 3 estuvo entre 760 y 820, además, de que el 3 está un poco más concentrado mostrando una menor dispersión.

4.2.2. Packet Loss

Bitrate de 256 Kbps

En la figura 4.3 se puede observar que en el eje de las x se muestra la proporción de pérdidas en *intervalos de 0,02*, mientras que el eje de las y se muestra las *frecuencias relativas* con las que esas pérdidas aparecen. Consistentemente con lo que paso en el throughput, los servidores 2 y 3 tuvieron menos pérdida que el 1. El servidor 3 se mantuvo siempre debajo del 7 %, mientras que el servidor 2 se mantuvo debajo del 9 %. Casi el 30 % del tráfico desde el servidor 3 no tuvo pérdida o tuvo muy poca (entre nada y un 2 %). En cambio el servidor 1 llegó a tener hasta un 20 % (o un poco más) de pérdida. Observando este histograma y el del throughput, se puede ver con claridad el vínculo que hay entre el packet loss y el throughput, ya que a menor packet loss mayor throughput. El servidor 3 que tiene el menor packet loss tiene el mejor throughput, le sigue el servidor 2 que tiene un poco más de packet loss y un peor throughput que el servidor 3, y por último, le sigue

el servidor 1 que tiene el packet loss más alto y el peor throughput.

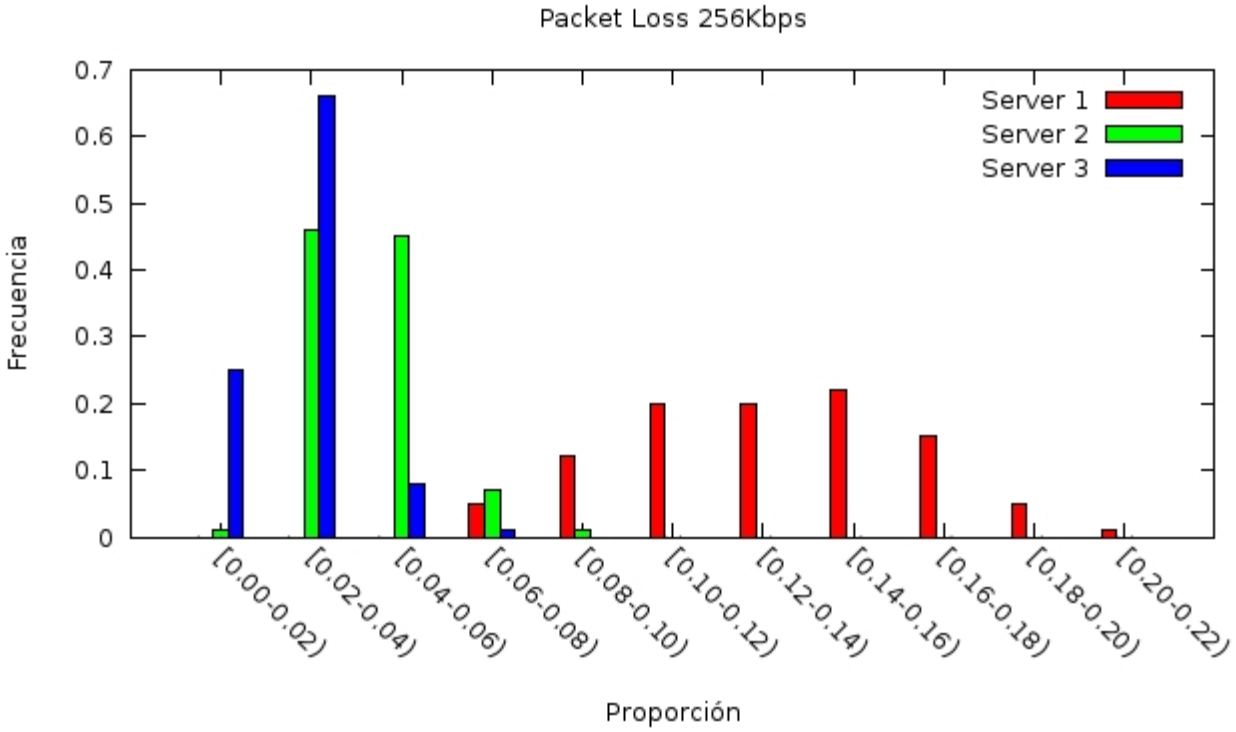


Figura 4.3: Histograma para la pérdida de 256 Kbps

Bitrate de 1 Mbps

El packet loss para el caso de 1 Mbps que se puede observar en la figura 4.4. Como era esperable es peor que para el caso de 256 Kbps. Consistentemente con lo que paso en el throughput, los servidores 2 y 3 tuvieron menos pérdida que el 1. El servidor 3 está entre el 12,65 % y el 21,74 %, mientras que el 2 está entre el 15,58 % y el 26,9 %. Por otro lado el servidor 3 se concentró fuertemente entre 12 % y 22 % y el servidor 2 se concentró fuertemente entre el 14 % y el 28 %. El servidor 1 que tiene el peor desempeño alcanza pérdidas de hasta el 38 % y nunca por debajo de 24 %. Se puede repetir el análisis que vincula el mayor packet loss con el menor throughput al igual que en el caso de 256

Kbps.

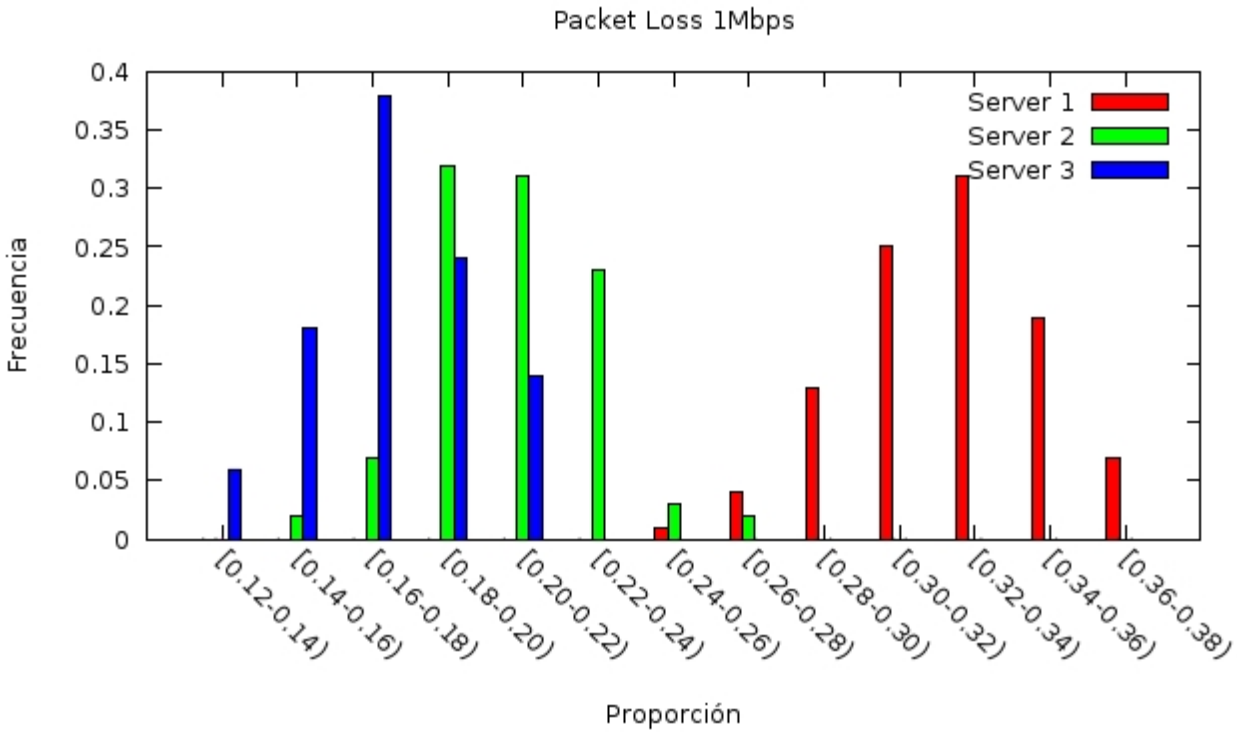


Figura 4.4: Histograma para la pérdida de 1 Mbps

4.3. Relación entre las simulaciones largas y cortas

Es de interés poder determinar si los resultados de la simulaciones largas y cortas son equivalentes en términos estadísticos. Este procedimiento permite obtener una corroboración de que los resultados obtenidos con las simulaciones más prolongadas, son aún válidas para simulaciones de menor duración que tienen más aplicabilidad en una situación práctica.

En esta situación se trata de ver que ocurre entre dos grupos. Se va a trabajar con la hipótesis de que existen diferencias entre ambos grupos. Para verificar estas diferencias se va a hacer un test de diferencias de medias poblacionales conocido como *z-test* [17]. Para poder

hacer uso de este test se necesitan dos muestras de dos poblaciones, cuyos tamaños pueden ser diferentes. Las dos muestras tienen que ser independientes. Ambas poblaciones tienen que tener distribución normal o el tamaño de las muestras tiene que ser lo suficientemente grande como para poder asumir la distribución normal (usualmente las muestras tienen que tener un tamaño mayor a 30 elementos). Por último, el desvío estándar de las poblaciones tiene que ser conocido.

Las muestras van a compararse según el test, tomando los resultados del throughput para el servidor 1, el servidor 2 y el servidor 3, para las dos velocidades (256Kbps y 1Mbps), y análogamente para el packet loss. Resultando de esta forma 12 tests, 6 para el throughput y 6 para el packet loss, donde 3 van a ser para una velocidad y los otros 3 para la otra con cada uno de los servidores. Por ejemplo, se van comparar las muestras de los throughputs para la simulación con duración larga y con duración corta de 256Kbps y servidor 1.

Con el test lo que se está tratando de afirmar es si a partir de las dos muestras hay evidencia suficiente para afirmar que las medias son los suficientemente distintas para un nivel de significación dado (hipótesis alternativa), o por el contrario, la evidencia no es suficiente para afirmar que son distintas por lo que se asumen iguales (hipótesis nula).

Formalmente, se tiene:

- $H_0 : \mu_1 - \mu_2 = 0$, o alternativamente, $\mu_1 = \mu_2$ (hipótesis nula)
- $H_1 : \mu_1 - \mu_2 \neq 0$, o alternativamente, $\mu_1 \neq \mu_2$ (hipótesis alternativa)

Para calcular el test estadístico se hace uso de la siguiente fórmula:

$$\hat{z} = \frac{(\hat{\mu}_1 - \hat{\mu}_2) - D}{\sqrt{(\frac{\sigma_1^2}{m} + \frac{\sigma_2^2}{n})}} \tag{4.1}$$

donde,

- $\hat{z}$, es el estadístico a calcular
- $\hat{\mu}_1$, es la media de la primera muestra
- $\hat{\mu}_2$, es la media de la segunda muestra
- σ_1, es el desvío estándar de la primera muestra
- σ_2, es el desvío estándar de la segunda muestra
- D, es la diferencia de medias
- m, es el tamaño de la primera muestra
- n, es el tamaño de la segunda muestra

Particularmente, en este caso ambas muestras (m y n) son de 100 elementos y se busca que las medias sean iguales por lo que D es cero. La fórmula resultante es:

$$\hat{z} = \frac{\hat{\mu}_1 - \hat{\mu}_2}{\sqrt{(\frac{\sigma_1^2}{100} + \frac{\sigma_2^2}{100})}} \tag{4.2}$$

Se pueden observar los valores de $\hat{z}$ para cada uno de los 12 casos en la tabla 4.5.

El siguiente paso es encontrar el p-valor que es el menor nivel de significación según el cual se rechazaría la hipótesis nula (o el mayor valor según el cual no se rechazaría).

$$p - valor = P(Z \leq -|\hat{z}|) + P(Z \geq |\hat{z}|) = 2P(Z \geq |\hat{z}|) = 2(1 - P(Z \leq |\hat{z}|)) \tag{4.3}$$

Se pueden observar los valores de $p - valor$ para cada uno de los 12 casos en la tabla 4.6.

Una vez conocido el p-valor, este se compara con el nivel de significación (α). Si el p-valor es menor que α entonces la diferencia de medias observada permite rechazar hipótesis

Elementos de comparación	$\hat{z}$
Throughput, 256 Kbps, Servidor 1	0,6485798
Throughput, 256 Kbps, Servidor 2	1,519934
Throughput, 256 Kbps, Servidor 3	-0,341513
Throughput, 1 Mbps, Servidor 1	-1,355578
Throughput, 1 Mbps, Servidor 2	1,753412
Throughput, 1 Mbps, Servidor 3	0,567448
Packet Loss, 256 Kbps, Servidor 1	-0,6486063
Packet Loss, 256 Kbps, Servidor 2	-1,519836
Packet Loss, 256 Kbps, Servidor 3	0,341462
Packet Loss, 1 Mbps, Servidor 1	1,398203
Packet Loss, 1 Mbps, Servidor 2	-1,691942
Packet Loss, 1 Mbps, Servidor 3	-0,5034686

Cuadro 4.5: Valores del estadístico z para el *z-test*

Elementos de comparación	**p-valor**
Throughput, 256 Kbps, Servidor 1	0,5166
Throughput, 256 Kbps, Servidor 2	0,1285
Throughput, 256 Kbps, Servidor 3	0,7327
Throughput, 1 Mbps, Servidor 1	0,1752
Throughput, 1 Mbps, Servidor 2	0,0795
Throughput, 1 Mbps, Servidor 3	0,5704
Packet Loss, 256 Kbps, Servidor 1	0,5165
Packet Loss, 256 Kbps, Servidor 2	0,1285
Packet Loss, 256 Kbps, Servidor 3	0,7327
Packet Loss, 1 Mbps, Servidor 1	0,1620
Packet Loss, 1 Mbps, Servidor 2	0,0906
Packet Loss, 1 Mbps, Servidor 3	0,6146

Cuadro 4.6: Valores obtenidos para el *p-valor*

nula y por lo tanto, se rechaza la hipótesis nula en favor de la hipótesis alternativa. En cambio, si el p-valor es mayor que el nivel de significación (α) entonces la evidencia no permite rechazar la hipótesis nula.

Resumiendo:

1. $p - valor < \alpha$ se rechaza la H_0
2. $p - valor > \alpha$ no se rechaza la H_0

Un valor razonable para el nivel de significación (α), que habitualmente se usa es $\alpha =$ 0,05. Todos los p-valores calculados son mayores a α, lo que ubica el resultado del z-test en la segunda opción donde no hay evidencia contundente como para rechazar la hipótesis nula, por lo que ésta es aceptada. Aceptando la hipótesis nula, se considera que las medias, tanto de las simulaciones más largas como las de las más cortas son iguales, con lo que se puede afirmar que realizar el conjunto de simulaciones más cortas es suficiente para realizar el análisis propuesto.

4.4. Métrica elegida

El *Coeficiente de Variación del Throughput* (CoTV - Coefficient of Throughput Variation) puede ser utilizado como métrica de referencia reducida gracias a sus características. Dicho CoTV es relativamente fácil de calcular, comparar e interpretar, a la vez que es útil como métrica para interiorizase de cambios en el desempeño de la red. Por otra parte utiliza el throughput como entrada para su cálculo y éste tiene una dependencia directa con el packet loss, ya que el throughput se degrada a medida que aumenta el packet loss.

Para reforzar el motivo de la elección de esta métrica, cómo base para el ordenamiento propuesto, se mencionan una publicación y una RFC que hacen uso de la misma.

La variación de delay juega un rol importante en la degradación del desempeño de una red y afecta la percepción de calidad del usuario, especialmente en casos de servicios como el streaming, VoIP, etc. Métodos livianos para la detección de problemas de desempeño son

deseables en lugar de tareas de medición intensiva y su análisis. El trabajo [18] utiliza el CoTV como un indicador para detectar problemas de desempeño en la red, y de esta manera poder rápidamente y de manera confiable detectar cuellos de botella en el comportamiento entre dos puntos arbitrarios de la red. El trabajo muestra la relación entre la variación del delay y el CoTV, mediante una emulación de red que inyecta datagramas UDP de tamaño constante, que atraviesan un *shaper* que la da la variación al tráfico. Finalmente, concluye que el CoTV puede ser utilizada para detectar y cuantificar problemas de desempeño, y en particular, se interesa en métodos prácticos que le permitan encontrar cuellos de botella usando técnicas livianas, que involucren medición y análisis, como es el CoTV.

La RFC *Metrics for the Evaluation of Congestion Control Mechanisms* [19] plantea las métricas a ser consideradas en la evaluación de los mecanismos de control de congestión con el objetivo de alcanzar la estabilidad de las conexiones en Internet. La métrica buscada intenta minimizar las oscilaciones del delay por encolamiento o del throughput de la conexión. Para lograr esto se analiza una métrica relacionada con la estabilidad que es frecuentemente asociada con la velocidad de las fluctuaciones o varianza. Cambios en la velocidad de las variaciones puede resultar en fluctuaciones en el tamaño de las colas de los routers y por consiguiente en el desbordamiento de esas colas. El desbordamiento de las colas puede causar pérdida de sincronización entre los flujos coexistentes y subutilización transitoria de la capacidad de los enlaces, lo que puede considerarse como signos de inestabilidad de la red. Las variaciones en los flujos es habitualmente la métrica usada para monitorizar la estabilidad de los protocolos de transporte. Para medir las variaciones de velocidad un conjunto de autores mencionados en la RFC utilizan el Coeficiente de Variación del Throughput. Como las variaciones de velocidad son función de las escalas de tiempo, tiene sentido medir dichas variaciones de velocidad sobre distintas escalas de tiempo.

Dicho coeficiente se calcula de la siguiente manera:

$$CoTV = \frac{\sigma}{|\mu|} \tag{4.4}$$

4.5. Conclusiones

Se han cumplido los objetivos específicos de la tesis:

- Establecer una taxonomía de los servicios que se pueden brindar en redes IP.
- Analizar diferentes formas de modelizar redes de datos, enfocando redes IP en cuanto a topología, velocidad de enlaces, agregación de tráfico, congestión, etc.
- Diagramar y ejecutar simulaciones sobre este modelo para cuantificar los parámetros significativos de la comunicación.
- Establecer un criterio que permita ordenar los nodos que ofrecen el servicio considerando los resultados de las simulaciones.

El aporte más significativo de la tesis ha sido desarrollar la métrica CoTV para establecer un criterio que permita ordenar los nodos que ofrecen el servicio deseado.

La métrica CoTV ha sido analizada a la luz de los resultados obtenidos en las simulaciones, concluyendo que *a medida que el valor del CoTV se reduce mejora la estabilidad de la conexión.*

Teniendo en cuenta los factores mencionados, se establecería el orden de preferencia para el caso de 256 Kbps de la siguiente forma:

- 1^{ro} : Servidor 3
- 2^{do} : Servidor 2
- 3^{ro} : Servidor 1

Esto es debido a que el CoTV para el Servidor 3 es menor que para el Servidor 2, y a su vez, el del Servidor 2 es menor que el del Servidor 1 ($0,0099 < 0,014 < 0,0361$). Se puede observar gráficamente la diferencia en el desempeño mirando el histograma del throughput 4.1 y el histograma del packet loss 4.3.

En el caso de 1 Mbps, sucede algo similar a lo que pasa con 256 Kbps, situación que resulta coherente ya que los enlaces (y por ende los routers intermedios) entre el cliente y los servidores son los mismos que para el caso de 256 Kbps pero ahora con otra velocidad de datos. El orden establecido es entonces igual al del caso anterior:

- 1^{ro} : Servidor 3
- 2^{do} : Servidor 2
- 3^{ro} : Servidor 1

Análogamente al caso anterior, el CoTV para el Servidor 3 es menor que para el Servidor 2, y a su vez, el del Servidor 2 es menor que el del Servidor 1 ($0,0251 < 0,0275 < 0,037$). Se puede observar gráficamente la diferencia en el desempeño mirando el histograma del throughput 4.2 y el histograma del packet loss 4.4.

Como conclusión, independientemente de la velocidad de inyección de los datos en la red, el orden de conveniencia de los servidores será inverso al de su valor de CoTV (en nuestro caso particular Servidor 3 – Servidor 2 – Servidor 1).

4.6. Líneas de investigación futuras

Prácticamente todos los temas tanto teóricos cómo prácticos abordados en este trabajo pueden seguir desarrollándose. Hay dos líneas de investigación que se desprenden inmediatamente del tema planteado. El primero de ellos tiene que ver con la incorporación al análisis de otros protocolos o tipos de tráficos, aparte del streaming, especialmente aquellos que son de tiempo real (telefonía IP, videollamadas, multiconferencias, etc.) y que ponen en juego otros parámetros de la red o de los enlaces como el delay y el jitter. También es interesante analizar el impacto de los codecs que se utilizan en estos protocolos de tiempo real. ¿El aumento del ancho de banda sin calidad de servicio, es suficiente? ¿Es la red best-effort suficiente o adecuada para prestar estos servicios?

El segundo tema que podría desprenderse tiene que ver con la posibilidad de componer el streaming a partir de múltiples fuentes. Es decir, en este trabajo se eligió uno entre varios servidores, pero que pasaría si el mismo recurso, que está en varios servidores, es solicitado por el cliente para ser transferido desde más de uno con el objetivo de hacer uso de varios servidores simultáneamente. ¿Podría ser esto útil? ¿Cómo se podría volver a ensamblar el stream para su correcta reproducción en el cliente? ¿Existiría un verdadero beneficio que justifique esta modalidad?

Apéndice A

Análisis estadístico

A.1. Análisis estadístico de los resultados

A partir de los valores obtenidos de throughput y de packet loss en cada uno de los casos se realizó un procedimiento con el objetivo de poder determinar a qué distribución de probabilidad mejor se ajusta la muestra generada por el simulador. Dicho procedimiento consta de dos etapas. En la primer etapa, utiliza algunos mecanismos para elegir un grupo pequeño de distribuciones candidatas (habitualmente entre dos y cuatro). Luego la selección final se lleva a cabo en una segunda etapa que valida esa distribución o modelo elegido.

Para la primer etapa se evalúa cuál es el ajuste de las diferentes distribuciones a una muestra y para eso se utilizan herramientas estadísticas entre las que están el "ajuste por momentos", la "estimación por máxima verosimilitud" y la "estimación por ajuste de fractiles". El procedimiento consiste en hallar para cada uno de los métodos una manera de ordenar los modelos de acuerdo con su capacidad de ajuste a la muestra. Para los dos últimos es posible definir ordenadores, que son valores estadísticos escalares de la medida de ajuste alcanzada. Para el caso de los momentos, no existe una estadística tal, así que no es posible utilizarlo para realizar la elección.

La selección se lleva a cabo de la siguiente manera:

1. Estimando por máxima verosimilitud los parámetros correspondientes a las distintas distribuciones y utilizando como criterio de ordenamiento los valores de la verosimilitud máxima obtenida para cada modelo.

2. Por medio del ajuste lineal y no lineal de fractiles a la muestra para distintas dis-

tribuciones, utilizando como ordenadores a estadísticas basadas en la suma de los cuadrados de las diferencias entre la función de distribución teórica y la muestral.

3. Utilizando los llamados "diagramas de momentos", que permiten la selección de candidatos pero no su ordenamiento. Estos sirven excepcionalmente como herramienta de validación.

Dado los inconvenientes derivados del uso del método de momentos, y dado que el ajuste no lineal de fractiles requiere del uso de técnicas matemáticas complejas (ajuste no lineal de funciones), los mecanismos cuya utilización resultan más simples son: el por máxima verosimilitud y el ajuste lineal de fractiles.

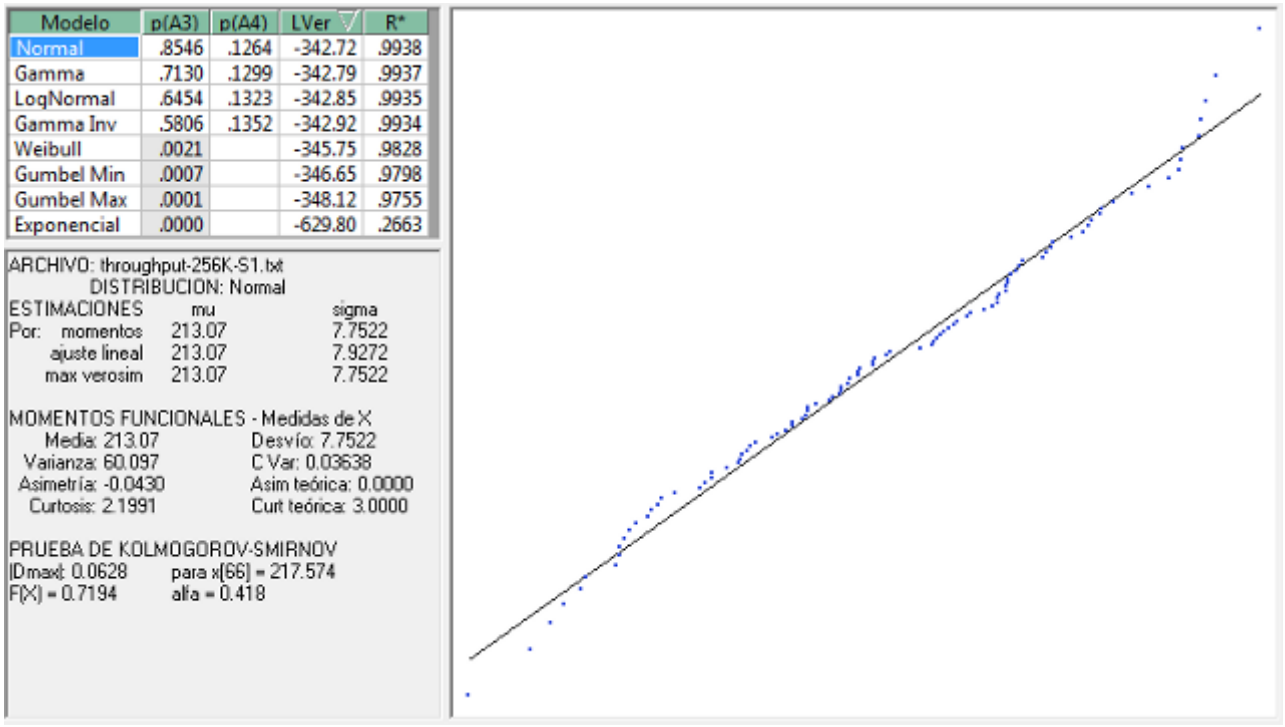
Para poder realizar los cálculos a partir de las muestras obtenidas se utilizó el software MOVAC (Modelos para Variables Aleatorias Continuas). El mismo incluye las etapas de ordenamiento de modelos lo que permite seleccionar algunos candidatos, y luego la validación que pone en juego la calidad de los mismos. Esta segunda etapa da lugar a la exclusión de algunos de los modelos quedando habitualmente unos o dos.

El software MOVAC pude trabajar en dos modalidades, una con datos individuales y otra con datos agrupados. Para el caso de datos individuales, que es el que se utiliza tanto para el throughput como para el packet loss, el software permite ordenar los modelos siguiendo el criterio de valores decrecientes de las verosimilitudes máximas.

Para cada uno de los modelos posibles el software calcula los estimadores de los parámetros de cada una de las distribuciones [15].

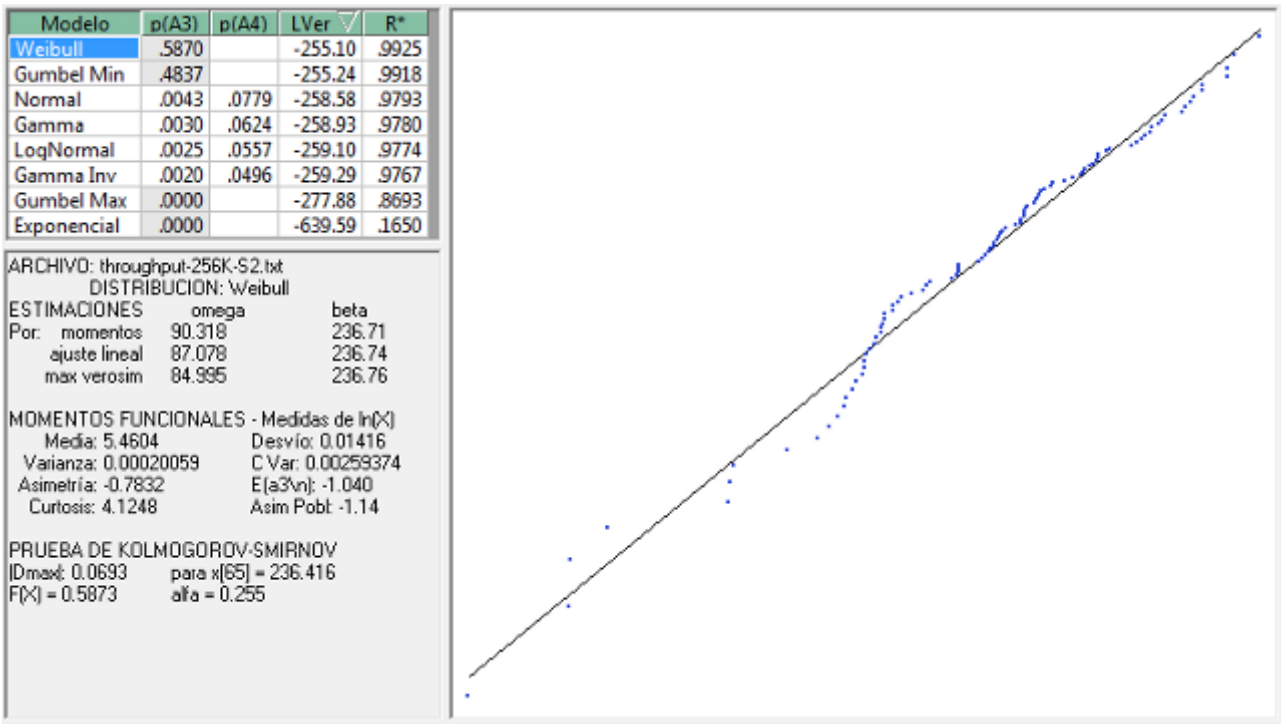
A.1.1. Inferencias estadísticas del throughput

Utilizando MOVAC se arma la tabla, para las distintas muestras del throughput, y se ordena por las verosimilitudes máximas de cada uno de los modelos en forma decreciente. Esto se puede observar para los casos de 256 Kbps en las figuras A.1 para el servidor 1, A.2 para el servidor 2 y A.3 para el servidor 3 y para los casos de 1 Mbps en A.4 para el servidor 1, A.5 para el servidor 2 y A.6 para el servidor 3.



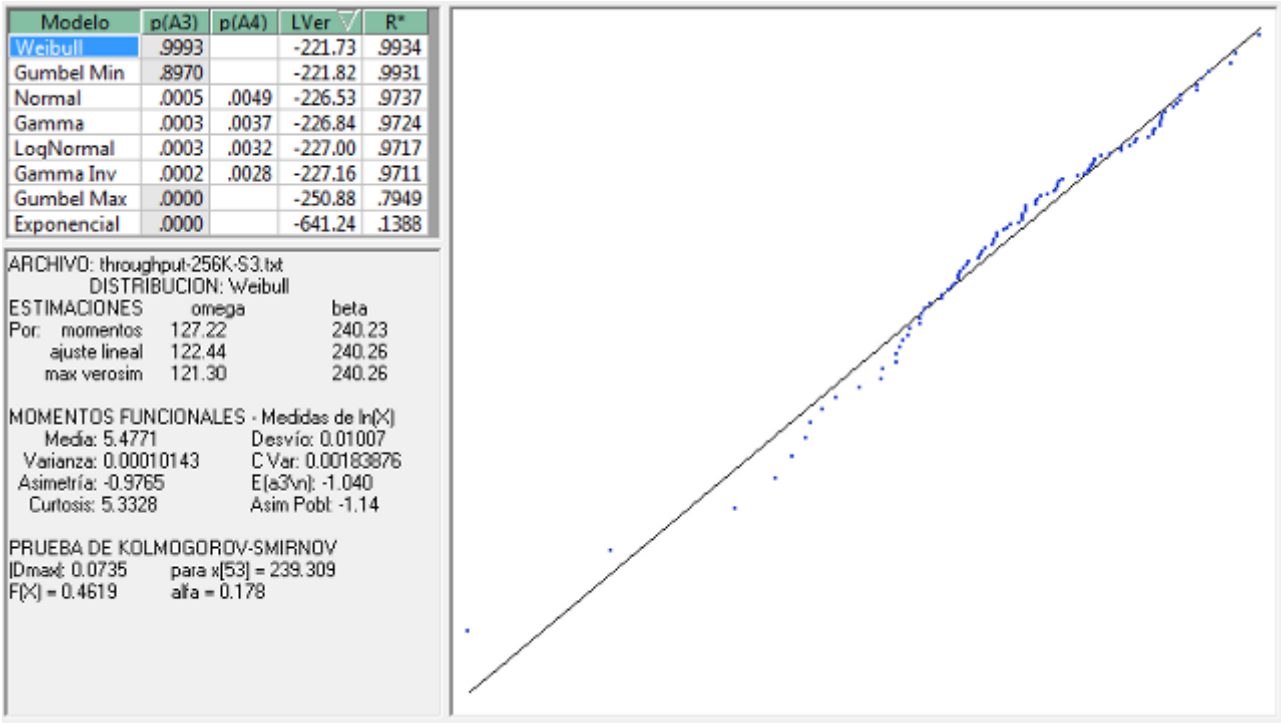
Modelo	p(A3)	p(A4)	LVer	R*
Normal	.8546	.1264	-342.72	.9938
Gamma	.7130	.1299	-342.79	.9937
LogNormal	.6454	.1323	-342.85	.9935
Gamma Inv	.5806	.1352	-342.92	.9934
Weibull	.0021		-345.75	.9828
Gumbel Min	.0007		-346.65	.9798
Gumbel Max	.0001		-348.12	.9755
Exponencial	.0000		-629.80	.2663

Figura A.1: Distribución del throughput para el bitrate de 256 Kbps y el Servidor 1



Modelo	p(A3)	p(A4)	LVer	R*
Weibull	.5870		-255.10	.9925
Gumbel Min	.4837		-255.24	.9918
Normal	.0043	.0779	-258.58	.9793
Gamma	.0030	.0624	-258.93	.9780
LogNormal	.0025	.0557	-259.10	.9774
Gamma Inv	.0020	.0496	-259.29	.9767
Gumbel Max	.0000		-277.88	.8693
Exponencial	.0000		-639.59	.1650

Figura A.2: Distribución del throughput para el bitrate de 256 Kbps y el Servidor 2



Modelo	p(A3)	p(A4)	LVer	R*
Weibull	.9993		-221.73	.9934
Gumbel Min	.8970		-221.82	.9931
Normal	.0005	.0049	-226.53	.9737
Gamma	.0003	.0037	-226.84	.9724
LogNormal	.0003	.0032	-227.00	.9717
Gamma Inv	.0002	.0028	-227.16	.9711
Gumbel Max	.0000		-250.88	.7949
Exponencial	.0000		-641.24	.1388

Figura A.3: Distribución del throughput para el bitrate de 256 Kbps y el Servidor 3

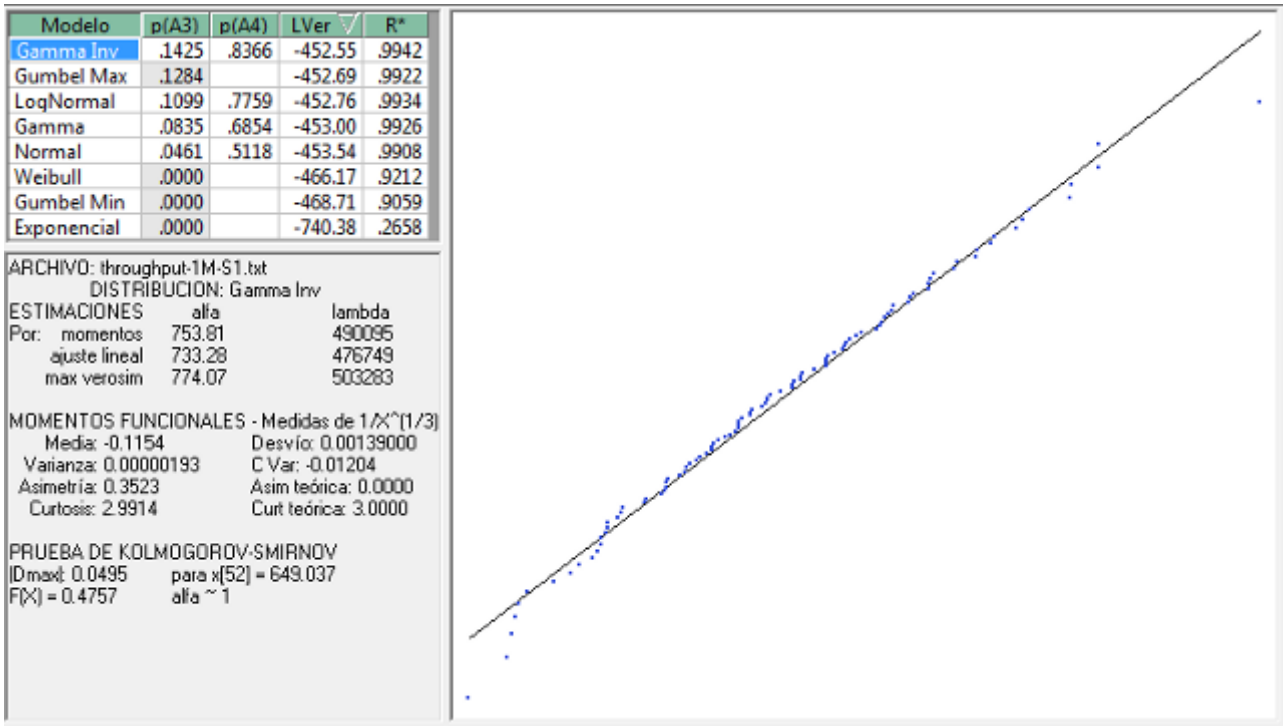
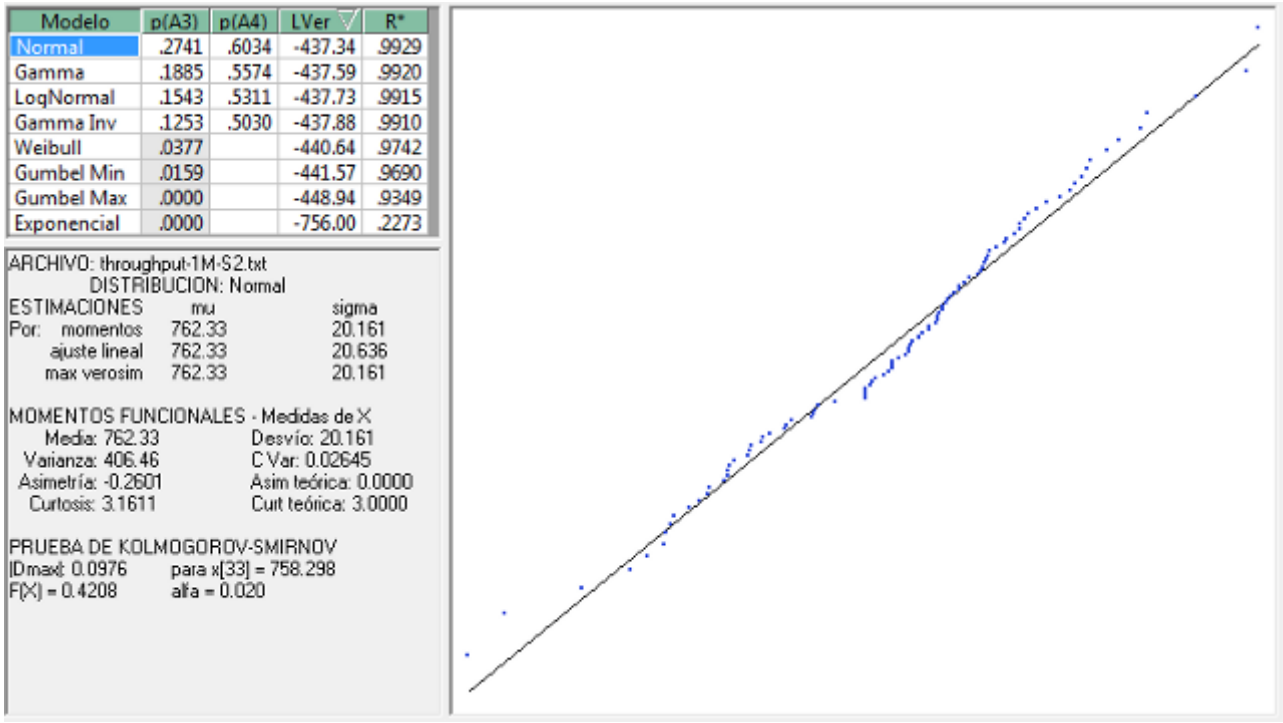
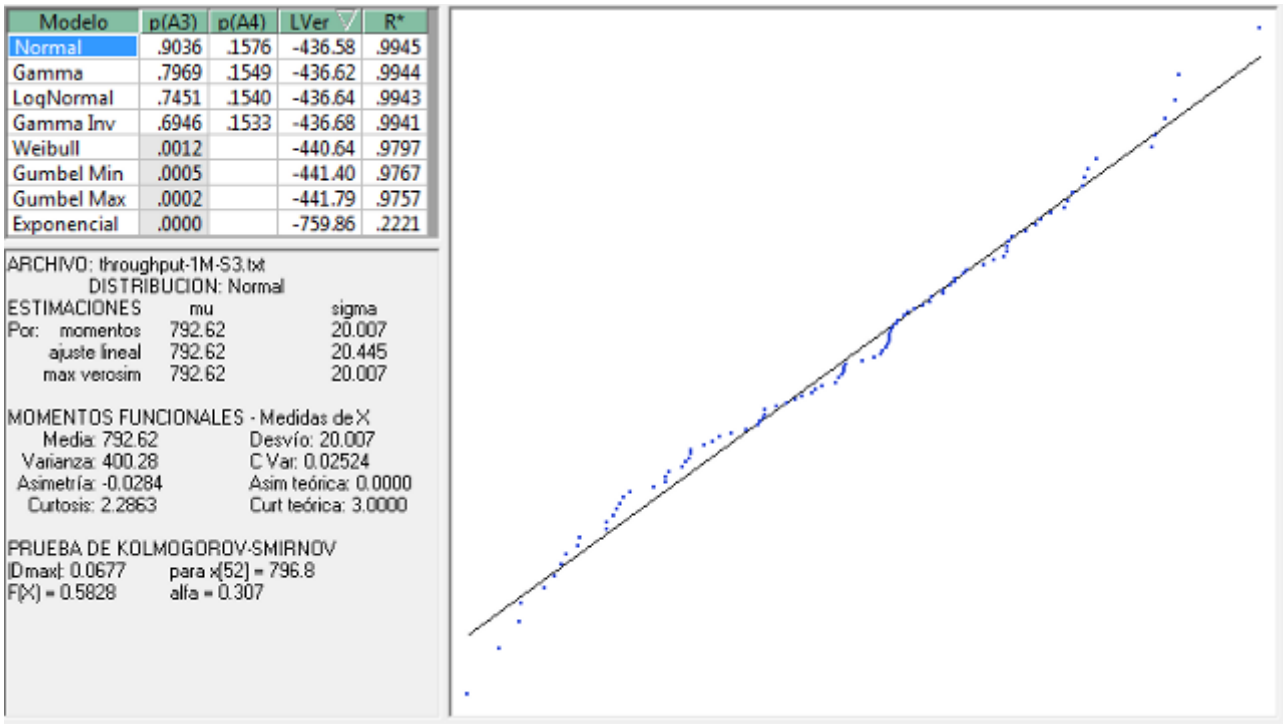


Figura A.4: Distribución del throughput para el bitrate de 1 Mbps y el Servidor 1



Modelo	p(A3)	p(A4)	LVer	R*
Normal	.2741	.6034	-437.34	.9929
Gamma	.1885	.5574	-437.59	.9920
LogNormal	.1543	.5311	-437.73	.9915
Gamma Inv	.1253	.5030	-437.88	.9910
Weibull	.0377		-440.64	.9742
Gumbel Min	.0159		-441.57	.9690
Gumbel Max	.0000		-448.94	.9349
Exponencial	.0000		-756.00	.2273

Figura A.5: Distribución del throughput para el bitrate de 1 Mbps y el Servidor 2



Modelo	p(A3)	p(A4)	LVer	R*
Normal	.9036	.1576	-436.58	.9945
Gamma	.7969	.1549	-436.62	.9944
LogNormal	.7451	.1540	-436.64	.9943
Gamma Inv	.6946	.1533	-436.68	.9941
Weibull	.0012		-440.64	.9797
Gumbel Min	.0005		-441.40	.9767
Gumbel Max	.0002		-441.79	.9757
Exponencial	.0000		-759.86	.2221

Figura A.6: Distribución del throughput para el bitrate de 1 Mbps y el Servidor 3

Comparando los distintos modelos (cada una de las potenciales distribuciones adoptadas), en todos los casos, se observa que el gráfico de ajuste lineal de fractiles no varia sustancialmente entre los primeros tres candidatos. Por ejemplo, para el caso de 1 Mbps y el servidor 2, las distribuciones Normal, Gamma y LogNormal generan prácticamente el mismo gráfico (que aparece en cada una de las figuras como una recta acompañada de un grupo de puntos que corresponde al modelo seleccionado). Este gráfico resulta ser un buen indicador de la validez del modelo. Por este motivo en los seis casos se tomo como modelo más factible al primero de la lista de las verosimilitudes máximas.

Bibliografía

[1] Luciano Iglesias y Luis Marrone. Desempeño de tráfico tipo streaming en una red de datos simulada. *CACIC 2011 XVII Congreso Argentino de Ciencias de la Computación*, October 2011.

[2] D Miras. Network QoS needs of advanced internet applications a survey. Technical report, Internet2, November 2002.

[3] William Stallings. *High-speed networks and internets: performance and quality of service*. Prentice Hall, 2002.

[4] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, December 2004.

[5] D. Lee, B. Carpenter, and N. Brownlee. Observations of UDP to TCP ratio and port numbers. Technical report, 2009.

[6] Cisco visual networking index: Forecast and methodology, 2011-2016 [Visual networking index (VNI)]. Technical report, 2011.

[7] G.R. Dattatreya. *Performance Analysis of Queuing and Computer Networks*. Chapman and Hall/CRC, 1 edition, June 2008.

[8] Vern Paxson and Sally Floyd. Wide-area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, pages 226–244, 1995.

[9] Richard M. Fujimoto, Kalyan S. Perumalla, and George F. Riley. *Network Simulation*. Morgan & Claypool Publishers, January 2007.

[10] BRITE: boston university representative internet topology gEnerator. http://www.cs.bu.edu/brite/.

[11] GTNetS - home. http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/.

[12] VNUML-WIKI. http://neweb.dit.upm.es/vnumlwiki/index.php/Main_Page.

[13] The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[14] ns-3. http://www.nsnam.org/.

[15] Osvaldo L. Mermoz and Roberto M. García. *Distribuciones univariantes de probabilidad: Modelos y su identificación*. Nueva librería, 2006.

[16] Joaquim P. Marques de Sá. *Applied Statistics Using SPSS, STATISTICA, MATLAB and R*. Springer, 2nd edition, July 2007.

[17] Jay L. Devore. *Probability & Statistics for Engineering and the Sciences*. Cengage Learning, 2012.

[18] Tahir Nawaz Minhas, Markus Fiedler, and Patrik Arlos. Quantification of packet delay variation through the coefficient of throughput variation. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, IWCMC '10, page 336–340, New York, NY, USA, 2010. ACM.

[19] S. Floyd. Metrics for the evaluation of congestion control mechanisms. http://tools.ietf.org/html/rfc5166.