

**ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN**

UNIVERSIDAD DE MÁLAGA



PROYECTO FIN DE CARRERA

*GENERACIÓN DE TRÁFICO SINTÉTICO CON
DIFERENCIACIÓN DE TIPOS DE OBJETOS PARA UNA
CACHE WEB*

INGENIERÍA DE TELECOMUNICACIÓN

MÁLAGA, 2007

RAÚL JIMÉNEZ JIMÉNEZ

**ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA DE TELECOMUNICACIÓN**

UNIVERSIDAD DE MÁLAGA

Titulación: Ingeniería de Telecomunicación

Reunido el tribunal examinador en el día de la fecha, constituido por:

D./D^a. _____

D./D^a. _____

D./D^a. _____

para juzgar el Proyecto Fin de Carrera titulado:

***GENERACIÓN DE TRÁFICO SINTÉTICO CON DIFERENCIACIÓN
DE TIPOS DE OBJETOS PARA UNA CACHE WEB***

del alumno D. RAÚL JIMÉNEZ JIMÉNEZ

dirigido por D. FRANCISCO JAVIER GONZÁLEZ CAÑETE

ACORDÓ POR _____ OTORGAR LA
CALIFICACIÓN DE _____

Y, para que conste, se extiende firmada por los componentes del tribunal, la presente diligencia

Málaga, a _____ de _____ de _____

El Presidente

El Vocal

El Secretario

Fdo.: _____ Fdo.: _____ Fdo.: _____

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD DE MÁLAGA

*GENERACIÓN DE TRÁFICO SINTÉTICO CON DIFERENCIACIÓN DE TIPOS
DE OBJETOS PARA UNA CACHE WEB*

REALIZADO POR:

Raúl Jiménez Jiménez

DIRIGIDO POR:

Francisco Javier González Cañete

DEPARTAMENTO DE: Tecnología Electrónica

TITULACIÓN: Ingeniería de Telecomunicación

PALABRAS CLAVE: Tráfico sintético, Zipf, localidad temporal, modelado, tipos de documento, Web cache.

RESUMEN:

En este proyecto Fin de Carrera se ha desarrollado un generador de muestras de tráfico con diferenciación de tipos de objeto, introduciendo para ello una serie de parámetros de entrada que permiten caracterizar las muestras a generar. La asignación de estos parámetros se puede realizar para el conjunto de todas las muestras generadas o bien distinguiendo para cada uno de los tipos de documentos presentes en la Web. Posteriormente, se ha verificado que los resultados obtenidos tras la generación coinciden con los resultados esperados. Estas muestras de tráfico sintético pueden ser usadas para evaluar el rendimiento de cachés Web en función de los parámetros característicos de las muestras.

Málaga, Junio 2007

Índice de contenidos

CAPÍTULO 1: Introducción	1
CAPÍTULO 2: Conceptos teóricos	7
2.1 Internet.....	7
2.2 Cliente Web.....	9
2.3 Servidores Web	10
2.4 Servidor <i>Proxy</i>	12
2.5 ¿Qué es una <i>cache</i> ?	14
2.5.1 <i>CACHES WEB</i>	16
2.5.2 TOPOLOGÍAS DE LAS <i>CACHES</i>	19
2.5.3 <i>CACHE PROXY</i>	20
2.6 Ley de Zipf.....	23
2.7 One - timers	25
2.8 Distribución de Pareto.....	26
2.9 Distribución logarítmico normal	27
CAPÍTULO 3: Manual de usuario	29
3.1. Ejecución.....	29
3.2. Menú principal	30
3.3 Menú “ <i>General</i> ”	32
3.4 Menú “ <i>Por tipo de archivos</i> ”	37
3.5. Menú “ <i>Analizar</i> ”	39
3.6. Menú “ <i>Mezclar</i> ”	42
CAPÍTULO 4: Implementación.....	45

4.1. Descripción de la aplicación	45
4.2. “One-timers”	46
4.2.1. INTRODUCCIÓN	46
4.2.2. IMPLEMENTACIÓN	47
4.3. Popularidad.....	49
4.3.1 INTRODUCCIÓN	49
4.3.2 IMPLEMENTACIÓN	50
4.4. Distribución del tamaño de los documentos	52
4.4.1. INTRODUCCIÓN	52
4.4.2. IMPLEMENTACIÓN	52
4.4.2.1. Modelación de la cola	53
4.4.2.2. Modelación del cuerpo de la distribución	55
4.4.2.3. Unión de ambas distribuciones	57
4.5. Correlación entre el tamaño de los archivos y su popularidad	58
4.6. Localidad temporal.....	61
4.7. Generación de muestras	64
 CAPÍTULO 5: Pruebas	 67
5.1. Validación	69
5.2. Variación de los parámetros de entrada	75
5.2.1 “ONE-TIMERS”	75
5.2.2 PENDIENTE DE ZIPF	77
5.2.3 ÍNDICE DE COLA PESADA	79
5.2.4 CORRELACION	81
5.2.5 MODO DE LA PILA	83
5.3. Mezclas de distintos tipos de archivos	84
 CAPÍTULO 6: Conclusiones y líneas futuras.....	 89
 BIBLIOGRAFÍA.....	 93

Índice de figuras

Figura 2.1. Conexión directa entre cliente y servidor.....	8
Figura 2.2. Conexión cliente – servidor a través de un <i>proxy</i>	9
Figura 2.3. Funcionamiento básico HTTP.....	11
Figura 2.4. Acceso a Internet a través de un servidor <i>proxy</i>	12
Figura 2.5. Ejemplo de uso de un <i>proxy</i>	14
Figura 2.6. Ejemplo de uso de un <i>proxy cache</i>	15
Figura 2.7. Ejemplo de acierto en <i>cache</i>	18
Figura 2.8. Distribución de Zipf en ejes logarítmicos	25
Figura 2.9. Función de distribución de Pareto.....	26
Figura 2.10. Cola pesada en una distribución de Pareto.....	27
Figura 2.11. Densidad de probabilidad de una distribución logarítmica normal	28
Figura 2.12. Función de distribución acumulada logarítmica normal	28
Figura 3.1. Modificar el directorio de trabajo	30
Figura 3.2. Pantalla principal de la aplicación.....	30
Figura 3.3. Pantalla del menú “ <i>General</i> ”	32
Figura 3.4. La parte superior de la pantalla corresponde siempre al Menú Principal	32
Figura 3.5. <i>Parámetros</i> del menú “ <i>General</i> ”	33
Figura 3.6. Mensaje de error por fuera de rango.	34
Figura 3.7. Modo de funcionamiento de la pila.....	35
Figura 3.8. Error al no conseguir el 100% en su totalidad	36
Figura 3.9. Guardamos el archivo y generamos muestras.....	36
Figura 3.10. Menú “por tipo de archivos”	37
Figura 3.11. Se puede elegir de qué tipo de archivo se generan muestras	38
Figura 3.12. Generación de tráfico tipo “ <i>Aplicación</i> ”	38
Figura 3.13. Opción “ <i>Analizar</i> ”	39
Figura 3.14. Selección de cualquier archivo.....	40
Figura 3.15. Análisis de muestras de tráfico	41
Figura 3.16. Menú “ <i>Mezclar</i> ”	43
Figura 4.1. Algoritmo para la generación del modelo de <i>one-timers</i>	48
Figura 4.2. Algoritmo de generación de popularidades según la distribución de Zipf... 51	51
Figura 4.3. Algoritmo de generación de la cola de Pareto en la distribución de tamaño 55	55
Figura 4.4. Algoritmo de generación de valores para una distribución logarítmica normal.....	57
Figura 4.5. Ejemplo de generación de muestras.....	65
Figura 5.1. Análisis del comportamiento de la popularidad.....	70
Figura 5.2. Análisis de la correlación entre el tamaño y la frecuencia de acceso	71
Figura 5.3. Análisis de la función de distribución acumulada.....	71
Figura 5.4. Análisis de la función de densidad de probabilidad del tamaño	72
Figura 5.5. Análisis de la pendiente de cola de Pareto	73
Figura 5.6. Análisis de la localidad temporal en modo dinámico	73
Figura 5.7. Mayor pendiente de Zipf significa mayor número de muestras.....	78

Figura 5.8. Resultados gráficos para simulación con pendiente de Zipf de 0,65	78
Figura 5.9. Representación gráfica de la simulación con pendiente de Zipf de 0,85	79
Figura 5.10. Representación gráfica de la simulación con pendiente de cola de 1,1	80
Figura 5.11. Representación gráfica de la simulación con pendiente de cola de 1,3	80
Figura 5.12. La cola con pendiente menor tiene mayor área.....	81
Figura 5.13. Representación gráfica de la simulación con correlación positiva	82
Figura 5.14. Representación gráfica de la simulación con correlación negativa	82
Figura 5.15. Representación gráfica de la simulación con modo de pila estático.....	84
Figura 5.16. Resultados gráficos de la mezcla de distintos patrones de muestras	86

Índice de Tablas

Tabla 5.1. Valores típicos para la generación de muestras de tráfico	70
Tabla 5.2. Valores obtenidos tras la simulación con valores típicos.....	74
Tabla 5.3. Comparativa de simulaciones con distinto porcentaje de <i>one-timers</i>	76
Tabla 5.4 Comparativo de tipos de archivos obtenidos para distintos <i>one-timers</i>	77
Tabla 5.5. Comparativo de tipos de archivos obtenidos para distintas pendientes de Zipf	77
Tabla 5.6. Comparativo de tipos de archivos obtenidos para distintas pendientes de cola pesada	79
Tabla 5.7. Datos estadísticos de simulaciones con correlación positiva y negativa respectivamente	83
Tabla 5.8. Parámetros de entrada para la mezcla de documentos	85
Tabla 5.9. Parámetros resultantes de la mezcla de simulaciones de diferentes tipos de archivos.....	86

CAPÍTULO 1: Introducción

En los últimos años se ha producido un aumento considerable de la popularidad de las páginas Web, que se han introducido en todos los ámbitos de nuestra sociedad. Muchas razones explican este incremento, entre las que destacamos: la disponibilidad de interfaces gráficas de usuario para navegar, la existencia de editores y herramientas de soporte para la creación y publicación de documentos Web; una tendencia de investigadores, instituciones educacionales y públicas, y organizaciones comerciales para usar la Web como diseminador de información y de enlace con sus clientes.

Este crecimiento ha llevado consigo la aparición de problemas tales como: grandes congestiones en la red, un bajo ancho de banda, altas latencias al momento de extraer un documento, sobrecarga en servidores, accesos simultáneos masivos a un servidor, entre otros.

Muchos han sido los intentos para paliar estos efectos, pero sin duda, una de las soluciones más utilizadas es el uso de *caches*, cuya misión es la de almacenar aquellos documentos que han sido solicitados por los usuarios con mayor frecuencia. Cuando se produce la petición de un documento ya consultado, éste será servido directamente por la *cache* disminuyendo, por tanto, el tiempo de servicio, sobrecarga de servidores,... pero antes, se comprobará que no haya expirado y deberá ser validado ya que en caso contrario requerirá la conexión con el servidor remoto para que proporcione de nuevo el documento tanto para la *cache* como para el usuario que lo solicitó.

Teniendo en cuenta, que la mayoría de los accesos a documentos Web son estáticos, como documentos de audio, video o las propias páginas Web entre otros, si estos documentos estuviesen disponibles en una *cache* de primer nivel ubicada en el navegador del usuario, ya podríamos reducir una parte importante del tráfico generado.

Si tenemos en cuenta que es muy usual que una red de usuarios se conecten a Internet a través de un servidor *proxy* (servidor que se conecta a Internet por nosotros) y que en éste también se puede hacer uso de una *cache* (que en este caso sería equivalente

a una *cache* de segundo nivel) los beneficios mencionados anteriormente serían aún mayores.

Los servidores *proxy* actúan como intermediarios entre los servidores Web de Internet y los usuarios. Fundamentalmente, tienen como misión el proporcionar a los usuarios de una red, un acceso a Internet seguro, actuando como cortafuegos, encargándose de recibir las peticiones y enviarlas a los servidores Web remotos de modo que una vez recibida las respuestas, sean reenviadas al usuario correspondiente.

Surge así el *web proxy cache* que almacena en su disco duro las páginas Web descargadas de forma que, en próximas consultas, pueda acceder a ellas de forma muy rápida. De esta forma optimizamos el canal de acceso a Internet y mejoramos la sensación de navegación del usuario, siendo un proceso totalmente transparente a éste.

De este modo, con el uso de estas *caches* podemos conseguir, entre otras, reducir el tráfico de la red, la latencia media, el tiempo de espera entre la petición de un documento y la recepción del mismo, así como la carga soportada por los servidores Web.

Se han realizado un gran número de análisis de las características del tráfico Web [1],[2],[3] y las características comunes encontradas son: (1) la popularidad de los documentos solicitados suele seguir la ley de Zipf [1],[2],[10]; (2) la distribución del tamaño de los archivos para documentos Web es de cola pesada [1],[12]; (3) muchos (e.j. 50-70%) de los documentos son *one-timers* (solicitados una sola vez) [1],[3],[12]; y (4) existe localidad temporal entre las peticiones de documentos en una *cache Web* [3],[4],[5].

Todas estas características son muy importantes para el estudio de una *cache Web*, ya que dependiendo de la estrategia de reemplazo que se utilice para una determinada *cache* será más o menos eficiente, y estas estrategias de reemplazo dependerán de dichas características fundamentales.

La estrategia de reemplazo es el mecanismo por el cual una *cache* determina que documento almacenado debe abandonar dicha *cache* (ya que tiene un tamaño limitado) para que entre un documento nuevo que haya sido solicitado por el usuario.

Por tanto, el uso de *caches* es una buena solución para agilizar la atención de peticiones Web, pero el estudio de estas *caches* necesita de tráfico real en la red, y cada vez que se realicen pruebas se va a necesitar de este tipo de tráfico lo cual retardaría mucho estas simulaciones, ya que se deberá contar con un elevado número de usuarios para que generasen estas muestras de tráfico, además, el ajuste de los parámetros del tráfico a simular va a ser muy complejo. Es aquí donde surgió la necesidad de la creación de un generador de tráfico sintético.

Actualmente, existen diferentes simuladores de tráfico sintético: ProWGen [13], TrGen [14], GenSyn [15]. Estos simuladores son capaces de crear muestras de tráfico sintético a partir de la introducción de una serie de parámetros o bien mediante la introducción de muestras de tráfico real, que son analizadas y posteriormente se simula su comportamiento.

Cada día, el análisis de tráfico de Internet es más variado y profundo, este análisis también distingue el tipo de tráfico que circula por Internet, y se pueden extraer distintas conclusiones acerca del tipo de tráfico en función de distintas variables socio-económicas, culturales,

Es por esto, que para una mejor optimización de los recursos, la eficiencia de una *cache* se incrementaría notablemente si se consiguiese adaptar dichas *caches* a tipos de tráfico diferentes según las características de los usuarios, es decir, dependiendo del tipo de tráfico que circula por la red, así debe actuar la *cache*, partiendo de este hecho, se vuelve a repetir la misma circunstancia ya descrita anteriormente, y es que, para poder hacer pruebas con este tipo de *caches* y conseguir su óptima implementación es necesario la creación de un generador de tráfico sintético que sea capaz de diferenciar el tipo de tráfico a simular.

El objetivo de este Proyecto Fin de Carrera, es la elaboración de una aplicación que genere muestras de tráfico, a partir de una serie de parámetros, y que podamos distinguir el tipo de tráfico que queremos simular.

Los tipos de tráfico que se podrán simular serán audio, video, texto, imagen, mensaje y aplicación, puesto que son los más representativos. Y se podrá mezclar distintos tipos de tráfico en una misma simulación.

En principio, la implementación sería en lenguaje JAVA, pero posteriormente, se optó por hacerlo usando MATLAB puesto que se parte con algunos algoritmos básicos ya implementados y se ha realizado de modo que la ampliación a nuevos tipos de tráfico o de cualquier otra característica sea lo más sencillos posible y que no sería tal en caso del lenguaje JAVA.

La presente memoria consta de 6 capítulos que se describen a continuación. Un primer capítulo de introducción donde se describe el marco tecnológico donde se engloba el proyecto, así como los objetivos que se persiguen con la elaboración del mismo.

En el segundo capítulo se presentarán los conceptos y conocimientos teóricos básicos para la comprensión del Proyecto Fin de Carrera.

En el tercer capítulo se muestra un manual de usuario en el que se dará a conocer el funcionamiento del generador de muestras así como algunos aspectos conceptuales aplicados al generador.

En el cuarto capítulo, se explica detalladamente cómo ha sido el proceso para el diseño de la aplicación, detallando los modelos matemáticos utilizados.

En el quinto capítulo, se explica cómo se ha realizado el procesamiento de las muestras, y después se realiza un plan de pruebas donde se verifica el correcto funcionamiento del generador, es decir, que los resultados obtenidos coinciden con los resultados teóricos que se deben obtener.

En el capítulo seis, se expone las conclusiones a las que se han llegado tras la realización del proyecto, así como un breve resumen del mismo. Para terminar, se describirá brevemente algunas líneas futuras de investigación.

CAPÍTULO 2: Conceptos teóricos

En este capítulo, se presentarán los conceptos teóricos sobre *proxy*, *cache* así como de los parámetros fundamentales para caracterizar tráfico de Internet.

2.1 Internet

Hasta principio de los 90's, los principales usuarios de Internet eran estudiantes, administración pública e investigadores y las principales aplicaciones usadas eran el correo electrónico, noticias, conexiones remotas, y transferencias de archivos. Una nueva aplicación, la WWW (*World Wide Web*) lo revolucionó todo, acercando a millones de personas de toda índole a Internet.

Originalmente, fue concebida por la CERN (*Conseil Européen pour la Recherche Nucléaire* – Consejo Europeo para la Investigación Nuclear), la WWW fue ideada como un espacio de hipertexto global [17] en la cual cualquier información accesible de la red, podía ser referenciada mediante una única “*Universal Document Identifier* – Identificador Único de Documento”, lo que más tarde sería URL (*Uniform Resource Locator* – Localizador Uniforme de Recurso). El diseño de la WWW estaba basado en unos pocos conceptos básicos: un sistema de información debe ser capaz de relacionar asociaciones entre objetos arbitrarios; los usuarios no deben estar obligados a usar un lenguaje particular o sistema operativo; la información debe estar disponible en todas las plataformas, incluidas las futuras; la entrada o modificación de la información debe ser trivial.

La arquitectura de la Web está basada en un modelo cliente-servidor [6], donde el cliente accede a la información de un servidor remoto usando un software especial para Web¹. Actualmente, existen muchos navegadores, que van desde los basado en texto a

¹ El primer navegador Web fue escrito por Tim Berners – Lee y lo llamó “WorldWideWeb”, pero después se le cambió el nombre para evitar confusión [17].

los GUI (*Graphical User Interface* – Interfaz gráfico de usuario) como son el Netscape Communicator y Microsoft Internet Explorer.

Estos navegadores se comunican con los servidores Web usando el protocolo HTTP (*Hypertext Transfer Protocol* – Protocolo de transferencia de hipertexto). El protocolo HTTP depende del protocolo subyacente TCP/IP mientras está trayendo una página desde un servidor. TCP es un protocolo orientado a conexión fiable que permite el flujo de datos desde un equipo hacia otro equipo dentro de una intranet o de Internet.

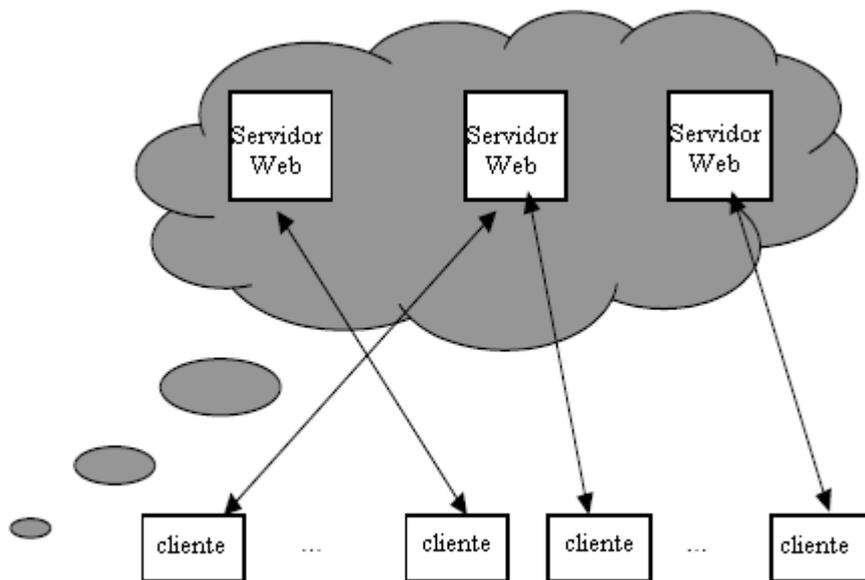


Figura 2.1. Conexión directa entre cliente y servidor

Hay dos modos distintos de obtener una página Web (e.j. archivos de texto, audio, video, imágenes,...) desde un servidor Web: mediante una conexión directa entre el cliente y el servidor o indirectamente conectándonos a un servidor *proxy*. Estos dos escenarios son representados en la figura 2.1 y en la figura 2.2. En el primer escenario (figura 2.1), un cliente Web establece una conexión directamente con el servidor, éste responde con la página solicitada que es visualizada por el navegador. En el segundo caso (figura 2.2), la petición del cliente es enviada a un servidor *proxy*, que remite la petición al servidor (si es necesario) y cuando le llegue la respuesta, se la mandará al navegador Web para que la visualice dicho usuario. Estos servidores *proxy* también pueden disponer de una *cache* local para satisfacer futuras peticiones.

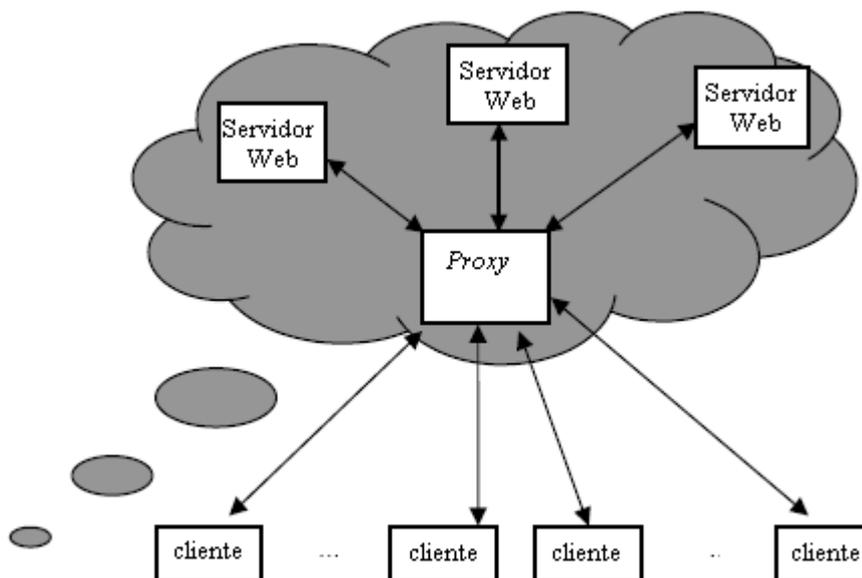


Figura 2.2. Conexión cliente – servidor a través de un *proxy*

En los siguientes apartados se describirán las principales entidades descritas en las figuras 2.1 y 2.2.

2.2 Cliente Web

La WWW es un almacén de archivos de diferentes tipos, y usualmente nos referimos a ellos como *documentos*. Cada documento puede contener enlaces a otros documentos en cualquier parte del mundo, aquellas documentos que apuntan a otras documentos se dice que usan *hipertexto*.

Los navegadores Web (p. ej. Netscape Navigator y Microsoft Internet Explorer) son programas que permiten la visualización de las páginas Web. Para visualizar una página Web, el navegador establece una conexión con un servidor Web y solicita una página Web, cuando la página llega, el navegador interpreta el texto y los comandos contenidos en él y visualiza la página en la pantalla con el formato adecuado. Con navegadores GUI, los enlaces a otras páginas están normalmente subrayados y/o en diferente color. A estos enlaces se les conoce como *hiperenlaces*. Algunos navegadores emplean una

memoria *cache* para almacenar las páginas más recientemente visitadas para aquellos casos en los que el usuario quiera visitar páginas que ya ha visualizado.

2.3 Servidores Web

Los servidores Web son procesos software que escuchan a través del puerto 80 peticiones HTTP del cliente estableciendo una conexión entre los equipos tal y como se muestra en la figura 2.3. Un navegador solicitará una página Web estableciendo una conexión con el servidor a través del puerto 80, una vez realizada la conexión, el navegador manda una petición GET, para buscar la página en cuestión, el servidor responde con un mensaje de respuesta HTTP con el siguiente formato [48]:

$$\begin{aligned} \text{Respuesta} &= \text{Línea de estado} \\ &\quad [\text{Cabecera}] \\ &\quad \text{CRLF} \\ &\quad [\text{Cuerpo del mensaje}] \end{aligned}$$

Línea de estado consiste normalmente en la versión del protocolo HTTP seguido de un número de código de estado y su descripción asociada:

$$\text{Línea de estado} = \text{Versión HTTP} \quad \text{Código de estado} \quad \text{Descripción}$$

El código de estado es un número de 3 dígitos que representa el estado de la petición del cliente, mientras la descripción informa brevemente acerca de lo que significa el código de estado. Hay 5 clases básicas de código de estado representadas por su primer dígito. Estas clases son:

- 1xx: los códigos que empiezan con 1 indican que la petición ha sido recibida por el servidor y está siendo procesada.
- 2xx: este código indica que el servidor es capaz de interpretar la petición del cliente y responderle satisfactoriamente.
- 3xx: necesita hacer más tareas para poder ser capaz de responder a la petición satisfactoriamente.

- 4xx: el servidor no puede responder satisfactoriamente la petición de cliente. (p.ej. error de usuario).
- 5xx: aunque la petición es válida, el servidor no puede realizarla (p.ej. error de servidor).

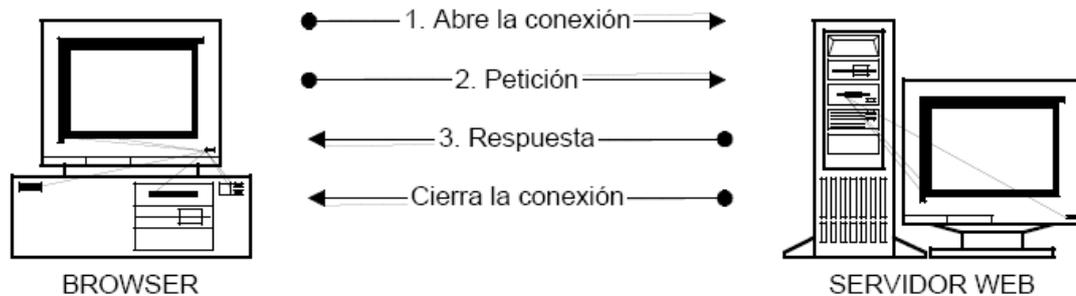


Figura 2.3. Funcionamiento básico HTTP

Sobre el servicio Web podemos disponer de aplicaciones Web. Éstas son fragmentos de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

- Aplicaciones en el lado del cliente: el cliente Web es el encargado de ejecutarlas en la máquina del usuario. Son las aplicaciones tipo *java* o *javascript*: el servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta. Es necesario, por tanto, que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones (también llamadas *scripts*). Normalmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje *javascript* y *java*, aunque pueden añadirse más lenguajes mediante el uso de *plugins*.
- Aplicaciones en el lado del servidor: el servidor Web ejecuta la aplicación; ésta, una vez ejecutada, genera cierto código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP.

Las aplicaciones de servidor suelen ser la opción por la que se opta en la mayoría de las ocasiones para realizar aplicaciones web. La razón es que, al ejecutarse ésta en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad adicional,

como sí ocurre en el caso de querer ejecutar aplicaciones *javascript* o *java*. Así pues, cualquier cliente dotado de un navegador web básico puede utilizar este tipo de aplicaciones.

2.4 Servidor *Proxy*

Un servidor *proxy* Web proporciona acceso a la Web para personas en subredes cerradas que tan sólo pueden acceder a Internet a través de un determinado servidor (como se muestra en la figura 2.4), siendo éste su principal uso aunque suele actuar también como cortafuegos. Incluso usuarios que no dispongan de DNS (*Domain Name System* – Sistema de nombres de dominios) pueden acceder a Internet a través de él siempre que conozcan la dirección IP (*Internet Protocol* – Protocolo Internet) del servidor *proxy*.

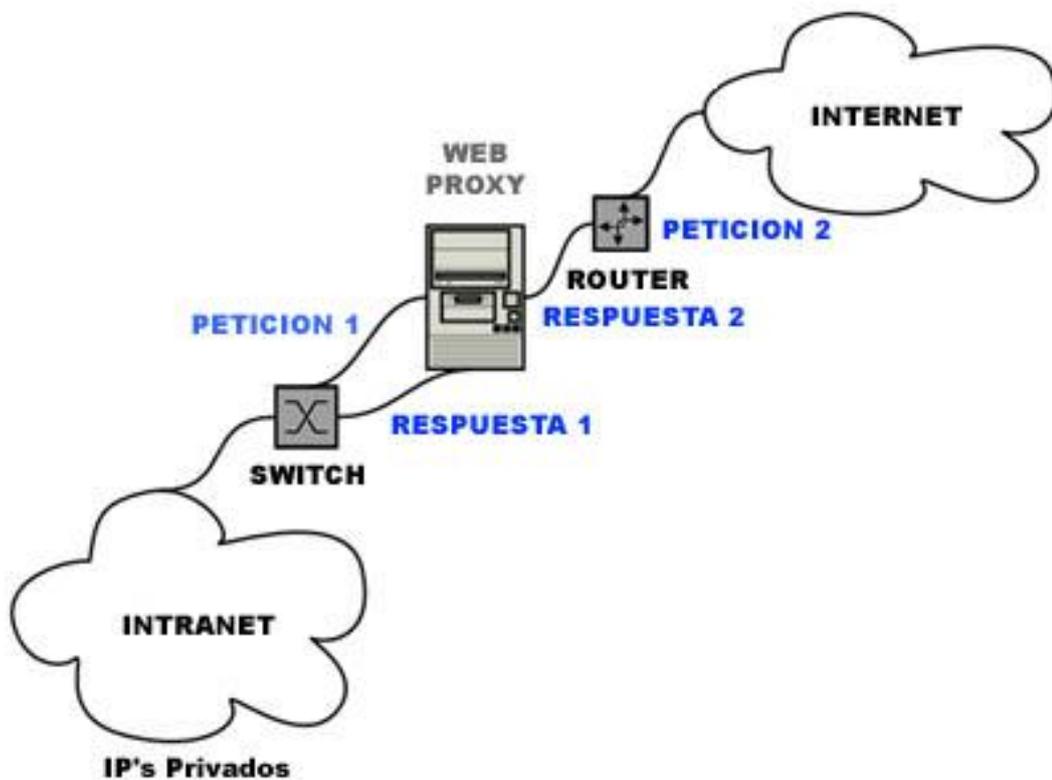


Figura 2.4. Acceso a Internet a través de un servidor *proxy*

El servidor *proxy* espera la llegada de una petición procedente de un usuario desde dentro del cortafuego. Una vez que se produce, si el servidor *proxy* no dispone del documento solicitado, o bien ya ha expirado, dirige la petición a un servidor remoto fuera del cortafuegos, lee la respuesta, y se la manda de vuelta al usuario que la solicitó, almacenándola en su memoria local. Actúa, por tanto, como intermediario entre el usuario y el servidor donde se almacena la información requerida y suele ser usado por todos los usuarios de una subred. Los usuarios no pierden funcionalidad haciendo uso del *proxy* excepto si es necesario hacer algún procesamiento especial.

El servidor *proxy* permite el almacenamiento en *cache*, de forma local, de las páginas consultadas con mayor frecuencia, o cualquier otro criterio, y así ante una petición de un usuario, si el documento solicitado está presente en *cache* y no ha alcanzado aún su tiempo de expiración o bien ha sido validado (el servidor remoto ha dado su visto bueno), podrá ser servido directamente de *cache*. La diferencia entre ambos casos se encuentra en que si no ha expirado, se considera que el documento sigue siendo válido, mientras que si lo ha hecho, pero el servidor ha confirmado que aún no ha sido modificado (comprobando por ejemplo, la cabecera *Last-Modified* al realizar una petición *If-Modified-Since*) también se puede seguir usando el documento y no es necesario volver a traerlo desde el servidor remoto original.

De este modo se obtiene un notable incremento en la velocidad de transferencia de la información con el consiguiente uso eficiente del ancho de banda y un menor tiempo de respuesta, también permite un ahorro en espacio de disco puesto que una sola copia es almacenada y usada por múltiples usuarios. Incluso puede permitir mostrar páginas Web aún cuando la red exterior no esté disponible, proporcionándolas de su propia *cache*. De este modo, el uso de un servidor *proxy* resulta interesante siempre a cualquier usuario.

El servidor *proxy* puede actuar como filtro de contenidos, como traductor de formato de archivos, adaptador de protocolos (sin pérdida de funcionalidad) o para verificar la seguridad (virus, accesos permitidos, etc.) incrementando la seguridad de la red interna y pudiendo ser tan efectivo como un cortafuegos.

A la hora de hacer transacciones con el servidor proxy, el cliente siempre usará HTTP aún cuando el recurso pedido se encuentre en un servidor remoto que use un protocolo distinto, como por ejemplo FTP (*File Transfer Protocol* – Protocolo de transferencia de archivos), como se muestra en la figura 2.5. A la hora de hacer la petición al servidor *proxy*, se especificará la URL completa y no sólo el nombre de la ruta u otras claves de búsqueda adicionales como ocurre en HTTP.

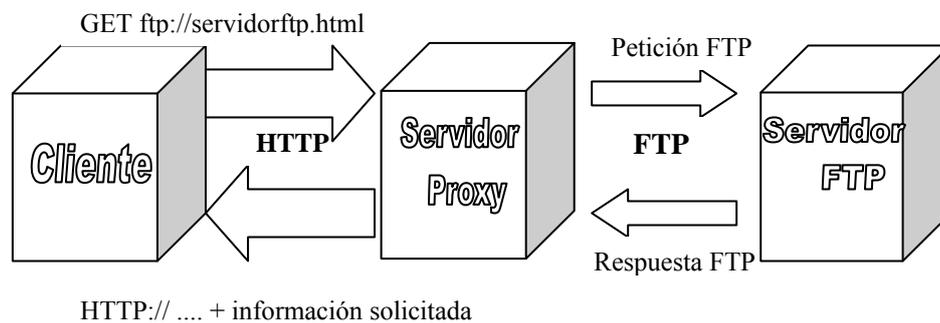


Figura 2.5. Ejemplo de uso de un *proxy*

Por tanto, el servidor proxy deberá realizar funciones tanto de servidor como de cliente. Actuará como servidor cuando acepte las peticiones HTTP de los usuarios conectados a él pero actuará como cliente cuando curse esas peticiones hacia los servidores remotos para poder obtener de ellos los documentos pedidos por los usuarios.

2.5 ¿Qué es una *cache*?

Se encarga del almacenamiento de los documentos más frecuentemente consultados por los usuarios, como se muestra en la figura 2.6. El uso de una *cache* sólo será beneficioso cuando el coste de almacenamiento de un documento sea menor al que supondría traer dicho documento directamente de un servidor origen. El concepto de *cache* es aplicable a casi todos los aspectos de la computación y los sistemas de red. Los procesadores de los computadores tienen *caches* tanto de datos como de instrucciones mientras que los sistemas operativos tienen *caches* para manejadores de disco y archivos de sistema.

El buen funcionamiento de las *caches* radica en el principio de localidad que siguen las referencias. Existen dos tipos de localidad: temporal y espacial. La localidad temporal hace referencia a la popularidad de los documentos (en un período de tiempo existe más probabilidad que se soliciten ciertos documentos que otros), mientras que la localidad espacial tiene que ver con el hecho de que ciertos documentos tienen la tendencia de ser pedidos conjuntamente con otros, como por ejemplo las imágenes que puedan haber en una determinada página Web. Así, al solicitarse una página Web también existe una elevada probabilidad de que se soliciten las imágenes que cohabitan en la misma página.

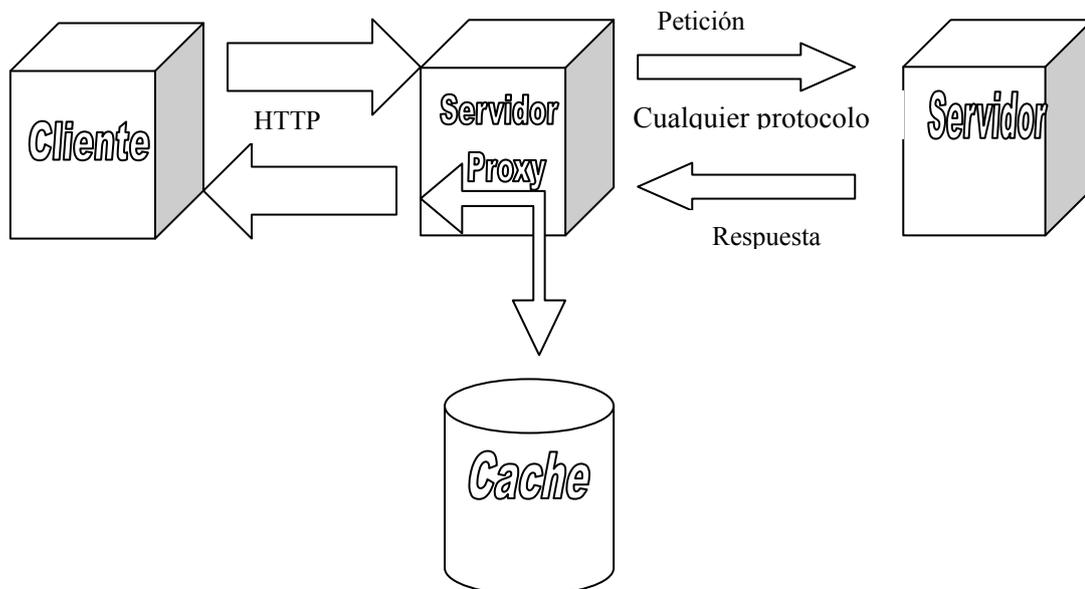


Figura 2.6. Ejemplo de uso de un proxy cache

Las *caches* usan la localidad de referencia para predecir accesos futuros basados en otros previos y en el caso de que la predicción sea correcta se producirá un incremento notable del rendimiento. Cuando un usuario solicita un documento y se verifica que se encuentra en *cache* entonces se le llama acierto en *cache* mientras que en caso de que haya que ir al servidor por dicho documento se hablará de fallo en *cache*. La mayoría de las tareas de procesamiento de datos exhiben localidad de referencia y por lo tanto se benefician del almacenamiento en *cache*.

Cualquier sistema que use una *cache* tiene que disponer de mecanismos de mantenimiento de consistencia en *cache*. Este es el procedimiento por el que los documentos almacenados en *cache* se mantienen actualizados respecto de los originales. Los documentos podrán ser frescos, es decir, se está seguro que el documento almacenado en *cache* es él mismo que el almacenado en el servidor, u obsoletos, en cuyo caso no se tendrá certeza que coincidan dichos documentos. Los frescos pueden ser utilizados inmediatamente pero los obsoletos requieren una validación previa para su uso. A la hora de llevar a cabo dicha validación, los algoritmos para mantener la consistencia pueden ser fuertes o débiles. En los algoritmos débiles, la *cache* a veces devuelve documentos obsoletos. Los fuertes, sin embargo, obligan siempre a una validación previa de los documentos antes de que puedan ser enviados al usuario. La CPU (*Central Process Unit* – Unidad Central de Procesos) y los archivos de sistema requieren consistencia fuerte mientras que otros sistemas como aquellos situados en *routers* o encaminadores son efectivos usando consistencia débil.

2.5.1 CACHES WEB

El uso de las *caches* Web se justifica en base a tres beneficios principales como son: la reducción en la latencia, en el uso de ancho de banda y en la carga de tráfico soportada por los servidores remotos, donde se encuentran originalmente los documentos a los que se desea acceder.

La latencia, básicamente, está referida a los retardos que sufre la transmisión de la información de un punto a otro. La transmisión de información sobre circuitos eléctricos y ópticos está limitada por la velocidad de la luz, de hecho, los pulsos eléctricos y ópticos viajan aproximadamente a dos tercios de la velocidad de la luz en cables y fibras, lo que genera un retardo que se incrementa con la distancia y que en conexiones transoceánicas es bastante considerable. Por otro lado, si en las comunicaciones se emplean enlaces que están próximos a su límite de máximo uso, los documentos sufren largos retardos en las colas de los encaminadores y conmutadores. Esto puede ocurrir en un elevado número de puntos a lo largo del camino seguido por la información, lo que puede ocasionar grandes retardos e incluso la pérdida de la información en el caso de

que sea descartada de la cola de un encaminador si dicha cola se llena. Con el uso de protocolos fiables como TCP, los paquetes perdidos se pueden retransmitir, pero una pequeña cantidad de paquetes perdidos puede dar lugar a una disminución importante del rendimiento de la comunicación a causa de las retransmisiones que se pueden originar.

Una *cache* Web situada cerca de sus usuarios reduce la latencia para los aciertos en *cache*. Los retardos de transmisión que se producen son mucho menores porque los sistemas se encuentran próximos. Además, los retardos originados por retransmisiones o por la introducción de los paquetes en colas son menos probables pues en la transmisión se ven involucrados menos encaminadores y enlaces. Por otro lado, un fallo en *cache* no deberá sufrir un retardo mucho mayor del que supondría una transferencia directa entre el usuario y el servidor original. Por lo tanto, los aciertos en *cache* reducen la latencia media de todas las peticiones.

Otro de los beneficios principales del uso de *caches* Web es la reducción en el ancho de banda utilizado, puesto que cada petición que supone un acierto en *cache* permite un ahorro de dicho ancho de banda. Si una conexión de Internet está congestionada, instalando una *cache* se mejora el rendimiento de otras aplicaciones de red, ya que todas compiten por el mismo ancho de banda. Así, con las *caches* Web se consigue reducir el ancho de banda consumido por tráfico HTTP dejándolo libre para otras aplicaciones que también lo requieran.

El tercer beneficio principal del uso de *cache* Web es la reducción en la carga de tráfico soportado por los servidores origen de los documentos. El tiempo de respuesta de un servidor aumenta conforme la tasa de peticiones al mismo es mayor, mientras que, por el contrario, un servidor libre de peticiones es más rápido respondiendo a la llegada de alguna petición, por lo que, si haciendo uso de *caches* Web se consigue esa disminución de carga de trabajo se está favoreciendo la mayor efectividad en las conexiones con los servidores remotos.

Por todo ello, se buscó la alternativa del uso de *caches* Web para conseguir que los usuarios apreciaran unos menores retardos en sus peticiones de documentos, los gestores de las redes un menor tráfico y los servidores Web una menor tasa de

peticiones y todo ello gracias a disponer de copias de los documentos más populares en servidores próximos a los usuarios. En definitiva, el funcionamiento de la *cache* se basa en que, una vez la petición de un usuario es almacenada en la *cache* del servidor *proxy*, ante una segunda petición de la misma página por parte de otro usuario, el servidor *proxy* podrá servirla directamente de *cache*, siempre que la copia no haya expirado o de haber sido así, haya sido validada, evitando de este modo el acceso remoto, tal y como se muestra en la figura 2.7.

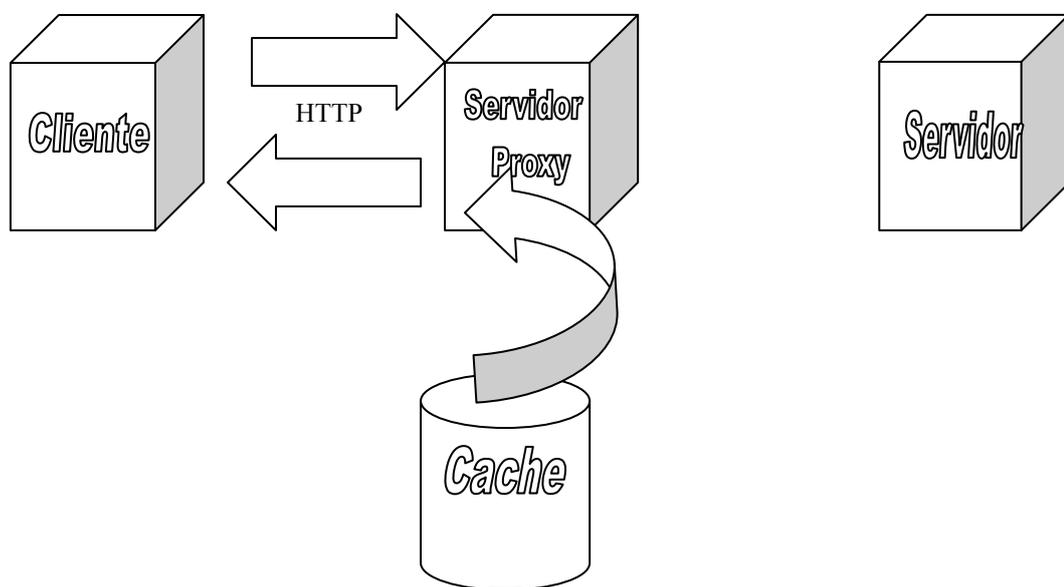


Figura 2.7. Ejemplo de acierto en *cache*

Por tanto, la *cache proxy* suele actuar como una *cache* de segundo nivel ya que recibe sólo los fallos en *cache* procedentes de los usuarios Web que ya usan una *cache* a nivel de usuario. De los aciertos en *cache* de usuario tan sólo recibe peticiones para comprobar la frescura del documento existente en la *cache* de usuario y en el caso de comprobarse la validez del mismo, permite un ahorro en ancho de banda para ambas *caches* simultáneamente.

Existen distintos tipos de caches Web entre los que podemos distinguir:

- *Cache* de navegador (o de usuario): como su nombre indica está asociada al navegador que usa el usuario. Esta *cache* funciona siguiendo unas reglas bastante simples y se encarga de comprobar que los documentos existentes

en la *cache* están actualizados, por lo general, una vez por sesión. Es una *cache* especialmente útil cuando se usa el botón de “atrás” de un navegador o se trata de acceder a páginas recientemente visitadas. Trata de aprovechar la localidad en las peticiones del usuario para reducir el tráfico en la red y el tiempo de respuesta. Dentro de ellas se distinguen dos tipos:

- Persistentes: mantiene los documentos en memoria ante distintas invocaciones del navegador Web.
 - No persistentes: libera la memoria usada cuando el usuario cierra el navegador.
- *Cache proxy*: trabajan siguiendo el mismo principio que las *cache* de navegador pero a una escala mucho mayor puesto que se encuentran en servidores *proxy* que pueden dar servicio a cientos o miles de usuarios, de ahí que las *cache proxy* sean *caches* compartidas. Permiten ahorro de ancho de banda y latencia, pero tienen el riesgo de ocasionar cuellos de botella.
 - *Cache* de pasarela: al igual que las *caches proxy*, son intermediarias, pero en lugar de ser usadas por los administradores de las redes para ahorrar ancho de banda son típicamente usadas por los propios gestores Web para hacer sus páginas más escalables, fiables y que proporcionen un mejor servicio, para lo cual almacenan en memoria principal los documentos más solicitados para de esta forma, reducir los accesos a disco.

2.5.2 TOPOLOGÍAS DE LAS *CACHES*

Las principales topologías a la hora de organizar las memorias *cache* son:

- Distribuida: el contenido de la *cache* se distribuye entre una serie de servidores *proxy* todos conectados entre sí en una red. El grupo de servidores coopera en la creación de un sistema de almacenamiento robusto y eficiente, con la capacidad de compartir carga. Este sistema funciona bien incluso si alguno de los servidores falla y el flujo de datos hacia los distintos usuarios no se detiene. Con esta tecnología, la búsqueda de información en la *cache* es bastante rápida.

- Jerárquica: como su nombre indica, sigue una estructura jerárquica, es decir, las peticiones del usuario son primero procesadas en un *proxy* local y si no se encuentra la petición solicitada en éste, se dirige al siguiente servidor *proxy* de acuerdo a la jerarquía que se encuentre establecida y siendo, habitualmente, el flujo de peticiones, en sentido ascendente. El siguiente servidor al que se dirija la petición puede encontrarse a su mismo nivel de jerarquía o a un nivel superior, según la estructura existente, y en el caso de que exista más de un acierto hay formas de decidir qué servidor se prefiere, tomándose la información requerida de él. En el caso de no producirse acierto en ninguna de las *caches* de la jerarquía, la petición se enviaría al servidor original.

2.5.3 CACHE PROXY

La principal característica de la *cache* de un servidor *proxy* es su habilidad para almacenar documentos que puedan ser solicitados para su uso posterior, con lo que se consigue un importante ahorro en tiempo y ancho de banda. Las *caches proxy* además presentan otra amplia gama de características adicionales, muy valoradas por muchas organizaciones y la mayoría de las cuales están asociadas únicamente a su uso conjuntamente con un servidor *proxy* pero que tienen relativamente poco que ver con el propio almacenamiento en cache:

- Autenticación: un servidor *proxy* puede pedir a los usuarios que se autenticuen antes de servirles documentos. Esto es especialmente útil para cortafuegos pues si cada usuario autorizado tiene un nombre de usuario y contraseña, sólo éstos podrán acceder a la Web desde dentro de esa red privada.
- Filtrado de peticiones: los servidores *proxy* que disponen de almacenamiento en *cache*, pueden emplearse para realizar el filtrado de las peticiones de los usuarios. Los servidores *proxy* pueden ser configurados para denegar peticiones de los usuarios a contenidos fuera de los permitidos por la política de cada organización.

- Filtrado de respuestas: de igual modo que con las peticiones, se puede hacer un filtrado de las respuestas recibidas. Ello habitualmente implica el control de los contenidos de los documentos descargados, como por ejemplo para realizar chequeos contra ataques de virus.
- Hacer descargas por adelantado: este proceso consiste en traerse imágenes o enlaces referenciados en un fichero HTML (*HypeText Markup Language* – lenguaje de marcado de hipertexto) antes de que sean pedidos, suponiendo que van a ser solicitados por el usuario en breve. Una predicción correcta supone una reducción de la latencia pero si es incorrecta, se malgasta ancho de banda.
- Traducción y *transcoding*: ambos se refieren a procesos que cambian el contenido de algún objeto sin que con ello exista un cambio significativo de su significado o apariencia. Un ejemplo de traducción puede ser una aplicación que realice la traducción de una página de texto de un idioma a otro conforme es descargada. El segundo proceso supone cambios a bajo nivel en los datos digitales. Un ejemplo de esto podría ser pasar una imagen de formato GIF (*Graphics Interchange Format*) a JPEG (*Joint Photographic Experts Group*) porque en el formato JPEG los documentos resultan más pequeños y con ello se puede llevar a cabo una transferencia de forma más rápida.
- Modelado de tráfico (*Traffic shaping*): llevar a cabo un control del ancho de banda usado, empleando para ello el servidor *proxy*, pues de esta forma se puede conseguir información adicional que los administradores de red encuentran útil.

A pesar de todos los beneficios derivados del uso de *caches proxy*, existen una serie de aspectos y consecuencias a tener en cuenta a la hora de usar una *cache* Web:

- En primer lugar puede ser difícil garantizar la consistencia para una *cache* Web. La *cache* podría devolver documentos no actualizados a los usuarios ya que los servidores Web no suelen dar mucha información acerca de la frescura de los documentos y en algunos casos no se da información alguna. Puesto que la *cache proxy* ha de devolver documentos actualizados a los usuarios, puede que la demanda de validación sea la única forma para conseguirlo y ello implica unas pérdidas de tiempo relativamente grandes en

comparación con otros sistemas. Además, la petición de validación puede no alcanzar al servidor remoto por un fallo en la red o el servidor, con lo que la *cache* no sabrá realmente si el documento almacenado en ella está actualizado o no.

- Por otro lado, el uso de cache hace más complejo saber qué usuarios han visitado qué páginas y con qué frecuencia lo han hecho. Los aciertos en *cache* no se registran en el servidor origen y los servidores *proxy* tienden a ocultar la identidad de los usuarios, pues todos los usuarios conectados a un servidor *proxy* tendrán la misma dirección IP. De esta forma los suministradores de contenidos no pueden realizar un adecuado análisis de los accesos a sus páginas Web.
- El copyright ha supuesto también, durante bastante tiempo, un problema al uso de las *caches*, porque éstas no permitían al autor de un determinado documento controlar la distribución de su trabajo. HTTP permite especificar a los autores cómo han de ser manejados y distribuidos sus documentos por los distintos tipos de *caches* aunque el protocolo no trata el copyright de una forma directa.
- Existe cierta creencia que en el futuro el contenido Web será más dinámico y personalizado [66]. Estas respuestas dinámicas, normalmente no se deben almacenar en *cache* porque no se pueden reutilizar en peticiones futuras. Si la predicción es cierta, el uso de *cache* será menos importante con el paso del tiempo. Otros autores, por el contrario, creen que los contenidos cada vez son más estáticos y será más fácil diferenciar información estática y dinámica, en este caso el uso de cache sería cada vez más efectivo.

Por tanto, existe una lucha por el control de los contenidos Web, donde usuarios y proveedores de servicio quieren altos porcentajes de acierto en *cache* porque así ahorran tiempo y ancho de banda, mientras que los autores de contenidos quieren menos aciertos de las *cache proxy* para controlar mejor la distribución de sus documentos y llevar a cabo una cuenta más exacta de los accesos a sus páginas. A pesar de estos problemas, las ventajas proporcionadas por el uso de *caches* se presentan como más determinantes, sin ignorar, por supuesto, todas estas problemáticas asociadas.

2.6 Ley de Zipf

En su libro *Human Behavior & The Principle of Least Effort*, George K. Zipf define la esencia de lo que posteriormente llegó a ser conocida como su ley, llamada Principio del Mínimo esfuerzo, como “el principio primario que gobierna nuestro comportamiento individual y colectivo, incluyendo el comportamiento de nuestro lenguaje y prejuicios”[17]. Zipf dice que el principal “predictor” del comportamiento humano es siempre aquel que intente minimizar nuestro esfuerzo. Por tanto, el trabajo de Zipf se aplica a casi todos los campos donde la tarea humana esté involucrada. Zipf apunta en su principio que una persona intentará resolver su problema inmediato y su “posible problema futuro”, también aplica su principio a otros aspectos cotidianos; desde la distribución del tamaño de población entre ciudades y el número de periódicos; y el balanceo entre herramientas y trabajo.

Básicamente, la Ley de Zipf predice el hecho que usamos palabras familiares con alta frecuencia, de modo que minimizamos el esfuerzo en recordar o alternar palabras similares tendiendo a mantener el uso de las mismas palabras o frases a lo largo de un documento. Zipf prefirió explicar estos resultados empíricos como una condición humana, donde siempre es más fácil escribir una palabra conocida que usar una menos conocida. Fenómenos similares aparecen en otros ámbitos como el número de citas bibliográficas a un artículo dado o las poblaciones de ciudades. Diversos autores, entre ellos Mandelbrot y Millar, argumentaron más tarde que en realidad la ley de Zipf representa la consecuencia de las leyes de las probabilidades en procesos asociados a codificación de información donde hay mucho de azar. Sin querer tomar partido en esta disputa científica, cierta o no cierta, la ley de Zipf aparece frecuentemente en la práctica y refleja bien la actitud natural de minimizar el esfuerzo, exceptuando los casos extremos, que serían en el ejemplo inicial, usar muy pocas palabras o usar muchas. Tal vez esta ley sólo explica la diversidad humana, que se inclina más por la pereza que por la erudición.

Originalmente, la ley de Zipf era aplicada a la popularidad de las palabras que relacionaba su frecuencia de uso y su ranking (cuando eran ordenados según esa

frecuencia de uso)². Eso hacía que si ordenábamos la popularidad de las palabras usadas en un texto (ρ), según su frecuencia de uso (P) entonces:

$$P \approx \frac{1}{\rho} \quad [1]$$

A partir de esto, muchos investigadores han observado que la frecuencia relativa con la que las páginas web son solicitadas, siguen una distribución de Zipf [3][4][15]. Y es que la ley de Zipf expresa una relación exponencial entre la popularidad (P) de un elemento, es decir, el número de veces que se solicita una determinada página o documento, y su rango r , esto es, una relación ordenada de páginas o documentos según su número de repeticiones. Esta relación es de la forma:

$$P = \frac{C}{r^\beta} \quad [2]$$

Donde C es una constante, y β suele ser cercano a 1.

En el contexto Web, algunos investigadores han encontrado que el valor de β es cercano a 1 [4],[24], precisamente siguiendo la ley de Zipf. Otras fuentes [3][15][31] han encontrado que el valor de β es menor que 1, y que la distribución puede ser descrita sólo “como la de Zipf”, con el valor de β variando según la muestra de tráfico. Los datos empíricos de distintas simulaciones muestran como valores típicos de un web proxy β menor que 1 y para pequeños servidores web $\beta \approx 1$ [referencia].

El comportamiento típico de una distribución “como la de Zipf”, resulta una línea recta de pendiente β negativa en un gráfico logarítmico de la popularidad (P), frente al ranking ordenado según dicha popularidad (r), un ejemplo típico puede verse en la figura 2.8 con un valor de $\beta = 1$. Cuando comparamos una distribución “como la de Zipf” con la resultante de analizar las muestras de una traza de tráfico resulta totalmente coincidente para el cuerpo principal de la distribución, sin embargo, se desvía suavemente en ambos extremos, tanto en los documentos más populares, como en los menos populares (los “one-timers”, es decir, los referenciados una sola vez).

² La ley de Zipf ha sido aplicada también a otros ejemplos de popularidad en las ciencias sociales.

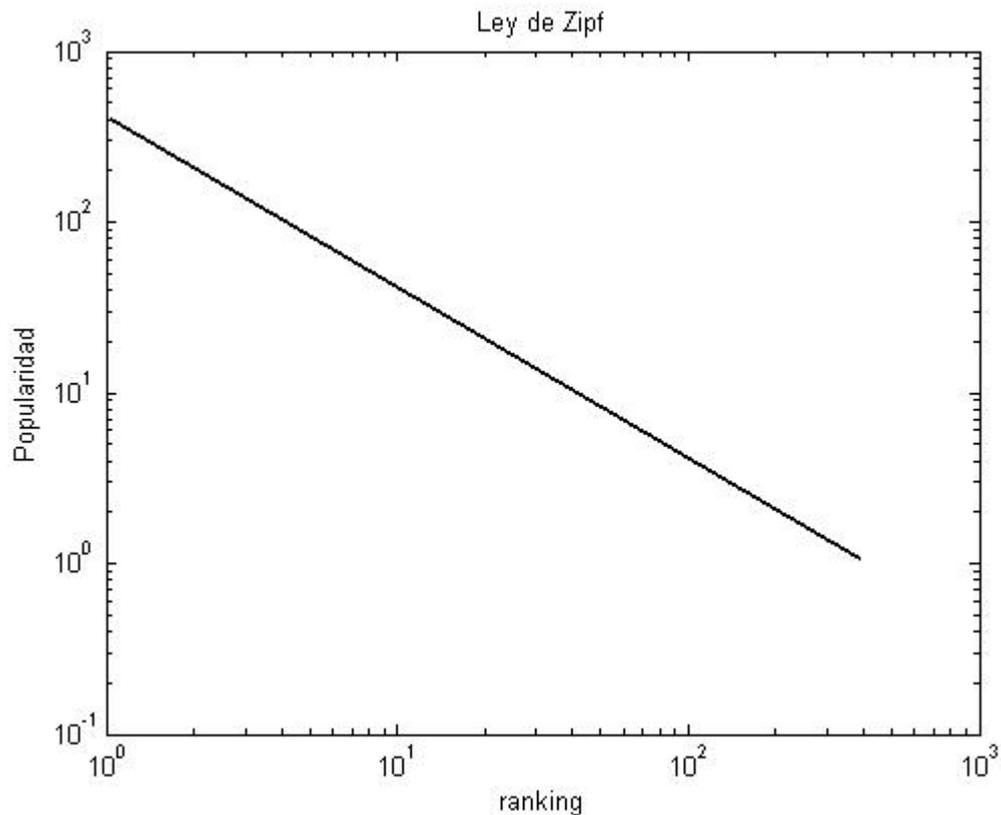


Figura 2.8. Distribución de Zipf en ejes logarítmicos

2.7 One – timers

Numerosos estudios de servidores Web y de Web *proxy*, han demostrado que muchas solicitudes de documentos a un servidor o a un proxy son efectuados una sola vez, independientemente de la duración del acceso [1],[8],[31], a este tipo de documentos se les llama “one-timers” en la bibliografía. Obviamente, no hay ningún beneficio en almacenar en cache estos “one-timers” ya que nunca más van a ser requeridos. De hecho, son necesarios algoritmos en la cache para discriminar este tipo de documentos de tal modo que no llenen la cache reduciendo así su efectividad [8].

Arlitt y Williamson [8], concluyeron que entre el 15 y el 40% de todos los archivos distintos accedidos desde un servidor Web son accedidos una sola vez, y éste número aumenta en el caso de usar servidores proxy a un 50-70%.

2.8 Distribución de Pareto

La distribución de la variable aleatoria X , recibe el nombre de distribución de Pareto de parámetros $c > 0$ y $\alpha > 0$ si $1 - F_X(x)$ ($P[X > x]$), decrece de forma potencial. Por ello se suele definir asociada a su función de distribución[referencia]:

$$F_X(x) = \begin{cases} 0 & \text{para } x < c \\ 1 - \frac{c^\alpha}{x^\alpha} & \text{para } x \geq c \end{cases} \quad \text{donde } c, \alpha > 0$$

La función de densidad:

$$f_X(x) = \begin{cases} 0 & \text{para } x < c \\ \frac{\alpha c^\alpha}{x^{\alpha+1}} & \text{para } x \geq c \end{cases}$$

Esta variable, que toma valores a partir de la constante c , no suele utilizarse para modelar cuantías de ocurrencias que sean elevadas, sin embargo, dado que es una distribución para la que la cola de la derecha (el comportamiento de la función de densidad para valores altos), es más pesada (converge a cero más lentamente que el resto de las distribuciones exponenciales), es utilizada para intentar explicar los valores altos de las ocurrencias.

En la figura 2.9 podemos ver un ejemplo típico de una distribución de Pareto.

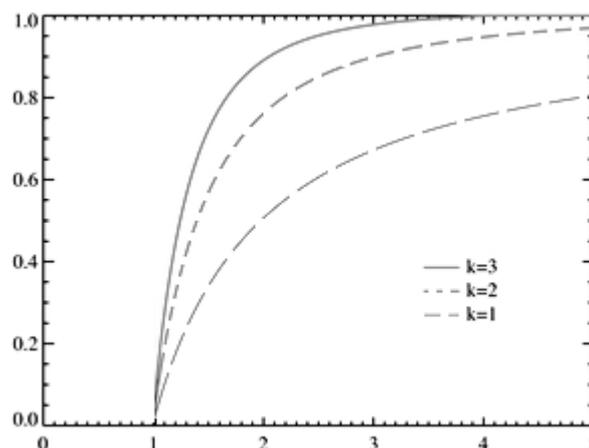


Figura 2.9. Función de distribución de Pareto

El principio de Pareto afirma que en todo grupo de elementos o factores que contribuyen a un mismo efecto, unos pocos son responsables de gran parte de dicho efecto, y por otro lado el efecto producido por una gran mayoría es escaso, dando lugar al efecto de “cola pesada”, que se puede apreciar claramente en la figura 2.10.

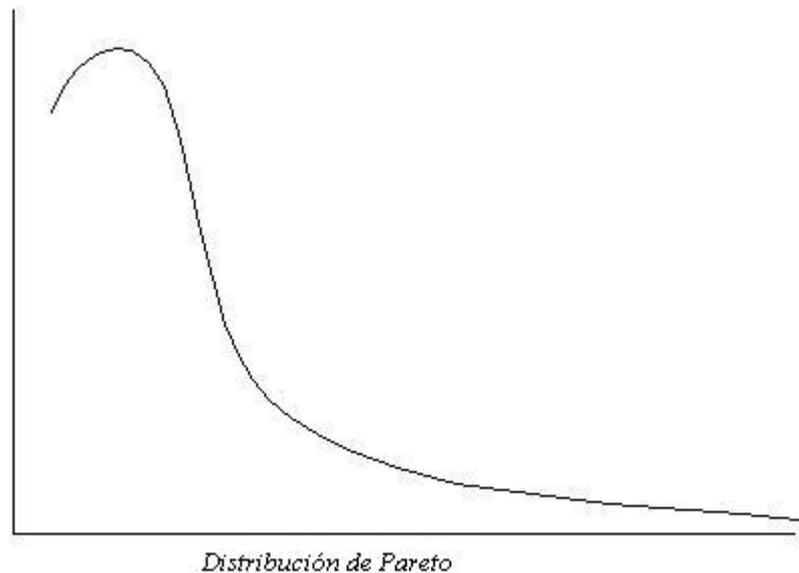


Figura 2.10. Cola pesada en una distribución de Pareto

2.9 Distribución logarítmico normal

La distribución de variable aleatoria X , recibe el nombre de logarítmico-normal de parámetros μ y σ , ($X \sim \text{LN}(\mu, \sigma)$), sigue una distribución $N(\mu, \sigma)$. Por tanto: $X = e^Y$, es una variable aleatoria positiva.

La función de distribución de esta variable:

$$F_X(x) = P[X \leq x] = P[\ln X \leq \ln x] = P[Y \leq \ln x] = F_Y(\ln x) \quad x \geq 0$$

Por lo que la función de densidad de X :

$$f_X(x) = \frac{d}{dx} F_X(x) = \frac{d}{dx} F_Y(\ln x) = F_Y'(\ln x) \frac{d}{dx} \ln x = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}, \quad x > 0$$

En las figuras 2.11 y 2.12 podemos observar las gráficas de la función de densidad de probabilidad y la función de distribución de una distribución logarítmica normal

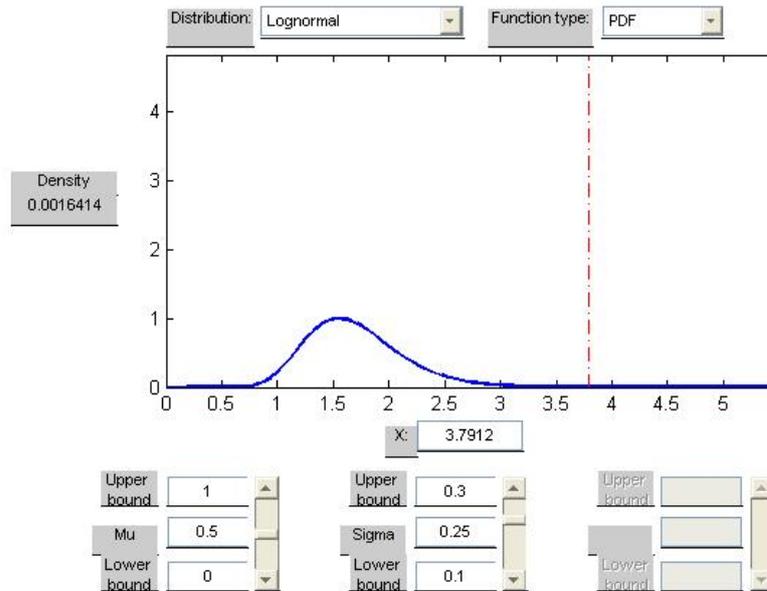


Figura 2.11. Densidad de probabilidad de una distribución logarítmica normal

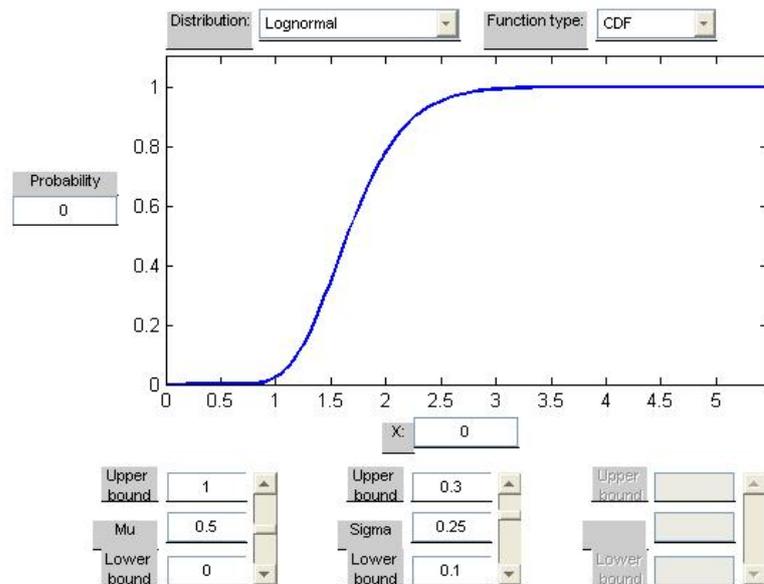


Figura 2.12. Función de distribución acumulada logarítmica normal

CAPÍTULO 3: Manual de usuario

Se ha creado una aplicación para la generación de muestras de tráfico sintético que diferencia entre distintos tipos de tráfico, estos tipos pueden ser: aplicaciones, audio, imágenes, mensaje, texto, y video.

Para una mejor descripción, se utilizarán capturas de pantalla de la aplicación, y de este modo, podremos realizar una explicación en profundidad detallando todas y cada una de las funcionalidades de la misma.

3.1. Ejecución

Para la elaboración de la aplicación así como para el diseño del interfaz gráfico se ha utilizado el entorno Matlab, utilizando para ello librerías y procedimientos propios del mismo.

Por tanto, para la ejecución de la aplicación, se puede realizar una vez que se ha introducido en dicho entorno, y se procede al lanzamiento del mismo. Para ello, se debe escribir el nombre de la aplicación “*nuevo*”, y pulsar la tecla *enter*. Antes de esto, se debe verificar que el directorio de trabajo es el mismo que donde se encuentra ubicada la aplicación, se puede cambiar dicho directorio de trabajo en la opción “*Current Directory*” mostrado en la figura 3.1.

Por otro lado, se ha conseguido compilar el código fuente de la aplicación independientemente de Matlab, de este modo se va a poder ejecutar la aplicación pulsando sobre el icono correspondiente.



Figura 3.1. Modificar el directorio de trabajo

3.2. Menú principal

Una vez ejecutada la aplicación sobre MATLAB o bien directamente pulsando en el icono correspondiente, al usuario se le muestra el menú principal en pantalla, tal y como se presenta en la figura 3.2, donde se pueden elegir las distintas opciones del menú.

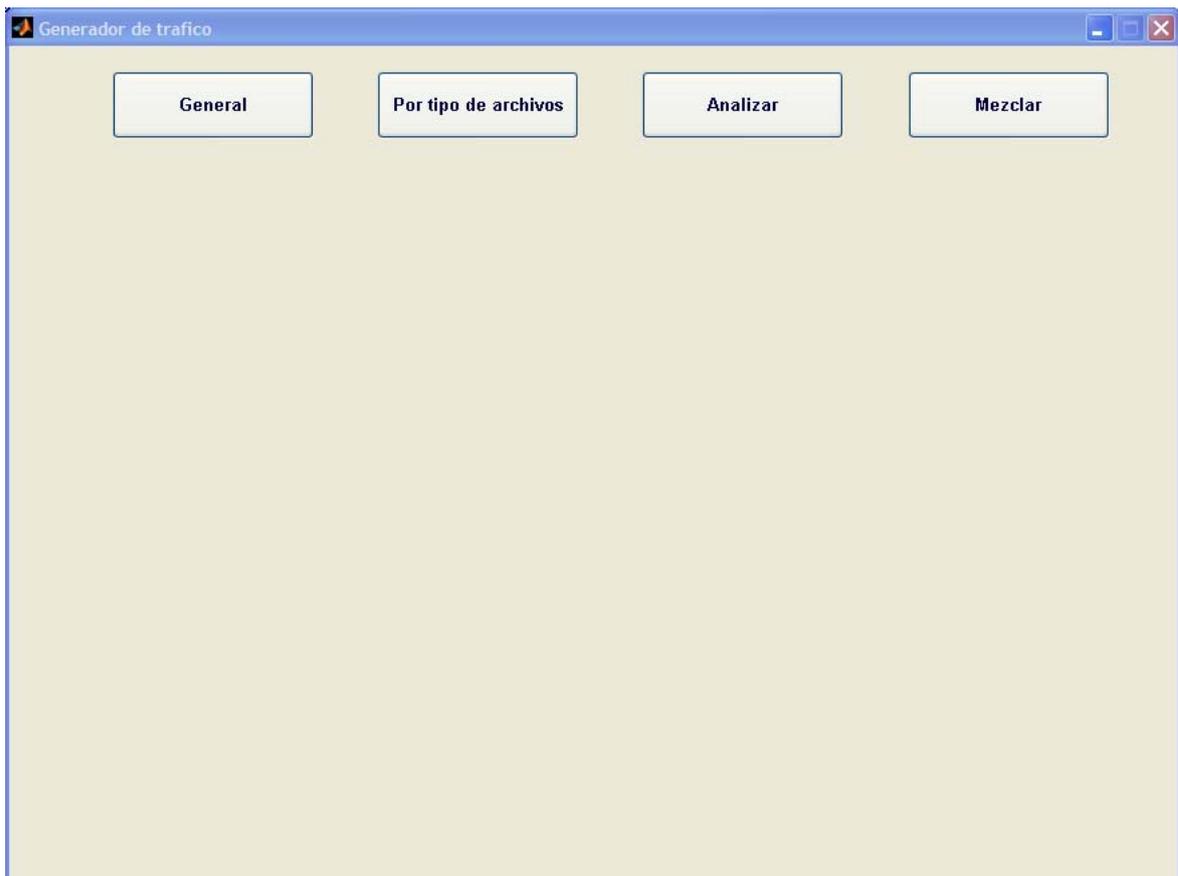


Figura 3.2. Pantalla principal de la aplicación

En este menú, se presenta 4 funciones principales:

- *General*: Esta es la opción más importante, ya que en ella es donde se van a introducir todos los parámetros necesarios para la generación de las muestras de tráfico sintético. También será necesario indicar el porcentaje de peticiones de archivos de los distintos tipos existentes, así como el nombre del archivo a generar.
- *Por tipos de archivos*: Este apartado es muy similar al anterior, la principal diferencia es que antes de introducir los parámetros para la generación de las muestras de tráfico, se deberá indicar de qué tipo de documentos serán las muestras que se van a generar, una vez elegido el tipo de documento, se introducirán los parámetros necesarios y el nombre del archivo a generar.
- *Analizar*: Esta opción, se ha introducido puesto que ha sido una herramienta clave en la elaboración de esta aplicación ya que nos ha permitido verificar el correcto funcionamiento del mismo. Nos muestra información relativa a las muestras que contenga el archivo seleccionado, como la Ley de Zipf, la función de probabilidad acumulada del cuerpo de la distribución de tamaño, la función de probabilidad acumulada de la cola de la distribución de tamaño, la correlación entre el tamaño de los documentos y su frecuencia de aparición, la función de probabilidad de los tamaños de los documentos y la localidad temporal de las repeticiones de las peticiones de los documentos comparadas con sus respectivas gráficas ideales.
- *Mezclar*: Esta herramienta nos permite poder mezclar muestras de distintos archivos. A priori, se realiza una mezcla aleatoria, dejando abierta la posibilidad de incluir cualquier otro tipo de distribución para dicha mezcla.

Independientemente de la opción que se haya elegido, este menú principal siempre será accesible desde cualquier pantalla. Por otro lado, una vez que se han introducido los parámetros, éstos permanecerán inalterados hasta que sean modificados por el propio usuario aunque se visiten otras opciones.

3.3 Menú “General”

Esta opción es, sin duda, la más importante. Con ella se podrán generar muestras de tráfico en función de los parámetros de entrada que hayamos indicado. Al pulsar en la opción “General”, se mostrará una pantalla que similar a la figura 3.3.

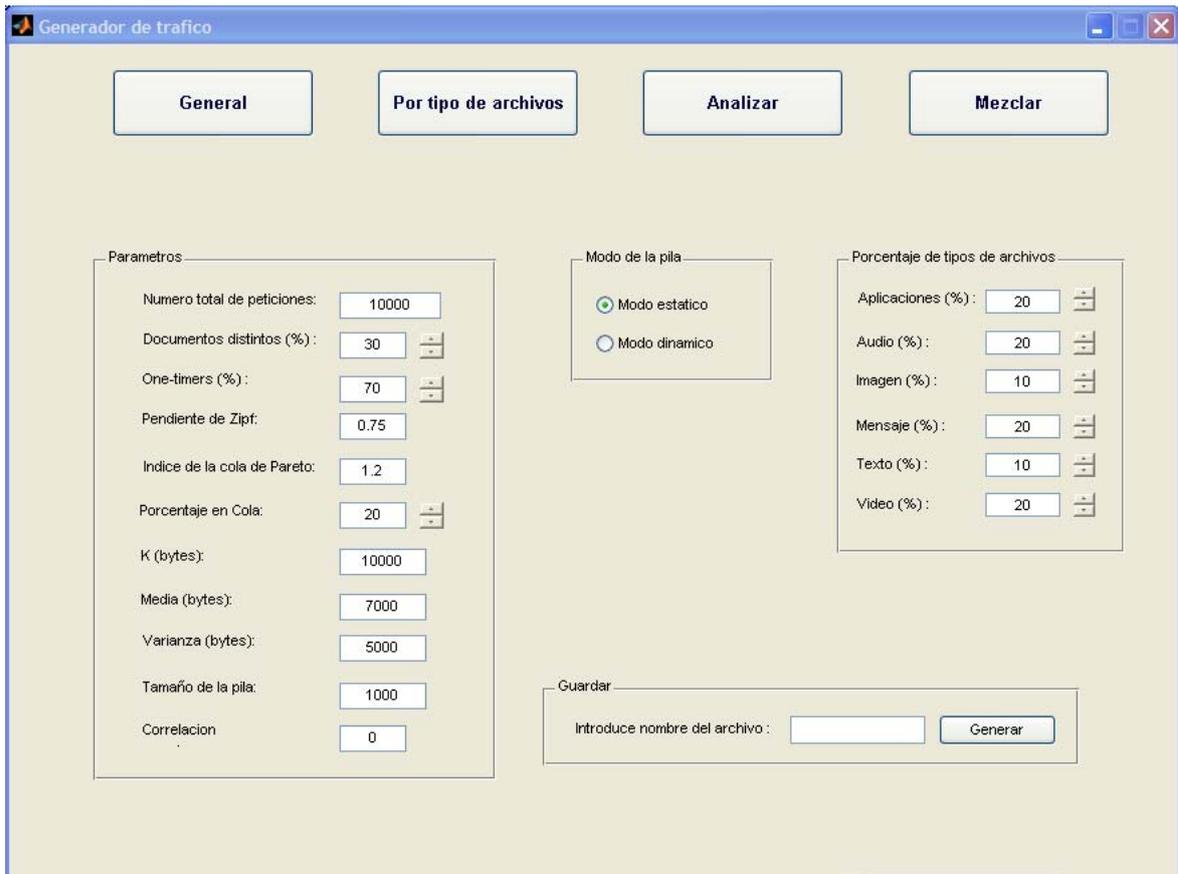


Figura 3.3. Pantalla del menú “General”

Como se comentó anteriormente, la estructura del Menú principal queda fija en la parte superior de la pantalla como puede verse en la figura 3.4.



Figura 3.4. La parte superior de la pantalla corresponde siempre al Menú Principal

El resto de la pantalla corresponde a los parámetros necesarios para la generación de muestras de tráfico. Esquemáticamente se ha dividido en 4 secciones:

- **Parámetros:** En este apartado se introducirán los parámetros más generales y sin duda, los que más caracterizan a las muestras para su posterior generación, es decir, son aquellas características intrínsecas de las muestras de tráfico. Todas estas características se muestran en la figura 3.5 y a continuación se explica brevemente cada una de estas características:

Parámetro	Valor
Numero total de peticiones:	10000
Documentos distintos (%):	30
One-timers (%):	70
Pendiente de Zipf:	0.75
Indice de la cola de Pareto:	1.2
Porcentaje en Cola:	20
K (bytes):	10000
Media (bytes):	7000
Varianza (bytes):	5000
Tamaño de la pila:	1000
Correlacion	0

Figura 3.5. Parámetros del menú “General”

♦ *Número total de peticiones:* En este campo se debe introducir el número de peticiones de documentos que se desea generar, independientemente de que los documentos se repitan o el tipo de documento.

♦ *Documentos distintos (%):* Con este parámetro se indica el porcentaje de documentos distintos sobre el total de documentos. Es decir, con este dato y el anterior se podrá conocer el número total de peticiones correspondientes a

documentos distintos. En este apartado se podrá introducir el dato manualmente, o bien incrementando/disminuyendo el valor marcado mediante cursores. Dicho valor debe pertenecer al rango de valores [0 - 100], en caso contrario, la aplicación mostrará un mensaje de error cuando se quiera generar las muestras tal y como se indica en la figura 3.6.



Figura 3.6. Mensaje de error por fuera de rango.

- ♦ *One-timers (%)*: Se introduce el porcentaje de documentos que son solicitados una sola vez, ese valor está referido al número total de peticiones correspondientes a documentos distintos que se calculó con los parámetros anteriores. En este caso, también se dispone de cursores para modificar el valor introducido. Como en el caso anterior, se deberá introducir un valor dentro del rango adecuado [0 – 100], en caso contrario se mostrará nuevamente un mensaje de error tal y como se mostraba en la figura 3.6.

- ♦ *Pendiente de Zipf*: Este valor corresponde al valor absoluto de la pendiente de la distribución de Zipf en unos ejes logarítmicos que se utilizarán para el estudio de la popularidad.

- ♦ *Índice de la cola de Pareto*: Muestra el valor absoluto de la pendiente de la distribución de Pareto que posteriormente se utilizará para generar los tamaños de los archivos.

- ♦ *Porcentaje en cola (%)*: Nos indica el porcentaje de documentos distintos cuyos tamaños se van a encontrar en la “cola” de Pareto. El valor especificado debe estar dentro del rango correcto [0 - 100], en caso contrario, se mostrará un mensaje de error como el de la figura 3.6.

♦ *K (Bytes)*: Indica el tamaño en bytes a partir del cual, aquellos archivos cuyos tamaños excedan de este valor se situarán en la “cola” de Pareto.

♦ *Media (Bytes)*: Indica la media en una distribución logarítmica normal que se utilizará para modelar el tamaño de los documentos. Se mide en bytes.

♦ *Varianza (Bytes)*: Presenta el valor de la varianza en una distribución logarítmica normal que se usará para modelar el tamaño de los documentos. Se mide en bytes.

♦ *Tamaño de la pila*: En este campo se introducirá el valor del tamaño de la pila que posteriormente se usará para modelar la localidad espacial de las distintas solicitudes de documentos.

♦ *Correlación*: En este apartado se va a definir la correlación existente entre el tamaño de los distintos documentos y su popularidad, permitiendo la existencia de correlación positiva, negativa o nula.

• Modo de la pila: En esta división se presenta el modo de trabajo de la pila con el que se modela la localidad espacial de las peticiones de documentos y que puede verse en la figura 3.7. Para ello se hace uso de un botón excluyente, el cual permitirá elegir una sola de las opciones, siendo el modo estático el valor por defecto.

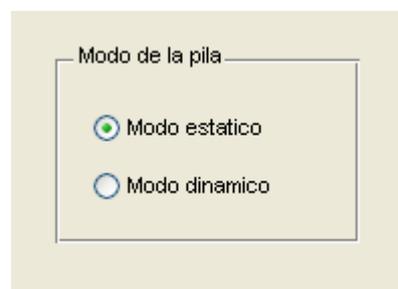


Figura 3.7. Modo de funcionamiento de la pila

• Porcentaje de tipos de archivos: En esta sección se define el porcentaje de los distintos tipos de documentos sobre el total de documentos distintos, esto es, antes de generar todas las repeticiones de solicitudes de documentos, se

define cuántos documentos son de un tipo o de otro. Para mejorar su manejo se introducen cursores para modificar los valores marcados. En todas estas casillas el rango de valores válidos debe estar comprendido entre 0 y 100, en caso contrario se mostrará error. Por otro lado, la suma total de los porcentajes de tipos de archivos debe ser 100, en caso contrario se mostrará un error similar al de la figura 3.8.



Figura 3.8. Error al no conseguir el 100% en su totalidad

- Guardar: En este apartado, se debe introducir el nombre del archivo a generar, este archivo será guardado en el mismo directorio donde se trabaja. La extensión por defecto que se asignará al archivo será *.txt*. Una vez que se ha introducido el nombre del archivo, al pulsar el botón “*Generar*”, véase figura 3.9, y tras un intervalo de tiempo, que dependerá del número de muestras a generar, se creará un documento con el nombre indicado que contendrá todas las muestras de tráfico de acuerdo con los parámetros introducidos en las secciones anteriores.



Figura 3.9. Guardamos el archivo y generamos muestras

3.4 Menú “*Por tipo de archivos*”

Esta opción, representada en la figura 3.10, es muy similar a la anterior pero con una peculiaridad importante, y es que se puede definir las características de tráfico para un determinado tipo de tráfico, es decir, se podrá asignar muestras de tráfico que sean tipo audio, video, texto,.... o cualquier otro tipo.



Figura 3.10. Menú “por tipo de archivos”

Una vez que se ha pulsado en el botón “*Por tipo de archivos*” aparece una nueva división, en donde se podrá elegir el tipo de archivo cuyas muestras serán generadas, véase figura 3.11.



Figura 3.11. Se puede elegir de qué tipo de archivo se generan muestras

Una vez elegido el tipo de archivo de las muestras a generar, aparecerán las mismas opciones, para la generación de la muestras en sí, que el menú “*General*”, con la diferencia fundamental, que en los encabezados de cada sección aparece el nombre del tipo de tráfico que se ha seleccionado. A continuación se muestra en la figura 3.12 la pantalla correspondiente al seleccionar uno de los distintos tipos de documentos que se pueden generar.

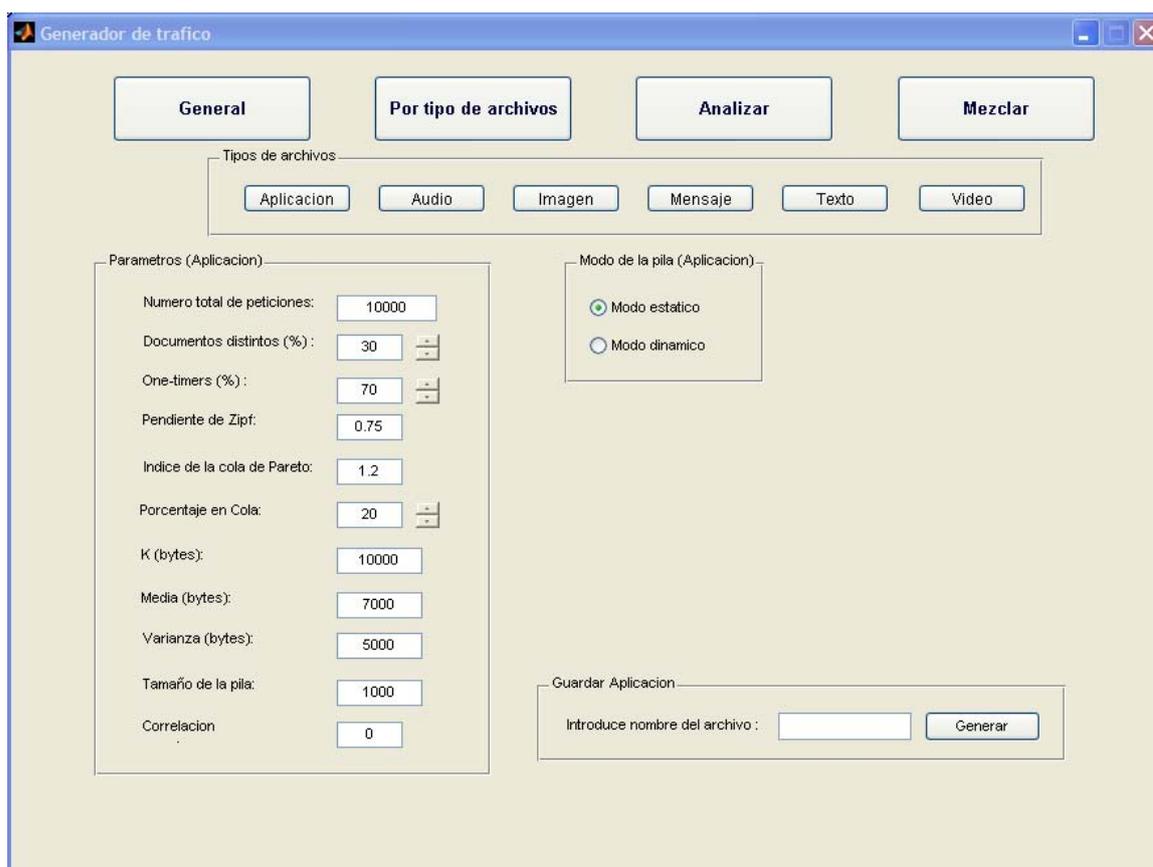


Figura 3.12. Generación de tráfico tipo “*Aplicación*”

El funcionamiento y significado de cada parámetro ya se explicó en la sección 3.3, por lo que para cualquier consulta remítase a dicha sección.

3.5. Menú “*Analizar*”

Esta opción surge como herramienta necesaria para analizar los resultados que se han ido generando a lo largo de la implementación de la aplicación, y se ha integrado en la misma por aportar facilidades al usuario para una correcta interpretación de los resultados obtenidos y de las muestras generadas. Al pulsar sobre el botón de “*Analizar*”, se visualiza una nueva sección, véase la figura 3.13, donde o bien se puede escribir directamente en el cuadro de texto habilitado para ello el nombre del archivo a analizar, o bien pulsar el botón de “Examinar” abriéndose en tal caso una ventana donde se podrá elegir entre los distintos archivos de la carpeta por defecto o incluso navegar entre todo el directorio de carpetas con el objetivo de elegir el archivo deseado, tal y como se muestra en la figura 3.14.

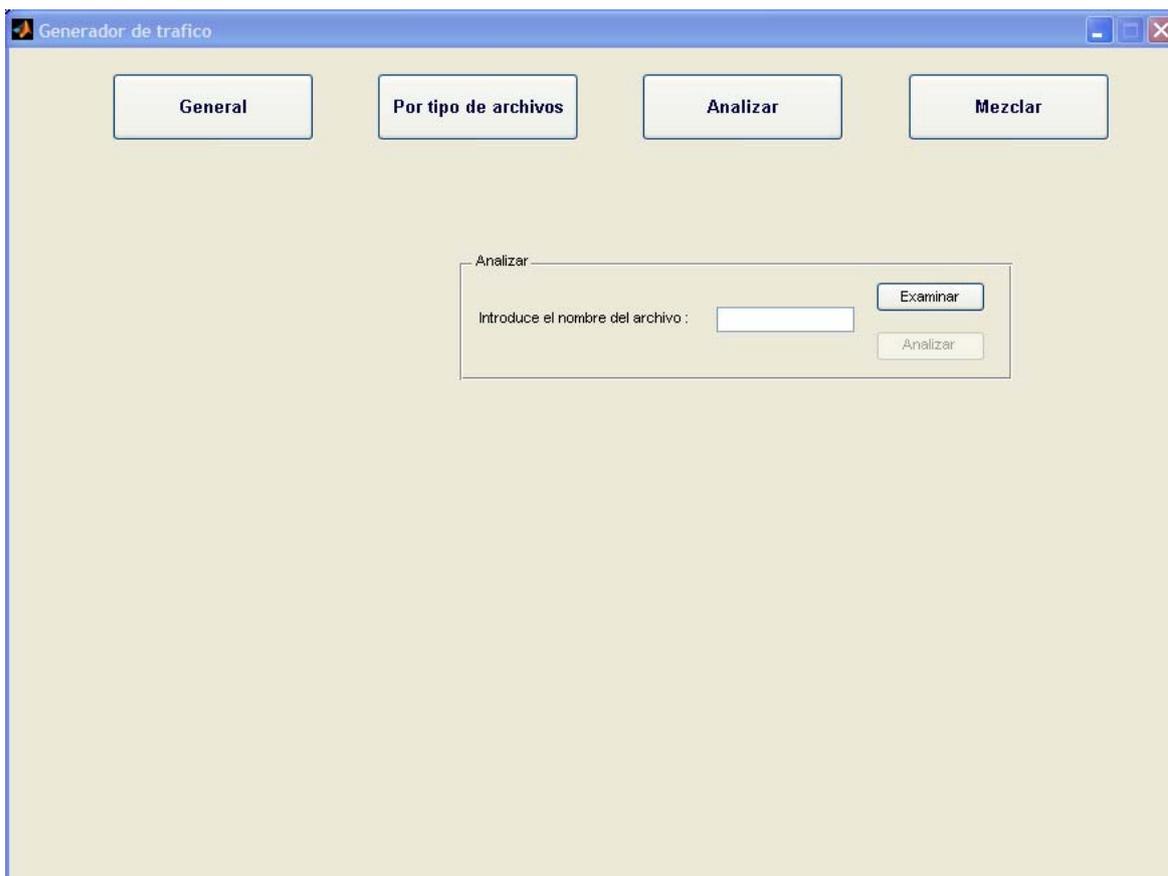


Figura 3.13. Opción “*Analizar*”

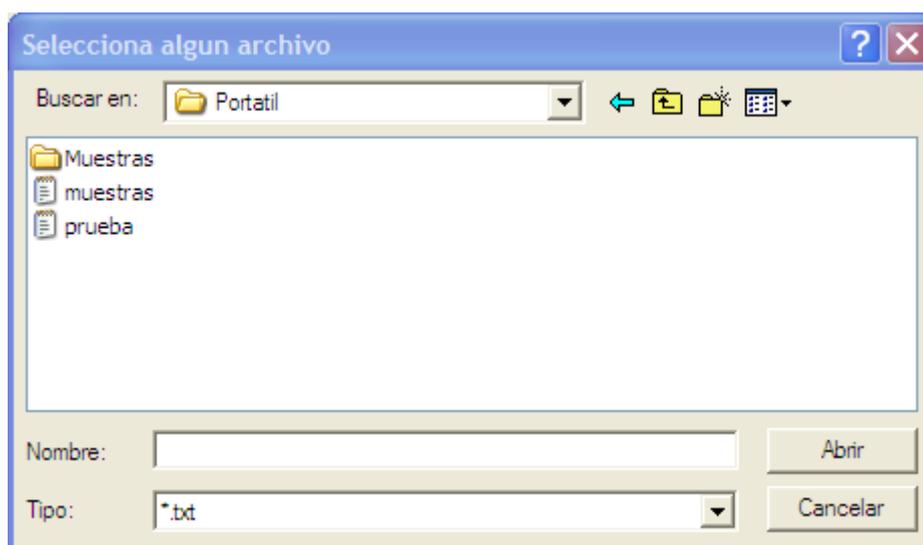


Figura 3.14. Selección de cualquier archivo

Una vez elegido el archivo a analizar, al pulsar dicho botón, se lanzan un conjunto de rutinas que muestran en pantalla una serie de datos relevantes procedentes del análisis del archivo correspondiente como: el número total de peticiones, el número total de objetos distintos, el porcentaje de documentos distintos, el número de *one – timers*, así como el número de documentos correspondientes a los diferentes tipos de archivos junto con su tamaño medio. Por último, se muestra el porcentaje de archivos incluidos en la distribución de Cola de Pareto, así como la pendiente de dicha cola cuando la representamos en ejes logarítmicos.

A continuación se muestra una figura con 6 representaciones gráficas, véase figura 3.15.

La primera figura (esquina superior izquierda), muestra la distribución de Zipf de todos los documentos, es decir, se han ordenado las popularidades de los distintos documentos y se ha representando en unos ejes logarítmicos, pudiendo observar que las muestras de tráfico generada tienen un comportamiento totalmente modelable por la distribución de Zipf.

En esta primera gráfica, se pueden comparar las leves diferencias existentes entre la representación real de la muestras y la representación ideal de una distribución de Zipf.

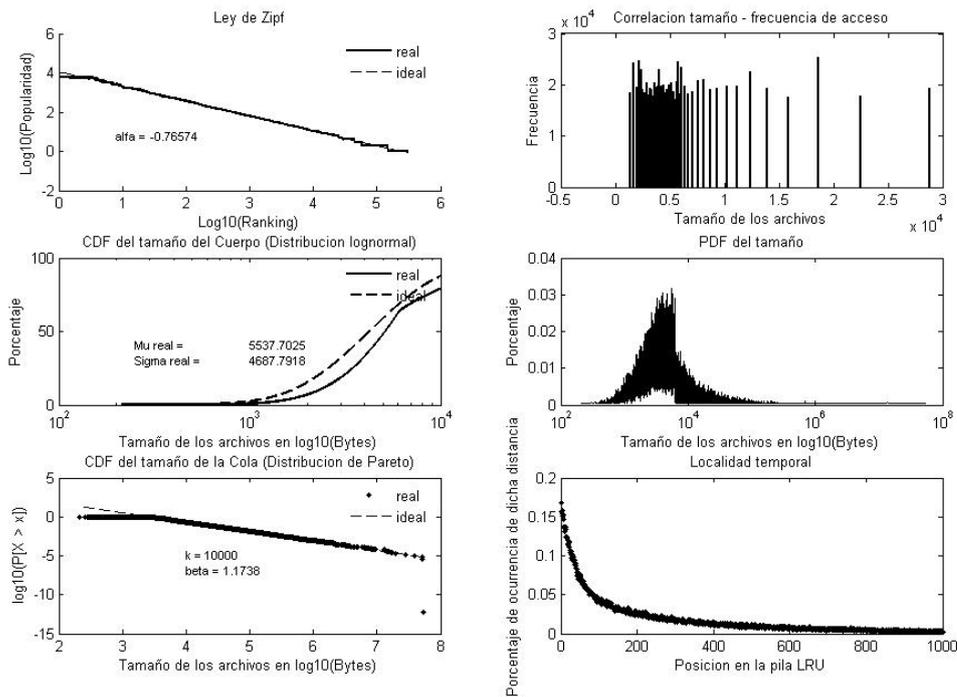


Figura 3.15. Análisis de muestras de tráfico

En la segunda gráfica (esquina superior derecha), se presenta como la correlación de las muestras es cero, y es que todos los tamaños de los archivos tienen aproximadamente el mismo peso sobre el total de los tamaños.

La tercera gráfica muestra la función densidad de probabilidad acumulada del tamaño de los archivos ubicados en el “cuerpo” del conjunto total de tamaños, es decir, esta es la distribución de tamaños de aquellos archivos cuyo tamaño es menor al indicado por el parámetro k .

La cuarta gráfica, presenta la función densidad de probabilidad de todos los tamaños de los distintos archivos, y viene a mostrar los distintos tamaños que toman los diferentes documentos.

La quinta gráfica, representa un concepto similar al visto en la tercera gráfica, y es la función densidad de probabilidad acumulada del tamaño de los archivos ubicados en la “cola” del conjunto total de tamaños, esto es, aquellos tamaños mayores que los indicados por el parámetro k , es a lo que llamamos “cola de Pareto”.

La última gráfica (esquina inferior derecha), indica la localidad temporal que se pone de manifiesto en las referencias a los mismos documentos dentro de la globalidad del total de peticiones.

Todas estas gráficas suponen una herramienta muy útil para poder comparar las características reales de las muestras generadas con las características ideales de las mismas.

3.6. Menú “*Mezclar*”

La última opción que falta por describir de este menú principal es la de “Mezclar”, en este apartado se mezcla en un mismo documento todas aquellos archivos de muestras generadas con distintos, o los mismos, parámetros. La metodología a seguir una vez pulsado el botón de “Mezclar” sería la siguiente, ver figura 3.21.

En primer lugar, se debe escribir el nombre del archivo resultante de mezclar varios archivos, éste archivo se creará por defecto con la extensión “.txt”. Para elegir los distintos archivos a mezclar se debe pulsar el botón “Añadir”, en tal caso, se abrirá un ventana similar a la figura 3.19, donde se podrá elegir el archivo deseado, si se desea elegir varios archivos se podrá hacer pulsando simultáneamente la tecla CTRL y, sin soltar la tecla, en los distintos archivos que a elegir, si lo que se desea es elegir una lista de archivos consecutivos se puede pinchar en el primer archivo de la lista y pulsando la tecla SHIFT y el último elemento de la lista se seleccionarán automáticamente los dichos archivos. Este proceso lo se podrá repetir todas las veces que se desee. Los archivos elegidos para su mezcla se indican en pantalla en un panel habilitado para ello.

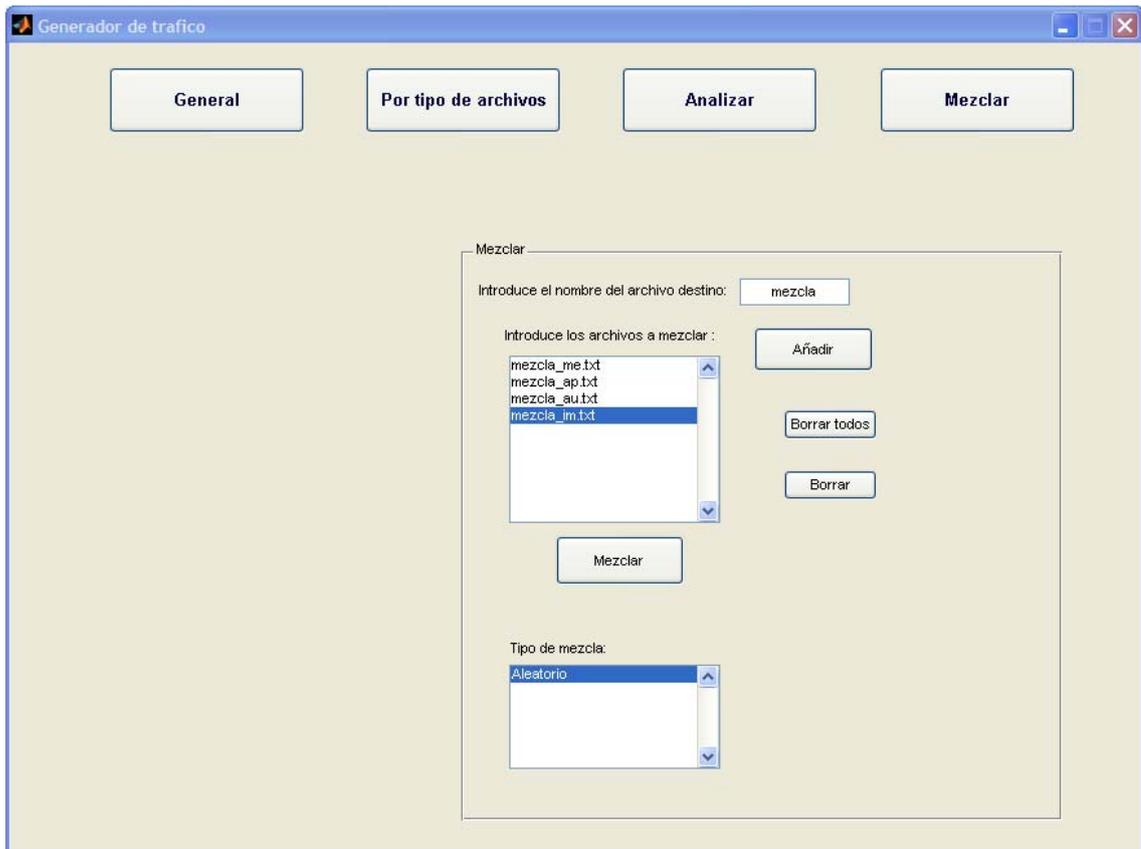


Figura 3.16. Menú “Mezclar”

Una vez escrito el nombre del archivo resultante así como elegido algunos archivos para su mezcla, se activará el botón de “Mezclar”, pudiendo realizar ya dicho proceso.

También se disponen de las siguientes opciones:

- “Borrar todos”: Esta opción elimina todos los archivos seleccionados para mezclar. El panel de archivos seleccionados se quedará vacío.
- “Borrar”: Con esta opción se puede eliminar el archivo que se desee de entre todos los elegidos para la mezcla. Para ello se elige el archivo a eliminar pulsando sobre él en el panel de archivos seleccionados y pulsando el botón de “Borrar”.

CAPÍTULO 4: Implementación

La aplicación ha sido generada sobre el entorno MATLAB y tiene como objetivo la generación de muestras de tráfico para su posterior procesamiento en simuladores de cachés Web. El resultado de la aplicación es la creación de ficheros donde se especifican estas muestras de tráfico que serán los que utilicen posteriormente estos simuladores.

La implementación de la aplicación se divide en dos partes, la primera es la generación en sí de las muestras de tráfico a partir de una serie de parámetros, y la segunda es la interfaz gráfica con el usuario donde se interactúa en la elección de los parámetros deseados.

Esta aplicación utiliza algoritmos matemáticos para modelar las características del tráfico y su principal objetivo es generar muestras de tráfico para la posterior evaluación de técnicas de uso de caché, sólo aquellas características que se han considerado como relevantes para la caché se han introducido en la generación de muestras. Este enfoque ha conseguido reducir la complejidad de los modelos, así como el tiempo y el espacio requerido para la generación y almacenaje de las muestras de tráfico.

4.1. Descripción de la aplicación

Hay dos posibles enfoques para sintetizar muestras de tráfico [9]. La primera de ellas consiste en muestrear o permutar muestras de tráfico reales reordenando las peticiones de modo que se genere una nueva carga de tráfico diferente, de algún modo, con la carga de tráfico original, este enfoque se llama *basada en traza real*. La segunda aproximación, llamada aproximación analítica, usa modelos matemáticos para las características más interesantes de tráfico Web, y utiliza la generación de números aleatorios para producir tráfico que estadísticamente coincide con dichos modelos.

Mientras el modelo basado en traza real, es bastante sencillo de implementar, tiene la limitación que el tráfico generado está inherentemente vinculado a un sistema conocido. Por otro lado, la aproximación analítica ofrece flexibilidad: cada modelo matemático puede tener uno o más parámetros para regular las características del tráfico a generar, por tanto, mucho más tráfico puede ser generado en un periodo corto de tiempo, en contra, de la necesidad de almacenar en primer lugar las muestras de tráfico real y posteriormente procesarlas como ocurre en la primera aproximación.

Por tanto, se ha elegido el enfoque analítico como aproximación para la generación de muestras de tráfico. El generador de tráfico incorpora 5 características de tráfico seleccionadas, las cuales han sido identificadas como importantes en estudios previos de muestras de tráfico en servidores [1],[3],[6],[13].

Las características de tráfico modeladas en la aplicación son:

- Documentos una sola vez referenciados (*one-timers*).
- Popularidad de los documentos.
- Distribución del tamaño de los archivos.
- Correlación entre el tamaño de los archivos y su popularidad.
- Localidad temporal.

Cada característica es modelada usando uno o más parámetros controlables, así que la generación de muestras con distintos grados de variación puede ser realizada con facilidad. A continuación se muestra cómo se ha implementado cada una de estas características así como una breve introducción a las mismas.

4.2. “One-timers”

4.2.1. INTRODUCCIÓN

Muchos estudios de tráfico Web en servidores y proxy han mostrado que la mayoría de las peticiones desde un servidor o un proxy son realizadas una sola vez,

independientemente de la duración del acceso [3],[8]. A este tipo de documentos se les denomina *one-timers* en la bibliografía. Es evidente, que no tiene ningún beneficio almacenar en caché este tipo de documentos puesto que nunca más van a ser requeridos, de hecho, sería deseable la existencia de algoritmos ubicados en las cachés para discriminar a estos documentos de modo que no inunden la caché reduciendo así su efectividad.

El estudio de los *one-timers* es importante por su prevalencia en el tráfico Web. Arlitt y Williamson [8], sostienen que entre el 15 y el 40% de los accesos a archivos distintos desde un servidor Web se realizan una sola vez. La situación se tuerce peor si hablamos de un Web proxy donde los *one-timers* pueden alcanzar entre el 50 y 70% del total de documentos [1].[3]. Es más, en una estructura jerárquica de proxies, el porcentaje de *one-timers* tiende a crecer por encima del 50% a medida que vamos subiendo por los niveles altos de la estructura, de tal modo que los archivos cacheados en los niveles más bajos puede ser que no vuelvan a ser requeridos nunca más de los niveles superiores [18].

Por todo esto, es importante la realización de un buen modelo que se ajuste a las características de los *one-timers*, afortunadamente la implementación de este modelo es relativamente sencillo.

4.2.2. IMPLEMENTACIÓN

El enfoque utilizado para modelar a las referencias *one-timers* es determinar cuántas de los documentos distintos en la traza de tráfico a crear deberían ser *one-timers*. Utilizando el porcentaje de *one-timers* como un parámetro, se permite al usuario especificar el valor deseado. Una vez especificado, se puede hallar con facilidad el número de *one-timers* y entonces las referencias a estos archivos pueden ser ya fijadas a 1 (recordar que los *one-timers* son aquellos documentos que se referencian una sola vez).

El algoritmo utilizado para producir una carga de tráfico sintético con el número deseado de *one-timers* se muestra en la figura 4.1.

Por ejemplo, suponiendo que $N = 1000$ peticiones a generar en 200 archivos distintos ($c = 20\%$), la mitad de los cuales son *one-timers* ($d = 50\%$), la aplicación calcula $a = 200$ y $b = 100$.

Dado:

- N, el número total de peticiones a generar
- c, el porcentaje de archivos distintos (relativos a N)
- d, el porcentaje de *one-timers* (relativos a c)

Entonces con estos valores se puede calcular

- a, el número de archivos distintos como $a = \frac{c}{100} \times N$
- b, el número de *one-timers* como $b = \frac{d}{100} \times a$

Ahora se debe asignar a 1 la popularidad de cada uno de estos b archivos *one-timers*.

Figura 4.1. Algoritmo para la generación del modelo de *one-timers*.

Puesto que cada documento tiene asignado un único número de referencias a dicho documento, si se ordenan estos documentos por popularidad, el archivo más popular (es decir, el archivo con mayor número de referencias) ocupará el primer puesto del ranking, el rango de esta ordenación varía entre 1 y a (el número total de documentos distintos). Puesto que b es el número de *one-timers* entre el total de los a documentos distintos, entonces todos aquellos documentos comprendidos entre $a - b + 1$ hasta a tiene un número de referencias igual a 1.

4.3. Popularidad

4.3.1 INTRODUCCIÓN

Una característica común en el tráfico Web es la muy irregular distribución de las referencias a los distintos documentos [2],[3],[4]. En muchos casos se aplica la ley de Zipf [17] para modelar la popularidad de los archivos [7],[10],[11]. La ley de Zipf expresa una relación exponencial entre la popularidad P (número de veces que se repite la petición de dicho documento) y su ranking r (que se extrae de la ordenación de los documentos según su popularidad). Esta relación es de la forma:

$$P = \frac{c}{r^\beta} \quad \text{donde: } c \text{ es una constante}$$

β suele estar cercano a 1

El ejemplo más claro de aplicación de la ley de Zipf es la frecuencia de ocurrencia de las palabras en inglés [17]; cuando el número de ocurrencias es representado frente al ranking de ordenación el resultado es una función exponencial con el exponente cercano a 1.

En el contexto Web, el comportamiento de las referencias sigue un comportamiento similar a la ley de Zipf [2],[7],[11], es decir, si los documentos Web son ordenados r de acuerdo con su popularidad P , entonces la popularidad de un documento sería:

$$P = \frac{c}{r^\beta} \quad \text{con } 0 < \beta < 1$$

Algunos investigadores han encontrado que el valor de β es cercano a la unidad [7],[11], precisamente siguiendo la ley de Zipf. Otros autores [3],[10],[13] han encontrado que el valor de β es menos que la unidad, y que la distribución puede ser descrito “como la de Zipf” con el valor de β variando dependiendo del tráfico. Este comportamiento típico puede describirse como una línea recta de pendiente (negativa) β si representamos en unos ejes logarítmicos P frente a r . Este ajuste lineal suele ser casi perfecto para la parte principal del cuerpo de la distribución, sin embargo, suele

desajustarse tanto para los elementos más populares como para los menos populares (debido a los *one-timers*) [3].

La carga de tráfico generada en la aplicación desarrollada puede modelar tanto la Ley de Zipf como el comportamiento “*como el de Zipf*”, después de fijar el número de *one-timers* como se explicó en la sección anterior, las referencias de los restantes documentos distintos se determina mediante una distribución “*como la de Zipf*”. El valor de β facilitado como parámetro será el que determine si las referencias siguen estrictamente la ley de Zipf ($\beta = 1$) o no ($0 < \beta < 1$).

4.3.2 IMPLEMENTACIÓN

Para determinar la popularidad de los restantes documentos, el primer paso es calcular la constante de proporcionalidad c de la fórmula de Zipf:

$$P = \frac{c}{r^\beta}$$

usando para ello los parámetros conocidos como el número de referencias (N), número de archivos distintos (a), y el número de *one-timers* (b). Puesto que todos los *one-timers* tienen un número de referencias igual a 1, el ranking r del primer *one-timers* es $a - b + 1$, y el ranking del último *one-timer* es a , la constante c puede ser calculada usando el *one-timer* situado en el centro del rango de los *one-timers*, $r = a - (b/2)$ y con popularidad $P = 1$. Sustituyendo estos valores en la fórmula de Zipf, obtenemos:

$$1 = \frac{c}{\left(a - \frac{b}{2}\right)^\beta}$$

despejando:

$$c = \left(a - \frac{b}{2}\right)^\beta$$

una vez que se ha calculado el valor de c , la popularidad de cada uno de los documentos que faltan por hallar (aquellos que ocupan el ranking desde 1 hasta $a - b$), puede ser calculada fácilmente directamente de la fórmula de Zipf. Este algoritmo se muestra en la figura 4.2. Destacar que no es necesaria la generación de ningún número aleatorio en

este paso, sino que todas las popularidades son generadas directamente de la fórmula de Zipf, naturalmente redondeamos hacia el entero más próximo.

Dado:

a , el número de documentos distintos

b , el número de *one-timers*

β , el exponente de la fórmula de Zipf

primero, calculamos c , la constante de proporcionalidad de la fórmula de Zipf:

$$c = \left(a - \frac{b}{2}\right)^\beta$$

luego, generamos la popularidad P de cada uno de los no *one-timers*:

$$P_r = \frac{c}{r^\beta}, \quad r = 1, 2, 3, \dots, a - b$$

Figura 4.2. Algoritmo de generación de popularidades según la distribución de Zipf

Desafortunadamente, después de determinar la popularidad de cada uno de los documentos como se ha indicado anteriormente, la suma de las popularidades de todos los documentos no siempre coincide exactamente con N (que es el número total de peticiones a generar). Hay dos razones principales para esto: el cálculo de la constante c no es exacto, y que todas las popularidades han sido redondeadas al entero más próximo. Para resolver este problema, se ha añadido un paso adicional para ajustar más la popularidad de cada documento, para ello se ha escalado la popularidad de los no *one-timers* (aquellos que ocupan los puestos de popularidad del 1 al $a - b$), manteniendo en todo momento la pendiente β . El factor de escala s se determina usando:

$$s = \frac{N - b}{\sum_{i=1}^{a-b} P_i}$$

Por otro lado, el cálculo de s excluye a los *one-timers* porque sus popularidades no se han visto alteradas. Por la misma razón, al ajustar la popularidad de los no *one-timers*, popularidades menores que 2 no son permitidas.

Siguiendo con estos cálculos, el número total de peticiones es muy cercano a N (típicamente dentro del 1-5% y casi siempre dentro del 10%).

4.4. Distribución del tamaño de los documentos

4.4.1. INTRODUCCIÓN

Estudios previos de tráfico Web [3],[8] han demostrado que la distribución de los tamaños de los documentos de tráfico Web es de cola pesada. Este tipo de distribuciones implica que relativamente pocos archivos de tamaño muy grande acumulan un porcentaje elevado del volumen total de datos del tráfico Web. Por tanto, la distribución del tamaño de los archivos afecta significativamente en el diseño de estrategias de cachés. Almacenando en caché sólo archivos pequeños puede reducir el número de peticiones enviadas a los servidores de origen, y puede convertirse en una tasa elevada de aciertos de documentos en cachés, pero también se traduce en un bajo número de tasa de aciertos por byte. En el otro extremo, almacenando en caché documentos de gran tamaño provoca una elevada tasa de acierto por byte a costa de la tasa de aciertos por documentos.

Para una evaluación efectiva de la gestión de las estrategias en caché, la distribución de cola pesada de los tamaños de archivos debe ser incorporada en la generación de muestras de tráfico, en particular, la “pesadez” de la cola debe ser ajustable de modo que su impacto en las cachés pueda ser evaluado.

4.4.2. IMPLEMENTACIÓN

En la aplicación, el modelado de la distribución del tamaño de los archivos se ha dividido en tres partes:

1. Se modela la cola de la distribución usando una distribución de Pareto (también conocida como distribución de cola pesada).
2. Se modela el cuerpo de la distribución usando una distribución logarítmica normal.
3. Se unen ambas distribuciones.

4.4.2.1. MODELACIÓN DE LA COLA

La cola de la distribución del tamaño de los archivos puede ser modelada con una función de cola pesada. Una distribución se dice de cola pesada si la forma asintótica de la distribución es hiperbólica. Esto es:

$$P(X > x) \approx x^{-\alpha} \quad \text{con } x \rightarrow \infty$$
$$0 < \alpha < 2$$

La distribución de Pareto de doble exponencial es un ejemplo de una distribución de cola pesada y ha sido la utilizada para modelar la cola de la distribución del tamaño de los archivos [x,x,x]. Su función densidad de probabilidad es:

$$p(x) = \alpha \cdot k^\alpha \cdot x^{-\alpha-1}, \quad \alpha, k > 0$$
$$x \geq k$$

y su función de distribución acumulada:

$$F(x) = P(X \leq x) = 1 - \left(\frac{k}{x}\right)^\alpha$$

El parámetro α , llamado índice de cola, determina la “pesadez” de la cola de la distribución. La distribución tiene varianza infinita y si $\alpha \leq 1$, entonces la distribución tiene media infinita. Esto implica que pequeños valores de α representan colas pesadas (la mayoría del volumen estaría presente en la cola de la distribución), k es un parámetro que determina dónde empieza la cola de la distribución (representa el valor más pequeño posible que puede tomar la variable aleatoria de la distribución de cola pesada).

Los parámetros α y k que caracterizan la distribución del tamaño de los archivos pueden ser determinados usando una representación logarítmica de la distribución complementaria [3],[8],[11]. Si trabajamos con la función de distribución acumulada complementaria $\overline{F}(x) = 1 - F(x)$, podemos comprobar como se verifica que

$$\frac{d \log \overline{F}(x)}{d \log x} = -\alpha, \quad \alpha > k$$

la pendiente α puede ser estimada al exhibir un comportamiento lineal. Este método es empleado para estimar α en los estudios de tráfico sintético (puesto que α y k son proporcionados como parámetros en el proceso de generación de muestras).

Incorporar el modelo de cola pesada en la aplicación comienza distinguiendo entre los archivos en dos grupos: aquellos que se encuentran en el cuerpo de la distribución y aquellos que se encuentran en la cola. El porcentaje de archivos en la cola es especificado como un parámetro de entrada del generador de tráfico.

Para generar el tamaño de los archivos se despeja en la distribución de la cola de Pareto, en función de los parámetros de entrada. Sea:

$$y = 1 - \left(\frac{k}{x}\right)^\alpha$$

El objetivo es obtener para un número aleatorio uniforme y entre $(0,1)$, el correspondiente valor de x . Despejando en la ecuación obtenemos:

$$x = \left(\frac{k^\alpha}{1-y}\right)^{\frac{1}{\alpha}}$$

que también puede ser escrita como:

$$x = \frac{k}{(1-y)^{\frac{1}{\alpha}}}$$

Si y es una variable aleatoria uniforme $(0,1)$ entonces $1 - y$ es también un valor aleatorio uniforme $(0,1)$. Por tanto, la ecuación anterior quedaría:

$$x = \frac{k}{(y')^{\frac{1}{\alpha}}} \quad \text{donde } y' = 1 - y$$

Así, una vez que α y k han sido especificados en la aplicación, los valores de cola pesada pueden ser generados usando esta fórmula. El algoritmo de la figura 4.3 usa esta función de mapeo inversa para incorporar las características de cola pesada en la distribución del tamaño de los archivos. No obstante, hay veces en que los valores generados son excesivamente grandes, por lo que se deja como opción la posibilidad de poder eliminar aquellos tamaños de archivos que excedan de cierto valor, en nuestro caso se opta a 50 MB el tamaño máximo, en caso de generarse un tamaño mayor se descarta y se genera un nuevo valor, esto se realiza fundamentalmente para ajustar lo más parecido posible las muestras de tráfico generadas con las muestras reales.

Dados:

α , el índice de la distribución de Pareto

k , el comienzo de la cola de la distribución

n , el número de valores de cola pesada a generar

repetiendo los siguientes pasos n veces

1. generamos y' como una variable aleatoria uniforme en el rango (0, 1)

2. calculamos x como $x = \frac{k}{(y')^{\frac{1}{\alpha}}}$

Figura 4.3. Algoritmo de generación de la cola de Pareto en la distribución de tamaño

4.4.2.2. MODELACIÓN DEL CUERPO DE LA DISTRIBUCIÓN

Para modelar el cuerpo de la distribución de tamaño de los archivos usamos una distribución logarítmica normal [19]. La distribución logarítmica normal tiene la propiedad que si $X \approx N(\mu, \sigma^2)$, es decir, X está normalmente distribuida con media μ y varianza σ^2 , entonces e^X tiene una distribución logarítmica normal con parámetros μ y σ^2 , denotado por $LN(\mu, \sigma^2)$. La función densidad de probabilidad de una distribución logarítmica normal es:

$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}$$

Aunque la distribución logarítmica normal no tiene una forma cerrada y por tanto no tiene una función de distribución acumulada, hay una aproximación sencilla que explota la propiedad anterior de la distribución logarítmica normal para generar variables logarítmicas normales.

Así, para generar una variable logarítmica normal como se muestra en la figura 4.4, es suficiente con generar una variable aleatoria x normal $N(\mu, \sigma^2)$, y entonces devolver e^x como una variable logarítmica normal. Sin embargo, μ y σ^2 son la media y la varianza de una distribución normal, por lo que debemos, a partir de dichos valores, obtener la media y la varianza de una distribución logarítmica normal. Esos valores son:

$$\mu_l = e^{\mu + \sigma^2/2} \quad \sigma_l^2 = e^{2\mu + \sigma^2} (e^{\sigma^2} - 1) \quad [19]$$

De este modo, si queremos generar una variable aleatoria logarítmica normal con un determinado μ_l y σ_l^2 , primero deberíamos resolver:

$$\mu = \ln\left(\frac{\mu_l^2}{\sqrt{\sigma_l^2 + \mu_l^2}}\right) \quad \text{y} \quad \sigma^2 = \ln\left[\frac{\sigma_l^2 + \mu_l^2}{\mu_l^2}\right]$$

Una vez que tenemos estos valores, podremos generar una variable aleatoria normal $N(\mu, \sigma^2)$, con valores provenientes de una distribución logarítmica normal, y con esa variable aleatoria devolver e^x , con lo que ya habremos generado una variable aleatoria logarítmica normal de media μ_l y varianza σ_l^2 .

Dados:

μ_i , la media de la distribución logarítmica normal

σ_i , la desviación típica de la distribución logarítmica normal

Calculamos μ y σ^2 , los parámetros de una distribución normal

$$\mu = \ln\left(\frac{\mu_i^2}{\sqrt{\sigma_i^2 + \mu_i^2}}\right)$$

$$\sigma^2 = \ln\left[\frac{(\sigma_i^2 + \mu_i^2)}{\mu_i^2}\right]$$

Ahora repetimos los siguientes pasos tantas veces como sea necesario para generar los valores de la distribución logarítmica normal:

1. Generamos x como una variable aleatoria uniforme (0, 1)
2. Calculamos $e^{\mu+\sigma x}$

Figura 4.4. Algoritmo de generación de valores para una distribución logarítmica normal

4.4.2.3. UNIÓN DE AMBAS DISTRIBUCIONES

Puesto que modelamos tanto la cola de la distribución como el cuerpo de la misma de forma separada, debemos tener en cuenta ciertas restricciones necesarias a la hora de unir ambas distribuciones. Primero, los valores que se encuentren en el cuerpo de la distribución no pueden encontrarse también en la cola. Segundo, el caso contrario tampoco debe producirse. Tercero, cuando representemos la función de distribución acumulada, debe producirse una transición suave desde el cuerpo hasta la cola.

Un primer paso para lograr estos objetivos sería limitar superiormente el tamaño de los archivos generados para el cuerpo de la distribución³. Es decir, si un valor mayor

³ Afortunadamente, ya existe un límite inferior, k , en los valores generados en la cola de la distribución de Pareto.

que k es generado, se descarta y se genera un nuevo valor. Los efectos colaterales de esta restricción son: (1) después de la generación de las variables logarítmicas normales, la media y la desviación típica de los valores resultantes son menores que los especificados en los valores de entrada, y (2) la distribución del tamaño de los archivos resultantes puede presentar una obvia discontinuidad donde se encuentran ambas distribuciones. Otra posibilidad sería eliminar la restricción de tamaño superior, resolviendo estos dos problemas, pero a cambio, se incrementaría el porcentaje de archivos en la cola de la distribución. En esta aplicación se ha optado por la segunda opción, no obstante, en el código del mismo se deja abierta la posibilidad de la primera opción.

4.5. Correlación entre el tamaño de los archivos y su popularidad

Numerosos estudios de tráfico Web en proxies muestran que muchos de los archivos transferidos en la Web tienen un tamaño pequeño [1],[3],[10]. Una cuestión natural que se plantea al respecto es si existe alguna correlación estadística entre la frecuencia de acceso a un determinado archivo y su tamaño. Algunos estudios [10],[18] han mostrado que hay una muy pequeña correlación entre su frecuencia de acceso y su tamaño, aunque esta cuestión es todavía tema de debate.

En esta aplicación, para otorgarla de mayor flexibilidad, se ofrece la posibilidad de que el tráfico generado tenga correlación positiva, negativa o cero entre la popularidad de los archivos y su tamaño. Correlación positiva significa que los archivos de mayor tamaño tienen mayor popularidad y correlación negativa significa que los archivos de menor tamaño tienen más probabilidad de ser requeridos (mayor popularidad). Mientras que correlación cero no otorga ninguna correlación entre la popularidad de los archivos y su tamaño. El hecho de permitir la existencia o no de correlación radica en la posibilidad de explorar distintos algoritmos de cacheo según cada una de estas opciones.

Modelar e incorporar estas características de correlación dentro de la generación de muestras de tráfico se realiza en 3 etapas: (1) generar un conjunto de popularidades de archivos usando la aproximación de la sección 4.3; (2) generar un conjunto de tamaños de archivos usando la aproximación de la sección 4.4; y (3) usar una técnica de mapeo para introducir correlación positiva, negativa o cero entre la popularidad de los archivos y su tamaño. El modelo sería el siguiente:

1. Se genera una lista P de popularidades para n archivos distintos usando la aproximación explicada en la sección 4.3. Se ordena la lista P en orden ascendente.
2. Se genera una lista S de tamaños de archivos para n archivos distintos usando la aproximación explicada en la sección 4.4. Se ordena la lista S en orden ascendente.
3. Se calcula los valores de la función de distribución acumulada para los elementos de la lista P . Se construye una nueva lista P_{nueva} , con valores de la forma (v_i, p_i) , donde v_i es un valor único de popularidad de la lista P y p_i representa su probabilidad acumulada.
4. Se calcula los valores de la función de distribución acumulada para los elementos de la lista S . Se construye una nueva lista S_{nueva} , con valores de la forma (v_i, p_i) , donde v_i es un valor único de tamaño de archivo de la lista S y p_i representa su probabilidad acumulada.
5. Se repiten los siguientes pasos n veces para generar una nueva lista L que represente la popularidad y el tamaño de los archivos distintos. La entrada j ($1 \leq j \leq n$) en L es de la forma (f_j, s_j) , donde f_j representa la popularidad y s_j el tamaño del archivo. La entrada (f_j, s_j) es determinada de la siguiente forma:
 - a) se genera un número aleatorio r_l dentro una distribución aleatoria uniforme de rango $(0, 1)$.

b) se busca en la lista ordenada P_{nueva} el primer elemento i que satisface $r_1 \leq p_i$. Si $p_i = r_1$ ó $i = 1$, entonces hacemos $f_j = v_i$. En otro caso, interpolamos linealmente usando $f_j = v_{i-1} + \frac{(r_1 - p_{i-1}) \times (v_i - v_{i-1})}{(p_i - p_{i-1})}$.

c) i. Si se desea correlación positiva, se usa r_1 para buscar en la lista S_{nueva} exactamente del mismo modo que se ha hecho en el paso b) para buscar el valor del tamaño de archivo s_j .

ii. Si se desea correlación negativa, se usa $1 - r_1$ para buscar en la lista S_{nueva} exactamente del mismo modo que se ha hecho en el paso b) para buscar el valor del tamaño de archivo s_j .

iii. Si se desea que no haya ningún tipo de correlación, se genera otro número aleatorio r_2 dentro de una distribución aleatoria uniforme de rango $(0, 1)$ y se usa para buscar en la lista S_{nueva} exactamente del mismo modo que se ha hecho en el paso b) para buscar el valor del tamaño de archivo s_j .

d) se añade el par de valores (f_j, s_j) a la lista L como la popularidad y el tamaño del archivo único j .

6. La lista L ahora representa una nueva lista de n archivos distintos, los cuales tienen su propia popularidad y tamaño de archivo con el deseado valor de correlación introducido.
7. Finalmente, se normaliza para ajustar la popularidad de los archivos al número total de peticiones a generar. Pero se va a seguir manteniendo el número de one-timers como aquel que se ha introducido como parámetro, eludiendo el hecho de que al ser elegidos aleatoriamente según su función de distribución acumulada saldrían muchos menos *one-timers*, ya que suponen un porcentaje menor sobre la función de distribución acumulada de popularidad.

4.6. Localidad temporal

Después de determinar el tamaño de los archivos así como su popularidad falta por determinar el orden relativo en el que las peticiones de los distintos archivos aparecerán en el conjunto total de tráfico. Es aquí donde entra en liza la localidad temporal.

La localidad temporal se refiere a la tendencia (en cargas reales de tráfico Web) para documentos referenciados en el pasado a volver a ser nuevamente referenciados en el futuro. De la información que ya se sabe sobre los distintos archivos, el número de veces que cada archivo debería de aparecer en las muestras de tráfico (la popularidad) ya se sabe. Sin embargo, dado que una referencia a un archivo es generada en un instante de tiempo t_0 , no está claro cuándo se va a producir la próxima referencia a dicho archivo. Es por esto, que la presencia de la localidad temporal en la generación de tráfico tiene un efecto muy importante para las pruebas con cachés.

La aproximación utilizada para la modelación de la localidad temporal está basada en el modelo de pila finita LRU (*Least Recently Used* – Menos usado recientemente). Una pila LRU es una lista de todos los archivos ordenados según hayan sido referenciados recientemente [7], esto es, el último que haya sido referenciado estará en primer lugar de la pila y el que fue referenciado hace más tiempo estará en la última posición. La pila es actualizada dinámicamente cada vez que se procesa una referencia. En muchos casos, esta actualización implica el tener que añadir un nuevo elemento en la cima de la pila empujando el resto hacia abajo, en otros casos, implica extraer un elemento existente en el interior de la pila y trasladarla a la cima de la misma, desplazando al resto de los elementos hacia abajo.

Una pila LRU de tamaño finito es una pila LRU que sólo puede almacenar un número m de archivos. Experimentos realizados por Mahanti [3], sugieren que $m = 1000$ es un número adecuado para capturar la presencia de localidad temporal en cargas de tráfico Web. No obstante, para la realización de todo tipo de pruebas en cachés, se permite que el tamaño de la pila sea especificado por el usuario como un parámetro.

El aspecto más importante en una pila LRU es que cada posición en la pila tiene asociada una probabilidad de referencia. Las probabilidades son asociadas a la posición de la pila y no a los documentos. Las probabilidades de las posiciones de la pila pueden ser proporcionadas por modelos u obtenidas de analizar muestras de tráfico reales.

Por ejemplo, supongamos que las popularidades de los distintos archivos en la carga de tráfico están representados por

$$D = \{x_1, x_2, \dots, x_n\}$$

donde

$$x_1 \geq x_2 \geq \dots \geq x_n$$

Entonces las probabilidades a_i pueden ser calculadas para cada archivo i usando

$$a_i = \frac{x_i}{\sum_{j=1}^n x_j} \quad \text{para } i = 1, 2, \dots, n$$

Hay dos formas posibles de utilizar esas probabilidades:

- Aproximación estática: si la pila finita tiene un tamaño fijo m de modo que $m \leq n$, la probabilidad acumulada y_i , se calcula como:

$$y_i = \sum_{j=1}^i a_j$$

La probabilidad acumulada calculada para cada posición de la pila es asignada al comienzo de la generación de muestras y no puede ser cambiada durante dicho proceso. Esta técnica genera una localidad temporal estadística homogénea para todos los documentos en la traza.

- Aproximación dinámica: cada vez que la pila LRU es modificada (ya sea al mover un archivo desde otra posición de la pila a la cima de la misma o bien por traer un nuevo archivo a la pila), las probabilidades acumuladas para cada posición de la pila son recalculadas usando los valores a_i de los archivos que actualmente ocupan cada posición de la pila.

Esta aproximación puede modelar propiedades de localidad temporal heterogénea. Por ejemplo, si en el instante temporal t_0 , las dos primeras posiciones de la pila contienen los archivos 1 y 2 respectivamente, entonces

sus estados pueden ser representados como $St_0 = \{a_1, a_2\}$, donde a_1 es la probabilidad de referenciar el archivo 1 y a_2 es la probabilidad de referenciar el archivo 2. Entonces, la probabilidad acumulada de referenciar esas dos posiciones de la pila son a_1 y $a_1 + a_2$, respectivamente. Asumiendo que en el instante t_1 , el archivo en la posición 2 en la pila es movido a la posición 1. Esta acción provocará que haya que recalcular las probabilidades acumuladas asociadas a cada posición de la pila, y entonces los estados de la pila se convierten en $St_1 = \{a_2, a_1\}$. Así las probabilidades acumuladas para referenciar estas dos posiciones de la pila son ahora a_2 y $a_2 + a_1$, respectivamente.

Una vez que se han descrito las dos aproximaciones utilizadas para modelar la localidad temporal, ya se puede describir el proceso de generación de referencias: al comienzo de la generación de las referencias, la pila LRU está vacía. Si se elige la aproximación estática, cada posición de la pila tiene asociada una probabilidad acumulada de referenciar a esa posición. En el caso de la aproximación dinámica, las probabilidades están sin inicializar. El proceso de generación de referencias comienza generando un número aleatorio x_i de una distribución aleatoria uniforme de rango $(0, 1)$. Después debemos comprobar si ese archivo seleccionado está en la pila o no (si $x_i \leq y_i$), ahora pueden producirse tres circunstancias: (1) la pila está vacía; (2) la pila no está vacía pero el próximo archivo a referenciar no está en la pila; (3) la pila no está vacía y el próximo archivo a referenciar ya está en la pila.

- En los casos 1 y 2, un archivo es seleccionado al azar del conjunto de archivos distintos que quedan aún por referenciar. Se genera una referencia para el archivo seleccionado y el contador de referencias que le faltan por generar (es decir, su popularidad) es decrementada en una unidad. Si no quedan más referencias por generar de este archivo seleccionado, entonces el archivo es eliminado del conjunto de archivos distintos que quedan por referenciar. En caso contrario, el archivo es movido hasta la cima de la pila (desplazando a los otros archivos hacia abajo en la pila si es el caso 2).
- En el caso 3, se busca desde el comienzo de la pila al mayor elemento de la misma que verifique $x_i \leq y_i$. Una vez encontrado se generará una referencia para ese archivo y se decrementará en una unidad el contador de referencias

que le quedan por generar (popularidad), si ya no le quedan más referencias por generar, el archivo es eliminado de la pila y todos los archivos situados debajo de él suben una posición en la pila. En otro caso, el archivo es movido a la cima de la pila, desplazando el resto de elementos (si los hubiera) una posición hacia abajo.

En el modelo de pila dinámica LRU, cada uno de estos casos provocará nuevos cálculos de las probabilidades acumuladas asociadas con esas posiciones de la pila.

4.7. Generación de muestras

Una vez que se han introducido todos los parámetros y se ejecuta la generación de muestras, la aplicación devuelve un archivo, de contenido similar a la figura 4.5, con el nombre indicado por el usuario. Este archivo consta de una tabla con 3 campos: el primer campo es el identificador de archivo, es decir, es el nombre del archivo que se ha generado, estos nombres han sido sustituidos por números para facilitar su procesamiento, cada archivo se repetirá tantas veces como indique su popularidad; el segundo campo nos indica el tamaño del archivo en bytes; y el tercer campo nos indica el tipo de archivo generado: aplicación, audio, imágenes, mensajes, texto y video.

39425	89	5
3492	1281	0
26513	3152	5
2031	42	5
18811	107464	1
37595	988	5
38038	20	3
51436	8654	2
43798	3153	5
10547	28	2
22087	135	4
48467	152	2

32134	20	0
36870	1065	0
45630	20869	2
30173	966	5
25411	74	5
32902	36	1
11526	7544	0

Figura 4.5. Ejemplo de generación de muestras

CAPÍTULO 5: Pruebas

A lo largo de este capítulo se va a realizar un estudio pormenorizado de las distintas alternativas de generación de muestras en función de todos sus parámetros de entrada. Paralelamente se analizarán e interpretarán dichos resultados.

En primer lugar, hay que resaltar que el formato numérico adoptado en la aplicación desarrollada para la generación de muestras proviene de un proceso previo ya existente. Este proceso ha sido realizado por el Departamento de Tecnología Electrónica de la Universidad de Málaga. Se parte de muestras de tráfico que contienen información de cada una de las peticiones HTTP que se han realizado a través de un proxy, del proyecto IRCache[20] situado en el *Research Triangle Park* en Carolina del Norte (EEUU) durante los días 7 al 11 de junio de 2004, como la hora a la que se realizó la petición, el tamaño del objeto, la URL solicitada, el tipo de objeto (*content-type*), el método de petición HTTP (GET, POST, ...) y el código de respuesta del servidor.

Estas muestras han sido preprocesadas para eliminar en primer lugar aquellas peticiones que han sido generadas dinámicamente mediante CGI (*Common Gateway Interface*), ya que los objetos que se devuelven en este tipo de peticiones son únicos para cada petición y, por lo tanto, no tiene sentido almacenarlos en caché [21]. Para ello se han descartado aquellas peticiones que contienen la cadena ‘cgi’, ‘cgi-bin’ o ‘?’. También se han filtrado aquellas peticiones que contienen la cadena ‘:3128’, ya que se corresponde con el puerto por el que las cachés del proyecto IRCache se comunican para colaborar entre ellas, al ser una arquitectura jerárquica. Como códigos de respuesta “cacheables” se han considerado el 200 (*OK*), 203 (*Partial*), 206 (*Partial Content*), 300 (*Multiple Choices*), 301 (*Moved*) y 302 (*Redirect*). Para el caso de las peticiones que tienen como código de respuesta el 304 (*Not Modified*), el tamaño de objeto que aparece en las trazas no se corresponde con el tamaño real del objeto, sino con el tamaño de la respuesta dada por el servidor para notificar al usuario que el objeto que tiene en su caché local es la misma que está en el servidor. Estos objetos han vuelto a ser pedidos al servidor Web original para poder averiguar su tamaño real. También se han vuelto a solicitar aquellos objetos de los que se desconocía su *content-type*.

Una vez filtrado todo el conjunto de muestras, se sustituye la información resultante por números para facilitar su posterior tratamiento por otras herramientas, por ejemplo, a cada URL distinta se le asigna un número de identificador distinto y por tanto cada referencia a una misma URL será identificada por el mismo número.

Por tanto, toda la información del tráfico queda reducida a un fichero donde cada línea contiene una tabla con 5 campos:

1. Instante temporal en que se hizo la petición.
2. Tamaño del objeto en bytes.
3. Identificador del objeto.
4. Primera parte del tipo de contenido (*content-type*):
 - 0 : Aplicaciones
 - 1 : Audio
 - 2 : Imágenes
 - 3 : Mensajes
 - 4 : Texto
 - 5 : Vídeo
5. Segunda parte del tipo de contenido, es un número que dependerá del campo anterior e identifica con más precisión el tipo de archivo (p.ej., si es *jpg* o *bmp* o *gif*, ...).

Esta asignación de identificadores para los campos explicados anteriormente es una semántica utilizada por otras aplicaciones desarrolladas por el Departamento de Tecnología Electrónica de la Universidad de Málaga [22]. Mediante una herramienta de procesado de las muestras de tráfico, permite una traducción de los campos de las peticiones HTTP más importante a dígitos decimales que facilitarán el posterior procesamiento y generación de dichas muestras de tráfico.

En el caso de la aplicación desarrollada, se hace un filtrado adicional tanto del instante de tiempo como de la segunda parte del tipo de contenido, puesto que no aportan ninguna información necesaria para el objetivo de dicha aplicación.

De este modo, los parámetros procesados obtenidos y que se introducen en la aplicación son: un primer número que identifica a los distintos URLs, un segundo número que indica el tamaño del archivo o documento, y un tercer número que aporta información acerca del tipo de documento.

Del mismo modo, la generación de muestras se realiza con idéntica sistemática, es decir, se genera un primer número que indica la URL, otro que indica su tamaño y un tercero que muestra el tipo de documento generado.

A continuación se muestra un estudio exhaustivo de generación de muestras de tráfico en función de los diferentes parámetros de entrada, comparándolo también con los resultados ideales según los modelos utilizados y explicando brevemente las características más importantes. Previamente, se validará la aplicación mediante el estudio y análisis de un ejemplo con los valores de entrada más comunes y que más se ajustan al tráfico Web. Por otro lado, indicar que después de cada generación de muestras se crea un archivo llamado "*parametros.txt*" que guarda los valores obtenidos más importantes de dicha generación y que posteriormente podrían ser usados por alguna futura aplicación.

5.1. Validación

La aplicación desarrollada requiere de doce parámetros de entrada para generar muestras de tráfico. Según diversos estudios de las características del tráfico Web [1],[3],[6], los parámetros para la generación de tráfico más usuales son los representados en la tabla 5.1.

Parámetro	Valor
Porcentaje de documentos distintos	30%
Porcentaje de <i>one-timers</i>	70%
Pendiente de Zipf	0.75
Índice de cola-pesada	1.2
Comienzo de la cola en bytes, <i>k</i>	10.000
Porcentaje de documentos en la cola-pesada	20%
Media de la distribución logarítmica normal (μ)	7.000
Varianza de la distribución logarítmica normal (σ)	11.000
Modo de la pila para la localidad temporal	Dinámico
Tamaño de pila para la localidad temporal	1.000

Tabla 5.1. Valores típicos para la generación de muestras de tráfico

Una vez que se realiza la generación de muestras con estos parámetros de entrada, los resultados obtenidos se muestran en las siguientes gráficas (5.1 – 5.6) junto con el análisis de las mismas.

En la gráfica mostrada en la figura 5.1 se puede verificar que las muestras de tráfico generadas tienen un comportamiento “como el de Zipf”, esto es, al trazar en ejes logarítmicos, la popularidad de los documentos frente a su ranking (producido al ordenar dicha popularidad) se observa como sigue una distribución lineal con una pendiente igual a -0.76045, el caso ideal sería -0.75. Mencionar que tanto la muestra generada como la muestra ideal están casi superpuestas y es sólo en los valores extremos donde se puede apreciar una mínima diferencia.

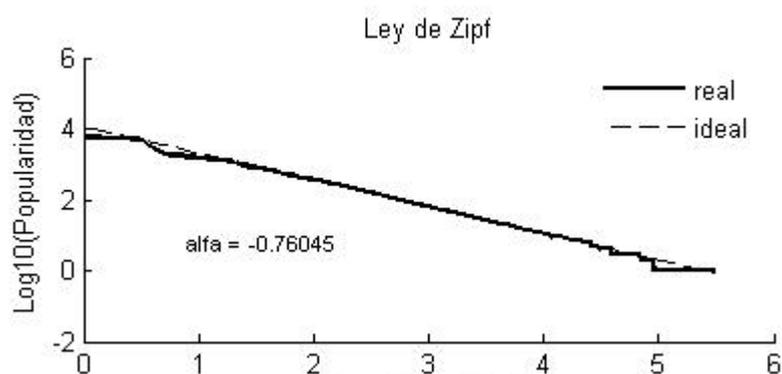


Figura 5.1. Análisis del comportamiento de la popularidad

A continuación, en la gráfica 5.2, se aprecia el comportamiento de la correlación entre el tamaño de los archivos y su frecuencia de acceso, y se observa como no hay ningún tamaño de archivo que predomine sobre el resto, es decir, todos los tamaños de archivo tienen el mismo peso. El valor de correlación obtenido es de 0.00029 pero el resultado que se expone es 0, puesto que sólo distinguiremos entre -1, 0 y 1. Por tanto, el resultado coincide con el valor de correlación de entrada.

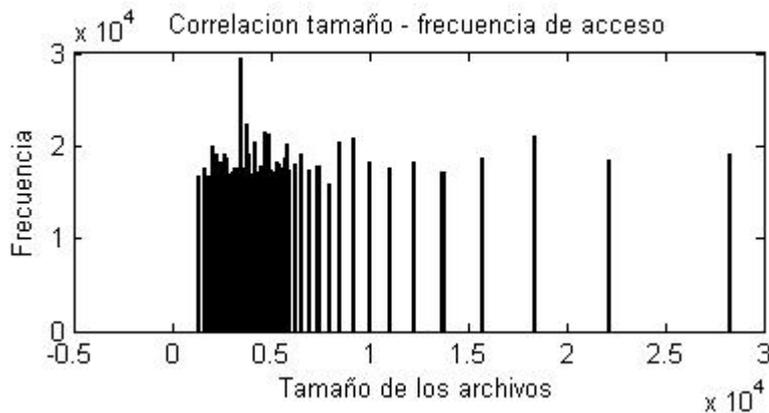


Figura 5.2. Análisis de la correlación entre el tamaño y la frecuencia de acceso

La figura 5.3 representa la función de distribución acumulada de los tamaños de los archivos en el cuerpo de la distribución, es decir, si el valor de k es de 10.000 bytes, significa cómo se distribuyen los tamaños de los archivos para valores menores que k . Es quizás, la característica que más se aleja al valor ideal:

$\mu_{ideal} = 7.000$	$\mu_{real} = 5.536,78$
$\sigma_{ideal} = 11.000$	$\sigma_{real} = 4.778,57$

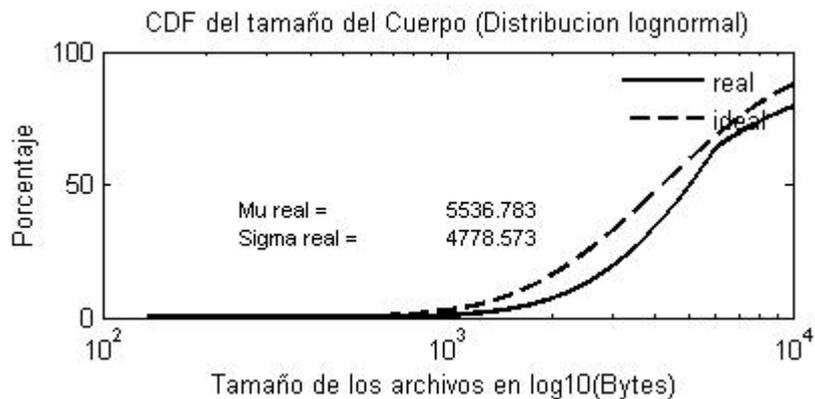


Figura 5.3. Análisis de la función de distribución acumulada

Se explica en el amplio rango de valores aleatorios que se pueden generar y en los errores de aproximación de los modelos matemáticos utilizados tanto para la generación de muestras como para su posterior análisis. Puesto que para la generación de muestras, se debe llegar a un compromiso entre todas las variables de entrada, se ha considerado que sea en el tamaño de los archivos donde recaiga el mayor margen de error. No obstante, sí que se puede apreciar que el comportamiento de las distribuciones de tamaño de las muestras son bastantes similares.

En la figura 5.4, se muestra la función de densidad de probabilidad de los tamaños de los archivos tanto en el cuerpo como en la cola de la distribución. Esta gráfica nos proporciona una idea muy intuitiva del tamaño de los archivos.

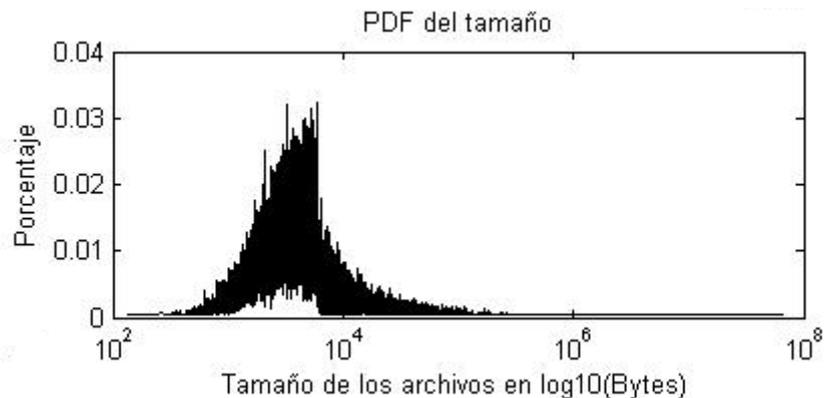


Figura 5.4. Análisis de la función de densidad de probabilidad del tamaño

En la gráfica 5.5, se muestra la pendiente de la cola de Pareto, tanto para las muestras reales obtenidas como para las muestras ideales, que se pasa como parámetro de entrada. El valor de inicio del comportamiento de cola de Pareto lo marca el valor de k . Ambos conjuntos de muestras son prácticamente coincidentes.

La gráfica 5.6, indica la localidad temporal presente en las referencias a los archivos. Es decir, cuando se produce una referencia a un archivo es relativamente más probable que en un instante corto de tiempo se vuelva a producir una nueva referencia a dicho documento. Éste fenómeno queda patente en la mencionada gráfica.

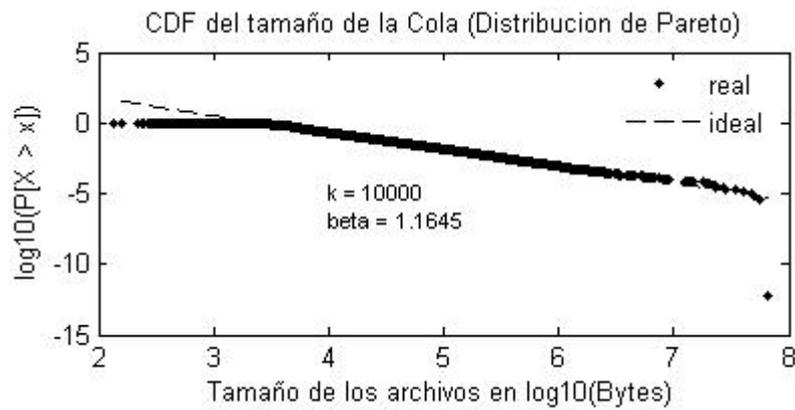


Figura 5.5. Análisis de la pendiente de cola de Pareto

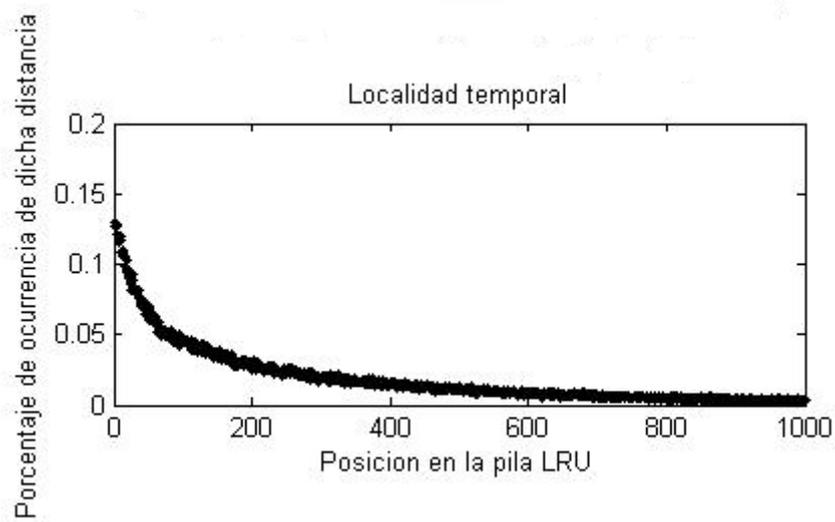


Figura 5.6. Análisis de la localidad temporal en modo dinámico

En la tabla 5.2 se muestran los resultados numéricos obtenidos de la generación de muestras realizada, esto es, la generación de muestras con aquellos valores de entrada que se han considerado como más habituales y que más se ajustan al tráfico habitual.

Parámetro	Valor
Número total de muestras	916.207
Porcentaje de documentos distintos	32.74%
Porcentaje de <i>one-timers</i>	70%
Pendiente de Zipf	0.76045
Índice de cola-pesada	1.16452
Correlación	0.00029
Porcentaje de documentos en la cola-pesada	20,443%
Media de la distribución logarítmica normal (μ)	5.536,78
Varianza de la distribución logarítmica normal (σ)	4.778.57
Total bytes generados	13,52 GB
Tamaño medio de los archivos generados	14.440 bytes
Tamaño del archivo mayor	58 MB

Tabla 5.2. Valores obtenidos tras la simulación con valores típicos

Una vez que se han estudiado los parámetros generales de la simulación, habrá que verificar los resultados en relación con los tipos de archivos generados. Como parámetros de entrada se han elegido estos valores, según dos criterios; por un lado, que se ajuste lo más posible a los valores medios obtenidos tras el análisis de muestras reales, y que cada tipo de archivo tenga, al menos, una presencia mínima:

Tipo aplicación: 9%
 Tipo audio: 10%
 Tipo imágenes: 55%
 Tipo mensaje: 5%
 Tipo texto: 11%
 Tipo video: 10%

Los resultados, en cuanto a tipos de archivos, obtenidos tras la generación de muestras son los siguientes:

Tipo aplicación: 82823 archivos → 9,03% con un tamaño medio de 17,35 KB
 Tipo audio: 92192 archivos → 10,06% con un tamaño medio de 13,62 KB

Tipo imágenes:	494829 archivos → 54,00% con un tamaño medio de 15,10 KB
Tipo mensaje:	48408 archivos → 5,28% con un tamaño medio de 13,13 KB
Tipo texto:	103572 archivos → 11,30% con un tamaño medio de 16,32 KB
Tipo video:	94382 archivos → 10,30% con un tamaño medio de 17,84 KB

Se observa como los valores resultantes se ajustan con bastante exactitud a los tipos solicitados.

5.2. Variación de los parámetros de entrada

En este apartado se realizarán generaciones de muestras variando uno de los parámetros de entrada y se estudiarán las repercusiones de dicha variación. El valor del resto de los parámetros de entrada será el mismo que los ya mostrados en la tabla 5.1 a excepción del parámetro a estudiar.

5.2.1 “ONE-TIMERS”

Se realizarán dos generaciones de muestras variando dicho parámetro. Los valores típicos encontrados en el análisis de muestras de tráfico reales concluyen que dicho parámetro oscila entre el 60% y 70% de las muestras totales [1].[3], por lo que se utilizarán valores más extremos para resaltar más las posibles diferencias entre una y otra generación. Por tanto, se realizarán generaciones con un 50% y un 80% de “*one-timers*”.

Los resultados numéricos obtenidos se representan en la tabla 5.3.

	Generación de Muestras 1	Generación de Muestras 2
% One- timers	50%	80%
<i>Resultados obtenidos</i>		
Porcentaje de documentos distintos	30,77%	32,56%
Porcentaje de <i>one-timers</i>	50%	80%
Número total de muestras	974.685	921.255
Pendiente de Zipf	0,761637	0,74973
Índice de cola-pesada	1,14915	1,1848
Correlación	Cero	Cero
Porcentaje de documentos en la cola-pesada	19,93%	23,5793%
Media de la distribución logarítmica normal (μ)	5528,59	5539,40
Varianza de la distribución logarítmica normal (σ)	4763,47	4585,72

Tabla 5.3. Comparativa de simulaciones con distinto porcentaje de *one-timers*

La primera conclusión que se extrae de los datos obtenidos es que ambas simulaciones obtienen unos resultados muy similares, es decir, todos los parámetros analizados se ajustan en gran medida a los parámetros de entrada introducidos.

Aún así, se puede extraer la siguiente lectura de dichos datos: al aumentar el número de *one-timers*, se disminuye el número de aquellos documentos que son solicitados más de una vez, por tanto, existen menos archivos con popularidad elevada y es por esto por lo que el total de muestras disminuye.

En relación con los tipos de archivos los resultados también se ajustan al mismo patrón común y se muestran en la tabla 5.4, donde se indica el número de archivos de cada tipo generado.

Algunos autores [13] concluyen que las distintas políticas de reemplazo de caché son relativamente insensibles a los cambios en los porcentajes de *one-timers*.

Tipos	Generación de Muestras 1	Generación de Muestras 2
Aplicación	85.892	79.393
Audio	99.899	94.927
Imágenes	527.933	515.416
Mensaje	49.954	42.652
Texto	111.923	96.934
Vídeo	99.984	91.933

Tabla 5.4 Comparativo de tipos de archivos obtenidos para distintos *one-timers*

5.2.2 PENDIENTE DE ZIPF

Se realizarán dos generaciones de muestras. Si bien los valores típicos oscilan entre 0,7 y 0,8 [1],[3] para resaltar las diferencias entre ambas situaciones, éstas se harán con valores de 0,65 y 0,85. Los resultados obtenidos se muestran en la tabla 5.5.

	Generación de Muestras 1	Generación de Muestras 2
Pendiente de Zipf	0,65	0,85
<i>Resultados obtenidos</i>		
Porcentaje de documentos distintos	34,23%	30,72%
Porcentaje de <i>one-timers</i>	70%	70%
Número total de muestras	876.208	976.253
Pendiente de Zipf	0,63553	0,84642
Índice de cola-pesada	1,1853	1,16831
Correlación	Cero	Cero
Porcentaje de documentos en la cola-pesada	21,75%	20,51%
Media de la distribución logarítmica normal (μ)	5536,64	5534,65
Varianza de la distribución logarítmica normal (σ)	4704,34	4733,10

Tabla 5.5. Comparativo de tipos de archivos obtenidos para distintas pendientes de Zipf

De nuevo, en la mayoría de los parámetros los resultados de entrada y salida coinciden, no obstante, se puede observar como el número total de muestras generadas es el único parámetro que presenta valores diferenciados aunque siempre dentro de un margen. La explicación es bien clara, y es que al aumentar la pendiente de Zipf aumenta el área bajo la curva de popularidades, véase figura 5.7, lo que significa que aumenta el número total de muestras.

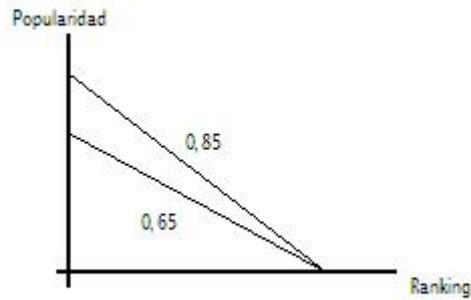


Figura 5.7. Mayor pendiente de Zipf significa mayor número de muestras

En las figuras 5.8 y 5.9 pueden verse gráficamente una comparativa de la diferencia de pendientes para ambos casos.

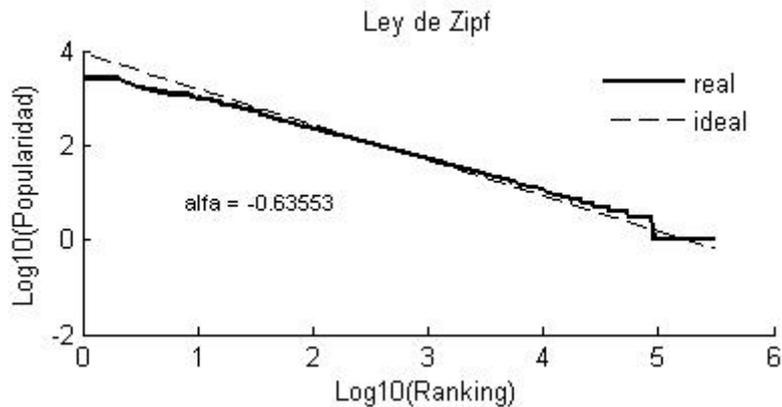


Figura 5.8. Resultados gráficos para simulación con pendiente de Zipf de 0,65

En cuanto a los tipos de archivos, el cambio de valores de entrada en la pendiente de Zipf, y como ya veremos el de ningún otro parámetro de entrada, no afecta en los resultados. De este modo obtenemos los mismos resultados que en la tabla 5.4.

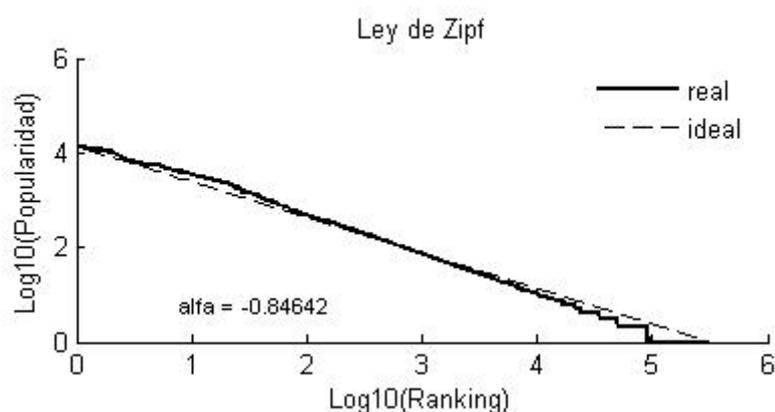


Figura 5.9. Representación gráfica de la simulación con pendiente de Zipf de 0,85

5.2.3 ÍNDICE DE COLA PESADA

Los valores típicos para este parámetro rondan la pendiente de 1,2 [1],[3] y para comparar simulaciones se realizaron pruebas con pendientes de 1,1 y de 1,3. Los resultados se muestran en la tabla 5.6.

	Generación de Muestras 1	Generación de Muestras 2
Índice de cola-pesada	1,1	1,3
<i>Resultados obtenidos</i>		
Porcentaje de documentos distintos	32,73%	32,61%
Porcentaje de <i>one-timers</i>	70%	70%
Número total de muestras	916.519	919.704
Pendiente de Zipf	0,72932	0,7787
Índice de cola-pesada	1,063	1,3029
Correlación	Cero	Cero
Porcentaje de documentos en la cola-pesada	23,49%	20,03%
Media de la distribución logarítmica normal (μ)	5538,93	5540,28
Varianza de la distribución logarítmica normal (σ)	4632,43	4717,56

Tabla 5.6. Comparativo de tipos de archivos obtenidos para distintas pendientes de cola pesada

En las gráficas 5.10 y 5.11 se pueden apreciar los efectos de éstos parámetros, de nuevo todos los resultados obtenidos son bastante similares a los parámetros de entrada y es en el porcentaje de documentos en la cola pesada donde se pueden apreciar los efectos de esa variación en el parámetro de entrada.

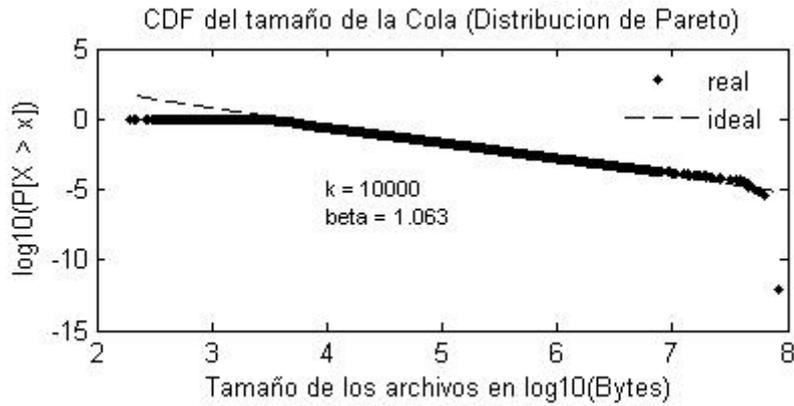


Figura 5.10. Representación gráfica de la simulación con pendiente de cola de 1,1

Si a partir del valor de k, que es el valor que marca el comienzo de la cola de Pareto, la pendiente es menor entonces quiere decir que hay un mayor número de archivos dentro de dicha área, véase figura 5.12.

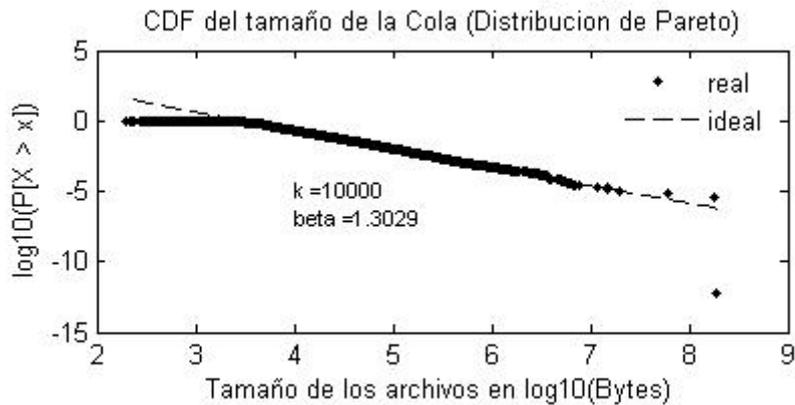


Figura 5.11. Representación gráfica de la simulación con pendiente de cola de 1,3

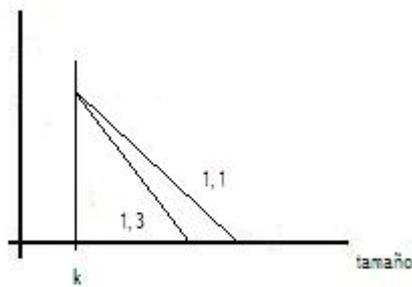


Figura 5.12. La cola con pendiente menor tiene mayor área

En cuanto al estudio de ambas generaciones respecto al tipo de archivos, los resultados obtenidos son similares a los mostrados en la tabla 5.4, si bien sería interesante destacar que el hecho de que existan más documentos en la cola de Pareto se traduce en que habrá mayor número de documentos con tamaño mayor que k , por tanto, significa que el tamaño medio de los archivos es mayor para el caso del índice de cola 1,1.

5.2.4 CORRELACIÓN

En principio, podría pensarse que el caso más habitual, sea aquel en el que la correlación entre el tamaño y su popularidad es cero, es decir, son independientes. No obstante, puede darse la circunstancia que se desee simular casos en los que los archivos de mayor tamaño tengan más popularidad, en ese caso se habla de correlación positiva, o bien que archivos de menor tamaño tengan más popularidad, se habla de correlación negativa (que suele ser lo más habitual). Teniendo en cuenta que ya se ha realizado la simulación para correlación cero, ver figura 5.2, a continuación se mostrarán los resultados gráficos, figuras 5.13 y 5.14, correspondiente a correlación positiva y correlación negativa.

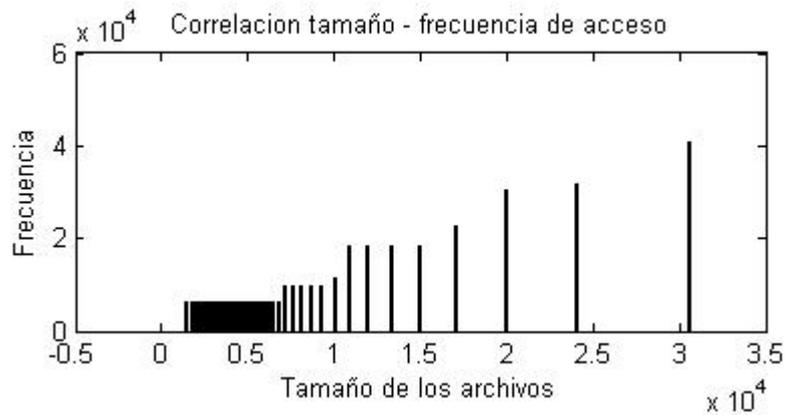


Figura 5.13. Representación gráfica de la simulación con correlación positiva

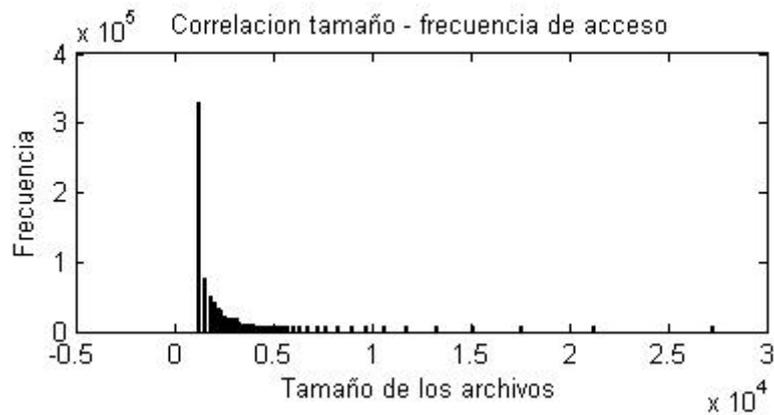


Figura 5.14. Representación gráfica de la simulación con correlación negativa

Los datos estadísticos mostrados en la tabla 5.7 realizan una comparativa entre las generaciones de muestras con correlación positiva y correlación negativa, de nuevo los parámetros de salida se ajustan a los valores deseados. La única cuestión que se puede destacar es el hecho que en el caso de correlación positiva (archivos de tamaño mayor tienen mayor probabilidad) el porcentaje de archivos en cola es mayor, algo bastante lógico. En cuanto a los tipos de archivos, los resultados son similares a los mostrados en la tabla 5.4.

	Generación de Muestras 1	Generación de Muestras 2
Correlación	1	-1
<i>Resultados obtenidos</i>		
Porcentaje de documentos distintos	32,91%	32,83%
Porcentaje de <i>one-timers</i>	70%	70%
Número total de muestras	911.538	913.675
Pendiente de Zipf	0,72983	0,74183
Índice de cola-pesada	1,1906	1,184
Correlación	1	-1
Porcentaje de documentos en la cola-pesada	22,50%	19,68%
Media de la distribución logarítmica normal (μ)	5540,62	5535,35
Varianza de la distribución logarítmica normal (σ)	4580,95	4776,64

Tabla 5.7. Datos estadísticos de simulaciones con correlación positiva y negativa respectivamente

5.2.5 MODO DE LA PILA

Al realizar la generación de muestras con modo de pila estático, el resultado de los parámetros naturales de dicha generación son exactamente los mismos que para el caso de de pila dinámica, es decir, se obtienen los mismos valores para la pendiente de Zipf, pendiente de cola de Pareto, número total de muestras,... (Véase tabla 5.2). Y es que el modo de la pila únicamente produce cambios en el reordenamiento de las peticiones generadas, no afectando en ningún momento a los parámetros intrínsecos de las peticiones generadas. En cuanto a los tipos de archivos, ocurre la misma circunstancia.

Por tanto, sólo será en la gráfica de la localidad temporal donde se pueda apreciar ciertas diferencias. Puesto que ya se mostró dicha gráfica para el modo de pila dinámico en la figura 5.6, se muestra el caso de modo de pila estático en la figura 5.15.

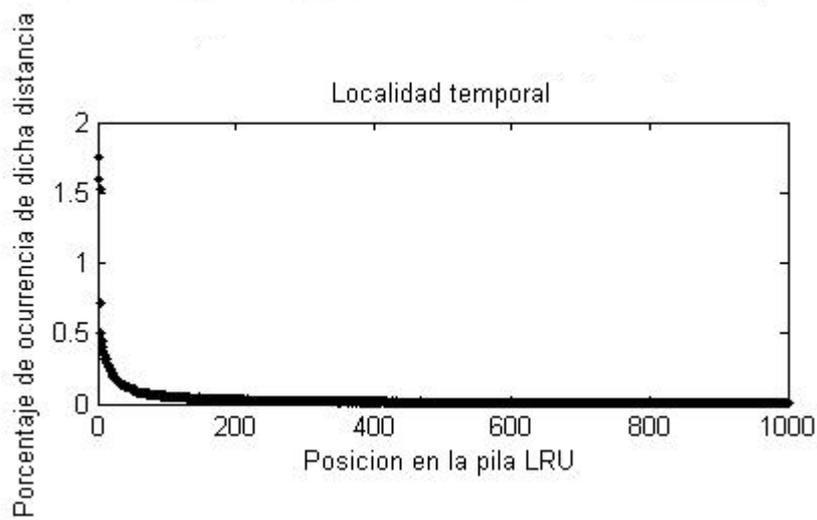


Figura 5.15. Representación gráfica de la simulación con modo de pila estático

5.3. Mezclas de distintos tipos de archivos

Como ya se comentó en el capítulo 4, la aplicación permite la mezcla de diferentes patrones de muestras generadas, esta mezcla, en principio, se realiza de modo aleatorio, pero se deja la posibilidad de poder añadir distintas distribuciones para su mezcla. En la tabla 5.8 se muestran los parámetros utilizados para la mezcla cuya representación gráfica resultante puede observarse en la figura 5.16.

Como puede apreciarse de los resultados obtenidos, la mezcla no ha seguido ningún razonamiento común sino que se han mezclado aleatoriamente, cada uno con sus características propias, y el resultado generado es consecuencia de dicha mezcla.

	Aplicación	Audio	Imágenes	Mensajes	Texto	Vídeo
Número de peticiones	100.000	100.000	200.000	50.000	100.000	80.000
Doc. Distintos (%)	30	30	30	30	33	28
One-timers (%)	70	70	70	70	70	72
Pendiente de Zipf	0,7	0,73	0,75	0,8	0,7	0,75
Índice de Pareto	1,2	1,18	1,2	1,25	1,2	1,2
Doc. En cola (%)	20	21	20	20	16	23
K (Bytes)	10.000	10.000	10.000	10.000	10.000	10.000
Media (Bytes)	7.000	4.000	2.000	6.000	2.000	9.000
Varianza (Bytes)	11.000	2.000	500	3.000	1.000	4.000
Tamaño de la pila	1.000	1.000	1.000	1.000	1.000	1.000
Correlación	0	0	0	0	0	0
Modo de la pila	Estático	Estático	Estático	Estático	Estático	Estático

Tabla 5.8. Parámetros de entrada para la mezcla de documentos

No obstante, se puede apreciar como algunos parámetros que eran muy similares, se han mantenido aún después de la mezcla, como es el caso de la pendiente de Zipf, el índice de cola pesada, así como el porcentaje de documentos en dicha cola, en cambio, otros parámetros como la correlación entre el tamaño de los archivos y su frecuencia ha resultado distinta al común de los parámetros de entrada y es que al variar la media y la varianza de los distintos tipos de archivos se ha provocado un cambio en los tamaños, produciendo la aparición de dicha correlación, a su vez, el modelado del cuerpo de la distribución de tamaño ya no sigue una distribución logarítmica normal tal y como se puede apreciar en la gráfica de la función de probabilidad acumulada del tamaño. Todos estos parámetros resultantes se muestran en la tabla 5.9.

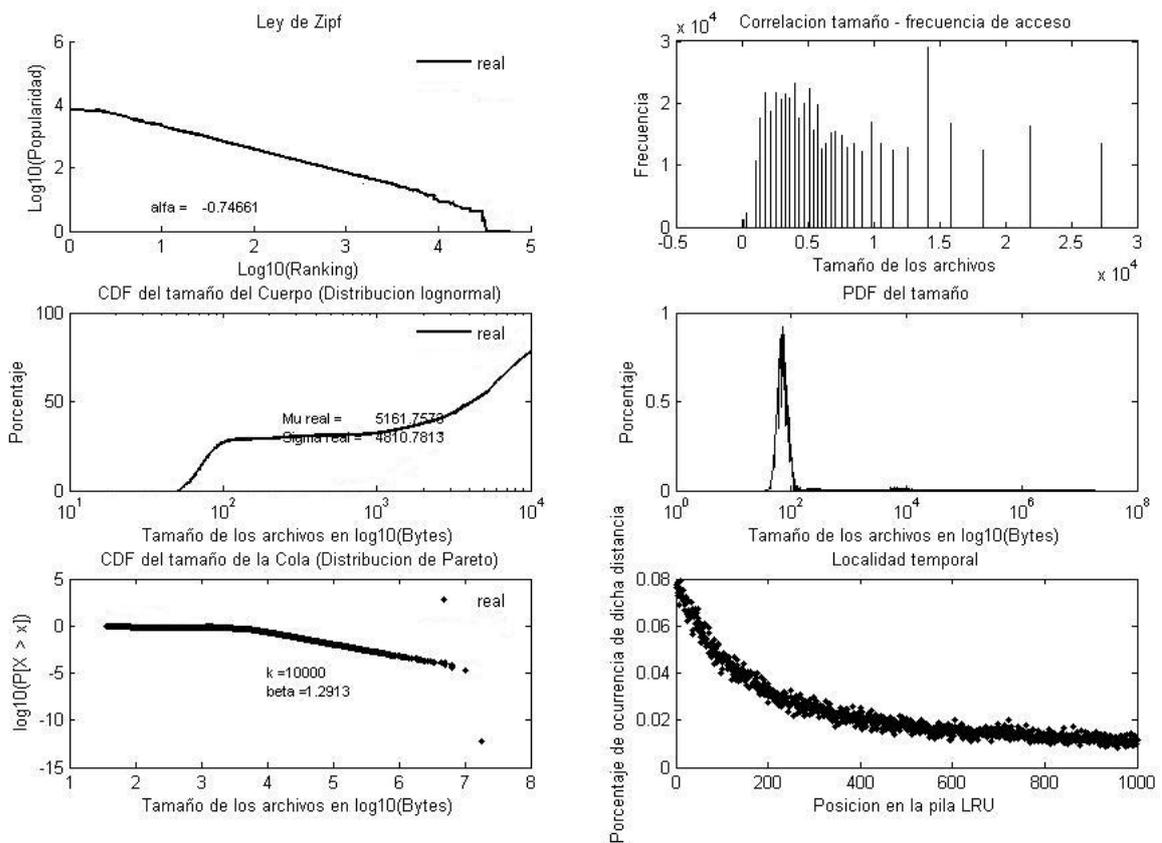


Figura 5.16. Resultados gráficos de la mezcla de distintos patrones de muestras

	Muestras tras la mezcla
Número total de muestras	580.182
Porcentaje de documentos distintos	10,34 %
Porcentaje de <i>one-timers</i>	45 %
Pendiente de Zipf	0,7466
Índice de cola-pesada	1,2913
Porcentaje de documentos en la cola-pesada	21,87 %
Media de la distribución logarítmica normal (μ)	5161,75
Varianza de la distribución logarítmica normal (σ)	4810,78
Correlación	1

Tabla 5.9. Parámetros resultantes de la mezcla de simulaciones de diferentes tipos de archivos

En cuanto a los tipos de archivos los resultados han sido los siguientes:

Tipo aplicación:	93068 archivos	➔	16,04%	con un tamaño medio de 11,68 KB
Tipo audio:	90816 archivos	➔	15,61%	con un tamaño medio de 11,19 KB
Tipo imágenes:	184402 archivos	➔	31,78%	con un tamaño medio de 10,99 KB
Tipo mensaje:	47713 archivos	➔	8,22%	con un tamaño medio de 10,21 KB
Tipo texto:	90588 archivos	➔	15,61%	con un tamaño medio de 14,31 KB
Tipo video:	73595 archivos	➔	12,68%	con un tamaño medio de 12,11 KB

En general, cuando se realizan mezclas de distintos patrones de muestras, no se pueden extraer grandes conclusiones estadísticas puesto que no hay ninguna distribución de mezcla establecida, pero sí que puede ser una herramienta bastante útil para el análisis de políticas de reemplazos en caché, por ejemplo, o para cualquier otro estudio con cachés.

CAPÍTULO 6: Conclusiones y líneas futuras

En este último capítulo, se muestran las conclusiones obtenidas tras la realización de este Proyecto Fin de Carrera y se resumirá brevemente el desarrollo de la aplicación. Posteriormente, se incluyen posibles líneas futuras de trabajo a seguir.

Se ha desarrollado un generador de muestras de tráfico cuyo objetivo principal es que sirva de herramienta para posteriores aplicaciones de estudio de *caches*. De este modo, se van a poder crear muchas muestras de tráfico que permita que otras aplicaciones puedan funcionar con un tráfico con determinadas características concretas.

Esta aplicación parte de una serie de parámetros de entrada que permiten modelar las características propias de distintas muestras de tráfico.

- Los “*one-timers*” son aquellos documentos que sólo son solicitados una vez, son parte fundamental de todo estudio de tráfico puesto que suponen cerca del 60% del total de documentos. Su implementación es prácticamente inmediata puesto que es directamente proporcional al número de muestras totales y al número de archivos distintos.
- La popularidad de los documentos sigue una distribución “como la de Zipf” con un exponente cercano a 1. Su implementación se realiza partiendo del modelo matemático y junto con los “*one-timers*” permite la generación de la popularidad de todos los documentos.
- El tamaño de los documentos viene marcado por dos tipos de distribuciones: aquellos tamaños menores que el parámetro de entrada k , sigue una

distribución logarítmica normal, y modelable mediante la generación de números aleatorios que siguen tal distribución; y aquellos tamaños mayores que k siguen una distribución de cola pesada (de Pareto), también modelable mediante números aleatorios que sigan dicha distribución. Finalmente, se unen ambas distribuciones suavizando el tránsito de una distribución a la otra.

- La correlación entre el tamaño de los documentos y su popularidad es cada día más importante, además supone un cambio brusco dependiendo de la opción escogida. Para su implementación, tras la generación de la popularidad de los documentos y posteriormente su tamaño, se realiza una técnica de mapeo para introducir la correlación.
- La localidad temporal es un concepto muy común en todo tipo de tráfico, y es que si se realiza la petición de un documento, es posible que en un breve instante de tiempo vuelva a ser requerida. La implementación de la misma se realiza una vez generada todas las muestras y supone una reordenación de las mismas según el modelo de pila finita LRU. Se distingue a su vez en dos modelos de pila: dinámica, se recalcula probabilidades para cada iteración; estática, se calculan las probabilidades una sola vez al comienzo de la generación.

Los parámetros de entradas solicitados para la generación de muestras pueden pertenecer a varias de las características de tráfico descritas anteriormente, por tanto, para adaptar las muestras de tráfico generadas a todas esas características se ha llegado a un compromiso entre ellas. De modo que al realizar las generaciones de muestras y proceder a su posterior análisis, los resultados que se obtienen están muy cerca de los parámetros ideales introducidos en la aplicación, si bien ha sido en la distribución logarítmica normal del tamaño de los archivos donde se ha realizado el mayor sacrificio para alcanzar el compromiso citado anteriormente.

Se han realizado generaciones variando todos y cada uno de los parámetros de entrada, verificando como los resultados obtenidos coinciden con los resultados esperados.

La aplicación permite que se puedan especificar los parámetros de entrada para la muestras en general o bien para cada tipo de archivos, y a su vez permite que se pueda especificar el porcentaje de archivos de cada tipo que desea generar. Y también, la posterior mezcla de muestras una vez que se han generado. Esta mezcla, en principio, se realiza de forma aleatoria.

Las líneas futuras a seguir en este proyecto Fin de Carrera podrían ser:

- Desarrollo de distintos algoritmos de mezcla de los archivos de muestras generadas. Se ha dejado la posibilidad de introducir nuevas distribuciones para las mezclas de las muestras ya generadas.
- Optimización de los algoritmos de generación de muestras, principalmente en cuanto a rapidez para la generación de las mismas.
- Introducción de nuevos tipos de tráfico o subtipos de tráfico dentro de cada tipo, esto es, se podría añadir que distinguiera entre tipos específicos, por ejemplo, que diferenciara entre una imagen *jpf* o *gif*.

BIBLIOGRAFÍA:

- [1] G. Abdulla, E. Fox, M. Abrams, y S. Williams, “WWW Proxy Traffic Characterization with Application to Caching” Technical Report TR-97-03, Computer Science Department, Virginia Tech., Marzo 1997.
- [2] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, y Tai Jin, “Evaluating Content Management Techniques for Web Proxy Caches” in 2nd Workshop on Internet Server Performance, Atlanta, Georgia, Mayo 1999.
- [3] A. Mahanti y C. Williamson, “Web Proxy Workload Characterization” Technical Report, Department of Computer Science, University of Saskatchewan, Feb. 1999.
- [4] C. Roadknight, I. Marshall y D. Vearer, “File Popularity Characterization” in Proceedings of the 2nd Workshop on Internet Server Performance (WISP 99), Atlanta, Georgia, Mayo 1999.
- [5] S. Jin y A. Bestavros, “Sources and Characteristics of Web Temporal Locality”, Proceedings of MASCOTS’2000, pp.28-35, San Francisco, CA, Agosto 2000
- [6] A. Mahanti, Web Proxy Workload Characterization and Modelling, M.Sc. Thesis, Department of Computer Science, University of Saskatchewan, Septiembre 1999.
- [7] V. Almeida, A. Bestavros, M. Crovella, y A. Oliveira, “Characterizing Reference Locality in the WWW” in Proceedings of the 1996 International Conference on Parallel and Distributed Information Systems (PDIS 96), pp. 92-103, Dic. 1996.

- [8] M. Arlitt and C. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications" in IEEE/ACM Trans. On Networking, vol. 5, no 5, pp. 631-645, Oct. 1997.
- [9] P. Badford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation" in Proceedings of the 1998 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp.151-160, Julio 1998.
- [10] L. Breslau, P. Cao, L. Fan, G. Phillips, y S. Shenker, "Web Caching and Zipf-like Distributions : Evidence and Implications" in Proceedings of the IEEE Infocom '99 Conference, New York, NY, Marzo 1999.
- [11] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Posible Causes" in IEEE/ACM Transactions on Networking, vol. 5, no.6, pp.835-846, Dic. 1997.
- [12] M. Busari y C. Williamson. "On the sensitivity of web proxy cache performance to workload characteristics". In Proceedings of IEEE INFOCOM'2001, pages 1225–1234, Anchorage, Alaska, Abril 2001.
- [13] M. Busari, C. Williamson, "ProWGen: A Synthetic Workload Generation Tool for the Simulation Evaluation of Proxy Caches", Computer Networks, pp. 779-794. Jun. 2002.
- [14] En Internet: <http://www.sc.ehu.es/acwmialj/papers/generadores.pdf>
- [15] En Internet: <http://www.item.ntnu.no/~poulh/GenSyn/gensyn.html>
- [16] T. Berners-Lee, "WorldWideWeb, the First Web Client". Disponible en <http://www.w3.org/People/Berners-Lee/WorldWideWeb.html>
- [17] Referencias a la ley de Zipf: <http://linkage.rockefeller.edu/wli/zipf>

- [18] A. Mahanti, C. Williamson, y D. Eager, “Traffic Analysis of a Web Proxy Caching Hierarchy” IEEE Network, vol. 14, no. 3, pp. 16-23, Mayo/Junio 2000.

- [19] A. Law y W. Kelton, “Simulation Modeling and Analysis”, Segunda Edición, Ed. Mc-Graw-Hill, 1991

- [20] S. Jin, A. Bestavros, “GreedyDual* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams”, Intl’ Journal of Computer Communications, Col. 24, no. 2, pp. 174-183, Febrero 2001.

- [21] Página del proyecto IRCaché con las muestras: <http://www.ircache.net/>

- [22] Página Web del Departamento de Tecnología Electrónica de la Universidad de Málaga, <http://www.dte.uma.es>