SINCRONIZACIÓN

DE ESTADÍSTICAS ENTRE

SERVIDORES Y PROXIES

9 (nueve)

Trabajo de Grado

Director: ***Lic. Javier Díaz***

Grupo: ***María José Catullo***

(Nro.Alumno: 27154/6)

"El futuro de la sociedad e inclusive el de una nación depende del futuro interior e individual de cada persona. Por que tanto a la sociedad como a una nación la componen personas individuales..."

Agradezco a mis padres,a mi marido y al Licenciado Miguel Luengo

Contenido:

Páginas

Páginas

I. Introducción:

El gran crecimiento y popularidad que ha desarrollado Internet en los últimos tiempos, va acompañado por la necesidad de optimizar el ancho de banda disponible y lograr una utilización más eficiente de los servidores.

La concentración de visitas en sitios populares provoca cargas excesivas, pérdida de ancho de banda por transferencias redundantes y además, inconvenientes en los tiempos de respuesta, lo que demanda grandes esperas en la recepción de documentos y provoca altos costos en su uso.

Estos problemas pueden ser aliviados por la migración de copias de documentos populares.

La migración puede hacerse mediante un modelo de distribución en el cuál el servidor controle dónde las copias de documentos son almacenadas, ó mediante un modelo de cache, en el cuál las copias migran automáticamente en respuesta de requerimientos.

El ***modelo de cache ó caching***, es el más popular hoy en día, permitiendo reducir la carga de la red y de los servidores, se podría decir entonces que sin el caching la WWW podría ser víctima de sus propios sucesos.

El caching efectivamente migra copias de documentos populares, desde servidores a clientes, así los clientes tienen poco retardo cuando requieren una URL.

El caching puede ser basado en el cliente o en el servidor de Web. La segunda alternativa es lo que se conoce cómo ***"proxy server"*** ó ***"proxy gateway"***.

Un caching proxy server, colecciona todas las páginas que pasan a través de él, así cuando un usuario pregunta por una página, el proxy server, si la tiene, la retorna de su cache, sin necesidad de acceder al servidor real. De ésta manera se ahorra ancho de banda y se agilizan las respuestas.

Cada requerimiento que arriba a un proxy ó a un servidor, se almacena en archivos de logs. Existen varios tipos de archivos de logs, que se utilizan para auditoría del rendimiento, monitoreo, control de errores, seguridad y estadísticas.

Así, existen actualmente muchos analizadores de logs de Web, que permiten realizar un estudio detallado de un sitio, ayudando a la planificación y a la auditoría de seguridad.

En éste trabajo, se plantea la problemática de no actualización de los logs de servidores cuando está de por medio un servidor proxy, ya que en éste esquema, el log del servidor no refleja la realidad de accesos al mismo.

Esto ocurre, porque cuando un servidor proxy retorna documentos al cliente desde su cache local, dichos movimientos quedan registrados en el log del proxy y no en el log del servidor.

Por éste motivo, la información recolectada por los analizadores de logs no es exacta y se distorsiona, provocando estadísticas no tan precisas.

Como solución, la idea es mantener el archivo de log de accesos del servidor actualizado, para lo cuál se necesita intercambiar información entre logs de proxies y logs de servidores, es decir, lograr una ***sincronización*** de los mismos.

Se proponen aquí dos modelos para resolver la problemática enunciada, y se analizan los pro y los contra de cada uno. Así, se adoptará alguno cómo solución, ó ambos como dos métodos alternativos.

II. Caching:

El fenómeno ***Internet***, ha entrado a formar parte de la rutina diaria de miles y miles de usuarios, provocando una importante demanda de ancho de banda y motivando la necesidad de buscar estrategias para optimizar el uso de la red.

La concentración de visitas en los sitios más populares sobrecarga de manera excesiva los servidores, hay pérdida de ancho de banda por transferencias redundantes y además hay inconvenientes en los tiempos de respuesta, lo que demanda grandes esperas en la recepción de documentos y provoca altos costos en su uso.

El modelo básico ***cliente-servidor*** de Internet, dónde los clientes se conectan directamente con los servidores, tiende a provocar un derroche de recursos, y sumado a esto los servicios de información basados en multimedia, transmisiones de audio/video en tiempo real y el comercio en la red, contribuyen a un alto uso de los enlaces y por consiguiente a una importante reducción del ancho de banda.

El ancho de banda disponible depende de la capacidad de camino entre el cliente y el servidor, el cuál está limitado por la sobrecarga o cuello de botella del enlace y la existencia de tráfico compitiendo por los recursos o congestión.

Podemos decir entonces, que el incremento del ancho de banda, es insuficiente en relación al gran crecimiento de Internet, y por lo tanto existe la necesidad de optimizar el ancho de banda disponible y lograr una utilización más eficiente de los servidores.

Como solución a éstos problemas, existen varias técnicas, una de las usadas es la técnica de Caching *[1][Povey-Harrison]*.

La técnica de caching es simple, consiste en que, cuando un cliente obtiene por primera vez un objeto, se almacena localmente para que los siguientes requerimientos puedan satisfacerse a través de la copia cacheada sin necesidad de conectarse

nuevamente con el servidor, de ésta manera se ahorra ancho de banda.

El caching tiene las siguientes características:

- Es transparente al usuario, ya que éste no necesita conocer la ubicación física del objeto cacheado.

- Está basado en la demanda, asegurando así que los objetos cacheados son los más populares y liberando al administrador de que decida que objetos cachear.

Esta técnica se utiliza en todos los browsers modernos como el Internet Explorer y el Netscape Navigator. Este esquema hace uso del cache local de los clientes, sin embargo no es suficiente, ya que clientes de un mismo servidor no pueden compartir objetos cacheados de otros clientes.

Otro esquema plantea caches compartidos utilizando servidores proxy. De ésta manera cuando un cliente desea obtener un objeto, debe dirigir su requerimiento al servidor proxy, quién obtendrá el objeto y lo pasará a través de un ***firewall***.

Una de las desventajas de éste esquema, es que a mayor cantidad de usuarios, el cache del servidor tendría que ser lo suficientemente grande, cómo para mantener la información necesaria.

El servidor proxy debe refrescar su cache en períodos de inactividad de los clientes. Cada documento en el cache tiene un tiempo de expiración, pero un cache puede expirar objetos basándose en la configuración del sitio, en la carencia de accesos, etc.

Existen varias políticas de borrado de documentos de un cache que hacen a la performance del mismo *[16]*.

Dichos procedimientos de borrado, pueden ejecutarse:

Según la demanda: se ejecuta la política de borrado cuando el tamaño del documento requerido, excede el tamaño disponible en el cache.

Periódicamente: se ejecuta la política de borrado cada "X"cantidad de tiempo.

Según la demanda y periódicamente: que sería una combinación de las dos anteriores, es decir se ejecuta la política de borrado al final de cada día y según la demanda.

Cuando es esencial que un objeto esté actualizado, se debe contactar al servidor remoto por cada requerimiento GET.

Para solucionar éste problema, el protocolo HTTP incluye el header *If-Modified-Since* al requerimiento, haciendo posible un requerimiento GET *condicional.*

Mediante éste header que contiene el día y hora en que el documento del cliente fue modificado por última vez, en algunos casos, puede obtenerse información de encabezado del documento en lugar del documento en sí mismo.

Si el documento no fue modificado a partir del día y hora contenidos en el header, el documento no se retorna desde el servidor remoto, sólo se retorna información de encabezado importante del documento, tal cómo nueva fecha de expiración. Si por el contrario, el documento fue modificado luego del día y hora del header, se retornará el documento como si fuese un GET tradicional.

Qué objetos se pueden cachear ?:

Algunos agentes de cache, no tienen permitido cachear objetos provenientes de servidores seguros (los que utilizan el protocolo SHTTP), con lo que se decrementa el riesgo de seguridad. Además, algunos browsers están configurados para no cachear documentos seguros, ya que de lo contrario, esos documentos serían vulnerables a robos o accesos no autorizados.

Existen algunas reglas para determinar qué documentos se pueden cachear:

- No pueden ser cacheados documentos retornados usando POST o que requieran autorización.

- No pueden ser cacheados los documentos con pragma: ***no cache***.

- Los documentos con tiempo de expiración ***"ahora"***, con un día de expiración vencido ó un día de expiración ilegal, tampoco se pueden cachear.

Las cookies no trabajan bien con proxy cache, por lo tanto los documentos con el header *Set-Cookie* no son cacheados.

Estructuras de caching:

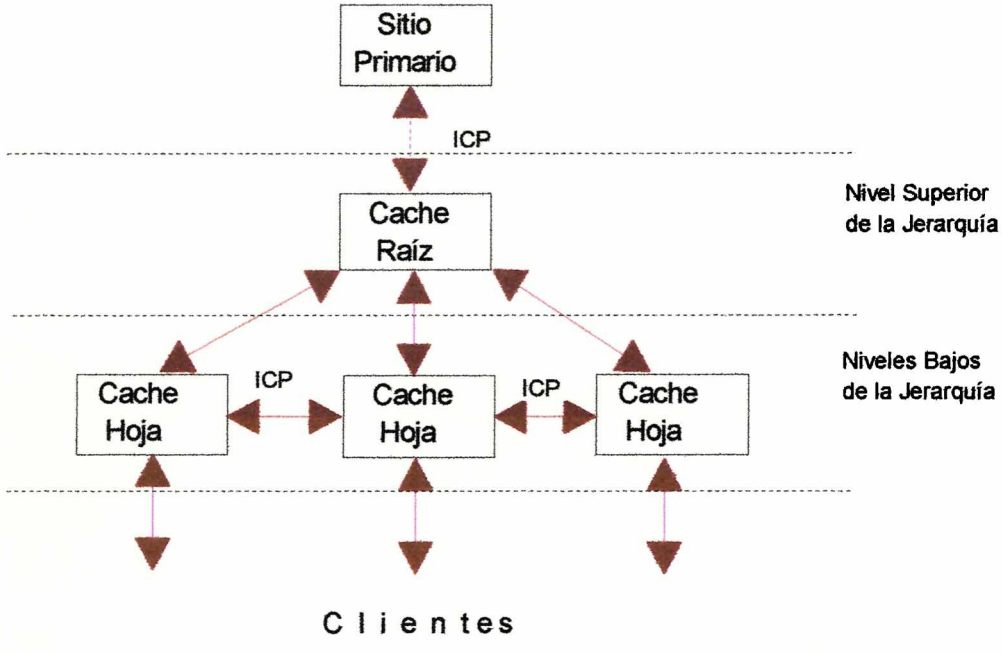
En una WAN, existen tres estructuras de chaching:

- **Caching Jerárquico**
- **Caching Distribuido**
- **Caching Cooperativo**

Caching Jerárquico:

El ***caching jerárquico*** permite distribuir la carga de los servidores de los sitios más populares y reducir tiempos de acceso. Así se puede minimizar la demanda de ancho de banda y evitar la transferencia redundante de información.-

En éste esquema, el nodo raíz es el responsable de resolver todos los pedidos no resueltos por los otros caches de la red. Para evitar la carga en el mismo si los requerimientos de los caches son muy grandes, los nodos hermanos se utilizan para hacer balanceo de carga.-



Cómo trabaja el caching jerárquico?:

Los clientes piden un documento a los caches hojas usando un protocolo estándar HTTP proxy. Si el servidor que recepcionó el pedido no lo puede resolver con su información local, consulta al padre y hermanos usando un protocolo conocido como ICP (Internet Cache Protocol). Además el cliente envía un mensaje de acceso hacia el port echo UDP del sitio primario.-

El servidor espera recibir un ***hit***, en caso de encontrar el documento requerido o un ***miss***, en caso contrario. Si todos responden con un miss el proceso continúa recursivamente con los niveles superiores de la jerarquía, consultando a padres y hermanos hasta que el pedido sea satisfecho ó se alcance el nodo raíz.-

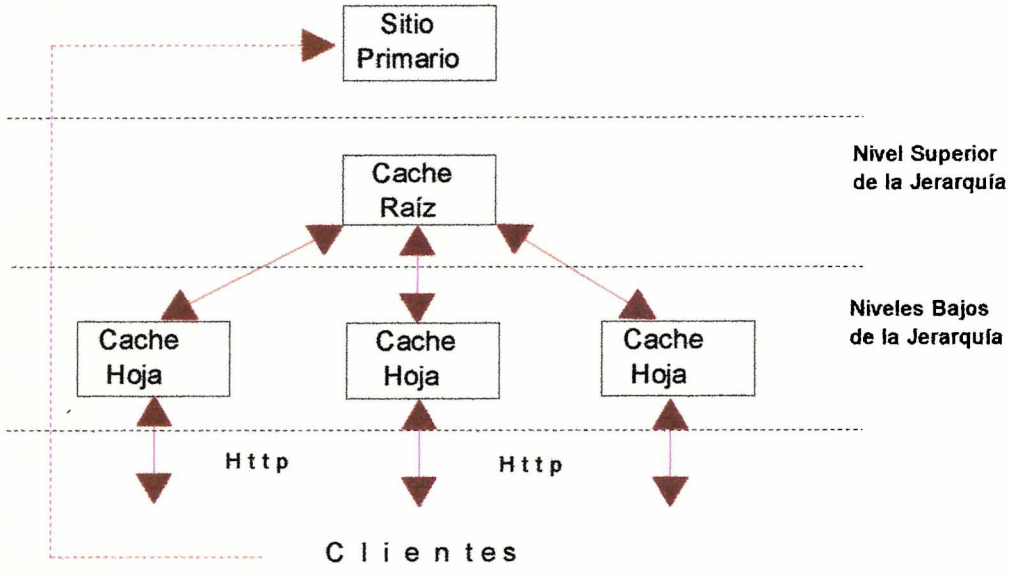
El servidor de la jerarquía recupera el documento del que primero conteste. Si el sitio primario responde antes que los caches padre o hermanos, el documento se recupera del sitio primario.-

Las desventajas del caching jerárquico son: el espacio requerido en disco para los niveles superiores, ya que para obtener una alta tasa de aciertos y para que haya reducción de transferencias redundantes el cache local debería contener la intersección de los contenidos de todos sus nodos hijos, y además cada nodo debe conocer la ubicación de cada uno de sus hermanos para dirigir los requerimientos directamente a ellos y evitar el problema de la carga excesiva de los servidores de niveles superiores.-

Caching Distribuido:

El esquema de ***caching distribuido*** resuelve el problema de que los niveles superiores de la jerarquía necesiten grandes espacios de almacenamiento, ya que en éste caso, los nodos hojas son los responsables de retornar los documentos, mientras que los caches de niveles superiores sólo mantienen punteros a los documentos en lugar de copias de los mismos.-

En el cache distribuido no se consulta a los hermanos para ver si ellos cachearon el objeto, cada padre **"aprende"** del contenido del cache de sus hijos, de ésta manera para consultar el contenido de sus hermanos, se consulta al padre.-



Cómo trabaja el caching distribuido ?:

Cuando un cliente desea obtener un documento, lo requiere a un servidor hoja, si éste no lo tiene en su cache local, consulta al padre para ver si tiene información de dónde está la copia cacheada, si tiene información, no retorna el objeto sino un puntero a dicho objeto, y si no tiene dicha información sigue consultando recursivamente hasta la raíz.-

De alcanzar la raíz y no encontrar información del objeto, se responde con un ***miss*** al cache hoja que lo solicitó y éste solicita el documento al sitio primario. Pero debe informar a los nodos de nivel superior que el objeto fue cacheado, para lo cual se utiliza un mecanismo llamado ***advertencia***.

Este mecanismo es el mismo que se usa para indicar cuando se elimina un objeto de los caches hojas.-

Las desventajas que encontramos en éste esquema, es la carga de los servidores hoja. Además, los mismos deben atender requerimientos de sus propios clientes y de los otros caches de la red, y esto puede desembocar en que dicho tráfico se transforme en un "cuello de botella".

Así se debería considerar la alternativa de reducción de advertencias y permitir rechazos por parte de los caches de requerimientos cuando están sobrecargados.-

Caching Cooperativo:

El esquema de ***caching cooperativo***, plantea agregar otro nivel de almacenamiento a la jerarquía para reducir la carga de los servidores, permitiendo a un cliente acceder a los bloques cacheados por otros clientes.-

Dicho almacenamiento es lo que se conoce como ***cache cooperativo***, y tiene la particularidad de estar distribuido a través de los clientes, por lo que comparte la memoria física de los mismos. Por ésta razón, cada cliente deberá controlar sus acciones.-

Acá empieza a jugar el término de ***manejadores***, que serían los encargados del control de ese cache cooperativo, es decir, encargados de la decisión de qué objetos cachear, esto es utilizando lo que se denomina política de reemplazo, y de cómo ubicar los mismos en dicho cache.-

Cómo funciona la política de reemplazo ?:

La ***política de reemplazo*** entra en juego cuando un cliente quiere reemplazar un objeto de su cache local. Dicho objeto puede descartarse o pasarse al cache cooperativo de otro cliente. La política es la que decide qué hacer en dicho caso.-

Para descartar el objeto debe considerar que no es tan importante y si se envía a un cliente destino, éste deberá tener un objeto de menor importancia en su cache cooperativo ó en su cache local.-

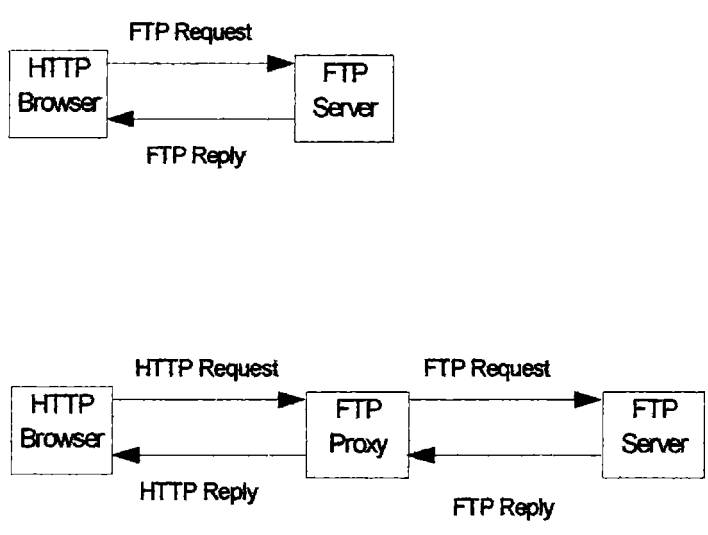
Esto demuestra una vez más, que de no existir una coordinación entre los clientes, se puede llegar a dar el caso de la existencia de varias copias del mismo objeto en distintos caches.-

III. Servidor Proxy:

Un ***Proxy Server***, es una clase de *gateway* que habla http para el browser, pero ftp, gopher y cualquier otro protocolo para el servidor. Este acepta requerimientos http y los traslada dentro de requerimientos ftp, así el browser no tiene que entender más que http. *[2][A.Tanenbaum]*

El proxy server puede ser un programa corriendo sobre la misma máquina que el browser, ó puede ser una máquina independiente donde sirva a muchos browsers.

La figura muestra un browser que habla ftp y una que usa un proxy.



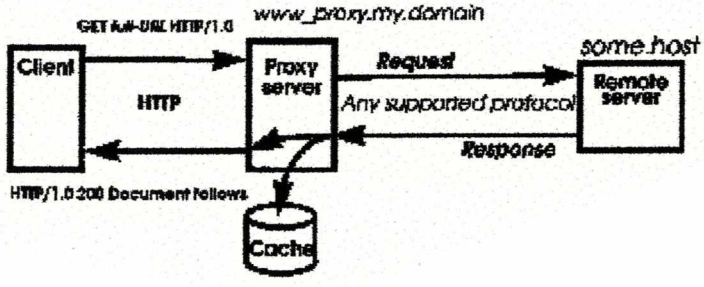
Los usuarios, pueden configurar sus browsers, con proxies para protocolos que el browser no habla, de ésta manera, se incrementa el rango de fuentes de información para el cuál el browser tiene acceso.

Además, los proxies servers, tienen otras funciones importantes, tales como el ***caching***. La capacidad de cachear documentos, hace a un proxy atractivo.

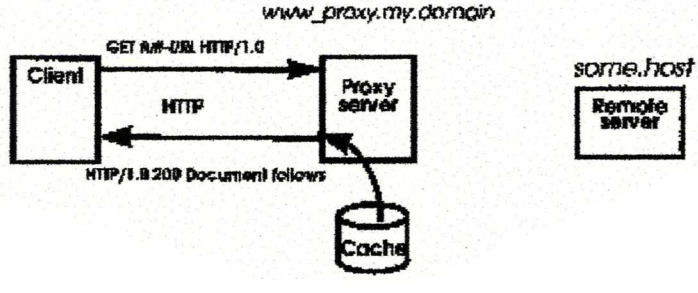
La configuración de un proxy server es fácil, y los programas de Web más populares ya incluyen soporte para proxy.

Un caching proxy server, colecciona todas las páginas que pasan a través de él, así cuando un usuario pregunta por una página, el proxy server chequea para ver si la tiene.

En ésta figura se ve cómo el documento es recuperado del servidor remoto y almacenado en el proxy server, para su posterior uso.



En ésta figura el documento se recupera del cache del proxy, sin necesidad de acceder al servidor remoto:



El programa cliente del usuario habla con el proxy server en lugar de con el servidor real. El proxy server evalúa el requerimiento y decide si pasa o no. Si el requerimiento es aprobado, el proxy server habla con el servidor real, y procede a retransmitir el requerimiento del cliente al servidor y a enviar la respuesta al cliente.

Un proxy server actúa cómo servidor y cliente. Es un servidor cuando acepta requerimientos http desde clientes conectados a él, y actúa cómo cliente cuando se conecta a servidores remotos para recuperar documentos.

Un proxy server completo podría hablar todos los protocolos de Web, desde el más importante comenzando por http, Gopher, Wais y NNTP.

Los proxies también pueden manejar un único protocolo de red de internet como http, pero un cliente de Web, tendría que acceder a otro proxy server para manejar los protocolos restantes.

El proxeado no requiere hard especial, aunque algunos servicios requieran soft especial.

Los sistemas proxy son efectivos solo cuando son usados con métodos de restricción de tráfico a nivel IP, entre los clientes y los servidores reales. Si hay una conectividad entre el cliente y servidor a nivel IP, el cliente puede bypasear el sistema proxy.

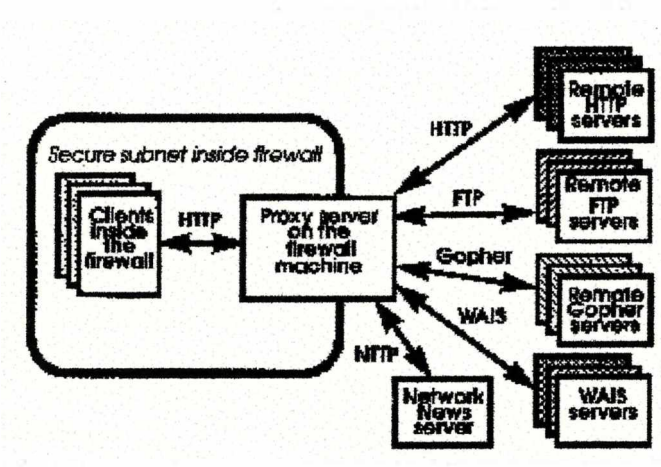
El uso primario de un proxy, es permitir acceso a la Web, desde adentro de un firewall sin comprometer la seguridad. Un proxy típicamente corre en una máquina firewall.

Un firewall es un sistema de seguridad que restringe accesos, permitiendo entradas / salidas sólo de paquetes provenientes de una máquina en particular, en éste caso el servidor proxy. *[3][O'Reilly]*

El proxy espera el requerimiento desde adentro del firewall, envía el requerimiento al servidor remoto fuera del firewall, lee la respuesta y la retorna al cliente. Así en el caso de sub-redes cerradas se puede configurar al grupo para que use caching proxy server, con lo que muchos documentos serán recuperados desde la cache local, reduciendo de ésta manera los costos de tráfico.

El caching es más efectivo sobre el proxy server que sobre cada cliente, ya que se tendría sólo una copia cacheada salvando espacio en disco, pero también es útil para saber con que frecuencia es referenciado cada documento por múltiples clientes, aportando información para algoritmos de predicción y estadísticas.

El siguiente esquema es una configuración de un proxy server, corriendo en una máquina firewall, y permitiendo acceso completo a Internet. *[4][Luotonen-Altis]*



Además, es posible usar caching proxies jerárquicamente, así los caching proxies cachean documentos desde otros caching proxies. En éste caso los caches pueden ser identificados cómo caches de primer nivel, segundo nivel y así siguiendo.

Ejemplos de caching proxies son: CERN proxy server, DEC SRC gateway, UNIX HENSA, Hyper-G en servidores locales.

Ventajas del Proxeado:

- *Los servicios proxy, permiten acceso a internet 'directamente':*

Con los servicios proxy, el cliente piensa que está interactuando directamente con los servicios de Internet.

Los servicios proxy permiten a los usuarios acceso a los servicios de Internet, pero no permiten el paso de paquetes directamente entre los usuarios del sistema e Internet.

- *Los servicios proxy son buenos para registrar log:*

Permite que la registración de los logs se realice de manera efectiva, ya que por ejemplo en lugar de registrar todos los datos transferidos, logea solamente la emisión del comando y la respuesta recibida por el servidor. este resultado es más pequeño y más útil.

Desventajas del Proxeado:

- *Los servicios proxy se retardan en comparación con los servicios no proxy:*

Generalmente el retardo depende de cómo este diseñado el servicio por el proxy

- *Los servicios proxy pueden necesitar diferentes servidores para cada servicio:*

Se puede necesitar un proxy server diferente por cada protocolo. Instalar y configurar todos esos servidores puede ser un trabajo pesado.

- *Los servicios proxy, generalmente requieren modificaciones para clientes, procedimientos ó ambos:*

Excepto para unos pocos servicios diseñados para proxeado, los servidores proxy requieren modificaciones para el cliente y/o los procedimientos.

- *Los servicios proxy, pueden complicar el trabajo de otros servicios:*

Por ejemplo, un servicio como el *talk* tiene dificultades con un proxy.

- *Los servicios proxy, no protegen de todos los protocolos débiles:*

Cómo solución a los problemas de seguridad, la realidad del proxy, es determinar que operaciones en un protocolo se hacen. No todos los protocolos proveen una manera fácil de hacer esto.

IV. Estadísticas en la Web:

Con el explosivo crecimiento en el número de usuarios de la Web, surge la necesidad de caracterizar quiénes usan ó acceden a la Web y qué páginas son las más visitadas.

Por ejemplo, anunciantes y vendedores podrían querer clasificar el número de páginas Web, reflejando cuán frecuentemente son accedidas y por cuánto tiempo son observadas, ó los proveedores podrían usar los datos demográficos de la Web para destinar su audiencia.

Existen dos maneras complementarias de aprender quienes usan la Web y cómo la usan, ésto es mediante encuestas a los usuarios ó analizando los archivos de logs que se generan on-line con los datos de requerimientos ó respuestas que pasan a través de los servidores.*[6][Abrams-Williams]*

Encuestas:

Los usuarios son encuestados voluntariamente a través de mail ó telefónicamente, con una serie de preguntas que ayudan a proveer información demográfica sobre quiénes usan la Web.

Con las encuestas, se plantean algunos problemas:

- Están sujetas a la selección de encuestados.
- Los encuestados podrían dar respuestas inexactas ó falsas.
- El resultado está sujeto a las preguntas formuladas.

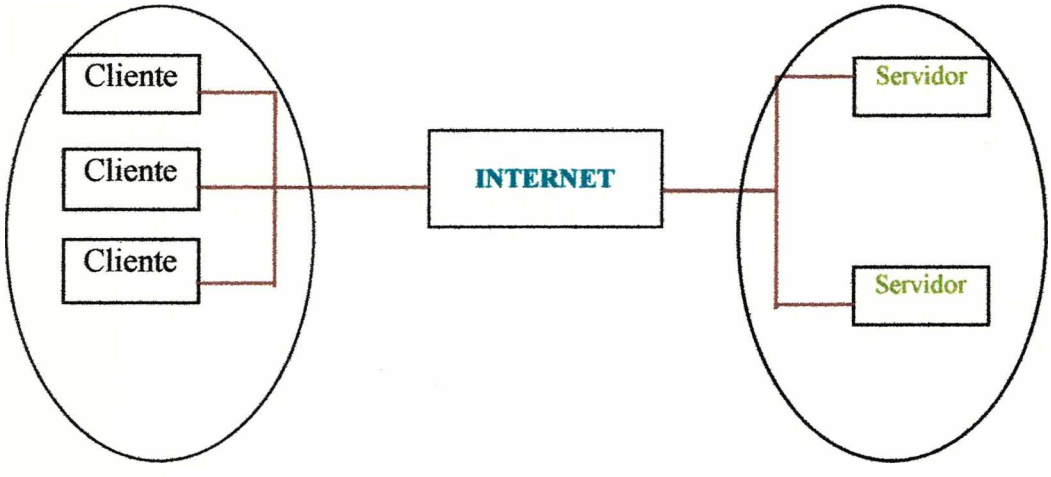
Con los archivos de logs, se pueden complementar las encuestas ayudando a validar la colección de datos recabados con las mismas. Por ejemplo, si se encuesta sobre el tiempo que usan los sitios *.com* de la Web, ó sobre cuántas horas por semana usan la Web, las respuestas pueden ser comparadas con los requerimientos en el archivo de logs, para esos usuarios.

Archivos de Logs:

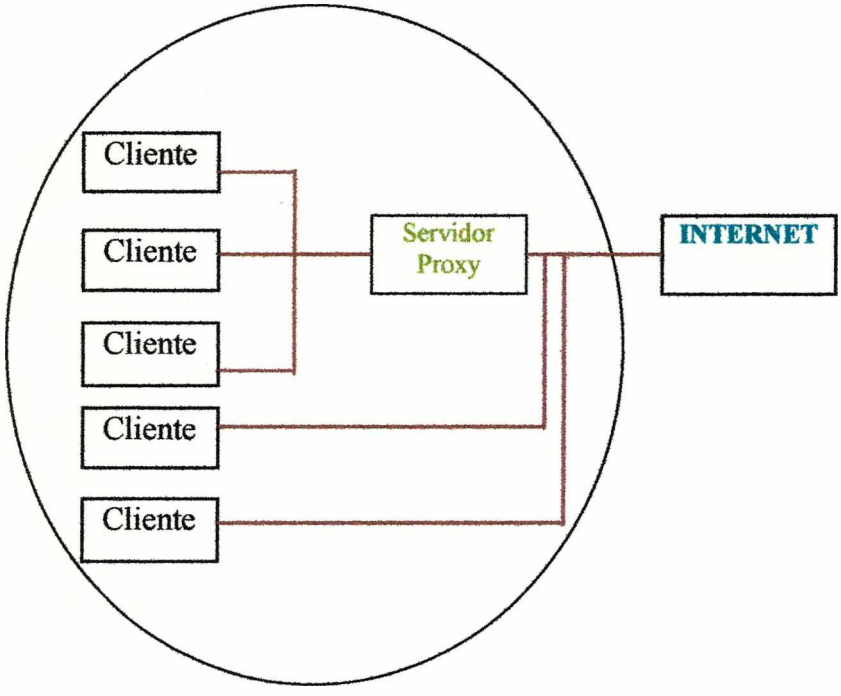
Los archivos de logs, representan un registro casi perfecto, de los requerimientos hechos al servidor de Web o proxy, que se envían a través de la red, ó se tipean dentro de un browser de Web.

Como registran las actividades del servidor, son útiles para monitoreo y resolución de errores.

En un esquema simple, cada vez que un cliente envía un requerimiento, envía un paquete HTTP desde el cliente al servidor, llamado campo URL (Universal Resource Locator), del requerimiento, el servidor retorna uno ó más paquetes conteniendo una respuesta ó un código de error.



La figura se complica cuando las organizaciones usan servidores proxy, ya sea para trabajar como firewalls, caches o ambos. Un firewall, puede prohibir o limitar acceso a internet. Un cache captura copias de algunas respuestas de requerimientos, lo cual reduce tiempo de espera y ancho de banda, si el requerimiento se satisface de la copia cacheada.



Existen cuatro lugares posibles para coleccionar logs, correspondientes a cuatro categorías de loging: en el ***servidor***, en el ***proxy***, en el ***cliente*** y en la ***red***.

Logs del Servidor:

Los archivos de logs del servidor, es la forma más usada. Estos producen información sobre los requerimientos de un sitio Web de clientes de cualquier parte del mundo. Los servidores de Web, pueden coleccionar en un log, requerimientos de documentos por cada HTTP GET, POST ó HEAD que recibe. Por ejemplo, NCSA *http*, colecciona 4 logs: un ***access log***, un ***agent log***, un ***error log*** y un ***referer log***.

- ***Access log:*** en formato común CERN y NCSA, es usado frecuentemente por herramientas para analizar servers, y contiene el nombre del host de clientes remotos, DNS o dirección IP, día y hora de recepción del requerimiento, la URL requerida, el código de estado de la respuesta del servidor y el número de bytes transferidos.

- ***Error log:*** contiene una lista de mensajes de error que el servidor y scripts ejecutables generan, e identifica la hora en que el servidor se inicia ó finaliza.

- ***Agent log:*** identifica el browser de Web usado en cada requerimiento http recibido por el servidor, algunas veces el browser de web también identifica en el string, el tipo de computadora usada (ej: "Mozilla/2.0b6a(WinNT;I)")

- ***Referer log:*** contiene información sobre los links, entre páginas Web y documentos locales.

Hay limitación en la información que los logs de servidores proveen. Primero, muchos de los requerimientos pueden venir de servidores proxy. Por lo tanto, el host name de los clientes puede ser anónimo, porque el log del server contiene el host name del proxy. Segundo, el campo nombre del usuario, si está presente en el log, puede contener información incorrecta, porque el protocolo *ident* que corre sobre el cliente que identifica el nombre del usuario, puede reportar información incorrecta. Tercero, el administrador de Web, puede poner en *off*, la opción de coleccionar el host name del usuario y el nombre del usuario por varias razones: el protocolo *ident*, puede reducir performance ó por razones de privacidad. Cuarto, los log del server sólo contienen

aquellos requerimientos que el mismo responde, excluyendo los casos en que los documentos son recuperados de un proxy server. Así el uso de caches, particularmente caches de proxy que están ampliamente extendidos en la Web, hacen declinar la exactitud de los log del servidor, con respecto a la medición del uso de las páginas Web.

Logs de Proxy:

Proveen información sobre los requerimientos que responde el proxy server, y así caracteriza un conjunto de requerimientos hechos por un grupo particular de clientes para servidores de cualquier parte del mundo.

Los logs de proxy también tienen limitaciones. Primero, muchos browsers de Web crean automáticamente caches de memoria o en disco. Así los requerimientos de usuarios para URL's que no son satisfechos desde la cache de memoria ó disco, no son enviados al proxy. Segundo, los browsers de Web sólo usan un proxy cuando son configurados explícitamente. Tercero, los usuarios pueden poner en *off* la opción de acceso al proxy.

Logs de clientes:

El log es coleccionado en la máquina del cliente, por su propio browser de Web ó por su proceso separado de monitoreo, corriendo en la máquina del cliente. Los logs de clientes son raros, pero existen varios proyectos que permiten recolectar información a nivel cliente. Ejemplos de logs de clientes están disponibles en *[17]*.

Los clientes de Web, normalmente no tienen la facilidad para generar logs. Así no hay un formato o seteo estándar de información listada en el mismo. Sin embargo, ésto es técnicamente realizable. Por ejemplo, el *NCSA Mosaic*, browser de Web, ha sido modificado por universitarios para coleccionar logs de clientes, tal cómo en la Universidad de Boston, en dónde se recupera: nombre de la máquina, hora, URL. tamaño, tiempo de respuesta, nombre del cliente windows, y desde dónde fue recuperado directamente.

Otro ejemplo potencial, es un *plug-in* de Netscape Navigator, que almacena en una base de datos, la URL, título, día y hora de acceso, día y hora de última modificación del documento y tamaño del documento.

La ventaja de los logs basados en el cliente, es que solucionan los problemas antes mencionados de los logs de servidores, ya que en log del cliente se pueden registrar todos los accesos que hace el cliente, sean a un proxy cache ó a un servidor de Web.

La limitación obvia, es que la información coleccionada, es sólo de aquellos clientes configurados para tal fin.

Logs de Red:

Existen herramientas, que permiten coleccionar logs de red del tráfico de la Web. Así se registran todos los requerimientos *http* que ocurren en la red al cuál el analizador está atachado.

Para el uso de éstas herramientas, se adjunta a la red un ***monitor de red*** que atiende pasivamente todo el tráfico que viaja a través de la red, identificando paquetes que contienen mensajes *http*, y construye un log de requerimientos de URL's.

Hay que tener en cuenta, que para el diseño de éstas herramientas surgen las siguientes preguntas:

- El formato de log generado por la herramienta, es compatible con los analizadores de Web existentes ?

- Pueden los logs ser mostrados en tiempo real cuando son recolectados ó deben ser recolectados primero y analizados off-line ?

- Para monitores de red, qué fracción de tráfico sobre la red es capturada en los logs ?

V. Formatos de Log Files:

Muchos servidores de Web, ofrecen la opción de almacenar logfiles en formato común, en formato flexible ó en un formato propietario. Estos formatos son configurables desde el Server Manager.

El formato común es soportado por todas las herramientas de análisis, pero la información que se registra es fija y por consiguiente puede presentar algunas ambigüedades, además en muchos casos surge la necesidad de registrar más información acerca de cada transacción. Así con el formato flexible se puede elegir que información registrar en el archivo de log.*[8][Sun]*

En un archivo de log se pueden configurar los siguientes items:

- ***Hostname del cliente***, si el DNS está deshabilitado es la dirección IP del cliente.
- ***Nombre del usuario***, si la autenticación es necesaria, se puede registrar el nombre del usuario.
- ***Fecha del sistema,*** día y hora del requerimiento.
- ***Requerimiento completo,*** el requerimiento exacto que hace el cliente.
- ***Estado,*** código de estado del servidor retornado al cliente.
- ***Longitud del contenido,*** longitud en bytes del documento enviado al cliente.
- ***HTTP header,"referer",*** el referer muestra la página previa que el cliente usó para llegar a la página corriente.
- ***HTTP header,"user-agent",*** incluye el tipo de browser que usa el cliente, version y tipo de sistema operativo.
- ***Método,*** el método del requerimiento usado.
- ***URI,*** Identificador universal de recursos, es parte de la URL. Ej: para http://www.a.com:8080/special/docs, el URI sería: /special/docs.
- ***String del query del URI,*** todo lo que está después del signo de interrogación en el URI. Ej: para http://www.a.com:8080/special/dosc?find_this, el string del query sería: find_this.

- ***Protocolo,*** protocolo de transporte y versión usado.
- ***Cache estado final,*** el método por el cual el documento en ubicado en el cache. Este puede ser escrito, actualizado ó refrescado, ó retornado por un *up-to-date* check.
- ***Código de estado del servidor,*** código de estado retornado por el servidor.
- ***Ruta al proxy,*** la ruta usada para retornar el recurso. El documento puede ser recuperado directamente, a través de un proxy ó a través de un servidor SOCKS.
- ***Tiempo de transferencia,*** la longitud de tiempo de la transferencia en segundos ó milisegundos.
- ***Longitud del header de la respuesta del servidor***
- ***Tamaño del header del requerimiento del proxy al server***
- ***Tamaño del header de la respuesta enviada al cliente***
- ***Tamaño del header del requerimiento recibido desde el cliente***
- ***Longitud del contenido del requerimiento del proxy al server,*** la longitud en bytes del documento que envía el proxy al server.
- ***Longitud del contenido recibido desde el cliente,*** la longitud en bytes del documento recibido desde el cliente.
- ***Longitud del contenido de la respuesta del servidor,*** la longitud en bytes del documento desde el servidor.
- ***Nombre del usuario dado por el servidor remoto durante la autenticación.***

Formato Común:

El formato común es el más básico de los formatos *[15].*

Sintaxis:

host - usr [time] "req"s1 c1

Campos:

host, es el DNS hostname del cliente. Si el DNS no está habilitado, va la dirección IP del cliente.

-, es el RFC 931. Este parámetro no está soportado a menos que esté corriendo el proxy cómo servidor SOCKS *[14].*

usr, es el nombre del usuario autenticado por el proxy.

time, es la fecha y hora del requerimiento.

req, es la primer línea del requerimiento HTTP que llega al proxy.

s1, es la respuesta del proxy al cliente.

c1, es la longitud del contenido enviada al cliente por el proxy.

El siguiente es un ejemplo de un típico archivo de log file en formato común:

ramses.unlp.edu.ar - - [11/Mar/1997:08:44:18 +0300] "GET ftp://ftp2.info.apple.com/Apple.Support.Area/Apple.Software.Updates/US/DOS-Windows/QuickTime/QTPLUG16.EXE HTTP/1.0" 500 514
ramses.unlp.edu.ar - - [11/Mar/1997:08:44:34 +0300] "GET ftp://ftp2.info.apple.com/Apple.Support.Area/Apple.Software.Updates/US/DOS-Windows/QuickTime/QTEASY16.EXE HTTP/1.0" 200 2436617
163.10.0.110 - - [11/Mar/1997:10:51:28 +0300] "GET ftp://ftp2.info.apple.com/Apple.Support.Area/Apple.Software.Updates/US/DOS-Windows/QuickTime/QTEASY16.EXE HTTP/1.0" 500 518
163.10.0.110 - - [11/Mar/1997:10:56:59 +0300] "GET ftp://www.quickTime.apple.com/sw/qtwin.html HTTP/1.0" 500 326
163.10.0.110 - - [11/Mar/1997:10:54:58 +0300] "GET ftp://ftp5.info.apple.com/Apple.Support.Area/Apple.Software.Updates/US/DOS-Windows/QuickTime/QTEASY16.EXE HTTP/1.0" 200 335872
163.10.0.110 - - [11/Mar/1997:11:01:54 +0300] "GET

ftp://ftp3.info.apple.com/Apple.Support.Area/Apple.Software.Updates/US/DOS-Windows/QuickTime/QTEASY16.EXE HTTP/1.0" 200 188416
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:00 +0300] "GET http://www.uvg.edu.gt/icons/blank.xbm HTTP/1.0" 404 245
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:00 +0300] "GET http://www.uvg.edu.gt/icons/back.xbm HTTP/1.0" 404 244
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:07 +0300] "GET http://www.uvg.edu.gt/icons/text.xbm HTTP/1.0" 404 244
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:00 +0300] "GET http://www.uvg.edu.gt/icons/image.xbm HTTP/1.0" 404 245
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:28 +0300] "GET http://www.uvg.edu.gt/icons/text.xbm HTTP/1.0" 404 244
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:28 +0300] "GET http://www.uvg.edu.gt/icons/back.xbm HTTP/1.0" 404 244
ramses.unlp.edu.ar - - [11/Mar/1997:18:12:50 +0300] "GET http://www.uvg.edu.gt/gr_lib/animado/ HTTP/1.0" 200 5017
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:28 +0300] "GET http://www.uvg.edu.gt/icons/blank.xbm HTTP/1.0" 404 245
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:28 +0300] "GET http://www.uvg.edu.gt/icons/image.xbm HTTP/1.0" 404 245
ramses.unlp.edu.ar - - [11/Mar/1997:18:13:47 +0300] "GET http://www.uvg.edu.gt:80/gr_lib/iconos/ HTTP/1.0" 200 2687
ramses.unlp.edu.ar - - [11/Mar/1997:18:14:04 +0300] "GET http://www.uvg.edu.gt:80/icons/back.xbm HTTP/1.0" 404 244
ramses.unlp.edu.ar - - [11/Mar/1997:18:14:20 +0300] "GET http://www.uvg.edu.gt:80/icons/text.xbm HTTP/1.0" 404 244
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:14:23 +0300] "GET http://home.netscape.com/escapes/search/images/tabbar_infoseek.gif HTTP/1.0" 200 1276
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:14:26 +0300] "GET http://home.netscape.com/escapes/search/images/search_infoseek12.gif HTTP/1.0" 200 1999
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:14:29 +0300] "GET http://home.netscape.com/escapes/search/images/search_infoseek13.gif HTTP/1.0" 200 173
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:14:11 +0300] "GET http://home.netscape.com/home/internet-search.html HTTP/1.0" 200 14994
ramses.unlp.edu.ar - - [11/Mar/1997:18:14:04 +0300] "GET http://www.uvg.edu.gt:80/icons/image.xbm HTTP/1.0" 404 147
ramses.unlp.edu.ar - - [11/Mar/1997:18:14:04 +0300] "GET http://www.uvg.edu.gt:80/icons/blank.xbm HTTP/1.0" 404 245
ramses.unlp.edu.ar - - [11/Mar/1997:18:15:11 +0300] "GET http://www.uvg.edu.gt:80/icons/blank.xbm HTTP/1.0" 404 147
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:14:08 +0300] "GET http://home.netscape.com/home/internet-search.html HTTP/1.0" 200 14994
ramses.unlp.edu.ar - - [11/Mar/1997:18:15:11 +0300] "GET http://www.uvg.edu.gt:80/icons/image.xbm HTTP/1.0" 404 147
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:14:48 +0300] "GET http://home.netscape.com/escapes/search/maps/search_tab.map?321,12 HTTP/1.0" 200 241
ramses.unlp.edu.ar - - [11/Mar/1997:18:15:11 +0300] "GET http://www.uvg.edu.gt:80/icons/back.xbm HTTP/1.0" 404 146
ramses.unlp.edu.ar - - [11/Mar/1997:18:15:11 +0300] "GET http://www.uvg.edu.gt:80/icons/text.xbm HTTP/1.0" 404 146
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:15:25 +0300] "GET http://home.netscape.com/escapes/search/images/search_yahoo.gif HTTP/1.0" 200 593
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:15:25 +0300] "GET http://home.netscape.com/escapes/search/images/tabbar_yahoo.gif HTTP/1.0" 200 1267
ramses.unlp.edu.ar - - [11/Mar/1997:18:14:49 +0300] "GET http://www.uvg.edu.gt:80/gr_lib/iconos/mail2.gif HTTP/1.0" 200 384
norma.fisica.unlp.edu.ar - - [11/Mar/1997:18:14:24 +0300] "GET http://home.netscape.com/escapes/search/images/search_infoseek11.gif HTTP/1.0" 200 1904
ramses.unlp.edu.ar - - [11/Mar/1997:18:11:08 +0300] "GET http://www.uvg.edu.gt/gr_lib/animado/mailmov.gif HTTP/1.0" 200 14893

Por ejemplo, si la línea es:

arrow.a.com - john[29/Mar/1996:4:36:53-0800] "GET /help HTTP/1.0" 401 571

Los campos serían:

arrow.a.com, en éste caso se muestra el hostname porque el DNS está habilitado

-, Información RFC 931, no está implementado

john, nombre del usuario

29/Mar/1996:4:36:53-0800, día y hora del requerimiento

GET /help, requerimiento

HTTP/1.0, protocolo

401, código de estado

571, bytes transferidos

Formato Extendido:

El formato extendido es más detallado que el formato común, porque éste incluye todos los campos del formato común, más algunos campos adicionales. *[7][Hallam Baker-Behelendorf].*

El formato extendido de log files permite capturar un amplio rango de datos y está diseñado para cumplir con las siguientes necesidades:

- Permitir control sobre los datos registrados
- Soporte de proxies, clientes y servidores en formato de log común
- Proveer manejo robusto de caracteres de escape
- Permitir intercambio de datos demográficos
- Permitir sumario de datos

El formato extensible, permite adaptar los log files para que se registren en formato leíble por herramientas de análisis genéricas.

Un formato extendido de log file, contiene una secuencia de líneas con caracteres ASCII, terminada por la secuencia LF ó CRLF. La terminación de la línea, depende de la plataforma dónde está siendo ejecutado.

Cada línea puede contener una directiva ó una entrada. Las entradas consisten en una secuencia de campos relacionados a una única transacción *http.*

Los campos están separados por blancos. Si un campo no es usado en una entrada en particular, se utiliza el signo "-", para marcar la omisión del campo.

Las directivas registran información sobre los propios procesos de logging.

Las líneas que comienzan con el carácter "#", contienen directivas.

Se definen las siguientes directivas:

Version: <integer>.<integer>

Indica la versión de log file usada.-

Fields: [<specifier>...]

Especifica los campos de datos que se registran en cada entrada en el archivo de log.

Software: string

Especifica el software que generó el log.

Start-Date: <date><time>

Indica el día y la hora en el cuál el log se creó.

End-Date: <date><time>

Indica el día y la hora en el cuál el log finalizó.

Date: <date><time>

Indica el día y la hora en el cuál la entrada fue agregada.

Remark: <text>

Para información de comentario. Los datos registrados en éste campo podrían ser ignorados por la herramienta de análisis.

Las directivas de *Version* y *Fields*, son necesarias y podrían preceder todas las entradas en el log.

El siguiente es un ejemplo del archivo de log, en cuál se indican las directivas en header del mismo, y a continuación todas las entradas al mismo:

#Version: 1.0
#Date: 12-Jan-1996 00:00:00
#Fields: time cs-method cs_uri
00:34:23 GET /foo/bar.html
12:21:16 GET /foo/bar.html
12:45:52 GET /foo/bar.html
12:57:34 GET /foo/bar.html

Fields:

La directiva de *#fields,* lista una secuencia de identificadores especificando la información registrada en cada campo.

El campo identificador puede tener una de las siguientes formas:

identificador:

Identificador relacionado a la transacción como un todo.

prefijo-identificador:

Identificador relacionado a la transferencia entre partes, definidas por el valor del prefijo.

prefijo (header):

Identifica el valor del campo header del http, para transferencias entre partes definidas en el prefijo. Los campos especificados de ésta manera siempre tienen el valor <string>.

Se definen los siguientes prefijos:

c

Cliente

s

Servidor

r

Remoto

cs

Cliente a Servidor

sc

Servidor a Cliente

sr

Servidor a Servidor Remoto, éste prefijo es usado por proxies

rs

Servidor Remoto a Servidor, éste prefijo es usado por proxies

x

Aplicación específica del identificador

Así, *cs-method*, se refiere al método en el requerimiento enviado por el cliente al servidor, mientras que *sc(Referer)*, se refiere al referer: campo de la copia. El identificador *c-ip*, se refiere a la dirección IP del cliente.-

Identificadores:

Los siguientes identificadores no requieren un prefijo:

date

Día en el cuál la transacción se realizó, el campo tiene tipo <date>

time

Hora en la cuál la transacción se realizó, el campo tiene tipo <time>

time-taken

Tiempo en segundos que ocupó la transacción, el campo es de tipo <fixed>

bytes

Bytes transferidos, es de tipo <integer>

cached

Registra cuando un *'hit'* de cache ocurre, el campo es de tipo <integer>, con <0> se indica un *'miss'*

Los siguientes identificadores requieren prefijo:

ip

Dirección IP y puerto, de tipo <address>

dns

DNS, de tipo <name>

status

Código de estado, el campo es de tipo <integer>

comment

Comentario retornado con el código de estado, de tipo <text>

method

Método, campo de tipo <name>

uri

URI, campo de tipo <uri>

uri-steam

Porción del tronco única del URI (omitiendo el query), de tipo <uri>

uri-query

Porción del query única del URI, de tipo <uri>

Campos especiales para sumarios de logs:

Las herramientas de análisis pueden generar sumarios de log. Una entrada de un sumario comienza con una cantidad especificando el número de veces que un evento en particular ocurre. Por ejemplo, un sitio puede estar interesado en contar el número de requerimientos de una URI en particular, con un campo *referer:* dado, pero puede no estar interesado en registrar información acerca de los requerimientos particulares tales cómo la dirección IP.

El siguiente campo es mandante y debe preceder a los otros:

count

Indica el número de entradas, es de tipo <integer>

Los siguientes campos pueden ser usados en lugar del tiempo, para permitir entradas sobre intervalos de tiempo:

time_from

Tiempo en el cuál comienza la muestra, es de tipo <time>

time-to

Tiempo en el cuál la muestra finaliza, es de tipo <time>

interval

Tiempo sobre el cuál ocurre la muestra en segundos, de tipo <integer>

Entradas:

El formato de datos para los campos de entrada se eligen para evitar ambigüedades, minimizar las dificultades de generación y análisis, y proveer una legibilidad humana.-

Cada entrada en el logfile consiste de una secuencia de campos separados por blancos y terminada con una secuencia CR ó CRLF.

El significado de los campos se define en el directiva *#fields*. Si un campo es omitido para una entrada en particular, es sustituido por el signo "-".

Los analizadores de logfiles, podrían ser tolerantes a errores. si una entrada contiene datos corruptos ó es terminada con un símbolo no esperado, el analizador podría re-sincronizarse usando la marca de fin de línea y continuar analizando la siguiente línea.

Las entradas no deben contener caracteres de control ASCII.

<entry> = *<field> <end-of-line>

<field> = <integer> | <fixed> | <uri> | <date> | <time> | <string>

<digit> = |0| |1| |2| |3| |4| |5| |6| |7| |8| |9|

Integer:

<integer> = 1* <digit>

Los enteros se representas como una secuencia de dígitos.

Fixed Format Float:

<fixed> = 1* <digit> [.<digit>]

URI:

Una URI es especificada por RFC1738, URI's relativas son especificadas por RFC1808. Las URI's no pueden por definición, contener espacios en blanco o caracteres de control ASCII.

Date:

<date> = 4<digit> "-" 2<digit> "-" 2<digit>

Los campos fecha se registran en formato YYYY-MM-DD.

Time:

<time> = 2<digit> ":" 2<digit> [":" 2<digit> ["." *<digit>]

La hora se registra de la forma HH:MM, HH:MM:SS ó HH:MM:SS.S

String:

<string> = ´ | ´ *<schar> ´ | ´

<schar> = xchar | ´ | ´ ´ | ´

Los strings están entre comillas, si un string contiene un carácter entre comillas, dicho carácter es repetido. el signo '-', no debe ser usado para strings vacíos.

Text:

<text> = <char>*

El campo texto es usado sólo por directivas.

Name:

<name> = <alpha> [| .| *<alpha>]

Representa el DNS.

Address:

<name> = <integer> [| .| *<integer>] [| :| <integer>]

Dirección numérica IP, y como opcional al definición del port.

Un ejemplo de formato extendido podría contener básicamente los siguientes campos:

host - usr [time] "req" s1 c1 s2 c2 b1 b2 h1 h2 h3 h4 xt

Los primeros 7 campos son los del formato común, y los campos adicionales corresponden a:

s2, es la respuesta HTTP del servidor remoto para el proxy, cuándo el proxy hace un requerimiento de parte del cliente. (código de estado)

c2, es la longitud del contenido que recibe el proxy.

b1, es el tamaño del cuerpo del mensaje del requerimiento HTTP del cliente. (en otras palabras es el POST-data, que puede ser forwardeado al servidor remoto)

b2, es el tamaño del cuerpo del mensaje del requerimiento HTTP del proxy. Es la cantidad de datos en el cuerpo del mensaje que pueden ser enviados al servidor remoto. Si no ocurren errores es igual al ***b1.***

h1, es el tamaño del header del requerimiento HTTP del cliente al proxy.

h2, es el tamaño del header de la respuesta del proxy al cliente

h3, es el tamaño del header del requerimiento del proxy al servidor remoto.

h4, es el tamaño del header de la respuesta del servidor al proxy.

xt, es el total del tiempo de la transferencia en segundos.

El siguiente es un ejemplo de access log usando formato flexible:

format=%Ses->client.ip% - %Req->vars.pauth-user% [%SYSDATE%] "%Req->reqpb.proxy-request%" %Req->srvhdrs.clf-status% %Req->vars.p2c-cl% %Req->vars.remote-status% %Req->vars.r2p-cl% %Req->headers.content-length% %Req->vars.p2r-cl% %Req->vars.c2p-hl% %Req->vars.p2c-hl% %Req->vars.p2r-hl% %Req->vars.r2p-hl% %Req->vars.xfer-time%
157.92.77.32 - - [20/Mar/1998:15:54:53 +0300] "GET http://www.hotmail.com/ HTTP/1.0" 304 - - - - - 284 115 - - 0

157.92.77.32 - - [20/Mar/1998:15:54:55 +0300] "GET http://209.1.112.251/bkgdhome1.gif HTTP/1.0" 304 - - - - - 311 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:54:55 +0300] "GET http://209.1.112.251/spacer.gif HTTP/1.0" 304 - - - - - 309 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:54:55 +0300] "GET http://209.1.112.251/hotmail_logo.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:54:55 +0300] "GET http://209.1.112.251/logo_fromMS.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:54:57 +0300] "GET http://209.1.112.251/slogan_bl.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:54:57 +0300] "GET http://209.1.112.251/home_lbllogin.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:54:57 +0300] "GET http://209.1.112.251/home_lblpasswd.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:54:57 +0300] "GET http://209.1.112.251/home_lblnoframes.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:54:59 +0300] "GET http://209.1.112.251/c9698.gif HTTP/1.0" 304 - - - - - 311 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:55:00 +0300] "GET http://209.1.112.251/home_lblframes.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:55:00 +0300] "GET http://209.1.112.251/home_lblreg2.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:55:00 +0300] "GET http://209.1.112.251/home_do_map.gif HTTP/1.0" 304 - - - - - 313 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:55:03 +0300] "GET http://209.1.112.251/home_lbldefault.gif HTTP/1.0" 304 - 304 - - - 368 42 403 42 4
170.210.120.120 - - [20/Mar/1998:15:55:16 +0300] "GET http://www2.net117.or.jp/~mb1996ax/hj07.html HTTP/1.0" 200 500 304 - - - 360 509 449 120 11
170.210.122.51 - - [20/Mar/1998:15:55:54 +0300] "GET http://www.microsoft.com/redirect.asp?Target=%2Fmsdownload%2Fieplatform%2Fie4rtw%2F02.htm&PageID=23&ItemID=48&ID=4&Param=&Referer=http%253A%252F%252Fwww%252Emicrosoft%252Ecom%252Fie%252Fie40%252Fdownload%252Fie4%252Ehtm HTTP/1.0" 302 343 302 343 - - 684 416 719 416 8
170.210.120.120 - - [20/Mar/1998:15:55:56 +0300] "GET http://www2.net117.or.jp/~mb1996ax/ho10.html HTTP/1.0" 200 375 304 - - - 360 509 449 120 9
170.210.122.51 - - [20/Mar/1998:15:56:01 +0300] "GET http://www.microsoft.com/redirect.asp?Target=%2Fmsdownload%2Fieplatform%2Fie4rtw%2F02.htm&PageID=23&ItemID=48&ID=4&Param=&Referer=http%253A%252F%252Fwww%252Emicrosoft%252Ecom%252Fie%252Fie40%252Fdownload%252Fie4%252Ehtm HTTP/1.0" 504 279 - - - - 684 108 - - 25
170.210.122.51 - - [20/Mar/1998:15:56:07 +0300] "GET http://register.microsoft.com/regwiz/redirect.asp?target=http://www.microsoft.com/msdownload/ieplatform/ie4rtw/02.htm&pageid=23&itemid=48&param=&referer=http%3A%2F%2Fwww%2Emicrosoft%2Ecom%2Fie%2Fie40%2Fdownload%2Fie4%2Ehtm HTTP/1.0" 302 384 302 384 - - 692 585 727 585 12
170.210.122.51 - - [20/Mar/1998:15:56:18 +0300] "GET http://www.kittyskorner.com/DsgnSt18/Applpupy.EXE HTTP/1.0" - 2647 200 - - - 476 273 511 - 7
170.210.122.51 - - [20/Mar/1998:15:56:23 +0300] "GET http://www.kittyskorner.com/DsgnSt18/Applpony.EXE HTTP/1.0" - 1187 200 - - - 476 273 511 - 24
170.210.122.51 - - [20/Mar/1998:15:56:30 +0300] "GET http://msid.msn.com/mps_id_sharing/redirect.asp?register.microsoft.com/regwiz/redirect.asp?target=http://www.microsoft.com/msdownload/ieplatform/ie4rtw/02.htm&pageid=23&itemid=48&param=&referer=http%3A%2F%2Fwww%2Emicrosoft%2Ecom%2Fie%2Fie40%2Fdownload%2Fie4%2Ehtm HTTP/1.0" 302 385 302 385 - - 723 581 758 581 22
170.210.122.51 - - [20/Mar/1998:15:56:33 +0300] "GET http://register.microsoft.com/regwiz/redirect.asp?target=http://www.microsoft.com/msdownload/ieplatform/ie4rtw/02.htm&pageid=23&itemid=48&param=&referer=http%3A%2F%2Fwww%2Emicrosoft%2Ecom%2Fie%2Fie40%2Fdownload%2Fie4%2Ehtm&newguid=c61c381ac02211d18b2208002bb74f3f HTTP/1.0" 302 343 302 343 - - 782 546 817 546 8
157.92.77.32 - - [20/Mar/1998:15:56:37 +0300] "GET http://209.1.112.251/bullet_logo.gif

HTTP/1.0" 304 - - - - - 396 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:56:44 +0300] "GET http://209.1.112.251/hotmail_logo_inset.gif HTTP/1.0" 304 - - - - - 386 115 - - 0
157.92.77.32 - - [20/Mar/1998:15:56:44 +0300] "GET http://209.1.112.251/bkgd_adframe1.gif HTTP/1.0" 304 - - - - - 408 115 - - 0

Otros Formatos:

Algunos servidores de Web permitir definir sus propios formatos de logs, llamados formatos propietarios.

Un ejemplo de éste tipo de formatos podría ser incluyendo todos los campos del formato extendido, más otros campos adicionales *[8][Sun]*.

Sintaxis:

host - usr [time] "req" s1 c1 s2 c2 b1 b2 h1 h2 h3 h4 xt route cs ss cs

Campos:

route, es la ruta usada para recuperar el recurso. Puede ser:

- DIRECTA, el recurso es recuperado directamente.
- PROXY (host:port), significa que el recurso fue recuperado a través de un proxy server en el host y port especificado.
- SOCKS (host:port), significa que el recurso fue recuperado a través de socks en el host y port especificado.

cs, es el estado final del cliente. Este campo especifica si el requerimiento del cliente se completó satisfactoriamente, si se presionó el botón de stop ó si se abortó por error. Este campo puede ser:

- -, significa que el requerimiento nunca comenzó.
- FIN, significa que el requerimiento se completó satisfactoriamente.
- INTR, significa que el requerimiento fue interrumpido o por el cliente, o por el proxy o por time out.

ss, es el estado final del servidor remoto. Especifica si se completó satisfactoriamente, interrumpido por el cliente ó abortado por error. Puede ser lo siguiente:

- -, significa que nunca comenzó
- FIN, significa que se completó satisfactoriamente.
- INTR, significa que el requerimiento fue interrumpido por el cliente ó terminado por el proxy.
- TIMEOUT, timeout por el proxy.

cs, es el caché estado final. Este campo especifica cuando un archivo fue escrito en el caché, refrescado ó retornado por un up-to-date check. Puede ser lo siguiente:

- -, significa que el recurso fue no cacheable
- WRITTEN, significa que el archivo fue escrito
- REFRESHED, significa que el archivo fue refrescado ó actualizado
- NO-CHECK, significa que el archivo fue retornado sin un up-to-date check
- UP-TO-DATE, significa que el archivo fue retornado con un up-to-date check

VI. Analizadores de logs de Web:

Los analizadores de logs de Web, son herramientas de análisis y reporte, que se utilizan para marketing y para búsqueda de técnicas de mercado en los sitios Web.

Con los datos que se encuentran en el log de accesos de un servidor, se puede saber quiénes acceden al mismo y a qué información acceden. Haciendo así un estudio detallado del sitio que ayuda a la planificación y a la auditoría de seguridad.

Estos analizadores, toman la información de los archivos de logs de los servidores y permiten responder a preguntas tales cómo:

- **Qué hace la gente cuando visita un sitio ?**
- **Existen padrones de la manera en que la gente usa el sitio ?**
- **Se pueden agrupar los visitadores ?**
- **Cuánta gente habla en él ?**

Así es posible recolectar información a través de ellos, y generar reportes de los datos de la manera que se necesite, en formatos preestablecidos ó con la creación de formatos propios.

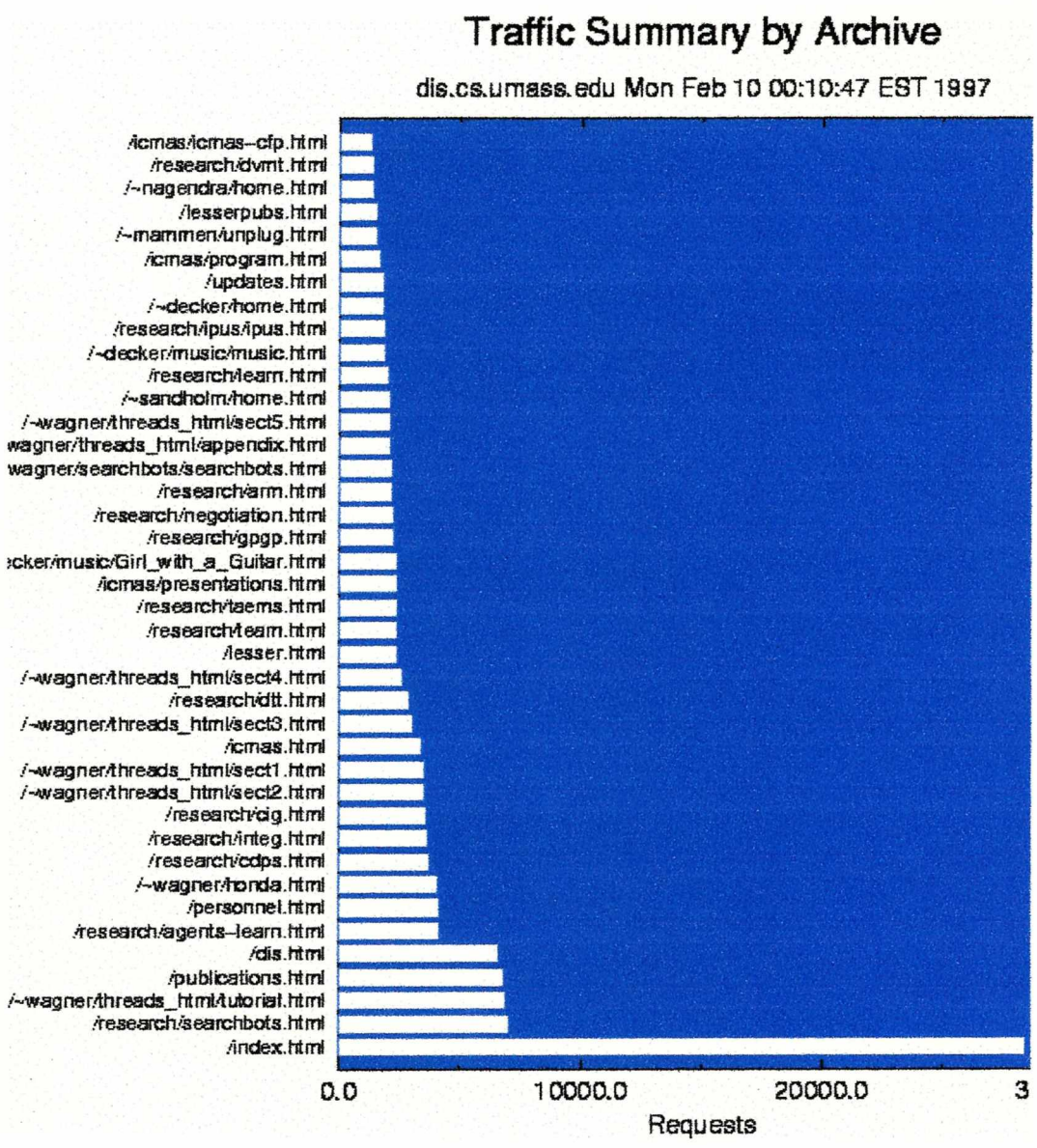
Existen actualmente analizadores poderosos que aplican técnicas de búsqueda de mercado para identificar padrones y trends en el uso del sitio Web, permiten crear tablas y gráficos propios, construir reportes desde cualquiera de ellos y actualizarlos cada vez que se agregue información a la base de datos.

Se pueden mencionar algunos analizadores de logs, tales cómo: Webalyser *[10]*, Gwstat *[13]*, Wwwstat *[12]*, Wusage *[11]*, *Analog 2.11[19]*,etc.

Ejemplos de tablas y gráficos que se pueden obtener a través de ellos son los siguientes:

WWWSTAT:

Gráfico de tráfico por archivo:



Gáfico de Tráfico por día:

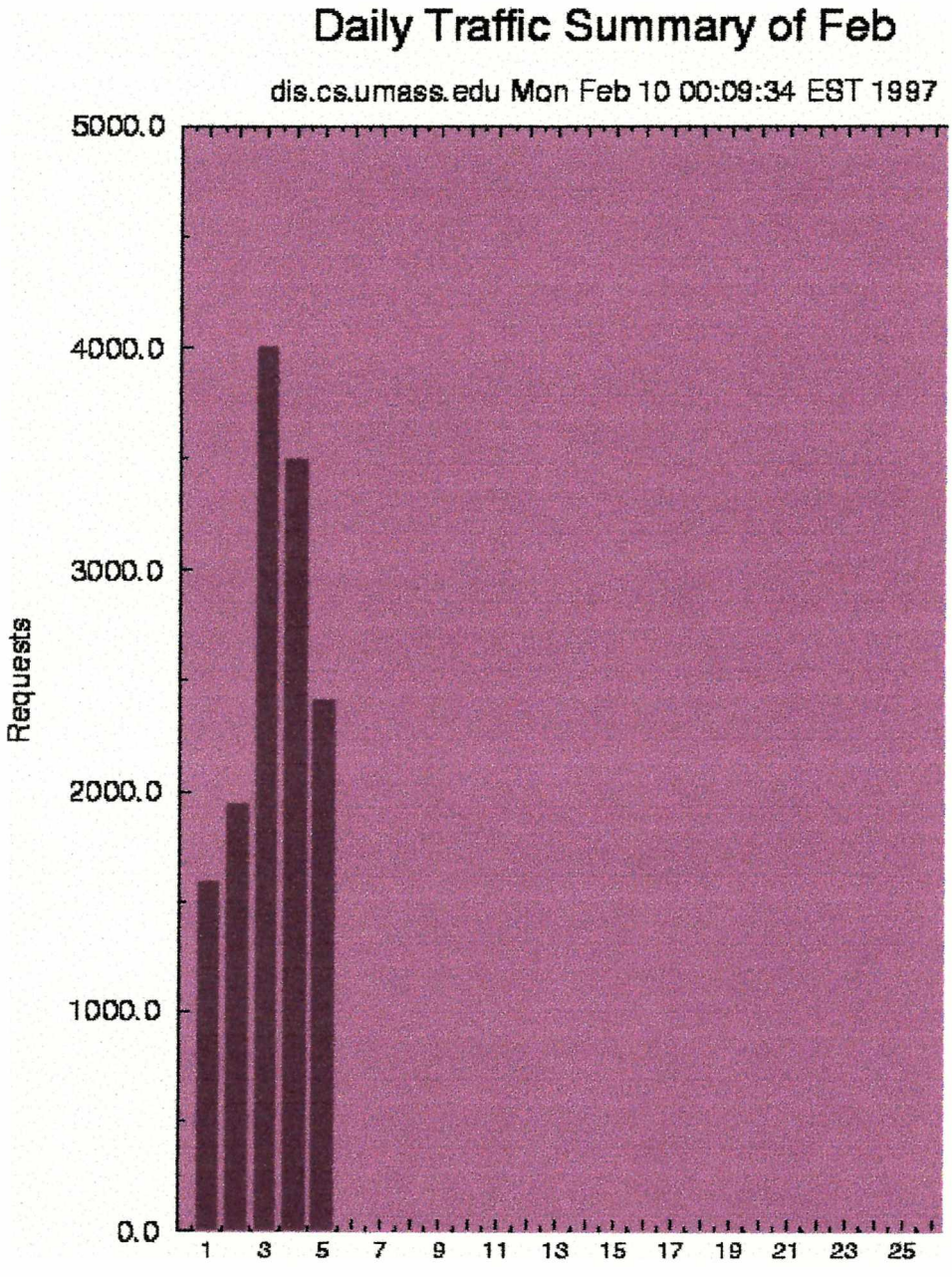
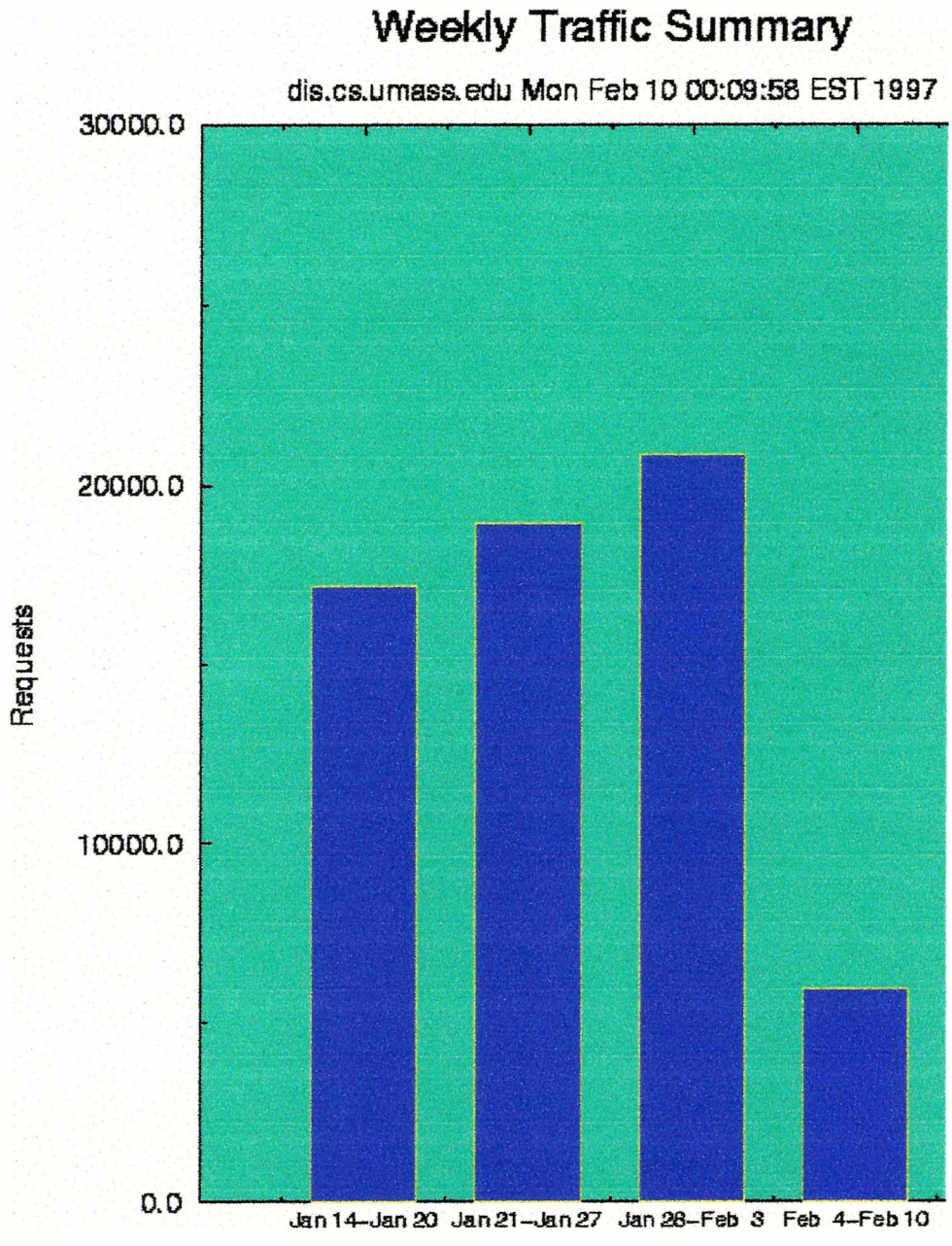


Gráfico de Tráfico por semana:



WEBALYSER:

Gráfico de línea que muestra las visitas por hora de cada día:

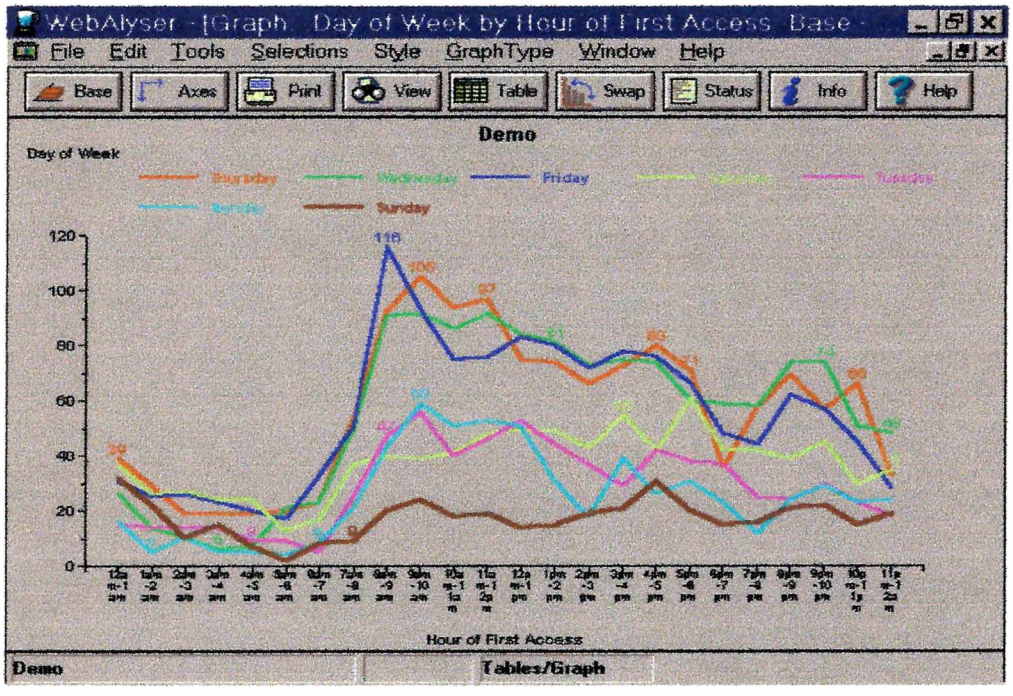


Gráfico de espacio que muestra las relaciones entre áreas del sitio:

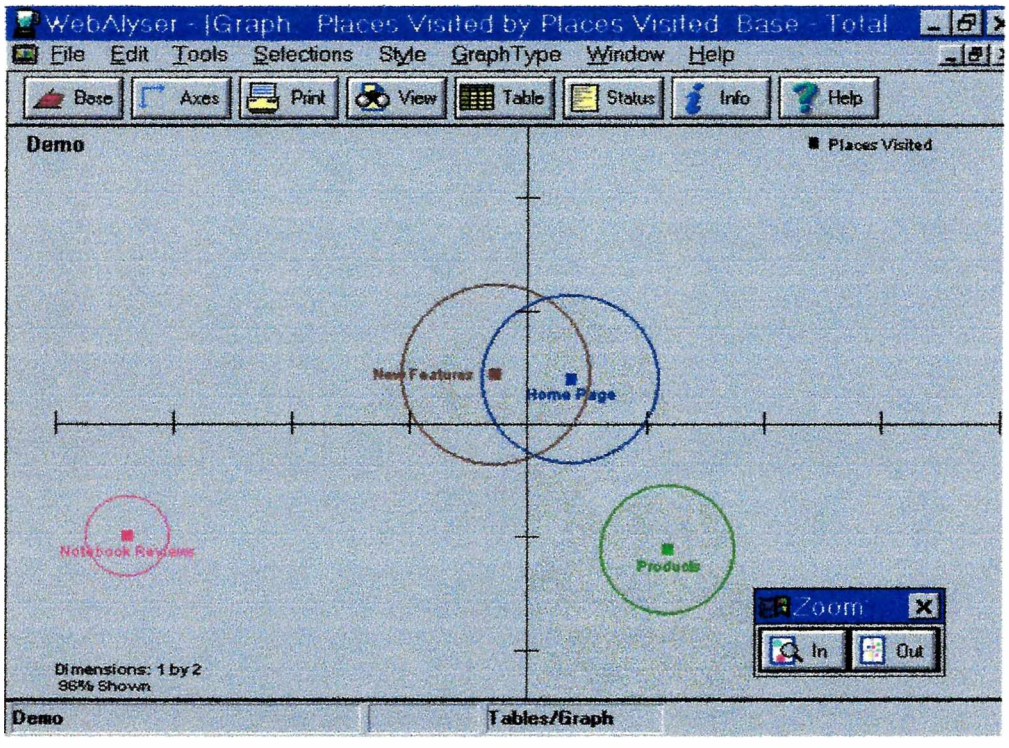
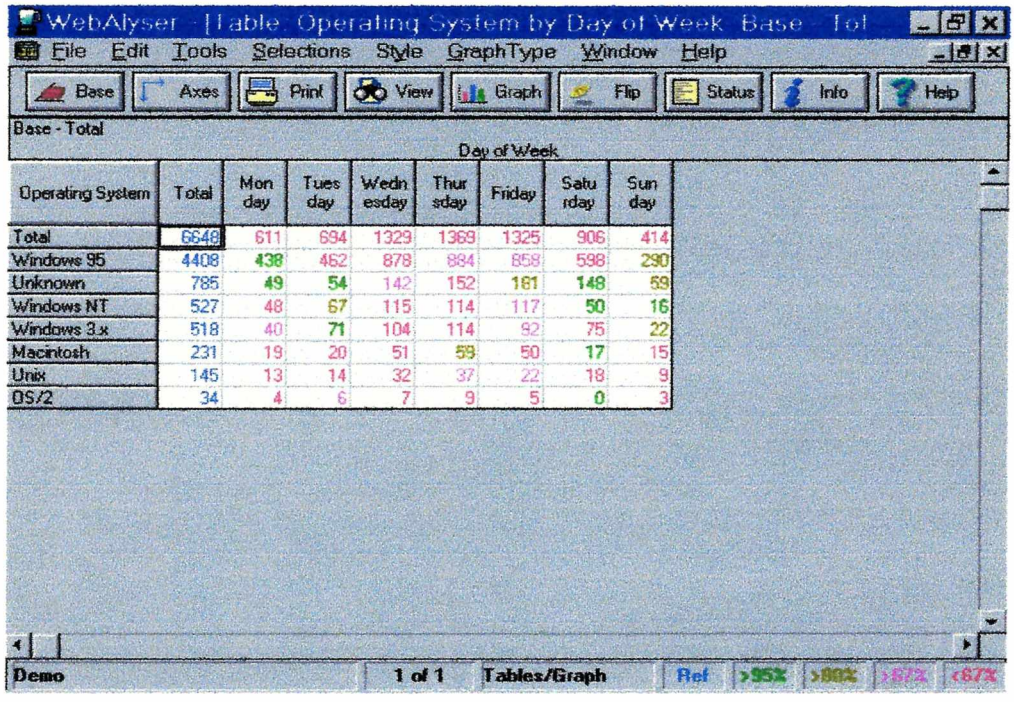


Tabla de cantidad de visitas según sistema operativo:



WebAlyser - [Table: Operating System by Day of Week Base - Tot

File Edit Tools Selections Style GraphType Window Help

Base Axes Print View Graph Flip Status Info Help

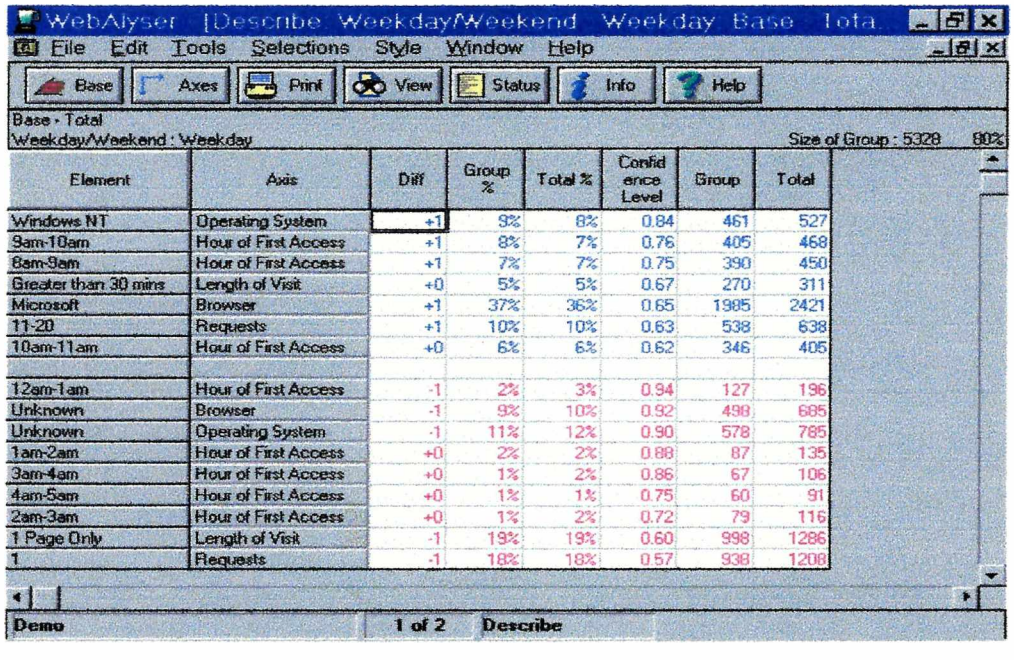
Base - Total

Day of Week

Operating System	Total	Mon day	Tues day	Wedn esday	Thur sday	Friday	Satu rday	Sun day
Total	6648	611	694	1329	1369	1325	906	414
Windows 95	4408	438	462	878	884	868	598	280
Unknown	785	49	54	142	152	161	148	59
Windows NT	527	48	67	115	114	117	50	16
Windows 3.x	518	40	71	104	114	92	75	22
Macintosh	231	19	20	51	59	50	17	15
Unix	145	13	14	32	37	22	18	9
OS/2	34	4	6	7	9	5	0	3

Demo 1 of 1 Tables/Graph Ref >95% >80% >67% <67%

Tabla que muestra el tipo de visitadores asociados con áreas del sitio:



WebAlyser - [Describe Weekday/Weekend Weekday Base - Tota

File Edit Tools Selections Style Window Help

Base Axes Print View Status Info Help

Base - Total

Weekday/Weekend : Weekday

Size of Group : 5328 80%

Element	Axis	Diff	Group %	Total %	Confid ence Level	Group	Total
Windows NT	Operating System	+1	9%	8%	0.84	461	527
9am-10am	Hour of First Access	+1	8%	7%	0.76	405	468
8am-9am	Hour of First Access	+1	7%	7%	0.75	390	450
Greater than 30 mins	Length of Visit	+0	5%	5%	0.67	270	311
Microsoft	Browser	+1	37%	36%	0.65	1985	2421
11-20	Requests	+1	10%	10%	0.63	538	638
10am-11am	Hour of First Access	+0	6%	6%	0.62	346	405
12am-1am	Hour of First Access	-1	2%	3%	0.94	127	196
Unknown	Browser	-1	9%	10%	0.92	498	685
Unknown	Operating System	-1	11%	12%	0.90	578	785
1am-2am	Hour of First Access	+0	2%	2%	0.88	87	135
3am-4am	Hour of First Access	+0	1%	2%	0.86	67	106
4am-5am	Hour of First Access	+0	1%	1%	0.75	60	91
2am-3am	Hour of First Access	+0	1%	2%	0.72	79	116
1 Page Only	Length of Visit	-1	19%	19%	0.60	998	1286
1	Requests	-1	18%	18%	0.57	938	1208

Demo 1 of 2 Describe

Torta 3D de datos por sistema operativo:

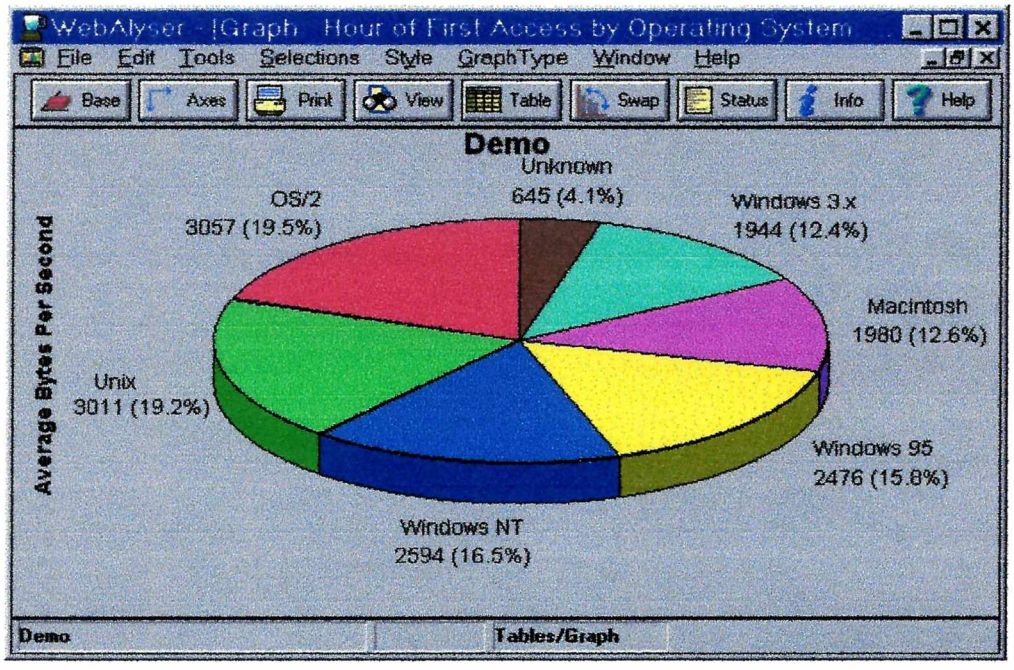
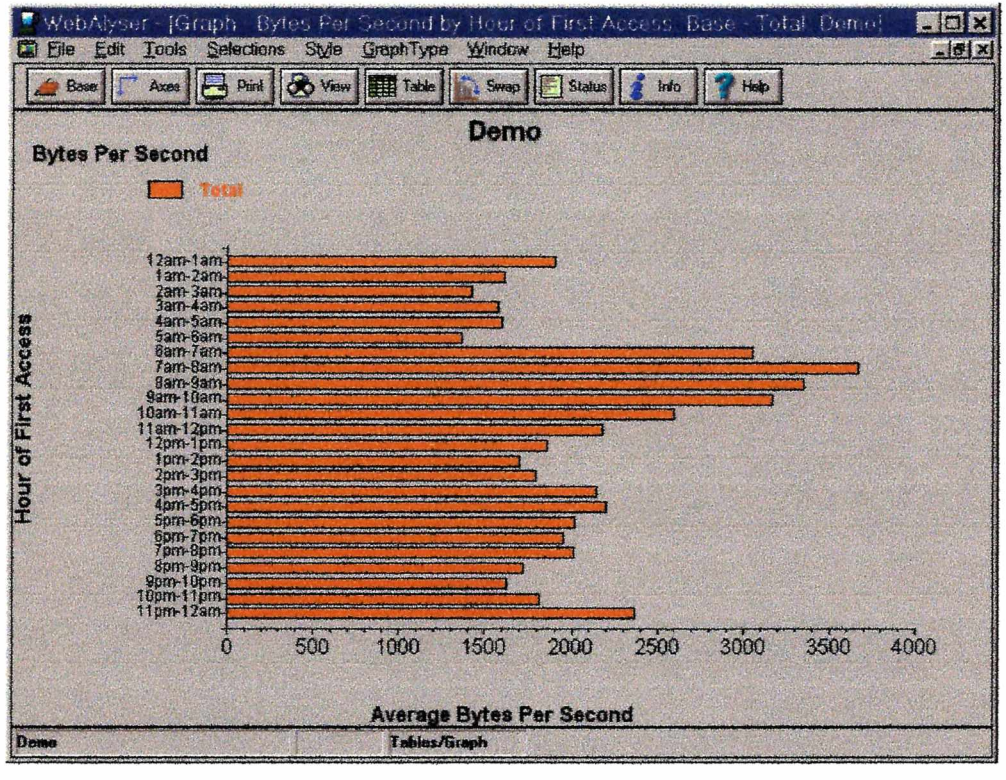


Gráfico de Barras que muestra el promedio de bytes que se mueven en cada visita por tiempo de visita:



ANALOG 2.11:

Es un programa freeware, que trabaja en máquinas UNIX, DOS, MAC y VMS.

Permite realizar reportes tales cómo: Informes Mensuales, Diarios, Resumen de Horarios, Información por tipo de fichero, Información de directorios, de peticiones, de hosts, de dominios y de códigos de estado.

Por ejemplo, tomando un período de análisis para un servidor de Web se pueden obtener las siguientes estadísticas:

Estadísticas calculadas el Lun 21 Sep 1998 17:00 hora local.
Análisis del periodo desde el Jue 01 Ene 1998 00:12 al Lun 21 Sep 1998 16:56 (263.7
días).

Número de peticiones atendidas: 576 084
Media diaria de peticiones atendidas: 2 185
Número de peticiones de páginas atendidas: 154 224
Media diaria de peticiones de páginas atendidas: 585
Número de peticiones fallidas: 26 861
Número de peticiones redirigidas: 627 158

Número de ficheros diferentes solicitados: 3 377
Número de hosts diferentes atendidos: 29 123
Líneas inválidas en el fichero de log: 2 703
Tráfico total: 1 829 Mbytes
Tráfico medio por día: 7 104 kbytes

Informe sobre tipos de Fichero:

N.pet: % pet: %bytes: extensión

1	358730:	62.27%:	36.88%:	.gif
2	91041:	15.80%:	28.42%:	.html
3	60480:	10.50%:	7.57%:	(directories)
4	38444:	6.67%:	0.72%:	(no extension)
5	12314:	2.14%:	0.24%:	.xbm
6	8462:	1.47%:	7.12%:	.jpg
7	2703:	0.47%:	0.95%:	.htm
8	1007:	0.17%:	0.81%:	.GIF
9	787:	0.14%:	0.03%:	.txt
10	678:	0.12%:	0.47%:	.cgi
11	396:	0.07%:	0.02%:	.ps

Informe de Directorios:

N.pet: % pet: %bytes: directorio

1	152035:	26.39%:	18.18%:	/figures/icons/
2	77379:	13.43%:	11.68%:	[root directory]
3	67559:	11.73%:	3.49%:	/figures/logos/
4	39688:	6.89%:	0.53%:	/figures/patterns/
5	35620:	6.18%:	0.47%:	/cgi-bin/
6	29393:	5.10%:	0.14%:	/figures/balls/
7	17779:	3.09%:	0.71%:	/pic/
8	14389:	2.50%:	0.11%:	/figures/arrows/
9	12925:	2.24%:	2.98%:	/ditupmCourses/
10	11971:	2.08%:	3.12%:	/snh/
11	9289:	1.61%:	4.15%:	/figures/photos/madrid/

Informe de Peticiones:

N.pet: % pet: %bytes: fichero

```
----- ------ ------ -------
  1 51453:  8.93%:  5.89%: /
  2  5140:  0.89%:  0.57%: /ditupmCourses/indice.html
  3  4999:  0.87%:  1.32%: /ditupmCourses/grado.html
  4  4115:  0.71%:  1.29%: /ditupmPeople.html
  5  2958:  0.51%:  2.11%: /links/spain.html
  6  2302:  0.40%:  0.25%: /cursos.html
  7  2242:  0.39%:  0.12%: /links.html
  8  1925:  0.33%:  3.37%: /infowin/atmeurope/atmrefs.html
  9  1777:  0.31%:  0.07%: /figures/maps/m40_map.html
 10  1742:  0.30%:  0.53%: /whatsnew.html
 11  1549:  0.27%:  0.62%: /travelinfo.html
```

Informe de Códigos de Estado:

N.ocu: nr. descripción

```
------ ---------------
413948: 200 OK
  1729:    [Miscellaneous successes]
627158: 302 Document found elsewhere
160407: 304 Not modified since last retrieval
   31: 400 Bad request
   72: 401 Authorisation required
   50: 403 Forbidden
 26388: 404 Document not found
  152:    [Miscellaneous client/user errors]
  123: 500 Internal server error
   45: 501 Request type not supported
```

VII. Actualización de los archivos log del servidor:

La información que se registra en los archivos de logs, da un panorama casi perfecto del comportamiento de los usuarios en un sitio Web.

Esta información se utiliza para diversos fines, cómo monitoreo, control de errores, seguridad, marketing y estadísticas, para lo cuál existen actualmente varios analizadores de logs de Web que permiten realizar tablas y gráficos referidos al sitio.

Cuando estamos ante un modelo de proxies, se plantea un problema: *los logs de servidores no reflejan la realidad de accesos*, esto es porque cuando el proxy server es el que responde a los requerimientos de los clientes retornando documentos desde su cache local, dichos movimientos quedan registrados en el log del proxy y no en el log del servidor, por lo que dicho archivo de accesos queda desactualizado.

Así la información recolectada por los analizadores de estadísticas, no es tan exacta y puede en algunos casos distorsionar los resultados esperados.

Para solucionar éste problema, la idea es mantener el archivo de log del servidor actualizado, para lo cuál surge la necesidad de diseñar un modelo que permita intercambiar información entre logs de proxies y logs de servidores, es decir *sincronizar* los logs.

Si le diéramos un enfoque impuro a la solución, se podría utilizar el header *no-cache* por ejemplo, aplicado a una página de interés, así se podrían contabilizar los accesos a la misma. Pero hay que tener en cuenta que ésto es aplicable en forma específica y no a todo el sitio, por lo que igualmente la información estaría limitada.

Así, para sincronizar logs, se plantean aquí dos modelos:

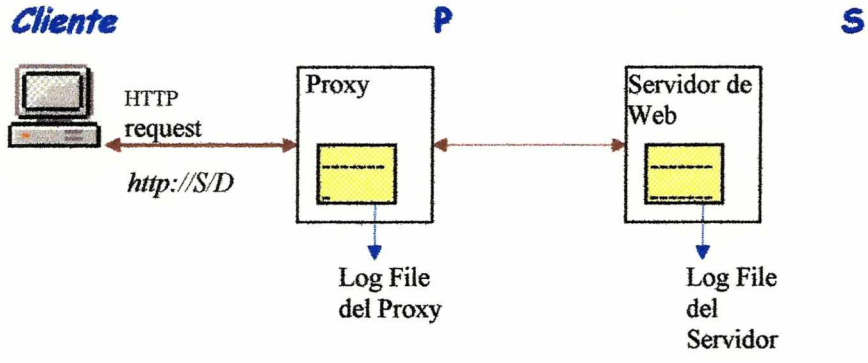
Modelo1: propone trabajar con el archivo de log del proxy, leyendo las líneas de requerimientos que se registran en el mismo y filtrando la información de accesos resuelta por el mismo proxy.

Modelo 2: adiciona el uso de una herramienta que se utiliza para el análisis de tráfico de una red, el *tcpdump [18]*, y que sirve a su vez para analizar los requerimientos que arrivan al proxy y su estado final.

El objetivo del trabajo es analizar ambos modelos y poder descubrir los pro y los contras de cada uno, con lo cuál se llegará a adoptar uno de los dos ó por el contrario, definir dos métodos alternativos de solución.

Modelo 1:

Para comprender mejor la situación, se plantea un esquema sencillo:



Supongamos un cliente configurado para usar un proxy 'P', que requiere un documento 'D'del servidor de Web 'S'.

El cliente envía un mensaje HTTP GET: ***http://S/D*** a P. Se pueden presentar tres casos: (1) P tiene una copia de D, (2) P tiene una copia de D inconsistente y (3) P no tiene una copia de D.

En el caso (1), P sirve la copia local, esto sería un ***hit.***

En (2), P envía un Get condicional a S, con el tiempo de la copia, (Last-Modified), si el original fue modificado después de ese tiempo, S replica a P con una nueva copia, esto sería un ***miss***, sino P sirve de la copia local, esto sería un ***hit***.

En (3), P envía un mensaje a otro proxy server ó al servidor S, esto sería un ***miss***.

En los casos en que se produce un ***hit de cache***, el requerimiento del cliente queda registrado en el log de accesos del proxy, por lo tanto la idea es recuperar dichos requerimientos, para actualizar el log del servidor real.

El modelo que se plantea aquí, consiste de dos pasos: primero la recuperación de la información que se registra sólo en el log del proxy, y segundo la actualización de los logs de los servidores involucrados.

Así se definen dos procesos los cuales se ejecutan en el proxy.

Estos procesos, son controlados y sincronizados por el comando de tareas programadas, por ejemplo en Unix, el *deamon cron*, que según especificaciones permite disparar los procesos automáticamente. La idea es que cada 'x' cantidad de tiempo se ejecute el proceso de lectura del log del proxy, y cada 'y' cantidad de tiempo se ejecute el proceso de actualización de los logs de servidores.

Para el proceso de recuperación de los requerimientos que no llegan al servidor real, se propone trabajar con el archivo de log del proxy, leyendo las líneas de requerimientos del mismo.

Cada requerimiento leído es filtrado considerando sólo los que tienen el header HTTP y un hit de cache.

Con los requerimientos seleccionados se genera un archivo temporario, para luego actualizar el log de accesos del servidor correspondiente. El formato de dicho archivo, será el mismo que el del log del proxy y del servidor de Web, ya que dicho formato debe ser preestablecido.

Cómo un proxy atiende requerimientos de varios usuarios y esos requerimientos pueden ser de distintos servidores, el modelo debe ser lo suficientemente flexible como para controlar los requerimientos de cada uno de los servidores a los cuales se solicita información.

Es así que a medida que se procesa la información registrada en el archivo de log, se va generando un archivo de actualización para cada servidor que el mismo involucra.

Cada uno de esos archivos de actualización se crean en un directorio en

particular para facilitar el mecanismo posterior de actualización, en éste caso el subdirectorio es llamado ***ac***, y los archivos de actualización llevan como nombre el DNS ó dirección IP que se especifica en el requerimiento.

Por ejemplo, si la línea en el archivo de log es :

ramses.unlp.edu.ar - - [11/Mar/1997:18:13:00 +300] "GET http://www.uvg.edu.gt/icons/blank.xbm HTTP/1.0"404 245

el archivo que se genera se llamará: **www.uvg.edu.gt,** el cuál se crea si no existe ó se adiciona la línea si existe.

Por lo tanto se generan tantos archivos de actualización como servidores especificados en los requerimientos leídos en el archivo de log del proxy.

El segundo paso del modelo, es la actualización de logs de servidores, para lo cuál el proceso lee los archivos generados en el subdirectorio ***ac***, y accede a cada servidor, según el nombre del archivo, adicionando al log del servidor real el archivo temporario que se formó con los requerimientos leídos del log del proxy.

El primer proceso está implementado en lenguaje C, y el proceso de actualización en Shell.

Resumen de Procesos que se ejecutan en el Servidor Proxy:

- Se utiliza un archivo de "control" *(control.txt)*, en el cuál se va registrando fecha y línea de última lectura. Este archivo leerá el proceso antes de ir al archivo de log ya que la línea indica el renglón en el archivo de log próxima a leer.

- Cada 'x'cantidad de tiempo, se dispara el proceso de lectura del archivo de log del proxy, teniendo en cuenta fecha y línea de última corrida. Esto lo realiza el *deamon cron* del Unix, según el archivo crontab con las especificaciones para la corrida del proceso.

- Se seleccionan los requerimientos considerando sólo los que tienen un *hit de cache*, es decir los que su campo cached es distinto de 0 y los que tengan el header HTTP (GET,HEAD,HTTP y POST).

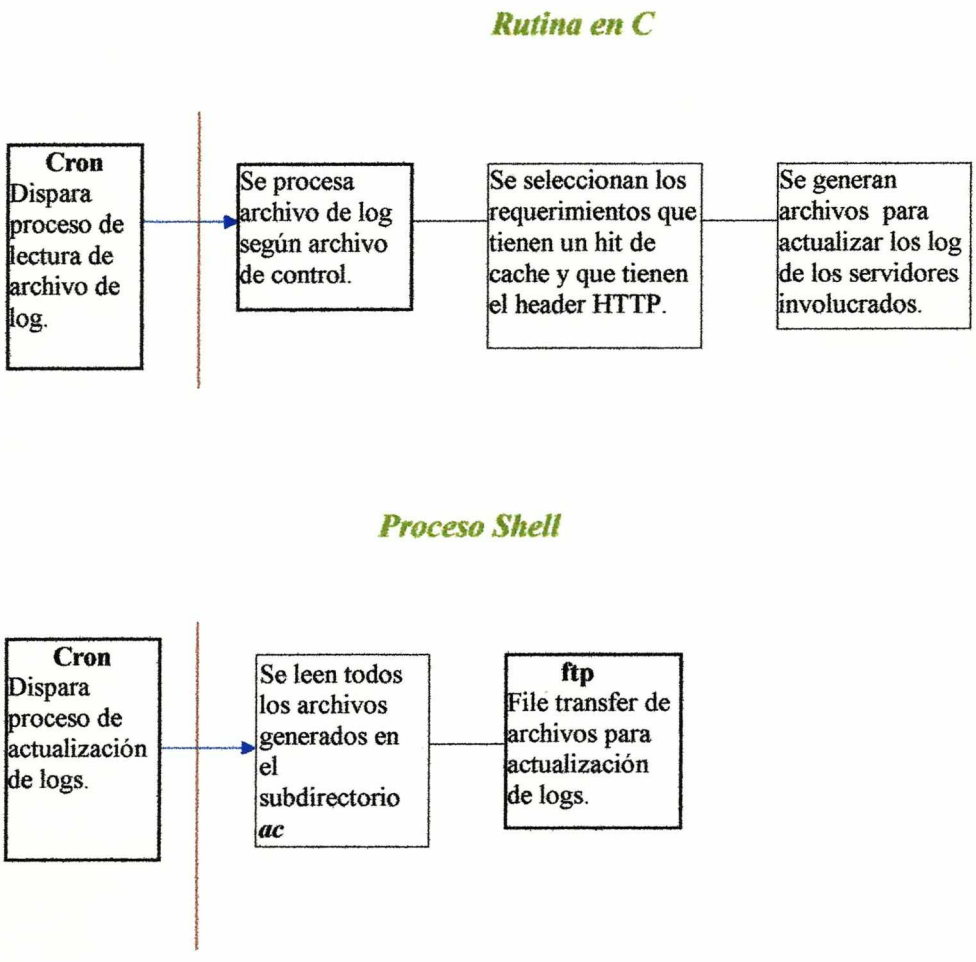
- Se arma así un archivo adicional, con idéntico formato al archivo de log del servidor. Se genera un archivo de actualización para cada servidor que se informe en el archivo de log.

- Se actualiza archivo de control, con la nueva información, es decir fecha y última línea leída.

- Cada 'y'cantidad de tiempo, se dispara el proceso de actualización de los archivos de log de los servidores. Esto lo realiza el *deamon cron* del Unix, según el archivo crontab con las especificaciones para la corrida del proceso.

Gráfico de Procesos:

Gráficamente se vería de la siguiente manera:



Características del escenario de prueba del modelo:

- Tanto el log file del proxy, cómo el de los servidores reales, deben tener en mismo formato.-

- Los archivos de logs, se deben registrar por día.-

- En el archivo de log del proxy se debe poder identificar el hit de cache, por ejemplo en un formato extendido es posible definir el campo cached, el cual indica con un 1 si se produjo un hit y con un 0 si se produjo un miss:

#fields: time cs-method cs-uri cached

Existen formatos propietarios que tienen un campo type/code, el cuál indica cómo el requerimiento fue manejado por el cache, y así se identifica un hit o un miss con la palabra en sí, cómo por ejemplo el Squid.

- Se debe definir un usuario en cada servidor, para poder acceder al mismo y actualizar el archivo de log. Dicho usuario tendrá permisos de escritura sobre el archivo de log de accesos.-

Una Simulación del modelo:

Para probar dicho comportamiento, como un paso inicial, implementé una simulación del modelo en FoxPro para Unix, ya que éste es para mí un lenguaje familiar y de programación sencilla.

Definí un ambiente de prueba del mismo a través de dos máquinas, una con Unix y la otra con Linux, conectadas vía TCP/IP, considerando la Linux como si fuese un proxy y la otra cómo si fuese un servidor de Web, y tomando un archivo de log accesos fijo que recuperé de un proxy Apache de la facultad.

El proceso lee las líneas de entrada en archivo de log y las filtra según ciertas condiciones para generar un archivo de actualización llamado *actual.txt*, que será el que se enviará al servidor para adicionar a su propio archivo de log.

El nombre que se utiliza aquí para el archivo de log del proxy, es un nombre supuesto, el cuál se forma por dos posiciones para el año, dos para el mes y dos para el numero de día, y con extensión *txt*.

Los movimientos que se adicionan al archivo de actualización, no son sólo los requerimientos que tienen un hit de caché, tal como se describe en el modelo, sino todos los requerimientos que tienen el header HTTP, ya que el log de prueba tiene formato común y por lo tanto no contiene el campo *cached* cómo es posible definir en el formato extendido.

El pseudocódigo es el siguiente:

```
Toma mes, día y año de la fecha del sistema.
Arma nombre del archivo de log.
Abre archivo control.txt
If (no existe archivo control.txt)
   Crea archivo control.txt
Else
   Toma última línea leída para esa fecha
Endif
If (no existe archivo actual.txt)
```

Crea archivo actual.txt
Endif
Abre archivo de log
Inicializa contador de líneas leídas y contador de líneas grabadas
Do while (haya líneas en archivo de log)
Lee línea
If (tiene el header HTTP)
If (hay líneas en archivo de control)
If (contador de líneas > línea en archivo de control)
Graba línea en actual.txt
Incrementa contador de líneas grabadas
Endif
Else
Graba línea en actual.txt
Incrementa contador de líneas grabadas
Endif
Endif
Incrementa contador de líneas
Enddo
Cierra archivo de log
Cierra archivo actual.txt
If (contador de líneas grabadas <> 0)
Graba fecha y ultima línea leída en control.txt
Endif
Cierra archivo control.txt
Retorna

El código fuente en FoxPro es el siguiente:

```
**--------------------------------------------------------------------------------
** PROGRAMA: LOG1.PRG
** OBJETIVO: Implementacion (Modelo 1), para actualizacion de logs de servidores
**--------------------------------------------------------------------------------

SET EXCLU OFF
SET DATE FRENCH
SET HISTORY OFF
SET TALK OFF

mes=SUBSTR(CMONTH(DATE()),1,3)
dia=SUBSTR(DTOC(DATE()),1,2)
anio=SUBSTR(DTOC(DATE()),7,2)
archi=anio+mes+dia+".txt"
**--- Abre el archivo control para tomar fecha y linea de ultima lectura
line=0
IF .NOT. FILE('control.txt')
 f_con=FCREATE('control.txt')
```

```
ELSE
 f_con=FOPEN('control.txt',2)
 can=FSEEK(f_con,0,2)
 IF can > 0
  =FSEEK(f_con,can-17)
  regi=FREAD(f_con,17)
  IF CTOD(SUBSTR(regi,1,8))=DATE()
   line=VAL(SUBSTR(regi,10,6))
  ENDIF
 ENDIF
ENDIF
**--- lee el archivo de log, segun inf.archivo control
**--- y genera actual.txt

IF .NOT. FILE ('&archi')
 WAIT "Archivo "+archi+" no existe" WINDOW
 RETURN
ENDIF

f_log=FOPEN('&archi')
f_ac=FCREATE('actual.txt')

can=FSEEK(f_log,0,2)   && cuenta cantidad de caracteres totales
=FSEEK(f_log,0)        && va al principio del archivo
STORE 0 TO can_lin,rg
i=1
DO WHILE i<=can
 **--- arma el registro
 li=""
 DO WHILE i<=can
  cara=FREAD(f_log,1)
  IF cara<>CHR(13)
   li=li+cara
   i=i+1
  ELSE
   EXIT
  ENDIF
 ENDDO
 IF LEN(li)<>0
  can_lin=can_lin+1
  li=li+CHR(13)
  **--- Me fijo si tiene el header HTTP
  xpos=AT('http:',li)
  IF xpos<>0
   **--- quiere decir que tiene el header HTTP
   IF line<>0
    IF can_lin>=line          && si es la linea siguiente
     =FWRITE(f_ac,li)
     rg=rg+1
    ENDIF
```

```
    ELSE
     =FWRITE(f_ac,li)
     rg=rg+1
    ENDIF
   ENDIF
  ENDIF
ENDDO
=FCLOSE(f_log)
=FCLOSE(f_ac)

**--- actualiza archivo de control
IF rg<>0
 lin_c=DTOC(DATE())+" "+STR(can_lin,6)
 =FPUTS(f_con,lin_c)
ENDIF
=FCLOSE(f_con)
QUIT
```

Implementación del modelo:

La implementación del modelo se realizó en lenguaje C, ya que su generalidad lo hace conveniente para éste tipo de procesos.

Se tomó el mismo ambiente de prueba que en la simulación anterior: dos máquinas conectadas vía TCP/IP, una Unix y la otra Linux. En la Linux correrían los procesos del proxy manejados por el cron: lectura del archivo de log y generación de archivos de actualización, y actualización de los logs reales de cada servidor involucrado.

El pseudocódigo del proceso C es el siguiente:

```
Lee fecha del sistema
Arma nombre de archivo de log del proxy según fecha leída
If (existe archivo de control)
   Abre archivo de control de líneas leídas (control.txt)
   While (haya líneas en archivo de control)
        If (fecha igual a fecha del sistema)
            Guarda número de ultima línea leída
   Cierra archivo de control
Inicializa contador de líneas a 0
Abre archivo de log
While (hay líneas en archivo de log)
     Incrementa contador de líneas
     If (tiene el header HTTP y un hit de cache)
         Arma nombre de archivo de actualización, según DNS o IP
         Abre archivo de actualización
         If (número de última línea leída > 0)
             If (contador de líneas > a número de última línea leída)
                 Agrega línea a archivo de actualización
         Else
             Agrega línea a archivo de actualización
         Cierra  archivo de actualización
```

Cierra archivo de log

Abre archivo de control

Guarda fecha y número de línea leída

Cierra archivo de control

El código fuente es el siguiente:

```
/*-------------------------------------------------------------------------*/
/* PROGRAMA: LOG1.C                                                        */
/* OBJETIVO: Implementacion (Modelo 1), para actualizacion de logs de      */
/*                servidores (en C)                                        */
/*-------------------------------------------------------------------------*/


#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#define MAXI 1000
#define MAX 50

int getline(char li[],FILE *f_log);
int strindex(char li[],char bus[]);
int nomarchi(char li[],int pos,char actu[]);
int subs(char *fecha,int p,int t,char vars[]);
char bus[]="http://";     /* Patron a buscar */

main()
{
int line,can_lin,rg,pos,m,l,fecha1;
time_t fec,*ti;
char *fecha;
char li[MAXI],anio[4],mes[3],dia[2];
char actu[MAX],vari[MAX],archi[MAX];
char lin_c1[9],lin_c2[2],lin_c3[6],lin_c[MAX];
FILE *f_con;
FILE *f_ac;
FILE *f_log;

/* Inicializa el nombre del archivo de log, segun fecha del sistema */

printf("%s\n","toma fecha del sistema..");
fec=time(ti);
if (fec==-1)
{
 printf("%s","fecha del sistema no disponible");
 return;
}
fecha=ctime(ti);

printf("%s\n","arma nombre de archivo archi...");
```

```
/* toma anio+mes+dia, y arma nombre de archivo */

m=subs(fecha,20,23,anio);
strcpy(archi,anio);
m=subs(fecha,4,6,mes);
strcat(archi,mes);
m=subs(fecha,8,9,dia);
strcat(archi,dia);
fecha1=atoi(archi);
sprintf(lin_c1,"%s",archi);
strcat(archi,".txt");
printf("%s\n",archi);


/* Abre el archivo control para tomar fecha y linea de ultima lectura*/

printf("%s\n","lee archivo de control...");
line=0;
if ((f_con=fopen("control.txt","r"))!=NULL)
{
 while (getline(li,f_con)>0)
 {
  m=subs(li,0,9,vari);
  printf("%d %d\n",atoi(vari),fecha1);
  if (atoi(vari)==fecha1)
  {
   m=subs(li,10,16,vari);
   line=atoi(vari);
   printf("%d\n",line);
  }
 }
 fclose(f_con);
}

/* lee el archivo de log, segun inf.archivo control y genera actual.txt */

if ((f_log=fopen(archi,"r"))==NULL)
{
 printf("Archivo %s no existe\n",archi);
 return;
}

printf("%s\n","procesa archivo de logs...");
can_lin=0;
while (getline(li,f_log)>0)                    /* lee lineas del archivo de log */
{
 ++can_lin;
 printf("%s\n",li);
 if ((pos=strindex(li,bus))>=0)                /* si tiene el header HTTP */
 {
```

```
   if (nomarchi(li,pos,actu)>0)          /* nombre del arch.de actualizacion */
   {
    f_ac=fopen(actu,"a");                /* abre archivo de actualizacion */
    if (line>0)                          /* si se leyeron lineas anteriormente */
    {
     if (can_lin>line)                   /* si es la linea siguiente */
     {
      fputs(li,f_ac);
      rg++;
     }
    }
    else
    {
     fputs(li,f_ac);
     rg++;
    }
    fclose(f_ac);
   }
  }
}
fclose(f_log);

/* actualiza archivo de control */

printf("%s\n","actualiza archivo de control...");
f_con=fopen("control.txt","a");
if (rg>0)
{
 sprintf(lin_c2,"%s"," ");
 sprintf(lin_c3,"%d",can_lin);
 strcat(lin_c,lin_c1);
 strcat(lin_c,lin_c2);
 strcat(lin_c,lin_c3);
 l=strlen(lin_c);
 lin_c[l]='\n';
 lin_c[++l]='\0';
 fputs(lin_c,f_con);
}
fclose(f_con);
printf("%s\n","fin de programa...");

}
```

Procedimientos y funciones:

```
/*-------------------------------------------------------*/
/* getline: trae una linea y la deja en s y regresa su longitud      */

int getline(char s[],FILE *fp)
{
  int c,i;
  i=0;
  while ((c=getc(fp))!=EOF && c!='\n')
  {
    s[i++]=c;
  }
  if (c=='\n')
    s[i++]=c;
  s[i]='\0';
  return i;
}

/*-------------------------------------------------------*/
/*strindex: regresa el indice de t en s, -1 si no existe            */

int strindex(char s[],char t[])
{
  int i,j,k;
  for (i=0;s[i]!='\0';i++) {
    for (j=i,k=0;t[k]!='\0' && s[j]==t[k];j++,k++)
      ;
    if (k>0 && t[k]=='\0')
      return i;
  }
  return -1;
}

/*-------------------------------------------------------*/
/*nomarchi: regresa el nombre del archivo de actualizacion en nom     */

int nomarchi(char s[],int p,char nom[])
{
  int i,j;
  j=0;
  nom[j++]='.';
  nom[j++]='/';
  nom[j++]='a';
  nom[j++]='c';
  nom[j++]='/';
  for (i=p+7;s[i]!='\0' && s[i]!='/';i++) {
    nom[j++]=s[i];
  }
```

```
  nom[j++]='.';
  nom[j++]='a';
  nom[j++]='c';
  nom[j]='\0';
  return j;
}

/*-------------------------------------------------------*/
/*subs: regresa de s el string que va desde p a p+t caracteres */

int subs(char s[],int p,int t,char vars[])
{
  int i,j;
  j=0;
  for (i=p;s[i]!='\0' && i<=t;i++) {
    if (s[i]!=32)
      vars[j++]=s[i];
  }
  vars[j]='\0';
  return j;
}
/*-------------------------------------------------------*/
```

Proceso Shell:

Para completar la ejecución del *modelo 1*, se define un proceso shell, ***actu***, el cuál se encarga de realizar la transferencia de cada archivo de actualización al servidor correspondiente.

Por ejemplo podría ser:

proceso actu:
```
for i in `ls *.ac`
  do
    echo $i
    cp $i actual.txt
    ftp -in <tran > /dev/null
    rm $i
  done
```

Este proceso lee los archivos que se generaron en el directorio ***ac,*** que contiene los archivos de actualización de logs. Copia cada uno como actual.txt, realiza el file transfer al servidor y luego elimina el archivo leído del directorio ac.

El *ftp* tiene como entrada el archivo ***tran***, que es dónde se encuentran los comandos para el mismo.

archivo tran:
```
open [DNS o IP]
user [usuario] [passwd]
append actual.txt [nombre del archivo de log del servidor]
close
```

En éste archivo se detallan los comandos de apertura de sesión, usuario y password de acceso al servidor, comando que adiciona el archivo actual.txt al archivo de log del servidor real, y cierre de sesión lo cuál marca el fin de conexión con el servidor remoto.

Cómo cada archivo de actualización lleva como nombre el DNS o IP del servidor original, el archivo ***tran*** con las especificaciones del *ftp*, debe ser generado para cada archivo de actualización que se va a transferir. Así el proceso original queda de la siguiente manera:

```
for i in `ls ./ac/* | awk '{print(substr($0,6,length($0)))}'`
   do
        echo $i
        cp ./ac/$i actual.txt

        echo 'open '$i > tran
        echo 'user tesis tesis'>>tran
        echo 'append actual.txt access.log'>>tran
        echo 'close'>>tran

        ftp -in <tran  > /dev/null
        rm $i
done
```

Archivo Crontab:

Tanto el programa de generación de los archivos con las novedades cómo el proceso shell de actualización de logs, se ejecutan en background a través del comando de tareas programadas (comando *cron* en Unix).

Este comando es un *clock deamon*, que ejecuta comandos en días y horas determinados.

Para ésto es necesario definir un archivo con las especificaciones de corrida de cada proceso, así que para éste caso en particular, se definiría el archivo en el usuario que se use para la actualización de logs. Dicho archivo en Unix se llama *crontab* y reside en */usr/spool/cron/crontabs/*<username>

Este archivo consiste de seis campos, cada campo separado por un espacio en blanco. Los primeros 5 corresponden a: minuto (0-59), hora (0-23), día del mes (1-31), mes del año (1-12), y día de la semana (0-6, 0=Domingo). El símbolo '*' en cualquiera de los campos significa siempre. En el último campo se especifica el comando a ejecutar.

Por ejemplo para éste caso en particular el archivo crontab podría ser el siguiente:

30 * * * * /usr/<username>/log1.out
0 * * * * /usr/<username>/actu

Es decir que al minuto 30 de todas las horas, todos los días de la semana, todos los meses, ejecuta el programa ***log1*** de lectura del archivo de log del proxy, y al minuto 0 de todas las horas, todos los días de la semana, todos los meses, ejecuta el proceso ***actu*** que envía las actualizaciones a los servidores correspondientes.

Aplicación y prueba del modelo para un proxy Squid:

Para probar el funcionamiento del modelo 1 en un proxy real, se instala un proxy Squid en el LINTI.

El proxy Squid, registra su archivo de logs de accesos en el directorio /usr/local/squid/logs/access.log

El formato de este archivo no sigue la regla del formato común, ni la del extendido, tiene un formato propietario que consiste de los siguientes campos *[23]*:

timestamp elapsed src-address type/code size method URL

en dónde:

timestamp: fecha y hora en que se completó el requerimiento, expresado en milisegundos.

elapsed: tiempo elapsed del requerimiento en milisegundos

src-address: dirección IP del cliente

type: indica cómo el requerimiento fue manejado por el cache.

code: codigo de estado.

size: tamaño de los datos requeridos en bytes.

method: método del requerimiento (GET,POST,etc.)

URL: la URL del requerimiento.

Los tipos pueden ser:

"TCP_" referencias a requerimientos sobre el port HTTP (3128)

TCP_HIT: una copia válida del objeto requerido está en el cache.

TCP_MISS: el objeto requerido no está en el cache.

TCP_EXPIRED: el objeto está en el cache, pero expiró.

TCP_REFRESH: el usuario fuerza un refresh ("reload")

TCP_IFMODSINCE: un requerimiento GET If_Modified_Since.

TCP_SWAPFAIL: el objeto está en el caché, pero no puede ser accedido.

TCP_DENIED: acceso denegado para éste requerimiento.

"UDP_"referencias a requerimientos sobre el port ICP (3130)

UDP_HIT: una copia válida del objeto requerido, está en el caché.

UDP_MISS: el objeto requerido no está en el cache.

UDP_DENIED: acceso denegado para el requerimiento.

Una muestra del log de accesos es la siguiente:

910208698.166 1278 163.10.10.6 ERR_DNS_FAIL/400 905 GET http://www.sendmail.rg/ - NONE/- -
910208707.577 1364 163.10.10.6 TCP_MISS/200 3978 GET http://www.sendmail.org/ - DIRECT/www.sendmail.org text/html
910208709.018 1336 163.10.10.6 TCP_MISS/200 3073 GET http://www.sendmail.org/gif/batorng.gif - DIRECT/www.sendmail.org image/gif
910208709.089 1408 163.10.10.6 TCP_MISS/200 2736 GET http://www.sendmail.org/gif/bakorng.gif - DIRECT/www.sendmail.org image/gif
910208709.094 1390 163.10.10.6 TCP_MISS/200 272 GET http://www.sendmail.org/gif/dotorng.gif - DIRECT/www.sendmail.org image/gif
910208709.158 1294 163.10.10.6 TCP_MISS/200 1154 GET http://www.sendmail.org/gif/barorng.gif - DIRECT/www.sendmail.org image/gif
910208941.566 22783 163.10.10.6 TCP_MISS/200 3635 GET http://squid.nlanr.net/ - DIRECT/squid.nlanr.net text/html
910208942.535 1101 163.10.10.6 TCP_MISS/304 136 GET http://squid.nlanr.net/Squid/Squidlogo2.gif - DIRECT/squid.nlanr.net -
910208942.685 1095 163.10.10.6 TCP_MISS/304 134 GET http://squid.nlanr.net/Squid/squidnow.gif - DIRECT/squid.nlanr.net -
910208947.770 6181 163.10.10.6 TCP_MISS/304 142 GET http://www.nlanr.net/Images/cache_now.gif - DIRECT/www.nlanr.net -
910208950.193 8596 163.10.10.6 TCP_MISS/304 143 GET http://cache.jp.apan.net/proxy-checker/proxy-checker.gif - DIRECT/cache.jp.apan.net -
910208979.085 4853 163.10.10.6 TCP_MISS/200 4495 GET http://squid.nlanr.net/Squid/Scripts/ - DIRECT/squid.nlanr.net text/html

910209060.772 1302 163.10.10.6 TCP_MISS/304 135 GET http://squid.nlanr.net/Squid/documentation.html - DIRECT/squid.nlanr.net -
910209081.595 14388 163.10.10.6 TCP_MISS/304 88 GET http://cache.is.co.za/squid/ - DIRECT/cache.is.co.za -
910209083.758 2043 163.10.10.6 TCP_MISS/304 88 GET http://cache.is.co.za/squid/images/forwards.gif - DIRECT/cache.is.co.za -
910209083.770 2053 163.10.10.6 TCP_MISS/304 88 GET http://cache.is.co.za/squid/images/index.gif - DIRECT/cache.is.co.za -
910209113.858 1957 163.10.10.6 TCP_MISS/304 88 GET http://cache.is.co.za/squid/detail/acl.html - DIRECT/cache.is.co.za -
910209115.956 1979 163.10.10.6 TCP_MISS/304 88 GET http://cache.is.co.za/squid/images/back.gif - DIRECT/cache.is.co.za -
910209116.016 2039 163.10.10.6 TCP_MISS/304 88 GET http://cache.is.co.za/squid/images/next.gif - DIRECT/cache.is.co.za -
910209125.260 1922 163.10.10.6 TCP_MISS/304 88 GET http://cache.is.co.za/squid/detail/client.html - DIRECT/cache.is.co.za -
910209162.396 2965 163.10.10.6 TCP_MISS/200 5403 GET http://cache.is.co.za/squid/index.html - DIRECT/cache.is.co.za text/html
910209177.628 6876 163.10.10.6 TCP_MISS/200 22578 GET http://squid.nlanr.net/Squid/FAQ/FAQ.html - DIRECT/squid.nlanr.net text/html
910209194.102 2213 163.10.10.6 TCP_MISS/200 17993 GET http://squid.nlanr.net/Squid/FAQ/FAQ-6.html - DIRECT/squid.nlanr.net text/html
910209527.317 1032 163.10.10.6 TCP_MISS/304 136 GET http://squid.nlanr.net/Squid/features.html - DIRECT/squid.nlanr.net -
910209636.719 639 163.10.10.6 ERR_DNS_FAIL/400 905 GET http://www.sendmail.rg/ - NONE/- -
910209646.336 1920 163.10.10.6 TCP_MISS/200 13141 GET http://www.altavista.com/ - DIRECT/www.altavista.com text/html
910209647.094 1185 163.10.10.6 TCP_MISS/200 4254 GET http://www.altavista.com/av/gifs/new/front_hdr.gif - DIRECT/www.altavista.com image/gif
910209647.167 1258 163.10.10.6 TCP_MISS/200 227 GET http://www.altavista.com/av/gifs/pixel.gif - DIRECT/www.altavista.com image/gif
910209647.427 1069 163.10.10.6 TCP_MISS/200 227 GET http://www.altavista.com/av/gifs/pixel.gif - DIRECT/www.altavista.com image/gif
910209647.628 1718 163.10.10.6 TCP_MISS/200 1539 GET http://ad.doubleclick.net/ad/altavista.digital.com/sponsor-button/homepage_upperright;sz=120x60 - DIRECT/ad.doubleclick.net image/gif
910209648.202 1068 163.10.10.6 TCP_MISS/200 227 GET http://www.altavista.com/av/gifs/pixel.gif - DIRECT/www.altavista.com image/gif
910209648.268 1063 163.10.10.6 TCP_MISS/200 451 GET http://www.altavista.com/av/gifs/dart.gif - DIRECT/www.altavista.com image/gif
910209648.531 1069 163.10.10.6 TCP_MISS/200 150 GET http://ad.doubleclick.net/ad/altavista.digital.com/sponsor-button/homepage_text1;sz=6x6;num=60b - DIRECT/ad.doubleclick.net image/gif
910209648.702 1030 163.10.10.6 TCP_MISS/200 150 GET http://ad.doubleclick.net/ad/altavista.digital.com/sponsor-button/homepage_text2;sz=6x6;num=60b - DIRECT/ad.doubleclick.net image/gif
910209649.459 1220 163.10.10.6 TCP_MISS/200 150 GET http://ad.doubleclick.net/ad/altavista.digital.com/sponsor-button/homepage_text4;sz=6x6;num=60b - DIRECT/ad.doubleclick.net image/gif
910209649.838 1192 163.10.10.6 TCP_MISS/200 150 GET http://ad.doubleclick.net/ad/altavista.digital.com/sponsor-button/homepage_text3;sz=6x6;num=60b - DIRECT/ad.doubleclick.net image/gif
910209649.842 1106 163.10.10.6 TCP_MISS/200 153 GET http://ad.doubleclick.net/dot.gif - DIRECT/ad.doubleclick.net image/gif
910209650.283 1668 163.10.10.6 TCP_MISS/200 3634 GET http://ad.doubleclick.net/ad/altavista.digital.com/sponsor-button/homepage125;sz=125x125 - DIRECT/ad.doubleclick.net image/gif
910209650.471 980 163.10.10.6 TCP_MISS/200 971 GET http://www.altavista.com/av/gifs/new/ns.gif - DIRECT/www.altavista.com image/gif
910209660.019 1580 163.10.10.6 TCP_MISS/302 542 GET http://jump.altavista.com/cat/comp - DIRECT/jump.altavista.com text/html
910209662.546 774 163.10.10.6 TCP_MISS/200 352 GET http://altavista.looksmart.com/i/izf_bg.gif - DIRECT/altavista.looksmart.com image/gif
910209662.589 812 163.10.10.6 TCP_MISS/200 281 GET http://altavista.looksmart.com/i/b.gif - DIRECT/altavista.looksmart.com image/gif
910209662.594 819 163.10.10.6 TCP_MISS/200 307 GET http://altavista.looksmart.com/i/zf_pow.gif - DIRECT/altavista.looksmart.com image/gif

910209662.727 2699 163.10.10.6 TCP_MISS/200 8126 GET http://altavista.looksmart.com/r - DIRECT/altavista.looksmart.com text/html
910209663.553 831 163.10.10.6 TCP_MISS/200 319 GET http://altavista.looksmart.com/i/bluearro.gif - DIRECT/altavista.looksmart.com image/gif
910209663.672 787 163.10.10.6 TCP_MISS/200 702 GET http://altavista.looksmart.com/i/izf_homeands.gif - DIRECT/altavista.looksmart.com image/gif
910209663.795 1140 163.10.10.6 TCP_MISS/200 3397 GET http://www.altavista.com/av/gifs/partners/categories_head.gif - DIRECT/www.altavista.com image/gif
910209664.633 1013 163.10.10.6 TCP_MISS/200 227 GET http://altavista.com/av/gifs/pixel.gif - DIRECT/altavista.com image/gif
910209664.657 941 163.10.10.6 TCP_MISS/302 217 GET http://ad.doubleclick.net/ad/altavista.looksmart.com/sponsor-button/us/ros;se=us53832;se=us1;ord=364718 - DIRECT/ad.doubleclick.net text/html
910209664.882 2194 163.10.10.6 TCP_MISS/200 10822 GET http://ad.doubleclick.net/ad/altavista.looksmart.com/explore;e=us53832;e=us1;ord=931633 - DIRECT/ad.doubleclick.net image/gif
910209666.186 1487 163.10.10.6 TCP_MISS/200 7113 GET http://ad.doubleclick.net/viewad/8556-120ticket_f.gif - DIRECT/ad.doubleclick.net image/gif
910209683.836 1828 163.10.10.6 TCP_MISS/200 8440 GET http://altavista.looksmart.com/eus1/eus53832/eus62416/r - DIRECT/altavista.looksmart.com text/html
910209684.200 985 163.10.10.6 TCP_MISS/302 217 GET http://ad.doubleclick.net/ad/altavista.looksmart.com/sponsor-button/us/ros;se=us62416;se=us53832;se=us1;ord=799479 - DIRECT/ad.doubleclick.net text/html
910209684.247 1030 163.10.10.6 TCP_MISS/302 218 GET http://ad.doubleclick.net/ad/altavista.looksmart.com/explore;e=us62416;e=us53832;e=us1;ord=670768 - DIRECT/ad.doubleclick.net text/html
910209686.255 1997 163.10.10.6 TCP_MISS/200 10540 GET http://ad.doubleclick.net/viewad/5038-468x60_ears2.gif - DIRECT/ad.doubleclick.net image/gif
910209688.268 4359 163.10.10.6 TCP_MISS/200 327 GET http://altavista.looksmart.com/i/flag_urllist.gif - DIRECT/altavista.looksmart.com image/gif
910209701.118 1821 163.10.10.6 TCP_MISS/200 7221 GET http://altavista.looksmart.com/r - DIRECT/altavista.looksmart.com text/html
910209701.426 956 163.10.10.6 TCP_MISS/302 217 GET http://ad.doubleclick.net/ad/altavista.looksmart.com/sponsor-button/us/ros;se=us53832;se=us1;ord=906015 - DIRECT/ad.doubleclick.net text/html
910209701.475 979 163.10.10.6 TCP_MISS/302 215 GET http://ad.doubleclick.net/ad/altavista.looksmart.com/explore;e=us53832;e=us1;ord=744706 - DIRECT/ad.doubleclick.net text/html
910209714.363 1576 163.10.10.6 TCP_MISS/200 13163 GET http://www.altavista.com/cgi-bin/query - DIRECT/www.altavista.com text/html
910209731.713 2564 163.10.10.6 TCP_MISS/200 14701 GET http://www.altavista.com/cgi-bin/query - DIRECT/www.altavista.com text/html
910209732.442 1184 163.10.10.6 TCP_MISS/200 2192 GET http://www.altavista.com/av/gifs/new/result_hdr.gif - DIRECT/www.altavista.com image/gif
910209733.014 976 163.10.10.6 TCP_MISS/200 369 GET http://ad.doubleclick.net/adl/altavista.digital.com/result_front;kw=squid+logs;ord=1301933300 - DIRECT/ad.doubleclick.net text/html
910209735.006 1957 163.10.10.6 TCP_MISS/200 8151 GET http://ad.doubleclick.net/viewad/0-exfpav.gif - DIRECT/ad.doubleclick.net image/gif
910209765.959 2762 163.10.10.6 TCP_MISS/200 1994 GET http://163.130.4.101/squid/logs/ - DIRECT/163.130.4.101 text/html
910209767.379 1933 163.10.10.6 TCP_MISS/200 385 GET http://163.130.4.101/icons/blank.gif - DIRECT/163.130.4.101 image/gif
910209767.407 1943 163.10.10.6 TCP_MISS/200 453 GET http://163.130.4.101/icons/back.gif - DIRECT/163.130.4.101 image/gif
910209767.460 1368 163.10.10.6 TCP_MISS/200 462 GET http://163.130.4.101/icons/folder.gif - DIRECT/163.130.4.101 image/gif
910209767.484 2021 163.10.10.6 TCP_MISS/200 466 GET http://163.130.4.101/icons/text.gif - DIRECT/163.130.4.101 image/gif
910209768.601 1180 163.10.10.6 TCP_MISS/200 482 GET http://163.130.4.101/icons/unknown.gif - DIRECT/163.130.4.101 image/gif
910209782.669 1224 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/000-000.gif - DIRECT/163.130.4.101 text/html
910209782.790 3277 163.10.10.6 TCP_MISS/200 8150 GET http://163.130.4.101/squid/logs/19981010.html - DIRECT/163.130.4.101 text/html
910209783.291 1111 163.10.10.6 TCP_MISS/404 303 GET http://163.130.4.101/img/ampm-bar.gif - DIRECT/163.130.4.101 text/html

910209783.331 1150 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/090-100.gif - DIRECT/163.130.4.101 text/html
910209783.855 1158 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/080-090.gif - DIRECT/163.130.4.101 text/html
910209784.083 1097 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/060-070.gif - DIRECT/163.130.4.101 text/html
910209784.456 1146 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/040-050.gif - DIRECT/163.130.4.101 text/html
910209784.489 1139 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/020-030.gif - DIRECT/163.130.4.101 text/html
910209784.996 1126 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/000-010.gif - DIRECT/163.130.4.101 text/html
910209785.480 1379 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/050-060.gif - DIRECT/163.130.4.101 text/html
910209785.947 1483 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/010-020.gif - DIRECT/163.130.4.101 text/html
910209785.984 1495 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/030-040.gif - DIRECT/163.130.4.101 text/html
910209786.130 1117 163.10.10.6 TCP_MISS/404 302 GET http://163.130.4.101/img/070-080.gif - DIRECT/163.130.4.101 text/html
910209808.601 1394 163.10.10.6 TCP_MISS/200 471 GET http://163.130.4.101/squid/logs/last/ - DIRECT/163.130.4.101 text/html
910209837.906 4361 163.10.10.6 TCP_MISS/200 6576 GET http://www.altavista.com/cgi-bin/query - DIRECT/www.altavista.com text/html
910209838.105 994 163.10.10.6 TCP_MISS/200 590 GET http://www.altavista.com/av/gifs/graph.gif - DIRECT/www.altavista.com image/gif
910209838.665 994 163.10.10.6 TCP_MISS/200 542 GET http://ad.doubleclick.net/adl/altavista.digital.com/result_front;kw=squid+logs;ord=-2081587571 - DIRECT/ad.doubleclick.net text/html
910209840.690 1987 163.10.10.6 TCP_MISS/200 10048 GET http://ad.doubleclick.net/viewad/6730-Excbutt.gif - DIRECT/ad.doubleclick.net image/gif
910209885.463 5981 163.10.10.6 TCP_MISS/200 20785 GET http://www.altavista.com/cgi-bin/query - DIRECT/www.altavista.com text/html
910209886.721 1066 163.10.10.6 TCP_MISS/200 566 GET http://ad.doubleclick.net/adl/altavista.digital.com/result_front;kw=squid+logs;ord=-895585850 - DIRECT/ad.doubleclick.net text/html
910209888.749 1994 163.10.10.6 TCP_MISS/200 9349 GET http://ad.doubleclick.net/viewad/19994-ellabestmusic.gif - DIRECT/ad.doubleclick.net image/gif
910209912.258 8279 163.10.10.6 TCP_MISS/200 3809 GET http://cssvh.cs.knct.ac.jp/~fgk/squid/install.html - DIRECT/cssvh.cs.knct.ac.jp text/html
910209912.642 1655 163.10.10.6 TCP_MISS/200 219 GET http://cssvh.cs.knct.ac.jp/~fgk/.grp/99CC99.gif - DIRECT/cssvh.cs.knct.ac.jp image/gif
910209912.675 1688 163.10.10.6 TCP_MISS/200 219 GET http://cssvh.cs.knct.ac.jp/~fgk/.grp/000000.gif - DIRECT/cssvh.cs.knct.ac.jp image/gif
910209913.673 1221 163.10.10.6 TCP_MISS/200 219 GET http://cssvh.cs.knct.ac.jp/~fgk/.grp/508060.gif - DIRECT/cssvh.cs.knct.ac.jp image/gif
910209913.959 2972 163.10.10.6 TCP_MISS/200 2963 GET http://cssvh.cs.knct.ac.jp/~fgk/.grp/2014-03.gif - DIRECT/cssvh.cs.knct.ac.jp image/gif
910209914.150 1430 163.10.10.6 TCP_MISS/200 219 GET http://cssvh.cs.knct.ac.jp/~fgk/.grp/508060.gif - DIRECT/cssvh.cs.knct.ac.jp image/gif
910209914.177 1485 163.10.10.6 TCP_MISS/200 245 GET http://cssvh.cs.knct.ac.jp/~fgk/.grp/arrow-l.gif - DIRECT/cssvh.cs.knct.ac.jp image/gif
910209914.979 1269 163.10.10.6 TCP_MISS/200 227 GET http://cssvh.cs.knct.ac.jp/~fgk/.grp/transparency.gif - DIRECT/cssvh.cs.knct.ac.jp image/gif
910209945.499 3495 163.10.10.6 TCP_MISS/200 20772 GET http://www.altavista.com/cgi-bin/query - DIRECT/www.altavista.com text/html
910209946.735 1008 163.10.10.6 TCP_MISS/200 566 GET http://ad.doubleclick.net/adl/altavista.digital.com/result_front;kw=squid+configuration+logs;ord=-921946560 - DIRECT/ad.doubleclick.net text/html
910209968.831 6912 163.10.10.6 TCP_MISS/200 19390 GET http://www.altavista.com/cgi-bin/query - DIRECT/www.altavista.com text/html
910209969.922 959 163.10.10.6 TCP_MISS/200 369 GET http://ad.doubleclick.net/adl/altavista.digital.com/result_next;kw=squid+configuration+logs;ord=1313831199 - DIRECT/ad.doubleclick.net text/html
910210002.565 7338 163.10.10.6 TCP_MISS/200 19412 GET http://www.altavista.com/cgi-bin/query - DIRECT/www.altavista.com text/html
910210003.759 1014 163.10.10.6 TCP_MISS/200 369 GET

http://ad.doubleclick.net/adl/altavista.digital.com/result_next;kw=squid+configuration+logs;ord=-909333426 - DIRECT/ad.doubleclick.net text/html
910210019.190 5742 163.10.10.6 TCP_MISS/200 19222 GET http://www.altavista.com/cgi-bin/query - DIRECT/www.altavista.com text/html
910210020.310 973 163.10.10.6 TCP_MISS/200 369 GET http://ad.doubleclick.net/adl/altavista.digital.com/result_next;kw=squid+configuration+logs;ord=-2074305201 - DIRECT/ad.doubleclick.net text/html
910210583.498 488 163.10.10.6 ERR_DNS_FAIL/400 905 GET http://www.sendmail.rg/ - NONE/- -
910210618.253 1438 163.10.10.6 TCP_MISS/200 1876 GET http://home.netscape.com/inserts/images/bonzi_14000ihn2_3netboost_ad.gif - DIRECT/home.netscape.com image/gif
910210618.522 8062 163.10.10.6 TCP_MISS/200 18882 GET http://home.netscape.com/ - DIRECT/home.netscape.com text/html
910210618.665 1907 163.10.10.6 TCP_MISS/200 4134 GET http://home.netscape.com/images/ns_netcenter_newmem.gif - DIRECT/home.netscape.com image/gif
910210620.078 1351 163.10.10.6 TCP_MISS/200 1624 GET http://home.netscape.com/h.js - DIRECT/home.netscape.com application/x-javascript
910210630.939 35 163.10.10.6 TCP_IMS_HIT/304 159 GET http://www.sendmail.org/ - NONE/- -
910210631.208 155 163.10.10.6 TCP_IMS_HIT/304 159 GET http://www.sendmail.org/gif/bakorng.gif - NONE/- -
910210631.255 201 163.10.10.6 TCP_IMS_HIT/304 159 GET http://www.sendmail.org/gif/batorng.gif - NONE/- -
910210631.278 203 163.10.10.6 TCP_IMS_HIT/304 157 GET http://www.sendmail.org/gif/dotorng.gif - NONE/- -
910210631.299 112 163.10.10.6 TCP_IMS_HIT/304 158 GET http://www.sendmail.org/gif/barorng.gif - NONE/- -
910210673.501 3002 163.10.10.6 TCP_MISS/200 18673 GET http://home.netscape.com/ - DIRECT/home.netscape.com text/html
910210699.139 20 163.10.10.6 TCP_IMS_HIT/304 159 GET http://squid.nlanr.net/ - NONE/- -
910210700.265 1025 163.10.10.6 TCP_MISS/304 134 GET http://squid.nlanr.net/Squid/squidnow.gif - DIRECT/squid.nlanr.net -
910210700.326 1086 163.10.10.6 TCP_MISS/304 142 GET http://www.nlanr.net/Images/cache_now.gif - DIRECT/www.nlanr.net -
910210700.647 1407 163.10.10.6 TCP_MISS/304 136 GET http://squid.nlanr.net/Squid/Squidlogo2.gif - DIRECT/squid.nlanr.net -
910210700.737 1496 163.10.10.6 TCP_MISS/304 143 GET http://cache.jp.apan.net/proxy-checker/proxy-checker.gif - DIRECT/cache.jp.apan.net -
910210715.617 1213 163.10.10.6 TCP_MISS/304 136 GET http://squid.nlanr.net/Squid/features.html - DIRECT/squid.nlanr.net -
910210759.482 1610 163.10.10.6 TCP_MISS/200 3412 GET http://home.netscape.com/home/internet-search.html - DIRECT/home.netscape.com text/html
910210762.100 1356 163.10.10.6 TCP_MISS/200 1274 GET http://home.netscape.com/escapes/search/images/nnc_lockup.gif - DIRECT/home.netscape.com image/gif
910210762.174 2661 163.10.10.6 TCP_MISS/200 19586 GET http://home.netscape.com/escapes/search/ntsrchrnd-3.html - DIRECT/home.netscape.com text/html
910210762.681 1544 163.10.10.6 TCP_MISS/200 2665 GET http://home.netscape.com/escapes/search/images/search_lycos.gif - DIRECT/home.netscape.com image/gif
910210763.255 2160 163.10.10.6 TCP_MISS/200 8808 GET http://home.netscape.com/ads/images/ad_1a1.gif - DIRECT/home.netscape.com image/gif
910210763.278 1142 163.10.10.6 TCP_MISS/200 281 GET http://home.netscape.com/escapes/search/images/star.gif - DIRECT/home.netscape.com image/gif

Se debe destacar, que para éste formato en particular, el hit de cache se identifica con el string TCP_HIT en la línea del requerimiento a diferencia de otros formatos en dónde se define el campo cached, el cual tiene un 0 para identificar un miss ó un 1 para identificar un hit, así, para éste caso en particular se debe filtrar cada línea de entrada buscando el string HIT.

Los patrones de filtrado son parametrizables. Cómo una mejora se podrían leer de un archivo de parametros, y así no se tendría que modificar el código.

El programa del modelo 1, se adapta para éste caso y el código es el siguiente:

```
/*-------------------------------------------------------------------------*/
/* PROGRAMA: SQUID.C                                                       */
/* OBJETIVO: Implementacion (Modelo 1), para actualizacion de logs de      */
/*          servidores (en C) para Squid                                   */
/*-------------------------------------------------------------------------*/

#include <stdio.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#define MAXI 1000
#define MAX 50

int getline(char li[],FILE *f_log);
int strindex(char li[],char bus[]);
int nomarchi(char li[],int pos,char actu[]);
int subs(char *fecha,int p,int t,char vars[]);

char bus[]="http://";      /* Patron a buscar para filtrar header HTTP*/
char bus1[]="HIT";         /* Patron a buscar para filtrar HIT de cache*/

main()
{
int line,can_lin,rg,pos,m,l,fecha1;
time_t fec,*ti;
char *fecha;
char li[MAXI],anio[4],mes[3],dia[2];
char actu[MAX],vari[MAX],archi[MAX];
char lin_c1[9],lin_c2[2],lin_c3[6],lin_c[MAX];
FILE *f_con;
FILE *f_ac;
FILE *f_log;

/* Inicializa el nombre del archivo de log, segun fecha del sistema */
printf("%s\n","toma fecha del sistema..");
fec=time(ti);
printf("%d\n",fec);
if (fec==-1)
{
 printf("%s","fecha del sistema no disponible");
 return;
}
fecha=ctime(ti);

printf("%s\n",fecha);
```

```
printf("%s\n","arma nombre de archivo archi...");
/* toma anio+mes+dia, y arma nombre de archivo */
m=subs(fecha,20,23,anio);
strcpy(archi,anio);
m=subs(fecha,4,6,mes);
strcat(archi,mes);
m=subs(fecha,8,9,dia);
strcat(archi,dia);
fecha1=atoi(archi);
sprintf(lin_c1,"%s",archi);
strcat(archi,".txt");
printf("%s\n",archi);

/* Abre el archivo control para tomar fecha y linea de ultima lectura*/
printf("%s\n","lee archivo de control...");
line=0;
if ((f_con=fopen("control.txt","r"))!=NULL)
{
 while (getline(li,f_con)>0)
 {
  m=subs(li,0,9,vari);
  printf("%d %d\n",atoi(vari),fecha1);
  if (atoi(vari)==fecha1)
  {
   m=subs(li,9,16,vari);
   line=atoi(vari);
   printf("%d\n",line);
  }
 }
 fclose(f_con);
}

/* lee el archivo de log, segun inf.archivo control y genera actual.txt */
if ((f_log=fopen(archi,"r"))==NULL)
{
 printf("Archivo %s no existe\n",archi);
 return;
}

printf("%s\n","procesa archivo de logs...");
can_lin=0;
while (getline(li,f_log)>0)          /* lee lineas del archivo de log */
{
 ++can_lin;
 printf("%s\n",li);

 /*filtra requerimientos con el header HTTP y un HIT de cache */
 if (((pos=strindex(li,bus))>=0) & (strindex(li,bus1))>=0)
 {
  if (nomarchi(li,pos,actu)>0)      /* nombre del arch.de actualizacion */
  {
   f_ac=fopen(actu,"a");          /* abre archivo de actualizacion */
   if (line>0)                  /* si se leyeron lineas anteriormente */
   {
```

```
        if (can_lin>line)          /* si es la linea siguiente */
        {
          fputs(li,f_ac);
          rg++;
        }
      }
      else
      {
        fputs(li,f_ac);
        rg++;
      }
      fclose(f_ac);
    }
  }
}
fclose(f_log);

/* actualiza archivo de control */
printf("%s\n","actualiza archivo de control...");
f_con=fopen("control.txt","a");
if (rg>0)
{
 sprintf(lin_c2,"%s"," ");
 sprintf(lin_c3,"%d",can_lin);
 strcat(lin_c,lin_c1);
 strcat(lin_c,lin_c2);
 strcat(lin_c,lin_c3);
 l=strlen(lin_c);
 lin_c[l]='\n';
 lin_c[++l]='\0';
 fputs(lin_c,f_con);
}
fclose(f_con);
printf("%s\n","fin de programa...");

}
/*-------------------------------------------------------------*/
/* getline: trae una linea y la deja en s y regresa su longitud      */
int getline(char s[],FILE *fp)
{
 int c,i;
 i=0;
 while ((c=getc(fp))!=EOF && c!='\n')
 {
   s[i++]=c;
 }
 if (c=='\n')
   s[i++]=c;
 s[i]='\0';
 return i;
}
/*-------------------------------------------------------------*/
/*strindex: regresa el indice de t en s, -1 si no existe          */
int strindex(char s[],char t[])
{
```

```
  int i,j,k;
  for (i=0;s[i]!='\0';i++) {
    for (j=i,k=0;t[k]!='\0' && s[j]==t[k];j++,k++)
      ;
    if (k>0 && t[k]=='\0')
      return i;
  }
  return -1;
}
/*-------------------------------------------------------------*/
/*nomarchi: regresa el nombre del archivo de actualizacion en nom    */
int nomarchi(char s[],int p,char nom[])
{
  int i,j;
  j=0;
  nom[j++]='.';
  nom[j++]='/';
  nom[j++]='a';
  nom[j++]='c';
  nom[j++]='/';
  for (i=p+7;s[i]!='\0' && s[i]!='/';i++) {
    nom[j++]=s[i];
  }
  nom[j]='\0';
  return j;
}
/*-------------------------------------------------------------*/
/*subs: regresa de s el string que va desde p a p+t caracteres */
int subs(char s[],int p,int t,char vars[])
{
  int i,j;
  j=0;
  for (i=p;s[i]!='\0' && i<=t;i++) {
    if (s[i]!=32)
      vars[j++]=s[i];
  }
  vars[j]='\0';
  return j;
}
/*-------------------------------------------------------------*/
```

Realizando una corrida de dicho programa sobre el archivo de log del ejemplo anterior, ésto en el proxy Squid, se generan en el directorio ***ac*** los siguientes archivos:

squid.nlanr.net:

910210699.139 20 163.10.10.6 TCP_IMS_HIT/304 159 http://squid/nlanr.net/ - NONE/- -

www.sendmail.org:

910210630.939 35 163.10.10.6 TCP_IMS_HIT/304 159 GET http://www.sendmail.org/ - NONE/- -
910210631.208 155 163.10.10.6 TCP_IMS_HIT/304 159 GET http://www.sendmail.org/gif/bakorng.gif - NONE/- -
910210631.255 201 163.10.10.6 TCP_IMS_HIT/304 159 GET http://www.sendmail.org/gif/batorng.gif - NONE/- -
910210631.278 203 163.10.10.6 TCP_IMS_HIT/304 157 GET http://www.sendmail.org/gif/dotorng.gif - NONE/- -
910210631.299 112 163.10.10.6 TCP_IMS_HIT/304 158 GET http://www.sendmail.org/gif/barorng.gif - NONE/- -

Se puede apreciar, que sólo se generaron 2 archivos para actualizar, ya que sólo para esos servidores hubo objetos en el caché del proxy y se seleccionaron para los mismos las líneas de requerimientos que tenían un HIT.

Luego, el proceso Shell, será el que se encargará de actualizar los logs de los servidores correspondientes a dichos archivos y borrarlos del directorio *ac*.

Modelo 2:

La idea de éste modelo es utilizar un analizador de tráfico de red para identificar los requerimientos que llegan al servidor proxy en lugar de leer directamente del archivo de log como se definió en el modelo anterior.

La información recolectada por el analizador, debe pasar antes por varios filtros o procesos, hasta llegar a formar un archivo con las actualizaciones, es decir con los requerimientos que resuelve el proxy sin necesidad de acceder al servidor real.

Al igual que en el modelo anterior (modelo 1), se generan tantos archivos finales, cómo servidores haya involucrados en los requerimientos recolectados.

La herramienta a la cuál se hace referencia y se utilizó para éste modelo, es el popular *tcpdump*.

Para éste caso en particular, se ejecuta el tcpdump con seteos en dónde sólo pasen paquetes direccionados al port 80 (port estándar del HTTP).

Cada línea de entrada se convierte de hexadecimal a ASCII, y sólo se tienen en cuenta los paquetes que tengan el header HTTP, para lo cuál se buscará en los caracteres iniciales del campo de datos de los mismos, si existe alguno de los strings: GET, HEAD, HTTP y POST, en cuyo caso serán seleccionados.

Así, con los paquetes seleccionados se arma un archivo de actualización con igual formato que el archivo de log del servidor en cuestión, para lo cuál debe existir un formato preestablecido.

Una vez generados los archivos correspondientes a cada servidor mencionado, se corre un proceso shell, que envía y actualiza el log real vía *ftp*, para lo cuál se debe convenir en la definición de un usuario para dicho fin.

Al igual que en el modelo 1, los archivos de actualización se crean en un subdirectorio llamado ***ac***, y llevan como nombre el DNS ó dirección IP, que se especifica en el requerimiento.

Así el proceso shell que se utiliza es el del modelo 1, el cuál lee los archivos que se generaron en el directorio ***ac***, y va adicionando cada uno al servidor correspondiente.

Ejemplo de una ejecución del tcpdump:

El siguiente es un ejemplo de una corrida del tcpdump:

```
10:42:22.459546 0:60:97:21:2f:1e > 0:0:1b:28:1c:5c sap e0 ui/C len=50
                                ffff 0032 0011 a100 0000 0000 0000 0001
                                0451 0000 0026 0060 9721 2f1e 4007 2222
                                0110 0100 4800 0000 4889 0100 0000 0000
                                0010 00
10:42:22.459546 0:0:1b:28:1c:5c > 0:60:97:21:2f:1e sap e0 ui/C len=56
                                ffff 0038 0011 0000 0026 0060 9721 2f1e
                                4007 a100 0000 0000 0000 0001 0451 3333
                                0110 0100 0000 0010 0004 0000 0000 0000
                                0000 3a
10:42:22.459546 0:60:97:21:2f:1e > 0:0:1b:28:1c:5c sap e0 ui/C len=50
                                ffff 0032 0011 a100 0000 0000 0000 0001
                                0451 0000 0026 0060 9721 2f1e 4007 2222
                                0210 0100 4800 0000 4887 0000 000f 53f9
                                024c 3a
10:42:22.459546 0:0:1b:28:1c:5c > 0:60:97:21:2f:1e sap e0 ui/C len=378
                                ffff 017a 0011 0000 0026 0060 9721 2f1e
                                4007 a100 0000 0000 0000 0001 0451 3333
                                0210 0100 0000 0151 0120 3135 5052 4730
                                3032 53
10:42:22.539546 0:60:97:21:2f:1e > 0:0:1b:28:1c:5c sap e0 ui/C len=50
                                ffff 0032 0011 a100 0000 0000 0000 0001
                                0451 0000 0026 0060 9721 2f1e 4007 2222
                                0310 0100 4800 0000 f942 0100 000c 51c2
                                021c 53
10:42:22.539546 0:0:1b:28:1c:5c > 0:60:97:21:2f:1e sap e0 ui/C len=176
                                ffff 00b0 0011 0000 0026 0060 9721 2f1e
                                4007 a100 0000 0000 0000 0001 0451 3333
                                0310 0100 0000 0088 2020 2036 3032 3731
                                3939 38
10:42:22.559546 0:60:97:21:2f:1e > 0:0:1b:28:1c:5c sap e0 ui/C len=50
                                ffff 0032 0011 a100 0000 0000 0000 0001
                                0451 0000 0026 0060 9721 2f1e 4007 2222
                                0410 0100 4800 0000 50c0 0000 0000 0000
                                0010 38
10:42:22.559546 0:0:1b:28:1c:5c > 0:60:97:21:2f:1e sap e0 ui/C len=56
                                ffff 0038 0011 0000 0026 0060 9721 2f1e
```

4007 a100 0000 0000 0000 0001 0451 3333
0410 0100 0000 0010 0004 0000 0000 0000
0000 11

10:42:22.629546 0:a0:24:f2:94:61 > 0:60:b0:1b:db:81 sap e0 ui/C len=48
ffff 0030 0111 33b3 cf73 0000 0000 0001
0451 0000 0027 0000 0108 8506 4003 2222
9b02 0200 3e02 0953 5545 5c43 5c42 4154
0000 11

10:42:22.629546 0:60:b0:1b:db:81 > 0:a0:24:f2:94:61 sap e0 ui/C len=44
ffff 002c 0011 0000 0027 0000 0108 8506
4003 33b3 cf73 0000 0000 0001 0451 3333
9b02 0100 0000 0000 86ff ffff 5c42 4154
0000 11

10:42:22.629546 0:a0:24:f2:94:61 > 0:60:b0:1b:db:81 sap e0 ui/C len=54
ffff 0036 0111 33b3 cf73 0000 0000 0001
0451 0000 0027 0000 0108 8506 4003 2222
9c02 0200 3f00 0086 ffff 060a 4e4f 5041
3430 ae

10:42:22.629546 0:60:b0:1b:db:81 > 0:a0:24:f2:94:61 sap e0 ui/C len=70
ffff 0046 0011 0000 0027 0000 0108 8506
4003 33b3 cf73 0000 0000 0001 0451 3333
9c02 0100 0000 0100 0000 4e4f 5041 3430
2e53 4c

10:42:22.629546 0:a0:24:f2:94:61 > 0:60:b0:1b:db:81 sap e0 ui/C len=48
ffff 0030 0111 33b3 cf73 0000 0000 0001
0451 0000 0027 0000 0108 8506 4003 2222
9d02 0200 3e02 0953 5545 5c43 5c42 4154
2e53 4c

10:42:22.629546 0:60:b0:1b:db:81 > 0:a0:24:f2:94:61 sap e0 ui/C len=44
ffff 002c 0011 0000 0027 0000 0108 8506
4003 33b3 cf73 0000 0000 0001 0451 3333
9d02 0100 0000 0000 87ff ffff 5c42 4154
2e53 4c

10:42:22.629546 0:a0:24:f2:94:61 > 0:60:b0:1b:db:81 sap e0 ui/C len=54
ffff 0036 0111 33b3 cf73 0000 0000 0001
0451 0000 0027 0000 0108 8506 4003 2222
9e02 0200 3f00 0087 ffff 060a 4e4f 5041
3430 ae

Resumen de Procesos que se ejecutan en el Servidor Proxy:

- Se invoca al *tcpdump* para capturar todos los paquetes destinados al port 80 sobre cualquier dirección IP.

- Se convierte el contenido del paquete de hexadecimal a ASCII

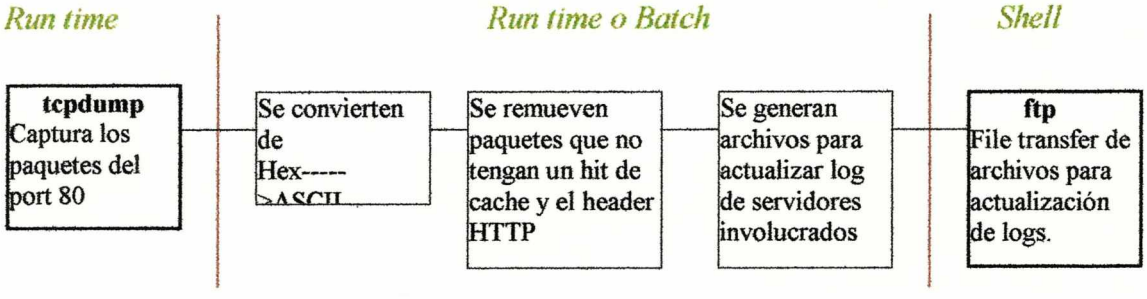
- Se seleccionan los paquetes, considerando sólo los que tienen el header HTTP (GET, HEAD, HTTP y POST) y un hit de cache.

- Se arma un archivo de log con las actualizaciones, es decir con los paquetes que sólo llegaron al proxy.

- El proceso shell, se encarga de leer los archivos de actualización y enviarlos vía *ftp* al servidor correspondiente.

Gráfico de Procesos:

Gráficamente se vería de la siguiente manera:



Limitaciones del modelo 2:

Una limitación es que el tcpdump monitorea tráfico desde un port específico del servidor, así se debe conocer a priori que número de port sobre la red está corriendo el servidor de Web.

Por lo tanto son varios los ports que se deben monitorear, pero a mayor número de ports, hay que tener en cuenta que se incrementa el volumen total de tráfico de red a filtrar.

Además puede ser que haya servidores de Web sobre ports no especificados, y así no se tendrían en cuenta, perdiendo de ésta manera información.

Una limitación importante es que el *tcpdump* requiere poner la interface de red en modo promiscuo, lo cuál es altamente no recomendable desde el punto de vista de la seguridad.

Otra cuestión a considerar es la performance, ya que la computadora sobre la cuál se corra el analizador de tráfico con los procesos de filtrado y conversión de información, debe tener un procesador rápido, suficiente memoria y espacio en disco, para que dicha ejecución no genere un cuello de botella y degrade la performance de la red. Y si estamos ante una jerarquía de proxies, la carga de la red sería muy importante, ya que el analizador correría en cada servidor proxy.

En éste modelo se analiza cada entrada del *tcpdump*, dejando de lado el archivo de log del proxy, y si por alguna razón se interrumpe su corrida, se perdería también información.

Se puede ver entonces, que es mejor trabajar sin ninguna herramienta adicional para leer los accesos, ya que leyendo del archivo de log sería más seguro.

Todo requerimiento que alcance al proxy, estará siempre en su log de accesos, y

no se precisa más que su lectura para identificar los requerimientos que resuelve el mismo, ésto es suficiente para armar el archivo con la información que no se registró en el servidor en cuestión.

Así, debido al estudio de las limitaciones encontradas en el *modelo 2*, se puede apreciar que éste no optimiza al *modelo 1* y que con el mismo se estarían adicionando cargas innecesarias, tanto a la red cómo al ó los servidores proxy, por lo que se adopta el ***modelo 1*** cómo solución a la problemática de ***sincronización de los logs de servidores y proxies***.

VIII.Conclusión:

La información que se va registrando on-line, en los archivos de logs, da un panorama casi perfecto del comportamiento de los usuarios en un sitio Web.

Mediante ésta información y configurada de la manera más conveniente, es posible realizar un monitoreo de la utilización del sitio aplicable a diversos fines, para lo cuál existen actualmente varios analizadores de logs de Web que lo permiten.

La dificultad se encuentra cuando los datos registrados están lejos de la realidad, ya que se distorsionan los resultados.

Así, cuando estamos ante la presencia de servidores proxy cache, los archivos de logs se tornan inclompletos, encontrándose parte de los requerimientos en el log del proxy.

De aquí la necesidad de lograr una sincronización de dichos logs, con lo que se aportaría a la optimización de las estadísticas de un sitio.

Con la definición del ***modelo 1*** antes descripto, se ha llegado a complementar la información que se registra en el log del servidor real a través de la información que se archiva en el log del proxy, y ésto mediante procesos sincronizados que se disparan automáticamente y sin consumo de rendimiento.

IX. Bibliografía / Referencias:

[1] **Dean Povey, John Harrison, "A Distributed Internet Cache", School of InformationTechnology, University of Queensland, Brisbane, QLD, 4072 {povey,harrison}@cs.uq.edu.au.**

Este paper, presenta al caching cómo solución al problema de optimización del uso de red y utilización eficiente de los servidores.

Definición de estructuras de caching existentes, con ventajas y desventajas para cada caso.

[2] **Computer Networks, 3ra Edición, Andrew S. Tanenbaum**

Definición de proxy server, y gráficos de involucran un browser que habla *ftp* y uno que usa un proxy.

[3] **Building Internet Firewalls, O'Reilly**

Capítulo 7 - Proxy Systems, ventajas y desventajas del proxeado, cómo trabajar con proxeado, SOCKS, proxeado y servicios de Internet.

[4] **Luotonen,A., K.Altis, "World-Wide Web Proxies", Proc. 1st Inter.Conf. on the WWW, Geneva, May 1994.**
http://www1/cern.ch/PapersWWW94/luotonen.ps.

Este paper, da un panorama de los proxies y su estado actual.

Esquemas de proxy server corriendo en un host firewall proveyendo acceso

completo a Internet, ó en una máquina dentro de un firewall haciendo conexiones a través de SOCKS u otro soft de firewall.

[5] **M.Abrams, C.Standrifge, G.Abdulla, S.Williams, E. A. Fox, "Caching Proxies: Limitations and Potentials", Proc, 4th International World Wide Web Conference, Boston, MA, December 1995.**
http://ei.cs.vt.edu/~succeed/WWW4/WWW4.html.

Limitaciones y potencialidades de los caching proxies, según un estudio estadístico de su uso.

[6] **Marc Abrams, Stephen Williams, "Complementing Surveying and Demographics with Automated Network Monitoring", Departament of Computer Science, Virginia Tech Blacksburg, VA 24061-0106**
http://www.cs.vt.edu/~chitra/docs/96w3j/96w3j.htm.

Se presentan dos maneras complementarias para aprender acerca de quiénes usan la Web y cómo la usan, esto es mediante encuestas a los usuarios o por medio de los archivos de logs.

Se describen los pro y los contras de los 4 métodos de logings: basado en el servidor, basado en el proxy, basado en el cliente y basado en la red. Y se presentan 2 herramientas para generar logs de red.

[7] **Phillip M. Hallam-Baker <hallam@w3.org>, Brian Behelendorf <brian@organic.com>, "Extended Log File Format", W3C Working Draft WD-logfile-960323, March 1996.**
http://www.w3.org/TR/WD-logfile.html

Se presenta un formato mejorado de archivo de logs para servidores de Web. El formato es extensible y permite capturar un amplio rango de datos. Esto está motivado

por la necesidad de capturar datos para análisis demográfico y por las necesidades de proxy caches.

[8] **Help On-Line de Sun**

Capítulo de Monitoreo del servidor y Trabajo con archivos de logs.

Se detallan los códigos de estado que devuelve el servidor al cliente y los formatos de logs files disponibles.

Muestra cómo se configuran las preferencias para los logs y muestra la sintaxis en cada uno de los formatos: Common, Extended y Extended-2.-

[9] **El Lenguaje de Programación C, Brian W.Kernighan / Dennis M.Ritchie.**

Para el desarrollo del proceso de lectura del archivo de log del proxy descripto en el modelo 1, se tomó cómo referencia éste libro de programación en lenguaje C.

[10] **WebAlyser - Website Log file Analisys Software.**
http://www.webalyser.com

Analizador de logs de Web. Ejemplos de gráficos y tablas que permite construir.

[11] **Boutell, T. Wusage.**
http://www.boutell.com/usage/

Analizador de logs de Web.

[12] **Fielding, R. Wwwstat.**
http://www.ics.uci.edu/WebSoft/wwwstat/

Analizador de logs de Web. Ejemplos de tablas y gráficos que permite construir.

[13] **Long, q. Gwstat.**
http://dis.cs.umass.edu/stats/gwsat.html

Analizador de logs de Web.

[14] **StJohns, M., Authentication Server, RFC 931, Jan 1985.**
ftp://nic.ddn.mil/rfc/rfc931.txt

Protocolo de Autenticación para la comunidad Internet.

[15] **World Wide Web Consortium, The Common Log Format.**
http://www.w3.org/pub/WWW/Daemon/User/Config/Logging.html#common-logfile-format

Formato Común para archivos de logs.

[16] **Williams, S. M.Abrams, C.Standridge, g.Abdulla, E.A.Fox, "Removal Policies in Network Caches for World wide Web Documents", Stanford CA, August 1996.**
http://ei.cs.vt.edu/~suceed/96sigcomm.html.

Este paper estudia la performance de un cache considerando las políticas de borrado de documentos del mismo, y utilizando para la simulación medidas del *hit rate (HR)*, ó fracción de requerimientos satisfechos por el proxy y del *weighted hit rate (WHR)*, ó número de bytes no enviados por el servidor.

[17] **Cunha,C.R., A.Bestavros, M.E.Crovella, "Characteristics of WWW Client Based Traces, TR:BU-CS-95-010, Boston University, July 1995.**
ftp://cs-ftp.bu.edu/techreports/95-010-www-client-traces.tar.gz.

En éste paper se describen métodos usados para recolectar requerimientos de usuarios basados en el cliente. Esto permite evaluar la performace de uso de la Web a nivel usuario y a lograr una figura exacta del tráfico de la red, aportando así a evaluar técnicas de reducción de ancho de banda.

[18] **Lawrence Berkeley Laboratory Network Research Gruop, tcpdump.**
ftp://ftp.ee.lbl.gov/tcpdump.tar.Z.

Herramienta de monitoreo de red.

[19] **Analog 2.11**
http://www.dit.upm.es/analog/analog.html

Programa que analiza logfiles de servidores de Web, y permite realizar estadísticas y reportes atractivos.

[20] **The Common Gateway Interface**
http://hoohoo.ncsa.uiuc.edu/docs/cgi/intro.html

Este paper da un panorama del uso de la interface genérica CGI, entre el servidor y el programa gateway.

[21] **Writing secure CGI scripts**
http://hoohoo.ncsa.uiuc.edu/docs/cgi/secutiry.html

Ejemplos de scripts CGI.

[22] **An Introduction to The Common Gateway Interface**
http://www.utoronto.ca/webdocs/CGI/cgi1.html

Especificaciones de la interface CGI.

[23] Squid Features - Release Notes for version 1.0 of the Squid cache
http://squid.nlanr.net/

Características y mejoras del Squid cache. Formato de log file y contenido del mismo.

X. Apéndice I:

El presente diskette cuenta con los programas para la utilización del ***modelo 1***:

- ***log1.c :*** programa base para lectura de logs y generación de archivos de actualización.
- ***squid.c:*** programa utilizado para la corrida sobre un proxy *Squid*.
- ***actu:*** proceso shell, que complementa la actualización de logs, haciendo el file transfer de los archivos generados por el proceso de filtrado del log del proxy.