Una Alternativa Económica para la Implementación de Servicios de Web Localmente Distribuidos

Sergio A. Davicino Javier Echaiz*† Jorge R. Ardenghi†

Laboratorio de Investigación de Sistemas Distribuidos (LISiDi)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
T.E.: +54 291–4595135 Fax: +54 291–4595136
e-mail: {sad,je,jra}@cs.uns.edu.ar

Resumen

El rápido crecimiento de Internet, tanto en contenidos como en cantidad de usuarios, alienta la creación de nuevos mecanismos para mejorar los servicios brindados.
En este trabajo se persigue el objetivo de mejorar los servicios de Web teniendo como premisa básica la compatibilidad con los estándares y protocolos de Web existentes, i.e., sin la necesidad de modificaciones en los protocolos de Internet, estándares de Web o código fuente de las aplicaciones cliente.
Nuestra propuesta de implementación utiliza iptables como el aglutinante que permite unir políticas y mecanismos para llevar a cabo la distribución de carga. Este paper identifica iptables como una herramienta adicional en un campo diferente a su habitual empleo en la implementación de firewalls.

Términos Clave: servicios de Web distribuidos, balance de carga, iptables.

1 Introducción

La World Wide Web (WWW) puede ser considerada como un gran sistema de información distribuida que brinda acceso a distintos objetos de datos. El gran crecimiento experimentado por la Web debido a su creciente popularidad enfrenta a sus usuarios con dos de los problemas que sufre, congestión de la red y sobrecarga en los servidores. Estos dos problemas con los cuales la Web se enfrenta se manifiestan a los usuarios como grandes retardos en las respuestas a sus pedidos de información.

Dada la complejidad de su infraestructura, pueden surgir problemas de performance en distintos puntos de una transacción Web que hacen dificultosa la labor de identificar en forma precisa la causa del retardo. Existen estudios como el llevado a cabo en [BC00] en donde se propone un método, basado en técnicas de camino crítico, para detectar la fuente de retardos en aplicaciones de Internet. Este método permite distinguir con cierto grado de precisión qué retardos se deben al servidor y qué retardos se atribuyen a la red. El inconveniente con el mismo es que no se puede determinar para cada uno de los retardo cuales son sus orígenes.

*Becario de la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires, Argentina.
†Miembro del Instituto de Investigación en Ciencia y Tecnología Informática.

Por ejemplo, si el retardo debido a la red es a causa de los routers, backbones u otros servicios dependientes, e.g. el Domain Name System (DNS).

En los últimos tiempos han surgido distintos métodos para el análisis y mejora de las transacciones Web que caen en dos categorías: la referente a los servidores y la referente a las redes y protocolos. Con respecto a las redes, las mejoras se centran en la performance de la infraestructura de la red para las aplicaciones de Internet. Por parte de los protocolos es fácil concluir que los estudios y desarrollos propiciaron una variedad de mejoras en los mismos. Por ejemplo, en el Hypertext Transfer Protocol (HTTP) se han incluido características como: compresión de datos, conexiones persistentes y pipelining de operaciones, que forman parte de la especificación del HTTP/1.1 [FGM+97]. En lo que respecta a los servidores son varias las propuestas para mejorar su performance. Una de ellas la representan los esquemas de *caching* que almacenan objetos de datos populares (usados frecuentemente) en locaciones cercanas al cliente. El uso de servidores *proxy* [LA94], que surge en un primer momento como un método para que usuarios ubicados detrás de un firewall puedan acceder a Internet, resulta en la actualidad una solución efectiva para aliviar el embotellamiento en la Web, reduciendo el tráfico en Internet [Wan99].

Otros esquemas de caching constituyen los llamados *Web Server Accelerators*. El esquema subyacente en los web server accelerators se basa en el caching de datos en el sitio hosting de forma tal que las páginas accedidas frecuentemente sean entregadas desde la cache, evitando la sobrecarga tanto del servidor (remoto) como de la red. En [SLAID00] se presentan varias alternativas de diseño para Web server accelerators basadas en la utilización de un *array* de servidores cache y un router que puede ubicarse en la capa 4 ó 7 del stack del modelo ISO/OSI.

Una buena alternativa para mitigar el problema de la sobrecarga y brindar un servicio escalable de Web, es la representada por el empleo de arquitecturas de servicio de Web distribuido compuesto por múltiples nodos servidores, en donde algún componente distribuye los pedidos que arriban entre los distintos servidores. Esta distribución de los pedidos permite llevar a cabo lo que en la jerga de sistemas distribuidos se conoce como *distribución de carga*.

Los nodos servidores pueden hallarse distribuidos geográficamente o pueden residir en una única locación (*geográficamente distribuido* o *localmente distribuido*). Los *mirrored-servers* constituyen un ejemplo muy conocido de servicio de Web distribuido geográficamente, en donde los servidores poseen replicas del sitio y es el usuario quien manualmente selecciona el nombre del servidor Web que va a atender su pedido. En los esquemas en los que el usuario no selecciona el servidor que atiende su pedido surgen varias alternativas en cuanto a la selección del lugar donde se toma la decisión. Éstas pueden ser:

- A *nivel del cliente*, la entidad responsable de seleccionar el servidor es una aplicación cliente, generalmente un *browser* de Internet.
- A *nivel de DNS*, durante la fase de resolución de nombre, la entidad que se encarga de la distribución de pedidos es el *authoritative DNS server* para el sitio en cuestión.
- A *nivel de red*, el pedido del usuario puede ser dirigido por un router al servidor que corresponda.
- A *nivel del sistema de Web*, la entidad que se encarga de la distribución puede ser alguno de los servidores del sistema o algún dispositivo que realiza la distribución de pedidos y que es puesto como portal de la arquitectura del sitio Web.

En este trabajo proponemos una alternativa económica para implementar un sistema de servicio de Web localmente distribuido, en donde la distribución de pedidos se realiza a nivel

de red. Para ello utilizamos los medios disponibles en nuestro entorno y nos basamos principalmente en las facilidades que nos otorgan Netfilter e iptables. Nuestro entorno está constituido por varias redes privadas divididas por routers que utilizan iptables para implementar los distintos niveles de filtrado de paquetes. Lo interesante de este entorno es que actualmente se encuentra muy difundido, tanto en ambientes académicos como industriales.

En la siguiente sección se analizan las arquitecturas básicas de los sistemas de Web distribuidos. En la sección 3 se describen los mecanismos de ruteo aplicables a estos sistemas y se proponen dos alternativas a implementar con iptables. Luego, la sección 4 analiza las políticas (mecanismos) básicas de balance de carga y se analizan las facilidades que brinda nuestra herramienta al respecto. A continuación se presenta detalladamente el esquema de distribución de carga utilizando DNS round-robin, se analizan sus deficiencias y se propone una implementación alternativa basada en clusters de Web. Por último, en la sección 6, se resumen las conclusiones y futuras direcciones.

2 Descripción de la Arquitectura

La premisa básica en la que se basa la arquitectura propuesta en este trabajo es la siguiente: las soluciones propuestas son compatibles con los estándares de Web existentes y los protocolos, i.e., no se requieren modificaciones en los protocolos de Internet, estándares de Web o código del cliente.

De acuerdo a la taxonomía presentada en [CC01] nuestra arquitectura es clasificada como *Cluster-based Web System*. La principal diferencia con lo que se denomina *Distributed Web System*, es que en este último las direcciones IP de los nodos servidores que componen la arquitectura son visibles al cliente mientras que en un esquema de cluster-based Web system la dirección IP de los servidores es enmascarada.

En la sección 5 se presenta un esquema de distributed Web system, basado en *DNS-round-robin (DNS-RR)*, se analizan los inconvenientes que éste presenta y se describe una posible solución utilizando iptables para convertirlo en un cluster-based system eliminando dichos inconvenientes.

Un cluster-based Web system es una colección de nodos servidores que se encuentran situados juntos en un único sitio geográfico y se encuentran interconectados mediante una LAN. Además presentan al exterior una imagen simple del sistema ([Ech02]), i.e., los usuarios interactúan con el Web-server system como si este fuese un único servidor con alta performance. Cada nodo servidor puede responder cualquier pedido de un cliente dado que tiene una réplica del sitio Web o utiliza algún mecanismo de file sharing, como por ejemplo Network File System (NFS) [CPS95].

Aunque el cluster consiste de varios nodos servidores, el mismo es publicado con un único nombre (por ejemplo, www.sitio.edu.) y una única dirección IP (por ejemplo, 200.49.224.5) denominada dirección IP virtual (VIP). Por lo tanto, el authoritative DNS server para el sitio siempre realiza un mapeo uno a uno para trasladar el nombre del sitio a la dirección VIP, que es la dirección del nodo portal denominado *front-end*. Este nodo es la interface entre el cluster e Internet y por este motivo hace a la naturaleza distribuida de la arquitectura del sitio, transparente a ambos, usuarios y aplicaciones cliente. El nodo front-end, también llamado *Web switch*, recibe todos los paquetes que el cliente envía a la dirección VIP y los distribuye entre los nodos servidores. Un esquema general del Web cluster que será utilizada en próximos ejemplos es presentado en la figura 1.

El Web switch puede ser implementado con una máquina con una o dos NICs (interfaces

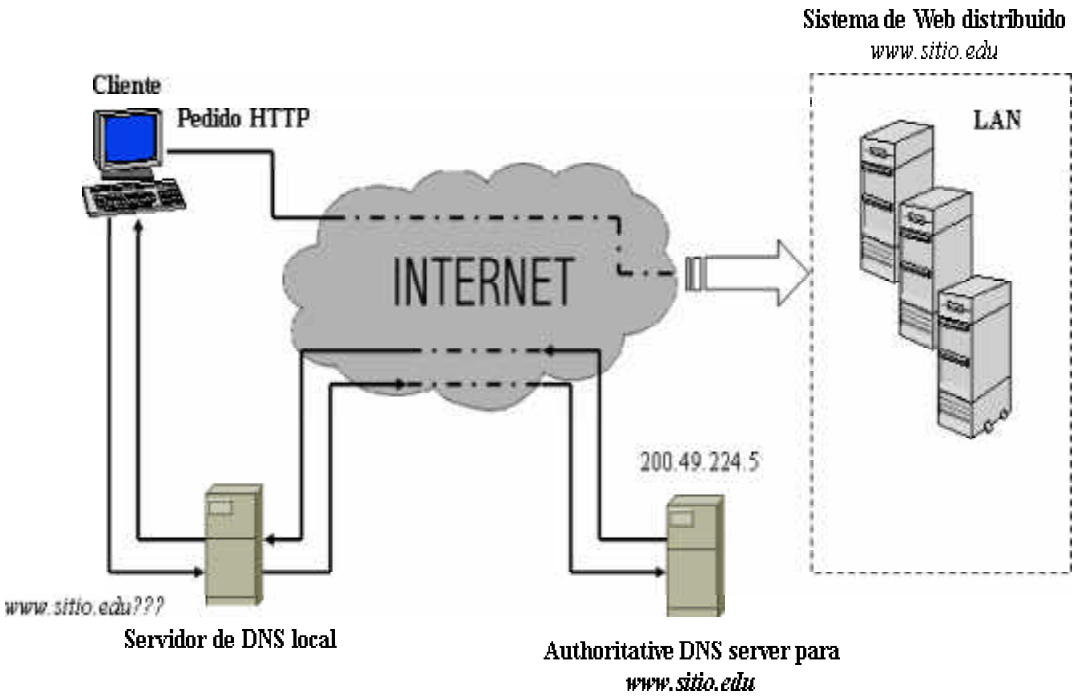

Figura 1: Esquema general del Web cluster.

de red) y la elección dependerá de la configuración que se utilice para el tráfico entrante y saliente del cluster. El sistema operativo a ejecutar en esta máquina puede ser cualquiera de las distribuciones de Linux (Red Hat, Slackware, Suse, Caldera, etc.) basadas en el kernel 2.4.x[1], el cual se debe configurar con el paquete iptables. Iptables es una herramienta embebida en el kernel de Linux que permite realizar principalmente filtrado de paquetes a varios niveles del modelo ISO/OSI y además brinda facilidades para llevar a cabo network address translation (NAT), en su dos formas: *destination address translation* (DNAT) y *source address translation* (SNAT) [And01].

3 Mecanismos de Ruteo

El Web switch es capaz de identificar unívocamente al resto de las máquinas que componen el cluster a través de su dirección IP. En nuestro ejemplo, a cada nodo se le asigna una dirección IP que dependiendo de la configuración del cluster puede ser una dirección IP correspondiente a las clases reservadas para redes privadas, o puede ser una dirección IP pública ([Pos81]).

Existen varias técnicas de ruteo a explorar en el ámbito de Web switch. Un buen mecanismo para clasificar las distintas alternativas es agruparlas de acuerdo a la capa o nivel dentro del protocolo stack del modelo ISO/OSI en que el switch toma las decisiones de ruteo. Las alternativas son las siguientes:

- En la *capa 4*: a este nivel el switch realiza lo que se denomina en [CC01] *content-blind routing*, debido a que el switch determina el servidor destino en el momento en el que el cliente trata de establecer una conexión TCP/IP, i.e., cuando arriba el primer paquete `TCP SYN` al switch.

[1]El kernel de linux se encuentra disponible en forma gratuita en `ftp://ftp.kernel.org`.

- En la *capa 7*: a este nivel el switch realiza lo que se conoce como *content-aware routing*. El switch establece una conexión completa con el cliente, examina el pedido de HTTP a nivel de aplicación y luego lo envía al servidor seleccionado para su procesamiento.

La selección del mecanismo de ruteo incide profundamente en las políticas de distribución de carga debido a que la clase de información disponible en el switch varía de acuerdo a la capa en que se esté trabajando. En la capa 4 los paquetes no alcanzan en el switch el nivel de aplicación, por lo tanto el mecanismo de ruteo es más eficiente. Sin embargo realizar el ruteo a nivel de aplicación permite que el switch implemente políticas de distribución de carga con una granularidad más fina, dado que posee mayor información acerca del pedido del cliente.

En nuestra propuesta las distintas políticas de distribución de carga y el mecanismo de ruteo se implementan a nivel de TCP/IP (capa 4). Esta decisión tiene su fundamento en el hecho que, implementar la distribución de carga a nivel de aplicación impone una mayor sobrecarga en la labor del switch, atentando contra la escalabilidad del sistema dado que el switch se vuelve un cuello de botella. Además requiere un mayor desarrollo en las aplicaciones y nuestro objetivo en esta instancia es lograr una arquitectura de Web distribuida utilizando los elementos disponibles en el entorno con un mínimo de modificación en las aplicaciones.

Los clusters de Web basados en la capa 4 pueden ser clasificados en base al mecanismo utilizado para rutear los paquetes entrantes al servidor y los paquetes salientes al cliente. La principal diferencia reside en la forma en que los paquetes retornan desde el servidor hacia el cliente. En este sentido podemos distinguir dos arquitecturas:

- arquitecturas two-way: tanto los paquetes entrantes como los salientes pasan por el Web switch.
- arquitecturas one-way: solo los paquetes entrantes pasan por el Web switch; los paquetes salientes se transmiten directamente desde el servidor que atiende el pedido hasta el cliente.

3.1 Implementación de los Mecanismos de Ruteo y Detalles de la Arquitectura

3.1.1 Arquitectura Two-way

Como alternativa para esta arquitectura y por el hecho de que tanto los paquetes entrantes como salientes pasan por el Web switch, decidimos una configuración en la que los servidores del cluster se encuentran en una red privada.

Esto requiere que el Web switch posea dos interfaces de red, una interface externa configurada con la dirección VIP y una interface interna configurada con una dirección IP correspondiente a alguna de las clases reservadas. Para el ejemplo utilizamos la clase C cuyo numero de red es 192.168.0.0/24, para asignar direcciones IP tanto a los servidores, como a la interface interna del Web switch, según se observa en la figura 2.

El mecanismo de ruteo es implementado utilizando iptables de la siguiente forma: cuando un paquete que proviene desde un cliente tratando de conectarse al sitio llega al Web switch, su dirección IP destino es modificada con la dirección del server seleccionado para dar atención al pedido. Esto se lleva a cabo agregando al kernel reglas que permitan realizar DNAT.

Cuando el servidor recibe el paquete contesta directamente al cliente pero lo hace a través del Web switch. La respuesta que proviene del servidor tiene como dirección fuente la dirección IP del mismo (una dirección IP privada y no ruteable), por lo tanto, se debe hacer SNAT con la dirección VIP. Esto es, se modifica la dirección fuente del paquete con la dirección VIP y se

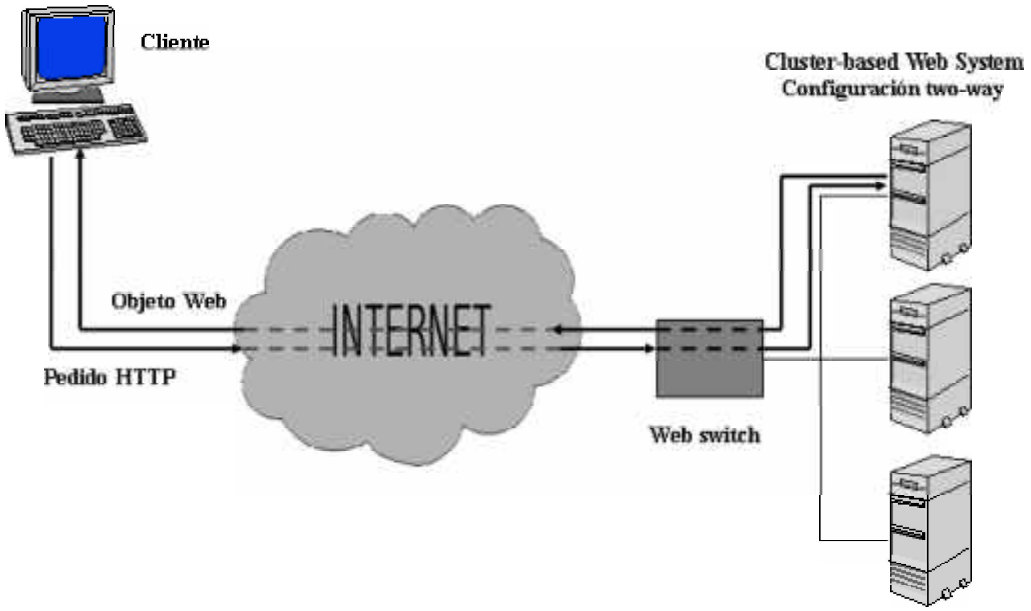

Figura 2: Arquitectura Two-way.

envía el paquete al cliente. El cliente ve a este paquete como si hubiese sido generado por el Web switch.

En el siguiente ejemplo se detallan conjuntos de reglas que permitirían implementar DNAT y SNAT en un esquema sin distribución de carga. Se asume que la dirección IP de la interface externa (`eth0`) del Web switch es la dirección VIP (200.49.224.5), que la dirección IP del servidor es 192.168.0.1 y que el servidor tiene configurado como default router al Web switch:

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 200.49.224.5

iptables -t nat -A PREROUTING -p tcp -d 200.49.224.5 --dport 80 -j DNAT --to-destination 192.168.0.1
```

3.1.2 Arquitectura One-way

El principal beneficio de este esquema es que los servidores envían sus replicas directamente a los clientes, resultando en una disminución del tráfico que debe atravesar el Web switch. Para esta arquitectura se decidió no utilizar una red privada dado que se perderían los beneficios del esquema al tener que realizar SNAT de los paquetes salientes.

En el ejemplo que se muestra en la figura 3 cada uno de los servidores tiene asignada una dirección pública y al igual que en el Web switch es condición necesaria tener instalado Linux con soporte de iptables en el kernel. El Web switch tiene una única interface configurada con la dirección VIP.

El mecanismo de ruteo se implementa de la siguiente manera: cuando un paquete que proviene desde un cliente tratando de conectarse al sitio llega al Web switch, su dirección IP destino es modificada con la dirección del server seleccionado para dar atención al pedido y el paquete es enviado al mismo.

El servidor envía la réplica directamente al cliente, pero a diferencia del esquema anterior,

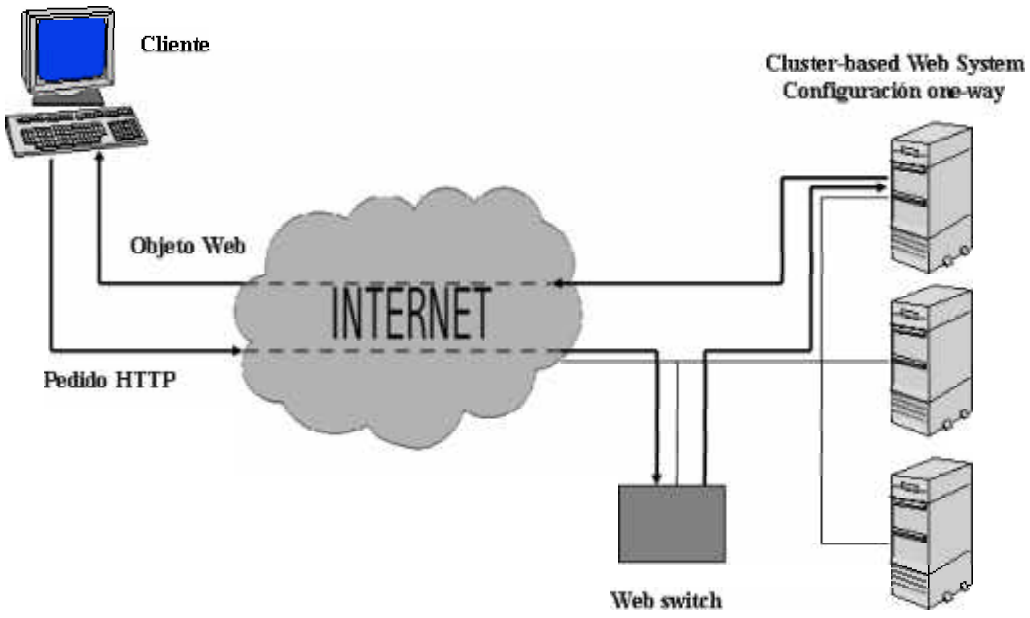

Figura 3: Arquitectura One-way.

éste debe modificar la dirección fuente, dado que el cliente está esperando una respuesta del Web switch. Para ello realiza SNAT con la dirección VIP antes de enviar el paquete.

En el siguiente ejemplo se dan las reglas necesarias que se necesitan sumar al kernel del Web switch y al kernel del servidor para implementar un esquema sin distribución de carga. Se asume que la dirección IP del servidor encargado de procesar el pedido es 200.49.224.6.

- En el Web switch
  iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 200.49.224.6
- En el servidor
  iptables -t nat -A POSTROUTING -p tcp --sport 80 -j SNAT --to-source 200.49.224.5

4 Mecanismos de Distribución de Carga

Luego de analizar los mecanismos para el ruteo de pedidos, en esta sección describimos las políticas que pueden ser implementadas para distribuir los pedidos entre los servidores del cluster. Como se puede observar en la taxonomía dada en [EDA03], existen varias alternativas a la hora de elegir el esquema para la política de distribución de carga, pero solo un subconjunto de estas pueden ser aplicadas en un cluster de Web. Este tipo de arquitectura con un único Web switch que recibe todos los pedidos conduce la elección a un algoritmo de distribución de carga centralizado. Luego la alternativa real esta en decidir entre algoritmos estáticos o dinámicos. Sin embargo la selección entre los algoritmos dinámicos se ve acotada por el hecho de que el Web switch no puede utilizar algoritmos muy sofisticados puesto que este debe tomar decisiones rápidas en lo que concierne al ruteo.

A continuación analizaremos los mecanismos que brinda iptables para llevar a cabo distribución de carga, los cuales solo permiten implementar políticas de distribución estáticas. Además se analizarán herramientas que permiten tratar con algunas cuestiones como heterogeneidad de los nodos que componen el cluster, calidad de servicio y tolerancia a fallos.

- *Round-Robin:* los pedidos se distribuyen entre los servidores que componen el cluster según el orden en el que arriban, asignando un pedido a cada uno. Por ejemplo, suponiendo que se tiene un poll de cuatro servidores cuyas direcciones IP van de 192.168.0.1 a 192.168.0.4, la política round-robin se implementa de la siguiente forma:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.0.1-192.168.0.4
```

  Si el rango de direcciones IP no es continuo se puede utilizar para el mismo fin el *nth patch* [Mar02]. Por ejemplo, suponiendo que se tiene un poll de tres servidores cuyas direcciones IP son: 200.49.224.1, 200.49.224.6 y 200.49.224.8. Las reglas para implementar esta política son:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -m nth --counter 7 --every 3 --packet 0 -j DNAT \
    --to-destination 200.49.224.1
iptables -t nat -A PREROUTING -p tcp --dport 80 -m nth --counter 7 --every 3 --packet 1 -j DNAT \
    --to-destination 200.49.224.6
iptables -t nat -A PREROUTING -p tcp --dport 80 -m nth --counter 7 --every 3 --packet 2 -j DNAT \
    --to-destination 200.49.224.8
```

- *Random:* los pedidos pueden ser distribuidos en forma random y uniforme entre los servidores. Por ejemplo, suponiendo que se tiene un poll de cuatro servidores cuyas direcciones IP son: 192.168.0.1, 192.168.0.15, 192.168.0.21 y 192.168.0.12. La política random se puede implementa con el *random patch* de la siguiente manera:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -m random --averange 25 -j DNAT \
    --to-destination 192.168.0.1
iptables -t nat -A PREROUTING -p tcp --dport 80 -m random --averange 33 -j DNAT \
    --to-destination 192.168.0.15
iptables -t nat -A PREROUTING -p tcp --dport 80 -m random --averange 50 -j DNAT \
    --to-destination 192.168.0.21
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.0.12
```

  Las reglas anteriores tiene un match en forma random de aproximadamente 1/4 de la cantidad total de paquetes cada una. Este mismo patch puede ser utilizado para tratar el problema de heterogeneidad de los nodos del cluster. Suponiendo que se tiene dos servidores con distinta configuración en la arquitectura y se desea que uno de ellos (el que tiene mejores recursos de hardware), atienda el 75% de los pedidos y que los restantes pedidos sean enviados al otro servidor. Si las direcciones IP son 200.49.224.3 y 200.49.224.8 respectivamente, las reglas quedarían de la siguiente manera:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -m random --averange 75 -j DNAT \
    --to-destination 200.49.224.3
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 200.49.224.8
```

Algunos estudios, como los presentados en [Voi02] y [VL00], tratan la sobrecarga de los servidores desde el punto de vista del servicio diferencial y la calidad de servicio (QoS). El

servicio diferencial trata básicamente de manejar de forma preferencial algunos pedidos, i.e., aumentando su prioridad si el servidor está sobrecargado. La calidad de servicio intenta brindar un servicio predecible al cliente independientemente de la demanda del mismo. Iptables brinda algunas facilidades para llevar a cabo QoS y servicio diferencial en forma rudimentaria, algunas de ellas son:

- *Iplimit patch:* permite restringir el número de conexiones TCP paralelas desde un host o red particular. Por ejemplo, si deseamos limitar el número de conexiones paralelas que se pueden hacer desde un mismo lugar a cuatro:

  ```
  iptables -A FORWARD -p tcp --dport 80 -m iplimit --iplimit-above 4 -j REJECT
  ```

- *Fuzzy patch:* permite controlar el match de una regla dados dos parámetros que dan el intervalo de filtrado. La idea es la siguiente:

  - Cuando el *rate* en que llegan los paquetes está por debajo del limite inferior, la regla nunca tiene un match.
  - Cuando el rate en que llegan los paquetes está entre el limite inferior y el limite superior, los match ocurren con un rate incremental.
  - Finalmente cuando el rate en que llegan los paquetes supera el valor límite, el rate en que se producen los match se mantiene a su máximo valor.

  Por ejemplo, si se quiere limitar el rate a 1000 paquetes por segundo y que a partir de los 100 paquetes por segundo se comiencen a descartar paquetes en forma gradual:

  ```
  iptables -A FORWARD -p tcp --dport 80 -m fuzzy --lower-limit 100 -upper-limit 1000 -j REJECT
  ```

Ambos ejemplos son bastante simples y sólo sirven a los fines de ilustrar cada una de las alternativas. Para aplicaciones reales se deben utilizar combinaciones de las distintas reglas, de manera de eliminar solamente los paquetes `TCP SYN` correspondientes a nuevas conexiones y no paquetes que pertenezcan a conexiones ya establecidas.

Por último, una cuestión importante a tratar es el problema de la tolerancia a fallas. Es decir, qué sucede cuando un nodo servidor queda fuera de servicio por algún tipo de falla. Sin una forma efectiva de anticipar esta situación, todos los pedidos que son dirigidos por el Web switch al nodo caído quedarían sin atención cuando en realidad podrían ser atendidos por los nodos servidores aún en funcionamiento. Una solución al problema de tolerancia a fallos utilizando iptables, puede implementarse con el patch *condition match*. Este patch permite inhabilitar reglas usando variables de condición almacenadas en el `proc` file system. La características de este match son:

- Las variables de condición son almacenadas en el directorio `/proc/net/ipt_condition/`.
- A una variable de condición puede asignársele el valor `0` (falso) o `1` (verdadero).
- El estado de una variable puede afectar a una o más reglas.
- Las variables de condición son creadas en forma automática cuando son referenciadas por primera vez en una regla.
- Una variable de condición es eliminada cuando se elimina la ultima regla que la referencia.

Por ejemplo, para brindar tolerancia a fallas se podría escribir un shell script que realice un chequeo de la disponibilidad de los servidores y que ante la eventualidad de la caída de alguno de ellos, asigne un valor a la variable de condición que corresponda para inhabilitar la regla que asigna pedidos a dicho servidor.

5 Distribución de carga utilizando DNS round-robin (DNS-RR)

Utilizar DNS para la distribución de carga fue una de las primeras propuestas para manejar múltiples servidores que representan un sitio Web. Originalmente fue concebido para servidores de Web localmente distribuidos, aunque ahora es utilizado comúnmente para sistemas de servicio de Web geográficamente distribuidos.

La distribución de los pedidos entre los distintos servidores que conforman el sistema se lleva a cabo durante la resolución de nombre en la transacción Web. Es decir, cuando el nombre del sitio Web debe ser mapeado a la dirección IP de alguno de los servidores que componen el sistema.

El mecanismo subyacente es bastante simple, el authoritative DNS server para el sitio retorna una lista con las distintas direcciones IP asociadas al mismo y el cliente selecciona la primera. Luego el servidor retorna para cada uno de los pedidos de resolución de nombre una lista de direcciones IP cuyo orden varía con cada pedido.

Cada mapeo entregado por el authoritative DNS server tiene asociado un valor *time to live* (TTL). Este valor determina cuánto tiempo puede ser almacenado el mapeo en la cache de los distintos servidores, entre el authoritative name server y el servidor de DNS del usuario.

El caching de mapeos *name-to-address* afecta a la distribución de carga dado que un pedido de resolución puede ser satisfecho por cualesquiera de los servidores que posea una copia fresca del mapeo y no por el authoritative name server, el encargado de implementar el mecanismo.

Una posible solución a este inconveniente es configurar al authoritative name server para que retorne el mapeo con un time-to-live muy bajo o nulo. Esto hace que el mapeo permanezca un corto tiempo en cache o directamente no sea almacenado (TTL=0). Los problemas que tiene esta solución son los siguientes: existen servidores que inicializan el TTL a un valor predeterminado cuando el valor asociado al mapeo es nulo o muy bajo. El valor de TTL que trata de imponer el authoritative name server no afecta a las copias del mapeo almacenadas en la cache del navegador. Por último, el principal problema es que inhabilitar el caching en los servidores non-authoritatives requiere que el cliente contacte al authoritative name server para cada resolución, incrementando la latencia en los accesos Web. Además esto puede degradar la escalabilidad del DNS debido a que todos los pedidos deben ser resueltos por el authoritative name server [STA01].

La solución que planteamos al esquema anterior es convertir el sistema Web distribuido en un cluster-based web system, con una arquitectura one-way en donde el Web switch implementa una política de distribución basada en round-robin. En este caso, el authoritative name server es configurado para contestar con la dirección VIP. Cabe aclarar que esta solución es aplicable sólo en los casos en que el sistema original no se encuentra geográficamente distribuido.

6 Conclusiones y Trabajo Futuro

En este trabajo analizamos distintos mecanismos y estrategias de distribución de carga aplicables a sistemas de Web localmente distribuidos basados en cluster. Este análisis nos permitió

comparar las herramientas actuales y plantear nuevas y económicas soluciones al problema.

En este trabajo planteamos por primera vez la aplicabilidad de las iptables en el campo de la distribución de carga en sitios Web, extendiendo su habitual uso en la construcción de firewalls. Además la utilización de estas permite concluir que la implementación de un cluster-based Web es escalable pues su empleo en el área de filtrado de paquetes ha demostrado su flexibilidad y eficiencia.

En un futuro próximo se planea desarrollar un módulo para iptables que implemente una política de distribución de carga dinámica que no requiera del típico intercambio de información de carga. La idea es que si se cuenta con un cluster localmente distribuido donde por el Web switch pasan todos los pedidos de conexión, éste puede registrar para cada uno de los servidores la cantidad de conexiones activas. De esta forma, para un cluster homogéneo el algoritmo de distribución es trivial: se envía el pedido actual a quien menos conexiones esté atendiendo. En caso de tratarse de un cluster heterogéneo simplemente debe ponderarse la capacidad de cada server en la toma de decisión.

Se puede argumentar que este mecanismo de balance de carga no brinda una buena distribución de carga cuando por ejemplo, los servidores consumen una importante porción de su poder de cómputo en el procesamiento de páginas dinámicas, pues el Web switch no puede predecir esta carga adicional. Una posible solución a este problema es agregar al sistema servidores back-end destinados a atender los pedidos dinámicos. Estos nuevos servidores reciben trabajo proveniente de los servidores del primer nivel, ahora dedicados a procesar el trabajo estático (que al afectar su carga de forma conocida no presenta problemas al Web switch) y de derivar el trabajo dinámico. Este mecanismo puede ser implementado en los servidores del primer nivel debido a que establecen conexiones con los clientes, pudiendo tomar decisiones más atinadas al contar con información más específica acerca de los pedidos.

En este tipo de situaciones el esquema planteado en este trabajo sigue siendo relevante para la distribución de las conexiones entre los nodos del primer nivel.

Referencias

[And01] Oskar Andreasson. Iptables tutorial 1.1.11. En *Netfilter. http://www.netfilter.org/documentation/index*, 2001.

[BC00] Paul Barford y Mark Crovella. Critical path analysis of tcp transactions. En *In Proceedings of ACM Sigcomm 2000*, páginas 127–138, Agosto 2000.

[CC01] Valeria Cardellini y Emiliano Casalicchio. The state of art in locally distributed web-server systems. Technical Report RC22209(W0110-048), IBM Research Division, 2001.

[CPS95] B. Callaghan, B. Pawlowski y P. Staubach. Hypertext transfer protocol. En *IETF RFC 1813*, 1995.

[Ech02] Javier Echaiz. Survey: Single System Image. Reporte Técnico, Laboratorio de Invetigación de Sistemas Distribuidos (LISiDi), 2002.

[EDA03] Javier Echaiz, Sergio Davicino y Jorge Ardenghi. Clasificación de las políticas de distribución de carga. Reporte Técnico, Laboratorio de Invetigación de Sistemas Distribuidos (LISiDi), 2003.

[FGM+97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk y T. Berners-Lee. Hypertext transfer protocol. En *IETF RFC 2068*, 1997.

[LA94] A. Luotonen y K. Altis. World wide web proxies, computer networks and isdn systems. En *First International Conference on WWW*, 1994.

[Mar02] Fabrice Marie. Netfilter extensions howto. En *Netfilter. http://www.netfilter.org/documentation/index*, 2002.

[Pos81] Jon Postel. Internet protocol specification. En *IETF RFC 791*, 1981.

[SLAID00] Junehwa Song, Eric Levy-Abegnoli, Arun Iyengar y Daniel Dias. Design alternatives for scalable web server accelerators. En *In Proc. of 2000 IEEE Intl Symp. on Performance Analysis of Systems and Software*, páginas 184–192, Abril 2000.

[STA01] A. Shaikh, R. Tewari y M. Agrawal. On the effectiveness of DNS-based server selection. En *Proceedings of IEEE Infocom 2001, Anchorage, Alaska*, 2001.

[VL00] Nikolaos Vasiliou y Hana Lutfiyya. Providing a differentiated quality of service in a world wide web server. En *ACM Performance Evaluation Review*, páginas 22–28, 2000.

[Voi02] Thiemo Voigt. Architectures for service differentiation in overloaded internet servers. Reporte Técnico, Dept. of Information Technology Uppsala University, 2002.

[Wan99] Jia Wang. A survey of web caching for the internet. En *ACM Computer Communication Review*, páginas 36–46, Junio 1999.