SERVIDORES VIRTUALES IMPLEMENTADOS EN EL SISTEMA OPERATIVO LINUX

EFRAIM WAINERMAN

Para la asignatura de Teleinformática y Redes, Universidad Nacional de Luján, rutas 5 y 7, Luján (6700), Argentina
e-mail: efraimw@unlu.edu.ar

Se analiza la tecnología para la implementación de servidores virtuales disponible en el sistema operativo Linux conocida como *Linux Virtual Server* [1]. Esta permite asignar uno o más equipos como redirectores o balanceadores de carga para una red conformada por varios equipos. En el caso de los redirectores son también conocidos como servidores virtuales, ya que son percibidos por los clientes como servidores de una aplicación específica, mientras que los últimos son llamados servidores reales, debido a que en estos es donde se ejecuta realmente la aplicación servidora.
En una primera parte se explican el objetivo, arquitectura y componentes que forman parte de esta solución. En una segunda parte se describe la experiencia de laboratorio en la que se montó un servidor web virtual utilizando la técnica de NAT (*Network Address Translation*) [8].

1 Introducción

Debido al continuo crecimiento que ha tenido Internet en los últimos años hoy día no es difícil encontrarse con servidores que reciban miles o millones de conexiones diarias, teniendo que atender concurrentemente quizás a cientos o miles de ellas. Esto trae aparejada la necesidad de contar con una enorme cantidad de recursos computacionales para enfrentar las demandas crecientes por parte de los usuarios del servicio.

Los requerimientos para los sistemas que hagan frente a estas demandas pueden resumirse en los siguientes:

1. **Escalabilidad**: cuando los requerimientos de un servicio aumentan, el sistema debe poder extenderse para poder atender todas las solicitudes sin que los tiempos de respuesta se vean afectados.
2. **Alta disponibilidad**: el servicio debe estar disponible en todo momento, para lo cual debe ser tolerante a fallas.
3. **Costos aceptables**: la inversión inicial y los costos de mantenimiento y expansión del sistema deben ser accesibles.

Para resolver estas dificultades se han desarrollado diversas soluciones basadas en el procesamiento paralelo. Estas se dividen en dos grandes enfoques: el *fuertemente acoplado* y, como contrapartida, el *débilmente acoplado*.

El primer enfoque se basa en la utilización de equipos con múltiples

procesadores que pueden ejecutar instrucciones concurrentemente y comparten el acceso a los datos locales, mientras que en el segundo se utilizan varios equipos independendientes que colaboran entre sí para cumplir un objetivo.

El enfoque de computación paralela fuertemente acoplada ha sido utilizado para todo tipo de tareas, sin embargo, esta solución, aunque efectiva y ampliamente aceptada, tiene serias desventajas principalmente en cuanto a escalabilidad y costos.

En la actualidad están emergiendo numerosas opciones basadas en computación distribuida que se presentan como una alternativa razonable a los equipos de mediano y gran tamaño.

2 Objetivos

El proyecto *Linux Virtual Server* tiene por objetivo utilizar una red de computadoras u otros dispositivos de forma tal que sean accedidos desde el exterior como si fuera un único servidor, el cual se conoce como servidor virtual.

Esto permite asignar numerosos equipos de pequeño tamaño y costo reducido conectados mediante una red local de alto rendimiento para que presten el servicio colaborando para lograr una capacidad computacional que crece casi linealmente con respecto a la capacidad computacional que agrega cada equipo en forma individual.

3 Componentes

El principal componente del software es un conjunto de *patches*, es decir modificaciones, que se aplican al código fuente del núcleo del sistema operativo Linux y que le agregan nuevas funcionalidades. Para ello, este nuevo código interactua con las rutinas de manejo del protocolo TCP/IP.

En las distribuciones más nuevas de Linux es probable que los *patches* ya hayan sido aplicados al kernel estándar y se incluyan también las herramientas de usuario por lo que no se requerirá la instalación de software adicional a menos que se necesite una versión más reciente que incluya alguna característica novedosa o la corrección de un error que afecte en forma notoria el funcionamiento de la aplicación a utilizarse.

En caso de compilarse el kernel de Linux, se requieren en forma genérica que las siguientes opciones se encuentren activadas –para la compilación estática o en forma de módulos-[1]:

1 Las opciones se verificaron con la versión 2.4.8-26mdk que se incluye con la distribución de Mandrake 8.1, deberían ser las mismas que para cualquier distribución del kernel de linux de las series 2.4.x. En LVS-mini-HOWTO se incluyen configuraciones para kernels de la serie 2.2.x [5].

Sección *"Networking Options"*:

- Packet socket
- Kernel/User netlink socket
- Routing messages
- Netlink device emulation
- Network packet filtering
- Socket Filtering
- Unix domain sockets
- TCP/IP networking
- IP: multicasting
- IP: advanced router
- IP: policy routing
- IP: use netfilter MARK value as routing key
- IP: equal cost multipath
- IP: use TOS value as routing key
- IP: verbose route monitoring
- IP: large routing tables
- IP: tunneling

Sección "*IP: Netfilter Configuration*":

- Connection tracking (requerida por masq/NAT)
- FTP protocol support
- IP tables support (requerida por filtering/masq/NAT)
- limit match support
- MAC address match support
- netfilter MARK match support
- Multiple port match support
- TOS match support
- Connection state match support
- Packet filtering
- REJECT target support
- Full NAT
- MASQUERADE target support
- REDIRECT target support
- Packet mangling
- TOS target support
- MARK target support
- LOG target support

Estos parámetros están asociados a componentes que ya son estándar en el kernel de Linux y por lo tanto se pueden utilizar para otro tipo de aplicaciones y también podrán variar de acuerdo a las necesidades específicas de la aplicación que se vaya a utilizar.

Tanto si los *patches* ya hubiesen sido aplicados al código fuente del kernel de Linux como si se utilizara una versión que ya incorpore estas modificaciones, se dispondrá de la sección "*IP: Virtual Server Configuration*". Las opciones para esta sección pueden ser activadas en su totalidad sin ningún problema y son las siguientes:

- virtual server support
- IP virtual server debugging
- IPVS connection table size = 12 (por defecto)
- round-robin scheduling
- weighted round-robin scheduling
- least-connection scheduling scheduling
- weighted least-connection scheduling
- locality-based least-connection scheduling
- locality-based least-connection with replication scheduling
- destination hashing scheduling
- source hashing scheduling
- FTP protocol helper

Las opciones de esta sección están asociadas en su mayoría con el código asociado a la implementación de los diversos *schedulings* que soporta el sistema y es un área donde permanentemente se hacen avances agregándose nuevos métodos para distribuir la carga en los servidores reales [2].

Otro componente que se desarrolla dentro del proyecto de LVS es la herramienta ipvsadm que, ejecutándose a nivel de usuario, permite la configuración del servidor virtual. Mediante esta utilidad se modifican los parámetros de funcionamiento del software de servidor virtual que se ejecuta en el kernel a través de su ejecución mediante el intérprete de comandos o bien como un listado de órdenes dentro de un script. Además se cuenta con los scripts ipvsadm-save e ipsadm-restore que permiten guardar los parámetros actuales del sistema en una archivo y restaurar los parámetros respectivamente.

Es necesario para el correcto funcionamiento del sistema que estos componentes se encuentren disponibles en el/los redirectores. Sin embargo no se necesitan modificaciones en las aplicaciones clientes y servidoras, ya que los clientes perciben al sistema como un único servidor y, por otro lado, los servidores perciben los requerimientos que le llegan como si vinieran directamente del cliente.

De esta forma y en la actual implementación se requiere que el/los redirectores ejecuten el sistema operativo Linux, sin embargo no existen requisitos especiales de software para clientes y servidores, excepto para las configuraciones utilizando

IP Tunneling y Direct Routing, donde se requerirá una configuración especial del sistema operativo, pero no restringiéndose únicamente a Linux (ver la siguiente sección).

4 Arquitectura

Se utiliza una arquitectura débilmente acoplada que permite utilizar un equipo como redirector de las conexiones entrantes al sistema distribuyéndolas de acuerdo a un algoritmo de *scheduling* determinado en un grupo de servidores. Esto permite que los distintos equipos compartan la carga de trabajo, pudiéndose escalar el sistema simplemente agregando nuevos servidores a la red.

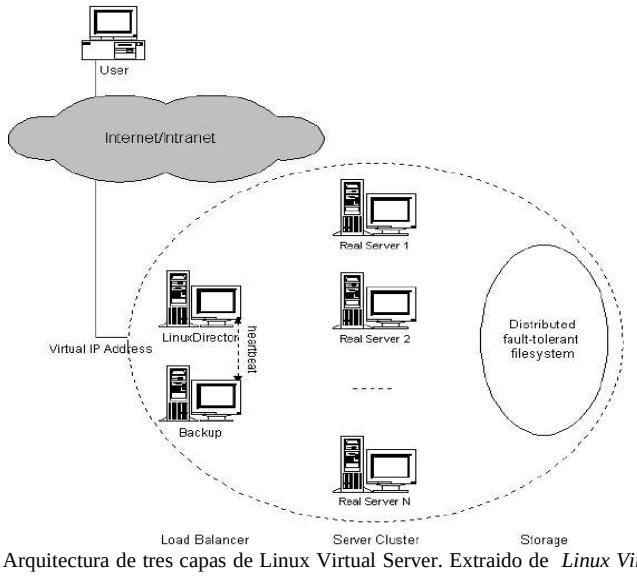

Figura 1. Arquitectura de tres capas de Linux Virtual Server. Extraido de *Linux Virtual Server for Scalable Network Services* [1].

Como se ve en la figura 1 el sistema utiliza una arquitectura de tres capas formadas por el/los redirectores, los servidores reales y el sistema de archivos compartido (aunque este último componente no es estrictamente indispensable).

4.1 El redirector

Es visto desde el exterior como el servidor ya que tiene asignada la dirección IP con que se publica el servicio. Puede recibir múltiples conexiones y redirigirlas a númerosos servidores reales según el o los criterios de balanceo de carga que se hallan adoptado.

Dado que la aplicación servidora no se ejecuta en este equipo y que el software encargado de redireccionar las conexiones se ejecuta en su kernel disminuyendo drásticamente el overhead de procesamiento, es capaz de atender muchas más conexiones que las que podría atender si las conexiones se hicieran directamente a este equipo, manteniendo los tiempos de respuesta.

4.2 Los servidores reales

Es un conjunto de servidores en cada uno de los cuales se ejecuta la aplicación. Esta puede ser cualquiera que utilice el protocolo TCP/IP. Sin embargo en el caso de que el servidor tenga que iniciar la apertura activa de conexiones con el cliente -por ejemplo FTP-, esto puede causar problemas, lo cual se soluciona a través de la incorporación de módulos específicos en el redirector para el manejo de la aplicación.

4.3 El repositorio de datos compartido

Aunque este no es un componente fundamental del sistema, provee la facilidad de poder administrar los datos en forma centralizada.

Este repositorio común de datos podría no estar presente en el sistema, utilizándose un acceso local a los datos por parte de los servidores reales, pero de esta manera se haría muy dificultoso el proceso de modificar los datos de la aplicación como son por ejemplo los archivos html en un servidor Web.

Debido a que estos repositorios serán accedidos por todos los servidores reales al mismo tiempo, es necesario que sean también escalables y tolerantes a fallas.

Otro aspecto a tener en cuenta al implementar el sistema de archivos compartido es que, en caso de que la aplicación realizara cambios en los datos, es necesario un esquema de manejo de bloqueos, ya que las aplicaciones que se ejecutan en los servidores reales accederán a los datos en forma concurrente y es fundamental para la coherencia de los datos que no se permita a dos aplicaciones modificar un mismo dato compartido al mismo tiempo. Este esquema puede estar incorporado en el sistema de archivos compartido o puede ser manejado por alguna aplicación específica como por ejemplo un servidor de base de datos.

Ejemplos de este tipo de sistemas de archivos en Linux son GFS [10], Coda [11] e Intermezzo [12].

5 Alta Disponibilidad

Aunque el software del proyecto LVS no provee alta disponibilidad, se pueden implementar mecanismos que permitan monitorear el estado de los servidores reales para que, en caso de falla de alguno de ellos, simplemente se elimine de la

lista que mantiene el redirector y el sistema siga funcionando normalmente.

Para evitar la caida del sistema en caso de falla del redirector principal se utiliza un equipo de respaldo que cuando detecta una falla en el primero toma su lugar apoderándose de su dirección IP. Esto se realiza utilizando la técnica de arp-spoofing [9].

La detección tanto de la caida de un servidor real como del redirector por parte del servidor de respaldo se realiza a través del monitoreo constante a través de demonios específicos como por ejemplo *hearbeat*[2] ejecutándose en el redirector y el redirector de respaldo respectivamente [3, 4].

6 Técnicas de Balanceo de carga

Como se mencionó anteriormente, Linux Virtual Server permite la implementación de un servicio virtual a través del uso de un balanceador de carga para una red formada por los servidores reales que ejecutan la aplicación.

En esta arquitectura las peticiones de servicio de los clientes son enviados hacia el balanceador de carga. El primer paquete que le llega a este es uno de inicio de conexión, el cual tiene como destino la dirección IP que el redirector tiene asignada para el servicio virtual. El redirector examina la dirección IP y el puerto de destino, si coincide con alguno de los servicios publicados según la tabla de servicios virtuales elige un servidor real para ese servicio mediante el algoritmo de scheduling que haya sido seleccionado para ese servicio. Una vez que el servidor real ha sido seleccionado, se agrega la conexión a la tabla correspondiente y se reenvía el paquete al servidor real correspondiente haciendo uso de alguna de las tres técnicas de balanceo de carga que se describen en las siguientes secciones. La técnica a utilizar dependerá del servidor real elegido, ya que este parámetro se puede configurar independientemente para cada servidor real.

Los paquetes que lleguen posteriormente y que pertenezcan a esta conexión serán identificados mediante la tabla de conexiones y luego reenviados al servidor real correspondiente.

Al recibir los paquetes el servidor real, este los percibe como provenientes del cliente, los procesa y luego devuelve la contestación al cliente. La forma en que es enviada esta información al cliente varía según la técnica de balanceo de carga utilizada, pero la aplicación percibirá siempre que está enviando los datos directamente al cliente.

2 Estos demonios envian mensajes que pueden ser tipo ping o bien de prueba del servicio a intervalos regulares y si no reciben respuesta pueden proceder a eliminar un servidor real de la lista que se encuentra en el redirector o pueden iniciar la toma de posesión de la dirección IP del balanceador de carga para tomar su lugar.

6.1 *LVS a través de NAT* (*Network Address Translation*)

Esta técnica consiste en la modificación de los encabezados de los datagramas IP con el objeto de reemplazar sus direcciones IP de fuente (*source NAT*) o destino (*destination NAT*) realizando el mapeo de un grupo de direcciones a otro [8].

Un tipo especial de source NAT conocido como enmascaramiento (*masquerading*) es ampliamente utilizado en las redes actuales principalmente para dar acceso a Internet a equipos que utilizan direcciones IP privadas y que no podrían hacerlo directamente. Se utiliza entonces un dispositivo intermedio actuando como gateway que posee una dirección pública de Internet el cual modifica los paquetes provenientes de las estaciones de trabajo asignándoles como fuente su propia dirección IP para que los datagramas puedan atravesar la red de Internet.

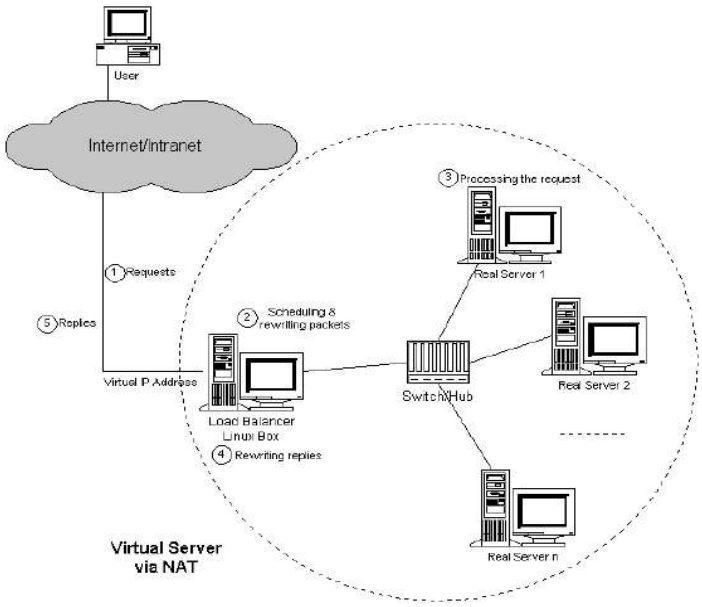

Figura 2. Arquitectura de Linux Virtual Server mediante NAT. Extraido de ***Linux Virtual Server for Scalable Network Services***.

Una extensión a NAT es la técnica conocida como *network address port translation* la cual se basa en el mapeo de direcciones de red y puertos TCP/UDP de los paquetes en otros.

Esta última técnica permite que numerosos servicios ejecutándose en distintas direcciones de red y escuchando en distintos puertos puedan ser trasladados a un único puerto en la dirección de red del servidor virtual para lograr la apariencia de

un único servicio.

En la figura 2 se puede observar una configuración típica de Linux Virtual Server a través de NAT en la que los servidores reales y el balanceador de carga se hallan interconectados a través de un Switch/Hub utilizando una topología de estrella. Esta es una configuración que utiliza en forma eficiente el ancho de banda disponible en la red al separar los datos a enviarse a los servidores reales en sus respectivos canales.

El reenvio de los paquetes paquetes provenientes de los clientes mediante NAT se realiza modificando los campos de dirección IP y puerto de destino de los mismos por los correspondientes a la configuración del servidor real escogido para recibir la petición.

Una vez que los servidores reales procesan la petición, envían su respuesta al cliente. Debido a que los servidores reales en esta configuración utilizan como gateway al balanceador de carga, la respuesta será enviada a través de este. El mismo entonces reescribe la dirección y puerto de origen de manera de que coincidan con los correspondientes al servicio virtual y lo reenvia al cliente.

La configuración de un servidor virtual mediante NAT es muy transparente, ya que no requiere modificaciones ni configuraciones especiales en los sistemas operativos de los servidores reales, lo cual facilita su implementación.

Como desventaja de esta técnica se puede mencionar la restricción de que los servidores reales se deban encontrar en la misma red que el balanceador de carga, sin embargo esto normalmente no representa un impedimento grave dado que esta es una configuración estándar.

Sin embargo, la principal desventaja de esta técnica es su escalabilidad, debido al overhead en que se incurre al tener que reescribir todos los paquetes de datos originados por los servidores reales que se hace notorio cuando el número de estos últimos es muy grande. Este efecto adverso puede disminuirse utilizando un equipo más potente como redirector.

La solución a este problema consiste en que los servidores reales envíen los paquetes de datos de salida directamente hacia los clientes sin pasar por el redirector a través de las técnicas que se describen en las siguientes secciones.

6.2 LVS a través de IP Tunneling

La técnica de IP Tunneling consiste en encapsular un datagrama IP dentro de otro. Se hace posible entonces que el balanceador de carga encapsule los datagramas IP provenientes de los clientes dentro de otros datagramas IP que son enviados hacia los servidores reales logrando que estos luego envíen los paquetes de respuesta directamente hacia los clientes.

En la figura 3 se muestra la arquitectura de Linux Virtual Server utilizando la técnica de IP Tunneling. Para el reenvio de los paquetes, el redirector encapsula los datagramas con dirección IP del cliente como origen y destino en la dirección IP del servicio virtual dentro de datagramas con dirección IP del redirector como

origen y dirección IP del servidor real como destino.

Estos nuevos paquetes pueden dirigirse directamente hacia los servidores reales o ser reenviados por dispositivos intermedios como routers, atravesando varias redes IP en el camino. Esto se logra porque estos paquetes son vistos por cualquier dispositivo de red como paquetes IP normales.

Una vez que el paquete llega al servidor real, este lo desencapsula y comprueba que está dirigido hacia la dirección IP virtual configurada en su dispositivo de tunneling, así que procesa el datagrama y luego envía la respuesta directamente hacia el cliente.

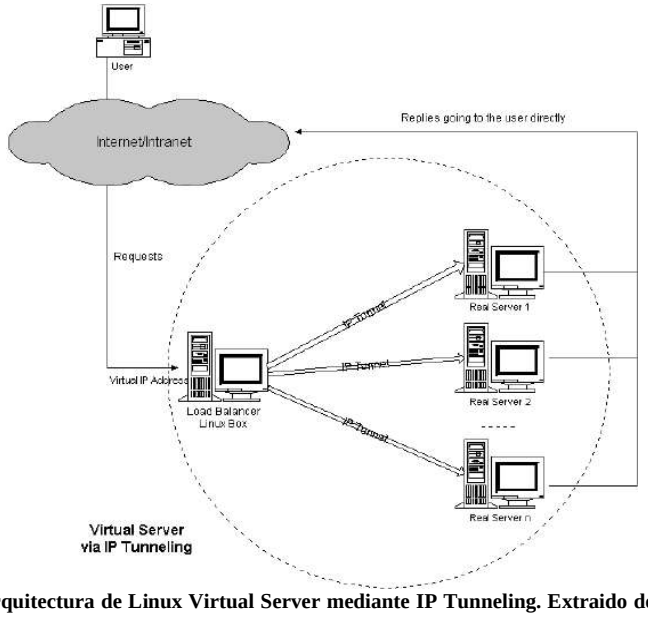

Figura 3. Arquitectura de Linux Virtual Server mediante IP Tunneling. Extraido de *Linux Virtual Server for Scalable Network Services.*

La configuración de Linux Virtual Server a través de IP Tunneling es la más flexible respecto a la configuración física de la red, no se tienen restricciones en cuanto a la ubicación física de los servidores reales, los cuales podrían no encontrarse en la misma red local.

Por otro lado, esta técnica impone la restricción de que los sistemas operativos de los servidores reales deben soportar el protocolo IP Tunneling y estar configurados apropiadamente.

Utilizando esta técnica se evita el overhead que representa realizar NAT en el redirector para los paquetes de salida. Esto multiplica la escalabilidad del sistema para prácticamente todas las aplicaciones cliente/servidor ya que normalmente el volumen de los datos que envía el cliente al servidor es significativamente menor con respecto a los que el servidor envía al cliente.

De esta forma el límite en cuanto a cantidad de servidores reales que pueden agregarse al sistema obteniendo un aumento lineal de rendimiento es mucho más alto que el que se tiene utilizando NAT.

6.3 LVS a través de Direct Routing

Esta técnica de balanceo de carga utiliza una red LAN para la interconexión de los servidores y el balanceador de carga con la particulad de que sus interfaces de red deben estar interconectadas utilizando un segmento ininterrumpido de la red como por ejemplo un Switch/HUB (ver fig. 4).

En esta configuración, todos los servidores reales tienen un dispositivo de red al cual se ha asignado la dirección IP del servidor virtual. Este dispositivo podrá ser cualquiera que pueda ser configurado en el servidor, típicamente el dispositivo de loopback, pero para que el sistema funcione correctamente el mismo no deberá responder a los mensajes ARP, ya que esto ocasionaría conflictos de red al utilizar todos los equipos una única dirección (ver *the arp problem* en *LVS HOWTO* [6]).

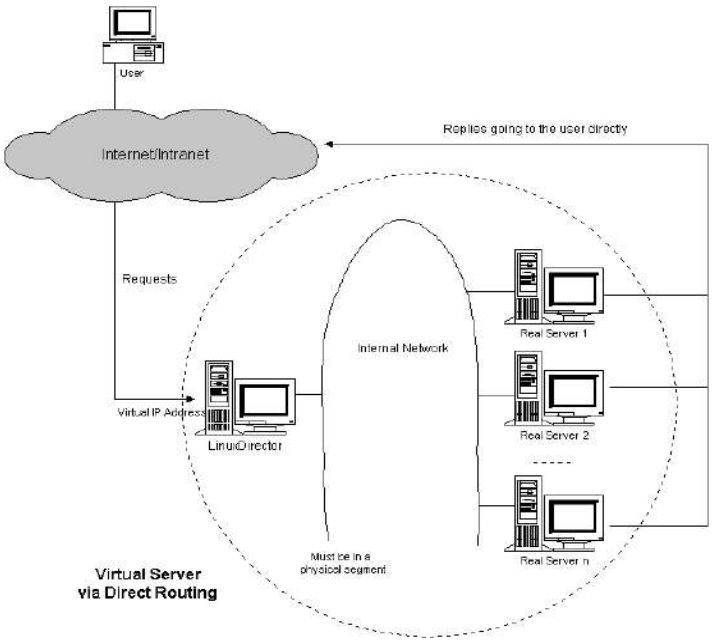

Figura 4. Arquitectura de Linux Virtual Server mediante Direct Routing. Extraido de *Linux Virtual Server for Scalable Network Services.*

Para realizar el reenvio de los paquetes, el balanceador de carga sólo cambia

las direcciones MAC para direccionar los paquetes al servidor real elegido mientras que la dirección IP de destino sigue siendo la del servicio virtual. Los servidores reales pueden aceptar los paquetes IP entrantes ya que tienen un dispositivo con esa dirección IP asignada. Luego procesan el requerimiento y envían la respuesta directamente hacia el cliente.

La configuración de Linux Virtual Server a través de Direct Routing impone la restricción de que los dispositivos de red que tengan asignada la dirección del servicio virtual deban no responder a las peticiones de ARP, lo cual puede no ser un problema trivial en algunos sistemas operativos (curiosamente esto se da en los kernels 2.2.x y 2.4.x de Linux, ver *LVS HOWTO* [6]).

Por otro lado esta técnica, al igual que la de IP Tunneling, multiplica la escalabilidad del sistema al descargar del balanceador de carga la tarea de redireccionar los paquetes de datos de los servidores a los clientes, pero como ventaja adicional con respecto a IP Tunneling cuenta con el hecho de que se elimina el overhead derivado del encapsulamiento/desencapsulamiento de los datagramas IP.

7 Experiencia de laboratorio: Implementación de un servidor web virtual mediante NAT

7.1 Introducción

Para realizar la experiencia práctica de implementación de un servidor virtual en linux se optó por configurar un servicio de http, ya que este es de fácil configuración y, en su modo de operación básica, no presenta poblemas para ser implementado mediante un servicio virtual.

7.2 Software utilizado

El principal componente de software que se usó para realizar la experiencia fue el código encargado de ejecutar el servicio de balanceo de carga que provee Linux Virtual Server, el cual se integra al núcleo del sistema operativo linux. En el caso de la experiencia realizada se optó por una solución a través de núcleo precompilado utilizando el que se halla incluido en la distribución *Linux Mandrake 8.1* (cuya versión es la 2.4.8)[3].

Para poder utilizarse la funcionalidad presente en el núcleo de linux es necesario contar con la utilidad a nivel de usuario *ipvsadm* que es la que se encarga de agregar los servicios virtuales y servidores reales a la lista que maneja el módulo de software que se ejecuta en modo kernel. Esta utilidad también puede descargarse gratuitamente de la página principal del proyecto de *Linux Virtual Server*. La que se utilizó en la experiencia fue la incluida en la distribución de Linux Mandrake

3 Véase la sección 3 para más detalles acerca de este componente de software.

8.1.

Para completar los requisitos de software en el servidor virtual fue necesario utilizar un componente de software capaz de realizar NAT, ya que esta fue la técnica utilizada para implementar el servidor virtual. En este caso, el software del proyecto LVS se integra con el que ya se provee en forma estándar dentro del núcleo de Linux para manejo de NAT. En series 2.2.x del núcleo de Linux, esta funcionalidad se lleva a cabo mediante el software *ipchains*, mientras que en las más recientes series 2.4.x esta funcionalidad se realiza mediante *Netfilter,* manteniéndose sin embargo la retrocompatibilidad con ipchains.

Ambas versiones del componente de NAT de Linux, en forma similar al software de LVS, constan de una parte que se integra al núcleo de Linux y una utilidad que se ejecuta en modo usuario para la administración de las reglas de NAT, en el caso de *ipchains*, la utilidad de usuario lleva el mismo nombre, mientras que en el caso de la utilidad para la administración de *Netfilter*, esta utilidad es *iptables*. Debido a que la versión nativa para el núcleo utilizado es *Netfilter*, esta fue la elegida como soporte para NAT.

El software que se utilizó como aplicación servidora fue el servidor web Apache. En este caso también se utilizaron las versiones del software incluidas con las respectivas distribuciones, es decir la versión 1.3.14 para la PC 15 con Conectiva Linux mientras que en la PC 16 con Suse Linux se utilizó la versión 1.3.17 del producto.

Por último el cliente utilizado fue Netscape Navigator versión 4.76 ejecutándose en Windows NT Server 4.0.

7.3 Configuración de la red

Para realizar la experiencia se utilizaron cuatro computadoras personales (Pcs) conectadas mediante una red ethernet. Adicionalmente se utilizó una quinta computadora para realizar la captura de las tramas enviadas por la red LAN durante la prueba.

La conexión física consistió en un único segmento de red que utiliza un hub para la interconexión en bus de los equipos. La topología a nivel de red es la que se muestra en la figura 5.

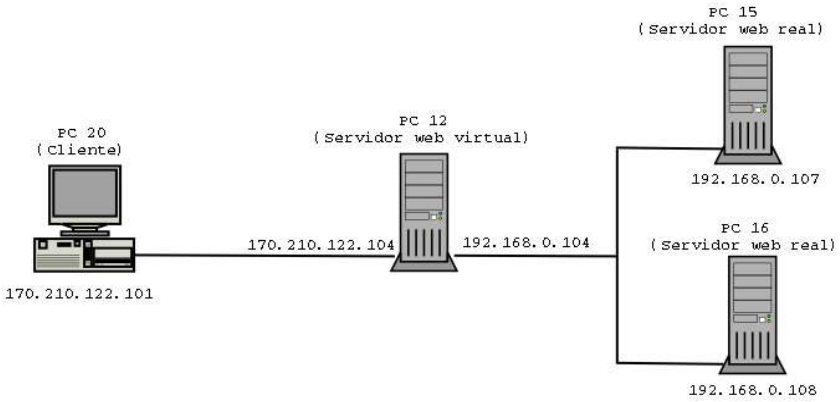

Figura 5. *Topología de la red que se utilizó para la implementación del servidor virtual.*

Como puede advertirse, el cliente y el servidor virtual utilizan una configuración en donde ambos pertenecen a la misma red IP 170.210.122.0/25, es decir que pueden alcanzarse sin necesidad de utilizar nodos intermedios. Aunque esto no es un requisito indispensable, sí es necesario que exista un camino por el cual los paquetes puedan transmitirse entre estos dos equipos. A su vez la dirección IP del servidor virtual es una dirección públicamente conocida – y eventualmente, aunque este no es el caso, registrada con un nombre en un servidor DNS -.

Para los servidores reales se definió la red IP privada 192.168.0.0/25.

La configuración de hardware y software de cada una de las máquinas se muestra en la tabla 1.

Tabla 1. Configuración de hardware y software de las máquinas utilizadas para realizar la experiencia.

Nombre	***CPU***	***Memoria***	***Interfaces de red***	***Dirección IP***	***Sistema Operativo***
PC12	Pentium 233 MMX	64 MB	eth0: placa de red ethernet de 10 mbps PCI. eth0:0: interface de red virtual	170.210.122.104 192.168.0.104	Mandrake Linux 8.1
PC15	Pentium 233 MMX	64 MB	eth0: placa de red ethernet de 10 mbps PCI	192.168.0.107	Suse Linux 7.1
PC16	Pentium 233 MMX	64 MB	eth0: placa de red ethernet de 10 mbps PCI	192.168.0.108	Conectiva Linux 6.0
PC20	Pentium 233 MMX	64 MB	eth0: placa de red ethernet de 10 mbps PCI	170.210.122.101	Windows NT Server 4.0

7.4 Configuración de los equipos

En esta sección se comentarán los aspectos relativos a la configuración realizada en cada uno de los equipos a fin de poner en funcionamiento la red que se mostró en la anterior sección.

7.4.1. El cliente

Este equipo no cuenta con ninguna configuración especial contando solamente con un navegador web para realizar el requerimiento http al servidor virtual. Cabe mencionar sin embargo que se modificó la configuración de red del navegador para que no hiciera los requerimientos mediante proxy como en su configuración normal, sino que por el contrario las realizase mediante conexión directa con el objeto de hacer más claras las capturas de red para su posterior análisis.

7.4.2. El redirector

Este equipo posee una interfaz de red tipo ethernet referenciada por el sistema como eth0 a la cual se ha asignado la dirección IP pública 170.210.122.104. Esta dirección está disponible para ser accedida desde cualquier máquina en Internet.

Sobre esta interfaz física se ha configurado una interfaz virtual denominada eth0:0 a la cual se ha asignado la dirección IP privada 192.168.0.104 para poder efectuar la comunicación con los servidores reales.

Dicha configuración se realizó utilizando la herramienta *linuxconf*.

Luego de realizada la configuración básica de la red, se agregaron las reglas de funcionamiento de LVS y NAT mediante el script para configuración del firewall que en la distribución utilizada se encuentra en /etc/rc.d/rc.firewall. El contenido de este archivo se muestra en el anexo 1.

La primera línea del script establece cual será el intérprete de comandos a utilizar, en este caso sh.

```
#!/bin/sh
```

A continuación se activa el reenvio de paquetes, es decir la posibilidad de rutear paquetes de red que le lleguen al redirector y cuya dirección IP de destino no sea este equipo.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

A continuación se describen las líneas del script utilizadas para realizar la

configuración de la funcionalidad de NAT en el equipo. Esto es necesario debido a que la técnica de balanceo de carga elegida fue justamente esta última.

Debido a que en la configuración del núcleo utilizado el código encargado del manejo de NAT se encuentra compilado en forma de módulo, este debe ser cargado para poder usarse. En la siguiente línea se carga dicho módulo [7].

```
modprobe iptable_nat
```

Luego se establece cual será el comando encargado de realizar la configuración de los parámetros del módulo. En este caso sólo se utilizará para la configuración de las funciones de NAT.

```
IPTABLES=/sbin/iptables
```

Se define la red local donde se encuentran los servidores reales. Se usa la dirección 192.168.0.0 y la máscara de red 255.255.255.128.

```
REDLOCAL=192.168.0.0/25
```

A continuación se limpia cualquier configuración anterior que pudiera encontrarse en memoria.

```
$IPTABLES -F
```

En la siguiente línea se agrega una regla a la tabla POSTROUTING [8] que indica que a cualquier datagramas IP que llegue desde cualquier terminal dentro de la red local debe aplicársele la técnica de masquerade (ver sección 6.1).

```
$IPTABLES -t nat -A POSTROUTING -s $REDLOCAL -j MASQUERADE
```

Adicionalmente debe agregarse esta línea para que se acepte el ruteo de los datagramas originados en la red local.

```
$IPTABLES -A FORWARD -s $REDLOCAL -j ACCEPT
```

En la última sección del script se configuran las reglas del servidor virtual propiamente dicho.

Para esto se define en primer lugar cual será el comando encargado de modificar los parámetros de funcionamiento del módulo LVS. Se configura con ipvsadm, utilidad incluida con el software de LVS.

```
IPVSADM=/sbin/ipvsadm
```

Luego se limpia cualquier configuración previa del módulo que pudiera

encontrarse en memoria.

$IPVSADM -C

Por último se define el servicio virtual que se va a poner en funcionamiento.
En este caso se especificó un servicio virtual en la dirección IP 170.210.122.104 y puerto 80 con un scheduling utilizando el algoritmo de round robin. Todo datagrama IP dirigido a la dirección de red 170.210.122.104 y que contenga un segmento TCP con dirección de puerto 80 ya no será procesado por el código normal de TCP/IP sino que será "interceptado" por el código de LVS que se ejecuta dentro del núcleo.

$IPVSADM -A -t 170.210.122.104:80 -s rr

Por último se agregan al servicio virtual definido anteriormente dos servidores reales -con direcciones IP 192.168.0.107 y 192.168.0.108 respectivamente-. El parámetro -t especifica que este es un servicio tipo TCP, mientras que -m especifica que el balanceo de carga se hará mediante masquerade -es decir NAT-. Cabe aclarar que el puerto donde aceptan peticiones no necesariamente tiene que ser coincidente con el del servicio virtual.

$IPVSADM -a -t 170.210.122.104:80 -r 192.168.0.107:80 -m
$IPVSADM -a -t 170.210.122.104:80 -r 192.168.0.108:80 -m

Es importante notar que dada la forma en que se definen los servicios virtuales se permiten tener múltiples servicios virtuales utilizando cada uno diferentes algoritmos de scheduling. Por otro lado, tal como se definen los servidores reales pueden usarse distintas técnicas de balanceo de carga con cada uno de ellos (por ejemplo podría tenerse algún servidor real que se encontrase fuera de la red local haciendo necesario el uso de IP Tunneling).

7.4.3. Los servidores reales

En los mismos se estableció como router por defecto la dirección 192.168.0.104 correspondiente a la interfaz virtual eth0:0 del redirector. La configuración se realizó utilizando las herramientas de configuración del sistema propias de cada una de las distribuciones de linux sobre las que se montaron los servidores web (YaST en SuSE y Linuxconf en el caso de Conectiva).
Por último hay que mencionar que ejecutan un servidor web apache con una configuración estándar atendiendo a los requerimientos http en su puerto 80.

7.5 Análisis del funcionamiento de LVS a través de las capturas de red

En esta sección se mostrará un caso práctico del funcionamiento del sistema a través del comentario de las capturas de red que se tomaron durante una sencilla prueba realizada utilizando la implementación que se describió durante el transcurso de esta sección. La misma consistió en realizar una petición http de una página web conteniendo texto y con tamaño lo suficientemente pequeño como para poder ser transportado en una sola trama ethernet para no generar demasiadas tramas que sólo entorpecieran el análisis de la captura.

Las capturas de red se presentan completas en el anexo 2, y se hallan numeradas para ser referenciadas en el comentario que se desarrolla en la siguiente sección.

7.5.1. Descripción de las tramas

La primera trama ethernet corresponde al pedido ARP que realiza el cliente al servidor virtual. Cabe aclarar que esto es así debido a que tanto el cliente como el servidor virtual se hallan en el mismo segmento de red. En un caso típico este pedido ARP podría ser realizado por un router de entrada a la red local. El objetivo de esta trama es obtener la dirección ethernet del servidor virtual conociendo su dirección IP.

La trama 2 es la contestación que realiza el servidor virtual dando a conocer la dirección física de la interfaz de red donde presta servicio.

La trama 3 lleva un paquete IP que a su vez contiene un segmento TCP de apertura de conexión (flag SYN activado) enviado desde el cliente hacia el servidor virtual. El puerto destino para realizar esta conexión es el 80 correspondiente al servicio web (en este caso virtual, pero este es un detalle que el cliente no conoce), mientras que se seleccionó 1454 como puerto efímero del lado del cliente.

La trama 4 corresponde al pedido ARP que realiza el servidor virtual mediante su interfaz de red virtual eth0:0 -con dirección IP 192.168.0.104- con el objeto de obtener la dirección física de la interfaz de red correspondiente a la dirección 192.168.0.108 (PC 16) dentro de su misma red IP. Esta dirección corresponde al servidor real escogido de acuerdo al algoritmo de scheduling asignado al servicio virtual, en este caso round robin.

El redirector, a continuación agrega la dirección IP del servidor real junto con la identificación de la conexión dada por el par 170.210.122.101:1454 como un ítem dentro de la tabla hash de conexiones. Esto permite que los nuevos paquetes entrantes al redirector provenientes del cliente y que correspondan con esta conexión sean redirigidos hacia un único servidor real durante el tiempo que dure la misma con el objetivo de mantener la coherencia.

La trama 5 es la contestación al pedido ARP por parte de la PC 16.

La trama 6 corresponde al reenvío del segmento de pedido de apertura d e conexión originado en el cliente y dirigido hacia el redirector. En este caso se observa como el redirector realizó el proceso de NAT y más precisamente *destination NAT*, al modificar la dirección de destino del datagrama IP que pasó de ser 170.210.122.104 a ser 192.168.0.108.

En la trama 7, el servidor real confirma el pedido de apertura de conexión mediante un segmento TCP con los flags SYN y ACK activados. Desde el punto de vista de IP, se trata de un datagrama con origen en la dirección del servidor real y destino en la dirección del cliente. Sin embargo la dirección ethernet corresponde a la del redirector. Esto es así porque el servidor real está utilizando al redirector como *gateway* por defecto para enviar el datagrama al cliente.

En la trama 8, el redirector reenvía la respuesta del servidor real hacia el cliente, con la particularidad de que modifica la dirección IP de origen utilizando la suya. Este proceso es otra forma de NAT conocida como *masquerading*.

En la trama 9, el cliente termina el proceso de establecer la conexión TCP enviando al servidor virtual un segmento TCP con el flag ACK activado.

En la trama 10 el redirector reenvia el datagrama anterior hacia el servidor real, realizando nuevamente *destination NAT.* En este caso la dirección de destino no se elige mediante un mecanismo de scheduling, ya que se asignó anteriormente el servidor con dirección IP 192.168.0.108 como destino para esta conexión.

En la trama 11, el cliente envia al servidor virtual el pedido de http, el cual es reenviado por el redirector hacia el servidor real en la trama 12.

Las tramas 13 y 14 corresponden a la confirmación del pedido http que genera el servidor real y que es enviada hacia el cliente pasando por el redirector.

Las tramas 15 y 16 llevan los segmentos TCP con la carga http como respuesta del servidor real al pedido realizado por el cliente.

Las tramas 17 a la 24 corresponden al proceso de cierre de la conexión TCP. Este es iniciado por el servidor real realizándose como siempre a través del redirector y no se diferencia por lo demás de un proceso normal de cierre de conexión TCP.

8 Discusión

En una primera parte del trabajo se ha dado una descripción general de la tecnología de *Linux Virtual Server*. La misma ofrece una opción muy flexible para la implementación de servidores con altas exigencias en cuanto a la carga de trabajo.

Se destaca como una de las características más importantes la de su gran escalabilidad, pudiéndose instalar una infraestructura de red mínima para poner en funcionamiento el servicio para luego hacerla crecer según vayan aumentando los requerimientos. Esto puede lograrse en forma sencilla agregando nuevos servidores reales y realizando cambios mínimos de configuración en el equipo que funcione como balanceador de carga.

Por otro lado se puede lograr alta disponibilidad en el servicio utilizando otros paquetes de software libre como por ejemplo los que se desarrollan a través del proyecto ultramonkey [3] o linux-ha [4].

En una segunda parte se realizó una experiencia de laboratorio configurando un servidor web virtual mediante NAT. Esta última es una técnica muy extendida y bien conocida por los administradores de red y se presenta como una opción muy adecuada para la implementación del servicio virtual al no requerir grandes modificaciones en los equipos donde se ejecutará la aplicación servidora.

Al realizar la prueba se observó que la puesta en marcha del sistema no es demasiado compleja y se realiza en forma muy transparente en los servidores reales.

Por último cabe destacar que el *Linux Virtual Server* es un proyecto de software libre, por lo que se encuentra en permanente proceso de desarrollo, dando soporte a las nuevas tecnologías que aparecen en el ámbito de las redes, incorporando nuevas funcionalidades y mejorando las ya existentes.

9 Agradecimientos

Quisiera agradecer especialmente al personal del Centro de Cómputos de la Facultad de Ingeniería de la Universidad Nacional de La Pampa por su colaboración facilitándome el acceso a la red de computadoras en la cual se realizó la experiencia de laboratorio.

Referencias

1. Wensong Zhang, Linux Virtual Server for Scalable Network Services, home page http://www.linuxvirtualserver.org.
2. Virtual Server Scheduling Algorithms, home page http://www.linuxvirtualserver.org/docs/scheduling.html.
3. Proyecto UltraMonkey: http://ultramonkey.sourceforge.net.
4. Proyecto de Alta Disponibilidad en Linux, home page http://www.linux-ha.org/.
5. Joseph Mack, LVS mini-HOWTO, abril de 2001, disponible en http://www.linuxvirtualserver.org/Joseph.Mack/mini-HOWTO/LVS-mini-HOWTO.html.
6. Joseph Mack, LVS HOWTO, diciembre de 2001, disponible en http://www.linuxvirtualserver.org/Joseph.Mack/HOWTO/LVS-HOWTO-1.html.
7. Oskar Andreasson, Iptables Tutorial 1.1.2, 2001.
8. Rusty Russell, Linux 2.4 NAT Como, disponible en http://netfilter.samba.org/documentation/HOWTO/es/NAT-HOWTO.html.
9. Sean Whalen, An Introduction to ARP Spoofing, abril de 2001, disponible en http://chocobospore.org/arpspoof.
10. Proyecto Global File System, home page http://ww.globalfilesystem.org.
11. Proyecto Coda, home page http://www.coda.cs.cmu.edu/.
12. Proyecto Intermezzo, home page http://intermezzo.org.

Anexo 1: archivo /etc/rc.d/rc.firewall en el redirector

```
#!/bin/sh

# En este archivo se han incluido los parámetros estrictamente necesarios para el
# funcionamiento del servidor virtual, excluyéndose cualquier configuración
# referente a aspectos de seguridad.

# Se carga el módulo de NAT
modprobe iptable_nat

# Se habilita el reenvío de paquetes IP
echo 1 > /proc/sys/net/ipv4/ip_forward

# Se define el comando para la administración de iptables
IPTABLES=/sbin/iptables

# Definición de la red local donde se encuentran los servidores reales.
REDLOCAL=192.168.0.0/25

# Se limpia cualquier configuración anterior de iptables
$IPTABLES -F

# Se indica que para cualquier paquete con origen en la red local se realizará el
# proceso de masquerading
$IPTABLES -t nat -A POSTROUTING -s $REDLOCAL -j MASQUERADE

# Se permite el reenvío de los datagramas con origen en la red local
$IPTABLES -A FORWARD -s $REDLOCAL -j ACCEPT

# Se define el comando para la administración del servidor virtual.
IPVSADM=/sbin/ipvsadm

# Se limpia cualquier configuración anterior de ipvsadm
$IPVSADM -C

# Se agrega el servicio virtual en el puerto 80 utilizando round robin (rr) como
# scheduler
$IPVSADM -A -t 170.210.122.104:80 -s rr

# Se agregan los servidores reales asociados al servicio virtual definido en la
# llamada anterior.
# Estos se identifican con sus direcciones IP y puertos donde atienden
# el servicio.
```

```
# Se especifica en ambos que la técnica de balanceo a utilizarse será la de NAT
# (mediante el parámetro -m)
$IPVSADM -a -t 170.210.122.104:80 -r 192.168.0.108:80 -m
$IPVSADM -a -t 170.210.122.104:80 -r 192.168.0.107:80 -m
```

Anexo 2: Capturas completas

Trama 1:

Ethernet (1013902103.199682)

Hardware source: 00:80:ad:c8:19:91
Hardware destination: ff:ff:ff:ff:ff:ff
Type / Length: 0x806 (ARP)
Media length: 60

ARP Header

Hardware type: 0x1 (Ethernet)
Protocol type: 0x800 (IP)
Hardware length: 6
Protocol length: 4
Opcode: 1 (request)
Sender Ether address: 00:80:ad:c8:19:91
Sender IP address: 170.210.122.101
Target Ether address: 00:00:00:00:00:00
Target IP address: 170.210.122.104
Padding: 0x20202020202020202020202020202020202000

Trama 2:

Ethernet (1013902103.199967)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:91
Type / Length: 0x806 (ARP)
Media length: 60

ARP Header

Hardware type: 0x1 (Ethernet)
Protocol type: 0x800 (IP)
Hardware length: 6
Protocol length: 4
Opcode: 2 (reply)
Sender Ether address: 00:00:21:6c:5a:7f
Sender IP address: 170.210.122.104
Target Ether address: 00:80:ad:c8:19:91
Target IP address: 170.210.122.101
Padding: 0xc5760100000100000000000047063313200

Trama 3:

Ethernet (1013902103.200136)

Hardware source: 00:80:ad:c8:19:91
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 44
Identification: 11285
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 33860
Source address: 170.210.122.101
Destination address: 170.210.122.104

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 388762
Acknowledgement number: 0
Header length: 6
Unused: 0
Flags: S
Window size: 8192
Checksum: 14614
Urgent: 0
Option: 2 (maximum segment size)
Length: 4
MSS: 1460

HTTP

Trama 4:

Ethernet (1013902103.200635)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: ff:ff:ff:ff:ff:ff
Type / Length: 0x806 (ARP)
Media length: 60

ARP Header

Hardware type: 0x1 (Ethernet)
Protocol type: 0x800 (IP)
Hardware length: 6
Protocol length: 4
Opcode: 1 (request)
Sender Ether address: 00:00:21:6c:5a:7f
Sender IP address: 192.168.0.104
Target Ether address: 00:00:00:00:00:00
Target IP address: 192.168.0.108
Padding: 0xc576010000010000000000000047063313200

Trama 5:

Ethernet (1013902103.201081)

Hardware source: 00:80:ad:c8:19:74
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x806 (ARP)
Media length: 42

ARP Header

Hardware type: 0x1 (Ethernet)
Protocol type: 0x800 (IP)
Hardware length: 6
Protocol length: 4
Opcode: 2 (reply)
Sender Ether address: 00:80:ad:c8:19:74
Sender IP address: 192.168.0.108
Target Ether address: 00:00:21:6c:5a:7f
Target IP address: 192.168.0.104

Trama 6:

Ethernet (1013902103.201435)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:74
Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 44
Identification: 11285
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 59498
Source address: 170.210.122.101
Destination address: 192.168.0.108

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 388762
Acknowledgement number: 0
Header length: 6
Unused: 0
Flags: S
Window size: 8192
Checksum: 40252
Urgent: 0
Option: 2 (maximum segment size)
Length: 4
MSS: 1460

HTTP

Trama 7:

Ethernet (1013902103.205322)

Hardware source: 00:80:ad:c8:19:74
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 58

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 44
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 64
Protocol: 6 (TCP)
Header checksum: 21632
Source address: 192.168.0.108
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927351
Acknowledgement number: 388763
Header length: 6
Unused: 0
Flags: SA
Window size: 5840
Checksum: 49836
Urgent: 0
Option: 2 (maximum segment size)
Length: 4
MSS: 1460

Trama 8:

Ethernet (1013902103.208711)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:91
Type / Length: 0x800 (IP)

Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 44
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 63
Protocol: 6 (TCP)
Header checksum: 61785
Source address: 170.210.122.104
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927351
Acknowledgement number: 388763
Header length: 6
Unused: 0
Flags: SA
Window size: 5840
Checksum: 24198
Urgent: 0
Option: 2 (maximum segment size)
Length: 4
MSS: 1460

HTTP

Trama 9:

Ethernet (1013902103.209159)

Hardware source: 00:80:ad:c8:19:91
Hardware destination: 00:00:21:6c:5a:7f

Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 11541
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 33608
Source address: 170.210.122.101
Destination address: 170.210.122.104

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 388763
Acknowledgement number: 1412927352
Header length: 5
Unused: 0
Flags: A
Window size: 8760
Checksum: 27355
Urgent: 0

HTTP

Trama 10:
Ethernet (1013902103.209393)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:74
Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 11541
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 59246
Source address: 170.210.122.101
Destination address: 192.168.0.108

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 388763
Acknowledgement number: 1412927352
Header length: 5
Unused: 0
Flags: A
Window size: 8760
Checksum: 52993
Urgent: 0

HTTP

Trama 11:

Ethernet (1013902103.258309)

Hardware source: 00:80:ad:c8:19:91
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 390

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 376
Identification: 11797
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 33016
Source address: 170.210.122.101
Destination address: 170.210.122.104

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 388763
Acknowledgement number: 1412927352
Header length: 5
Unused: 0
Flags: PA
Window size: 8760
Checksum: 43972
Urgent: 0

HTTP

Header: GET /index.html HTTP/1.0
Header: If-Modified-Since: Sat, 26 Jan 2002 00:33:49 GMT; length=485
Header: Connection: Keep-Alive
Header: User-Agent: Mozilla/4.76 [en] (WinNT; U)
Header: Host: 170.210.122.104
Header: Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Header: Accept-Encoding: gzip
Header: Accept-Language: en

Header: Accept-Charset: iso-8859-1,*,utf-8

Trama 12:

Ethernet (1013902103.258859)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:74
Type / Length: 0x800 (IP)
Media length: 390

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 376
Identification: 11797
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 58654
Source address: 170.210.122.101
Destination address: 192.168.0.108

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 388763
Acknowledgement number: 1412927352
Header length: 5
Unused: 0
Flags: PA
Window size: 8760
Checksum: 4075
Urgent: 0

HTTP

Header: GET /index.html HTTP/1.0
Header: If-Modified-Since: Sat, 26 Jan 2002 00:33:49 GMT; length=485

Header: Connection: Keep-Alive
Header: User-Agent: Mozilla/4.76 [en] (WinNT; U)
Header: Host: 170.210.122.104
Header: Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Header: Accept-Encoding: gzip
Header: Accept-Language: en
Header: Accept-Charset: iso-8859-1,*,utf-8

Trama 13:

Ethernet (1013902103.259269)

Hardware source: 00:80:ad:c8:19:74
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 54

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 64
Protocol: 6 (TCP)
Header checksum: 21636
Source address: 192.168.0.108
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927352
Acknowledgement number: 389099
Header length: 5

Unused: 0
Flags: A
Window size: 6432
Checksum: 54985
Urgent: 0

Trama 14:

Ethernet (1013902103.259698)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:91
Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 63
Protocol: 6 (TCP)
Header checksum: 61789
Source address: 170.210.122.104
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927352
Acknowledgement number: 389099
Header length: 5
Unused: 0
Flags: A
Window size: 6432
Checksum: 29347
Urgent: 0

HTTP

Trama 15:

Ethernet (1013902103.449884)

Hardware source: 00:80:ad:c8:19:74
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 683

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 669
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 64
Protocol: 6 (TCP)
Header checksum: 21007
Source address: 192.168.0.108
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927352
Acknowledgement number: 389099
Header length: 5
Unused: 0
Flags: PA
Window size: 6432
Checksum: 39864
Urgent: 0

HTTP

Header: HTTP/1.1 200 OK
Header: Date: Sat, 16 Feb 2002 23:28:23 GMT
Header: Server: Apache/1.3.17 (Unix) (SuSE/Linux)
Header: Connection: close
Header: Content-Type: text/html

Trama 16:

Ethernet (1013902103.451788)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:91
Type / Length: 0x800 (IP)
Media length: 683

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 669
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 63
Protocol: 6 (TCP)
Header checksum: 61160
Source address: 170.210.122.104
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927352
Acknowledgement number: 389099
Header length: 5
Unused: 0
Flags: PA
Window size: 6432
Checksum: 14226
Urgent: 0

HTTP

Header: HTTP/1.1 200 OK
Header: Date: Sat, 16 Feb 2002 23:28:23 GMT
Header: Server: Apache/1.3.17 (Unix) (SuSE/Linux)
Header: Connection: close
Header: Content-Type: text/html

Trama 17:

Ethernet (1013902103.483036)

Hardware source: 00:80:ad:c8:19:74
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 54

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 64
Protocol: 6 (TCP)
Header checksum: 21636
Source address: 192.168.0.108
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927981
Acknowledgement number: 389099
Header length: 5
Unused: 0
Flags: FA
Window size: 6432
Checksum: 54355

Urgent: 0

Trama 18:

Ethernet (1013902103.483491)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:91
Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 63
Protocol: 6 (TCP)
Header checksum: 61789
Source address: 170.210.122.104
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927981
Acknowledgement number: 389099
Header length: 5
Unused: 0
Flags: FA
Window size: 6432
Checksum: 28717
Urgent: 0

HTTP

Trama 19:

Ethernet (1013902103.484057)

Hardware source: 00:80:ad:c8:19:91
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 12053
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 33096
Source address: 170.210.122.101
Destination address: 170.210.122.104

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 389099
Acknowledgement number: 1412927982
Header length: 5
Unused: 0
Flags: A
Window size: 8131
Checksum: 27018
Urgent: 0

HTTP

Trama 20:

Ethernet (1013902103.484294)

Hardware source: 00:00:21:6c:5a:7f

Hardware destination: 00:80:ad:c8:19:74
Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 12053
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 58734
Source address: 170.210.122.101
Destination address: 192.168.0.108

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 389099
Acknowledgement number: 1412927982
Header length: 5
Unused: 0
Flags: A
Window size: 8131
Checksum: 52656
Urgent: 0

HTTP

Trama 21:

Ethernet (1013902104.041686)

Hardware source: 00:80:ad:c8:19:91
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 60

```
-------------------------------------------------------------------------------
                    IP Header
-------------------------------------------------------------------------------
Version:              4
Header length:        5 (20 bytes)
TOS:                  0x00
Total length:         40
Identification:       12309
Fragmentation offset: 0
Unused bit:           0
Don't fragment bit:   1
More fragments bit:   0
Time to live:         128
Protocol:             6 (TCP)
Header checksum:      32840
Source address:       170.210.122.101
Destination address:  170.210.122.104


-------------------------------------------------------------------------------
                    TCP Header
-------------------------------------------------------------------------------
Source port:          1454 (unknown)
Destination port:     80 (http)
Sequence number:      389099
Acknowledgement number: 1412927982
Header length:        5
Unused:               0
Flags:                FA
Window size:          8131
Checksum:             27017
Urgent:               0
-------------------------------------------------------------------------------
                    HTTP
-------------------------------------------------------------------------------

Trama 22:
                    Ethernet (1013902104.041894)
-------------------------------------------------------------------------------
Hardware source:      00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:74
Type / Length:        0x800 (IP)
Media length:         60
-------------------------------------------------------------------------------
```

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 12309
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 128
Protocol: 6 (TCP)
Header checksum: 58478
Source address: 170.210.122.101
Destination address: 192.168.0.108

TCP Header

Source port: 1454 (unknown)
Destination port: 80 (http)
Sequence number: 389099
Acknowledgement number: 1412927982
Header length: 5
Unused: 0
Flags: FA
Window size: 8131
Checksum: 52655
Urgent: 0

HTTP

Trama 23:

Ethernet (1013902104.042630)

Hardware source: 00:80:ad:c8:19:74
Hardware destination: 00:00:21:6c:5a:7f
Type / Length: 0x800 (IP)
Media length: 54

IP Header

Version: 4
Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 64
Protocol: 6 (TCP)
Header checksum: 21636
Source address: 192.168.0.108
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927982
Acknowledgement number: 389100
Header length: 5
Unused: 0
Flags: A
Window size: 6432
Checksum: 54354
Urgent: 0

Trama 24:

Ethernet (1013902104.043129)

Hardware source: 00:00:21:6c:5a:7f
Hardware destination: 00:80:ad:c8:19:91
Type / Length: 0x800 (IP)
Media length: 60

IP Header

Version: 4

Header length: 5 (20 bytes)
TOS: 0x00
Total length: 40
Identification: 0
Fragmentation offset: 0
Unused bit: 0
Don't fragment bit: 1
More fragments bit: 0
Time to live: 63
Protocol: 6 (TCP)
Header checksum: 61789
Source address: 170.210.122.104
Destination address: 170.210.122.101

TCP Header

Source port: 80 (http)
Destination port: 1454 (unknown)
Sequence number: 1412927982
Acknowledgement number: 389100
Header length: 5
Unused: 0
Flags: A
Window size: 6432
Checksum: 28716
Urgent: 0

HTTP
