Routers Virtuales para Aplicaciones Cloud

Sergio de Miguel Novalbos
Daniel Expósito Romero
Álvaro De Francisco Rus

Director de proyecto
Rafael Moreno Vozmediano

Proyecto de Sistemas Informáticos
Universidad Complutense de Madrid
Curso 2011-2012

Índice

1.- Cesión de derechos

Los alumnos Sergio de Miguel Novalbos, Daniel Expósito Romero, Álvaro De Francisco Rus, aquí firmantes, autorizan a la Universidad Complutense de Madrid a la difusión de la memoria, implementación, y código del proyecto realizado con fines exclusivamente académicos y mencionando expresamente a los autores del mismo.

Sergio de Miguel Novalbos

Daniel Expósito Romero

Álvaro De Francisco Rus

2.- Resumen

El *Cloud Computing* consiste en proporcionar como servicios a través de Internet todo lo que un sistema informático puede ofrecer. Es una plataforma altamente escalable que promete un acceso rápido al recurso hardware o software y donde el usuario no necesita ser experto para su manejo y acceso. Las nubes suelen apoyarse en tecnologías como la virtualización, técnicas de programación como el multitenancy o habilidades para la escalabilidad, balanceo de carga y rendimiento óptimo, para conseguir ofrecer el recurso de una manera rápida y sencilla.

Nuestro trabajo gira en torno a la virtualización. Podríamos definirla como la creación a través de software de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red. Entre las ventajas que la virtualización nos ofrece están el ahorro de costes, la posibilidad de crear entornos de pruebas, entornos aislados seguros u olvidarnos de los problemas de compatibilidad entre programas.

Tomando como base estos conceptos, nuestro proyecto tiene como objetivo la contextualización de imágenes en entornos cloud, en otras palabras, configurar imágenes precargadas en entornos virtuales de forma automática. De esta forma ahorramos costes al tener en la nube imágenes más simples o con una configuración básica, las cuales serán más baratas que las que ofrezcan un mayor nivel de prestaciones.

Palabras clave: Cloud Computing, OpenNebula, virtualización, servicio, appliance

3.- Abstract

Cloud Computing consists in providing as services through Internet all that a computer system can offer. It is a highly scalable platform that promises quick access to hardware or software resource and where the user does not need being an expert for management and access. Clouds tend to rely on technologies such as virtualization, programming techniques as multitenancy or skills for scalability, load balancing and optimum performance, towards delivering the resource quickly and easily.

Our project revolves around virtualization. We can define it as the creation through software of a virtual version of some technological resource, like a hardware platform, an operative system, a storage device or other network resources. Among the advantages of virtualization we can stress the costs reduction, the possibility to create test environments, safe isolated environments or the lack of compatibility problems with the programs.

Based on these concepts, the aim of our project is the contextualization of virtual images in cloud environments, in other words, the configuration of existing images in virtual scenes automatically. This way, we can save costs because we have simpler images in the cloud or with a basic configuration, which will be cheaper than others with more features.

4.- Introducción y Objetivos

Este proyecto de Sistemas Informático se basa en la contextualización o post-configuración de imágenes en entornos cloud. En otras palabras, configurar imágenes precargadas en entornos virtuales con el fin de evitar tener cargadas en el servidor diferentes imágenes con las mismas características (appliances). Este proyecto se encuadra en el área del Cloud Computing. Una primera definición podría ser la tendencia a basar las aplicaciones en servicios alojados de forma externa, en la propia web. Se tratará un capítulo a este tema.

Una de las bases, se trata en tener imágenes limpias, sin ninguna configuración previa. Si se necesita realizar una aplicación para un fin en concreto, se tendrá un script de configuración que se cargará en las imágenes en blanco y se obtendrá el resultado esperado. Son ventajas tanto a nivel de hardware, puesto que nos evita tener en nuestro ordenador un alto número de programas, como a nivel de software, ya que sólo se tendrá que tener en cuenta el script. Hay programas avanzados que se dedican a dado dicho script dejarte corriendo la aplicación en tu ordenador. Nos referimos a Open Nebula.

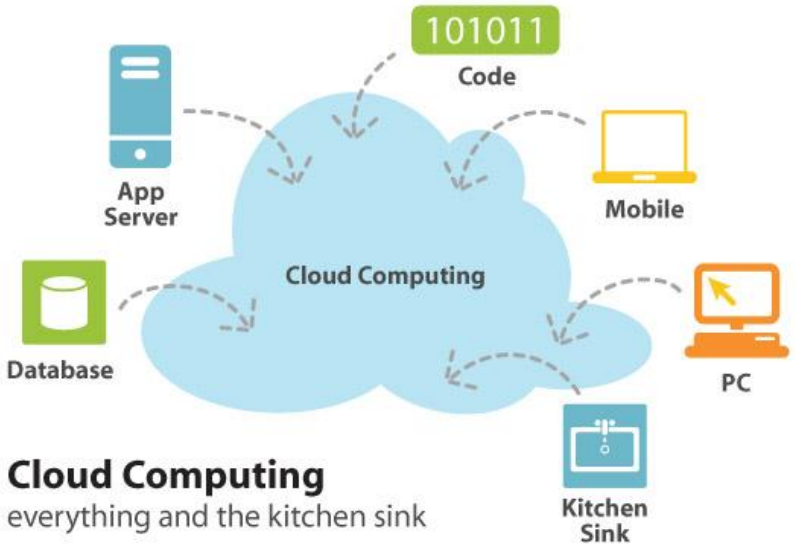
4.1.- Introducción computación en nube

La "**Computación en Nube**" o "**Cloud Computing**" se basa principalmente en aprovechar todo el potencial que ofrece la ilimitada capacidad del espacio que poseen centros de datos como Amazon y Google. Esto es, dar prioridad a los tipos de datos que se encuentran en constante dinamismo.

Actualmente muchas empresas están realizando aplicaciones capaces de utilizarse enteramente en "La Nube", sin la más mínima necesidad de estar instalados.

Entre las principales **ventajas**, podemos considerar que no se requieren múltiples licencias, no se requiere sistema operativo o plataforma específica, puesto que es multiplataforma, no se requiere dispositivos de almacenamiento secundario de gran espacio y disponemos de acceso inmediato a los servicios.

A su vez, indicamos los **inconvenientes** que se nos pueden presentar al trabajar en "La Nube". Nuestros datos están en manos de terceros, es decir, tenemos una pérdida de control de los mismos. Una opción para evitarlo es mantener los datos más privados en servidores propios y el resto subirlos a la nube. Por otro lado, no disponemos de la posibilidad de acceso al código fuente y por último, dependemos completamente de Internet, puesto que no siempre se puede trabajar *offline*.



Establecer un sistema para crear imágenes virtuales personalizadas en la nube

Las empresas están recurriendo a imágenes virtuales como un medio para mejorar la prestación de software dentro de sus centros de datos. A medida que se hagan un mayor uso de imágenes virtuales, los desafíos en este ámbito serán cada vez mayores. Como ejemplo la cantidad de contenido para poner en una sola imagen y la mejor forma de construirlas, son procesos que se están desarrollando cada vez con más fuerza.

El principal desafío en el uso de imágenes virtuales es mantener un equilibrio entre el grado de usabilidad y tener presente el valor del usuario. Para enfrentar este reto, se debe adoptar un enfoque medido para la construcción de sus imágenes virtuales. Para habilitar este tipo de enfoque, tendremos que tener en cuenta dos aspectos importantes:

- Tipo de contenido a instalar

  En general, el contenido que se divide en las siguientes categorías es un buen candidato para la instalación directa en las imágenes:

  - Componentes de gran tamaño, tales como el sistema operativo u otro software de tamaño considerable. La no instalación de este tipo de componentes, ralentizaría el proceso de creación de la imagen.

  - Los componentes más comunes. Se definen programas que van a ser utilizados por todos los usuarios, tales como software anti-virus, herramientas *office*, agentes de monitorización. No sólo ahorrará tiempo al usuario, sino que también disminuye la probabilidad de que acaban con los entornos no estándares.

  - Aplicaciones que implican un alto tiempo de instalación. Esto permite a los usuarios saltarse estas acciones y ponerse a trabajar rápidamente cuando se pone en marcha la imagen.

- Manejo del contenido requerido

  Estas tres categorías representan sólo un subconjunto de las necesidades generales de los usuarios de imágenes virtuales. Pero no se refieren al aspecto de que los usuarios tienen que realizar configuraciones específicas para sus ámbitos de uso.

  De este concepto, surgen dos opciones, crear imágenes estáticas, en las cuales tú configuras otros aspectos a través de scripts o imágenes dinámicas. Mucho más aconsejables.

  El fundamento de la construcción de las máquinas virtuales de manera dinámica es la inclusión de un conjunto especial de guiones de activación. Estas secuencias de comandos realizan las actividades de configuración que tienen que ocurrir para cada despliegue de la imagen. Esto puede incluir la actualización de la dirección IP, la instalación de contenido que es de tamaño pequeño, la realización de acciones de configuración en el software previamente instalado, o cualquier otra acción.

La metodología que se propone se basa en determinar qué contenido debe ser instalado directamente en la imagen virtual y cuál no, además de que se ofrecen maneras de configurar de forma dinámica el contenido instalado en las imágenes virtuales. En esencia, se habla de cómo construir una imagen a través de la nube a medida que es altamente reutilizable, que proporciona la mayor cantidad de usabilidad con la menor cantidad de trabajo por parte del usuario.

Esto permite que una imagen virtual única pueda asumir muchas formas diferentes durante el proceso de activación, con el fin de satisfacer una amplia variedad de escenarios de uso.

Parametrizar imágenes en la nube de casos personalizados

Existen dos maneras de crear una imagen personalizada en la nube. La forma estática consiste en crear una instancia, posteriormente realizar la configuración propia, y luego tomar la imagen de esta instancia personalizada. La forma dinámica, en la cual se pone en marcha la transferencia de información para el proceso de creación de la instancia con el fin de crear una imagen personalizada final, implica un beneficio clave del cloud computing: la capacidad de crear una instancia a medida sobre la marcha.

En general, los parámetros son marcadores de posición para los valores reales o tipos. El concepto de parametrización es el proceso de decidir y definir los parámetros necesarios para una especificación completa o relevante.

Se trata este tema de manera más teórica, puesto que la puesta en marcha de un ejemplo de parametrización saldría fuera de este proyecto.

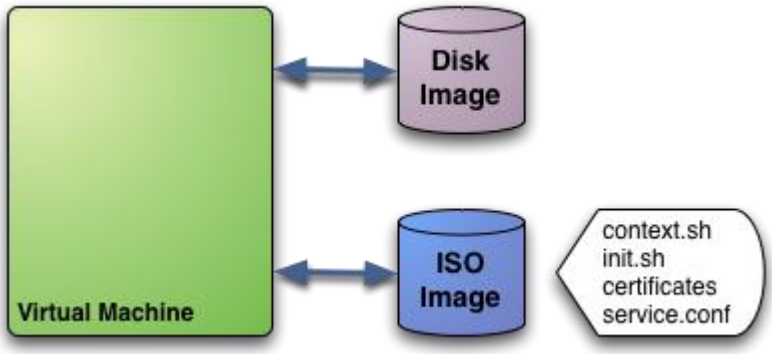
4.2.- Open Nebula

OpenNebula es un software *open-source* que permite construir diferentes tipos de cloud. Ha sido diseñado para ser integrado con cualquier tipo de red y almacenamiento, para así adaptarse a los centros de datos existentes.

OpenNebula gestiona el almacenamiento, las redes y las tecnologías de virtualización. Proporciona la posibilidad de desplegar servicios en infraestructuras distribuidas, combinando recursos de centros de datos así como de clouds remotos, de acuerdo con las políticas de despliegue.

El método usado es dar una imagen ISO para configurar los parámetros necesarios a una nueva imagen iniciada. El método que se utiliza hace posible configurar también interfaces de redes.

En el archivo de descripción de la máquina virtual se puede especificar el contenido del fichero ISO (archivos y directorios). El dispositivo de la imagen ISO será accesible y se podrán especificar los parámetros de configuración que se escribirán en un archivo para su uso posterior en la máquina virtual.



En este ejemplo, se puede contemplar, una máquina virtual con dos discos asociados. La imagen del disco contiene el fichero de sistema desde el cual el sistema operativo correrá. En la imagen ISO, se define la contextualización para esta máquina virtual.

Siguiendo con el ejemplo, se deja como posible ampliación la integración del proyecto para Open Nebula. Sería suficiente con adaptar los ficheros de configuración, según los archivos que se ven en la imagen ISO.

5.- Router

Router es un término inglés que puede traducirse como enrutador, ruteador o direccionador. Se trata de un dispositivo de hardware que interconecta segmentos de red o redes enteras. Hace pasar paquetes de datos entre redes tomando como base la información de la capa de red.

El router toma decisiones (basado en diversos parámetros) con respecto a la mejor ruta para el envío de datos a través de una red interconectada y luego redirige los paquetes hacia el segmento y el puerto de salida adecuados.

Como hemos mencionado anteriormente, opera en la capa tres del modelo OSI, es decir, en la capa de red. El objetivo de esta capa es hacer que los datos lleguen desde el origen al destino, aun cuando ambos no estén conectados directamente. En este último caso, necesitamos la presencia de routers para poder llevar a cabo esta operación.

El router tiene múltiples usos más o menos complejos. En su uso más común, un enrutador permite que en una casa u oficina pequeña, varios ordenadores aprovechen la misma conexión a Internet. En este sentido, el router opera como receptor de la conexión de red para encargarse de distribuirlo a todos los equipos conectados al mismo. Así, se conecta una red o Internet con otra de área local.

Si hablamos de un router wi-fi o inalámbrico, significa que utiliza el espacio u ondas de radio para transmitir la información desde y hacia los ordenadores conectadas al mismo; en otras palabras, el router wi-fi permite conectar a los ordenadores sin cables. Obviamente los ordenadores deben tener sus respectivas tarjetas de red wi-fi.

Los router wi-fi físicamente se distinguen de los que usan cables por sus antenitas, que se utilizan para recibir y enviar información.

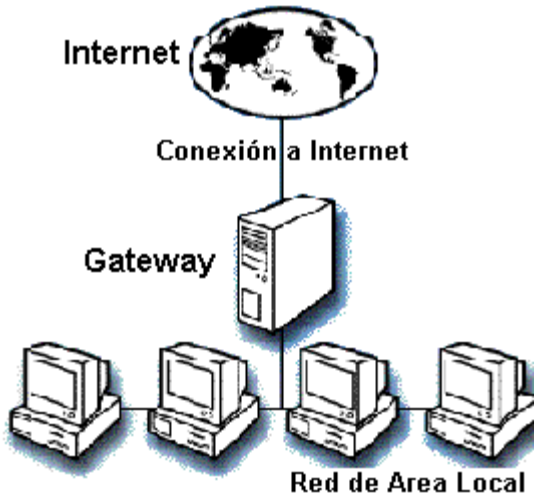
5.1.- Tipos

En la actualidad existen distintos tipos de router que pueden utilizarse para muy diversas cuestiones, bien sean proporcionar conectividad dentro de las empresas, entre Internet y las propias empresas en sí, o los ISP (Internet Services Provider, Proveedores de Acceso a Internet) en nuestra casa.

El router puede ser un ordenador convencional, con una aplicación corriendo en él o, más habitualmente, tratarse de un equipamiento específicamente diseñado para estas funciones. En líneas generales podemos distinguir 2 clases de routers en función del tráfico gestionado:

- Routers de Red Núcleo (Core Routers): se trata de equipamiento de interconexión que constituye la red de datos de los proveedores de Internet de grandes corporaciones.

- Routers de Salida (Gateway o pasarela): es el equipo con el que se realiza la conexión a Internet o a otra subred.



Tipos de router domésticos

- Router Neutros

  Suelen emplearse en conexiones de cable modem, no disponen de módem xDSL integrado por lo tanto si lo usamos para acceder a internet con una conexión xDSL, necesitaremos el modem router xDSL para tener una conexión a internet. El Router neutro dispone de una conexión WAN que usa un conector Ethernet RJ-45 para conectar el Router neutro al Modem-cable/Router xDSL, y de varios conectores Ethernet RJ-45 que sirven para dar conexión a los equipos.

- Módem Router xDSL

  Se emplean en conexiones de banda ancha xDSL como ADSL, ADSL2 y ADSL2+, es decir que con un único router tenemos acceso a la red local y a internet. El Router xDSL dispone de una conexión xDSL que usa un conector telefónico RJ-11 para conectar el Router xDSL a la roseta del teléfono, y varios conectores Ethernet RJ-45 que sirven para dar conexión a los equipos.

  Los router xDSL se pueden configurar como:

  - Monopuesto: Independientemente de que el router sea monopuerto o multipuerto, el router actúa como un módem con todos los puertos abiertos. En caso de tener un sólo equipo puede ser una buena opción pero necesitamos un firewall (cortafuegos) por software para aumentar el nivel de seguridad del equipo ya que al tener todos los puertos abiertos el equipo es menos seguro.

  - Multipuesto: Independientemente de que conectemos uno o varios equipos, en el caso de la configuración multipuesto el router actúa de filtro (si queremos mayor seguridad podemos poner un firewall por software sobre todo para controlar la salida

de datos ya que la entrada la controlaría el router), en este caso es necesario abrir los puertos del router. Para ello suele emplearse NAT (se explicará más adelante).

Otra forma de clasificar los routers domésticos según su fisonomía puede ser:

- Monopuerto (Router xDSL)

  Tienen un solo puerto Ethernet (normalmente suele ser de 10 ó 100 Mbps (Fast Ethernet), no tienen Wifi) y en algunos casos pueden incluso funcionar por USB. Algo poco aconsejable salvo que no haya otra opción, normalmente son los modelos más sencillos y que suelen "regalar" los ISP al darte de alta en una conexión xDSL. En algunos casos estos router pueden ser compatibles con la modalidad Multipuesto (acceso para varios equipos) pero necesitan un Hub (Concentrador) o Switch (Conmutador) para aumentar el número de conexiones Ethernet o para poder tener un punto de acceso Wifi, lo cual encarece el precio final del equipo de red y además aumenta el número de conexiones eléctricas. Los Router monopuerto son útiles para usuarios que tengan un único equipo ya que si necesitas más de un toma Ethernet o conexión Wifi es mejor opción un router Multipuerto al ser más compacto y utilizar un sólo enchufe eléctrico.

- Multipuerto (Router Neutros y xDSL)

  Tienen varios puertos Ethernet (Generalmente 4 Fast Ethernet de 100 Mbps, aunque actualmente en el mercado existen algunos modelos Gigabit Ethernet de hasta 1.000 Mbps), y pueden llevar opcionalmente una conexión Wifi que habitualmente suele ser 11g (hasta 54 Mbps), generalmente estos router también los ofrecen los ISP pero no suelen regalarlos sino que tienen un sobrecoste.

  Actualmente no es habitual que los ISP vendan a los usuarios router multipuerto con Gigabit Ethernet (hasta 1.000 Mbps) y Wifi 11n (hasta 300 Mbps), por lo que si necesitamos un router de este tipo no queda más remedio que adquirirlo por cuenta propia del usuario pagando el coste del mismo.

5.2.- Servicios que ofrece un router

En primer lugar, vamos a realizar un pequeño tour por aquellos servicios más significativos a la hora de configurar un router. En la siguiente sección, se verán en profundidad aquellos que han sido configurados en este proyecto.

Si hablamos de servicios para configurar un router nos encontramos con una amplia variedad: DNS, DDNS, NAT, VPN, DMZ, ip forwarding, DHCP, port forwarding. Cada uno con una finalidad concreta, veamos de que se encargan:

- **DNS**: asocia una dirección URL a una dirección IP. Nace de la necesidad de recordar fácilmente los nombres de todos los servidores conectados a Internet.

- **DDNS**: es una variación del anterior. DNS dinámico es un sistema que permite la actualización en tiempo real de la información sobre nombres de dominio situada en un servidor de nombres. El uso más común que se le da es permitir la asignación de

un nombre de dominio de Internet a un ordenador con dirección IP variable (dinámica). Esto permite conectarse con la máquina en cuestión sin necesidad de tener que rastrear las direcciones IP.

- **NAT**: la idea básica que hay detrás de NAT es traducir las IP privadas de la red en una IP publica para que la red pueda enviar paquetes al exterior; y traducir luego esa IP publica, de nuevo a la IP privada del ordenador que envió el paquete, para que pueda recibirlo una vez llega la respuesta.

- **VPN**: tecnología de red que brinda la posibilidad de conectarse a una red pública generando una extensión a nivel de área local. Por ejemplo, este tipo de redes se utilizan a la hora de conectar dos o más oficinas de una empresa a través de Internet. Esto facilita la conexión y el intercambio, y permite que miembros de un mismo equipo se conecten entre sí desde locaciones remotas.

- **DMZ**: Demilitarized zone o Zona DesMilitarizada. Es una red local que se ubica entre la red interna de una organización y una red externa, generalmente Internet. El objetivo de una DMZ es que las conexiones desde la red interna y la externa a la DMZ estén permitidas, mientras que las conexiones desde la DMZ sólo se permitan a la red externa, los equipos en la DMZ no pueden conectar con la red interna. Esto permite que los hosts de la DMZ puedan dar servicios a la red externa a la vez que protegen la red interna en el caso de que intrusos comprometan la seguridad de los equipos situados en la zona desmilitarizada.

- **IP forwarding**: se encarga de recepción y el envío de paquetes IP. Su función principal es la de retransmisión de los paquetes que se reciben por una interfaz física y de retransmitirlos por otra interfaz hacia otro nodo.

- **DHCP**: permite asignar dinámicamente una dirección IP a un cliente a partir de la base de datos de direcciones IP del servidor DHCP de la red local. Esto es debido a que cada equipo de una red TCP/IP debe tener una dirección IP única. La dirección IP (junto con su máscara de subred relacionada) identifica al equipo host y a la subred a la que está conectado. Al mover un equipo a una subred diferente se debe cambiar la dirección IP. Es por este motivo que es necesario el servicio DHCP.

- **Port forwarding**: se define como la asignación o reenvío de puertos para transmitir información a través de una red. Esta técnica utiliza el protocolo TCP/IP, y se encarga de transmitir paquetes de información entre servidores externos a los servidores internos de una red particular. Port Forwarding se utiliza, por lo general, cuando el equipo que recibirá los paquetes de información se encuentra dentro del radio de acción de un router, permitiendo comunicación entre usuarios externos a una red LAN, con usuarios privados de la red local.

Una vez, que tenemos una visión de los principales servicios de un router, centrémonos más en profundidad en aquellos que han sido configurados en el proyecto: DNS, NAT, ip forwarding, port forwarding y DHCP.

6.- Servicios

6.1.- DNS

DNS es una abreviatura para Sistema de nombres de dominio (en inglés Domain Name System). Dicho así, no nos dice nada significativo, por lo que se expresará en profundidad.

Una URL (dirección para acceder a una página web) está compuesta por palabras separadas por puntos (ej: www.fdi.ucm.es). Para acceder a la misma, sólo se debe recordar estas palabras. Esta dirección URL está asociada a un número (dirección IP) que identifica el servidor que se ha de contactar para verla (ej: 147.96.85.71). El servicio DNS se encarga de asociar una dirección URL a una dirección IP.

La siguiente ilustración muestra un uso básico de DNS, consistente en la búsqueda de la dirección IP de un equipo basada en su nombre.

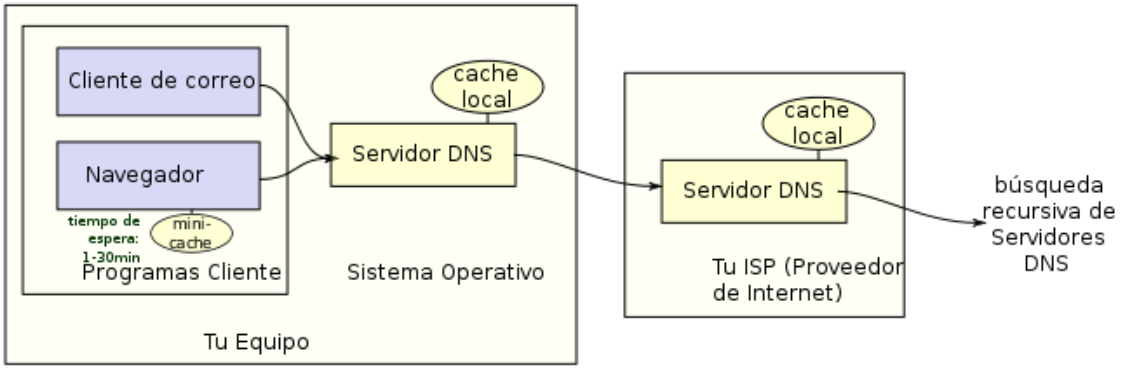
En este ejemplo, un equipo cliente consulta a un servidor DNS, preguntando la dirección IP de un equipo configurado para utilizar “host-a.ejemplo.microsoft.com” como nombre de dominio. Como el servidor puede utilizar la base de datos local para responder la consulta, contesta con una respuesta que contiene la información solicitada, un registro de recursos de host que contiene la información de dirección IP para host-a.ejemplo.microsoft.com.

El ejemplo muestra una consulta DNS sencilla entre un único cliente y un servidor DNS. En la práctica, las consultas DNS pueden ser más complicadas que ésta e incluyen pasos adicionales que no se muestran aquí.

Los usuarios generalmente no se comunican directamente con el servidor DNS: la resolución de nombres se hace de forma transparente por las aplicaciones del cliente (por ejemplo, navegadores, clientes de correo y otras aplicaciones que usan Internet). Al realizar una petición que requiere una búsqueda de DNS, la petición se envía al servidor DNS local del sistema operativo. El sistema operativo, antes de establecer alguna comunicación, comprueba si la respuesta se encuentra en la memoria caché. En el caso de que no se encuentre, la petición se enviará a uno o más servidores DNS.

La mayoría de usuarios domésticos utilizan como servidor DNS el proporcionado por el proveedor de servicios de Internet. La dirección de estos servidores puede ser configurada de

forma manual o automática mediante DHCP. En otros casos, los administradores de red tienen configurados sus propios servidores DNS.



En cualquier caso, los servidores DNS que reciben la petición, buscan en primer lugar si disponen de la respuesta en la memoria caché. Si es así, sirven la respuesta; en caso contrario, iniciarían la búsqueda de manera recursiva. Una vez encontrada la respuesta, el servidor DNS guardará el resultado en su memoria caché para futuros usos y devuelve el resultado.

6.1.1.- Tipos de servidores DNS

Servidores primarios o maestros

Mantiene la base de datos con la información sobre la zona. Los cambios sobre la información del dominio se llevan a cabo en el servidor primario.

Servidores secundarios o esclavos

Poseen una copia de la base de datos del servidor primario. Proporciona redundancia frente a fallos. Permiten equilibrar la carga de la red, ya que pueden resolver nombres igual que los servidores primarios. Periódicamente se sincronizan con el servidor primario para tener siempre la información actualizada.

Servidores de sólo cacheo

No mantiene ninguna zona. Sólo almacena en su memoria temporal las consultas que recibe de los clientes, para utilizarlas en caso de una nueva consulta.

6.1.2.- Dnsmasq

El paquete Dnsmasq permite poner en marcha un servidor DNS de una forma muy sencilla. Simplemente instalando y arrancando el servicio Dnsmasq, sin realizar ningún tipo de configuración adicional, nuestro PC se convertirá en un servidor caché DNS. La resolución funcionará tanto en sentido directo como en sentido inverso, es decir, resolverá la IP dado un

nombre de PC y el nombre del PC dada la IP. Adicionalmente, Dnsmasq dispone de servidor DHCP y permite resolver los nombres de los PCs a los que les ha asignado dirección IP dinámica.

Las máquinas que son configuradas por el DHCP tienen sus nombres incluidos automáticamente en el DNS y los nombres pueden ser especificados por cada máquina o manualmente, asociando un nombre a un dirección MAC en el archivo de configuración del dnsmasq.

En el apartado de configuración veremos como poner en marcha, reiniciar o para el servicio Dnsmasq. Es necesario tener en cuenta este concepto, puesto que en el proyecto dentro del servicio DNS, trabajaremos concretamente con Dnsmasq.

6.2.- NAT

NAT que en inglés significa *Network Address Translation*, en español podría verse como “Traducción de Direcciones de Red”.

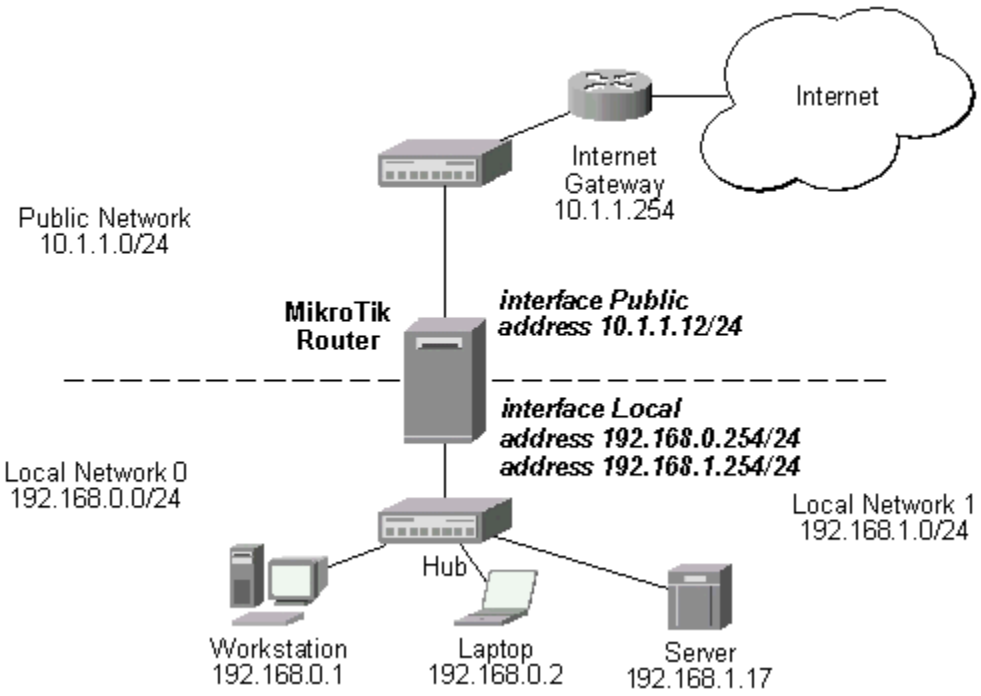
En primer lugar, vamos a ver las diferencias entre direcciones IP públicas y direcciones privadas. Esto nos hará entender mucho mejor el funcionamiento de NAT.

Las direcciones privadas son rangos especiales de direcciones IP que se reservan para ser utilizadas en redes locales, y se llaman privadas (o no-enrutables) porque no pueden ser utilizadas en Internet. Los routers intermedios que componen todo Internet, no “entienden” este tipo de direcciones y no las encaminan.

Esto no pasa con las direcciones públicas, que son las que se usan en Internet, y han de pertenecer a un único equipo (host). No puede haber varios con la misma IP pública.

Aquí surge el problema y la necesidad de NAT, puesto que cuando se envía un paquete a internet, si figurara nuestra dirección IP privada como destino, a la hora de devolvernos el paquete no sería posible, puesto que las direcciones privadas pueden repetirse en las redes locales.

La idea básica que hay detrás de NAT es traducir las IPs privadas de la red en una IP publica para que la red pueda enviar paquetes al exterior; y traducir luego esa IP publica, de nuevo a la IP privada del ordenador que envió el paquete, para que pueda recibirlo una vez llega la respuesta.



El mecanismo que utiliza NAT para las asociaciones entre IP pública e IP privada es una tabla (tabla de NAT) en la que guarda una entrada por cada conexión. Cuando un host de la red local inicia una conexión hacia el exterior, el software de NAT asigna una entrada en la tabla, para que a partir de ahora, todo lo que llegue perteneciente a esa conexión sepa traducirlo hacia la IP privada que inició la conexión.

Si queremos permitir conexiones desde el exterior a un PC de nuestra red local, hemos de añadir una entrada fija en la tabla de NAT, indicando que todo el tráfico que llegue que vaya a determinado puerto, sea dirigido al PC en cuestión. El puerto es el único elemento que tenemos para “distinguir” conexiones, ya que todo llegará a la IP del router, pero tendrán un puerto de destino según sea una conexión u otra.

NAT se utiliza debido al progresivo agotamiento de las direcciones IPv4. Se espera que con la llegada de IPv6 no sea necesario continuar con esta práctica.

6.3.- IP forwarding

La recepción y el envío de paquetes IP es una de las funciones más frecuentes en los sistemas de red y por ello es también uno de los benchmarks más utilizados para evaluar las prestaciones de los distintos procesadores de red.

Realizar el IP forwarding implica:

- Examinar la dirección de destino de los paquetes entrantes.

- Determinar la dirección del encaminador siguiente al que enviar.

- Enviar el paquete al siguiente encaminador.

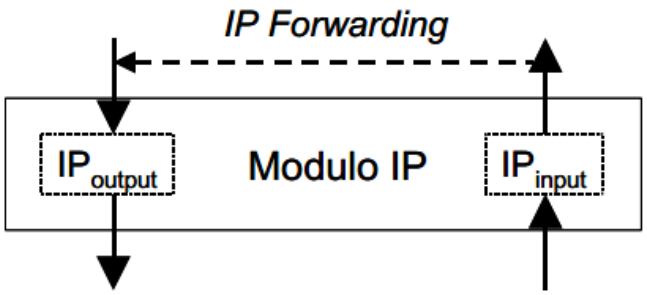
Para determinar la dirección del siguiente equipo al que enviar el paquete hay que consultar la Tabla de Enrutamiento almacenada. Esta tabla la crea y la mantiene el protocolos de encaminamiento (como por ejemplo el BGP, Border Gateway Protocol).

El mecanismo de IP forwarding se encarga de la retransmisión de los paquetes que se reciben por una interfaz física y de retransmitirlos por otra interfaz hacia otro nodo.

Cuando un paquete IP se recibe por una interfaz física, el módulo IP de entrada (IPinput) procesa el paquete. Si la dirección IP destino del paquete se corresponde con la del dispositivo se procesa el paquete y se pasa al modulo TCPinput.

En caso de que la dirección IP destino no se corresponda con la del dispositivo y el módulo IP forwarding está desactivado, el paquete IP se descarta.

En el caso de que el módulo IP forwarding esté activado, se pasa el paquete al módulo IP de salida (IPoutput), se consulta la tabla de encaminamiento y el paquete se retransmite por la interfaz correspondiente.



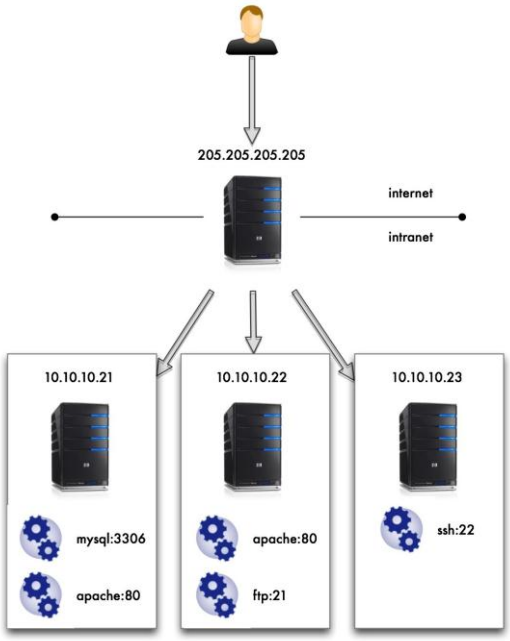
6.4.- Port forwarding

Podemos definir Port Forwarding, como la asignación o reenvío de puertos para transmitir información a través de una red. Esta técnica utiliza el protocolo TCP/IP, y se encarga de transmitir paquetes de información entre servidores externos a los servidores internos de una red particular.

El Port Forwarding se utiliza, por lo general, cuando el equipo que recibirá los paquetes de información se encuentra dentro del radio de acción de un Router, permitiendo comunicación entre usuarios externos a una red LAN, con usuarios privados de la red local.

Esta técnica permite conectar el ordenador con servicios externos que no son soportados por la red local. También, hace visible las direcciones IP de un equipo dentro de la

red a usuarios externos, direcciones que de manera predeterminada son inaccesibles a los usuarios externos.



Como podemos ver en la ilustración, un usuario quiere acceder a todos los servicios de las máquinas de la intranet 10.10.10.0. Sin embargo, sólo puede acceder a la dirección IP pública 205.205.205.205. Gracias a port-forwarding, vamos a poder tener acceso a las diferentes máquinas a través de los diferentes puertos.

6.5.- DHCP

El protocolo de configuración dinámica de host (Dynamic Host Configuration Protocol) es un estándar IP diseñado para simplificar la administración de la configuración IP del host. El estándar DHCP permite el uso de servidores DHCP para administrar la asignación dinámica a los clientes de la red, de direcciones IP y otros detalles de configuración relacionados.

Cada equipo de una red TCP/IP debe tener una dirección IP única. La dirección IP (junto con su máscara de subred relacionada) identifica al equipo host y a la subred a la que está conectado. Al mover un equipo a una subred diferente se debe cambiar la dirección IP. DHCP permite asignar dinámicamente una dirección IP a un cliente a partir de la base de datos de direcciones IP del servidor DHCP de la red local.

Las ventajas que nos ofrece DHCP son principalmente dos:

- **Configuración segura y confiable**.

  DHCP evita los errores de configuración que se producen por la necesidad de escribir los valores manualmente en cada equipo. Así mismo, DHCP ayuda a evitar los conflictos de direcciones que se producen al configurar un equipo nuevo en la red con una dirección IP ya asignada.

- **Reduce la administración de la configuración.**

  La utilización de servidores DHCP puede reducir significativamente el tiempo necesario para configurar y modificar la configuración de los equipos de la red. Los servidores se pueden configurar para que suministren un conjunto completo de valores de configuración adicionales al asignar concesiones de direcciones. Estos valores se asignan mediante opciones DHCP.

  Así mismo, el proceso de renovación de concesiones de DHCP ayuda a garantizar que en las situaciones en que sea necesario actualizar a menudo la configuración de los clientes (como en el caso de usuarios con equipos móviles o portátiles que cambian frecuentemente de ubicación), se comuniquen directamente con los servidores DHCP y puedan realizar estos cambios de forma eficaz y automática.

DHCP utiliza un modelo cliente-servidor:

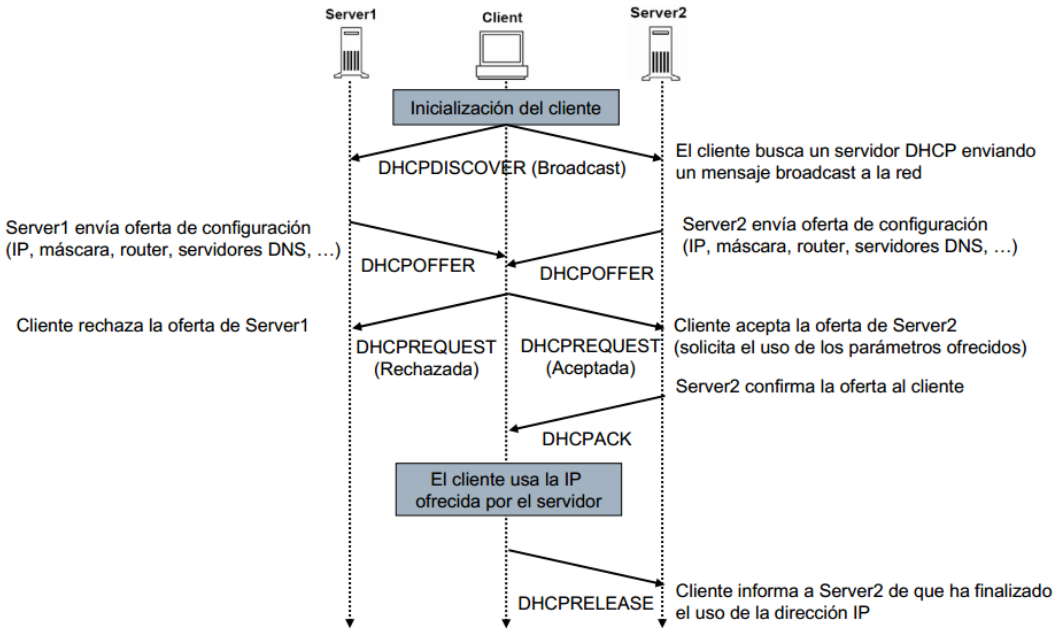
- **Clientes DHCP**

  No disponen de una configuración de red fija. Cuando arranca el sistema busca un servidor DHCP que le proporcione la información de configuración de red necesaria.

- **Servidor DHCP**

  Proporciona los parámetros de configuración de la red a los clientes que lo solicitan.

Cuando hay un servidor DHCP instalado y configurado en la red, los clientes habilitados para DHCP pueden obtener dinámicamente sus direcciones IP y los parámetros de configuración relacionados cada vez que inician una sesión y se unen a la red. Los servidores DHCP proporcionan esta configuración a los clientes que la solicitan, en forma de una oferta de concesión de direcciones.


Server1
Client
Server2
Inicialización del cliente
DHCPDISCOVER (Broadcast)
El cliente busca un servidor DHCP enviando un mensaje broadcast a la red
Server1 envía oferta de configuración (IP, máscara, router, servidores DNS, …)
Server2 envía oferta de configuración (IP, máscara, router, servidores DNS, …)
DHCPOFFER
DHCPOFFER
Cliente rechaza la oferta de Server1
Cliente acepta la oferta de Server2 (solicita el uso de los parámetros ofrecidos)
DHCPREQUEST (Rechazada)
DHCPREQUEST (Aceptada)
Server2 confirma la oferta al cliente
DHCPACK
El cliente usa la IP ofrecida por el servidor
DHCPRELEASE
Cliente informa a Server2 de que ha finalizado el uso de la dirección IP

7.- Tecnologías empleadas

7.1.- XML

XML son las siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades, de ahí que se le denomine metalenguaje. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Algunas de las ventajas de XML son:

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones.

- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.

- Transformamos datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de trozos de información.

Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las

etiquetas tienen la forma <nombre>, donde *nombre* es el nombre del elemento que se está señalando.

Partes de un documento XML

Un documento XML está formado por el prólogo y por el cuerpo del documento así como texto de etiquetas que contiene una gran variedad de efectos positivos o negativos en la referencia opcional a la que se refiere el documento, hay que tener mucho cuidado de esa parte de la gramática léxica para que se componga de manera uniforme.

Prólogo

Aunque no es obligatorio, los documentos XML pueden empezar con unas líneas que describen la versión XML, el tipo de documento y otras cosas.

El prólogo de un documento XML contiene:

- Una declaración XML. Es la sentencia que declara al documento como un documento XML.
- Una declaración de tipo de documento. Enlaza el documento con su DTD (definición de tipo de documento), o el DTD puede estar incluido en la propia declaración o ambas cosas al mismo tiempo.
- Uno o más comentarios e instrucciones de procesamiento.

Cuerpo

A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener solo un elemento raíz, característica indispensable también para que el documento esté bien formado. Sin embargo es necesaria la adquisición de datos para su buen funcionamiento.

Elementos

Los elementos XML pueden tener contenido (más elementos, caracteres o ambos), o bien ser elementos vacíos.

Atributos

Los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento. Deben ir entre comillas.

Entidades predefinidas

Entidades para representar caracteres especiales para que, de esta forma, no sean interpretadas como marcado en el procesador XML.

Como veremos en los siguientes apartados, XML está asociado a gran número de tecnologías, a las cuales ayuda a ser más esquemáticas y claras. Por otra parte nosotros nos hemos servido del estándar 1.0 para elaborar los archivos de configuración del router, los cuales almacenarán de forma permanente los parámetros y serán modificados por la interfaz. Su estructura se explica más adelante.

7.2.- XHTML

XHTML es un lenguaje similar a HTML, pero con algunas diferencias que lo hacen más robusto y aconsejable para la modelación de páginas web. Las siglas corresponden con las palabras inglesas eXtensible Hypertext Markup Language, que vendría a significar en castellano algo así como lenguaje extensible de marcación hipertexto.

Como sabemos, HTML es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes o animaciones. HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>) y puede describir, hasta un cierto punto, la apariencia de un documento. Además puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML tiene diferentes versiones, en las que ha cambiado bastante con respecto a la idea inicial. La web ha crecido muy rápido y ha evolucionado con mayor velocidad que los propios estándares de HTML. Por ello, el propio lenguaje HTML se ha visto superado por las necesidades de las empresas y desarrolladores y ha crecido muchas veces sin atender al plano director creado desde el W3C.

XHTML es una vuelta hacia atrás, que intenta recuperar la línea marcada por los estándares, pero que trata de solucionar diversos casos de uso del HTML, a la vez que lo prepara para adaptarse a las nuevas necesidades y corrientes tecnológicas.

Podemos decir que XHTML es la versión XML de HTML. Desde el punto de vista del desarrollador, veremos que XHTML hereda la rigidez de XML, con lo cual no se puede escribir documentos XHTML de cualquier manera, como ocurría con HTML, sino atendiendo a unas normas. Ese detalle, que en principio pueda parecer una limitación, en realidad tiene una serie de ventajas.

- Se puede procesar su contenido por cualquier programa informático (igual que ocurre con el XML)

- Los navegadores no tienen por qué volverse locos intentando interpretar lo que el desarrollador ha querido escribir ni solucionar los posibles errores de código cometidos, como ocurría con HTML.

La versión de XHTML utilizada ha sido la 1.0, la cual extiende HTML4. Esta tecnología nos ha servido para crear la base de la interfaz gráfica. El problema es que actualmente XHTML por sí solo no ofrece demasiadas posibilidades, simplemente nos permite definir los elementos básicos que aparecerán en pantalla. Es gracias al resto de tecnologías que lo complementan que hemos sido capaces de dotarla del aspecto y funcionalidad finales. Todas ellas se explican a lo largo de los siguientes apartados.

7.3.- CSS

El nombre hojas de estilo en cascada viene del inglés Cascading Style Sheets, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "<style>".

Nosotros hemos utilizado la tecnología CSS para mejorar el aspecto visual de nuestra interfaz gráfica. En el archivo *estilo.css* se han añadido todas las modificaciones necesarias para que la web alcance su aspecto final. Entre ellas podemos destacar la colocación de las capas, su color o sus márgenes. Además nos ha permitido dotar de mayor dinamismo a la página, pudiendo hacer que algunos elementos cambien en respuesta a determinados eventos de ratón.

7.4.- JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Entre las acciones típicas que se pueden realizar en Javascript tenemos dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, javascript nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

JavaScript ha sido otra de las tecnologías que nos ha permitido mejorar la interfaz. Con ella hemos aumentado el dinamismo de la página, pudiendo hacer que responda a más eventos generados por el usuario. El más importante ha sido el poder mostrar u ocultar los campos a configurar de cada parámetro en función de si su check-box (y por tanto el parámetro) estaba activo o no.

Aunque en realidad es parte de JavaScript, estas últimas funcionalidades han sido implementadas con la ayuda de la biblioteca jQuery. La explicamos en el siguiente apartado.

jQuery

jQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Consiste en un único fichero JavaScript que contiene todas estas funcionalidades.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos.

jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. Para ello utiliza las funciones $() o jQuery().

La versión utilizada ha sido la 1.7.1 y su uso nos ha permitido agilizar y disminuir el código necesario para añadir las funcionalidades mencionadas en el apartado anterior, ya que simplemente usando JavaScript hubiera sido todo más difícil.

7.5.- PHP

PHP es el acrónimo de Hipertext Preprocessor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores

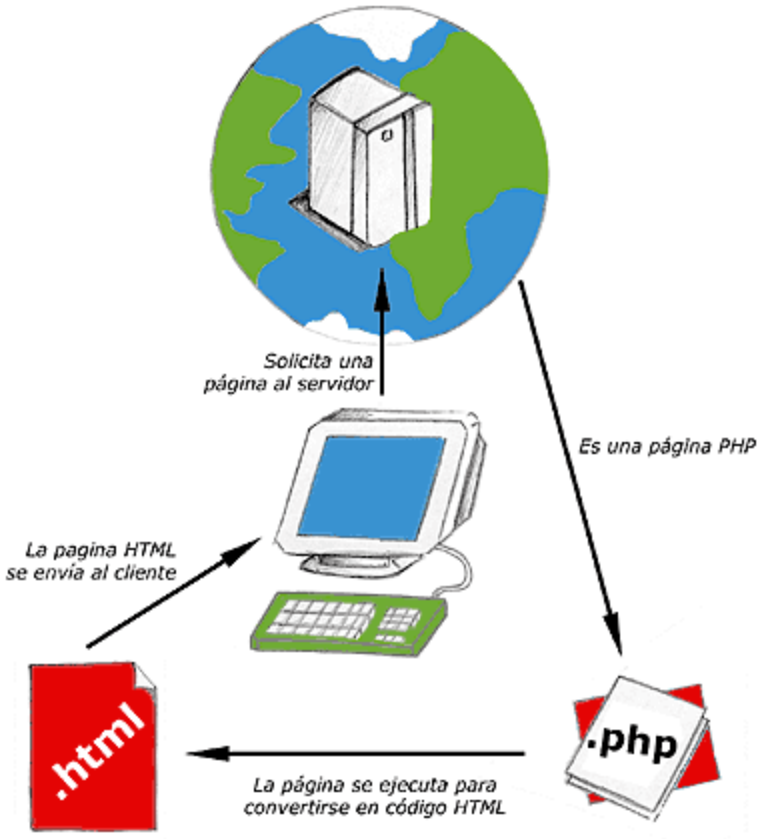


Diagrama del funcionamiento de las páginas PHP.

PHP se escribe dentro del código HTML, lo que lo hace realmente fácil de utilizar, al igual que ocurre con el popular ASP de Microsoft, pero con algunas ventajas como su gratuidad, independencia de plataforma, rapidez y seguridad.

Es independiente de plataforma, puesto que existe un módulo de PHP para casi cualquier servidor web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante, ya que permite portar el sitio desarrollado en PHP de un sistema a otro sin prácticamente ningún trabajo.

PHP, en el caso de estar montado sobre un servidor Linux u Unix, es más rápido que ASP, dado que se ejecuta en un único espacio de memoria y esto evita las comunicaciones entre componentes COM que se realizan entre todas las tecnologías implicadas en una página ASP.

Por último señalábamos la seguridad, en este punto también es importante el hecho de que en muchas ocasiones PHP se encuentra instalado sobre servidores Unix o Linux, que son de sobra conocidos como más veloces y seguros que el sistema operativo donde se ejecuta las ASP, Windows NT o 2000. Además, PHP permite configurar el servidor de modo que se permita o rechacen diferentes usos, lo que puede hacer al lenguaje más o menos seguro dependiendo de las necesidades de cada cual.

Algunas de las más importantes capacidades de PHP son: compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, y ODBC, por ejemplo. Incluye funciones para el envío de correo electrónico, upload de archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

El 13 de julio de 2004, fue lanzado PHP 5, utilizando el motor Zend Engine 2.0 (o Zend Engine 2). La última versión estable es la 5.4.3 lanzada el 8 de mayo de 2012. En nuestro caso nos hemos visto obligados a usar la versión 4.3.13 debido a problemas de compatibilidad con xAjax.

PHP ha sido una de las tecnologías con más peso dentro de nuestra interfaz. Por un lado necesitábamos una forma de gestionar la recogida y envío de información a los formularios, ya que es nuestra forma de comunicarnos con el usuario. La única vía que tiene de configurar las distintas opciones es mediante los campos que le ofrecen los distintos parámetros, pero como ya hemos dicho, está información tiene que ser gestionada de forma correcta para que los scripts se ejecuten correctamente.

Además gracias a PHP hemos comunicado a la interfaz con otras tecnologías, como XML o Ruby. Hemos utilizado las bibliotecas que ofrece para leer y escribir XML, así como para ejecutar scripts de Ruby. Este ha sido el punto clave que ha hecho que fuéramos capaces de automatizar todo el proceso.

7.6.- Ajax

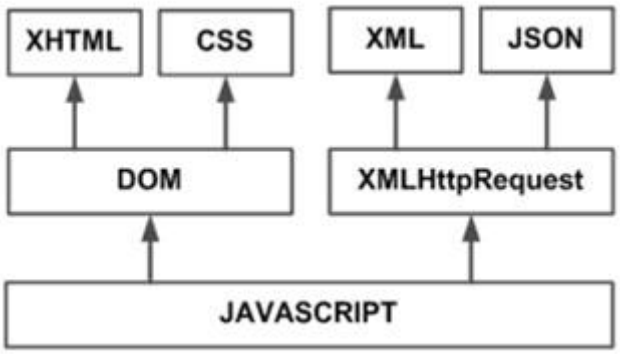
AJAX es un acrónimo de Asynchronous JavaScript + XML, que se puede traducir como "JavaScript asíncrono + XML".

El término AJAX se presentó por primera vez en el artículo "Ajax: A New Approach to Web Applications". En él se define AJAX de la siguiente forma:

> *"Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes."*

Las tecnologías que forman AJAX son:

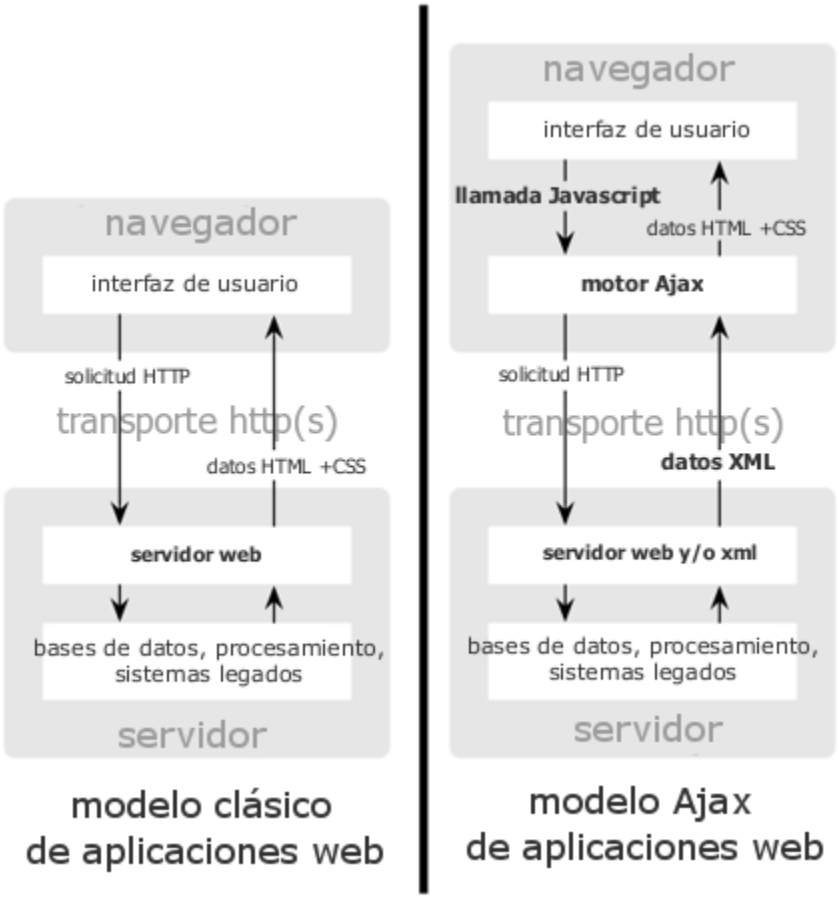
- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.



Tecnologías agrupadas bajo el concepto de Ajax.

En las aplicaciones web tradicionales, las acciones del usuario en la página (pinchar en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX:



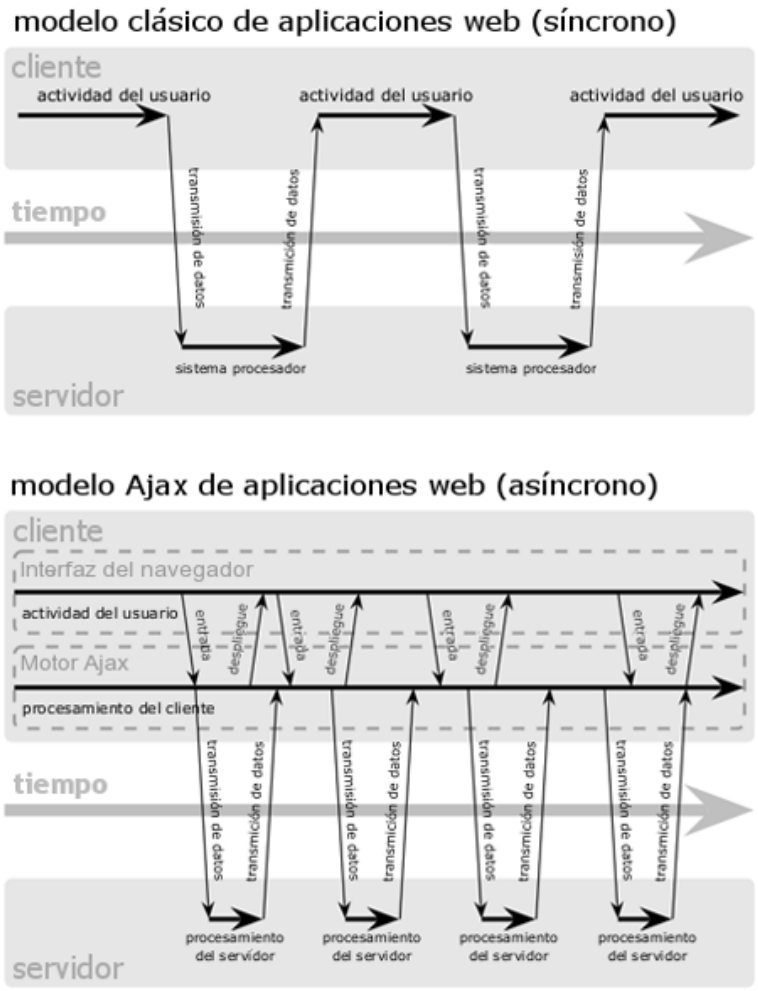
Comparación gráfica del modelo tradicional de aplicación web y del nuevo modelo propuesto por AJAX.

Esta técnica tradicional para crear aplicaciones web funciona correctamente, pero no crea una buena sensación al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto.

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

El siguiente esquema muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX. La imagen superior muestra la interacción síncrona propia de las aplicaciones web tradicionales. La imagen inferior muestra la comunicación asíncrona de las aplicaciones creadas con AJAX:



Comparación entre las comunicaciones síncronas de las aplicaciones web tradicionales y las comunicaciones asíncronas de las aplicaciones AJAX.

Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor.

El uso de Ajax en nuestro proyecto surgió de la necesidad de manejar la información de los formularios en la interfaz sin necesidad de recargar la página. Además descubrimos que nos ayudaba a cargar la información en las distintas capas con más eficacia. El inconveniente era que no estábamos demasiado familiarizados con la tecnología y resultaba algo complicado su uso. Investigando descubrimos la biblioteca Xajax incluida en PHP, la cual nos facilitaba mucho el trabajo. La describimos en el siguiente apartado.

Xajax

Ajax es una tecnología que utiliza a su vez otra combinación de tecnologías, como XML y JavaScript, para realizar peticiones de contenido o computación de servidor sin tener que recargar la página en la que está el usuario. Es una tecnología que permite una nueva gama de aplicaciones interactivas en la web, mucho más ricas y rápidas, dado que no precisamos recargar todo el contenido de una página para realizar peticiones al servidor.

Si hemos intentado alguna vez trabajar con Ajax para programar una página web, habremos comprobado que la tarea se complica bastante, teniendo que realizar diferentes trozos de código en distintos lenguajes de programación y en distintos archivos. Todo esto puede provocar dolores de cabeza o páginas con códigos difíciles de entender y de mantener. Esta cuestión sin dudas es uno de los problemas que trae Ajax a los programadores, sobre todo a los que intentan dar sus primeros pasos. Pero herramientas como Xajax pueden ayudarnos bastante.

Xajax es una biblioteca de código abierto para PHP que permite crear de manera fácil y simple aplicaciones Web basadas en AJAX usando además HTML, CSS, y JavaScript. Las aplicaciones desarrolladas con Xajax pueden comunicarse asíncronamente con funciones que se encuentran del lado del servidor y así actualizar el contenido de una página sin tener que cargarla nuevamente, su última versión es la 0.6 Beta que cambia comparado con las versiones anteriores 2.5.x y anteriores.

En un principio se crea una instancia de objeto Xajax (*xajax object*). Este objeto manejará todo el procesamiento a través de Xajax. En segundo lugar debemos registrar todas las funciones que hemos definido previamente en el objeto Xajax, esto se puede hacer usando el método *xajax->register()*. Finalmente todas las respuestas serán procesadas utilizando el método *xajax->processRequest()*.

En la actualidad se encuentran diversas clases de bibliotecas y frameworks de AJAX disponibles para los desarrolladores de aplicaciones Web. Algunas de ellas son Prototype y script.aculo.us. Sin embargo para optar por estas soluciones los desarrolladores deben tener un conocimiento, si bien no muy elevado, pero si sólido de Javascript. Por tal hecho, la mayoría de ellos y sobre todo los que utilizan PHP, al hacer uso de estos frameworks podrían encontrar dificultades.

Por otro lado, la biblioteca Xajax es una biblioteca puramente centrada en PHP. Con la introducción de la misma el manejo de AJAX en PHP se hace mucho más sencillo y sobre todo solo se necesita escribir sentencias muy cortas y simples en Javascript, lo que supone, según sus creadores, una ventaja frente a otras soluciones.

Otras de las características son:

- Xajax es compatible con Firefox, Mozilla, Internet Explorer, Opera y Safari
- Xajax puede ser usado para actualizar estilos, clases CSS, botones de selección, casillas de verificación y botones de radio o cualquier otro atributo de un elemento
- Cada función registrada para ser accesible a través de Xajax puede tener distintos tipos de petición.

La versión que hemos utilizado ha sido la última disponible, la 0.6 Beta.

Con Xajax hemos sido capaces de introducir las mejoras ya citadas en el anterior apartado de una manera muy cómoda y rápida. Simplemente añadiendo una carpeta al directorio de nuestra interfaz disponíamos ya de esta biblioteca. Las funciones que se apoyan en esta tecnología están en el archivo *funciones.php*. Todas ellas tienen la estructura básica descrita en este apartado además de su funcionalidad característica, que en la mayoría de los casos suele ser escribir en los archivos XML de forma asíncrona y sin molestia para el usuario.

7.7.- Ruby

Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995. Combina una sintaxis inspirada en Python y Perl con características de programación orientada a objetos similares a Smalltalk. Comparte también funcionalidad con otros lenguajes de programación como Lisp, Lua, Dylan y CLU. Ruby es un lenguaje de programación interpretado en una sola pasada y su implementación oficial es distribuida bajo una licencia de software libre.

La última versión estable de la rama 1.8 es la 1.8.7, de la rama 1.9 es la 1.9.3. Diferencias en rendimiento entre la actual implementación de Ruby (1.8.6) y otros lenguajes de programación más arraigados han llevado al desarrollo de varias máquinas virtuales para Ruby.

El creador del lenguaje ha dicho que Ruby está diseñado para la productividad y la diversión del desarrollador, siguiendo los principios de una buena interfaz de usuario. Sostiene que el diseño de sistemas necesita enfatizar las necesidades humanas más que las de la máquina. Ruby sigue el "principio de la menor sorpresa", lo que significa que el lenguaje debe comportarse de tal manera que minimice la confusión de los usuarios experimentados.

Ruby es orientado a objetos: todos los tipos de datos son un objeto, incluidas las clases y tipos que otros lenguajes definen como primitivas, (como enteros, booleanos, y "nil"). Toda función es un método. Las variables siempre son referencias a objetos, no los objetos mismos. Ruby soporta herencia con enlace dinámico, mixins y métodos singleton (pertenecientes y definidos por una sola instancia más que definidos por la clase). A pesar de que Ruby no soporta herencia múltiple, las clases pueden importar módulos como mixins. La sintaxis procedural está soportada, pero todos los métodos definidos fuera del ámbito de un objeto son realmente los métodos de la clase Object. Como esta clase es padre de todas las demás, los cambios son visibles para todas las clases y objetos.

Ruby ha sido descrito como un lenguaje de programación multiparadigma: permite programación procedural (definiendo funciones y variables fuera de las clases haciéndolas parte del objeto raíz Object), con orientación a objetos, (todo es un objeto) o funcionalmente (tiene funciones anónimas, clausuras o closures, y continuations; todas las sentencias tiene valores, y las funciones devuelven la última evaluación). Soporta introspección, reflexión y metaprogramación, además de soporte para hilos de ejecución gestionados por el intérprete. Ruby tiene tipado dinámico, y soporta polimorfismo de tipos (permite tratar a subclases utilizando la interfaz de la clase padre). Ruby no requiere de polimorfismo de funciones al no ser fuertemente tipado (los parámetros pasados a un método pueden ser de distinta clase en cada llamada a dicho método).

Gracias a la facilidad de uso de este lenguaje hemos podido generar de forma rápida los scripts que realmente interactúan con el router para su configuración. Nos ha permitido elaborar el código que lee la información de los archivos XML (en los cuales se almacenan los datos básicos de cada parámetro), además de las instrucciones del sistema que se ejecutarán con dicha información.

7.8.- Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, *a patchy server* (un servidor "parcheado") suena igual que *Apache Server*.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web. Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet. Los programadores de aplicaciones web a veces utilizan una versión local de Apache con el fin de previsualizar y probar código mientras éste es desarrollado.

La mayor parte de la configuración de Apache se realiza en el fichero apache2.conf o httpd.conf, según el sistema donde esté corriendo. Cualquier cambio en

este archivo requiere reiniciar el servidor, o forzar la lectura de los archivos de configuración nuevamente.

La última versión estable de Apache es la 2.4.2, lanzada el 17 de abril de 2012, siendo ésta la que utilizamos en el proyecto.

Nosotros hemos hecho un doble uso de Apache. Por un lado nos ha servido de apoyo a la hora de elaborar la interfaz gráfica. Debido a que ésta contiene PHP y Xajax, las pruebas deben realizarse con un servidor ya que es necesario emular la comunicación con éste. La forma más rápida y sin coste es instalar Apache de forma local en los ordenadores en los que se realizan las pruebas y colocar en éstos la página web, pudiendo acceder a ella desde cualquier navegador.

Por otro lado es parte de nuestro proyecto, ya que si queremos usar la interfaz como forma de configuración del router necesitamos alojarla en un servidor dentro de éste. Nosotros partíamos de una configuración de Ubuntu con los servicios básicos, por lo que lo primero fue instalar Apache y adaptarlo a nuestras necesidades, como se explica más delante.

7.9.- Ubuntu

Ubuntu es un sistema operativo mantenido por Canonical y la comunidad de desarrolladores. Utiliza un núcleo Linux, y su origen está basado en Debian. Ubuntu está orientado al usuario novel y promedio, con un fuerte enfoque en la facilidad de uso y mejorar la experiencia de usuario. Está compuesto de múltiple software normalmente distribuido bajo una licencia libre o de código abierto. Según estadísticas, el porcentaje de mercado de Ubuntu dentro de "distribuciones linux" es de aproximadamente 49%, y con una tendencia a subir como servidor web.

Su patrocinador Canonical, es una compañía británica propiedad del empresario sudafricano Mark Shuttleworh ofrece el sistema de manera gratuita y que se financia por medio de servicios vinculados al sistema operativo y vendiendo soporte técnico. Además, al mantenerlo libre y gratuito, la empresa es capaz de aprovechar los desarrolladores de la comunidad para mejorar los componentes de su sistema operativo.

Canonical además de mantener Ubuntu, también provee de una versión orientada a servidores, Ubuntu Server, una versión para empresas, Ubuntu Business Desktop Remix, una para televisores, Ubuntu TV, y una para usar el escritorio desde smartphones, Ubuntu for Android.

En su última versión, Ubuntu soporta oficialmente dos arquitecturas de hardware en computadoras personales y servidores: 32-bit y 64-bit. Sin embargo, extraoficialmente, Ubuntu ha sido portado a dos arquitecturas más: SPARC y IA-64. A partir de la versión 9.04, se empezó a ofrecer soporte oficial para procesadores ARM, comúnmente usados en dispositivos móviles. Al igual que la mayoría de los sistemas de escritorio basados en Linux, Ubuntu es capaz de actualizar a la vez todas las aplicaciones instaladas en la máquina a través de repositorios.

La máquina virtual que actúa como router en nuestro proyecto lleva el sistema operativo Ubuntu. La única pega es que no puede tener nada instalado ya que en principio el objetivo del

proyecto es lograr la configuración de la máquina como encaminador partiendo de cero. Por ello, hemos utilizado una versión de Ubuntu con los servicios básicos, Ubuntu JeOS, descrita en el siguiente apartado.

Ubuntu JeOS

Ubuntu JeOS es una eficiente variante del Sistema Operativo de servidores Ubuntu creada específicamente para el diseño de virtual appliances de VMware. Está disponible para su descarga como imagen ISO, JeOS es una instalación especializada de Ubuntu Server con un kernel compilado que contiene únicamente los componentes del sistema operativo base, necesarios para ser ejecutado dentro de ambientes virtuales. Ubuntu JeOS fue modificado para utilizar las ventajas de las tecnologías clave de virtualización de los productos de VMware. Esta combinación de tamaño reducido y performance optimizada garantiza que Ubuntu JeOS realiza un uso eficiente de los recursos de servidores en grandes ambientes virtuales.

No tiene controladores innecesarios y dispone solo de los paquetes mínimos, por lo que se puede configurar el sistema operativo exactamente como se requiera. Por lo tanto los usuarios que utilicen virtual appliances basados en JeOS deberán realizar un mantenimiento menor sobre sus equipos en comparación con otros virtual appliances basados en las versiones completas del sistema operativo.

Características:

- Tamaño: 100 Mb, en formato de imagen ISO
- Optimizada por VMware ESX, VMware Server
- Procesador: Arquitecturax86 Intel o AMD
- RAM óptima: Memoria mímina recomendada de 128 Mb
- No posee un entorno gráfico precargado.
- Se necesitan conocimientos básicos de administración en Linux, y dkpg y aptitude recomendados para comenzar a construir su propia aplicación.

La última versión estable es la 10.10 (Maverick Meerkat) lanzada el 10 de octubre de 2010. Nosotros utilizamos la 8.10, a la cual corresponden las características mencionadas anteriormente.

Como ya hemos dicho, necesitábamos cargar una imagen de Ubuntu limpia en nuestra máquina virtual (router), por lo que nuestro director de proyecto nos sugirió utilizar esta versión de Ubuntu JeOS. Por tanto podemos considerar este elemento como punto de partida de nuestro proyecto y sobre el cual nos hemos basado para trabajar en la creación de los scripts que lo configurarán como encaminador, aunque no de forma completa, pero sí dotándolo de varias características importantes.

8.- Configuración del entorno

Para simular la arquitectura de red con un router y varios host hemos utilizado VMware Fusion como software de virtualización, donde en cada una de las máquinas virtuales hemos instalado el sistema operativo Ubuntu JeOS.

8.1.- VMware Fusion

Es un software de virtualización de sistemas de escritorio desarrollado por VMware para ordenadores Macintosh con procesadores Intel.

Fusion permite instalar como sistemas "invitados" virtuales tanto sistemas Microsoft Windows como distintas distribuciones Linux, pudiendo ejecutar de forma simultánea varias máquinas.

En nuestro caso para realizar las pruebas instalamos dos máquinas con Ubuntu JeOS 8.10.

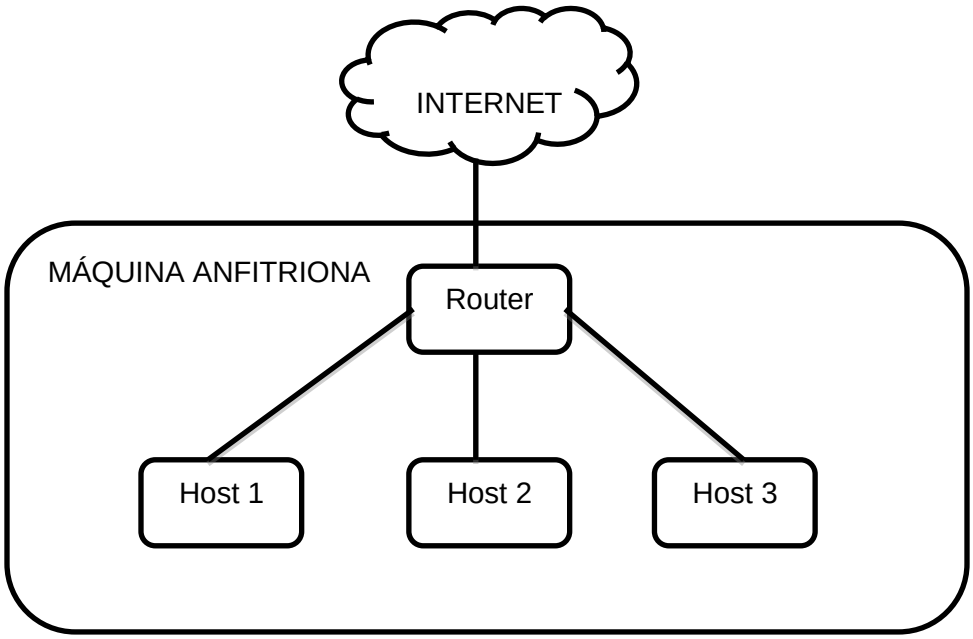
La máquina virtual que denominaremos "Router" la configuramos con dos adaptadores de red:

- El primero de ellos comparte la conexión de red con el sistema anfitrión mediante NAT

- El segundo adaptador de red lo configuramos para que cree una red privada disponible y compartida entre todas las máquinas virtuales.

En la máquina virtual denominada que denominaremos “Host”, únicamente hemos configurado un interfaz de red que se conecte a la red privada.

Por tanto la arquitectura de red resultante sería la siguiente:



Para realizar esta virtualización de red existían otras alternativas como pueden ser **UML (User-Mode Linux), VirtualBox o Parallels**, pero independientemente del sistema de virtualización utilizado la configuración ha realizar en las máquinas virtuales es equivalente.

En nuestro caso elegimos VMware Fusion por tener una disponibilidad inmediata en el equipo con el que realizamos las pruebas, puesto que ya estaba instalado de antes y además el conocimiento del programa nos facilitaba el trabajo.

8.2.- Ubuntu JeOS

El sistema operativo instalado en las máquinas virtuales que utilizamos fue Ubuntu JeOS 8.10.

Ubuntu JeOS (Just Enough Operating System) es una variante de Ubuntu Server configurada especialmente para entornos virtuales. El kernel de esta variante está modificado para que posea los elementos básicos del sistema, proporcionando así un alto rendimiento y poco tamaño en disco.

Para configurar el Router es importante que la IP de la red privada sea fija, para ello durante la instalación de Ubuntu:

1. Cuando se alcanza la ventana de configuración de red hay que pulsar “esc” para acceder al menú de configuración.
2. Seleccionar la configuración manual de la red.
3. Introducir una IP fija, preferiblemente de clase C, y el resto de configuración de red como podría ser la máscara de red, la dirección de red o la dirección de broadcast.
4. Una vez esté realizada la configuración el instalador continuará con el resto de pasos.

Para poder ejecutar los scripts del router es necesario disponer de un usuario con permisos de administrador, es decir, sería interesante activar el usuario root tras la instalación:

```
% sudo -u root passwd
```

Una vez instalado nuestro sistema operativo es necesario instalar una serie de herramientas para que nuestros scripts funcionen y nuestra página web sea accesible:

8.3.- Ruby

Puesto que nuestros scripts están escritos en ruby necesitamos instalar el intérprete, para ello podemos utilizar el gestor de paquetes. Como estamos trabajando con Ubuntu:

```
% sudo apt-get install ruby irb doc
```

8.4.- Apache2

Debido a que los scripts de configuración serán llamados desde la web de configuración es necesario que el servidor de apache tenga permisos de administrador, aunque ello implique un problema de seguridad. Por estos problemas de seguridad apache viene precompilado para que no sea posible iniciar su demonio si su usuario tienes derechos de administrador, por tanto a la hora de instalar apache es necesario descargar el código fuente de su página web http://www.apache.org/ y compilarlo con la opción de compilación –DBIG_SECURITY_HOLE.

Para poder compilar apache necesitamos dos paquetes que nos permitirán compilar apache sin problemas:

```
% sudo apt-get install build-essential
% sudo apt-get install libpcre3-dev
```

Build-essential se trata de un meta-paquete que posee las instrucciones para compilar C/C++.

Los paquetes a los que hace referencia son los siguientes:

- g++
- gcc
- libc6-dev
- make

Por otro lado libpcre3-dev es un paquete que contiene una biblioteca de expresiones regulares compatibles con Perl5.

Una vez que disponemos de los paquetes necesarios se procedería a la compilación de apache:

```
% sudo env CFLAGS=”-DBIG_SECURITY_HOLE” ./configure
--prefix=/etc/apache2
```

Con estas líneas configuramos las opciones de configuración como la ruta de instalación de apache y ahora compilamos e instalamos:

```
% make ; make install
```

Ahora ya podemos configurar nuestro servidor apache cambiando el nombre de usuario y grupo en el archivo **httpd.conf** para otorgarle permisos de administrador.

Finalmente con nuestra configuración deseada arrancamos el demonio de apache:

```
% /etc/apache2/bin/apachectl –f /etc/apache2/conf/httpd.conf
```

8.5.- PHP5

Como nuestra página web de configuración está implementada en PHP es necesario instalarlo para que todo funcione correctamente. Como consecuencia de haber compilado e instalado apache manualmente, instalar php desde el gestor de paquetes da algunos problemas, por lo que la solución más cómoda es descargar el código fuente y compilarlo a mano al igual que hicimos con apache.

En este caso necesitamos instalar previamente otro paquete para la compilación:

```
% sudo apt-get install libxml2-dev
```

Este paquete es una librería XML para desarrolladores que permite crear tus propios lenguajes de marcado.

Ahora configuramos la ruta de instalación y le indicamos donde está instalado apache y donde queremos que se guarde el archivo de configuración de PHP:

```
% ./configure –prefix=/etc/php5 –with- apxs2=/etc/apache2/bin/apxs –with-
config-file-path=/etc/php5
```

Tras esto ya podemos compilar e instalar PHP5:

```
% make ; make install
```

Una vez instalado PHP es necesario crear un archivo de configuración, podemos usar el archivo que viene con la instalación:

```
% cp php.ini-development php.ini
```

Para que apache sepa como interpretar archivos .php es necesario modificar el archivo de configuración httpd.conf situado en /etc/apache2/conf donde habría que añadir un nuevo tipo:

```
AddType application/x-httpd-php .php
```

9.-Configuración de servicios

Una vez tenemos nuestro entorno, listo para ser usado, tras instalar las especificaciones explicadas anteriormente, ahora es momento de dejar configurados los servicios necesarios para el correcto funcionamiento del router.

Como se verá en esta sección, tenemos diferentes maneras de ir instalando los servicios en nuestra máquina virtual que funcionará como router.

En primer lugar, trataremos la versión más sencilla a priori: de manera manual. No requiere de ningún fichero adicional, bastará con añadir las instrucciones necesarias para la configuración de cada servicio. Esto presenta inconvenientes: es un proceso engorroso estar escribiendo las instrucciones una a una cada vez que iniciamos una máquina virtual limpia.

Surge la idea de tener una serie de scripts, que realizarán la misma función que en el caso anterior, pero que lo único que tendremos que hacer será lanzar estos scripts desde la máquina virtual y automáticamente se configurarán los parámetros en concreto que estemos tratando.

9.1.- Manual

Según lo explicado anteriormente, en esta sección, se adjuntarán las instrucciones necesarias para la configuración manual de los siguientes servicios:

9.1.1.- DNS

Activar:

```
/etc/init.d/dnsmasq start
```

Desactivar:

```
/etc/init.d/dnsmasq stop
```

9.1.2.- NAT

Activar:

```
iptables –t nat –A POSTROUTING –o #{int} –j MASQUERADE
```

Donde *#{int}* es la interfaz pública.

Desactivar:

```
iptables –t nat –F POSTROUTING
```

9.1.3.- Ip forwarding

Activar:

```
sysctl –w net.ipv4.conf.all.forwarding=1
```

Desactivar:

```
sysctl –w net.ipv4.conf.all.forwarding=0
```

9.1.4.- Port forwarding

Para activar cada regla:

```
iptables -t nat -A PREROUTING -i #{int} -p tcp --dport #{port} \ -j
DNAT --to #{ip}
```

Donde *#{int}* es la interfaz pública, *#{port}* es el puerto público que queremos redirigir, *#{ip}* es la IP y el puerto de la máquina hacia la que queremos redirigir.

Para desactivar:

```
iptables -t nat -F PREROUTING`
```

9.1.5.- DHCP

Para configurar el servidor dhcp tenemos que editar el fichero *"/etc/dhcp3/dhcpd.conf"* que tiene el siguiente formato:

```
subnet #{net} netmask #{netmask} {
     range #{minip} #{maxip} ;
     default-lease-time #{deflt} ;
     max-lease-time #{maxlt} ;
     option routers #{ip router} ;
     option broadcast-address #{ip broadcast} ;
     option subnet-mask #{subnetmask} ;
     option domain-name-servers #{ip dns} ;
}
```

Una vez editado el archive de configuración hay que reiniciar el servicio:

```
sudo /etc/init.d/dhcp3-server restart
```

Para desactivar el servidor dhcp, hay que parar el servicio:

```
sudo /etc/init.d/dhcp3-server stop
```

9.2.- Mediante scripts

En esta sección, nos encargaremos de ver los scripts necesarios para la configuración y una clara explicación de cada uno de los ficheros, a través de los cuales, comprobaremos que estamos realizando la misma funcionalidad que de manera manual.

9.2.1.-DNS

Fichero XML

```
<dns>
     <up>0</up>
</dns>
```

Fichero ruby

```
#! /usr/bin/env ruby
require "rexml/document"
file = File.open( "dnsconf.xml" )
doc = REXML::Document.new file

# Configurando Proxy-DNS

if doc.elements["dns/up"].get_text=='1'
     puts "Instalando dnsmasq.."
     `sudo apt-get install dnsmasq`
     `sudo /etc/init.d/dnsmasq start`
     puts "Dnsmasq activado."
else
     puts "Parando dnsmasq..."
     `sudo /etc/init.d/dnsmasq stop`
     puts "Dnsmasq parado."
```

```
end
```

En primer lugar vemos si en el fichero XML, el elemento up está activado o no. En caso afirmativo instalaremos dnsmasq y lo activaremos. En caso de que el valor de up sea 0, la única instrucción a ejecutar será la de parar el servidor dnsmasq.

9.2.2.- NAT

Fichero XML

```
<nat>
     <up>1</up>
     <interf>eth0</interf>
</nat>
```

Fichero ruby

```
#! /usr/bin/env ruby
require "rexml/document"
file = File.open( "natconf.xml" )
doc = REXML::Document.new file

# Configuración NAT

puts `sudo apt-get install iptables`
if doc.elements["nat/up"].get_text == '1'
     puts "Configurando NAT"
     puts "Masquerade..."
     int = doc.elements["nat/interf"].get_text
     `sudo iptables -t nat -A POSTROUTING -o #{int} -j MASQUERADE`
     puts "NAT activado"
else
     puts "Parando NAT"
     `sudo iptables -t nat -F POSTROUTING`
     puts "NAT parado"
end
```

En el caso de NAT, la primera comprobación será ver si el parámetro up está activado en el script XML. En caso de que valga 1, configuraremos NAT, añadiendo la regla *"sudo iptables -t nat -A POSTROUTING -o #{int} -j MASQUERADE"*, donde "int" es el valor de la interfaz leído, como en el caso de "up", del fichero XML. De esta manera quedará activado.

En el caso de que up tenga valor 0, simplemente pararemos NAT, mediante la instrucción correspondiente.

9.2.3.- IP forwarding

Fichero XML

```
<ip-for>
     <up>1</up>
</ip-for>
```

Fichero ruby

```
#! /usr/bin/env ruby
require "rexml/document"
file = File.open( "ipfconf.xml" )
doc = REXML::Document.new file

# Configurando IP Forwarding

if doc.elements["ip-for/up"].get_text=='1'
     puts "Activando IP Forwarding.."
     `sudo sysctl -w net.ipv4.conf.all.forwarding=1`
     puts "IP Forwarding activado."
else
     puts "Desactivando IP Forwarding..."
     `sudo sysctl -w net.ipv4.conf.all.forwarding=0`
     puts "IP Forwarding desactivado."
end
```

Como en los casos anteriores, para la configuración de IP forwarding, tendremos en cuenta el valor de up, que proviene del script XML. En caso de tenerlo a 1, activaremos el servicio IP forwarding, mediante la instrucción correspondiente. Si up vale 0, pararemos el servicio, como se puede ver en archivo ruby.

9.2.4.- Port forwarding

Fichero XML

```
<port-for>
</port-for>
```

Partiremos de un script XML limpio, sin ninguna regla, que más tarde se irán añadiendo, desde el fichero ruby o como veremos más adelante a través de la interfaz gráfica.

Fichero ruby

```
#! /usr/bin/env ruby
require "rexml/document"
file = File.open( "portforconf.xml" )
doc = REXML::Document.new file

# Configuración Port-forwarding
puts "Instalando iptables"
`sudo apt-get install iptables`
puts "Aplicando reglas"
numRegla = 1;
puts `sudo iptables -t nat -F PREROUTING`
doc.elements.each("port-for/regla") {
     |element|
     int = element.elements["interf"].get_text
     port = element.elements["port"].get_text
     ip = element.elements["ip"].get_text
     puts "Configurando regla #{numRegla}"
     `sudo iptables -t nat -A PREROUTING -i #{int} -p tcp --dport
#{port} \ -j DNAT --to #{ip}`
     numRegla=numRegla+1
}
puts "Reglas configuradas"
```

En port-forwarding se irá leyendo del archivo XML, cada una de las reglas que tenemos que añadir. Seleccionamos los valores de la interfaz, el puerto y la dirección IP de cada regla y utilizando *"sudo iptables -t nat -A PREROUTING -i #{int} -p tcp --dport #{port} \ -j DNAT --to #{ip}"* realizamos la adicción a iptables de la regla en cuestión.

9.2.5.- DHCP

Fichero XML

```
<dhcp>
     <up>0</up>
     <net>192.168.1.0</net>
     <netmask>255.255.255.0</netmask>
     <minip>192.168.1.100</minip>
     <maxip>192.168.1.200</maxip>
     <deflt>86400</deflt>
     <maxlt>60480</maxlt>
     <router>192.168.1.1</router>
     <broadcast>192.168.1.255</broadcast>
     <subnetm>255.255.255.0</subnetm>
     <dns>192.168.1.1</dns>
</dhcp>
```

Fichero ruby

```
#! /usr/bin/env ruby
require "rexml/document"
file = File.open( "dhcpconf.xml" )
doc = REXML::Document.new file

# Configuración DHCP
if doc.elements["dhcp/up"].get_text == '1'
     puts "Instalando DHCP3server.."
     puts `sudo apt-get --force-yes install dhcp3-server`
     puts "Configurando DHCP.."

         archivo = File.new("/etc/dhcp3/dhcpd.conf", "w")
         net = doc.elements["dhcp/net"].get_text
         netmask = doc.elements["dhcp/netmask"].get_text
         minip = doc.elements["dhcp/minip"].get_text
         maxip = doc.elements["dhcp/maxip"].get_text
```

```
                deflt = doc.elements["dhcp/deflt"].get_text
                maxlt = doc.elements["dhcp/maxlt"].get_text
                router = doc.elements["dhcp/router"].get_text
                broadcast = doc.elements["dhcp/broadcast"].get_text
                subnetm = doc.elements["dhcp/subnetm"].get_text
                dns = doc.elements["dhcp/dns"].get_text

                archivo.puts "subnet #{net} netmask #{netmask} { \n"
                archivo.puts "range #{minip} #{maxip} ; \n"
                archivo.puts "default-lease-time #{deflt} ; \n"
                archivo.puts "max-lease-time #{maxlt} ; \n"
                archivo.puts "option routers #{router} ; \n"
                archivo.puts "option broadcast-address #{broadcast} ; \n"
                archivo.puts "option subnet-mask #{subnetm} ; \n"
                archivo.puts "option domain-name-servers #{dns} ;\n"
                archivo.puts "}"
            archivo.close
            puts "DHCP configurado, reiniciando DHCP3server.."
            puts `sudo /etc/init.d/dhcp3-server restart`
            puts "DHCP configurado con éxito"
        else
            puts "Parando DHCP3server.."
            puts `sudo /etc/init.d/dhcp3-server stop`
            puts "DHCP3server parado."
        end
```

DHCP dispone de una alta cantidad de parámetros. Se irán leyendo cada uno de ellos del fichero XML. Para poner en marcha el servidor DHCP realizaremos una copia de los valores mencionados anteriormente, en el archivo de configuración */etc/dhcp.conf.* Al finalizar, reiniciaremos el servicio dhcp3. Todo esto se realizará si tenemos la variable up con valor 1. En caso contrario, simplemente paramos el servicio dhcp3, mediante la instrucción “sudo /etc/init.d/dhcp3-server stop”.

9.3.- Comprobación

Una vez hemos configurado los servicios, ya sea de manualmente o mediante scripts, podemos comprobar que todo se ha realizado de manera satisfactoria. Las instrucciones para dicha comprobación son las siguientes:

9.3.1.- DNS

```
pgrep -l dns
```

9.3.2.- NAT

```
iptables -L -t nat
```

9.3.3.- Ip forwarding

```
sysctl net.ipv4.conf.all.forwarding
```

9.3.4.- Port forwarding

```
iptables -L -t nat
```

9.3.5.- DHCP

```
pgrep -l dhcp
```

10.- Interfaz Gráfica

En el apartado anterior hemos visto como se configuran los servicios de forma manual o mediante scripts. Ahora veremos otra opción que ofrece nuestro proyecto, y es hacerlo todo a través de una interfaz gráfica, la cual automatizará todo el proceso anterior, dejando solo al usuario la introducción de los datos necesario (que serán diferentes según el tipo de parámetro).

Nuestra interfaz será una página web alojada en el router, más concretamente en el servidor apache que hemos instalado en él. Para poder acceder a la web solo tendremos que poner la dirección IP del router en cualquier navegador y lo primero que se mostrará será lo siguiente:

Se trata de una página de configuración en la cual no necesitamos registrarnos previamente ni tener una cuenta de usuario válida. Podemos acceder a la configuración de los distintos parámetros de forma independiente mediante el menú situado en la izquierda de la pantalla.

De forma general, cada vez que accedemos a cada una de las subpáginas, se cargará en la parte central de nuestra interfaz la información relacionada con el parámetro que queramos modificar. Dicha información proviene de los distintos archivos XML que se encuentran también alojados en el router y se corresponderá con la configuración que posee en ese momento.

Después de realizar los cambios que queramos debemos pulsar siempre el botón “Guardar cambios”, que estará disponible en todas las subpáginas, para que se grabe la información, ya que de lo contrario se ignorará. Una vez pulsado, la interfaz de forma

automática llamará a los scripts Ruby encargados de modificar los parámetros del router y modificará también la información de los XML para que los reflejen.

En los siguientes apartados se describen las distintas posibilidades de configuración que se ofrecen:

10.1.- IP Forwarding

Las dos únicas opciones para IP Forwarding es tenerlo activado o desactivado, lo cual gestionaremos con el check-box de esta subpágina. Dejando marcada la casilla indicamos que queremos activar IP Forwarding, y por el contrario dejándola sin marcar lo desactivará.

10.2.- NAT

NAT nos ofrece más opciones de configuración que el parámetro anterior, aunque la opción principal seguirá siendo tenerlo activado o desactivado, lo cual se modificará de igual forma mediante el check-box.

Solo si marcamos NAT como activo podremos configurar el resto de opciones. En este caso solo se nos permite indicar cual queremos que sea la interfaz del router que actuará como pública. La forma de hacerlo es escribiendo correctamente su nombre en el text-field de la interfaz. Si por el contrario NAT está desactivado, no podremos cambiar ningún parámetro.

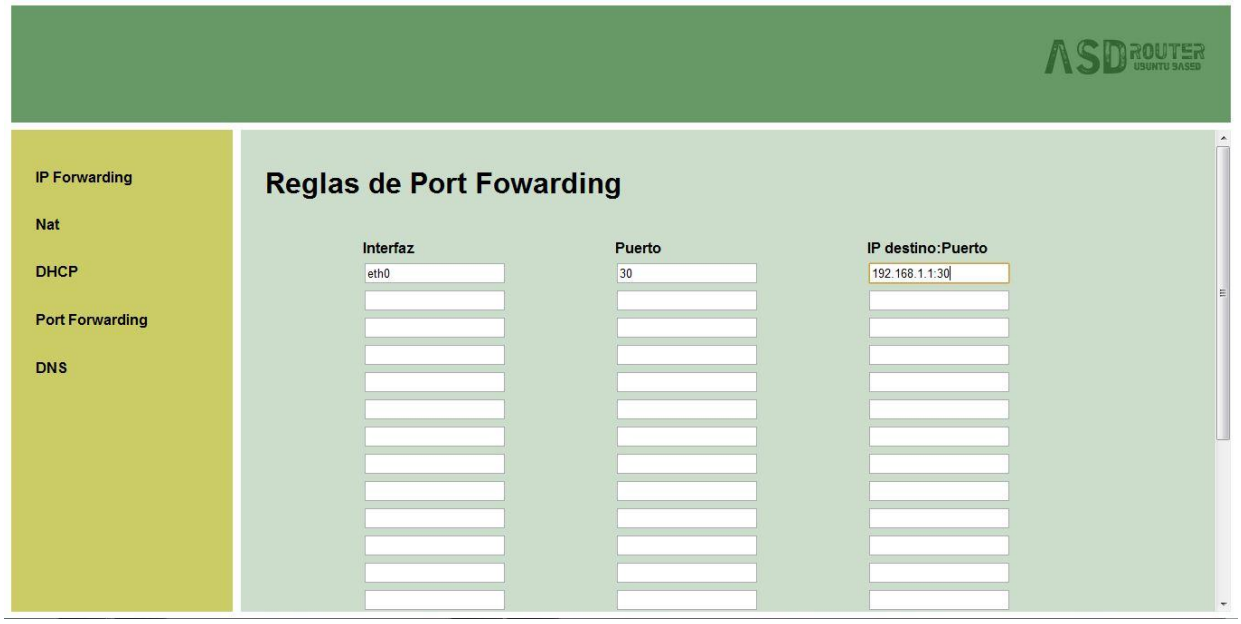
Además NAT funcionará de forma independiente a IP Forwarding, aunque para que funcione el primero deberá estar activado el segundo. Esto no se hace de forma automática, será el usuario en el encargado de gestionar correctamente estas dos características. En la interfaz gráfica simplemente se verá un texto recordatorio.

10.3.- DHCP

Al igual que los dos anteriores casos, si no tenemos activado DHCP no podremos configurar nada. En cambio si activamos el check-box tendremos acceso a todos los parámetros.

De la misma forma que NAT, tenemos a nuestra disposición una serie de text-fields que tendrán que ser rellenados con los datos de forma correcta. Tenemos la posibilidad de cambiar el rango de IPs que DHCP asignará a los distintos host que tenga a cargo el router, así como el tiempo que dichas asignaciones permanecerán activas.

10.4.- Port Forwarding



Port Forwarding tiene un sistema de configuración algo distinto al resto. Como podemos ver tenemos una tabla con los datos de cada una de las reglas. En principio se ofrece un número limitado de campos para introducir reglas.

Cada vez que queramos introducir una regla, debemos rellenar todos los campos que existen: el nombre de la interfaz que aplicará la regla, el puerto de entrada al router que será redirigido y la IP del host destino seguida del puerto al cual serán direccionadas las peticiones.

Solo las reglas que tengan todos estos campos con datos (siempre dejando libertad al usuario al rellenarlos pero esperando que lo haga de forma correcta) serán consideradas y cuando se pulse “Guardar cambios” serán añadidas al router. Todas aquellas que tengan al menos uno de sus campos vacío serán desechadas.

10.5.- DNS

DNS se configura de forma análoga a IP Forwarding. Sus dos únicas opciones es permanecer activado o desactivado, lo cual gestiona el check-box.

11.- Conclusiones

En primer lugar queremos valorar de forma muy positiva nuestra participación en este proyecto. Ya a priori nos pareció una idea bastante interesante puesto que el Cloud Computing es una tecnología muy actual y con un gran futuro por delante, aparte de ser algo que no se enseña a lo largo de la carrera. También nos motivó el hecho de que podía ser algo reutilizable en aplicaciones reales, por lo que no tuvimos ninguna duda en la elección.

Una vez que empezamos a trabajar en el proyecto, descubrimos que era más complejo de lo que en un principio puede parecer y necesitamos investigar sobre gran variedad de tecnologías que eran necesarias para su completa elaboración. Tuvimos que ampliar los conocimientos adquiridos en las asignaturas de Redes y Laboratorio de Redes sobre los servicios que podría ofrecer nuestro router, aprender el lenguaje de programación Ruby con el fin de parsear los archivos XML así como la programación Web para elaborar una interfaz gráfica que interactuara con nuestro router, entre otras cosas.

Además en el apartado personal tuvimos que buscar una forma de coordinarnos en grupo para avanzar en la elaboración del proyecto, puesto que en el primer cuatrimestre un componente del grupo se encontraba en el extranjero. Esto nos dificultó bastante y provocó que no pudiéramos ampliar con algún servicio más la funcionalidad de nuestro router. Pese a ello, estamos bastante satisfechos con todo lo logrado y en general ha sido una buena experiencia tanto a nivel personal como académico.

Apéndices

A.- Código de la interfaz web

A.1.- Index.php

```
<?php
require('xajax/xajax_core/xajax.inc.php'); //incluimos la clase Ajax
require('funciones.php');
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<?php
    $xajax->configure("javascript URI","xajax/");
    $xajax->printJavascript();
?>
<!-- Meta -->
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<!-- Titulo -->
<title>Configuracion</title>

<!-- Hojas de estilo -->
<link rel="stylesheet" href="estilo.css" type="text/css">

<!-- Archivos JavaScript -->
<script type="text/javascript" src="scripts.js"></script>
<script type="text/javascript" src="jQuery.js"></script>

</head>

<body>

<div id="parteSuperior">

    <div id="logo">
        <img id=img src="images/logo.png"/>

    </div>

</div>

<div id="menu">
    <table>
        <tr>
            <td><font class="menu"
onclick="xajax_cargarEnDiv('ipf.php','parteCentral')">IP
Forwarding</font></td>
        </tr>
        <tr>
            <td><font class="menu"
onclick="xajax_cargarEnDiv('nat.php','parteCentral')">Nat</font></td>
        </tr>
```

```
        <tr>
            <td><font class="menu"
onclick="xajax_cargarEnDiv('dhcp.php','parteCentral')">DHCP</font></td>
        </tr>
        <tr>
            <td><font class="menu"
onclick="xajax_cargarEnDiv('portf.php','parteCentral')">Port
Forwarding</font></td>
        </tr>
        <tr>
            <td><font class="menu"
onclick="xajax_cargarEnDiv('dns.php','parteCentral')">DNS</font></td>
        </tr>
    </table>
</div>

<div id="parteCentral"></div>

</body>

</html>
```

A.2.- ipf.php

```
<html>
<head>
<link rel="stylesheet" href="estilo.css" type="text/css">
<script type="text/javascript" src="scripts.js"></script>
<script type="text/javascript" src="jQuery.js"></script>
</head>
<body>

<?php

//Lectura del xml
$archivo = 'ipfconf.xml';
if(file_exists($archivo)){
  $datos = simplexml_load_file($archivo);
  if($datos){
    $up = $datos->up;
  } else echo "Sintaxis XML inválida";
} else echo "Error abriendo ipfconf.xml";

//Mostramos el formulario en función del valor de "up"
if($up==1) $chk="checked";
else $chk="";

echo "<form action=\"\" method=\"post\" name=\"ipfconf\">
        <br />
        <h1>Configuraci&oacute;n IP Forwarding</h1>
        <h3>IP Forwarding activado:  <input id=\"activaripf\"
type=\"checkbox\" $chk/></h3>
        <br />
```

```
        <input type=\"button\" value=\"Guardar cambios\"
onclick=\"xajax_escribirIpf(activaripf.checked)\">
</form>";

?>
</body>
</html>
```

A.3.- nat.php

```
<html>
<head>
<link rel="stylesheet" href="estilo.css" type="text/css">
<script type="text/javascript" src="scripts.js"></script>
<script type="text/javascript" src="jQuery.js"></script>
</head>
<body>

<?php

//Lectura del xml
$archivo = 'natconf.xml';
if(file_exists($archivo)){
  $datos = simplexml_load_file($archivo);
  if($datos){
    $up = $datos->up;
    $interf = $datos->interf;
  } else echo "Sintaxis XML inválida";
} else echo "Error abriendo natconf.xml";


//Mostramos el formulario en función del valor de "up"
if($up==1){
    $chk="checked";
    $dsp="block";
}else{
    $chk="";
    $dsp="none";
}


//Formulario
echo "<form action=\"\" method=\"post\" name=\"natconf\">
        <br />
        <h1>Configuraci&oacute;n NAT</h1>
        <h3>NAT activado:  <input id=\"activarnat\" type=\"checkbox\"
onClick=\"habilita('#activarnat','#parteOcultar')\" $chk/></h3>
        <br />
        <div id=\"parteOcultar\" style=\"display:$dsp\">
            <p>Interfaz de red p&uacute;blico: <input name=\"interfazpub\"
type=\"text\" value=\"$interf\" size=\"5\" maxlength=\"5\" /></p>
        </div>
        <br />
        <br />
```

```
        <input type=\"button\" value=\"Guardar cambios\"
onclick=\"xajax_escribirNat(xajax.getFormValues(natconf),activarnat.checked)\
">
    </form>";

?>
<br />
<br />
<br />
<strong> NOTA: <Strong> Para que funcione NAT debe estar activado IP
Forwarding

</body>
</html>
```

A.4.- dhcp.php

```
<html>
<head>
<link rel="stylesheet" href="estilo.css" type="text/css">
<script type="text/javascript" src="scripts.js"></script>
<script type="text/javascript" src="jQuery.js"></script>
</head>
<body>

<?php

//Lectura del xml
$archivo = 'dhcpconf.xml';
if(file_exists($archivo)){
  $datos = simplexml_load_file($archivo);
  if($datos){
    $up = $datos->up;
    $net = $datos->net;
    $netmask = $datos->netmask;
    $minip = $datos->minip;
    $maxip = $datos->maxip;
    $deflt = $datos->deflt;
    $maxlt = $datos->maxlt;
    $router = $datos->router;
    $broadcast = $datos->broadcast;
    $subnetm = $datos->subnetm;
    $dns = $datos->dns;
  } else echo "Sintaxis XML inválida";
} else echo "Error abriendo dhcpconf.xml";

//Mostramos el formulario en función del valor de "up"
if($up==1){
    $chk="checked";
    $dsp="block";
}else{
    $chk="";
    $dsp="none";
}

//Guardamos en un array por separado los 4 valores que forman la ip
```

```
$netdiv=explode(".",$net);
$minipdiv=explode(".",$minip);
$maxipdiv=explode(".",$maxip);

//Formulario
echo "<form action=\"\" method=\"post\" name=\"dhcpconf\">
        <br />
        <h1>Configuraci&oacute;n DHCP</h1>
        <h3>DHCP activado:  <input id=\"activardhcp\" type=\"checkbox\"
onClick=\"habilita('#activardhcp','#parteOcultar')\" $chk/></h3>
        <br />
        <div id=\"parteOcultar\" style=\"display:$dsp\">
            <h3>Rango de IP</h3>
                <p>Direcci&oacute;n IP de inicio:
$netdiv[0].$netdiv[1].$netdiv[2].<input name=\"minipdiv\" type=\"text\"
value=$minipdiv[3] size=\"3\" maxlength=\"3\" /></p>
                <p>Direcci&oacute;n IP de fin:
$netdiv[0].$netdiv[1].$netdiv[2].<input name=\"maxipdiv\" type=\"text\"
value=$maxipdiv[3] size=\"3\" maxlength=\"3\" /></p>
            <h3>Tiempos de asignaci&oacute;n</h3>
                <p>Tiempo por defecto: <input name=\"deftime\" type=\"text\"
value=$deflt size=\"6\" maxlength=\"6\" /> segundos</p>
                <p>Tiempo m&aacute;ximo: <input name=\"maxtime\"
type=\"text\" value=$maxlt size=\"6\" maxlength=\"6\" /> segundos</p>
        </div>
        <input name=\"net\" type=\"hidden\" value=$net />
        <input name=\"netmask\" type=\"hidden\" value=$netmask />
        <input name=\"minip\" type=\"hidden\" value=$minip />
        <input name=\"maxip\" type=\"hidden\" value=$maxip />
        <input name=\"deflt\" type=\"hidden\" value=$deflt />
        <input name=\"maxlt\" type=\"hidden\" value=$maxlt />
        <input name=\"router\" type=\"hidden\" value=$router />
        <input name=\"broadcast\" type=\"hidden\" value=$broadcast />
        <input name=\"subnetm\" type=\"hidden\" value=$subnetm />
        <input name=\"dns\" type=\"hidden\" value=$dns />
        <br />
        <br />
        <input type=\"button\" value=\"Guardar cambios\"
onclick=\"xajax_escribirDhcp(xajax.getFormValues(dhcpconf),activardhcp.checke
d)\">
</form>";

?>

</body>
</html>
```

A.5.- portf.php

```
<html>
<head>
<link rel=\"stylesheet\" href=\"estilo.css\" type=\"text/css\">
</head>
<body>

<?php
```

```
//Lectura del xml
$i = 0;
$archivo = 'portforconf.xml';
if(file_exists($archivo)){
  $datos = simplexml_load_file($archivo);
  if($datos){
   foreach ($datos->regla as $regla) {
      $i = $i + 1;
      $interf[$i]=$regla->interf;
      $port[$i]=$regla->port;
      $ip[$i]=$regla->ip;
   }
  } else echo "Sintaxis XML inválida";
} else echo "Error abriendo portforconf.xml";


//Formulario
echo "<form action=\"\" method=\"post\" name=\"portfconf\">
    <br />
    <h1>Reglas de Port Fowarding</h1>
    <br />
    <table class=\"other\">
            <th>Interfaz</th>
            <th>Puerto</th>
            <th>IP destino:Puerto</th>";

for ($j=1;$j<=$i;$j++){
    echo    "<tr>
                <td><input type=\"text\" name=\"int$j\"
value=\"$interf[$j]\"></td>
                <td><input type=\"text\" name=\"p$j\"
value=\"$port[$j]\"></td>
                <td><input type=\"text\" name=\"ip$j\"
value=\"$ip[$j]\"></td>
            </tr>";
};

for ($k=$j;$k<=20;$k++){
    echo    "<tr>
                <td><input type=\"text\" name=\"int$k\" value=\"\"></td>
                <td><input type=\"text\" name=\"p$k\" value=\"\"></td>
                <td><input type=\"text\" name=\"ip$k\" value=\"\"></td>
            </tr>";
};

echo        "</table>
            <br />
            <input type=\"button\" value=\"Guardar cambios\"
onclick=\"xajax_escribirPortf(xajax.getFormValues(portfconf))\">
        </form>";
?>
<br />


</body>
</html>
```

A.6.- dns.php

```
<html>
<head>
<link rel="stylesheet" href="estilo.css" type="text/css">
<script type="text/javascript" src="scripts.js"></script>
<script type="text/javascript" src="jQuery.js"></script>
</head>
<body>

<?php

//Lectura del xml
$archivo = 'dnsconf.xml';
if(file_exists($archivo)){
  $datos = simplexml_load_file($archivo);
  if($datos){
    $up = $datos->up;
  } else echo "Sintaxis XML inválida";
} else echo "Error abriendo natconf.xml";

//Mostramos el formulario en función del valor de "up"
if($up==1) $chk="checked";
else $chk="";

echo "<form action=\"\" method=\"post\" name=\"dnsconf\">
        <br />
        <h1>Configuraci&oacute;n Proxy DNS</h1>
        <h3>Proxy DNS activado:  <input id=\"activardns\" type=\"checkbox\"
$chk/></h3>
        <br />
        <input type=\"button\" value=\"Guardar cambios\"
onclick=\"xajax_escribirDns(activardns.checked)\">
</form>";

?>
</body>
</html>
```

A.7.- funciones.php

```
<?php


$xajax = new xajax(); //instanciamos el objeto de clase xAjax
$xajax->configure('setCharEncoding','ISO-8859-1');
$xajax->configure('decodeUTF8Input',true);

//Función para cargar páginas en el div "parteCentral"
function cargarEnDiv($path, $destino){
    ob_start();
    include($path);
    $codigo = ob_get_contents();
    ob_end_clean();
```

```
    $respuesta = new xajaxResponse(); //instanciamos el objeto para generar
la respuesta con ajax
    $respuesta->setCharacterEncoding('ISO-8859-1');

    //Se inserta en código en el div id=$destino
    $respuesta->assign($destino,"innerHTML",$codigo);
    return $respuesta;
}


//Función para escritura de IP Forwarding en el XML
function escribirIpf($chkActivarIpf){
    $strXML = "<ip-for>\n";

    if ($chkActivarIpf==true)  $strXML .= "\t<up>1</up>\n";
    else  $strXML .= "\t<up>0</up>\n";

    $strXML .= "</ip-for>";

    $respuesta = new xajaxResponse(); //instanciamos el objeto para generar
la respuesta con ajax
    $respuesta->setCharacterEncoding('ISO-8859-1');

    $xmlFile = fopen('ipfconf.xml', 'w');
    fwrite($xmlFile, $strXML);
    fclose($xmlFile);
    $salida=shell_exec ( 'ruby ipf.rb');
    $respuesta->Alert($salida);

    return $respuesta;  //tenemos que devolver la instanciación del objeto
xajaxResponse
}


//Función para escritura de NAT en el XML
function escribirNat($formNat,$chkActivarNat){
    $salida="";

    $strXML = "<nat>\n";

    if ($chkActivarNat==true){
        $strXML .= "\t<up>1</up>\n";
        if($formNat["interfazpub"]=="") $salida="El campo interfaz no puede
quedar vac\xEDo.";
        else $strXML .= "\t<interf>".$formNat["interfazpub"]."</interf>\n";
    }else{
        $strXML .= "\t<up>0</up>\n";
        $strXML .= "\t<interf></interf>\n";
    }

    $strXML .= "</nat>";


    $respuesta = new xajaxResponse(); //instanciamos el objeto para generar
la respuesta con ajax
    $respuesta->setCharacterEncoding('ISO-8859-1');
```

```
    if($salida!="")  $respuesta->Alert($salida);
    else{
        $xmlFile = fopen('natconf.xml', 'w');
        fwrite($xmlFile, $strXML);
        fclose($xmlFile);
        $salida =shell_exec('ruby nat.rb');
        $respuesta->Alert($salida);
    }

    return $respuesta;  //tenemos que devolver la instanciación del objeto
xajaxResponse
}



//Función para escritura de DHCP en el XML
function escribirDhcp($formDhcp,$chkActivarDhcp){
    $salida="";
    $netdiv=explode(".",$formDhcp["net"]);
    $netinicio=$netdiv[0].".".$netdiv[1].".".$netdiv[2].".";

    $strXML = "<dhcp>\n";

    if ($chkActivarDhcp==true){
        $strXML .= "\t<up>1</up>\n";
        $strXML .= "\t<net>".$formDhcp["net"]."</net>\n";
        $strXML .= "\t<netmask>".$formDhcp["netmask"]."</netmask>\n";
        if($formDhcp["minipdiv"]=="") $salida="Los valores no pueden quedar
vac\xEDos.";
        else $strXML .=
"\t<minip>".$netinicio.$formDhcp["minipdiv"]."</minip>\n";
        if($formDhcp["maxipdiv"]=="") $salida="Los valores no pueden quedar
vac\xEDos.";
        else $strXML .=
"\t<maxip>".$netinicio.$formDhcp["maxipdiv"]."</maxip>\n";
        if($formDhcp["deftime"]=="") $salida="Los valores no pueden quedar
vac\xEDos.";
        else $strXML .= "\t<deflt>".$formDhcp["deftime"]."</deflt>\n";
        if($formDhcp["maxtime"]=="") $salida="Los valores no pueden quedar
vac\xEDos.";
        else $strXML .= "\t<maxlt>".$formDhcp["maxtime"]."</maxlt>\n";
        $strXML .= "\t<router>".$formDhcp["router"]."</router>\n";
        $strXML .= "\t<broadcast>".$formDhcp["broadcast"]."</broadcast>\n";
        $strXML .= "\t<subnetm>".$formDhcp["subnetm"]."</subnetm>\n";
        $strXML .= "\t<dns>".$formDhcp["dns"]."</dns>\n";
    }else{
        $strXML .= "\t<up>0</up>\n";
        $strXML .= "\t<net>".$formDhcp["net"]."</net>\n";
        $strXML .= "\t<netmask>".$formDhcp["netmask"]."</netmask>\n";
        $strXML .= "\t<minip>".$formDhcp["minip"]."</minip>\n";
        $strXML .= "\t<maxip>".$formDhcp["maxip"]."</maxip>\n";
        $strXML .= "\t<deflt>".$formDhcp["deflt"]."</deflt>\n";
        $strXML .= "\t<maxlt>".$formDhcp["maxlt"]."</maxlt>\n";
        $strXML .= "\t<router>".$formDhcp["router"]."</router>\n";
        $strXML .= "\t<broadcast>".$formDhcp["broadcast"]."</broadcast>\n";
        $strXML .= "\t<subnetm>".$formDhcp["subnetm"]."</subnetm>\n";
        $strXML .= "\t<dns>".$formDhcp["dns"]."</dns>\n";
```

```
    }

    $strXML .= "</dhcp>";

    $respuesta = new xajaxResponse(); //instanciamos el objeto para generar
la respuesta con ajax
    $respuesta->setCharacterEncoding('ISO-8859-1');

    if($salida!="")  $respuesta->Alert($salida);
    else{
        $xmlFile = fopen('dhcpconf.xml', 'w');
        fwrite($xmlFile, $strXML);
        fclose($xmlFile);
        $salida = shell_exec ( 'ruby dhcp.rb');
        $respuesta->Alert($salida);
    }

    return $respuesta;  //tenemos que devolver la instanciación del objeto
xajaxResponse
}



//Función para escritura de DNS en el XML
function escribirDns($chkActivarDns){
    $strXML = "<dns>\n";

    if ($chkActivarDns==true)  $strXML .= "\t<up>1</up>\n";
    else  $strXML .= "\t<up>0</up>\n";

    $strXML .= "</dns>";

    $respuesta = new xajaxResponse(); //instanciamos el objeto para generar
la respuesta con ajax
    $respuesta->setCharacterEncoding('ISO-8859-1');

    $xmlFile = fopen('dnsconf.xml', 'w');
    fwrite($xmlFile, $strXML);
    fclose($xmlFile);
    $salida=shell_exec ( 'ruby dns.rb');
    $respuesta->Alert($salida);

    return $respuesta;  //tenemos que devolver la instanciación del objeto
xajaxResponse
}



//Función para escritura de PORT FORWARDING en el XML
function escribirPortf($formPortf){
    $respuesta = new xajaxResponse(); //instanciamos el objeto para generar
la respuesta con ajax
    $respuesta->setCharacterEncoding('ISO-8859-1');
    $strXML = "<port-for>\n";
    for ($i=1;$i<=20;$i++){
        $int="int".$i;
        $p="p".$i;
```

```
        $ip="ip".$i;
        if ($formPortf[$int]!=""&&$formPortf[$p]!=""&&$formPortf[$ip]!=""){
            $strXML .= "\t<regla>\n";
                $strXML .= "\t\t<interf>".$formPortf[$int]."</interf>\n";
                $strXML .= "\t\t<port>".$formPortf[$p]."</port>\n";
                $strXML .= "\t\t<ip>".$formPortf[$ip]."</ip>\n";
            $strXML .= "\t</regla>\n";
        }
    }
    $strXML .= "</port-for>\n";


    $xmlFile = fopen('portforconf.xml', 'w');
    fwrite($xmlFile, $strXML);
    fclose($xmlFile);
    $salida = shell_exec ( 'ruby portf.rb');
    $respuesta->Alert($salida);

    return $respuesta;  //tenemos que devolver la instanciación del objeto
xajaxResponse

}


$xajax->register(XAJAX_FUNCTION, 'cargarEnDiv'); //asociamos las funciones
creadas anteriormente al objeto xajax
$xajax->register(XAJAX_FUNCTION, 'escribirIpf');
$xajax->register(XAJAX_FUNCTION, 'escribirNat');
$xajax->register(XAJAX_FUNCTION, 'escribirDhcp');
$xajax->register(XAJAX_FUNCTION, 'escribirDns');
$xajax->register(XAJAX_FUNCTION, 'escribirPortf');


$xajax->processRequest(); //el objeto xajax tiene que procesar cualquier
petición

?>
```

A.8.- scripts.js

```
//Función que muestra/oculta la capa "div" según esté activado/desactivado
"checkb"
function habilita(checkb,div){
    $(document).ready(function(){
        if ($(checkb).is(':checked')){
            $(div).show("slow");
        }else{
            $(div).hide("slow");
        }
    });

}

//Función que crea los input de forma dinámica para introducir las reglas de
port forwarding
num = 0;
```

```
function crearRegla(interf,port,ip) {
    num++;
    str="regla"+num;
    $(document).ready(function(){
        $("<div/>", {
            id: str
        }).appendTo("#reglas");

        str="#"+str;

        cssData = {
            'width': 'auto',
            'padding': '5px',
            'margin': '5px',
            'display': 'inline-block',
            'border-style': 'dotted',
            'border-width': 'thin'
        }
        $(str).css(cssData);

        $(str).append("sudo iptables -t nat -A PREROUTING -i  ");

        $("<input/>", {
            id: "int"+num,
            type: "txt",
            value: interf
        }).appendTo(str);

        $(str).append("  -p tcp --dport  ");

        $("<input/>", {
            id: "port"+num,
            type: "txt",
            value: port
        }).appendTo(str);

        $(str).append("  \ -j DNAT --to  ");

        $("<input/>", {
            id: "ip"+num,
            type: "txt",
            value: ip
        }).appendTo(str);
    });
}

//Función que borra los input de las reglas de port forwarding de forma
dinámica
function borrarRegla() {
    if(num>0){
        str="#regla"+num;
        $(document).ready(function(){
            $(str).remove();
        });
        num--;
    }
}
```

```
function prueba(){ $('#botonCrear').click(); }
```

A.9.- estilo.css

```
@charset "utf-8";
/* CSS Document */

 #parteSuperior {
    position:absolute;
    top:1%;
    left:0.6%;
    width:98.8%;
    height:19%;
    background:#669966;
 }

 #menu {
    position:absolute;
    top:21%;
    width:18%;
    height:78%;
    left:0.6%;
    background:#CCCC66;
    overflow:auto;
 }

 #parteCentral {
    position:absolute;
    top:21%;
    left:19.2%;
    width:78.2%;
    height:78%;
    background:#CCDDCC;
    padding-left:2%;
    overflow:auto;
 }

 #logo{
    position: relative;
    height: 50%;
    top: 25%;
    bottom: 25%;
    right: 2%;
    float: right;
 }

 div.reglas{
    width: auto;
    padding: 5px;
    margin: 5px;
    display: inline-block;
    border-style: dotted;
    border-width: thin;
 }
```

```
img {
     max-width:100%;
     max-height:100%;
     float: right;
}

table {
   position:absolute;
   top:5%;
   left:10%;
   width:80%;
   height:50%;
   border:none;
}

table.other {
   position: relative;
}

table.other th {
   text-align: left;
}

body {
   font-family:sans-serif;
}

font.menu{
   font-weight:bold;
}

font.menu:hover {
   color:#FFFFFF;
   cursor:pointer;
}
```