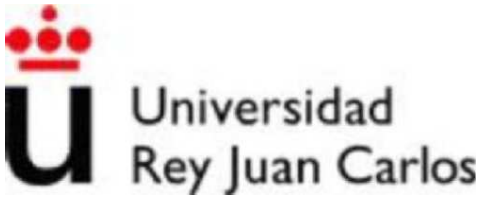


Ingeniería Informática

Escuela Superior de Ciencias Experimentales y Tecnología

Curso académico 2005-2006

Proyecto Fin de Carrera

NetWeb. Un nuevo enfoque para el interfaz de NetKit

Autor: Miguel Ángel Hernando Fernández
Tutor: Pedro de las Heras Quirós

Septiembre 2006

A mis padres y hermano

Agradecimientos

Este proyecto no hubiera sido posible sin el apoyo del Grupo de Sistemas y Comunicaciones (GSyC) de la Universidad Rey Juan Carlos y en especial a Pedro de las Heras, mi tutor, a quien le agradezco la confianza que ha depositado en mí.

Gracias a mis padres por darme la oportunidad de estudiar lo que realmente me gusta, sin su apoyo no hubiera podido llegar tan lejos.

A Raúl, mi hermano, que sin él mi vida hubiera sido muy tranquila.

Gracias a mi familia porque han sido un gran punto de apoyo durante toda mi vida.

A Noemí por haberme dado lo que nadie me dio, gracias por apoyarme.

A mis amigos, Rafa, Mila, Alberto, Dani, David y un gran etcétera. Gracias por haberme hecho pasar los mejores momentos de mi vida.

A Jesús por enseñarme a ver la vida de otra forma.

A mis compañeros de la Universidad, que sin ellos la Universidad hubiera sido muy aburrida.

Resumen

En los últimos años, Internet, y en especial el Web, ha cambiado el modo de vida de muchas personas y empresas, tanto es así que Internet es un medio inevitable de transmisión y recepción de información, ¿quién no lee el periódico por Internet?, ¿quién no tiene correo electrónico?. Estas preguntas, formuladas hace unos años podían ser respondidas únicamente por algunas empresas e instituciones publicas, por suerte actualmente conectarse a Internet es una acción cotidiana para cualquier persona.

Coloquialmente, se conoce como Internet al acceso a la información existente en la red mediante el Web. Cualquier contenido disponible en Internet y que se desee divulgar, inevitablemente debe estar accesible mediante el Web. Gracias a ello, Internet ha avanzado a pasos agigantados en la última década.

Este proyecto intenta realizar un sistema enfocado a la enseñanza y apoyo de Ingenieros. En titulaciones como Ingenierías Informáticas y Telecomunicaciones existen asignaturas en las cuales se explican los conceptos básicos de las redes de computadores, en concreto como funciona Internet. El problema que se encuentran muchas veces los docentes, es que el alumno no puede asimilar bien los conceptos teóricos con las prácticas existentes. Tener un laboratorio con infinidad de máquinas, todas ellas conectadas formando una red de ordenadores para hacer pruebas, puede ser un trabajo costoso, tanto en términos económicos como de espacio.

Actualmente, existe infinidad de software que permite realizar este tipo de simulaciones de entornos de red. El alumno normalmente se encuentra con bastantes dificultades a la hora de poder utilizar software demasiado complejo para poder diseñar entornos de red bastantes sencillos.

Netkit es un conjunto de aplicaciones que permite hacer simulaciones de máquinas GNU/Linux conectadas en red. El problema que frena el uso de Netkit a los alumnos, es la compleja instalación y ejecución de las simulaciones de red, todo esto unido a que Netkit no dispone de un interfaz de usuario. Actualmente, existen distintos sistemas que dotan a Netkit de un interfaz de usuario, aunque no eliminan al alumno la tarea de instalación y configuración de Netkit, tarea ardua para un usuario inexperto.

Para ello, en este proyecto se va a tomar un modelo de desarrollo en el cual el usuario pueda interactuar con Netkit sin necesidad de configurarlo ni realizar ninguna instalación de software adicional. ¿ Cómo se puede conseguir esto ?, pues realizando un interfaz de usuario basado en el Web. Todo el mundo posee un navegador Web, por ello se va a dotar a Netkit de un interfaz de usuario aplicado a este entorno, relegando la configuración e instalación del software de Netkit a otra máquina gestionada por un administrador experto. Todo esto no sería posible sin la evolución que se ha ido aplicando al Web en los últimos años, sobre todo gracias al nuevo concepto de entender el Web como una nueva forma de realizar aplicaciones, el **Web 2.0**.

Índice general

1. Introducción **9**
- 1.1. Netkit . . . 9
- 1.2. Internet . . . 9
- 1.3. Introducción al modelo del Web . . . 10
  - 1.3.1. HTML . . . 10
  - 1.3.2. Protocolo HTTP . . . 10
  - 1.3.3. Navegador Web . . . 13
  - 1.3.4. Servidor Web . . . 13
- 1.4. Evolución del Web . . . 14
- 1.5. Tecnologías Utilizadas aplicadas al Web . . . 15
  - 1.5.1. Ajax . . . 15
  - 1.5.2. Javascript . . . 15
  - 1.5.3. XML . . . 15
  - 1.5.4. CSS . . . 16
- 1.6. GNU/Linux . . . 16
  - 1.6.1. Kernel de Linux . . . 17
  - 1.6.2. GNU . . . 17
- 1.7. Licencia GPL . . . 17
- 1.8. Python . . . 17
- 1.9. Contenido de la Memoria . . . 18

2. Objetivos **19**
- 2.1. NetKit . . . 19
- 2.2. Interfaz de Usuario para Netkit . . . 21
  - 2.2.1. NetEdit . . . 21
  - 2.2.2. NetGUI . . . 22
- 2.3. Interfaz Web para Netkit . . . 23
- 2.4. Multiplataforma . . . 24
- 2.5. Ejecución de Netkit en Remoto . . . 25
- 2.6. Desarrollo usando Ajax . . . 26
- 2.7. Configuración y Comunicación mediante XML . . . 27
- 2.8. Desarrollo siguiendo un control de Versiones . . . 27
- 2.9. Licencia Software de Software Libre . . . 27
- 2.10. Adquisición de experiencia en programación . . . 28
- 2.11. Experiencia en la realización de un proyecto . . . 28

3. Metodología empleada **29**
3.1. Introducción . . . 29
3.2. Desarrollo en espiral . . . 29
3.2.1. Tareas . . . 30
3.2.2. Ventajas . . . 31
3.2.3. Inconvenientes . . . 31
3.2.4. Aplicación de este modelo al desarrollo de este proyecto . . . 31
3.3. Desarrollo basado en prototipos . . . 32
3.3.1. Aplicación de este modelo al desarrollo de este proyecto . . . 32

4. Diseño e Implementación **35**
4.1. Introducción . . . 35
4.2. Arquitectura . . . 35
4.2.1. Interfaz de Usuario . . . 36
4.2.2. Servidor Web . . . 36
4.2.3. Nodos Netkit . . . 37
4.2.4. Gestión de Nodos Netkit . . . 37
4.3. Prototipo 1: Construcción del Interfaz de Usuario . . . 38
4.3.1. Especificación . . . 38
4.3.2. Diseño . . . 39
4.3.3. Implementación . . . 40
4.4. Prototipo 2: Interacción entre Usuario y Servidor Web . . . 58
4.4.1. Especificación . . . 58
4.4.2. Diseño . . . 58
4.4.3. Implementación . . . 61
4.5. Prototipo 3: Ejecución Remota de los nodos virtual . . . 65
4.5.1. Especificación . . . 65
4.5.2. Diseño . . . 66
4.5.3. Implementación . . . 67
4.6. Prototipo Final: Acceso a los nodos por medio del Web . . . 71
4.6.1. Especificación . . . 71
4.6.2. Diseño . . . 71
4.6.3. Implementación . . . 72

5. Validación y Pruebas **79**
5.1. Prueba1: Configuración de una Red Local . . . 79
5.2. Prueba2: Encaminamiento . . . 81
5.3. Prueba3: ARP . . . 82
5.4. Prueba4: Encaminamiento Avanzado . . . 84

6. Conclusiones **87**
6.1. Conocimientos Adquiridos . . . 87
6.2. Posibles usos de NetWeb . . . 88
6.3. Trabajos Futuros . . . 88

Capítulo 1

Introducción

En este capítulo se introducirán al lector todos los conceptos necesarios para la lectura y comprensión del documento.

1.1. Netkit

Desarrollado por la Universidad de Roma, Netkit[1] fue creado para disponer de un entorno virtual donde poder configurar y realizar pruebas experimentales de topologías de Red.

La creación del entorno virtual esta realizada por medio de un gran conjunto de scripts de Shell para GNU/Linux. Estos scripts permiten crear nodos virtuales de máquinas GNU/Linux las cuales podemos interconectar, configurar y manipular como si fuera un ordenador físico. La ventaja que esto conlleva es que nos permite hacer simulaciones de entornos de red con un coste bastante bajo.

Netkit es software libre, bajo licencia GPL y esta basado en User Mode Linux (UML)[2].

1.2. Internet

Internet es una gran red de ordenadores, dicha red está formada por la comunicación de millones de ordenadores de todos los rincones del mundo. Estos ordenadores están conectados usando diversos protocolos, entre ellos TCP/IP, el cual es el más destacado.

Internet nació del conflicto entre los Estados Unidos y la URSS, conocido popularmente como guerra fría. La primera versión de la actual *Internet*, fue Arpanet. Arpanet fue creada por el departamento de Defensa de los Estados Unidos para poder comunicar los distintos órganos del gobierno de aquel país en la década de los 60. Esta red fue evolucionando hasta que en 1983 se estableció como espina dorsal el protocolo TCP/IP que conocemos actualmente.

Cabe destacar que los *Padres* de Internet Tim Berners-Lee, Larry Roberts, Vinton Cerf y Robert Kahn fueron galardonados con el premio Príncipe de Asturias de Investigación y Humanidades en el año 2002.

Tradicionalmente se cree que Internet únicamente esta compuesta por el Web, aunque lo que se conoce popularmente como Internet es un conjunto de protocolos, véase SMTP,IRC,SSH,FTP, etcétera.

[1] http://netkit.org

[2] http://user-mode-linux.sourceforge.net/

Como dato se debe destacar que la primera conexión plena desde España a la Internet tuvo lugar a mediados del año 1990, como un servicio experimental de RedIRIS que, a finales de ese año, interconectaba tan sólo cuatro centros: Fundesco, Departamento de Ingeniería Telemática (Universidad Politécnica de Madrid), Centro de Informática Científica de Andalucía y CIEMAT.

1.3. Introducción al modelo del Web

1.3.1. HTML

Acrónimo de **H**yper**T**ext **M**arkup **L**anguage. Es el lenguaje de marcado usado para estructurar la información que se desea mostrar. Dicho lenguaje es el formato estándar usado para la visualización de las páginas Web.

1.3.2. Protocolo HTTP

El Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que permite el intercambio de información entre los clientes Web (Navegadores) y los servidores HTTP. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web (WWW).

Desde el punto de vista de las comunicaciones, el protocolo HTTP se sitúa usando los servicios de TCP. Por defecto el puerto TCP usado es el 80, aunque puede usar cualquier otro puerto para establecer una conexión http, por ejemplo el 8080 que también es usualmente usado para dicho protocolo.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado,en el caso que no hubiera error.

Este protocolo permite usar una serie de métodos para indicar la finalidad de la petición. Se basa en otros conceptos y estándares como Uniform Resource Identifier (URI), Uniform Resource Location (URL) y Uniform Resource Name (URN), para indicar el recurso al que hace referencia la petición. Los mensajes se pasan con un formato similar al usado por el Internet Mail y el Multipurpose Internet Mail Extensions (MIME).

Métodos de HTTP

Para la interacción entre el navegadores y el servidor Web por medio del protocolo HTTP existen varios métodos, los cuales permiten realizar distintas operaciones. Estos métodos serán usados dependiendo de la información que se quiera enviar/solicitar al servidor.

- **OPTIONS**. Este método representa una petición de información sobre las opciones que están disponibles en el servidor Web. Si la URI es un *, entonces la petición se aplica al servidor como un conjunto.

- **GET**. El método GET requiere la devolución de información al cliente identificada por la URI. El método GET es el método habitual para la petición de paginas Web usadas por el navegador.

- **HEAD**. El método HEAD es igual que el método GET, salvo que el servidor no tiene que devolver el contenido, sino únicamente las cabeceras HTTP de la respuesta. Por ejemplo este método se usa para validar los link o saber las modificaciones recientes de una pagina HTML.
- **POST**. El método POST se usa para hacer peticiones en las que el servidor destino acepta el contenido de la petición. Este método se usa para enviar información al servidor por medio de formularios HTML.
- **PUT**. El método PUT permite guardar el contenido de la petición en el servidor bajo la URI de la petición.
- **DELETE**. Este método se usa para que el servidor borre el recurso indicado por la URI de la petición. No se garantiza al cliente que el resultado se satisfactorio.
- **TRACE**. Este método es usado para saber si existe el receptor del mensaje y usar la información para hacer un diagnóstico.

Existe una extensión de métodos de HTTP, denominado **WebDAV**. Aunque estos métodos van a ser irrelevantes en la realización de este proyecto.

Respuestas HTTP

Cuando el servidor Web recibe una petición de información de un navegador por medio de un método y la URI relacionada al recurso que desea acceder, el servidor Web debe poder indicar al navegador el resultado de la petición. Para ello el servidor web crea un mensaje con un código numérico que indica el resultado de la petición. En caso que la petición sea correcta también se adjunta en el mensaje el resultado de dicha petición.

El código de estado es un número de 3 dígitos que indica si la petición ha sido atendida correctamente o no. En caso de no haber sido atendida correctamente, se indica el motivo. Dichos códigos se dividen en:

- **1xx**. Informativo. Estas series de respuestas indican una respuesta provisional.
  - 100. Continuar.
  - 101. Cambio de protocolo.

  [...]
- **2xx**. Éxito. La petición recibida por el navegador ha sido atendida correctamente.
  - 200. Éxito
  - 201. Creado.
  - 202. Aceptado.

  [...]
- **3xx**. Redirección. Para realizar la petición se deben realizar más acciones.
  - 300. Múltiples elecciones
  - 301. Movido Permanentemente.

- 302. Movido Temporalmente.

[...]

- **4xx**. Error del cliente. No se puede realizar la petición al estar mal formada.
  - 400. Petición errónea.
  - 401. No autorizado.
  - 402. Pago requerido.

[...]

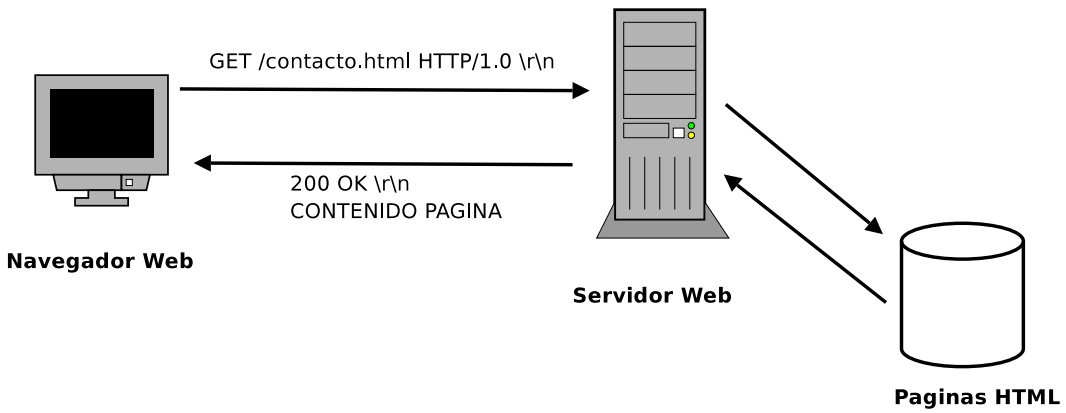
- **5xx**. Error del servidor. El servidor tiene un error al procesar la petición.
  - 500. Error interno del servidor.
  - 502. Puerta de enlace errónea.
  - 503. Servicio no disponible

[...]

Téngase en cuenta que no se ha enumerado ni definido todos los códigos de respuestas.

Modo de Funcionamiento

A continuación mostraremos el funcionamiento del protocolo HTTP y como interactúa el Cliente (Navegador) con el Servidor Web.



Interacción Cliente-Servidor usando HTTP

1. El cliente indica por medio del navegador Web la página que quiere solicitar, por ejemplo: http://deltatec-systems.com/contacto.html

2. A continuación el navegador Web abre una conexión TCP al puerto 80 del servidor deltatec-systems.com.

3. Una vez abierta la conexión, el navegador pediría la página por medio del protocolo HTTP, mediante el método GET.

```
GET /contacto.html HTTP/1.0  \r\n
```

4. El servidor recibirá la petición de solicitud de la página, leerá dicha página del lugar donde la tenga almacenada, normalmente en su disco duro, y le responderá al cliente.

```
200 OK \r\n
CONTENIDO DE LA PAGINA
```

5. Cuando el navegador reciba la respuesta de la página pedida, mostrará el contenido del HTML al usuario por medio de su interfaz.

1.3.3. Navegador Web

Un navegador web, o web browser, es una aplicación software que permite al usuario visualizar documentos HTML. También, el navegador Web, es el encargado de interactuar con el servidor.

La funcionalidad principal del navegador web es la de visualizar y renderizar el HTML recibido de la petición que se ha realizado al servidor Web. Cabe recordar que para realizar esta solicitar y recibir el documento HTML se realiza por medio del protocolo HTTP.

Los primeros navegadores únicamente soportaban un versión muy simple de HTML. Actualmente los navegadores web no solo se encargan de renderizar HTML, sino que también son capaces de visualizar imágenes,vídeos y sonidos.

En la actualidad existen diversos navegadores Web, a continuación describiremos los más importantes:

- **Mozilla Firefox**[3]. Es un navegador web multiplataforma,quiere decir que esta disponible es diversos sistemas operativos. Es unos de los navegadores mas extendidos ya que su licencia es software libre y cumple todos los estándares impuestos por la World Wide Web Consortium (W3C)[4].

- **Internet Explorer**. Es el navegador de Microsoft[5]. Únicamente esta disponible para la plataforma Windows y MAC. Es software propietario, por tanto su código fuente no esta liberado ni disponible a ningún usuario.

- **Opera**. Es un navegador creado por la empresa Opera Software [6]. Opera es un navegador gratuito pero no Libre.

Durante el desarrollo del proyecto se ha usado el navegador Mozilla Firefox, el cual hemos creído mas acertado porque es el mas estricto usando los estándares para renderizar HTML. No obstante, el proyecto ha sido testeado en los demás navegadores, con resultados óptimos.

1.3.4. Servidor Web

Un servidor web es un programa que implementa el protocolo HTTP. En esencia un servidor Web es un sistema de ficheros, el cual sirve unos ficheros (paginas HTML) por medio de la red.

[3]http://www.mozilla.com/firefox/
[4]http://www.w3c.org
[5]http://www.microsoft.com
[6]http://opera.com

Los primeros servidores Web que aparecieron fueron únicamente simples programas que servían paginas HTML a los navegadores, actualmente los servidores Webs son bastantes mas complejos. Esta complejidad ha derivado en que no únicamente sirven paginas HTML, sino que también puede interactuar con el cliente (navegador) para poder crear aplicaciones destinadas al Web.

Actualmente en el mercado existen distintos servidores web

- **Apache**[7]. Este servidor Web es Software Libre y multiplataforma, el cual dispone de una gran cantidad de plugins (extensiones). Todas estas características han contribuido a ser el servidor Web más usado en la red, con una presencia del 70 %.
- **Internet Information Services**. Conocido también por sus siglas *ISS*, es el servidor Web de Microsoft que únicamente esta disponible para la plataforma Windows.

Para la realización de este proyecto hemos creído oportuno realizarlo sobre el servidor Web Apache, ya que era el Servidor que mas se amoldaba a nuestros intereses.

1.4. Evolución del Web

En los inicios del web, nos encontramos con páginas HTML totalmente estáticas, con las cuales el usuario no podía interactuar con el servidor Web. Con el paso del tiempo, el avance tecnológico y la popularidad de Internet, el web ha ido tomando un nuevo enfoque. Desde los inicio del Web hasta nuestros días, ha habido distintas definiciones del mismo, las cuales detallaremos a continuación.

- **Web 1.0**. Es llamado al Web tradicional, páginas HTML estáticas donde el usuario no puede interactuar con el servidor.
- **Web 1.5**. El Web Tradicional evolucionó a una nueva concepción de la navegación Web. Ahora las páginas ya no eran estáticas , sino que se creaban dinámicamente, por ejemplo consultando una parte del HTML en una Base de Datos. A partir de este momento y en adelante, el servidor Web llevará la lógica de la aplicación. Muchos lenguajes como PHP, ASP, JSP ayudaron a que los servidores Web pudieran soportar esta evolución del Web.
- **Web 2.0**. Acuñado por Dale Dougherty de O'Reilly Media. El Web 2.0 puede definirse como la transacción que se ha dado a las aplicaciones normales de escritorio a las aplicaciones Web. Como en el Web 1.5, el Web 2.0 el usuario puede interactuar con el servidor por medio de un interfaz muy parecido al que tendría si estuviera ejecutando una aplicación normal de escritorio. La empresa Google [8], esta lanzando muchos servicios basados en el Web 2.0. Un ejemplo muy claro de aplicaciones basadas en el Web 2.0 puede ser
  - Google Maps *http://maps.google.com*
  - Google Spreadsheets *http://spreadsheets.google.com/*

En este proyecto intentaremos realizar una aplicación Web basada en el modelo de Desarrollo Web 2.0.

[7] http://apache.org
[8] http://google.es

1.5. Tecnologías Utilizadas aplicadas al Web

A continuación se describirán las tecnologías utilizadas para la realización de este proyecto.

1.5.1. Ajax

Ajax, cuyo acrónimo corresponde a **A**synchronous **J**avaScript **A**nd **X**ML. En realidad Ajax no es una tecnología única sino que es un conjunto de técnicas para el desarrollo de aplicaciones Web. La idea fundamental de Ajax, que cambia con respecto al modelo tradicional de las aplicaciones Web, es que con Ajax se mantiene una comunicación asíncrona con el servidor, el cual lleva todo la lógica de la aplicación.

Ajax,esta compuesto por tres técnicas que durante años se han usado por separado pero actualmente están siendo usadas en conjunto para el desarrollo de las nuevas aplicaciones Web.

- **Javascript**. Es el lenguaje utilizado para mostrar la información deseada y darle al navegador funcionalidad adicional.
- **XML**. Es el formato utilizado para la interacción con el servidor.
- **HTTPRequest**. Es un objeto que permite el intercambio asíncrono de información con el servidor.

Actualmente la mayoría de los navegadores existentes en el mercado soportan la combinación dichas técnicas que comprenden Ajax.

1.5.2. Javascript

Javascript es un lenguaje interpretado orientado al web, su sintaxis es muy similar a Java o C.

El lenguaje fue creado por Brendan Eich cuando trabajaba en Netscape. Javascript es un lenguaje que va embebido en el propio HTML. Actualmente la mayoría de los navegadores soportan javascript.

Se trata de un lenguaje interpretado por el navegador, a diferencia de otros lenguajes de servidor como PHP,ASP o JSP. Está orientado a objetos, con una serie de limitaciones, cuyo cometido principal es prevenir que se altere el sistema ficheros del cliente, por lo que no puede leer, escribir, crear, borrar o listar ficheros. Actualmente gracias a incorporación del WEB 2.0, javascript esta tomando mucho protagonismo para la creación de Aplicaciones Web.

1.5.3. XML

Xtensible **M**arkup **L**anguage, en castellano lenguaje de marcado extensible de etiquetas, desarrollado por la W3C. XML es un estándar creado para el intercambio de información estructurada entre diferentes plataformas y aplicaciones. Actualmente es usado tanto en editores, Bases de Datos, Aplicaciones en Red, etcétera. Los objetivos por los que fue creado XML fueron los siguiente:

- Que fuese fácil de leer y editar
- Que fuera idéntico a la hora de servir, recibir, y procesar la información del HTML para aprovechar toda la tecnología implantada de este.

- Que fuese fácil de implantar, programar y aplicar a los distintos sistemas.
- Que fuera extensible, para que lo puedan utilizar en todos los campos del conocimiento.

A simple vista, un documento XML tiene un aspecto muy similar al HTML. Al igual que el HTML, se basa en un documento de texto plano en los que se usan etiquetas para delimitar los elementos de los documentos.

A continuación se mostrará un ejemplo de un documento XML:

```
<?xml version="1.0"?>
<!DOCTYPE MENSAJE SYSTEM "mensaje.dtd">
<mensaje>
        <remitente>
                <nombre>Miguel A. Hernando</nombre>
                <mail>mhernand@gsyc.escet.urjc.es</mail>
        </remitente>
        <destinatario>
                <nombre>Josefina Fernandez</nombre>
                <mail>josefina.fernandez.sanchez@gmail.com</mail>
        </destinatario>
        <asunto>Hola Mama</asunto>
        <texto>
                <parrafo>¿Hola que tal? Hace <enfasis>mucho</enfasis> que
                no escribes. </parrafo>
        </texto>
</mensaje>
```

XML será el formato que se utilizará en este proyecto para el intercambio de información entre los distintos módulos.

1.5.4. CSS

El lenguaje HTML está limitado a la hora de aplicarle forma a un documento. Esto es así porque fue concebido para otros usos (científicos sobretodo), distinto a los actuales, mucho más amplios.

Las CSS complementan a otros lenguajes de descripción de páginas utilizados para publicar documentos en la Web, como HTML o XML, con el propósito principal de permitir la separación entre el contenido de las páginas y su forma de presentación.

El lenguaje de las Hojas de Estilo está definido en la Especificaciones CSS1 y CSS2 del World Wide Web Consortium (W3C) y, por lo tanto, es un estándar aceptado por toda la industria relacionada con la Web.

1.6. GNU/Linux

GNU/Linux (GNU con Linux) es la denominación defendida por Richard Stallman y otros para el sistema operativo que utiliza el kernel Linux en conjunto con las aplicaciones de sistema creadas por el proyecto GNU. Comúnmente este sistema operativo es denominado como Linux, aunque esta denominación no es correcta.

Gracias a la licencia libre y la unión del kernel de Linux y las herramientas GNU, han hecho del sistema operativo GNU/Linux un sistema muy popular, llegando incluso a ser el sistema utilizado en algunas comunidades autónomas, como La Junta de Extremadura.

GNU/Linux será el Sistema Operativo en el cual se desarrollará, probará y escribirá este proyecto.

1.6.1. Kernel de Linux

El kernel de Linux fue creado por un estudiante llamado Linus Torvalds. Linus creo el kernel de linux para poder ejecutar un sistema operativo de Tipo Unix en una máquina con arquitectura de Intel 80386. El kernel de Linux implementa el estándar POSIX de llamadas al sistema y esta escrito en C, lenguaje en el cual estaba escrito tanto Minix como Unix. El proyecto nació en 1991, cuando Linus escribió un famoso mensaje en el grupo de noticias de Minix, exponiendo que estaba desarrollando un núcleo totalmente libre y bajo licencia GPL. A raíz de este mensaje, mucha gente se intereso en el desarrollo del kernel de Linux.

1.6.2. GNU

GNU, GNU not Unix, es un proyecto creado por Richard Stallman que tiene como objetivo la creación de un sistema operativo totalmente libre. GNU adopto el kernel de Linux como núcleo de su sistema operativo, añadiéndole sus herramientas como bash, gcc y otras que hacían del kernel de Linux un sistema operativo usable.

1.7. Licencia GPL

Es la licencia por excelencia del mundo del Software Libre, dicha licencia obliga a todo desarrollo escrito bajo ella la divulgación del código fuente del software desarrollado.

Esta licencia fue creada por la Free Software Foundation [9] a mediados de los 80 como parte del ideario de esta organización de propugnar el software libre.

La licencia sólo habla de la distribución de software y nunca del uso. Por lo tanto es completamente posible utilizar versiones modificadas de un programa sin que se imponga la necesidad de compartir de ninguna manera esas modificaciones.

- Licencia original. http://www.gnu.org/copyleft/gpl.html
- Traducción al castellano. http://gugs.sindominio.net/licencias/gples.html

1.8. Python

Python es un lenguaje de programación Orientado a Objetos e interpretado. Actualmente es un lenguaje muy extendido y habitualmente es comparado, entre otros lenguajes, con Perl y Java. La licencia del interprete de python es Software Libre. Por curiosidad se comenta que su nombre viene de los celebres humoristas llamados Monty Python.

Python permite reubicar el código en módulos, los cuales se pueden reutilizar en cualquier programa realizado en python. Para ello hay módulos de GUI, tratamientos de E/S,sockets, etcétera.

[9] http://www.fsf.org/

Python es un lenguaje interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa. También es una calculadora muy útil.

Lo que diferencia a Python de otros lenguajes es que es mucho más fácil de codificar, leer y tiene gran cantidad de módulos existentes para poder ser utilizados.

Python ha sido elegido como lenguaje de programación para realizar parte de la lógica de la aplicación, dicha lógica estará alojada en el Servidor Web.

1.9. Contenido de la Memoria

Después de realizar una introducción a diversos conceptos que serán usados para el desarrollo de este proyecto, veremos el capítulo de Objetivos. Dicho capítulo, intenta explicar detalladamente cuales son los objetivos principales marcados para la realización de este proyecto.

Una vez concluido el capítulo de Objetivos, este documento contiene el capítulo de Metodología empleada, el cual explicará razonadamente el modelo de desarrollo seguido para la realización de este proyecto.

Avanzando un poco mas en el documento, nos encontramos con el capítulo de Diseño e Implementación, el cual explicará en detalle el diseño, la arquitectura y el desarrollo del proyecto.

A continuación se detallará el capítulo de Pruebas y Validaciones en el cual se expondrán pruebas reales realizadas al Proyecto que corroboran que su desarrollo se ha realizado con éxito.

Después de Pruebas y Validaciones, se encuentra el capítulo de Conclusiones y Trabajos futuros, el cual expondrá las conclusiones sacadas con la realización de este proyecto y cuales pueden ser los trabajos futuros que pueden derivar del mismo.

Y para finalizar se encuentra el capítulo de Bibliografía, el cual detalla la bibliografía utilizada para el desarrollo de este proyecto.

Capítulo 2

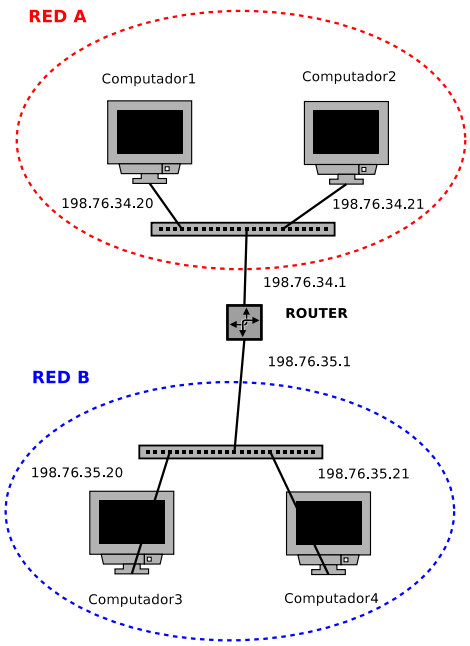
Objetivos

En este capítulo hablaremos de los objetivos que se han planteado para la realización de este proyecto.

2.1. NetKit

Netkit, es un conjunto de herramientas que permiten la creación de computadores virtuales y poder interconectarlos entre si para poder formar topologías de red. Para poder utilizar Netkit es necesario tenerlo instalado en una máquina GNU/Linux, esta instalación incluye software necesario para poder realizar todas estas tareas.

A continuación mostraremos un ejemplo de un diseño de red:



Entorno de Red

Como se puede observar existen dos redes de Área Local. La **Red A** contiene al computador1, computador2 y el Router, mientras que la **Red B** contiene al computador3, computer4 y al Router.

Con el diseño de red expuesto anteriormente, vamos a comprobar como se configuraría este entorno mediante Netkit. Lo primero que hay que hacer, es lanzar cada nodo indicándole la red a la que pertenece. Para ello usaremos el comando **vstart**, incluido en Netkit, de la siguiente forma:

```
vstart computer1 --eth0=A
vstart computer2 --eth0=A
vstart router --eth0=A --eth1=B
vstart computer3 --eth0=B
vstart computer4 --eth0=B
```

El primer argumento que recibe vstart es el nombre del nodo, y los demás son los interfaces de Red a las cuales pertenece el nodo. Téngase en cuenta que cada Red Local va a poseer un identificador, en nuestro caso A ó B. En el caso que hubiera más redes, los identificadores serían C, D ....

Ejemplo de Ejecución de Netkit

Como se puede observar en la imagen anterior, cada máquina virtual lanzada posee un terminal que permite interactuar con el usuario por medio de un Shell Bash.

Una vez que están ejecutados los nodos, se procedería a interactuar con ellos por medio de un Shell de Bash. Téngase en cuenta que cada nodo de Netkit ejecuta un sistema operativo GNU/Linux y su interfaz es similar al que nos podríamos encontrar en cualquier máquina GNU/Linux.

Lo primero que haríamos es asignarles una IPa cada una de las máquinas que estamos ejecutando mediante el terminal que disponen para ello. El comando **ifconfig** nos permitirá asignar direcciones IPa cada uno de los interfaces de las máquinas.

```
root@computer1:~# ifconfig eth0 up 198.76.34.20
root@computer2:~# ifconfig eth0 up 198.76.34.21
root@computer3:~# ifconfig eth0 up 198.76.35.20
root@computer4:~# ifconfig eth0 up 198.76.35.21
root@router:~# ifconfig eth0 up 198.76.34.1
root@router:~# ifconfig eth1 up 198.76.35.1
```

Una vez que cada máquina tiene asignada su dirección IP, es necesario configurar las tablas de encaminamiento para que las máquinas de las Red A puedan comunicarse con las de la Red B. Para ello deberemos configurar a computer1 y computer2 que tengan de Gateway a 198.76.34.1 y que las máquinas computer3 y computer4 tengan a 198.76.35.1. Para ello usaremos el comando router.

```
root@computer1:~# route add -net default gw 198.76.34.1
root@computer2:~# route add -net default gw 198.76.34.1
root@computer3:~# route add -net default gw 198.76.35.1
root@computer4:~# route add -net default gw 198.76.35.1
```

El router no encamina hacía otras redes, así que no es necesario configurar su tabla de encaminamiento.

Una vez que se hayan realizado todas las pruebas pertinentes con la red creado, sería necesario apagar cada uno de los nodos. Para ello Netkit dispone de un herramienta llamada **vhalt** que es ejecutada en la máquina donde están ejecutándose los nodos.

```
root@trinity:~# vhalt computer1
root@trinity:~# vhalt computer2
root@trinity:~# vhalt computer3
root@trinity:~# vhalt computer4
root@trinity:~# vhalt router
```

2.2. Interfaz de Usuario para Netkit

Para poder ejecutar Netkit, es necesario poder instalarlo en una máquina GNU/Linux. La configuración del entorno de Red que se quiere simular es algo complicado para cualquier usuario.

Para facilitar el uso de Netkit es necesario un interfaz que le permita al usuario poder manejar y configurar los entornos de red que quiere diseñar. Muchas veces el terminal de GNU/Linux puede repeler a un usuario final el usar una herramienta tan potente como puede ser NetKit.

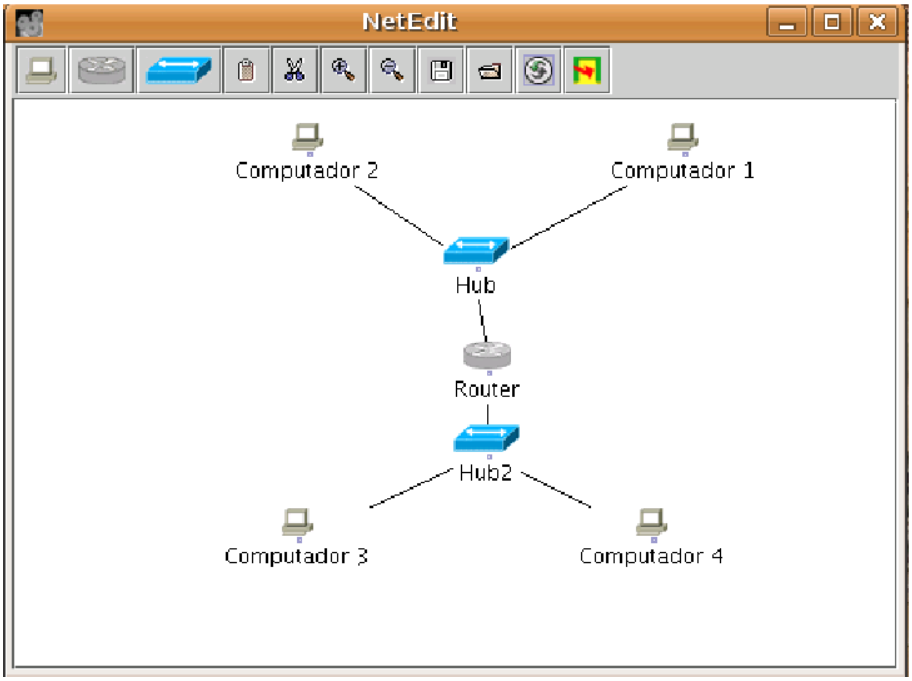
Existen varios proyectos software que han proporcionado a Netkit un interfaz que permite al usuario centrarse en el uso únicamente de la aplicación y no de la instalación y la configuración. Para la realización de este proyecto se toma como base varios interfaces de usuario para Netkit, los cuales se expondrán a continuación.

2.2.1. NetEdit

Desarrollado por la Universidad de Roma, Netedit [1] permite el diseño de redes por medio de una aplicación gráfica. Esta aplicación, realizada en Java, permite al usuario realizar los diseños de la redes en un entorno de desarrollo amigable. El usuario únicamente se encargaría de añadir los computadores, hubs o router, y posteriormente realizar los enlaces oportunos. Una vez que el diseño estuviera completo, se ejecutaría la simulación.

Cada nodo creado tendría su propio terminal para poder interactuar con él, igualmente como si se lanzará los nodos usando los comandos básicos de Netkit.

[1] http://www.netkit.org/download/netedit/netedit-1.0.tgz



Netedit

Las ventajas que un usuario encontraría usando Netedit, en vez de usar los comandos básicos que posee Netkit, son las siguientes:

- Disponer de un interfaz de Usuario para poder realizar los diseños.
- El usuario no se debería de preocupar de como usar los comandos de Netkit para realizar topologías de red.
- Permite guardar las configuraciones de red para posteriormente recuperarlas en posteriores sesiones.

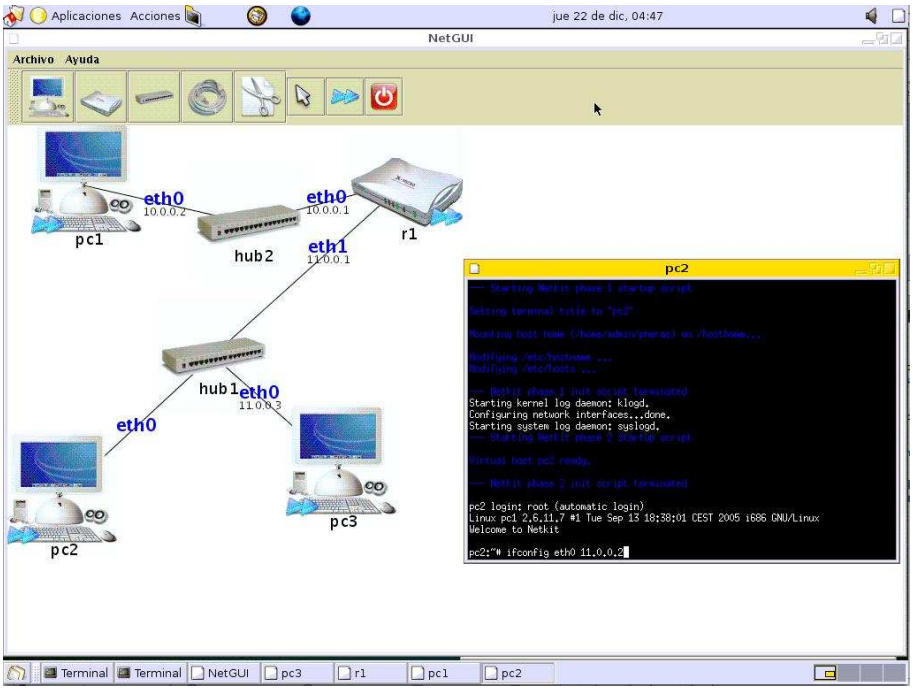
Aunque en esta aplicación hemos encontrados diversos inconvenientes:

- El usuario para poder usar la aplicación debería instalar y configurar Netkit en la máquina en la que ejecuta Netedit.
- La aplicación al estar realizada en Java, necesita la instalación de la máquina virtual de Java.
- Únicamente se podría ejecutar el interfaz en máquinas GNU/Linux.

2.2.2. NetGUI

Desarrollado por la Universidad Rey Juan Carlos, Netgui[2] ofrece un interfaz de Usuario a Netkit, igual que la aplicación comentada anteriormente.

[2]http://mobiquo.dat.escet.urjc.es/netgui/



NetGUI

NetGUI, ofrece un interfaz de Usuario bastante completo. Esta herramienta es bastante más sofisticada y potente que Netedit. Aunque su finalidad sea la misma, optaremos por NetGUI como referente a la hora de realizar este proyecto.

NetGUI, ha sido usada como método docente en asignaturas de la Titulación de Ingeniería de Telecomunicaciones de la Universidad Rey Juan Carlos, en concreto la asignatura de Arquitectura de Redes de Ordenadores. Esta herramienta ha permitido a los alumnos realizar diseños de redes y asentar los conocimientos teóricos.

El principal inconveniente que vemos de la utilización de esta aplicación, una vez más es la instalación de Netkit en la máquina donde se ejecuta NetGUI. También otro inconveniente, es que únicamente esta aplicación puede ser usada en entorno GNU/Linux y siempre que la máquina donde se ejecute disponga de una máquina virtual de Java.

2.3. Interfaz Web para Netkit

Como hemos visto anteriormente, ya existen herramientas que dotan a Netkit de un interfaz de Usuario bastante completo, en el cual el usuario se puede encontrar muy cómodo a la hora de diseñar un red de nodos virtuales.

El objetivo de este proyecto es dar un paso mas de lo que han dado las aplicaciones anteriores. Como bien hemos comentado anteriormente, las dos aplicaciones vistas usan como lenguaje de programación Java. Esto es una ventaja, ya que Java al ser multiplataforma podríamos ejecutar las aplicaciones en cualquier sistema operativo. Pero en realidad este escenario no es correcto, ya que estas aplicaciones deben estar ejecutándose en la misma máquina donde se tenga instalado Netkit.

Esto puede ser un problema, ya que Netkit esta pensado para funcionar sobre GNU/Linux, y por desgracia el numero de usuarios que disponen de este sistema operativos no es muy grande.

Un usuario que quisiera usar Netkit con cualquiera de las aplicaciones vistas anteriormente, debería poder instalarse un sistemas operativo GNU/Linux y posteriormente proceder a la instalación y configuración de Netkit. El proceso al ser complicado, descarta a usuarios potenciales de Netkit.

Para solventar este problema, se pensó en un modelo de aplicación el cual permitiera poder tener un Interfaz de Usuario para Netkit, dicho interfaz no conllevará la instalación complicada de Netkit y que se pudiera ejecutar en cualquier sistema operativo sin coste adicional para el usuario.

Para ello existe un modelo de desarrollo y comunicación, **El Web**. Usando el nuevo concepto de Web 2.0, se va a pretender diseñar un interfaz Web para Netkit, el cual tenga las características de las aplicaciones de escritorio, pero ejecutándose en un navegador web.

La evolución del Web y de nuevas tecnologías aplicadas al Web, puede permitir crear aplicaciones con aspecto e interacción muy similar a las encontradas en las aplicaciones de escritorio.

Como base usabilidad, se ha usado NetGUI, para dotar a este proyecto de una funcionalidad similar a la encontrada en la herramienta descrita anteriormente. Con la ventaja que el usuario necesitará solo un navegador para ejecutar la aplicación.

Los objetivos que se pretenden conseguir con este interfaz son:

- El usuario podrá añadir componentes (computadores, hubs o routers) a la red que pretende diseñar.
- El usuario podrá crear enlaces para crear las distintas redes que quiere simular.
- El usuario podrá interactuar con el navegador, de forma parecida como si estuviera interactuando con una aplicación de escritorio convencional.
- La máquina donde se ejecuta el navegador no debe de llevar ningún software adicional que dificulte su configuración.
- La aplicación Web, debe poder comunicarse con un componente software, que estará en otra máquina, que contenga Netkit.

Como se ha descrito anteriormente, la configuración e instalación de Netkit no irá en la máquina donde el usuario final ejecuta el Interfaz.

Con esto se pretende, que Netkit llegue a un número mayor de usuarios, y no únicamente a usuarios que solo usan sistemas operativos GNU/Linux.

2.4. Multiplataforma

Como bien se ha comentado en el apartado anterior, el principal objetivo por el cual se ha realizado este proyecto, es que Netkit llegue a los máximos usuarios posibles. Para ello se ha utilizado el Web como plataforma de desarrollo.

El protocolo HTTP, como HTML, al estar estandarizados, permiten que múltiples Sistemas, arquitecturas y plataformas puedan usar esta arquitectura.

Uno de los objetivo primordiales de este proyecto, es que el interfaz realizado pueda ser usado en múltiples sistemas operativos (Windows, GNU/Linux, MacOS X .....) y por múltiples navegador (Firefox, Opera, Internet Explorer, Safari).

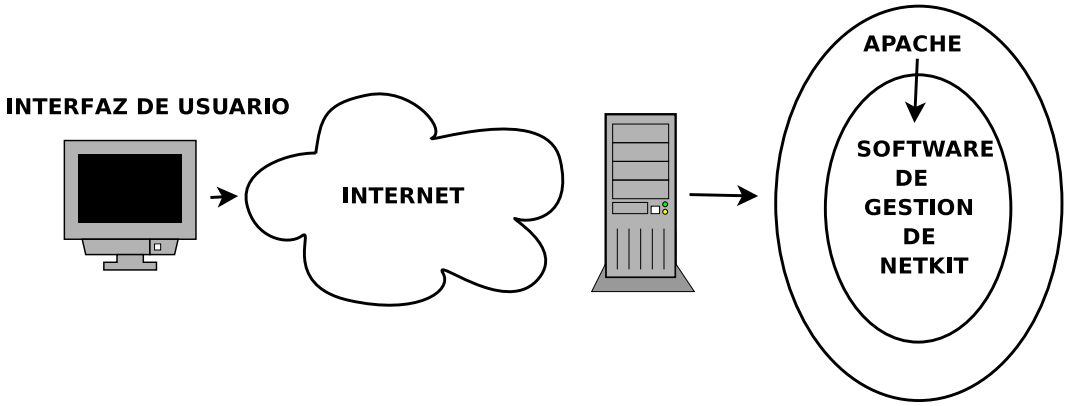
Separar el interfaz de Usuario del núcleo y gestión de Netkit, ha permitido que se pueda dar el escenario que hemos comentado. El usuario únicamente se deberá de preocupar de disponer de un navegador web que cumpla los estándares de HTML. Las demás consideraciones quedarán relegadas al software de gestión de Netkit, que se ejecutará en una máquina remota y estará administrado por personal especializado.

2.5. Ejecución de Netkit en Remoto

Para poder ejecutar remotamente, se necesita que el software de gestión de los nodos este ejecutándose en una máquina GNU/Linux. Se ha elegido el sistema GNU/Linux, ya que Netkit únicamente este disponible en este sistema.

El software que lleve la gestión de los nodos de Netkit, creados por medio del interfaz de usuario disponible desde el navegador, debe permitir llevar un control de dichos nodos.

Para ello el interfaz de usuario debe poder comunicarse con el software que gestiona los nodos por medio del protocolo HTTP. Para ello se necesita un servidor Web en la máquina que lleva dicha gestión, la cual será la misma que tenga instalado Netkit. El servidor Web que se utilizará para esta tarea será Apache, ya que es el servidor predominante en la mayoría de las máquinas que operan en internet. Otra razón por la cual se ha elegido Apache para realizar de intermediario con el software de Gestión de los nodos de Netkit, es que Apache permite una gran facilidad de ejecutar software interactuando con peticiones HTTP.



Ejecución Remota

El lenguaje elegido para realizar esta tarea, será Python. Python es un lenguaje fácil de usar y bastante potente para realizar este tipo de tareas, a su vez puede ser llamado directamente desde el propio servidor Apache.

Téngase en cuenta que únicamente Apache, será el encargado de hacer de intermediario entre el interfaz de Usuario y el propio software que lleva la gestión de los nodos virtuales, realizado en Python.

El software que llevará el control de los nodos Netkit debe permitir las siguientes acciones.

- Ejecución de todos los nodos que el usuario haya diseñado en el interfaz.
- Debe permitir que los nodos estén conectado con la misma topología de red que haya diseñado el usuario.

- El usuario podrá interactuar por medio del interfaz con los nodos ejecutados. De la misma forma como si estuviera ejecutando Netkit en su propia máquina.

- Debe permitir detener la simulación de la red que se ha diseñado, para posteriormente añadirle más elementos o directamente crear otra nueva topología.

El diseño de esta arquitectura puede permitir separar claramente del software de usuario, con el software de administrador y control de Netkit. Para ello el usuario únicamente se debe de preocupar del diseño y utilización de los servicios que aporta esta arquitectura, dejando la administración del servicio totalmente fuera del usuario final.

2.6. Desarrollo usando Ajax

Como bien se ha comentado anteriormente, se desea realizar un interfaz de Usuario que se asemeje a las aplicaciones de escritorio. El usuario podrá interactuar con el interfaz por medio de ratón, añadir elementos, mover elementos, crear enlaces etcétera, como se podría hacer con las aplicaciones descritas anteriormente.

Este tipo de aplicación pensada hace unos años, hubiera desencadenado en una aplicación Web en el cual el usuario tuviera que navegar por un gran numero de formularios y de páginas HTML. Actualmente la creación de aplicación Web es posible realizar con una única página HTML, sin tener que ir navegando por infinidad de páginas HTML.

Empresas como google [3], están lanzando al mercado en los últimos tiempos infinidad de servicios que ofrecen al usuario un interfaz muy similar a la encontrada en aplicaciones de escritorio. Algunos de los ejemplos pueden ser:

- **YouOS**. *http://youos.com/* . Aplicación Web que simula el comportamiento de un escritorio de un sistema operativo en nuestro navegador.

- **Google Maps**. *http://maps.google.es/*. Aplicación Web creada por Google que permite visualizar por medio de satélites la geografía mundial.

- **Google Spreadsheets**. *http://spreadsheets.google.com*. Hoja de Calculo creada por Google, que permite tener en nuestro navegador una potente aplicación con un interfaz muy similar al que se encontraría un usuario si usara Microsoft Office Excel.

Todas estas aplicaciones permiten al usuario poder disponer de ellas sin necesidad de instalar ningún software adicional, únicamente usando su propio navegador Web. Para ello se ha utilizado un conjunto de herramientas y de técnicas, denominado **Ajax**. Ajax permite realizar este tipo de Sistemas, separando claramente lo que es la lógica de la aplicación con el interfaz de usuario.

Este método ha sido el empleado para dotar a Netkit de un interfaz web que pueda permitir al usuario realizar las mismas tareas que podría hacer con las aplicaciones existentes para Netkit.

[3] http://google.com

2.7. Configuración y Comunicación mediante XML

Todo el sistema planteado anteriormente, tanto el interfaz de usuario como el software de gestión de nodos, deben de llevar una control de los datos procesados y de la configuración que se quiere dar a la plataforma, para ello se utilizará XML.

La mayoría de los lenguajes de programación permite tratar con información almacenada en XML, gracias a que XML al ser un estándar aprobado por World Wide Web Consortium (W3C) [4].

2.8. Desarrollo siguiendo un control de Versiones

Para la realización de este proyecto se va a seguir un control de versiones del software y documentación desarrollada. El control de versiones permite al programador llevar un histórico de las versiones del software que está desarrollando.

Imaginemos que hacemos un gran cambio del software en una jornada de desarrollo y al día siguiente nos encontramos que el trabajo realizado no cumple el modelo arquitectónico ni de diseño que hemos realizado anteriormente. Al programador le gustaría poder volver a una versiones del software que tuviera estable y que cumpliera los requisitos. Pues bien este problema podría estar solventado si el desarrollador mantiene un control de versiones del software que desarrolla.

Por ello creemos que es un objetivo que el software desarrollado este gestionado por un sistema de control de versiones. El sistema elegido ha sido **Subversion** [5], por las siguientes características:

- Subversion ha sido desarrollado con una licencia de Software Libre.
- Subversión es fácil de instalar, mantener y usar.
- Acceso de los repositorios y las versiones del software desarrollado por medio de la red (HTTP,SSH).
- Verdadero historial de versiones, al contrario de el que usaba el antiguo CVS.

2.9. Licencia Software de Software Libre

Un objetivo principal para el desarrollo de este proyecto es dotarle de una licencia de Software Libre, es nuestro caso la Licencia GPL descrita en el capítulo de introducción.

El software de este proyecto al ser liberado mediante una licencia GPL, contribuye a que algún usuario futuro pueda realizarle modificaciones o añadir funcionalidad adicional. Este software modificado puede ser utilizado como medio docente para asignaturas que traten sobre Redes de Ordenadores en titulaciones como Ingenierías Informáticas o Telecomunicaciones.

Una vez terminado este proyecto, el código quedará a disposición de cualquier usuario o persona que quiera usarlo, sin tener que pagar ninguna licencia y siempre suministrando todo el código fuente del software desarrollado.

[4] http://www.w3.org/
[5] http://subversion.tigris.com

2.10. Adquisición de experiencia en programación

Para la realización de este proyecto se necesitará una base en programación. Un proyecto de estas dimensiones requiere un volumen considerable de programación.

Un objetivo para el desarrollo de este proyecto es conseguir aumentar la base de programación y aprender a desarrollar aplicaciones Web con la nueva tecnología, **Ajax**.

2.11. Experiencia en la realización de un proyecto

Es un objetivo aprender como realizar un proyecto. Un proyecto de estas dimensiones requiere mucho esfuerzo. No sólo se desea implementar y resolver un problema que se ha enunciado, sino tener que ajustarse a unas normas, saber qué es lo que se quiere y como se debe hacer, siempre atendiendo a la linea que marca el tutor. Un proyecto de estas dimensiones supone tener que documentarse, analizar el problema planteado, buscar soluciones al problema, desarrollar la solución y por último, pero no menos importante, documentar todo el trabajo realizado.

Capítulo 3

Metodología empleada

3.1. Introducción

El desarrollo y mantenimiento del software es una tarea complicada, la cual conlleva mucho esfuerzo y dedicación . Este proceso es mantenido por la ingeniería del Software.

Un producto software va evolucionando según un proceso de desarrollo(requisitos, análisis, diseño, implementación y pruebas). Este proceso de desarrollo, ayuda a la realización del proyecto dividiéndolo en subfases, cada una con un objetivo claro:

- **Requisitos**. Especifica los requerimientos que va a tener nuestro producto software una vez haya sido terminado satisfactoriamente.
- **Análisis**. Analizar los distintos requisitos expuestos en la fase anterior. Se analizará la viabilidad, alcance, carga de trabajo, estimación de tiempos de todos los requisitos.
- **Diseño**. Consiste en determinar el funcionamiento general de nuestro producto software.
- **Implementación**. Implementación y codificación en un lenguaje de programación el diseño elegido para la realización del Software.
- **Pruebas**. Una vez terminado, se comprueba que el software cumple los requerimientos por el cual ha sido creado.
- **Mantenimiento**. Mantener y mejorar el software para enfrentar errores descubiertos y nuevos requisitos

Dentro del ámbito de la ingeniería del software existen multitud de paradigmas de desarrollo del software, para este proyecto se empleará un desarrollo en espiral y un desarrollo basado en prototipos.

3.2. Desarrollo en espiral

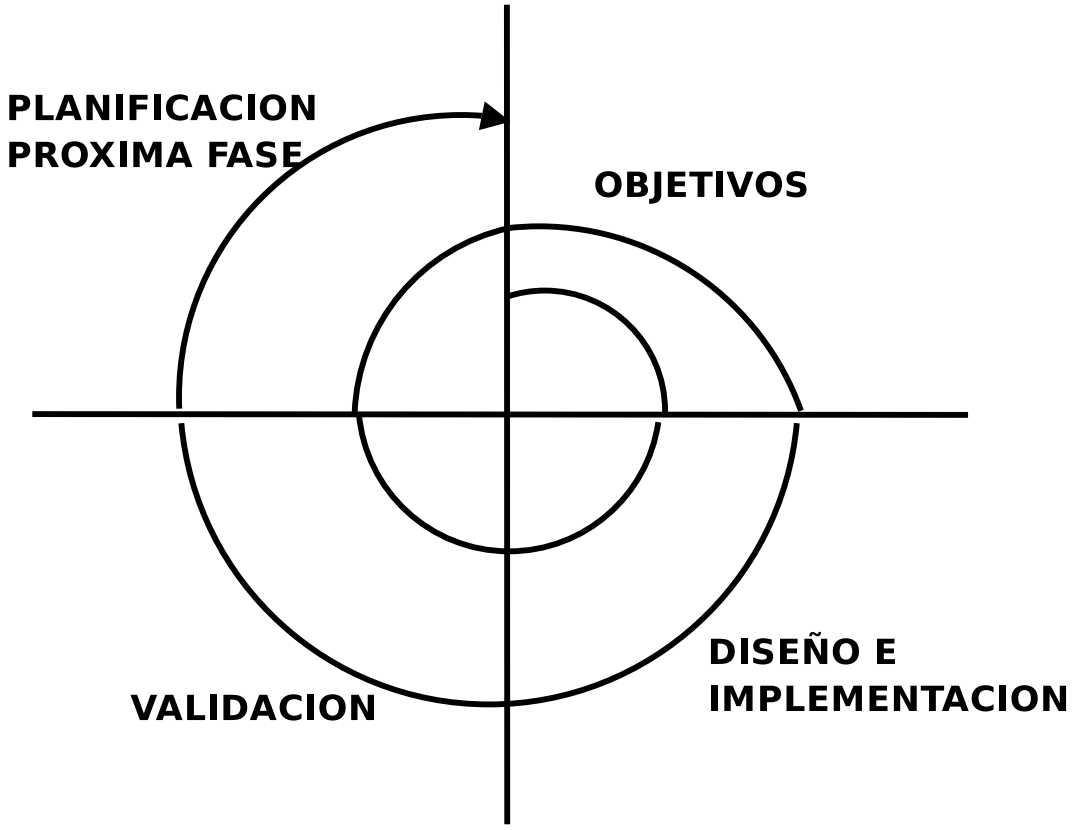
Desarrollado por B. Boehm, es un modelo de ciclo de vida, utilizado generalmente en la Ingeniería del software. Las actividades de este modelo son una espiral, cada bucle es una actividad. Las actividades no están fijadas a priori, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior.

Sus características son las siguientes:

- Usa un Modelo de Cascada para cada etapa.
- Está orientado a evitar riesgos de trabajo.
- No define en detalle el sistema completo a la primera.
- Los desarrollares deberían solamente definir las más altas prioridades.

3.2.1. Tareas

Para cada actividad o ciclo, habrá cuatro tareas como se define en la siguiente figura 4.1:



Modelo en Espiral

Fijar objetivos

- Fijar también los productos definidos a obtener: requerimientos, especificación, manual de usuario.
- Fijar las restricciones.
- Identificación de riesgos del proyecto y estrategias alternativas para evitarlos.
- Hay una tarea que sólo se hace una única vez: planificación inicial o previa.

Se estudian todos los riesgos potenciales y se seleccionan una o varias alternativas propuestas para reducir o eliminar los riesgos.

Diseño e Implementación

- Diseño de la arquitectura propuesta para la realización de los objetivos
- Desarrollo de los objetivos planteados como hitos para las actividades.

Validación y Pruebas

- Realización de las pruebas necesarias para comprobar que el diseño y la implementación cumplen los objetivos y requisitos expuestos anteriormente.
- En el caso que las pruebas no sean satisfactorias no se podrá proseguir en este modelo y se debería volver a fijar los objetos, para posteriormente volver a diseñarlos e implementarlos.

Planificar próximas fases

Una vez realizadas todas las pruebas satisfactoriamente, se debería planificar las fases posteriores del modelo del software.

3.2.2. Ventajas

El análisis del riesgo se hace de forma explícita y clara. Une los mejores elementos de los restantes modelos.

3.2.3. Inconvenientes

Genera mucho trabajo adicional. Exige una cierta habilidad en los analistas (es bastante difícil).

3.2.4. Aplicación de este modelo al desarrollo de este proyecto

El análisis de requisitos para este proyecto ha sido en esencia la comunicación entre mi tutor del proyecto y yo. Mi tutor ha sido el encargado de facilitarme los requisitos que deberían existir una vez finalizado el proyecto.

Sobre el análisis de riesgos existente en este proyecto, ha sido también el tutor del proyecto quien ha ido indicando los principales riesgos críticos que se podían dar en la realización del Software.

La fase de diseño e implementación se han propuestos distintas alternativas sobre la arquitectura del software y su implementación. Sobre esta fase se hablará en el capítulo siguiente.

Una vez que se ha llevado a cabo la implementación se realizarán una serie de pruebas al software para corroborar que se cumplen los requisitos.

Por último, y como marca el modelo en espiral, se han llevado a cabo un plan para la elaboración de las siguientes fases. Respecto a este proyecto, la elaboración de las fases posteriores se ha llevado a cabo basándose en las pruebas realizadas en las fases anteriores. Si las pruebas han sido correctas, en las posteriores fases se realizarán un incremento en la funcionalidad, en el caso que sean erróneas se incrementará la robustez del software, tal como dicta el desarrollo en espiral.

3.3. Desarrollo basado en prototipos

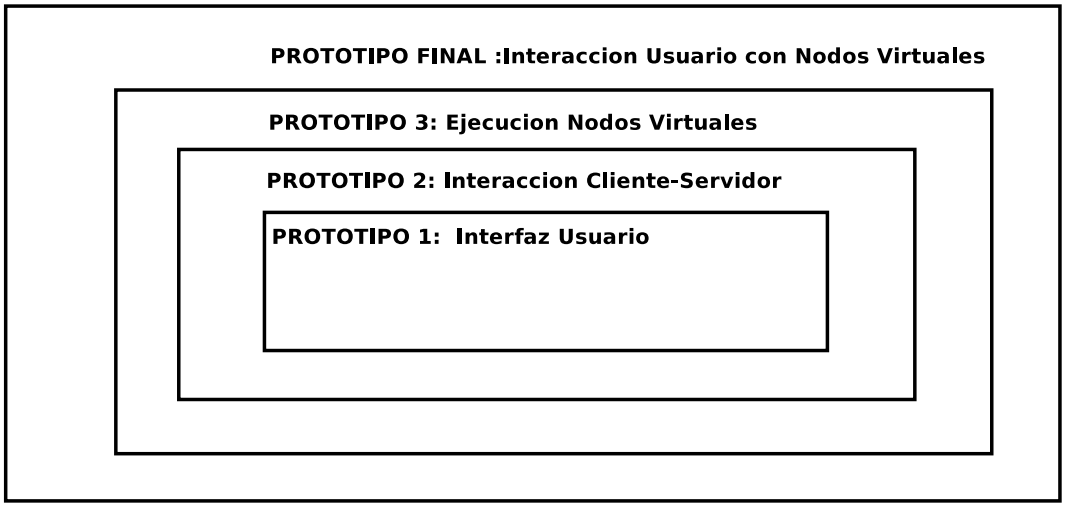
Un prototipo es una versión del producto software la cual satisface algunos de los requisitos expuestos para el producto. Cada prototipo va a tener su propia autonomía y va a ser desarrollado siguiendo el modelo en Espiral. Cada prototipo del producto Software va a ser una evolución, en términos de funcionalidad y requisitos, del prototipo desarrollado anteriormente. El resultado final del producto software corresponderá al último prototipo desarrollado de la evolución de los prototipos anteriores.

Para la realización de un prototipo se va a determinar en las siguientes fases de desarrollo:

- **Requisitos**. Identificación de los requisitos propuestos para dicho prototipo.
- **Análisis**. Análisis de los requisitos propuestos en la fase anterior.
- **Diseño e Implementación**. Diseño del modelo arquitectónico del prototipo e implementación del prototipo basándose en el diseño realizado.
- **Pruebas**. Fase de Pruebas para comprobar que el prototipo diseñado e implementado satisface los requisitos por el cual fue creado.

3.3.1. Aplicación de este modelo al desarrollo de este proyecto

Para la realización de este proyecto se han construido 3 prototipos previos a la versión final del producto Software.



Prototipos de NetWeb

A continuación se describirán los 4 prototipos.

- **Prototipo 1**. Este prototipo se centrará en la implementación del interfaz de usuario.
- **Prototipo 2**. Este prototipo será el encargado de la comunicación entre el interfaz de usuario y la máquina donde se aloja el servidor.
- **Prototipo 3**. Ampliando el prototipo anterior, nos encontramos que este prototipo implementará la función de ejecutar la topología de red diseñada por el usuario.

- **Prototipo Final**. Este prototipo corresponderá al software final del proyecto. En el se centrará la implementación de la ejecución de los comandos GNU/Linux en las máquinas simuladas mediante Netkit. Téngase en cuenta que esta interacción se hará por medio del interfaz de usuario, el navegador.

Capítulo 4

Diseño e Implementación

4.1. Introducción

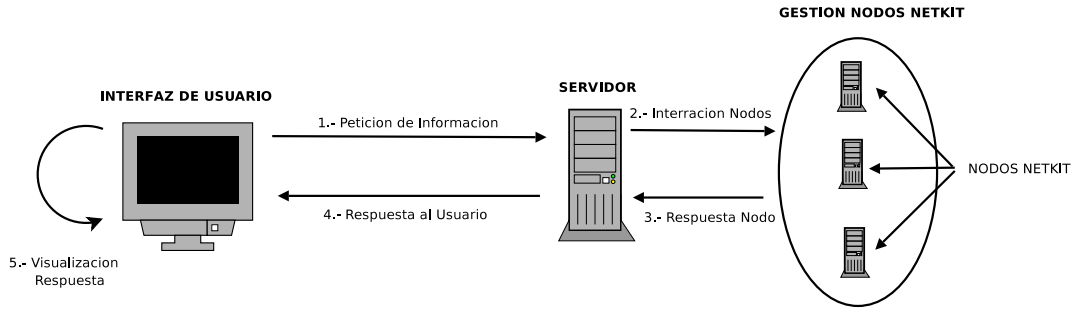
El contenido de este capítulo es fundamental para comprender el software desarrollado en este proyecto, siendo la parte más importante, informaticamente hablando.

Primero se detallará la arquitectura que se seguirá para la implementación del software. Posteriormente se detallarán las especificaciones, diseño e implementación de cada prototipo, en nuestro caso 4. Téngase en cuenta que cada prototipo debe cumplir los requisitos expuestos, y que cada prototipo será una evolución del anterior.

Una vez explicados los prototipos intermedios, el capítulo terminará exponiendo el prototipo final que cumplirá los requisitos iniciales expuestos para la realización de este proyecto.

4.2. Arquitectura

Para el desarrollo del proyecto usaremos una arquitectura de cliente-servidor. A continuación detallaremos los distintos módulos y partes que comprenden la arquitectura del sistema. A grandes rasgos, el sistema está compuesto por tres grandes módulos software, el interfaz de usuario, el servidor y el software para la gestión de los nodos virtuales de Netkit.



Arquitectura General

4.2.1. Interfaz de Usuario

El interfaz será la parte de la arquitectura encargada de interactuar con el usuario. Por medio de ella el usuario podrá manipular y utilizar las distintas máquinas virtuales ejecutadas en el servidor.

Téngase en cuenta que el interfaz de Usuario en esta arquitectura corresponderá a un navegador Web, ya que uno de los requisitos primordiales de este proyecto es dotar a Netkit de un interfaz que se pueda ejecutar en cualquier navegador Web.

El usuario deberá poder usar el interfaz Web de Netkit, como si realmente estuviera ejecutando una aplicación de escritorio tradicional. Pero una de las ventajas que tiene poder tener un interfaz Web es que cualquier usuario puede acceder a él sin la necesidad de instalar un software externo. El HTML es un lenguaje estándar que cualquier navegador ejecutando en cualquier sistema operativo puede utilizar. Para la realización de este proyecto se seguirá estrictamente el estándar de HTML, para que cualquier navegador que cumpla dicho estándar pueda usar el interfaz sin ningún problema adicional.

El navegador usado para la realización de este proyecto ha sido Firefox.

4.2.2. Servidor Web

El servidor Web hará de intermediario entre el cliente y el software que gestionará el control de los nodos Netkit que quiera ejecutar el usuario.

El servidor será el encargado de recibir las peticiones HTTP del cliente e interactuar con el software que gestiona los nodos Netkit que el usuario ha creado por medio del interfaz web.

Para ello, se ha tomado Apache como servidor Web. El Servidor Web podrá recibir distintos tipo de peticiones por parte del Interfaz Web, este tipo de peticiones siempre serán recibidas por medio del protocolo HTTP.

- **Ejecución de los nodos**. El Servidor Web recibirá la información de ejecución de los nodos que ha diseñado en el interfaz web.

- **Interacción con los nodos**. El usuario podrá ejecutar comandos GNU/Linux en cualquiera de los nodos que ha diseñado.

- **Parada de los nodos**. El usuario podrá detener los nodos que anteriormente ha ejecutado.

4.2.3. Nodos Netkit

Netkit al disponer de un entorno completo de simulación de redes de computadores GNU/Linux, dispone de mecanismos para poder comunicarse con los distintos nodos. El modo de interacción se hace por medio de un terminal con un interprete de comando (bash). Cada nodo virtual tiene el suyo propio.

Este mecanismo de comunicación no es útil para la interacción entre un nodo y un programa que haga de intermediario entre el usuario y el nodo. Para ello Netkit dispone de mas mecanismos para poder interactuar con los distintos nodos. Uno de ellos es que cada nodo atienda peticiones telnet en un puerto TCP, el nodo puede recibir por dicha conexión telnet comando de shell como si los recibiera directamente desde el terminal de bash que lanza por defecto.

Para la ejecución de un nodo virtual, es necesario la herramienta **vstart**, suministrada mediante Netkit y se ejecuta de la siguiente forma:

```
vstart computer1 --port=7658
```

Vstart ejecutaría una máquina virtual de nombre computer1 y que recibiría peticiones de comando de bash por medio de una conexión telnet que escucha en el puerto TCP 7658.

Uno de los principales inconvenientes que tiene que un nodo sea lanzado en un puerto TCP, es únicamente se puede hacer a él una única conexión telnet para interactuar. En el momento que dicha conexión sea cerrada, ya no habrá forma de comunicarse con el nodo. En este caso habría que usar el comando *vcrash* de Netkit.

```
vcrach computer1
```

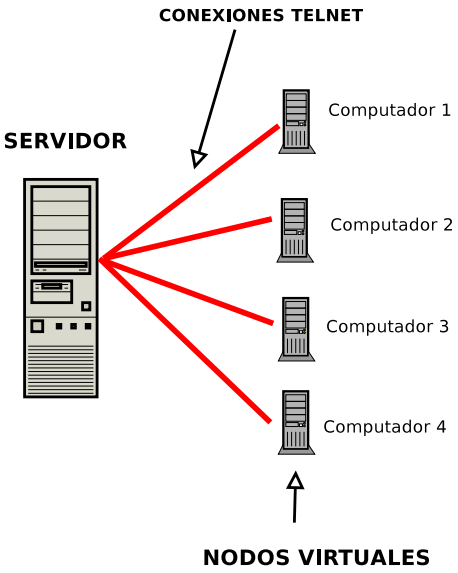
4.2.4. Gestión de Nodos Netkit

El servidor será el encargado de alojar la ejecución de los nodos virtuales y de hacer de intermediario entre el usuario y dichos nodos.

Como cada nodo únicamente puede recibir una única conexión TCP, el servidor debe poder almacenar las conexiones abiertas con cada uno de los nodos ejecutados. Por cada conexión que se ha realizado con cada uno de los nodos, el servidor puede mandar comandos y recibir la respuesta de la ejecución de dichos comando en el nodo.

Téngase en cuenta que por cada nodo, se almacena una conexión distinta y que dos nodos no pueden estar escuchando en el mismo puerto TCP.

El lenguaje elegido para la realización de este software de Gestión de los nodos virtuales ha sido Python. Python es un lenguaje muy sencillo de utilizar y se puede ejecutar directamente desde el Servidor Web Apache.



Comunicación Servidor Nodos Netkit

4.3. Prototipo 1: Construcción del Interfaz de Usuario

4.3.1. Especificación

El objetivo de este prototipo es dotar a Netkit un interfaz Web de usuario. Dicho interfaz deberá de cumplir las siguientes condiciones:

- **Multiplataforma**. El principal objetivo por el cual se ha decidido optar por un interfaz Web, es la multiplataforma, ya que se desea que este interfaz funcione en la mayoría de sistemas operativos. Un requisito principal para la realización de este prototipo es que el usuario no deba instalar ningún software para poder interactuar con el entorno de creación de redes de nodos virtuales. Para ello, se utilizarán los estándares apropiados para que cualquier navegador pueda interpretar tanto el HTML como las hojas de estilo CSS.

- **Creación de un Nodo Virtual**. Por medio del interfaz Web, el usuario podrá añadir nodos al diseño de red que desea realizar.

- **Tipos de Nodos**. El interfaz deberá permitir la creación de al menos tres tipos de elementos en la red.

  - **Computador**. Dicho nodo corresponderá a un computador convencional. Este tipo de computador únicamente deberá poseer un interfaz de red.
  - **Hub**. Este nodo permitirá la conexión entre distintos computadores y routers para poder formar una red local.
  - **Router**. Es el elemento necesario para poder conectar distintas redes locales. Téngase en cuenta que un router puede disponer de mas de un interfaz de red.

- **Enlace entre Nodos**. El interfaz deberá permitir la conexión entre distintos nodo del entorno de red.

- **Arranque de los nodos**. El usuario podrá arrancar los distintos nodos virtuales para posteriormente interactuar con ellos. Los nodos se arrancarán con la configuración de red que haya diseñado el usuario por medio del interfaz. Una vez que estén arrancados los nodos no se podrá modificar el entorno de red diseñado por el usuario.

- **Detención de los nodos**. El interfaz proporcionará un mecanismo por el cual se detendrán la ejecución de los nodos que anteriormente han sido ejecutados. Una vez que se haya detenido la ejecución de los nodos, se podrá añadir a la red diseñada mas elementos.

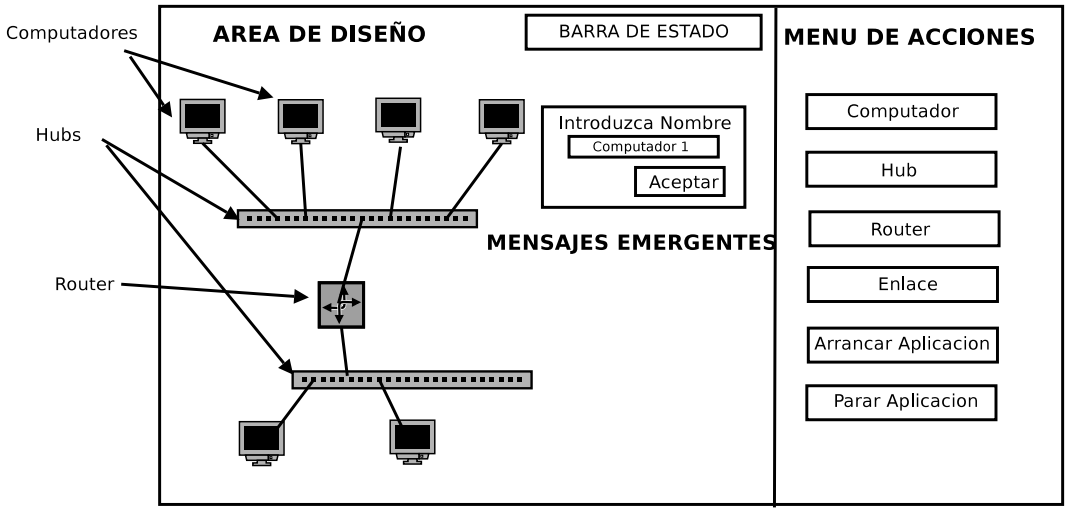
- **Interacción con el Usuario**. La interacción con el usuario será realiza por medio del ratón. El usuario debe encontrar un interfaz amigable y sencilla, igualmente que si se encontrará con una aplicación de escritorio normal. El usuario podrá seleccionar las opciones a realizar por medio de botones.

- **Navegación entre páginas**. No se permitirá la navegación entre distintas páginas, únicamente existirá una única página HTML que corresponderá a todo el interfaz.

4.3.2. Diseño

Interfaz de Usuario

Para poder cumplir la especificación definida anteriormente, se pretende diseñar un interfaz Web que sea intuitivo para el usuario. El interfaz de usuario contendrá tres sección muy bien definidas. Dicho interfaz tiene que ser sencillo y que cumpla los entandares de HTML para que pueda ser usado en la gran mayoría de navegadores Web que existen en el mercado.



Diseño Prototipo 1

El Interfaz de Usuario se compondrá de las siguientes partes.

- **Área de Diseño**. En la cual el usuario podrá diseñar su topología de red.

- **Barra de Estado**. Esta barra mostrará e indicará al usuario el estado actual que tiene la aplicación. La barra deberá mostrar al usuario los siguientes mensajes.

  - *Creación de enlaces*. Cuando la barra de estado muestre este mensaje se podrá conectar los distintos nodos de los que componen la red. Estos enlaces corresponderán a los enlaces físicos realizados por medio de un cable de red.
  - *Ejecución de la aplicación*. Cuando este mensaje esté mostrado en la barra de estado, significará que los nodos están ejecutándose y que ningún nodo se puede añadir a la red.

- **Menú de Acciones**. Es la parte del interfaz Web donde el usuario puede realizar las distintas acciones que se detallan a continuación. Esta interacción se va a realizar por medio de botones.
  - *Añadir Computador*. El usuario podrá añadir un computador al área de diseño.
  - *Añadir Hub*. El usuario podrá añadir un hub al área del diseño.
  - *Añadir Router*. El usuario podrá añadir un router al área de diseño.
  - *Crear enlaces*. El usuario podrá crear los distintos enlaces entre los elementos que compondrán la red que pretende diseñar.
  - *Arranque Nodos*. Por medio de un botón, el usuario podrá arrancar todo el entorno de red que ha diseñado y dibujado en el área de Diseño
  - *Parada de los Nodos*. El usuario podrá parar la ejecución de los nodos que anteriormente habrá arrancado.

- **Mensajes Emergentes**. La aplicación podrá interactuar con el usuario por medio de mensajes emergentes, por ejemplo para asignarle un nombre a un componente que se añada al área de diseño. Los mensajes emergentes también se usan para notificar al usuario que ha realizado una acción incorrecta.

4.3.3. Implementación

Para la implementación de este prototipo deberíamos escoger alguna librería que permita poder tener en el navegador un interfaz el cual se asemejara a la interfaz que pudiera tener cualquier usuario en una aplicación de escritorio, con la única ventaja que para poder usarla no se necesitará instalar ningún software adicional.

Este problema se solvento usando la librería de **Drag and Drop de Walterzon** [1]. Esta librería permite tener objetos en el HTML con el que el usuario puede interactuar, es decir puede desplazar y deslizar objetos HTML dentro de la propia página HTML.

El lenguaje utilizado para realizar el interfaz Web ha sido Javascript, ya que javascript es un lenguaje que ejecuta en los navegadores Web actuales. Javascript permite a los navegadores añadir funcionalidad que no se podría expresar únicamente con HTML.

Distribución del Interfaz

El primer paso del desarrollo de este prototipo nos lleva a distribuir el interfaz para que el usuario se sienta cómodo. Este interfaz debe ser intuitivo y tener las partes bien diferenciadas, como se explicó anteriormente en la especificación de este prototipo.

[1] http://www.walterzorn.com/dragdrop/dragdrop_e.htm

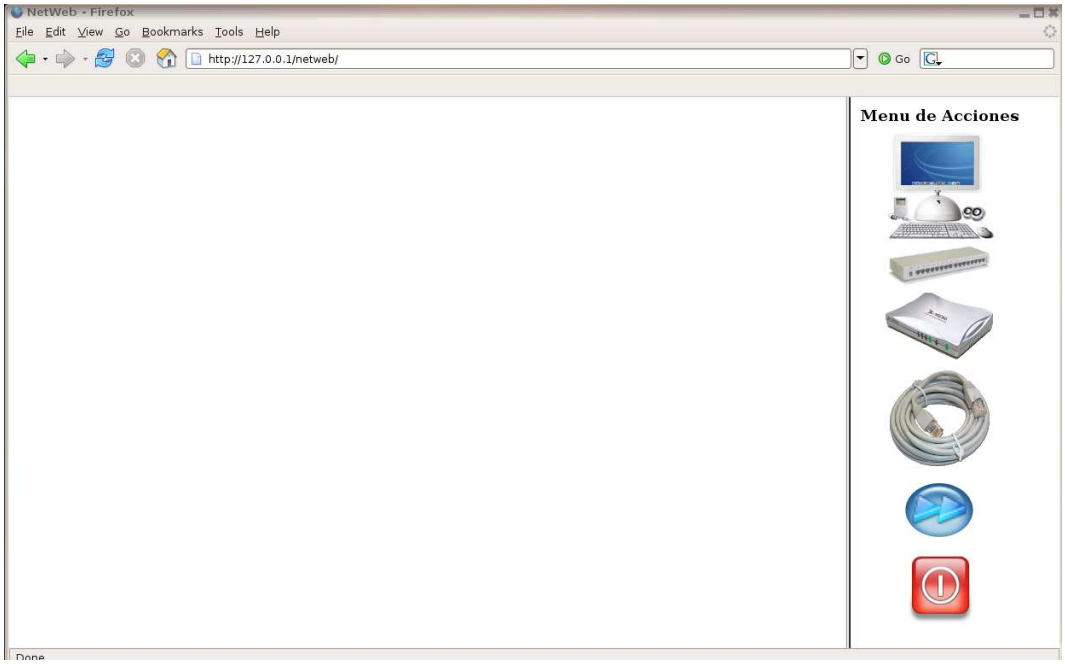
En este momento nos encontramos con dos grandes áreas en la que separaremos el interfaz, el área de acciones y el área de diseño. Para ello hemos diseñado dos HTML totalmente distintos que uniremos en una única pagina HTML por medio de un frame.

El área de Acciones, corresponderá a las distintas acciones que puede hacer el usuario en el sistema. Estas acciones serán activadas mediantes botones que ejecutarán código HTML, que explicaremos más adelante. Para ello se han elegido botones que permitan al usuario saber que acciones van a realizar con sólo observarlos. El diseño de este área esta definido en la pagina *left.htm*.

Para el área de diseño se ha elegido una página totalmente en blanco en la cual se podrán añadir los distintos elementos que compondrá la topología de red que el usuario pretende diseñar. Este área estará definida en el fichero *right.htm*.

```
<script type="text/javascript" src="scripts/netweb.js"></script>

<frameset name="index" id="index" cols="80%,20%">
 <frame name="left" src="left.htm">
 <frame name="right" src="right.htm">
</frameset>
```



Interfaz NetWeb

Como se puede observarse en la imagen, cada sección del interfaz del usuario esta dividida perfectamente por medio de un frame. A la derecha se encontraría los botones necesarios para interactuar con el sistema y la izquierda toda el área donde el usuario diseñará su Red Virtual.

El menú de Acciones comprenderá distintos botones.

	Cuando este botón es pulsado, se añade un computador al área de Diseño.
	Cuando este botón es pulsado, se añade un hub al área de diseño. El hub será usado para interconectar computadores en la misma red local.
	Cuando este botón es pulsado, se añade un router al área de diseño.El router será empleado para interconectar distintas redes locales.
	Cuando este botón es pulsado, el usuario podrá realizar y crear enlaces entre distintos elementos del diseño.
	Cuando este botón es pulsado por el usuario, se ejecutará la aplicación con la red creada en el área de diseño.
	Este botón representa la opción de parada de la aplicación.

Cuadro 4.1: Botones

Téngase en cuenta que para la ejecución de las distintas acciones detalladas, el usuario deberá presionar en los distintos botones para realizar las acciones pertinentes.

Se ha elegido una división vertical de las dos partes del interfaz, porque creemos que el área de diseño se aprovecharía más el espacio para dibujar los elementos que dispondrá las red diseñadas.

Toda la funcionalidad añadida en la parte del cliente y la interacción con el servidor será desarrollada e implementada en javascript e incluida en el fichero *netweb.js*.

Objetos Dinámicos

Una vez diseñado la distribución del interfaz, el siguiente paso es poder representar la adición de elementos al área de diseño por medio de las acciones que realiza el usuario mediante el menú de acciones.

Para ello nos encontramos con un problema grave, ya que el estándar de HTML no contempla el uso de HTML dinámico. Para ello hemos usado una librería de dibujo de objetos dinámicos HTML llamada **Walterzon**.

Esta librería escrita en javascript y usando hojas de estilo CSS, permite poder realizar sitios Web interactivos.

Lo primero de todo, para poder usar esta librería en el HTML que estamos diseñando, es incluirla en toda página HTML de nuestro interfaz. Para ello el HTML que representa el interfaz deberá incluirla con esta sentencia.

```
<script type="text/javascript" src="scripts/wz_dragdrop.js"></script>
```

Área de Diseño

El área de diseño es la parte encarga del interfaz, donde el usuario añade los distintos elementos para crear la topología de red. En el área de diseño se van a encontrar los distintos objetos dinámicos que se van a poder mover y colocar el usuario para diseñar su topología. El área de diseño corresponde al fichero *left.htm*.

Los objetos dinámicos que existirán en el área de diseño serán las imágenes de los computadores, hubs y router. Estas imágenes serán similares, en aspecto, a las disponibles en el cuadro 5.1.

Lo primero que hacemos es añadir en el html las imágenes correspondientes a un computador, hub y router.

```
<img name="computer"  src="images/computer.png"  title="computer">
<img name="hub" src="images/hub.png" title="hub">
<img name="router" src="images/router.png" title="router">
```

Una vez añadidas al HTML del área de diseño procederemos a definirlas como un HTML dinámico, para ello vamos a usar la funcionalidad de *wz_dragdrop* que hemos descrito anteriormente. Para definir las imagines como HTML dinámico usamos la función SET_HTML, indicando el nombre de las imágenes que queramos que sean dinamicas de la siguiente manera.

```
<script type="text/javascript">
SET_DHTML(CURSOR_MOVE,  TRANSPARENT, "computer","hub", "router","status","comment");
</script>
```

A parte de crear las imágenes como HTML dinámico, también le hemos añadido algunas características como que cada vez que se muevan sean transparentes. El efecto de transparencia no aporta nada en términos de funcionalidad, pero queda bastante vistoso a ojos del usuario.

También se ha definido en el área de diseño la barra de estado de la aplicación, que también será un elemento HTML dinámico, pero con la diferencia que no se permitirá que se mueva por medio del ratón.

La barra de estado será definida en la parte superior derecha del área de diseño, y tendrá el nombre de *status* en el HTML. Esta barra de estado mostrará el estado que tendrá la aplicación en cada momento.

Normalmente cuando el usuario procede a utilizar la aplicación, le interesa que el área de diseño este totalmente limpia y vacía de elementos, para que él puede añadirlo y diseñar su propia topología. Para ello cada vez que se cargue el interfaz se ocultarán todos los elementos antes creados.

Para este cometido se usará la función *init()*, creada en javascript, y ejecutada solamente cuando se carga por primera vez la aplicación. Esta función será la encargada de ocultar todos los elementos existentes en el área de diseño.

```
function init(){

 for (var i = 0; i < dd.elements.length; i++)
  {
   dd.elements[i].hide();
  }
}
```

Como se puede observar existe una variable en javascript llamada **dd**, la cual posee todos los elementos dinámicos creados anteriormente. Por medio de *dd.elements.length*, se puede saber exactamente cuantos elementos dinámicos han sido creados.

Cada elemento dinámico ocupa una posición del array *dd.elements*, y posee distintos métodos que afectan a su estado. En concreto el método que estamos usando es *hide()*, este método oculta a la vista del usuario el elemento en cuestión.

Añadir Computador

Como bien se ha descrito es la especificación de este prototipo, se debe permitir añadir elementos a la topología de red que el usuario pretende diseñar.

Para ello cada vez que se presione al botón que representa un computador, botón que se encuentra a la parte derecha del interfaz, se debe añadir una instancia del computador en el área de diseño.

Para ello la imagen que representa el computador se define de la siguiente forma

```
<img  name="computer" src="images/computer.png" onClick="create_copy('computer')"
title="A&ntilde;adir Computador">
```

Cada vez que se presione en la imagen, se ejecuta la función de javascript llamada *create_copy*. La función *create_copy*, se encuentra en el fichero *netweb.js*, dicha función recibe como parámetro la cadena computer, que distigirá que elemento se quiere añadir al área de diseño.

Se añadirá un elemento siempre y cuando la aplicación no se este ejecutando, para ello antes de crear la instancia en el área de diseño se comprueba sino se esta ejecutando la aplicación, en caso contrario se mostrará un mensaje al usuario indicándoselo que no puede realizar la acción por medio de una ventana emergente. Esta comprobación se hace por medio de la variable RUN.

```
if (parent.RUN){
    alert("No se puede agregar elementos mientras esta en ejecucion la aplicacion");
    return;
  }
```

Una vez que se haya comprobado que se puede añadir un computador al diseño que se desea realizar, se creará una copia de la imagen que se encuentra en el área de diseño.

Para ello se accede a é por medio del nombre del objeto dinámico, en nuestro caso la variable *obj* tendrá el valor de *computer*.

```
parent.left.dd.elements[obj].copy();
var copieslength = parent.left.dd.elements[obj].copies.length;
var newobject=parent.left.dd.elements[obj].copies[copieslength-1];
newobject.show();
```

Con *parent.left.dd.elements[obj].copy()*, se accede al método del objeto creado en el área de diseño. Para acceder específicamente al objeto computer del área de diseño se utiliza la variable obj, que ha sido pasada como parámetro cuando se ha realizado la llamada a la función.

Una vez obtenido el objeto, se procede a llamar al método *copy()*, este método crea una copia del objeto padre. Es decir crea una nueva imagen del objeto computer y se asigna a la variable *newobject*. Posteriormente se mostraría el objeto para que el usuario pudiera verlo en el área de diseño y poder posicionarlo donde él desea.

Cuando el objeto ha sido creado, se procederá a pedir al usuario el nombre que desea asignarle. Para ello se definirá una propiedad al nuevo objeto *newobjet*, denominada *name*.

```
var newname=prompt("Introduzca el nombre",newobject.name);
newname=newname.replace(" ","_");
if (newname==null)
      newname=newobject.name;
newobject.name=newname
```

Para pedir el nombre de usuario, se usará la función de javascript *prompt*. Esta función mostrará una ventana emergente en la cual el usuario podrá introducir el nombre de la nueva computadora. En el caso que el nombre tuviera espacios en blanco, se sustituirán por el carácter _.

A continuación, como todos los objetos que se crearían serían iguales, se ha definido una propiedad en el objeto que le define el tipo al que pertenece. La propiedad será *type*. En este caso tendrá el valor de *computer*.

```
newobject.div.type=obj;
```

Con estos pasos descritos, el nuevo computador ha sido añadido al área de diseño. Como puede haber multitud de computadores, se debe llevar un control de ellos. Para ello se ha creado una variable de tipo array llamada *computers*. Dicha variable contendrá el nombre de las computadoras añadidas al área de diseño.

```
MAXCOMPUTER=200
var computers=new Array(MAXCOMPUTER)
var ncomputers=0;
```

La variable ncomputers, llevará la cuenta de los computadores creados.

Cada vez que se añada un computador al área de diseño se añadirá su nombre a la estructura *computers*.

```
function add_computer(name){
 computers[ncomputers]=name;
 ncomputers++;
}
```

A continuación se mostrará los pasos a seguir para añadir un computador al área de diseño.

Paso1: Presionar el botón del computador

Paso2: Asignar nombre al computador

Paso3: Computador Añadido al área de diseño

Paso4: Repetir el proceso tantas veces como computadores se quieran añadir

Añadir Hub

El proceso para añadir un hub al área de diseño es muy similar al de añadir un computador. El primer paso es dotar al botón del hub de una acción:

```
<img name="hub" src="images/hub.png" onClick="create_copy('hub')" title="A&ntilde;adir Hub
```

Como se puede observar, el código es muy similar al que se ha explicado en la acción de añadir un computador, la única diferencia es que la llamada se hace pasando como argumento la palabra hub, que será utilizada en la función para crear las instancias de los hubs.

Por tanto el código es igual que al explicado en el apartado anterior, cabe descartar que cuando se crea la nueva instancia del hub, la propiedad del objeto *type* toma el valor de "hub".

Lo único que varía son las estructuras de datos en las cuales se lleva la información de los hubs añadidos al área de diseño. Para este caso se usarán las siguientes estructuras.

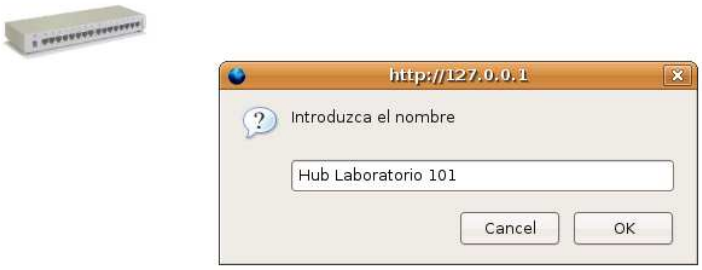
```
var hubs=new Array(MAXCOMPUTER);
var nhubs=0;

function add_hub(name){
        hubs[nhubs]=name;
        nhubs++;
}
```

Cada vez que se añada un hub al área de diseño, se almacenará su nombre usando la función *add_hub*. Dicha función será la encargada de almacenar el nombre en el array de hubs existentes, igualmente que se hacía con los computadores.

A continuación se muestran los pasos para añadir un hub al área de diseño:

Paso1: Presionar el botón del hub



Paso2: Asignar nombre al hub

Paso3: Repetir el proceso tantas veces como hubs se quieran añadir

Como se puede observar en la imagen anterior, el usuario puede disponer de hub y routers para diseñar su topología.

Añadir Router

El último elemento que queda por añadir al área de diseño es el router. La mecánica para añadir un router al área de diseño es similar a la vista anteriormente con un computador y con un hub.

Lo primero que debe hacer el usuario es pulsar sobre la imagen del router en el menú de acciones. El código HTML será:

```
<img name="router" src="images/router.png" onClick="create_copy('router')" title="A&ntilde
```

Lo único que cambia con respecto a añadir un computer o un hub que el parámetro de la función *create_copy*.

La única diferencia existente con la adicciones explicadas anteriormente es como se almacena el nombre de los routers que el usuario ha ido añadiendo en su topología, para ello se usará un array en el que cada posición se introducirá los nombres de los routers.

```
var routers=new Array(MAXCOMPUTER);

function add_router(name){
        routers[nrouters]=name;
        nrouters++;
}
```

Cada vez que se añada un router al área de diseño, se almacenará su nombre usando la función *add_router*. Dicha función será la encargada de almacenar el nombre del router en el array. Los pasos para añadir un router al área de diseño son los siguientes:

Paso1: Presionar el botón del Router

Paso2: Asignar nombre al Router

Paso3: Repetir el proceso tantas veces como hubs se quieran añadir

El router es el último elemento que se puede añadir a una topología de red, con ello queda concluido la función de añadir elementos al área de diseño.

Crear Enlaces

Una vez añadidos todos los elementos que compondrán nuestra topología de red, se procederá a realizar los enlaces de las computadoras, hubs y router para diseñar las redes que se desean simular.

Para crear un enlace lo primero que hay que hacer es seleccionar el icono en el menú de acciones

```
<img name="cable" src="images/cable.png"  title="Conectar" onClick="create_link() ">
```

Una vez que se presiona el icono de creación de enlace, se ejecutaría la función en javascript *create_link*.

```
function create_link(){
        if (parent.RUN){
                alert("Aplicación ejecutandose, no se puede crear el link");
                return;

        }
        if (!LINK){
                LINK=Boolean(1);
                show_message("Crea enlaces entre elementos");
        }
        else{
                LINK=Boolean(0);
                parent.left.dd.elements.status.hide();
                clicks=0;
        }
}
```

Como se puede observar, la función *create_link* únicamente comprueba la variable LINK. Dicha variable, indica si esta presionado el botón de crear enlace. Por defecto el valor de la variable LINK es FALSE, que quiere decir que no esta presionado el botón de crear enlace. En el momento que se presiona se le asignará el valor de TRUE y se procedería a crear los enlaces de los elementos añadidos al área de diseño. Dicha función también comprueba si se esta ejecutando la aplicación, en ese caso no se podrá añadir ningún enlace entre elementos.

La acción que debe realizar el usuario para poder crear un enlace, es la habitual que se encontraría en cualquier aplicación de escritorio. Se pincha en los elementos que se quiera enlazar y el enlace quedaría creado. Para ello cada vez que se crea un objeto en el área de diseño, por medio de la función *create_copy*, se le añade a cada objeto un evento llamado *onclick*.

```
newobject.div.onclick=link;
```

Este evento hace un llamada a la función *link* cada vez que se hace un click en un objeto.

```
function link(){
if (parent.LINK){
//se crea el link
}
```

Lo primero que hará función es comprobar si esta presionado el botón de crear el enlace, es caso contrario la función no realizará ninguna tarea.

Como el procedimiento para crear el enlace entre dos elementos es presionar primero en el primer objeto y luego en el segundo, se procederá a determinar si el objeto pulsado ha sido el primero o el segundo. Para ello existe un variable llamada *clicks* que nos dice si ha sido el primer click o el segundo.

```
 if (clicks==0){ //ha hecho el primer click
                        x1=parent.left.dd.elements[this.name].x;
                        y1=parent.left.dd.elements[this.name].y;
                        type1=this.type;
```

```
                        name1=this.name;
                        clicks++;
                }
```

Si ha sido el primer click, se almacena en las variables x1,y1 las coordenadas del primer elemento que compondrá el enlace. También se almacena el nombre y el tipo (computer,router,hubs) del elemento en el cual se ha realizado el click. Una vez almacenado los valores apropiados se incrementa la variable *clicks*, para que el siguiente elemento que se pinche sea el segundo elemento que compondrá el enlace.

```
if (clicks==1){
                        x2=parent.left.dd.elements[this.name].x;
                        y2=parent.left.dd.elements[this.name].y;
                        type2=this.type;
                        name2=this.name;
}
```

Una vez que el usuario presiona sobre el segundo elemento que compone el enlace, se procedería a almacenar las coordenadas, el nombre y el tipo del elemento.

Una vez que se posean los dos elementos que compondrán el enlace, así como la información relativa de cada elemento, se procederá a realizar el propio enlace.

Al empezar, se comprobará que el usuario no ha realizado un enlace incorrecto. Determinamos como enlaces incorrecto a los siguientes tipos de enlaces:

- Enlaces entre el mismo elemento. No se podrá realizar enlaces entre el mismo computador, hub o router.
- Enlaces entre computadoras. No se permitirá enlaces entre dos computadoras, para unir dos computadoras se utilizará un hub.
- Enlaces entre un router y una computadora. No se permitirá realizar enlaces entre routers y computadores. En el caso que se quiera realizar este tipo de enlaces se usará un hub para llevarlo a cabo.
- Un computador no puede tener mas de un enlace.

Téngase en cuenta que en el momento que se intente realizar un enlace erróneo, se le notificará al usuario que el enlace no se puede realizar y se anulará la información relativa al enlace que se ha querido crear. La notificación del error, se le mostrará al usuario por medio de una ventana emergente.

```
if (name1==name2){
              alert("No se puede realizar el enlace");
              LINK=Boolean(0);
              clicks=0;
              parent.left.dd.elements.status.hide();
              return ;
              }
if ((type1=="computer") && (type2=="computer")){
               alert("Este enlace no se puede realizar");
```

```
                LINK=Boolean(0);
                clicks=0;
                parent.left.dd.elements.status.hide();
                return ;
                }
if ((type1=="computer") && (type2=="router")){
                alert("Este enlace no se puede realizar");
                LINK=Boolean(0);
                clicks=0;
                parent.left.dd.elements.status.hide();
                return ;
                }
if ((type1=="router") && (type2=="computer")){
                alert("Este enlace no se puede realizar");
                LINK=Boolean(0);
                clicks=0;
                parent.left.dd.elements.status.hide();
                return ;
                 }
if ((type1=="computer")){
                if (existsLink(name1)){
                        alert("Un computador no puede tener mas de un enlace");
                        LINK=Boolean(0);
                        clicks=0;
                        parent.left.dd.elements.status.hide();
                        return;
                        }
                 }
if ((type2=="computer")){
                if (existsLink(name2)){
                        alert("Un computador no puede tener mas de un enlace");
                        LINK=Boolean(0);
                        clicks=0;
                        parent.left.dd.elements.status.hide();
                        return;
                        }
                }
```

La función *existsLink* comprueba que un computador no tiene mas de un enlace.

Una vez que se ha comprobado que el enlace se puede realizar, se procederá a realizar el enlace en sí. Para ello se ha creado una clase, llamada *ClassLink*. La definición de la clase se hace de la siguiente manera.

```
function ClassLink(source1,source2,type1,type2){
        this.source1=source1;
        this.source2=source2;
        this.type1=type1;
```

```
        this.type2=type2;
}

var linkscomputer=new Array(MAXCOMPUTER);
```

La clase contendrá las propiedad de los nombres de los elementos que compondrán el enlace así como el tipo de dicho elementos. También se creará una variable llamada *linkscomputer*. Dicha variable almacenará todos los links que ha realizado el usuario para diseñar la topología de red.

Para crear y almacenar la información relativa a un link, se usará la función *addLink*. Dicha función recibe los nombres y el tipo de los elementos que comprendren el link, creará la clase link y lo añadirá a variable *linkscomputer*.

```
function addLink(source1,source2,type1,type2){
        var linkaux=new ClassLink(source1,source2,type1,type2);
        linkscomputer[nlinkscomputer]=linkaux;
        nlinkscomputer++;
}
```

Una vez almacenado el enlace en las estructuras de datos necesarias, se procederá a dibujar gráficamente el enlace en el área de diseño. Para ello se ha usado la librería de javascript *wz_jsgraphics.js*.

Lo primero que se ha realizado es crear el área de dibujo para poder pintar los enlaces, ese área de dibujo corresponderá al mismo HTML del área de diseño. Para dibujar un enlace visual en el HTML se usará la función *draw_link*, la cual recibirá las coordenadas de la linea que representará el enlace que se pretende dibujar.

```
jg_doc=new jsGraphics();

function draw_link(x1,y1,x2,y2){
      jg_doc.setColor("#000000");
      jg_doc.setStroke(2);
      jg_doc.drawLine(x1,y1,x2,y2);
      jg_doc.paint();


}
```

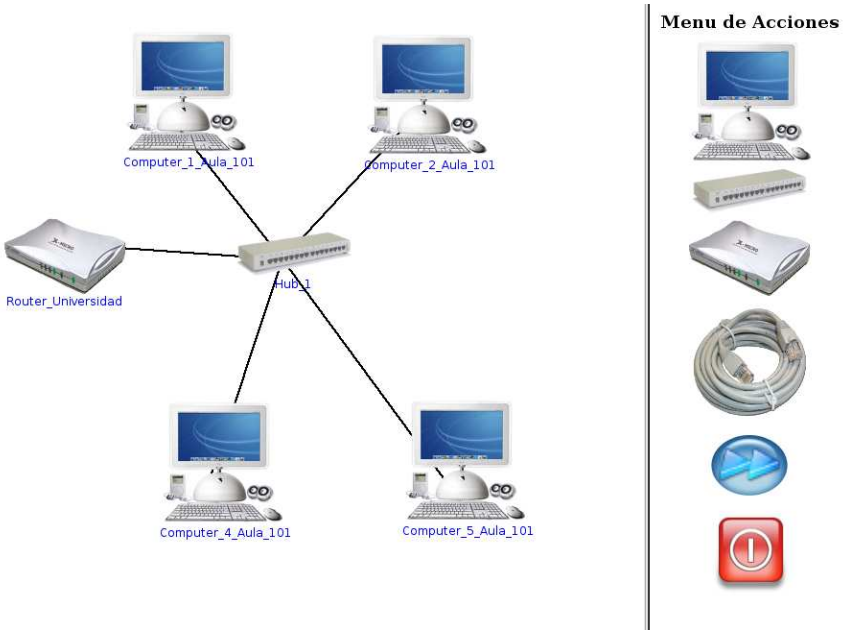
Cuando el enlace ya esta dibujado en el área de diseño, ya estaría completado el proceso de creación de un enlace.

Los pasos para realizar un enlace son los siguientes:

Paso1: Presionar el botón de creación de enlace

Paso2: Presionar sobre el primer elemento que compondrá el enlace

Paso3: Presionar sobre el segundo elemento que compondrá el enlace. Enlace Creado



Paso4: Repetir el proceso tantas veces como enlaces se quieran añadir

Barra de Estado

Como bien se ha comentado anteriormente nuestro sistema contendrá una barra de estado que indicará al usuario el estado actual de la aplicación.

La barra de estado en Javascript esta definida por la variable *status* y para cambiar la información se hará con la función *show_message* de la siguiente forma.

```
show_message("Aplicación Ejecutandose}
```

La definición de la función *show_message* se hará de la siguiente forma:

```
function show_message(msg){
  parent.left.document.getElementById("status").innerHTML=msg;
  parent.left.dd.elements.status.show();
}
```

Para ocultar la barra de estado se usará el siguiente código.

```
parent.left.dd.elements.status.hide();
```

Con esta funcionalidad quedarán implementados todos los objetivos propuestos para este prototipo.

Arranque de los nodos y parada de los nodos

Esta funcionalidad será implementada en prototipos posteriores.

4.4. Prototipo 2: Interacción entre Usuario y Servidor Web

4.4.1. Especificación

Una vez construido el interfaz Web de usuario, se procederá a realizar un prototipo que añadirá funcionalidad al prototipo realizado anteriormente. Este nuevo prototipo debe permitir la comunicación entre el interfaz de Usuario y la máquina donde se aloja el software para la gestión de nodos Netkit.

Uno de los objetivos prioritarios de este proyecto fue la separación en distintas máquinas del interfaz de usuario con el software de gestión de nodos, para ello se ha querido ubicar los dos módulos en máquinas distintas. Para la comunicación de las máquinas se ha decidido usar el protocolo HTTP, ya que es el protocolo usado en el Web.

Este prototipo pretende la comunicación entre el interfaz de usuario con la máquina que alberga el software de la gestión de los nodos Netkit. Para ello, cuando el usuario quiera ejecutar los nodos remotamente, el interfaz debe comunicar al software de gestión de nodos la topología elegida mediante este interfaz. En este prototipo únicamente nos centraremos en la comunicación realizada entre el interfaz Web y como el servidor recibe esa información.

Los pasos que se harán en este prototipo serán los siguientes

1. Una vez que el usuario haya realizado el diseño de la red que desea simular, pulsará el botón de ejecución.

2. El interfaz generará la información de la topología creada por el usuario.

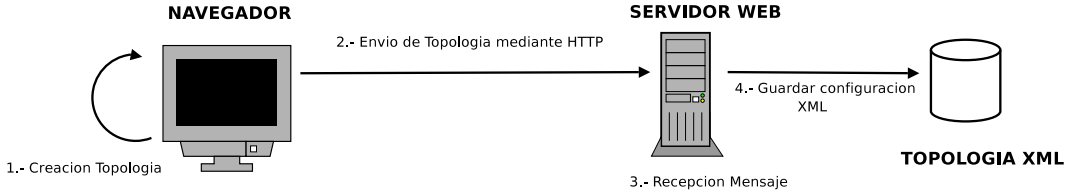
3. La información de las redes creadas será enviada por medio del protocolo HTTP al servidor Web que está disponible en la máquina que lleva la gestión de los nodos virtuales.

4. El servidor Web recibirá la información y la almacenará para posteriormente simular la red mediante Netkit.

4.4.2. Diseño

Arquitectura

A continuación se definirá la arquitectura elegida para el diseño del prototipo 2. En esta arquitectura entran en juego el navegador Web (interfaz de usuario) y un servidor web.



Arquitectura prototipo 2

Téngase en cuenta que este prototipo se ejecuta únicamente cuando el usuario pretende simular el diseño que anteriormente ha realizado mediante el interfaz de usuario. Los pasos que este prototipo realizará serán los siguientes:

1. El usuario presiona el botón de simulación.

2. El interfaz de usuario genera la topología de red que el usuario ha diseñado.

3. Envía por medio del protocolo HTTP dicha topología.

4. El servidor Web será el encargado de recibir la topología y almacenarla para posteriormente poder simularla mediante Netkit.

Comunicación por medio de HTTPRequest y HTTP

Para la comunicación entre el interfaz de usuario y el servidor, usaremos el objeto de javascript HTTPRequest. Este objeto permite la comunicación asíncrona entre un cliente (Interfaz Web) y un Servidor (Software de Gestión de Nodos).

Para el envío de información se usará el protocolo HTTP, protocolo usado en el Web. El Interfaz Web enviará, cada vez que el usuario quiera ejecutar un diseño, una petición HTTP al servidor. En la máquina que hará de servidor se estará ejecutando un software que realizará la tarea de servidor web, dicho software recibirá la petición realizada del usuario, dicha petición contendrá la topología que el usuario pretende simular mediante Netkit.

Se ha elegido HTTP ya que es un estándar de protocolo usado en el Web y porque ningún navegador va a tener problema al interactuar con el servidor por medio de este protocolo.

Servidor Web Apache

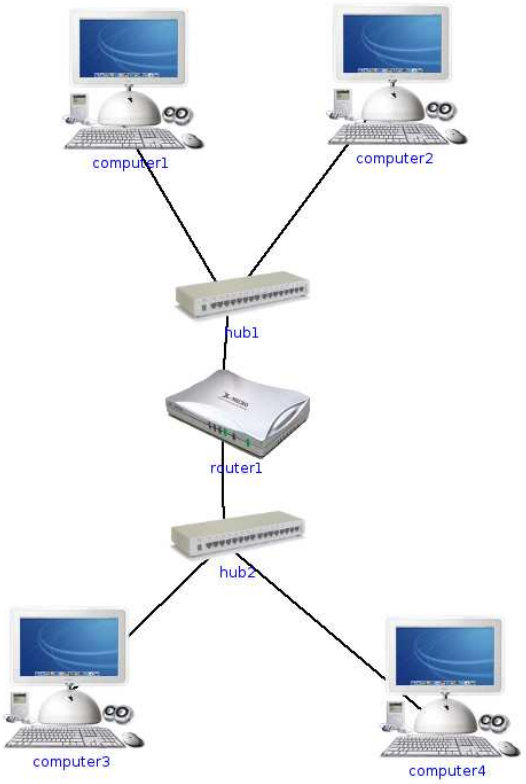
Para la recepción de la topología de red por medio del protocolo HTTP, se necesita instalar un software que haga de servidor web. Este software únicamente se encargará de recibir la red diseñada por el usuario. En prototipos posteriores veremos como el servidor Web se encargará de gestionar la información recibida, pero no es cometido del prototipo que pretendemos diseñar.

Hemos elegido Apache como servidor Web, ya que Apache es un servidor muy extendido en la comunidad de internet y porque nos permite una facilidad de programación que otros servidores web carecen. Apache permite ejecutar código en distintos lenguaje, en concreto Python será el lenguaje elegido para gestionar la información recibida por el navegador en el servidor Web.

Topología de Red en XML

Una vez que el usuario ha creado la topología de red que desea simular, esta topología debe ser enviada al servidor para que la pueda simularse mediante Netkit. Para ello se ha decidido crear un XML que contendrá la información de la topología de Red que el usuario ha diseñado.

Imaginemos la siguiente red creada por el usuario:



Topología de Red

Como se puede observar existen dos redes distintas. La Red A formada por el computer1, computer2 y router1, y la Red B formada por el computer3, computer4 y router2.

Una vez creado el diseño, se debe diseñar un XML que indique la topología de red creada. En el caso concreto de la red anterior, el XML generado será el siguiente:

```
<netweb>
        <elements>
                <name>router1</name>
                <name>computer1</name>
                <name>computer2</name>
                <name>computer3</name>
                <name>computer4</name>
        </elements>
        <nets>
                <net>
                        <name>
                        A
                        </name>
                        <elements>
                                <name>computer1</name>
                                <name>computer2</name>
                                <name>router1</name>
                        </elements>
```

```
                </net>
                <net>
                        <name>
                        B
                        </name>
                        <elements>
                                <name>computer3</name>
                                <name>computer4</name>
                                <name>router1</name>
                        </elements>
                </net>
      </nets>
</netweb>
```

La etiqueta *elements*, contendrá todos los elementos de la red. La etiqueta *nets* contendrá las redes locales, cada una englobada en una etiqueta *net* y que contendrá los elementos que componen esa red local.

Una vez que el interfaz Web ha creado el XML que desea simular, será enviado al servidor para que inicie la simulación de la Red por medio de Netkit.

4.4.3. Implementación

Construcción XML

Cuando el usuario presiona el botón de inicio de la simulación, se ejecutará la función de javascript *run()*.

```
<img name="arrancar" src="images/run.png" onClick="run()" title="Arrancar">   </td>
```

La función *run()* iniciará el proceso de construcción del XML que corresponde a la topología de red que el usuario ha creado. Lo primero que hará dicha función es comprobar si puede iniciar la ejecución.

```
if (parent.left.dd.elements["computer"].copies.length==0){
  alert("No hay maquinas para poder ejecutar la aplicacion");
  return;
}
if (parent.RUN){
  alert("La aplicacion ya esta ejecutandose");
  return;
}
```

Primero se comprueba si hay alguna máquina en el diseño y si la aplicación no esta ejecutándose. En caso afirmativo informará al usuario por medio de un mensaje emergente que no se puede iniciar la ejecución.

Una vez que se ha comprobado que se puede realizar la simulación, se procederá a crear el XML. Para ello se ha usado el objeto *Microsoft.XMLDOM* que permite crear un parse de XML.

```
function create_xml(){
        if (document.implementation && document.implementation.createDocument)
        {
                xmldoc = document.implementation.createDocument("", "", null);
        }
        else if (window.ActiveXObject)
        {
                xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
                xmlDoc.onreadystatechange = function () {
                        if (xmlDoc.readyState == 4) createTable()
                };
        }
        else
        {
                alert('Your browser can\'t handle this script');
                return;
        }
}
```

Una vez que se ha creado el xml llamado **xmldoc** se procederá a construir el XML.

Lo primero que realizará es añadir los elementos que componen el diseño al XML, bajo la etiqueta *elements*.

```
 var netwebxml = xmldoc.createElement("netweb");
        var elements=xmldoc.createElement("elements");
        for (i=0;i<nrouters;i++){
                routername=routers[i];
                var namexml=xmldoc.createElement("name");
                var textxml=xmldoc.createTextNode(routername);
                namexml.appendChild(textxml);
                elements.appendChild(namexml);
        }
        netwebxml.appendChild(elements);
        for (i=0;i<ncomputers;i++){
                computername=computers[i];
                var namexml=xmldoc.createElement("name");
                var textxml=xmldoc.createTextNode(computername);
                namexml.appendChild(textxml);
                elements.appendChild(namexml);

        }
```

Primero de todo se crea las etiquetas *netweb* y *elements*, bajo esta última etiqueta se introducirán el nombre de los elementos que componen el diseño del usuario. Téngase en cuenta que los hubs no se representan como máquina, sino que únicamente se usan para simular la unión de varios elementos y por eso no se almacenarán en el XML. Como se puede observar en el código javascript anterior el nombre de los computadores y los routers están contenidos en las variables *computers* y *routers* respectivamente. Una vez introducidos en el XML los nombre se procederá a introducir la información de las redes existentes en el diseño.

Una vez creado el XML que contendrá la topología de la red, ya se estará dispuesto a enviárselo al servidor.

Objeto HTTPRequest

Como bien se ha comentado anteriormente, para la comunicación entre el cliente y el servidor se va a usar el objeto HTTPRequest de Javascript.

El objeto HTTPRequest será el encargado de mantener la comunicación entre el cliente y el servidor. La definición del objeto se hará de la siguiente forma

```
function http_request() {
    try {
        xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
        try {
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (E) {
            xmlhttp = false;
        }
    }
    if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
        xmlhttp = new XMLHttpRequest();
    }
    return xmlhttp
}
```

Una vez definido el objeto, la instanciación se realizará cada vez que quiera hacer uso de él.

Envío XML por medio de HTTPRequest y HTTP

Cuando se haya definido el XML y el objeto HTTPRequest, se ejecutará la función *send_xml*. Dicha función será la encargada de enviar la información al servidor por medio de HTTP y utilizando el objeto HTTPRequest.

```
function send_xml()
{
    _objetus=http_request()
    _URL_="netweb.py?request=run";
    _objetus.open("PUT",_URL_,true);
    _objetus.onreadystatechange=function() {
        if(_objetus.readyState==4)
        {
            if(_objetus.status==200)
            {
                 null;
            }
            else
            {
                alert("ERROR");
```

```
            }
        }
        }
    _objetus.send(xmldoc);
    return
}
```

Como se puede observar, *_objetus* es una instancia del objeto HTTPRequest. La variable *_URL_* contendrá a quien va redirigida la petición HTTP que se va a generar. En este caso será el fichero netweb.py que será el script en python que se encontrará en el lado del servidor y que será el encargado de recoger la información enviada desde el interfaz. Como puede haber varios tipos de peticiones, cada petición quedará definida por la variable *request*, en nuestro caso tendrá el valor de *run*, ya que la acción que queremos realizar es la de arranque de los nodos en el servidor. Por ultimo se enviará el XML por medio del método *send* del objeto *_objetus*, este método recibirá un parámetro. Dicho parámetro será la variable *xmldoc*, la cual contendrá el XML de la topología diseñada por el usuario.

Configuración Apache y Mod Python

Una vez que se ha enviado el XML por medio del protocolo HTTP, el servidor Web será el encargado de recoger esa información. Como bien se ha comentado anteriormente, el servidor que vamos a usar es el servidor Web Apache. El servidor Web Apache puede ejecutar varios lenguajes de programación lanzados desde una petición HTTP, entre ellos se encuentra python. Python será el lenguaje encargado para tratar la información recibida del interfaz Web. Para ello se necesita instalar y configurar un módulo existente para Apache, que permite la ejecución de código escrito en Python. El módulo recibe el nombre de *mod_python*.

Lo primero es instalar Apache con mod_python. Una vez que la instalación de mod_python ha sido correcta se procederá a la configuración. Como bien se ha visto cuando se invoca al método HTTPRequest en la petición de arranque de nodo, se hace un método PUT a *netweb.py*. Dicho fichero será el que se ejecute y recoja la información del XML recibido del interfaz Web.

Para configurar mod_python en Apache habrá que añadir al fichero /etc/apache2/mods-available/mod_python.load las siguientes lineas

```
LoadModule python_module /usr/lib/apache2/modules/mod_python.so
<Directory /var/www/netweb/>
        AddHandler mod_python .py
        PythonHandler netweb
        PythonDebug On
</Directory>
```

Estas lineas indican a Apache que cada vez que se reciba una petición a *netweb.py* se debe interpretar el código como lenguaje en Python.

Recepción del XML en Apache

Una vez configurado Apache para que ejecute código en Python, se procederá a realizar el script que reciba el XML de la topología que el usuario quiere simular. Téngase en cuenta que

cada vez que se recibe una petición al fichero netweb.py este ejecuta el código de su manejador *handler* recibiendo como argumento el objeto *req*. Dicho objeto contendrá la información de la petición recibida.

```
from mod_python
import apache,util
import parse_netweb
telnetsopen={}
def handler(req):
        form=util.FieldStorage(req)
        request=form["request"]
        if request=="run":
                s = req.read()
                telnetsopen={}
                xmldoc=s
                while s:
                        s = req.read()
                        xmldoc=xmldoc+s

                f=open("/tmp/netweb.xml","w")
                f.write(xmldoc)
                f.close()
```

Lo primero que se realiza es sacar el tipo de la petición y almacenarla en la variable *request*. En nuestro caso la recepción del XML que contiene la topología de red que el usuario quiere simular, corresponde al valor *run*. Posteriormente se empieza a leer de la conexión que se ha recibido y se almacena en la variable *xmldoc* el contenido del XML recibido por el usuario. Dicho XML será almacenado en el fichero XML */tmp/netweb.xml*.

En este momento el servidor ya posee la información de la topología de red que quiere simular el usuario. Dicha información será utilizada por el prototipo posterior para simular la red diseñada.

4.5. Prototipo 3: Ejecución Remota de los nodos virtual

4.5.1. Especificación

Este prototipo va a llevar a cabo la implementación de todo el software de la gestión de los nodos virtuales mediante Netkit. Dicho software pretende ejecutar la topología de red diseñada. La red que se pretende diseñar se encuentra en el fichero */tmp/netweb.xml*, dicha información ha sido recibida mediante el software desarrollado en el prototipo anterior.

Las características que debe tener este prototipo son las siguientes:

- Se debe permitir simular todos los nodos que el usuario ha diseñado.
- Los nodos simulados deben de estar conectados mediante la topología diseñada por el usuario.
- La simulación se llevará a cabo mediante el software que posee el entorno de Netkit.

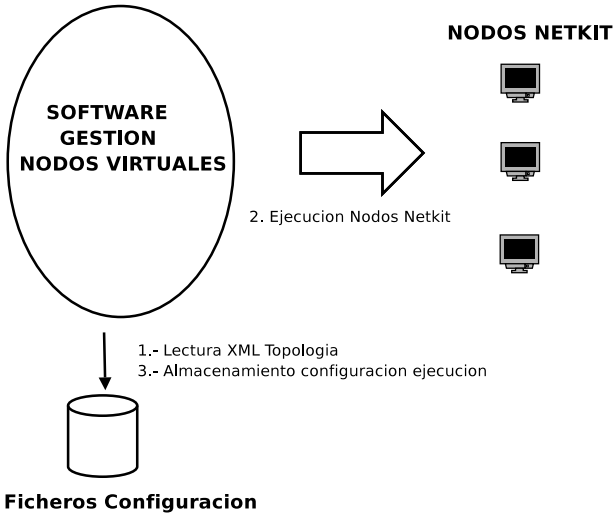
- Se debe poder tener un canal de comunicación con los nodos ejecutados, dicho canal será el encargado de ejecutar los comando que el usuario desea ejecutar en cada uno de los nodos simulados.

Una vez recibida la topología de red realizada por el usuario, se pretende realizar un software que gestione y ejecute la topología diseñada por el usuario.

4.5.2. Diseño

Arquitectura

La arquitectura de este prototipo estará centrada en el software de gestión de los nodos virtuales de Netkit, así como los propios nodos Netkit que el usuario pretende simular.



Arquitectura prototipo 3

Los pasos a seguir para la ejecución de este prototipo son los siguientes:

1. Se lee el XML recibido del Navegador, dicho XML contendrá la topología que se desea simular.

2. Una vez que se tiene claro la topología de red, se procede a simularla por medio de las herramientas que Netkit nos facilita.

3. Una vez que se ha simulado la topología, se guarda la información relativa a cada nodo. Esta información será necesaria para posteriormente ejecutar comando en cada uno de los nodos simulados.

Cabe recordar que una de las finalidades de este proyecto es la simulación de un entorno de red mediante Netkit. Una vez que el entorno este simulado, el usuario pretenderá acceder y ejecutar comandos en los nodos. Para ello se pretende guarda la información que nos permitirá comunicarnos con cada uno de los nodos.

Conexión Telnet con Nodo Virtual

Como bien hemos visto anteriormente, cada nodo Netkit exporta una Shell para poder interactuar con él. A ese interprete de comandos se puede acceder de distintas maneras, habitualmente se hace por medio de un terminal. Dicho terminal es simular al encontrado en cualquier máquina GNU/Linux. El problema que nos encontramos es que la interacción con un terminal normal de GNU/Linux es mas propia de un humano y no de un software, que es nuestro caso. Para ello Netkit permite accesos al nodo por otros medios, uno de ellos es mediante Telnet.

Cuando se ejecuta un nodo mediante Netkit, se le puede especificar que reciba comandos de GNU/Linux en un puerto TCP. Cuando el nodo es ejecutado, se lanzará un servidor Telnet en el puerto TCP especificado. A partir de ese momento se puede interactuar con el nodo por medio de una conexión Telnet, igualmente como si se estuviera interactuando con un terminal normal de GNU/Linux.

Con esta solución tenemos nuestro problema resuelto, ya que el software de gestión de nodos virtuales, tendrá por cada uno de los nodos una conexión telnet. Dicha conexión servirá para enviar comandos al nodo y recibir la respuesta de la ejecución de dichos comandos.

Ejecución de Nodos Virtuales

Como bien hemos visto anteriormente, cada nodo virtual debe estar escuchando en un puerto TCP, en dicho puerto recibirá los comandos que el usuario quiera ejecutar sobre él.

Para ello el software de gestión de nodos debe permitir ejecutar todos los nodos que el usuario ha diseñado mediante el interfaz web. Para ello Netkit dispone de un comando llamado **vstart**. Dicho comando permite ejecutar un nodo virtual escuchando en un puerto TCP determinado y con la configuración de red que el usuario ha diseñado.

4.5.3. Implementación

Instalación de Netkit

En este prototipo se va a llevar a cabo la implementación de el software que llevará la gestión de los nodos virtuales. Para ello la máquina donde esta alojada este software debe de tener configurado e instalado Netkit.

Lo primero que hay que hacer es descargarse de la página oficial de Netkit[2] todo el entorno.

- http://netkit.org/download/netkit/netkit-2.4.tar.bz2. Conjunto de Aplicaciones para usar con Netkit
- http://netkit.org/download/netkit-filesystem/netkit-filesystem-F2.2.tar.bz2. Sistema de Ficheros usado por Netkit
- http://netkit.org/download/netkit-kernel/netkit-kernel-K2.2.tar.bz2. El Kernel del sistema GNU/Linux que usará Netkit.

Una vez descargado los fichero se procederá a descomprimirlos, todos los ficheros serán almacenados en un directorio llamado *netkit2*. Una vez que estos ficheros estén descomprimidos procederemos a la configuración.

[2] http://netkit.org

Dentro del directorio *netkit2*, existe un fichero llamado *check_configuration.sh*, que chequeará nuestra configuración del sistema y nos avisará de los cambios que hay que hacer para poder ejecutar Netkit.

Nekit necesita crear o modificar ciertas variables de entorno, dependiendo siempre del Shell que se use, en nuestro caso *bash*.

1. Asignar a la variable *NETKIT_HOME* el path donde se encuentra la instalación de Netkit.
2. Añadir a la variable *PATH*, el path donde se encuentran los binarios de Netkit.
3. Añadir a la variable *MANPATH*, el path donde se encuentran las páginas de manual de Netkit.

Una vez que hayamos realizado estos pasos, ya podremos usar Netkit en nuestra máquina.

Téngase en cuenta que Netkit únicamente funciona en una máquina GNU/Linux.

Configuración del Software de Gestión de Nodos

Una vez que se tenga instalado Netkit, nuestro Software de Gestión de Nodos será el encargado de usarlo. Como cada instalación de Netkit es distinta, se ha creado un fichero de configuración donde se le asigna los distintos valores de las variable de entorno para que Netkit funcione correctamente. El fichero de configuración estará alojado en la máquina del servidor en */etc/netweb/config.cfg* y tendrá la siguiente forma.

```
[NETWEB]
NETKITHOME=/home/michael/source/netkit/netkit2
NETKITBIN=/home/michael/source/netkit/netkit2/bin
```

Como se puede observar el fichero de configuración es un simple fichero de texto el cual tiene los variable.

- **NETKITHOME**. Directorio donde se encuentra la instalación de Netkit.
- **NETKITBIN**. Directorio donde se encuentra los binarios de Netkit.

Si se quiere portar el software de gestión de nodos a otra máquina, que tiene la instalación de Netkit en otro directorio, únicamente habría que editar el fichero de configuración y cambiar las variables de valor.

Como el valor de estas variables tiene que ser utilizado por el programa de gestión de nodos, escrito en Python, se ha usado una librería llamada *ConfigParser*. Dicha librería nos permitirá leer las variables del fichero de configuración, para así poder usarlas en nuestro programa.

```
CONFIG=/etc/netweb/config.cfg
cfg=ConfigParser.ConfigParser()
cfg.readfp(file(CONFIG))
try:
      netkit_home= cfg.get('NETWEB','NETKITHOME'.lower())
except:
      print 'Incorrect value for NETKITHOME parameter'
```

```
try:
      netkit_bin= cfg.get('NETWEB','NETKITBIN'.lower())
except:
      print 'Incorrect value for NETKITBIN parameter'
```

A partir de este momento, las variables *netkit_home* y *netkit_bin* tendrán los valores introducidos en el fichero. Estos valores serán utilizados posteriormente para la ejecución de los nodos Netkit que pretenda simular el usuario.

Tratamiento del XML recibido

Una vez que el servidor ha recibido el XML con la topología de la red que se desea simular, este XML está almacenado en el fichero */tmp/netweb.xml*. Se procederá a parsear el XML. Para ello se va a usar la librería de parseo de XML en Python llamada **minidom**.

El fichero XML recibido del interfaz Web tiene un etiqueta llamada *elements*, la cual contiene las máquinas y routers que se quieren simular.

```
<elements>
                <name>router1</name>
                <name>computer1</name>
                <name>computer2</name>
                <name>computer3</name>
                <name>computer4</name>
</elements>
```

Primero de todo se irán cogiendo los nombre de los elementos usando el parser de la siguiente forma:

```
 for i in elementsxml.getElementsByTagName('name'):
                name=i.childNodes[0].data
```

Cada pasada al bucle, la variable *name* contendrá el nombre de los elementos de la red (router1,computer1, ...). Lo primero que se desea hacer es elegir un puerto TCP aleatorio donde escuchará el nodo mediante una conexión telnet.

```
portd=randrange(1024,9999)
```

La variable *portd*, contendrá un valor entero aleatorio comprendido entre 1024 y 9999. Este numero aleatorio servirá para lanzar el nodo Netkit escuchando en ese puerto TCP.

Una vez que se tendrá el puerto donde se va ejecutar el nodo Netkit, se procederá a saber a que redes pertenece dicho nodo. Para obtener dicha información contamos con el XML recibido del usuario, que nos indica las redes existentes y que nodos pertenecen a dicha Red. Usando el ejemplo anterior tenemos:

```
<nets>
      <net>
         <name>
         A
        </name>
        <elements>
```

```
            <name>computer1</name>
            <name>computer2</name>
            <name>router1</name>
        </elements>
    </net>
    <net>
        <name>
        B
        </name>
        <elements>
              <name>computer3</name>
              <name>computer4</name>
              <name>router1</name>
        </elements>
    </net>
</nets>
```

Como se puede deducir anteriormente, para el ejemplo del elemento *router1*, se dice que dicho elemento pertenece a la red A y B. Todo esto se deduce parseando el XML de configuración que hemos recibido.

La cadena resultante, será la linea que tengamos que ejecutar mediante el comando *vstart*. Vstart, es un comando existente en Netkit para ejecutar y simular nodos Netkit. Para el ejemplo anterior la cadena resultante será:

```
vstart router1 --port=9876 --eth0=A --eth1=B
```

Como se puede observar, la máquina que se quiere simular se llama router1 y tiene dos interfaces de red (eth0 y eth1). Dicho interfaces pertenecen a la red A y B respectivamente.

Este proceso se repetirá para todas las máquinas que quiera simular el usuario.

Ejecución nodos virtuales

Una vez que se ha obtenido el puerto TCP donde el nodo va a recibir peticiones telnet y las redes a las que pertenece, se procederá a ejecutar dichos nodos en la máquina del servidor.

Para ello se ejecutará la función *execute_node* que recibirá la linea de ejecución necesaria para ejecutar un nodo Netkit, así como las variable *netkit_home* y *netkit_bin*. Estas variables contienen los directorios donde se encuentra instalado Netkit en la máquina.

```
def execute_node(ejec,netkit_home,netkit_bin):
pid=os.fork()
if (pid==0):
      os.environ['NETKIT_HOME']=netkit_home
      os.environ['PATH']=os.environ['PATH']+":"+netkit_bin
      args=ejec.split(" ")
      os.execvp("vstart",tuple(args))
      os._exit(1)
os.wait()
```

Como se puede observar en el código anterior, por cada nodo Netkit se creará un proceso Hijo que ejecutará dicho nodo. Al final del proceso la máquina del servidor tendrá que tener ejecutando todos los nodos Netkit que el usuario ha querido simular. Cada nodo a su vez, dispondrá de una conexión TCP donde aceptará peticiones telnet.

4.6. Prototipo Final: Acceso a los nodos por medio del Web

4.6.1. Especificación

En el prototipo anterior se vio como poder ejecutar los distintos nodos Netkit que el usuario había diseñado en el interfaz Web. En este prototipo se intentará interactuar con los nodos que están disponibles en la máquina del servidor.

Para ello se pretenderá añadir al interfaz de usuario un Shell que permita al usuario poder interactuar con los distintos nodos, igualmente que si lo hiciera desde el propio terminal del nodo.

Las características que deberá tener este prototipo serán las siguientes:

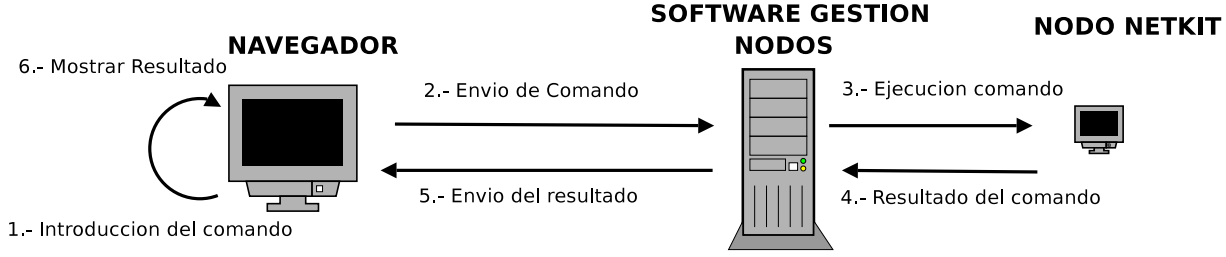
- El usuario podrá abrir un terminal desde el interfaz web para interactuar con el nodo que él haya elegido.
- El usuario podrá ejecutar cualquier comando GNU/Linux en el terminal de cada nodo por medio del interfaz Web.
- El usuario podrá visualizar el resultado de la ejecución del comando realizado en el nodo simulado.
- El servidor deberá poder guardar el estado de los nodos entre las interacción del usuario.

Todas estas características se podrán llevar a cabo siempre que estén ejecutándose los nodos en remoto.

4.6.2. Diseño

Arquitectura

La arquitectura de este prototipo estará centrada en la ejecución remota de comando GNU/Linux en las máquinas que el usuario ha simulado. La introducción del comando que el usuario quiere realizar como el resultado del comando ejecutado en el nodo Netkit, será realizada por el un Shell desde el navegador Web.



Arquitectura prototipo 4

Los pasos ha seguir serán los siguientes:

1. El usuario abre el Shell del nodo en el cual quiere ejecutar el comando, escribe el comando y pulsa un ENTER.
2. El comando que desea ejecutar viaja por la red en una petición HTTP.
3. El software de gestión de nodos recibe el comando y lo ejecuta en el nodo que ha determinado el usuario.
4. El resultado de la ejecución es recogido por el software de gestión de nodos.
5. Se muestra al usuario el resultado de la ejecución en el Shell Web del nodo.

Estos pasos se pueden repetir tantas veces como el usuario quiera, ya sea para configurar los interfaces de red de cada nodo, como cualquier comando que el usuario quiere ejecutar.

Interfaz de Usuario

La interacción entre el usuario con cada nodo se hará por medio de un Shell Web. Este shell web debe de tener las mismas ventajas que pudiera tener el usuario si ejecutará comando en un terminal normal de GNU/Linux. Para ello cada vez que el usuario quiera interactuar con un nodo Netkit remoto, se le mostrará el Shell de ese nodo para que el usuario pueda ejecutar tantos como comandos como él quiera, así como visualizar la ejecución del resultado de dichos comandos.

Envío de petición

Para el envío de la petición del comando que se desea ejecutar, se va a realizar mediante el objeto HTTPRequest de javascript y usando el protocolo HTTP.

Nótese que ahora habrá que distinguir el tipo de petición que se va a realizar, ya que en este instante tenemos dos tipos de peticiones. Una que arranca todos los nodos Netkit y otra que ejecuta comandos en cualquiera de los nodos que se han simulado.

Conexión Telnet con Nodos

Como bien hemos visto en el prototipo anterior, por cada nodo virtual que se ha ejecutado se tiene abierta una conexión Telnet para interactuar con el nodo. Así cada vez que reciba una petición de ejecución de un comando en un nodo, se procederá a escribir en dicha conexión el comando y leer el resultado de la ejecución.

Para ello el software de gestión de nodos virtuales debe poseer una conexión Telnet con cada uno de los nodos simulados.

4.6.3. Implementación

Creación XML para futuras conexiones

Al ser lanzados todos los nodos Netkit en un puerto TCP para futuras conexiones Telnet, se desea poder almacenar esa información para poder ser recuperada en un futuro.

Para ello se ha creado un fichero XML llamado */tmp/netweb.conf*, el cual contendrá los puertos de los distintos nodos que se han ejecutando con anterioridad.

Un ejemplo del fichero se muestra a continuación:

```
<?xml version="1.0"?>
<computers>
      <computer>
         <name>router1</name>
         <port>8772</port>
      </computer>
      <computer>
         <name>computer1</name>
         <port>4568</port>
      </computer>
      <computer>
         <name>computer2</name>
         <port>2135</port>
      </computer>
      <computer>
         <name>computer3</name>
         <port>9634</port>
      </computer>
</computers>
```

Este fichero de configuración será usado cada vez que se reciba la petición de ejecución de un comando en alguno de los nodos ejecutados en el servidor. Para ello con saber el nombre del nodo que se quiere interrogar, se podrá saber el puerto TCP para poder hacer la conexión telnet a él.

Creación de un Shell Web

Para la creación del Shell Web para la interacción con cada nodo, se ha buscado un interfaz muy similar al que pudiera tener un usuario cuando ejecuta Netkit en local.

Para acceder al Shell de cada nodo, únicamente habrá que realizar un doble Click en el nodo correspondiente. Para poder realizar esta tarea, cada vez que se hace una instancia de un nuevo objeto en el área de diseño, se le añade la función que se quiere ejecutar cuando se hace un doble click en el objeto.

```
newobject.div.ondblclick=parent.left.show_configuration;
function show_configuration(){
  change_terminal(this.name,this.type);
 }
```

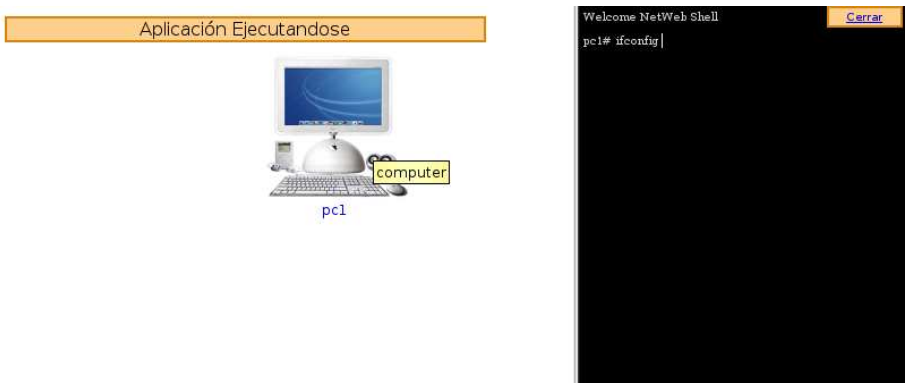
Cada vez que se realice un doble click en el elemento, se ejecutará la función *show_configuration*, que será la encarga de mostrar el terminal. Dicha función, en su definición hará una llamada a la función de javascript que abrirá la pantalla del Shell para el nodo dado. Esa función recibe el nombre de *open_terminal* y su definición será:

```
function open_terminal(){
 if (!parent.parent.RUN) {
      alert("No se puede abrir el terminal, primero deberias ejecutar la aplicacion");
      return;
```

```
}
   parent.right.location="shell.html";
}
```

Como se muestra en el código, el shell estará implementado en el fichero *shell.html*.

Esta función mostrará el terminal de Shell en la misma parte del interfaz donde se mostraba el menú de acciones.



Shell de NetWeb

Como se puede observar en la imagen, el interfaz que se ofrece por medio del navegador es muy similar al encontrado en un terminal de GNU/Linux.

Lectura de comando y Envío de comando

Una vez que se ha mostrado el terminal, se podrá introducir tantos comandos como el usuario desee. La interacción con cada nodo es similar a la encontrada en un terminal de GNU/Linux. Cada vez que se escribe un comando, se pulsa INTRO para ejecutarlo. Para ello se captura la cadena introducida por el usuario en el Shell Web y es enviada al servidor por medio de una petición HTTP.

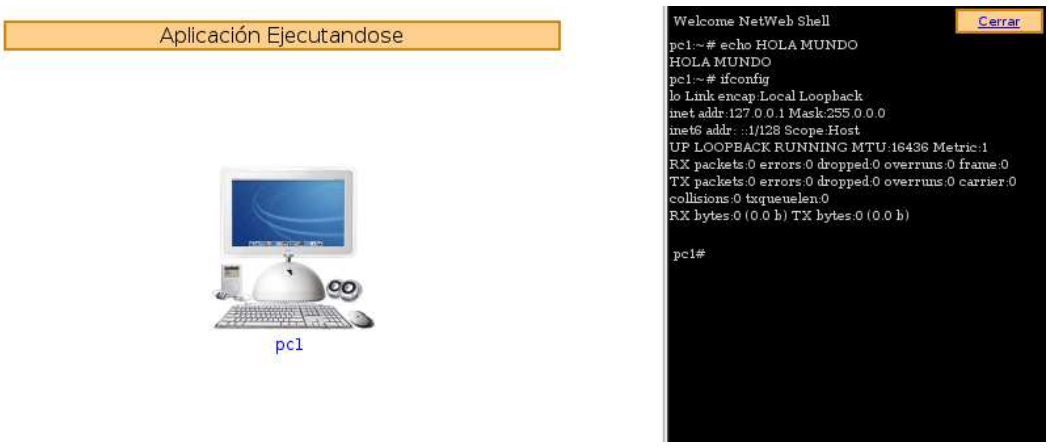
Para el envío de la petición se volverá a realizar por medio de objeto HTTPRequest. La función que se encarga de enviar la información al servidor es la función *send_request_computer*. Dicha función recibe dos parámetros, el primero el nombre de la máquina que se va a interactuar y el segundo el comando que se quiere ejecutar.

```
function send_request_computer(name,command){
    _objetus=http_request()
    _URL_="netweb.py?request=command&computer="+name;
    _objetus.open("PUT",_URL_,true);
    _objetus.onreadystatechange=function() {
     if(_objetus.readyState==4)
                {
                if(_objetus.status==200)
                {
                     result=_objetus.responseText;
                     printResult(name+":"+result);
                }
```

```
        else
           {
                alert("ERROR");
           }
     }
    }
    _objetus.send(command);
    return

}
```

Como se puede observar, la URL a la cual se le hace la petición es la misma que para la ejecución. La única diferencia es que ahora la variable request toma el valor de **command**.



NetWeb Shell

Recepción de petición en el servidor

Una vez que el interfaz Web ha enviado por HTTP la petición del comando a ejecutar en el nodo, se procederá a capturar esa información en el servidor. Esta captura la realiza el software de gestión de nodos virtuales.

```
if request=="command":
                computer=form["computer"]
                s=req.read()
                global telnetsopen
                result=parse_netweb.read_node_netweb(computer,s,telnetsopen)
                req.write(result)
```

Como se puede observar, primero se comprueba el tipo de petición, en nuestro caso la petición es de tipo *command*. Se almacena también el nombre del computador en el cual se quiere ejecutar el comando y el comando recibido (variable s). Una vez recibido todos los parámetros se hace una llamada a la función *read_node_netweb*, la cual devuelve el resultado de la ejecución del comando en el nodo, variable *result*. Ese resultado es devuelto al interfaz Web, que mostrará al usuario el resultado en el Shell del nodo donde ha ejecutado el comando.

Acceso mediante TELNET

Para la interrogación de los nodos vamos a usar el comando telnet de GNU/Linux. El único problema que conlleva es que únicamente se puede hacer una conexión con cada nodo, en el momento que la conexión telnet se cierre no se podrá acceder al nodo por medio de ninguna conexión Telnet.

Para la interacción entre el usuario con el nodo, por medio de telnet, se va a usar **pexpect**. Pexpect es una librería en Python del famoso expect de Unix. Expect permite la interacción entre un programa y distintas aplicaciones de interacción con humanos, como telnet, ssh, ftp ....

Por cada nodo ejecutado, se ejecutará un telnet y dicha conexión se almacenará indexando por el nombre del nodo, para ello existe la función addtelnet.

```
def addtelnet(name,telnetsopen,port):
        child = pexpect.spawn("telnet "+HOST+" "+str(port))
        child.expect('Welcome to Netkit')

        telnetsopen[name]=child
```

La variable que almacenará todas las conexiones telnet con todos los nodos, será la variable *telnetsopen*.

Una vez que se tiene almacenadas todas las conexiones telnet con cada uno de los nodos ejecutados, se procederá a ejecutar comandos en cada uno de los nodos.

Para ejecutar un comando en un nodo, se hace uso de la función *send_command*, la cual es invocada con el nombre del nodo, el comando se que quiere ejecutar y la conexión telnet. La función devolverá el resultado de la ejecución del comando en dicho nodo. Ese resultado será enviado al interfaz web para que muestre el resultado de la ejecución en el Shell del nodo.

```
result=send_command(computer,command,telnetsopen[computer])

[ ..... ]
def send_command(computer,command,child):
        child.expect(computer+':')
        child.sendline(command+'\n')
        child.expect(computer+':')
        result=child.before
        result=result.replace("\n","<br>")
        return result
```

La función *send_command*, haciendo uso de pexpect ejecutaría el comando y recogería el resultado. Nótese que se cambian los retornos de carro por la etiqueta html ¡*br*¿ . Esta etiqueta representa el retorno de carro en HTML.

Parada de los nodos

Una vez que el usuario ha interactuado con los nodos y quiere terminar la simulación, procederá a pulsar el botón de parada en el interfaz Web.

```
<img name="salir" src="images/exit.png" onClick="stop()" title="Parar">
```

Esta función en javascript, hace uso del objeto HTTPRequest el cual envía una petición a netweb.py con la variable request tomando el valor de stop. La variable request, téngase en cuenta que hace diferenciar al servidor que tipo de petición esta recibiendo del usuario.

El servidor recoge la petición

```
if request=="stop":
                result=parse_netweb.stop_netweb(telnetsopen)
```

La función *stop_netweb*, lo único que realiza es un *vhalt* de cada una de las máquinas virtuales que ha ejecutado el usuario. A partir de este momento no habría ninguna máquina virtual ejecutándose en el servidor, y por tanto no se podría realizar interacciones con ellas.

Capítulo 5

Validación y Pruebas

Una vez implementado el software nos disponemos a hacer una serie de pruebas para comprobar que el código desarrollado cumple con los requisitos expuestos al inicio del proyecto.

Para comprobar que el software cumple dichos objetivos, se van a realizar cuatro pruebas sobre el sistema. Cada prueba comprenderá una topología de red, dicha topología se probará exhaustivamente para certificar que el software desarrollado cumple los objetivos marcados.

Los pasos seguidos en cada prueba son los siguientes:

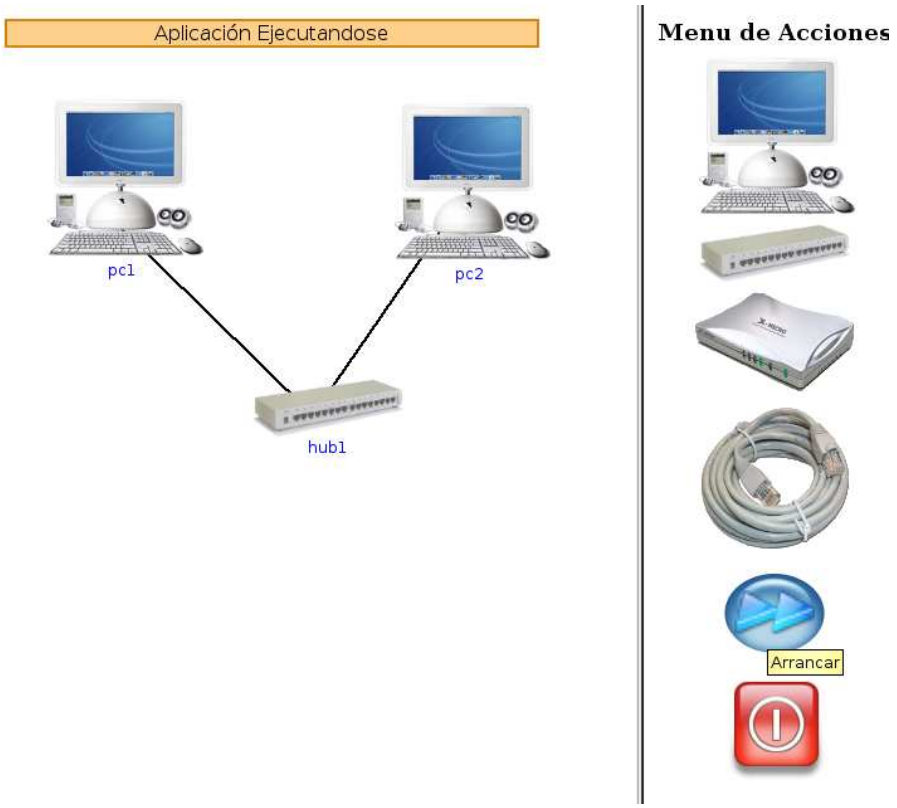
1. Especificar el problema.
2. Diseñar la topología de red mediante el interfaz.
3. Ejecutar el diseño en el sistema.
4. Asignar direcciones IP y crear tablas de encaminamiento (si las hubiera).
5. Realizar la prueba.
6. Comprobar el resultado de las pruebas.

5.1. Prueba1: Configuración de una Red Local

La primera prueba consiste en diseñar una topología de red formada por dos computadores(pc1 y pc2) unidos por hub (hub1). Estos dos ordenadores estarán unidos formando una red local.

La prueba consistirá en comprobar que los dos ordenadores están conectados en una red local y por tanto que existe conectividad entre ellos.

Lo primero de todo es diseñar la red en el interfaz Web del sistema y ejecutar el diseño realizado.



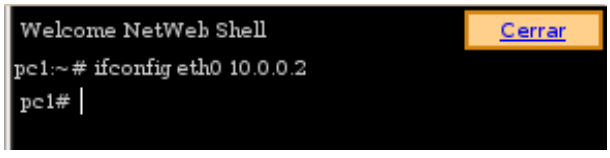
Diseño de la topología

Una vez diseñada la topología de red, se va a proceder a asignar direcciones IPa las máquinas.

pc1: *eth0*: 10.0.0.2

pc2: *eth0*: 10.0.0.3

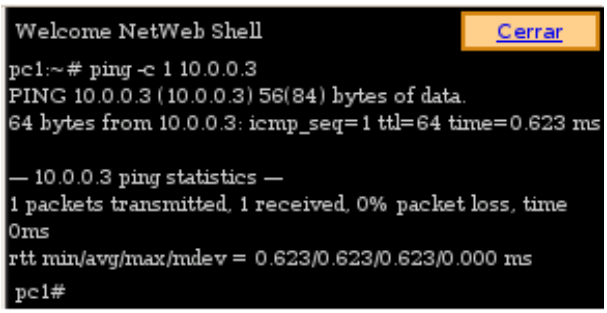
La asignación de las direcciones IPse hará por medio del shell que cada máquina posee mediante el interfaz. Para ello se usará el comando *ifconfig* para asignar una dirección IP al interfaz que deseemos, en nuestro caso *eth0*.



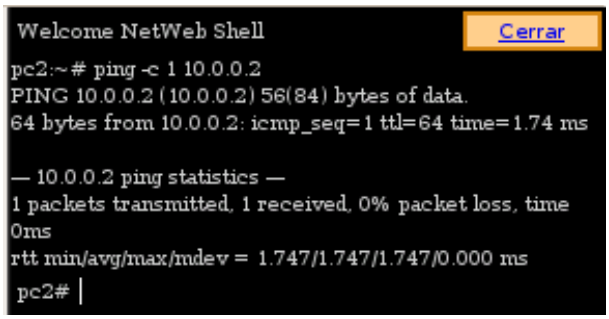
Asignación de IP a pc1

Asignación de IP a pc2

Una vez que las dos máquinas poseen dirección ip, vamos a comprobar que existe conectividad entre ellas. Para ello se utilizará el comando *ping* entre pc1 y pc2.



ping pc1 ->pc2



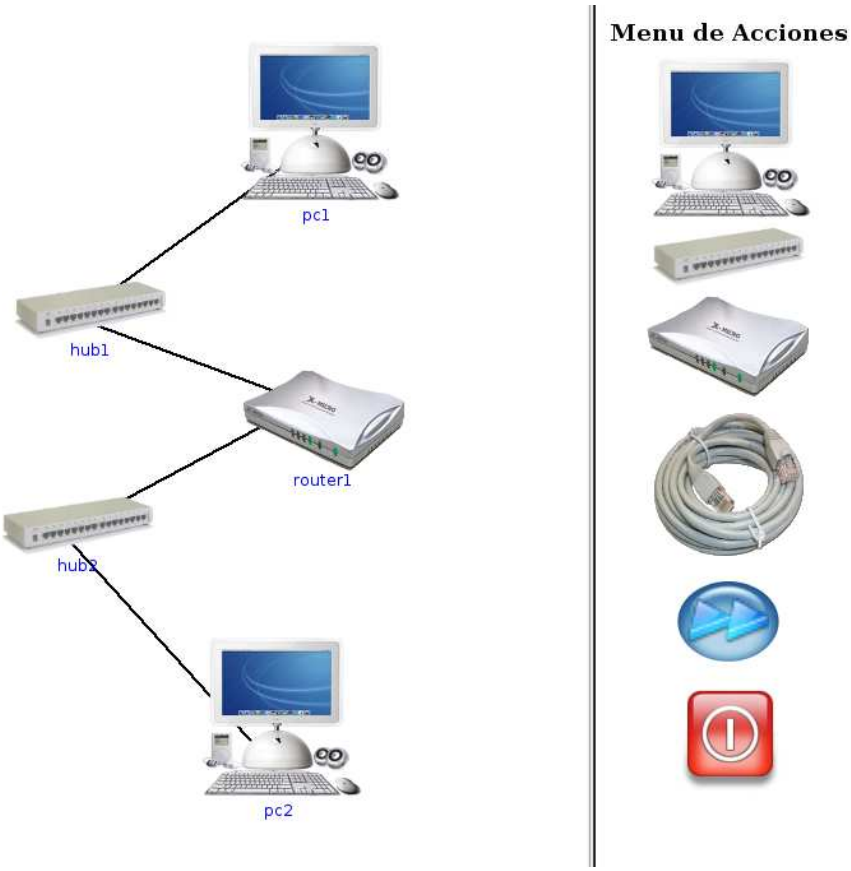
ping pc2 ->pc1

Una vez comprobado que existe conectividad entre las dos máquinas se da por concluida esta prueba con resultado satisfactorio.

5.2. Prueba2: Encaminamiento

En esta prueba se va a realizar un diseño de dos redes unidas por un encaminador. El resultado final debe ser que máquinas que estén en distintas redes puedan comunicarse por medio de un router.

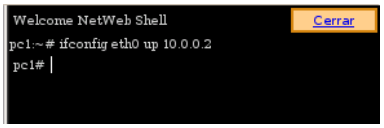
Primero diseñaremos la red en el interfaz web y ejecutaremos el diseño.



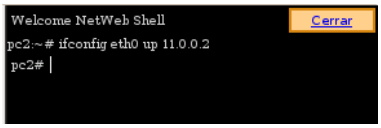
Diseño de la topología

A continuación asignaremos las direcciones IPa las máquinas de la siguiente forma:

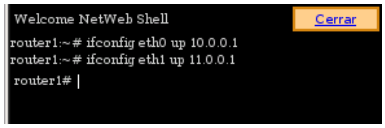
pc1: *eth0*: 10.0.0.2
pc2: *eth0*: 11.0.0.3
router1: *eth0*: 10.0.0.1 *eth1*: 11.0.0.1



Asignación de IP a pc1

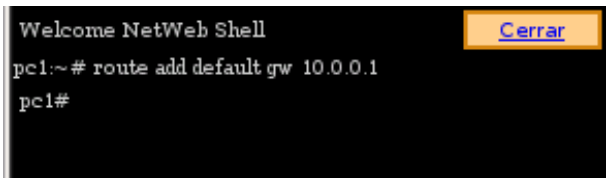


Asignación de IP a pc2

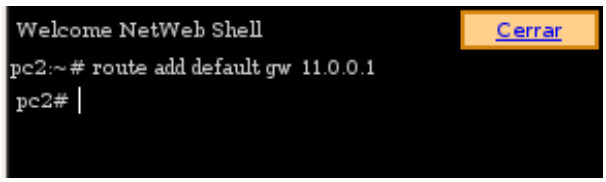


Asignación IP a router1

Una vez asignadas las direcciones IPs se va a proceder a configurar las puertas de enlace de pc1 y pc2, para ello se usará el comando *route*. Las puerta de enlace de pc1 será el interfaz eth0 del router1 y la de pc2 será el interfaz eth1 de router1.

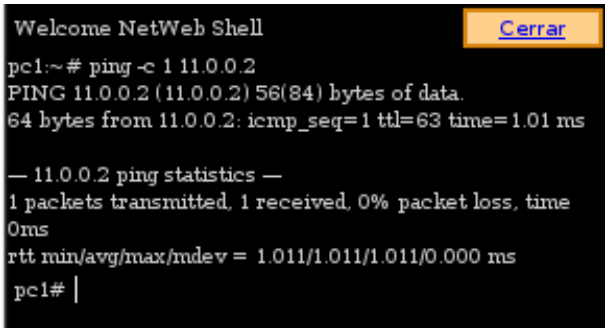


Crear puerta enlace pc1

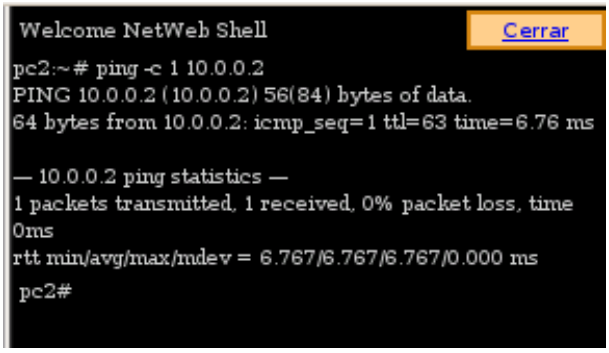


Crear puerta enlace pc2

Una vez creadas las puertas de enlaces entre las dos redes, vamos a comprobar la conectividad por medio del comando *ping*.



ping pc1 ->pc2



ping pc2 ->pc1

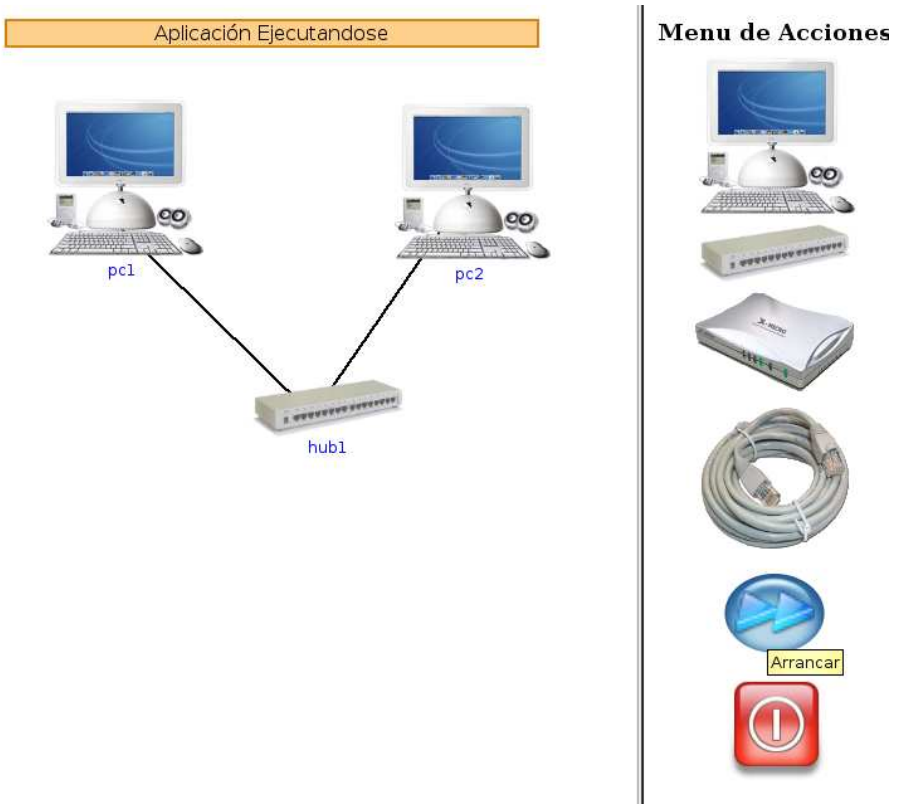
Como se puede observar existe conectividad entre pc1 y pc2 por medio de router1, con lo que el resultado de la prueba ha sido satisfactorio.

5.3. Prueba3: ARP

Para esta prueba, vamos a comprobar las tablas ARP de dos máquina. ARP es un protocolo de nivel de red responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP.

Para ello vamos a configurar una pequeña red formada por dos ordenadores y comprobar si cada máquina después de enviar un mensaje contiene la dirección Ethernet de la otra máquina.

Primero diseñaremos la red en el interfaz y posteriormente simularemos dicho diseño.



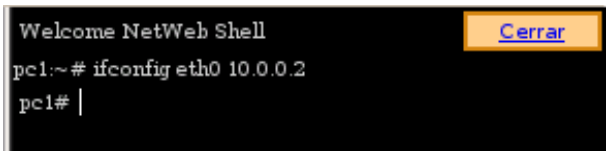
Diseño de la topología

Una vez diseñada la topología de red, se va a proceder a asignar direcciones IPa las máquinas.

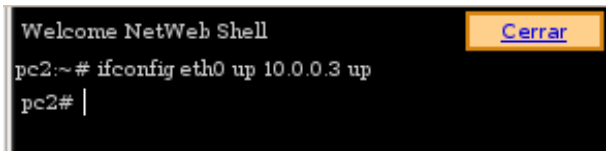
pc1: *eth0*: 10.0.0.2

pc2: *eth0*: 10.0.0.3

Como bien hemos comentando en las pruebas anteriores, usaremos el comando *ifconfig* para asignar las direcciones IP.

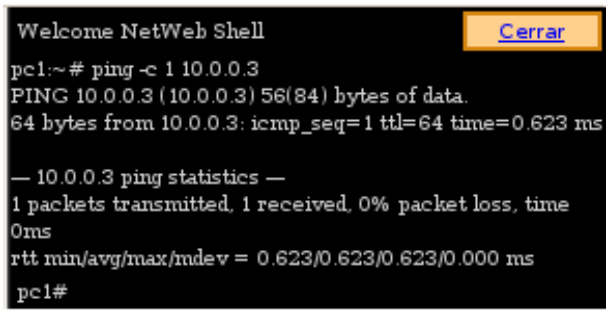


Asignación de IP a pc1



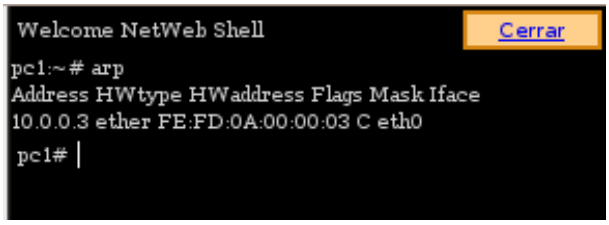
Asignación de IP a pc2

A continuación comprobaremos la conexión entre pc1 y pc2, usando el comando *ping* entre pc1 y pc2.

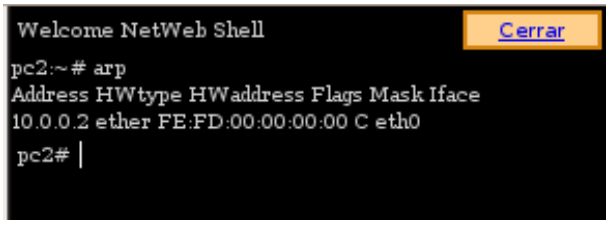


ping pc1 ->pc2

Vemos que existe conectividad entre pc1 y pc2, ahora procederemos a comprobar las tablas ARP de las dos máquinas. Para ello usaremos el comando *arp*.



Listado de Tabla ARP de pc1



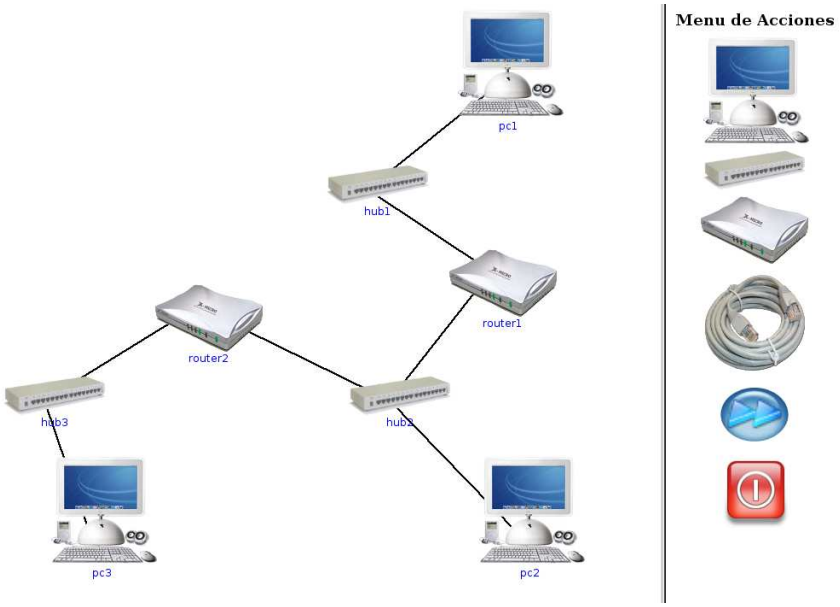
Listado de Tabla ARP de pc2

Como se puede observar en las imágenes anteriores, pc1 al hacer un ping a pc2 y no tener su dirección Ethernet realiza el protocolo ARP. Pc1 pregunta cual es la dirección Ethernet de la máquina con dirección IP de pc2. Responde pc2 con su dirección Ethernet y se apunta la dirección Ethernet de pc1 al igual que pc1 se apunta la dirección Ethernet de pc2 al responderle mediante el protocolo ARP. Ejecutando el comando *arp* en ambas máquinas se puede observar que poseen la dirección Ethernet de la otra máquina, con lo cual la prueba ha resultado satisfactoria.

5.4. Prueba4: Encaminamiento Avanzado

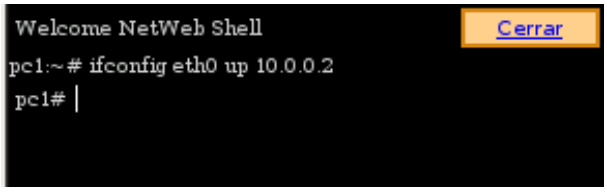
En esta última prueba vamos a comprobar la conectividad entre varias redes. Para ello diseñamos la topología de red con varios router. El resultado de esta prueba se va a intentar conseguir que pc1 se comunique con pc3, usando como encaminadores intermedios router1 y router2.

Primero diseñamos la topología de red.

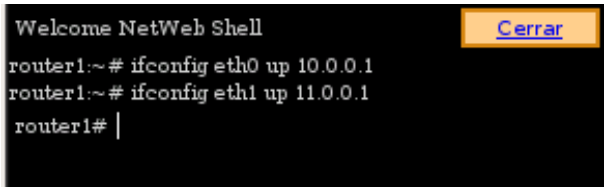


Diseño de la topología

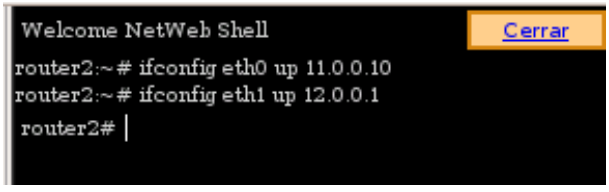
Ahora procedemos a asignar las direcciones IP según la siguientes directrices:
pc1: *eth0*: 10.0.0.2
pc2: *eth0*: 11.0.0.2
pc3: *eth0*: 12.0.0.2
router1: *eth0*: 10.0.0.1 *eth1*: 11.0.0.1
router2: *eth0*: 11.0.0.10 *eth1*: 12.0.0.1



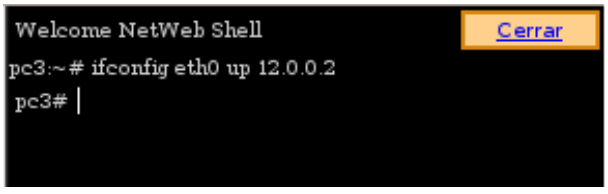
Asignación de IP a pc1



Asignación de IPs a router1



Asignación de IPs a router2



Asignación de IP a pc3

Una vez asignadas las direcciones IP a las máquinas, procederemos a realizar las tablas de encaminamiento. Todo mensaje que envíe pc1 fuera de su red local lo va a enviar a router1 para que lo encamine, igualmente router1 lo enviará a router2. En el caso de un mensaje que envíe pc3 hacia fuera de su red lo va encaminar router2 y este posteriormente se lo enviará a router1.

A continuación se muestra como se configura las tablas de encaminamiento de las distintas máquinas que van a entrar en juego en nuestra simulación.

Tabla de encaminamiento de pc1

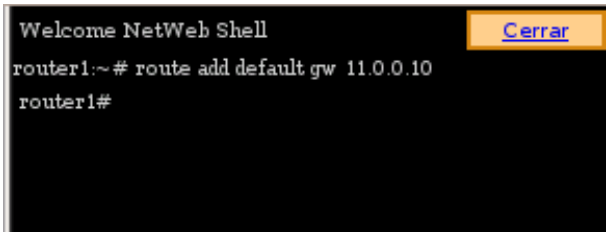


Tabla de encaminamiento router1

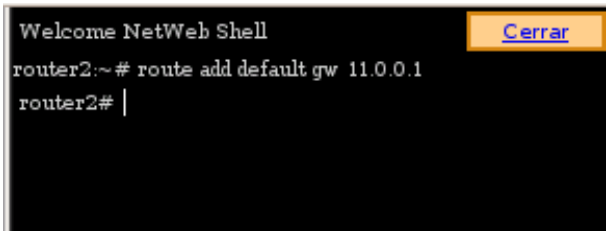


Tabla de encaminamiento router2

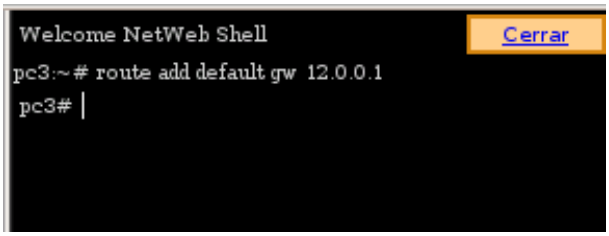
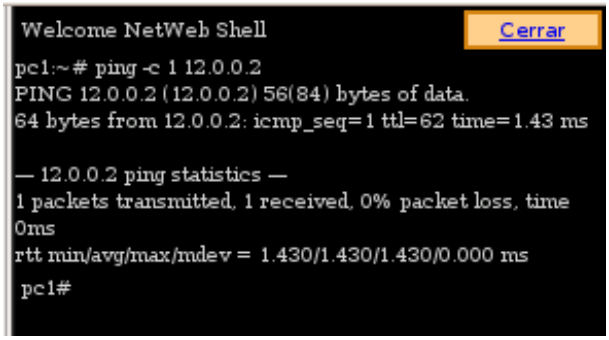
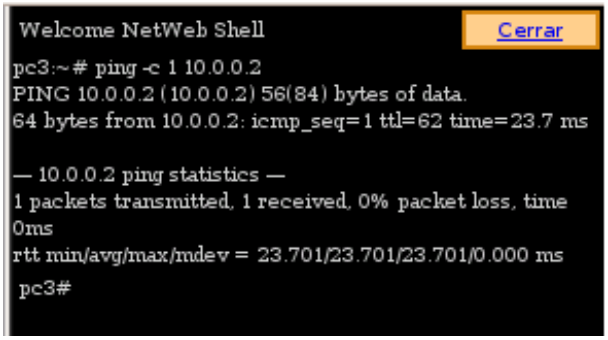


Tabla de encaminamiento pc3

Una vez configuradas las tablas de encaminamiento, vamos a comprobar mediante el comando *ping* si existe conectividad entre pc1 y pc3 y viceversa.



pc1 ->router1 ->router2 ->pc3



pc3 ->router2 ->router1 ->pc1

Comprobada la conectividad por medio de dos routers intermedios, se da por concluida la prueba con resultado satisfactorio.

Capítulo 6

Conclusiones

Una vez concluido este proyecto, hemos puesto de manifiesto que se puede entender el Web como un nueva forma de realizar interfaces de usuario.

El objetivo principal de dotar a Netkit de un interfaz de usuario web ha sido realizado con éxito. Una vez concluido el proyecto tenemos que el principal objetivo ha sido realizado con resultados satisfactorios, gracias a las pruebas realizadas en el capítulo anterior.

Para la implementación de este proyecto, sobre todo en el lado del usuario (interfaz), ha sido muy importante la comprensión de la tecnología Ajax. Gracias a Ajax, hemos podido realizar un interfaz de usuario muy similar al que se encontraría un usuario con cualquier aplicación de escritorio. Aunque cabe recordar que el interfaz de usuario es una parte de todo el sistema general que se ha desarrollado para este proyecto.

Se han cumplido también el objetivo de realizar un sistema multiplataforma, ya que cualquier usuario puede utilizar NetWeb sin tener que amoldarse a un Sistema Operativo o instalar algún software adicional.

Todos los ficheros y configuraciones realizadas durante el desarrollo del proyecto se ha realizado mediante XML. Por tanto el objetivo marcado de usar XML como estándar de configuración ha sido realizado satisfactoriamente.

Respecto a la licencia del software, NetWeb será liberado, una vez concluido este proyecto, mediante licencia GPL. Dicha licencia permitirá a cualquier usuario o programador añadir funcionalidad adicional a Netweb.

6.1. Conocimientos Adquiridos

Una vez concluido todo el desarrollo y documentación del proyecto se va a proceder a enumerar los conocimientos que se han adquirido desde el primer día que se empezó a trabajar en este proyecto.

- El principal conocimiento adquirido, ha sido la programación mediante la tecnología Ajax. Actualmente Ajax es una tecnología que esta siendo muy usada por empresas pioneras en informática y telecomunicaciones, como es Google. Ajax también esta presente en la mayoría de sitios webs importantes y hay que se este empezando a crear un modelo de negocio alrededor de esta tecnología. Gracias a estos conocimientos adquiridos se han abierto nuevos mercados laborales.

- Se han adquirido también conocimiento de gestión y diseño de un gran proyecto software,

gracias al cual hemos aprendido a capturar los requisitos necesarios para la realización, así como su análisis y posteriormente su diseño e implementación de los objetivos definidos al inicio del proyecto.

- También se han adquirido conocimientos a la hora de realizar software mediante el lenguaje de programación Python.
- Se ha mejorado el conocimiento sobre el protocolo HTTP.
- Se han adquirido conocimientos avanzados en diseño y configuración de redes bajo entornos GNU/Linux.
- La documentación es un tema muy importante en la realización de un proyecto, gracias al cual se han adquirido conocimiento de como enfocar, distribuir y redactar la memoria de un proyecto de estas características.
- Pero el principal conocimiento adquirido ha sido una maduración en términos informáticos que se me ha brindado a la hora de realizar este proyecto.

6.2. Posibles usos de NetWeb

NetWeb ha sido creada como herramienta docente para servir de apoyo a estudiantes de Ingeniería informática y/o telecomunicaciones.

En titulaciones como Ingeniería Informática e Ingeniería en Telecomunicaciones existen diferentes asignaturas en las cuales se estudian los conceptos de las arquitecturas de redes de computadores.

NetWeb, puede permitir a los alumnos el diseño de topologías de red y comprobar como funcionan algunos conceptos de las redes de computadores, por ejemplo como funciona el encaminamiento IP, como se configura un router o cualquier prueba que se pueda hacer con un conjunto de ordenadores conectados en red. Todas estas prácticas guiadas mediante NetWeb puede permitir al alumnos asentar los conocimientos teóricos vistos en clase.

Bien es sabido que a los alumnos les incomoda ir a hacer las prácticas a los laboratorios, pero gracias a NetWeb pueden tener un entorno de simulación en el que el alumno no necesitaría ninguna premisa previa para usar el sistema. Lo único que necesita es un ordenador conectado a internet y una navegador Web, con eso ya estaría dispuesto a usar NetWeb desde su propia casa y con cualquier Sistema Operativo.

6.3. Trabajos Futuros

Este proyecto ha sido el inicio de un camino hacia un sistema docente que en el futuro puede avanzar y convertirse en una herramienta de apoyo del alumno. Siguiendo esta linea se van a exponer los posible trabajos futuros que se pueden derivar de este proyecto:

- Hacer que el sistema pueda ser usado por múltiples usuarios desde sus casas. La idea que se pretende realizar en un futuro con este proyecto, es que los alumnos puedan hacer simulaciones de red desde sus propias casas.

- Permitir guardar las distintas topologías de red diseñadas. Imaginemos que hacemos una topología de red, la cual queremos guardar para poder retomarla otro día. Para ello se necesitaría guardar en algún fichero la configuración de red realizada para posteriormente ser recuperada. Para ello se podría usar XML como tecnología usada para almacenar las distintas topologías de red. Este trabajo no llevaría mucho tiempo, ya que toda la configuración existente en NetWeb esta realizada mediante XML. Únicamente habría que añadir una nueva opción de importar topologías de red desde un fichero externo.
- Permitir que distintos usuarios puedan acceder y manipular un entorno de red. La idea principal es que varias personas puedan estar trabajando sobre una misma configuración de red. Esto se podría solventar añadiendo cuentas de usuario en el lado del servidor, y que cada usuario pudiera interactuar con las simulaciones de red de otros usuarios.
- Mejorar la interfaz de usuario. Actualmente el interfaz de usuario es bastante minimalista, aunque cumple con las necesidades impuestas para la realización de este proyecto. Se podría mejorar el interfaz web de usuario añadiendo hojas de estilo CSS que ayudaría al usuario poder encontrar un sistema mucho mas amigable.

Todos estos trabajos futuros son posible gracias a que el software desarrollado posee una licencia de Software Libre GPL (GNU Public Licence).

Bibliografía

[1] BRIAN W. KERNIGHAN, ROB PIKE : *"The Practice of Programming"*, Addison-Wesley Professional, 1999.

[2] TANENBAUM, A. S.: *"Redes de computadoras"*, Prentice Hall, 1997.

[3] DIVE INTO PYTHON: *"http://diveintopython.org/"*.

[4] BRUCE W. PERRY: *"Ajax Hacks"*.

[5] BEN LAURIE, PETER LAURIE *"Apache: The Definitive Guide, Third Edition"*.

[6] MOD PYTHON *"http://modpython.org"*.

[7] SALINAS, B. C., SAORÍN, P. L., MIRA, J. M., RUIZ, A. P. Ruiz, GUILLÉN, S. S.: *"Latex una imprenta en sus manos"*, AD, 1997.

[8] KERNIGHAN, B. W., PIKE, R.: *"The Practice of Programming"*, Addison-Wesley, 1999.

[9] WORLD WIDE WEB CONSORTIUM: *"http://www.w3.org"*.

[10] .GRUPO DE INVESTIGACIÓN MOBIQUO: *"http://mobiquo.dat.escet.urjc.es"*

[11] .GRUPO DE SISTEMAS Y COMUNICACIONES (GSyC) DE LA URJC: *"http://gsyc.escet.urjc.es"*