UNIVERSIDAD DE ALCALÁ DE HENARES

Escuela Politécnica

ESTUDIOS COMPLEMENTARIOS A LA TITULACIÓN EN INGENIERÍA EN SISTEMAS DE INFORMACIÓN DE LA UNAN-LEÓN

PROYECTO DE FIN DE CARRERA

PLAN DOCENTE PARA LA ASIGNATURA DE APLICACIONES TELEMÁTICAS

DENIS LEOPOLDO ESPINOZA HERNÁNDEZ

ALCALÁ DE HENARES, ESPAÑA

JULIO 2007

UNIVERSIDAD DE ALCALÁ DE HENARES

Escuela Politécnica

ESTUDIOS COMPLEMENTARIOS A LA TITULACIÓN EN INGENIERÍA EN SISTEMAS DE INFORMACIÓN DE LA UNAN-LEÓN

PROYECTO DE FIN DE CARRERA

PLAN DOCENTE PARA LA ASIGNATURA DE APLICACIONES TELEMÁTICAS

Autor: Denis Leopoldo Espinoza Hernández
Directores: Manuel Moreno Martín
Antonio Garcia Herráiz

TRIBUNAL

Presidente: Daniel Meziat Luna

Vocal 1: Francisco Javier Ceballos Sierra

Vocal 2: Elena Campo Montalvo

Calificación: _____

Fecha: ……. de julio de 2007

A DIOS, por haber sido mi fuerza y mi canción durante mi estancia en España,

A la Virgen María, por haberme cuidado y consolado cuando me sentí triste,

A mi familia y mi novia, por haberme apoyado y animado a seguir siempre adelante,

A mis amigos, que me esperan en mi pais y a los que ahora dejo aquí en España.

INDICE

I. INTRODUCCIÓN

En el año 1994 se creó el Proyecto de Colaboración con Nicaragua, un convenio entre el Departamento de Automática de la Universidad de Alcalá de Henares (UAH) y el Departamento de Computación de la Universidad Autónoma de Nicaragua (UNAN-León) con el objetivo de apoyar la carrera de Licenciatura en Computación ahora Ingeniería en Sistemas de Información ofertada en la UNAN-León. El convenio está dirigido a impulsar la excelencia en la enseñanza que permitirá formar mejores profesionales en el área de la informática.

En función del inobjetable impulso que han tenido las redes y comunicaciones en general y como parte de la continuidad del programa de colaboración que esta Universidad tiene con la Universidad de Alcalá, el departamento de Computación de la UNAN –León decidió ampliar la oferta a los estudiantes de primer ingreso con una nueva carrera: Ingeniería en Telemática la cual tiene los 3 primeros años comunes con la carrera de Ingeniería en Sistemas.

En el presente documento se aborda el desarrollo del Plan de Docente de la asignatura Aplicaciones Telemáticas que se imparte en el cuarto año de la carrera de Ingeniería en Telemática de la UNAN-León. Además se presenta la situación actual de dicha asignatura, su relación con otras asignaturas, la metodología y material didáctico de apoyo para impartirla, el sistema de evaluación, la planificación temporal, la presentación del contenido teórico, el desarrollo de las prácticas y la bibliografía necesaria para las mismas.

El Plan docente está compuesto por dos partes. En la primera parte se desarrollan los contenidos teóricos que son necesarios para cumplir los objetivos de la asignatura y son los fundamentos básicos para la comprensión de las prácticas. En la segunda parte se desarrollan las prácticas de laboratorio que permitirán a los estudiantes reforzar y aplicar los conocimientos adquiridos en las clases teóricas.

Se buscado que cada una de las partes que se abordan en la parte teórica tenga su correspondencia directa en la práctica para que el alumno al finalizar esta asignatura, este capacitado para instalar y configurar diferentes servicios telemáticas.

II. OBJETIVOS

Los objetivos del presente proyecto docente son:

- Proporcionar a los estudiantes los conceptos necesarios para comprender los temas de Aplicaciones Clásicas, Servicios de Directorios, World Wide Web, Correo Electrónico y Voz sobre IP de la asignatura Aplicaciones Telemáticas.

- Contar con un documento guía para impartir la asignatura de Aplicaciones Telemáticas que permitir una mejor planificación y organización de la asignatura.

- Desarrollar prácticas de laboratorio que permitan afianzar los conocimientos adquiridos por los estudiantes en la parte teórica de la asignatura.

III. SITUACIÓN ACTUAL DE LA ASIGNATURA

La asignatura de Aplicaciones Telemáticas, se ha de impartir en la carrera de Ingeniería en Telemática de la UNAN-León, que ofrece el departamento de computación de la Facultad de Ciencias de dicha Universidad.

La carrera se desarrolla a lo largo de 10 semestres de estudios (5 años académicos/lectivos). Esta asignatura se imparte en el VIII semestre del plan de estudios actual (II semestre del IV año de la carrera).

La asignatura consta con un total de 64 horas las cuales incluyen tanto tiempo de teoría como de laboratorio distribuidas a los largo de 16 semanas lectivas. A continuación se presenta la división de las horas entre la teoría y la práctica:

	Horas Semanales	Horas Semestrales
Teoría	2	32
Práctica	2	32
Un total de 64 horas al Semestre		

El estudiante deberá tener ciertos conocimientos previos para lograr un total aprendizaje de la asignatura. En la sección siguiente se citan cuales son las asignaturas que de modo directo se relacionan con ésta, sin embargo, como mínimo, todo estudiante debe tener claros los siguientes conceptos:

En la teoría:

- La capa de red en Internet.
- La capa de transporte.

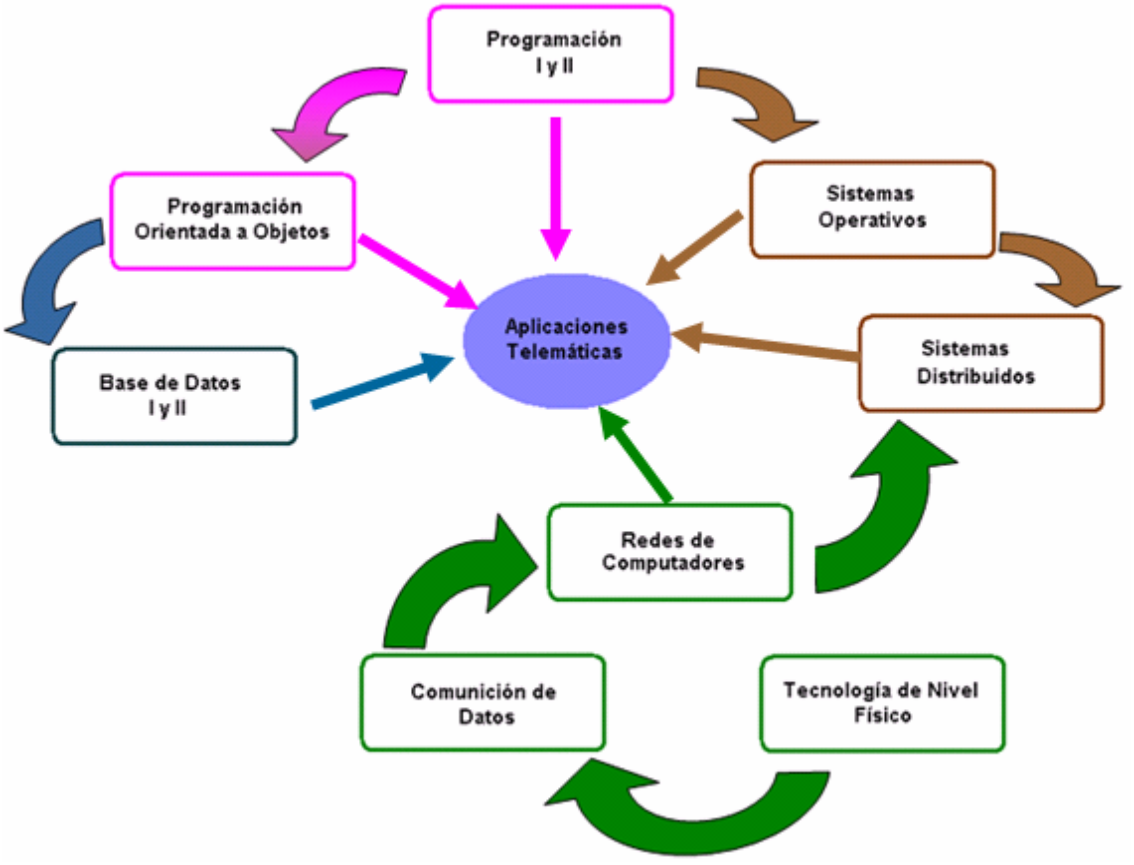
En la práctica:

- Administración del sistema operativo Linux.
- Programación con socket en Linux.

IV. RELACIÓN CON OTRAS ASIGNATURAS

El proceso de aprendizaje de la asignatura Aplicaciones Telemáticas, requiere de otros conocimientos que al llegar a esta asignatura deben estar claros y afianzados para una mejor comprensión de la misma. Estos conocimientos son adquiridos por los estudiantes a lo largo de su formación académica en otras asignaturas.

Por eso se hace importante plasmar la relación que tiene la asignatura de Aplicaciones de Telemática con otras asignaturas, relación que se muestra primero de manera gráfica y que se detalla más adelante.



En la figura anterior se ha dividido en 4 grupos las asignaturas con las cuales se relaciona la asignatura de Aplicaciones Telemáticas. Cada grupos ha sido representado por un color diferente.

Programación:

- **Programación I y II:** Estas asignaturas se imparten durante el tercer y cuarto semestre de la carrera respectivamente y darán las bases de programación sobre las cuales se sientan mucha de la teoría y práctica de diversas asignaturas.

- **Programación Orientada a Objetos:** Los conceptos de la programación orientada a objetos ayudaran a introducir la filosofía de trabajar con objetos que será de gran ayuda en ciertos capítulos de la asignatura como LDAP en el cual se introducen los conceptos de esquemas y objetos que tienen gran similitud con las bibliotecas y clases en C++.

Bases de Datos:

- **Base de Datos I y II:** Se tienen dos cursos de bases de datos durante los semestres cuarto y quinto respectivamente. Dichos conocimientos serán de gran utilidad por dos razones básicas, la primera porque en la mayoría de las prácticas manejaremos los usuarios a través de diferentes bases de datos; y la segunda porque permitirá establecer la diferencia entre las bases de datos y los servicios de directorios.

Sistemas Operativos:

- **Sistemas Operativos:** Se imparte en el séptimo semestre de la carrera. Permitirá conocer la manera de trabajar de los sistemas operativos y sobre todo, aclarar los conceptos de procesos e hilo, elementos muy utilizados dentro de las aplicaciones que se abordarán a los largo de la asignatura.

- **Sistemas Distribuidos:** Se imparte durante el octavo semestre de la carrera Ingeniería Telemática. Prácticamente todas las aplicaciones que se abordarán a lo largo de la asignatura son aplicaciones distribuidas y será esta asignatura la que explique más detalle como se programan dichas aplicaciones.

Redes de Ordenadores:

- **Tecnologías de Nivel Físico:** Se imparte en el séptimo semestre de la carrera. Proporcionará los conocimientos iniciales de las redes de computadores, centrándose en: conceptos básicos y el nivel físico.

- **Comunicación de Datos:** Se imparte en el octavo semestre de la carrera. Continuará el estudio de las redes de computadores estudiando el nivel de enlace e introduciendo conceptos básicos para el estudio del nivel de Red.

- **Redes de Computadores:** Se imparte en el octavo semestre de la carrera. Proporcionará a los estudiantes, los conocimientos sobre las estructuras y mecanismos que dan soporte a la comunicación entre subredes que utilizan la arquitectura de protocolos TCP/IP.

Gracias a todas las asignaturas anteriores con sus correspondientes prácticas de laboratorio, se podrá lograr que el estuante comprenda claramente la asignatura tanto en la parte teórica como las prácticas de laboratorio.

V. CONTENIDO DEL TEMARIO

TEMA 0: PRESENTACIÓN DE LA ASIGNATURA.

- Objetivos de la asignatura.
- Temario.
- Material de estudio.
- Metodología de evaluación.
- Tutorías.

TEMA 1: APLICACIONES CLÁSICAS.

- La Capa de Aplicación.
- Terminal Virtual.
- Sistemas de Ficheros en Red (NFS).
- Protocolo de Transferencia de Ficheros (FTP).

TEMA 2: SERVICIOS DE DIRECTORIOS.

- Introducción a los Servicios de Directorios.
- El estándar X.500 de ISO/ITU.
- Protocolo Ligero de Directorio Activo (LDAP).
- Servicio de Nombres de Dominio (DNS).

TEMA 3: WORLD WIDE WEB.

- Arquitectura de la Web.
- Documentos Web Estáticos.
- Documentos Web Dinámicos.
- Protocolo de transferencia de Hipertexto.
- Mejoras en el Desempeño.

TEMA 4: CORREO ELECTRÓNICO.

- El correo electrónico en Internet.
- Protocolo Simple de Transferencia de Correo (SMTP).
- Estructura y codificación de mensajes MIME.
- Protocolos de Entrega Final (POP3 e IMAPv4).

TEMA 5: VOZ SOBRE IP.

- Características de voz sobre IP.
- Transporte de Información en Telefonía IP.
- Señalización de control de llamadas.

VI. MÉTODOLOGÍA DIDACTICA Y MATERIAL DIDÁCTICO A UTILIZAR

Metodología Didáctica

- **En la parte Teórica:**

La metodología a utilizar para impartir esta asignatura serán las lecciones magistrales con una duración de dos horas, 1 vez a la semana durante las cuales se abordará de forma planificada el contenido teórico desarrollado en este plan. Cada sesión de dos horas se planificará en 2 bloques de 50 minutos cada uno, y si es posible en un futuro cambiar de 1 sesión de dos horas a la semana a dos sesiones de una hora a la semana.

Las lecciones serán complementadas con ejemplos que ayuden a una mejor comprensión de los conceptos expuestos. Además se propondrá la realización de trabajos investigativos tanto en la parte teórica como en la práctica para profundizar en los temas y despertar de esa manera la capacidad de búsqueda y el deseo de conocimiento en los alumnos.

- **En la práctica:**

En las prácticas de laboratorio se realizará una explicación de la práctica al iniciar cada una de ellas y puesto que las prácticas están guiadas, el profesor sólo aclarará dudas o resolverá algún otro problema. Se propondrán extensiones a las prácticas con el objetivo de incentivar al alumno a profundizar en cada tema.

Material Didáctico:

Tanto para la parte práctica como para la parte teórica se utilizará el mismo material didáctico que será:

- La pizarra para las explicaciones y ejemplos en que sea necesario.
- Láminas de acetato y retro-proyector.

- Una página Web de la asignatura en la cual se colocará toda la información, transparencias para la teoría, enunciados de las prácticas, apuntes, material de apoyo y enlaces a sitios de interés.
- Acceso a la bibliografía básica para este tema y a la información impresa del material de la Web la cual estará disponible en la biblioteca del departamento.

VII. PLANIFICACIÓN TEMPORAL

La planificación de la asignatura, para que sea de una manera objetiva, debe de realizarce año con año en función del calendario académico del que se disponga para poder, a partir de este, calcular el número de sesiones lectivas de las cuales se dispone.

La asignatura de Aplicaciones Telemáticas se imparte en el segundo semestre de IV año. Normalmente el segundo semestre consta con un total de 16 semanas hábiles, pero considerando los días festivos y dando margen para cualquier otra incidencia como realización de exámenes, etc., podemos deducir que se tienen 14 semanas netas para cumplir con el contenido de la asignatura.

Tanto para la teoría como para la práctica, constan de una sesión de 2 horas a la semana, por 14 semanas que hemos asumido como netas, suman un total 28 horas para la teoría y 28 horas para las prácticas.

El número de horas asignadas a cada tema y a cada práctica, se ha calculado en base a la profundidad con que se quiere abordar cada tema. De tal manera que la planificación temporal tanto para la teoría como para las prácticas es la siguiente:

Planificación temporal de la parte teórica:

Nº de Tema	Tema	Horas
0	Presentación de la Asignatura	1
1	Aplicaciones Clásicas	4
2	Servicios de Directorios	6
3	World Wide Web	8
4	Correo Electrónico	5
5	Voz sobre IP	4
Total de horas de Teoría		28

Planificación temporal de las prácticas:

Nº de Práctica	Práctica	Horas
0	Administración de Servicios en Linux.	2
1	Acceso Remoto con SSH.	2
2	Configuración de un Servidor NFS.	2
3	Configuración de un Servidor FTP.	2
4	Configuración de un Servicio de Directorios con LDAP.	4
5	Configuración de un Servidor DNS.	4
6	Configuración de LAMP.	2
7	Ejercicios de Páginas Web Estáticas y Dinámicas.	2
8	Configuración de un Servidor de Correo.	4
9	Configuración de una Centralita de voz sobre IP.	4
Total de horas de Laboratorio		28

VIII. METODOLOGÍA DE EVALUACIÓN

La asignatura está compuesta por las clases teóricas y las prácticas que son realizadas en el laboratorio de la misma. A lo largo del semestre se realizarán dos evaluaciones parciales que corresponderán al 60% de la nota final y el restante 40% se obtendrá en una evaluación al final del semestre. Cada una de las evaluaciones parciales se desglosa de la manera siguiente:

- **Examen Teórico**…….. 80%
- **Evaluación Práctica**……20%
- **Total**……100%

Las notas del primero y segundo parcial se promedian y se multiplican por 0.6 (ya que equivalen al 60% de la nota final) para obtener la nota de entrada al examen final. El examen final tiene un valor de 100 puntos que luego se multiplica por 0.4 (para obtener el 40% al que equivale) y el resultado se suma con la nota de entrada. Para aprobar la asignatura el alumno debe de obtener una nota mayor o igual que el 60. Lo anterior se refleja a continuación:

(1º_Parcial + 2º_Parcial)*0.6 + Examen_Semestral*0.4 >= 60

Evaluación de la Teoría:

El examen teórico que corresponde al 80% de cada parcial constará de preguntas que permitan evaluar el grado de compresión de los conceptos básicos que tiene el estudiante. También contendrá preguntas de carácter analítico y ejercicios en los cuales el alumno deberá de demostrar la capacidad de aplicar los conocimientos adquiridos.

Evaluación del Laboratorio

El 20% correspondiente a las evaluaciones del laboratorio se obtendrá a través de la entrega en tiempo y forma de las prácticas de laboratorio. El valor de cada práctica se distribuirá en función del número y la complejidad de las mismas.

IX. DESARROLLO DEL TEMARIO

TEMA 0: PRESENTACIÓN DE LA ASIGNATURA

Objetivo:

- Presentar al estudiante los principales aspectos relacionados con el desarrollo de la asignatura.

Contenido

- Se plantearán los objetivos que se pretenden alcanzar con la asignatura.
- Se hará una descripción general del contenido que ha de desarrollar en la asignatura.
- Se repasarán conceptos básicos del nivel de Red en Internet.
- Se recordarán las principales características de los protocolos de transporte TCP y UDP.

Material de Estudio

- Se informará al alumno del material que se utilizará de guía para el desarrollo de la asignatura (transparencias).
- Se listará la bibliografía básica para el desarrollo de la asignatura.
- Se presentará la página Web de la asignatura en la cual se colocará toda la información necesaria tanto para la teoría como para la práctica.

Metodología de Evaluación

- Se explicará al alumno la metodología de evaluación empleada durante el desarrollo de la asignatura.

Tutorías

- Todos los estudiantes podrán hacer consultas al profesor fuera del aula de clases en un determinado horario, por lo tanto se debe brindar el horario para estas tutorías.

TEMA 1: APLICACIONES CLÁSICAS

Objetivos

- Describir las principales características, principios y requerimientos de los protocolos de la capa de aplicación en Internet.
- Estudiar el funcionamiento básico del protocolo de transferencia de ficheros.
- Analizar la forma en que el protocolo SSH garantiza la seguridad en el acceso a equipos remotos.
- Resaltar la transparencia ofrecida por los Sistemas de Ficheros en Red de cara al usuario final.

Contenido

- La Capa de Aplicación.
- Terminal Virtual.
- Sistemas de ficheros en Red (NFS).
- Protocolo de Transferencia de Ficheros (FTP).

Bibliografía

- James F. Kurose, Keith W. Ross, Redes de Computadores: Un enfoque descendente basado en Internet, Segunda Edición. (Capitulo 2).
- http://es.wikipedia.org/wiki/File_Transfer_Protocol
- RFC 959: Referencia que define las características actuales, a fecha de Febrero del 2007, del protocolo FTP.
- http://es.wikipedia.org/wiki/Secure_Shell
- http://es.wikipedia.org/wiki/Network_File_System

1. La Capa de Aplicación

1.1 Introducción

Las aplicaciones son la razón de ser de las redes de computadores. A lo largo de los últimos 30 años se han diseñado numerosas aplicaciones de red, ingeniosas y extraordinarias. Estas aplicaciones van desde los correos de solo texto, hasta las aplicaciones multimedia de hoy en día.

En este tema, abordaremos los aspectos fundamentales de las diferentes aplicaciones de red.

1.2 Principios de los protocolos de la capa de Aplicación

A pesar de que las aplicaciones de red son diversas, el software casi siempre forma un papel central. Recuerde que el software que en este tipo de aplicaciones, el software siempre es distribuido (un emisor y un receptor).

Este software son procesos que pueden estar en la misma máquina o en máquinas separadas y que se comunican con mensajes a través de la red.

Figura 1.1 Comunicación a través de la red.

1.3 Protocolo de la capa de Aplicación

Es importante distinguir entre las aplicaciones de red y los protocolos de la capa de aplicación. Un protocolo de la capa de aplicación es únicamente una parte (aunque importante) de la aplicación de red. Como ejemplo tenemos que la aplicación Web, utiliza el protocolo de la capa de aplicación HTTP (Protocolo de Transferencia de Hipertexto) y el correo electrónico utiliza principalmente protocolo (SMTP).

Como se puede observar, a veces el protocolo tiene un nombre diferente de la aplicación, sin embargo existen otros casos en que se emplea el nombre del protocolo para referirse a la aplicación, ejemplo: FTP, DNS.

En particular, un protocolo de la capa de aplicación define:

- Tipo de mensajes intercambiados.
- Sintaxis de los distintos mensajes.
- La semántica de los campos.
- Las reglas que determinan cuándo y cómo un proceso envía mensajes y responde a los mensajes.

La mayoría de los protocolos de la capa de aplicación están disponibles en las RFC son dominio público a diferencia de otros que son propietarios como es el caso de algunos de telefonía (Skype).

1.4 Comunicación a través de la Red

La comunicación entre dos aplicaciones situadas en máquinas remotas se realiza a través de sockets. Un socket puede ser visto como la puerta de un proceso. Cuando un proceso desea enviar información a otro, la empuja a través de su puerta. Este proceso asume que debajo hay una infraestructura de transporte que llevará el mensaje a través de Internet hasta l puerta (socket) del proceso destino.

De forma más técnica, un socket es una interfaz entre la capa de aplicación y la capa de transporte del host, por lo que también se le denominan API (interfaz de programación de aplicaciones) sobre la que se construyen en Internet las aplicaciones.

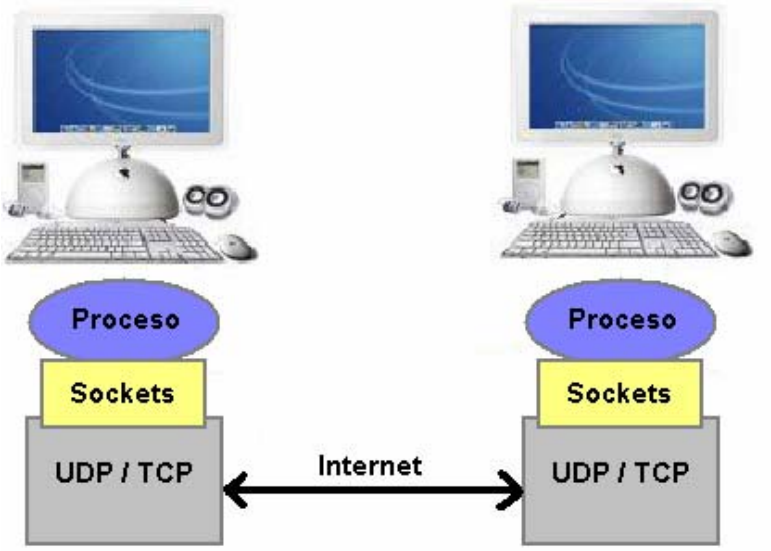


Figura 1.2 Comunicación de procesos a través de sockets.

1.4.1 Elección del Protocolo de Transporte a utilizar

El desarrollador de aplicaciones tiene completo sobre la capa de aplicación aunque el control es pequeño sobre la capa de transporte. El único control que tiene es decidir que protocolo de la capa de transporte desea utilizar y algunos parámetros de calidad de servicio.

Los protocolos entre los cuales el programador puede seleccionar de la capa de transporte son TCP y UDP, cada uno de los cuales ofrece un servicio diferente.

TCP (Protocolo de Control de Transmisión)

El protocolo de transporte TCP identifica a lo que normalmente se conoce como circuitos virtuales y posee las siguientes características:

- Servicio Orientado a Conexión, porque se da un proceso de establecimiento a tres bandas (handshaking) antes de la transferencia de datos.
- La conexión es full-duplex.
- Servicio fiable de transporte, ya que TCP garantiza que no habrá pérdida ni duplicado de bytes.
- Incluye mecanismos de control de congestión.
- No es óptimo para aplicaciones en tiempo real, ya que estas toleran perdidas y no necesita ser tan fiables.

- TCP garantiza el envío de todos los datos, pero no garantiza la tasa de fallos de envío ni los de retardos experimentados.

UDP (Protocolo de Datagrama de Usuario)

El protocolo de transporte UDP equivale a los datagramas:

- Servicio No Orientado a Conexión, porque no hay acuerdo antes que los procesos comiencen a comunicarse.
- Servicio no fiable de transporte, pues no garantiza que los mensajes lleguen, ni que no lleguen desordenados o duplicados.
- No incluye mecanismos de control de congestión.
- Es mejor para las aplicaciones en tiempo real aunque no hay garantías de retardo.

Aplicaciones	Protocolo de la capa de Aplicación	Protocolo de la capa de Transporte
Correo electrónico	SMTP [RF[]C 2821]	TCP
Acceso a terminales remotos	Telnet [RFC 854]	TCP
Web	http [RFC 2616]	TCP
Transferencia de archivos	FTP [RFC 959]	TCP
Servidor de archivos remotos	NFS [McKusik 1996]	UDP o TCP
Flujo de multimedia	Propietario normalmente (ejemplo, Real Network)	UDP o TCP
Telefonía Internet	Propietario normalmente (ejemplo, Dialpad)	Típicamente UDP

Figura 1.3 Aplicaciones con sus protocolos de la capa de aplicación y transporte.

1.4.2 Direccionamiento de Procesos

Para que un proceso pueda enviar un mensaje a otro proceso situado en una máquina remota, se necesitan dos cosas (1) el nombre o dirección de la máquina destino y (2) el identificador del proceso dentro la máquina.

Para lo primero se utiliza la dirección IP de la máquina la cual se obtiene ya sea de forma directa por que el proceso la conoce o a través de una consulta al DNS. El segundo elemento se obtiene gracias al número de puerto (TCP o UDP) que la aplicación utiliza dentro de la máquina en la que reside.

Existen ciertos puertos que se encuentran asignados a los protocolos más conocidos de la capa de aplicación. Por ejemplo, un servidor Web ocupa el puerto 80.

1.5 Agentes de Usuario

El agente de usuario es la Interfaz entre el usuario y la aplicación de red. Por ejemplo en el caso de HTTP, el agente de usuario es el navegador ya que nos abstrae de tener que escribir nosotros la petición de la página que deseamos visualizar.

1.6 Servicios que necesita una Aplicación

En general, toda aplicación fija sus requisitos basándose en tres elementos fundamentales que son:

1.6.1 Transferencia fiable de datos

Implica si una transmisión es tolerable a fallos o no. Por ejemplo, no se tolera que hayan fallos cuando se esta realizando una transferencia bancaria porque podría perderse dinero, sin embargo, si se toleran fallos en el caso de video o de audio.

1.6.2 Ancho de banda

Algunas aplicaciones para ser efectivas deben ser capaces de transferir datos a ciertas velocidades. Las aplicaciones de telefonía por Internet necesitan transmitir a la misma velocidad que se codifica la vos (32 kbps), mientras que el correo electrónico, la transferencia de archivos y la transferencia Web son aplicaciones flexibles.

1.6.3 Temporización

Ciertas aplicaciones requieren de restricciones de tiempo para el envío de datos como son las aplicaciones en tiempo real. Para as aplicaciones que no son en

tiempo real, los retardos menores son siempre preferibles aunque no existen restricciones estrictas.

Aplicación	Perdida de datos	Ancho de banda	Sensible al tiempo
Transferencia de archivos	No pérdida	Flexible	No
Correo electrónico	No pérdida	Flexible	No
Documentos Web	No pérdida	Flexible (pocos kbps)	No
Audio/Video en tiempo real	Tolerante	Audio: pocos kbps – 1 Mbps Video: 10 Kbps – 5 Mbps	Si: 100 mseg
Audio/Video almacenado	Tolerante	Igual que el anterior	Si: pocos seg
Juegos interactivos	Tolerante	Pocos kbps-10kbps	Sí: 100 mseg
Mensajería instantánea	No pérdida	Flexible	Sí y No

Figura 1.4 Requisitos de ciertas aplicaciones de red.

2. Terminal Virtual

2.1 Introducción

La Terminal virtual posibilita el acceso remoto a un host en modo Terminal, permitiendo:

- Ejecución de aplicaciones de manera remota.
- Acceder a otros servicios de Internet.
- Control remoto de servidores.
- Arreglos de fallos a distancia.
- Consultar datos a distancia.

Se requiere cuenta de usuario en el host destino para la autenticación. Soporta la interacción entre sistemas heterogéneos, es decir, con diferente juego de caracteres o sistemas más operativos y hardware.

2.2 Telnet

Aplicación clásica de Internet del tipo terminal virtual, tiene el mismo nombre que el protocolo que la soporta. Se ejecuta en el puerto 23. Utiliza señalización en banda en ambas direcciones, tanto los comandos como los datos emplean la misma conexión TCP.

2.2.1 Servicios básicos de Telnet

Terminal virtual de red, proporciona un interfaz estándar NVT (Network Virtual Terminal). Permite la negociación de opciones entre cliente y servidor. Además realiza la transferencia de información en claro, en modo texto no cifrada: datos, nombre de usuario y clave (password).

2.2.2 Terminal Virtual de Red

Se encuentra definida en la RFC 854, es una pseudoterminal (dispositivo imaginario, lógica que equivale a un dispositivo de teclado y pantalla). Tanto el cliente

como el servidor traduce su juego de caracteres al juego de caracteres NVT. Esto posibilita absorber las diferencias entre sistemas heterogéneos. Define como los datos y los comandos se envían a través de la red. Exonera al sistema local de tener que conocer los detalles del sistema remoto.

2.2.3 Operaciones Telnet

Para iniciar una sesión Telnet, el usuario debe teclear:

telnet <nombre_host_remoto> ó

telnet <direccion_IP_host_remoto>

A continuación, el sistema remoto solicita “nombre de usuario” y “password”, y se inicia la sesión en modo texto si se responde adecuadamente. Por defecto, el cliente hace eco a lo que el usuario teclea, lo que se conoce como eco local y genera meno tráfico en la red. Es posible modificarlo para que el servidor conteste lo que se teclea, esto genera menos rendimiento, más tráfico pero permite monitorizar la transferencia.

2.3 Shell Segura (SSH)

SSH (Secure SHell) es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red. Permite manejar por completo el ordenador mediante un intérprete de comandos, y también puede redirigir el tráfico de X para poder ejecutar programas gráficos si tenemos un Servidor X arrancado.

Además de la conexión a otras máquinas, SSH nos permite copiar datos de forma segura (tanto ficheros sueltos como simular sesiones FTP cifradas), gestionar claves RSA para no escribir claves al conectar a las máquinas y pasar los datos de cualquier otra aplicación por un canal seguro tunelizado mediante SSH.

2.4 Seguridad

SSH trabaja de forma similar a como se hace con telnet. La diferencia principal es que SSH usa técnicas de cifrado que hacen que la información viaje por el medio

de manera no legible y ninguna tercera persona pueda descubrir el usuario y contraseña de la conexión ni lo que se escribe durante toda la sesión; aunque es posible atacar este tipo de sistemas por medio de ataques de REPLAY.

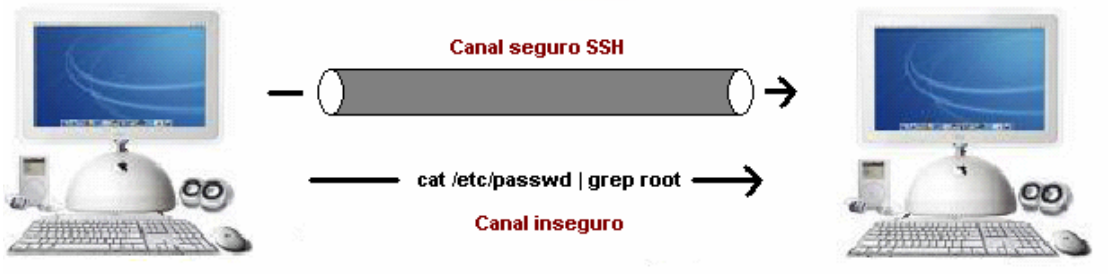


Figura 1.5 Tunel seguro con SSH

2.5 Historia

Al principio sólo existían los r-commands, que eran los basados en el programa rlogin, el cual funciona de una forma similar a telnet. La primera versión del protocolo y el programa eran libres y los creó un finlandés llamado Tatu Ylönen.En el año 1997 (dos años después de que se creara la primera versión) se propuso como borrador en la IETF.

A principios de 1999 se empezó a escribir una versión que se convertiría en la implementación libre por excelencia, la de OpenBSD, llamada OpenSSH.

2.6 Componentes de SSH

Tal como se describe en uno de los borradores de la especificación temporal "SSH Protocol Architecture", SSH es un protocolo para iniciar sesiones en máquinas remotas, que ofrece autenticación, confidencialidad e integridad. Consta de tres componentes:

- **Protocolo de transporte:** Que normalmente opera sobre TCP/IP dando autenticidad, confidencialidad e integridad.

- **Protocolo de autenticación de usuario:** Que autentica al usuario ante el servidor.

- **Protocolo de conexión:** Que multiplexa un canal cifrado en diversos canales lógicos.

3. Sistemas de Ficheros en Red (NFS)

3.1 Introducción

El Network File System (Sistema de archivos de red), o NFS, es un sistema de archivos distribuido para un entorno de red de área local. Posibilita que distintos sistemas conectados a una misma red accedan a ficheros remotos como si se tratara de locales. Originalmente desarrollado por Sun Microsystems en 1984.

El protocolo de NFS está diseñado para ser independiente de la máquina, el sistema operativo y el protocolo de transporte. Esto es posible porque se implementa sobre ONC RPC.

3.2 Características

Las principales características de NFS son:

- El sistema NFS está dividido al menos en dos partes principales: un servidor y uno o más clientes. Los clientes acceden de forma remota a los datos que se encuentran almacenados en el servidor.

- Las estaciones de trabajo locales utilizan menos espacio de disco debido a que los datos se encuentran centralizados en un único lugar pero pueden ser accedidos y modificados por varios usuarios, de tal forma que no es necesario replicar la información.

- Los usuarios no necesitan disponer de un directorio “home” en cada una de las máquinas de la organización. Los directorios “home” pueden crearse en el servidor de NFS para posteriormente poder acceder a ellos desde cualquier máquina a través de la infraestructura de red.

- También se pueden compartir a través de la red dispositivos de almacenamiento como disqueteras, CD-ROM y unidades ZIP. Esto puede reducir la inversión en dichos dispositivos y mejorar el aprovechamiento del hardware existente en la organización.

- Todas las operaciones sobre ficheros son síncronas. Esto significa que la operación sólo retorna cuando el servidor ha completado todo el trabajo asociado para esa operación. En caso de una solicitud de escritura, el servidor escribirá físicamente los datos en el disco, y si es necesario, actualizará la estructura de directorios, antes de devolver una respuesta al cliente. Esto garantiza la integridad de los ficheros.

3.3 Operaciones

El programa NFS soporta 18 procedimientos para todas las operaciones básicas de E/S, como por ejemplo:

- **LOOKUP:** Busca un fichero en el directorio actual y si lo encuentra, devuelve un descriptor a ese fichero más información sobre los atributos del fichero.
- **READ y WRITE:** Primitivas básicas para acceder el fichero.
- **RENAME:** Renombra un fichero.
- **REMOVE:** Borra un fichero.
- **MKDIR y RMDIR:** Creación/borrado de subdirectorios.
- **GET y SET-ATTR:** Devuelve conjuntos de atributos de ficheros.

Se corresponden con la mayoría de primitivas de E/S usadas en el sistema operativo local para acceder a ficheros locales. De hecho, una vez que se ha montado el directorio remoto, el sistema operativo local tiene que "reencaminar" las primitivas de E/S al host remoto. Esto hace que todas las operaciones de E/S sobre ficheros tengan el mismo aspecto, independientemente de si el fichero es local o remoto. El usuario puede trabajar con los comandos y programas habituales en ambos tipos de ficheros; en otras palabras, el protocolo NFS es completamente transparente al usuario.

3.4 Versiones

Hay tres versiones de NFS actualmente en uso.

- La versión 2 de NFS (NFSv2), es la más antigua y está ampliamente soportada por muchos sistemas operativos.

- La versión 3 de NFS (NFSv3) tiene más características, incluyendo manejo de archivos de tamaño variable y mejores facilidades de informes de errores, pero no es completamente compatible con los clientes NFSv2.

- NFS versión 4 (NFSv4) incluye seguridad Kerberos, trabaja con cortafuegos, permite ACLs y utiliza operaciones con descripción del estado.

4. Protocolo de Transferencia de Ficheros (FTP)

4.1 Introducción

FTP es un protocolo de transferencia de ficheros entre sistemas conectados basado en la arquitectura cliente-servidor, de manera que desde un equipo cliente nos podemos conectar a un servidor para descargar o enviar ficheros desde o hacia él.

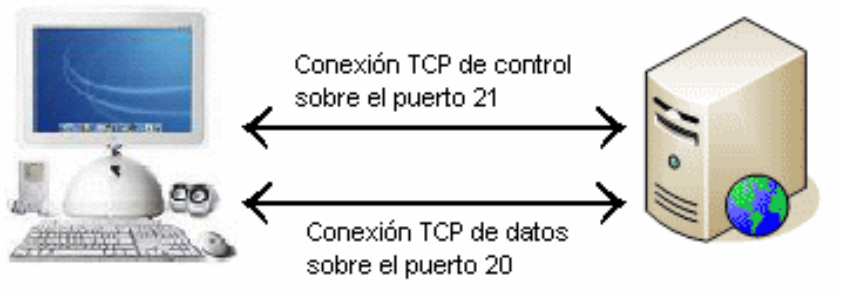


Figura 1.6 Conexiones de control y de datos FTP.

El Servicio FTP es ofrecido por la capa de aplicación del modelo de capas de red TCP/IP. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier fichero, se realiza en texto plano sin ningún tipo de cifrado.

Para solucionar este problema son de gran utilidad aplicaciones como SCP y SFTP, incluidas en el paquete SSH, que permiten transferir ficheros pero cifrando todo el tráfico.

4.2 El Modelo FTP

En el modelo, el intérprete de protocolo (PI) de usuario, inicia la conexión de control en el puerto 21, que obedece el protocolo Telnet. Las órdenes FTP estándar las genera el PI de usuario y se transmiten al proceso servidor a través de la conexión de control. Las respuestas estándar se envían desde el PI del servidor al PI de usuario por la conexión de control como contestación a las órdenes.

Estas órdenes FTP especifican parámetros para la conexión de datos (puerto de datos, modo de transferencia, tipo de representación y estructura) y la naturaleza

de la operación sobre el sistema de ficheros (almacenar, recuperar, añadir, borrar, etc.). El siguiente modelo representa el diagrama de un servicio FTP.

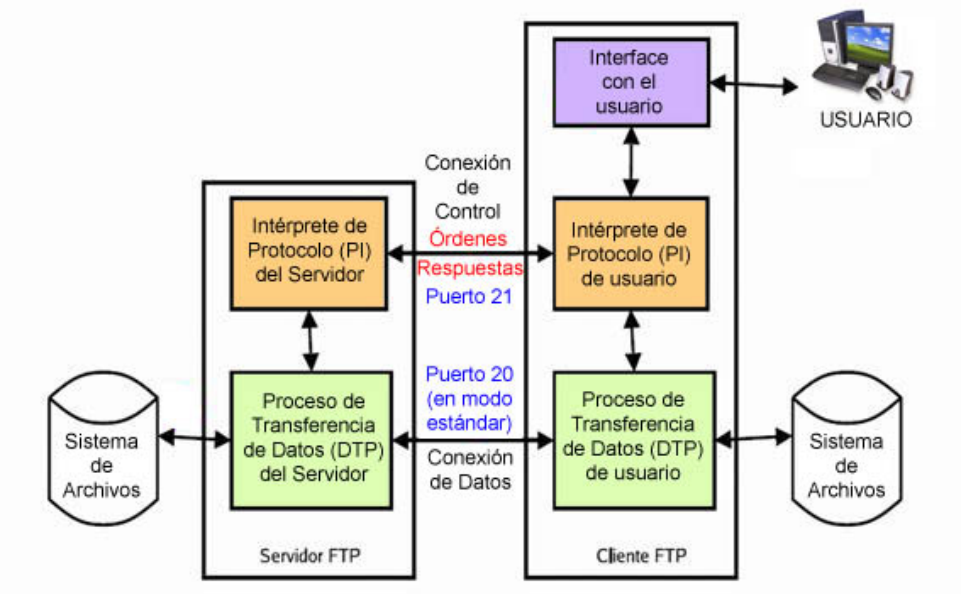


Figura 1.7 Modelo FTP

El proceso de transferencia de datos (DTP) de usuario u otro proceso en su lugar, debe esperar a que el servidor inicie la conexión al puerto de datos especificado (puerto 20 en modo activo o estándar) y transferir los datos en función de los parámetros que se hayan especificado.

La comunicación es independiente del sistema de archivos utilizado en cada ordenador porque las entidades que se comunican entre sí son los PI y los DTP, que usan el mismo protocolo estandarizado: el FTP. También hay que destacar que la conexión de datos es bidireccional, es decir, se puede usar simultáneamente para enviar y para recibir, y no tiene por qué existir todo el tiempo que dura la conexión.

4.3 Los Servidor FTP

Un servidor FTP es un programa especial que se ejecuta en un equipo servidor normalmente conectado a Internet. Su función es permitir el intercambio de datos entre diferentes servidores/ordenadores, no suelen encontrarse en los ordenadores personales.

Las aplicaciones más comunes de los servidores FTP suelen ser el alojamiento Web, en el que sus clientes utilizan el servicio para subir sus páginas y archivos; o

como servidor de backup (copia de seguridad). Para ello, existen protocolos de comunicación FTP en el que los datos se transmiten cifrados, como el SFTP (Secure File Transfer Protocol).

4.4 Los Clientes FTP

Un cliente FTP es un programa que se instala en el ordenador del usuario, y que emplea el protocolo FTP para conectarse a un servidor FTP y transferir archivos, ya sea para descargarlos o para subirlos. Para utilizar un cliente FTP, se necesita conocer el nombre del fichero, el ordenador en que reside (servidor, en el caso de descarga de archivos), el ordenador al que se quiere transferir el archivo (en caso de querer subirlo nosotros al servidor), y la carpeta en la que se encuentra.

Algunos clientes de FTP básicos en modo consola vienen integrados en los sistemas operativos. Hay disponibles clientes con opciones añadidas e interfaz gráfica. Aunque muchos navegadores tienen ya integrado FTP, es más confiable a la hora de conectarse con servidores FTP no anónimos utilizar un programa cliente.

4.5 Acceso anónimo

Los servidores FTP anónimos ofrecen sus servicios libremente a todos los usuarios sin tener una cuenta de usuario. Es la manera más cómoda fuera del servicio Web de permitir que todo el mundo tenga acceso a cierta información. Si un servidor posee servicio FTP “anonymous” solamente con teclear la palabra "anonymous", cuando pregunte por tu usuario tendrás acceso a ese sistema. No se necesita ninguna contraseña preestablecida, aunque tendrás que introducir una sólo para ese momento, normalmente se suele utilizar la dirección de correo electrónico propia.

Solamente con eso se consigue acceso a los ficheros del FTP, aunque con menos privilegios que un usuario normal. Normalmente solo podrás leer y copiar los ficheros existentes, pero no modificarlos ni crear otros nuevos. Normalmente, se utiliza un servidor FTP anónimo para depositar grandes ficheros que no tienen utilidad si no son transferidos a la máquina del usuario, como por ejemplo programas, y se reservan los servidores de páginas web (HTTP) para almacenar información textual destinada a la lectura en línea.

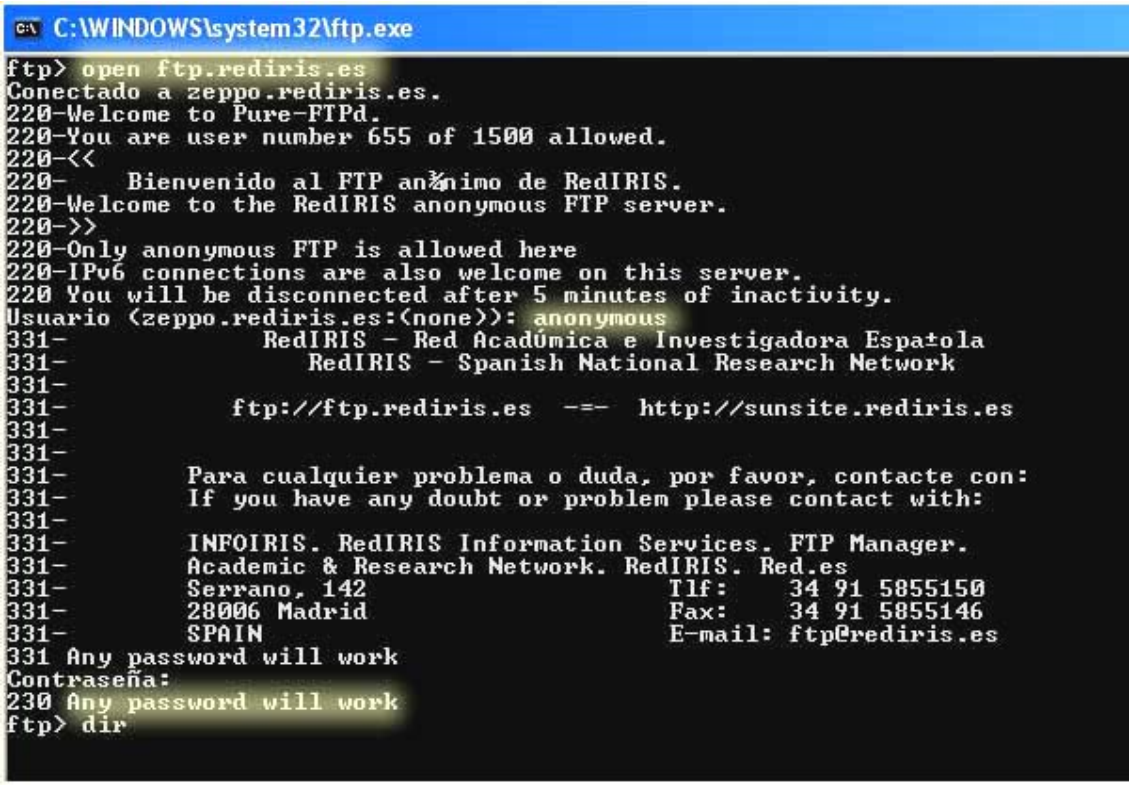


Figura 1.8 Conexión a ftp.rediris.es.

4.6 Acceso de usuario

Si se desean tener privilegios de acceso a cualquier parte del sistema de ficheros del servidor FTP, de modificación de archivos existentes, y de posibilidad de subir nuestros propios ficheros, generalmente se suele realizar mediante una cuenta de usuario. En el servidor se guarda la información de las distintas cuentas de usuario que pueden acceder a él, de manera que para iniciar una sesión FTP debemos introducir un “login” y un “password” que nos identifica unívocamente.

4.7 Acceso de invitado

El acceso sin restricciones al servidor que proporcionan las cuentas de usuario implica problemas de seguridad, lo que ha dado lugar a un tercer tipo de acceso FTP denominado invitado (guest), que se puede contemplar como una mezcla de los dos anteriores.

La idea de este mecanismo es la siguiente: se trata de permitir que cada usuario conecte a la máquina mediante su “login” y su “password”, pero evitando que

tenga acceso a partes del sistema de ficheros que no necesita para realizar su trabajo, de esta forma accederá a un entorno restringido, algo muy similar a lo que sucede en los accesos anónimos, pero con más privilegios.

4.8 Modos de conexión del cliente FTP

FTP admite dos modos de conexión del cliente. Estos modos se denominan Activo (o Estándar, o PORT, debido a que el cliente envía comandos tipo PORT al servidor por el canal de control al establecer la conexión) y Pasivo (o PASV, porque en este caso envía comandos tipo PASV). Tanto en el modo Activo como en el modo Pasivo, el cliente establece una conexión con el servidor mediante el puerto 21, que establece el canal de control.

4.8.1 Modo Activo

En modo Activo, el servidor siempre crea el canal de datos en su puerto 20, mientras que en el lado del cliente el canal de datos se asocia a un puerto aleatorio mayor que el 1024. Para ello, el cliente manda un comando PORT al servidor por el canal de control indicándole ese número de puerto, de manera que el servidor pueda abrirle una conexión de datos por donde se transferirán los archivos y los listados, en el puerto especificado.

Lo anterior tiene un grave problema de seguridad, y es que la máquina cliente debe estar dispuesta a aceptar cualquier conexión de entrada en un puerto superior al 1024, con los problemas que ello implica si tenemos el equipo conectado a una red insegura como Internet. De hecho, los cortafuegos que se instalen en el equipo para evitar ataques seguramente rechazarán esas conexiones aleatorias. Para solucionar esto se desarrolló el modo Pasivo.

4.8.2 Modo Pasivo

Cuando el cliente envía un comando PASV sobre el canal de control, el servidor FTP abre un puerto efímero (cualquiera entre el 1024 y el 5000) e informa de ello al cliente FTP para que, de esta manera, sea el cliente quien conecte con ese puerto del servidor y así no sea necesario aceptar conexiones aleatorias inseguras para realizar la transferencia de datos.

Antes de cada nueva transferencia, tanto en el modo Activo como en el Pasivo, el cliente debe enviar otra vez un comando de control (PORT o PASV, según el modo en el que haya conectado), y el servidor recibirá esa conexión de datos en un nuevo puerto aleatorio (si está en modo pasivo) o por el puerto 20 (si está en modo activo).

4.9 Tipos de transferencia de ficheros en FTP

Es importante conocer cómo debemos transportar un fichero a lo largo de la red. Si no utilizamos las opciones adecuadas podemos destruir la información del fichero. Por eso, al ejecutar la aplicación FTP, debemos acordarnos de utilizar uno de estos comandos (o poner la correspondiente opción en un programa con interfaz gráfica):

4.9.1 type ascii

Adecuado para transferir ficheros que sólo contengan caracteres imprimibles (ficheros ASCII, no ficheros resultantes de un procesador de texto), por ejemplo páginas HTML, pero no las imágenes que puedan contener.

4.9.2 type binary

Este tipo es usado cuando se trata de ficheros comprimidos, ejecutables para PC, imágenes, archivos de audio.

4.10 Ejemplos de comandos FTP:

Comando	Acción que realiza
open servidor	Inicia una conexión con un servidor FTP.
close disconnect	Finaliza una conexión FTP sin cerrar el programa cliente
Get archivo(s)	Ordena la recepción de los archivos indicados.
Put archivos(s)	Envía un archivo al directorio activo del servidor
ls	Muestra el contenido del directorio en el servidor.

Figura 1.9 Ejemplos de comandos FTP.

TEMA 2: SERVICIOS DE DIRECTORIOS

Objetivos

- Exponer las necesidades de los servicios de directorios y su diferencia con las bases de datos relacionales.
- Estudiar la forma en que se declaran, organizan y accede a los objetos dentro del directorio X.500.
- Describir los principales ámbitos en los cuales se puede utilizar LDAP como servicio de directorio.
- Analizar la importancia, estructura, funcionamiento del DNS dentro de Internet.

Contenido del Capítulo

- Introducción a los Servicios de Directorios.
- El estándar X.500 de ISO/ITU.
- Protocolo Ligero de Directorio Activo (LDAP).
- Servicio de Nombres de Dominio (DNS).

Bibliografía

- James F. Kurose, Keith W. Ross, Redes de Computadores: Un enfoque descendente basado en Internet, Segunda Edición
- Andrews S. Tanenbaum, Redes de Computadores, Cuarta Edición 2003.
- Fred Halsall, Redes de Computadores e Internet, Quinta Edición 2006.
- X.500: http://www.rediris.es/sdir/doc/intro.es.html
- http://www.ldap-es.org/book/print/21
- http://es.tldp.org/Tutoriales/doc-openldap-samba-cups-python/htmls/openldap-que-es.html
- http://www.it.uc3m.es/aat/Tema6.pdf

1. Introducción a los Servicios de Directorios

1.1 Introducción

Muchas veces nos encontramos con la necesidad de escribir o llamar a una determinada persona. Si queremos su número telefónico, con sólo saber el nombre y la localidad donde reside podemos encontrarlo en la guía telefónica.

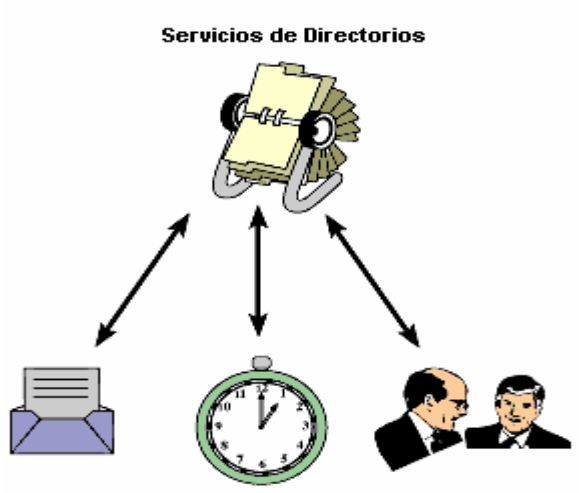


Figura 2.1 Servicios de Directorios.

A veces nos interesa mucho más como la dirección de correo electrónico del responsable de un determinado departamento o, sabiendo únicamente la dirección e-mail, el despacho que ocupa una persona concreta de la que ni tan siquiera sabemos el nombre de la institución donde trabaja o una foto de alguien con quien tenemos una reunión. Estos son los servicios que intentan proporcionar lo que se conocen como servicios de directorios.

1.2 Definición

Un servios de directorios, es un servicio que utilizan las redes para localizar recursos dentro de ellas. Estos recursos pueden ser: Información variada, direcciones email, host, impresoras, etc. Además posibilita a los usuarios y a las aplicaciones el acceso a dichos recursos de una manera transparente y prácticamente desde cualquier plataforma.

Además brinda un listado de información en un entorno distribuido, es decir, nos posibilita ver todo lo que hay distribuido en las diferentes máquinas como si solamente estuviesen en una (proporciona transparencia al usuario).

El servicio de directorio tiene dos partes principales que son:

- La base de datos, la cual se encuentra optimizada para la lectura, navegación y búsqueda.
- Un protocolo para acceder a los datos y poder manipularlos.

1.3 Características

Dentro de las principales características de los servicios de directorios tenemos:

- Organizan la información de forma jerarquizada en forma de árbol de objetos. Esta estructura refleja límites geográficos y organizacionales de la información almacenada. Se emplean métodos normalizados para referenciar los datos.

- Ofrecen a los usuarios, un servicio transparente en relación con la ubicación física de los datos, la topología de la red y los protocolos.

- Se tratan de servicios de directorios globales, donde los datos están distribuidos, repartidos en diferentes equipos que cooperan entre si para dar este servicio. Ejemplo: DNS.

1.4 Bases de Datos y Servicios de Directorios

Tanto las bases de datos como los ficheros de directorios tienen en común que ambos nos permiten el acceso a datos almacenados, sin embargo existen ciertas diferencias entre ellos, las cuales se describen de una manera rápida en el siguiente apartado pero que posteriormente se ampliarán cuando se vea el servicio de directorios LDAP. Entre las diferencias más significativas tenemos:

- A diferencias de las bases de datos que están preparados para realizar miles de actualizaciones por segundo, los servicios de directorios están preparados para optimizar las operaciones de lectura y búsqueda, no las de escritura. Estos no funcionan muy bien si hay actualizaciones frecuentes.

- Típicamente son distribuidos a diferencias de las bases de datos que por lo general son centralizadas.

1.5 Usos de un Servicio de Directorio

Los servicios de directorios son empleados típicamente para:

- Hacer accesible determinada información, por ejemplo abonados de una compañía, direcciones de e-mail, localización, etc.

- Guardar configuraciones asociadas a las aplicaciones, por ejemplo repositorios para usuarios itinerantes.

- Soporte a servicios de seguridad, por ejemplo almacén de firmas digitales, gestión de la infraestructura de clave pública, distribución de claves y cuentas de usuario.

Entre los principales estándares de servicios de directorios tenemos:

- X.500, estándar ISO/ITU.
- LDAP, estándar IETF.

2. El estándar X.500 de ISO/ITU

2.1 Introducción a X.500

A medida que las redes crecían, las bases de datos locales eran insuficientes y se pensó en almacenar toda la información que estaba hasta esos momentos dispersa en una única base de datos de ámbito universal. Esta gran base de datos estaría distribuida por todo el mundo de forma que pudiesen comunicar unas con otras y dieran al usuario la sensación de estar trabajando con una sola base de datos global.

En octubre de 1990 se presentó en las Jornadas Técnicas de IRIS un piloto de directorio para la red nacional. Durante 1991 se instalaron máquinas para formar una infraestructura básica del servicio consistente en servidores X.500, acceso público al directorio y servidores de archivo con software y documentación.

2.2 ¿Qué es el directorio X.500?

El directorio X.500 funciona sobre la torre de protocolos OSI y está basado en el directorio DAP (Directory Access Protocol), que es un protocolo para acceder a directorios y muy pesado ya que requiere muchos recursos computacionales. En principio todos los servicios de directorio se basan en este estándar, pero dada su complejidad y magnitud, no todos los suministradores lo cumplen al 100%.

El directorio es una base de datos destinada a mantener determinada información sobre objetos del mundo real. Es una base de datos distribuida pero de acceso transparente para el usuario. El Servicio de Directorio X.500 permite tener almacenados datos públicos de las Unidades organizativas y las personas que las componen, para poder realizar búsquedas de las mismas.

2.3 Infraestructura del Directorio X.500

La información contenida en el directorio se conoce como la Base de Información del Directorio, DIB (Directory Information Base). El directorio X.500 se compone de Agentes de Sistema de Directorio, DSAs (Directory System Agent), que mantienen la información distribuida del directorio.

Éstos constan de:

- Una base de datos propia que mantiene una parte de la información global del directorio (por eso es distribuida).

- Unos procedimientos de comunicaciones que permiten el diálogo entre los DSAs así como entre ellos y los usuarios por medio de los Agentes de Usuario de Directorio, DUA's (Directory User Agent).

La relación entre estos componentes se muestra en la siguiente figura:

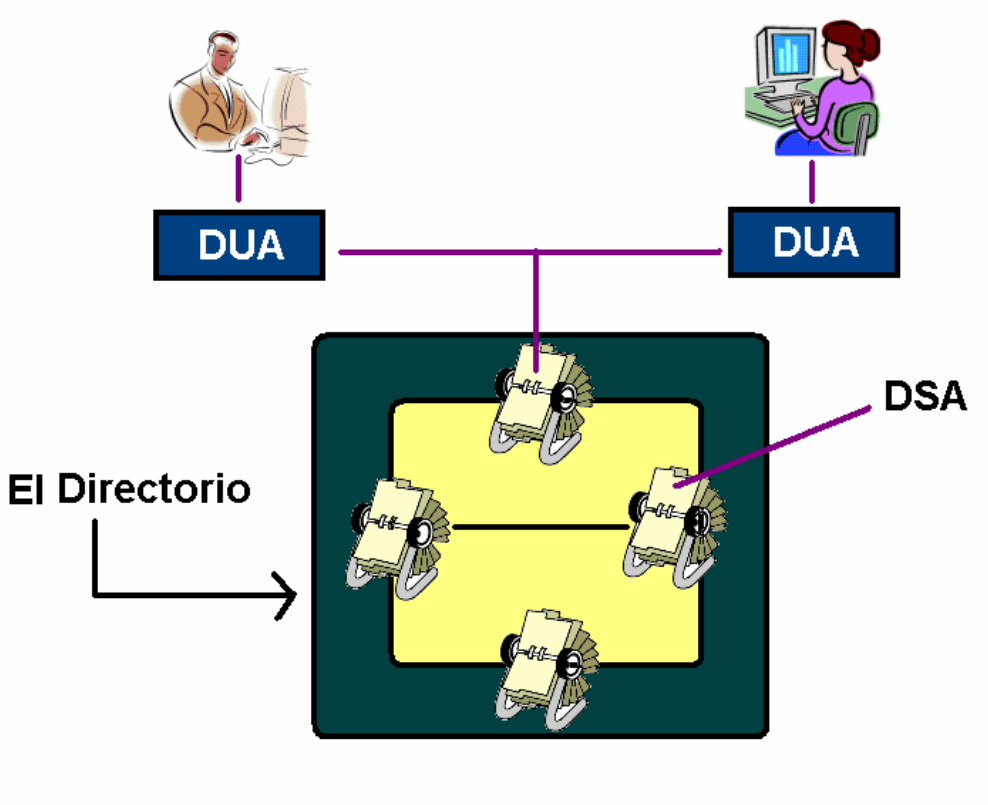


Figura 2.2 Estructura en X.500.

La información que se mantiene en el directorio está compuesta por objetos, entendiendo por éstos, entes muy diversos como cosas, animales, personas, grupos, organizaciones, países, aplicaciones OSI, etc.

2.4 Estructura de un objeto del directorio

Cada objeto tiene un nombre que es único de forma que permite localizarlo dentro del directorio. Este nombre, llamado Nombre Distintivo, DN (Distingued Name),

está formado por campos denominados atributos como: país (c), organización (o), unidad de organización (ou), nombre del objeto (cn) y muchas más propiedades del objeto.

El DN, se encuentra formado por una secuencia unida a Nombres Distintivos Relativos, RDN (Relative Distingued Name). La secuencia se forma, con el DN del padre y el atributo nombre del objeto, que actúa como Nombre Distintivo Relativo.

- DN: "@c=CL@o=Departamento Ing. Infomatica Uscah@cn= Juan Pérez"
- RDN 1: c = CL
- RDN 2: o = Departamento Ing. Infomatica Uscah
- RDN 3: cn = Juan Pérez

2.5 Estructura del Árbol de Información del Directorio (DIT)

La información contenida en el directorio se estructura de una forma jerárquica en niveles, partiendo de un punto ficticio llamado raíz que se sitúa en el nivel superior de la información y del que parten todos los demás. Algunos de los niveles utilizados en el servicio son:

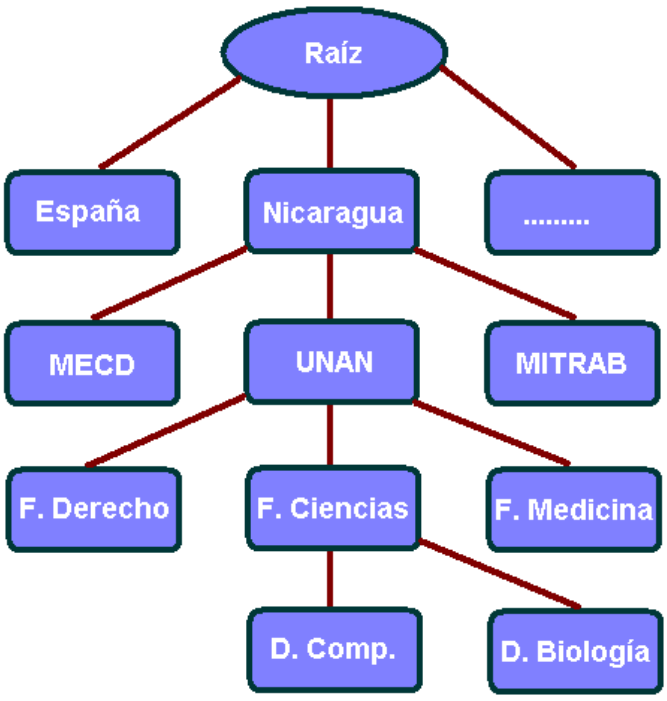


Figura 2.3 Ejemplos de niveles utilizados en X.500.

En cada uno de los niveles hay una persona encargada del mantenimiento de éste. Esta estructuración forma lo que se llama el árbol de Información del Directorio, DIT (Directory Information Tree), que pretende posibilitar una búsqueda de cada objeto, de la forma más rápida, segura y sencilla posible.

2.6 Agentes de Usuario de Directorio

El usuario podrá acceder al directorio mediante los Agentes de Usuario de Directorio (DUA's) que como se mencionó anteriormente, son los intermediarios entre el directorio y el usuario.

Estos agentes, los DUA's, pueden permitir al usuario hacer varias operaciones sobre el directorio como:

- Añadir una entrada.
- Modificar la entrada del propio usuario.
- Borrar la entrada del usuario.
- Lectura de atributos que describen un objeto, ya sean del usuario o de cualquier otro objeto.
- Listado de objetos que pertenecen a un país, organización, unidad de organización, etc.
- Búsqueda de objetos que cumplan unas determinadas condiciones.

2.7 Aplicaciones suministradas por el directorio

Las aplicaciones suministradas se pueden dividir en dos clases dependiendo del tipo de usuarios que hagan uso del directorio. Estas pueden ser: aplicaciones interpersonales y aplicaciones entre sistemas.

2.7.1 Aplicaciones interpersonales

En este tipo de aplicaciones se realiza un diálogo entre usuario y directorio mediante un DUA. Existen dos tipos:

1. Aplicaciones de Páginas Blancas:

Se accede a la información detallando el DN del objeto a buscar y seleccionando un conjunto de atributos que son los que el directorio mostrará. Como ejemplos de uso de este tenemos:

- Obtener la lista: nombre y dirección, de todos los funcionarios de la UNAN - León.
- Obtener la lista: usuarios del Departamento de Informática de la UNAN - León.

Esto despliega listado de los atributos mencionados como: teléfono, direcciones de mensajería, etc.

2. Aplicaciones de Páginas Amarillas:

Se pueden obtener datos de forma selectiva, sólo de los registros cuyos atributos coincidan con los valores de los atributos buscados, como cuando buscamos en las páginas amarillas de la guía telefónica a los médicos, abogados, etc. Como ejemplos de uso de este tenemos:

- Obtener los datos de las personas que se llamen Javier, vivan en León y beban Cacao.
- Obtener las personas que trabajen en el MINSA y no beban licor.

Estas aplicaciones de comunicación interpersonal son el objetivo inicial del servicio.

2.7.2 Aplicaciones entre sistemas

En este caso, los usuarios que acceden al directorio son aplicaciones OSI. El directorio permite que los usuarios accedan a las aplicaciones de una forma más amigable. Para ello el usuario especifica el servicio que desea ejecutar y el directorio llama a la aplicación que realiza ese servicio en la máquina correspondiente.

Por otra parte, también facilita a los técnicos de la red su gestión y administración, ya que cualquier cambio en alguna de las direcciones en el directorio, será visible automáticamente por todas las aplicaciones que la utilicen.

3. Protocolo Ligero de Acceso a Directorio (LDAP)

3.1 Descripción de LDAP

Estándar IETF promovido por la industria, en ocasiones denominado, X.500-lite. Simplifica las funcionalidades de X.500, eliminando las características de poco uso de éste. Es un protocolo de acceso a directorios, similar a DAP pero más ligero que éste y consume menos recursos.

LDAP es prácticamente un protocolo de acceso a directorio para el servicio de directorio de X.500. Inicialmente los clientes LDAP accedían al directorio a través de una pasarela. Se basa en una arquitectura cliente-servidor utilizando la arquitectura de comunicaciones TCP/IP:

- Servidores LDAP (servidores X.500 simplificados o servidores LDAP puros en las versión 3) que contienen los datos.
- Clientes LDAP.

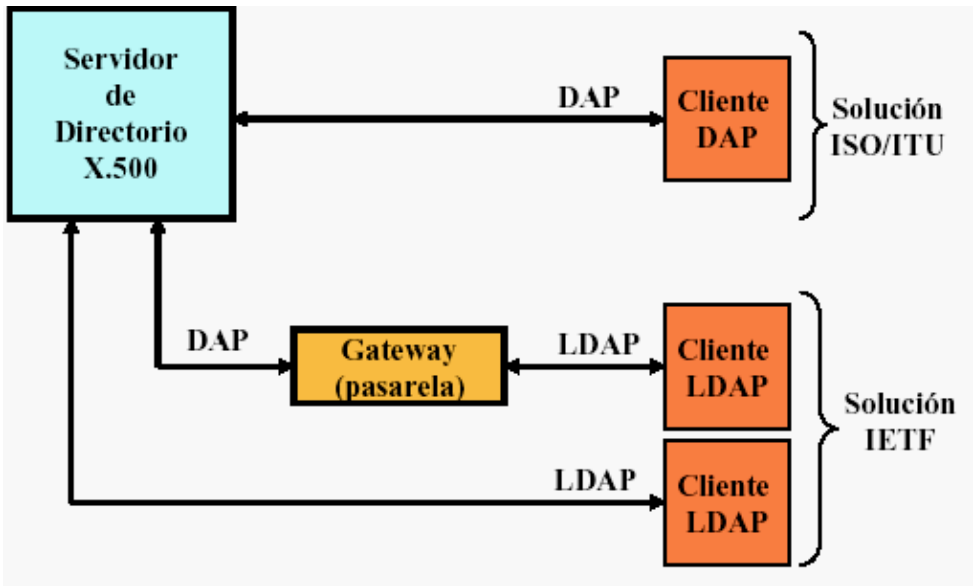


Figura 2.4 Acceso a un Servidor X.500 con LDAP.

Un servidor LDAP es usado para procesar peticiones (queries) a un directorio LDAP. LDAP es un tipo de base de datos, no relacional, ya que no está diseñada para procesar cientos o miles de cambios por minuto como los sistemas relacionales, sino para realizar lecturas de datos de forma muy eficiente.

3.2 Almacenamiento de la Información en LDAP

El modelo de información de LDAP está basado en entradas. Una entrada es una colección de atributos que tienen un único y global Nombre Distinguido (DN). El DN se utiliza para referirse a una entrada sin ambigüedades. Cada atributo de una entrada posee un tipo y uno o más valores.

Los tipos son normalmente palabras nemotécnicas, como “cn” para common name, o “mail” para una dirección de correo. La sintaxis de los atributos depende del tipo de atributo (cn =“Sergio González”; email=“sergio@ejemplo.com”.

En LDAP, las entradas están organizadas en una estructura jerárquica en árbol. Tradicionalmente, esta estructura reflejaba los límites geográficos y organizacionales. Las entradas que representan países aparecen en la parte superior del árbol. Debajo de ellos, están las entradas que representan los estados y las organizaciones nacionales. Debajo de estás, pueden estar las entradas que representan las unidades organizacionales, empleados, impresoras, documentos o todo aquello que pueda imaginarse.

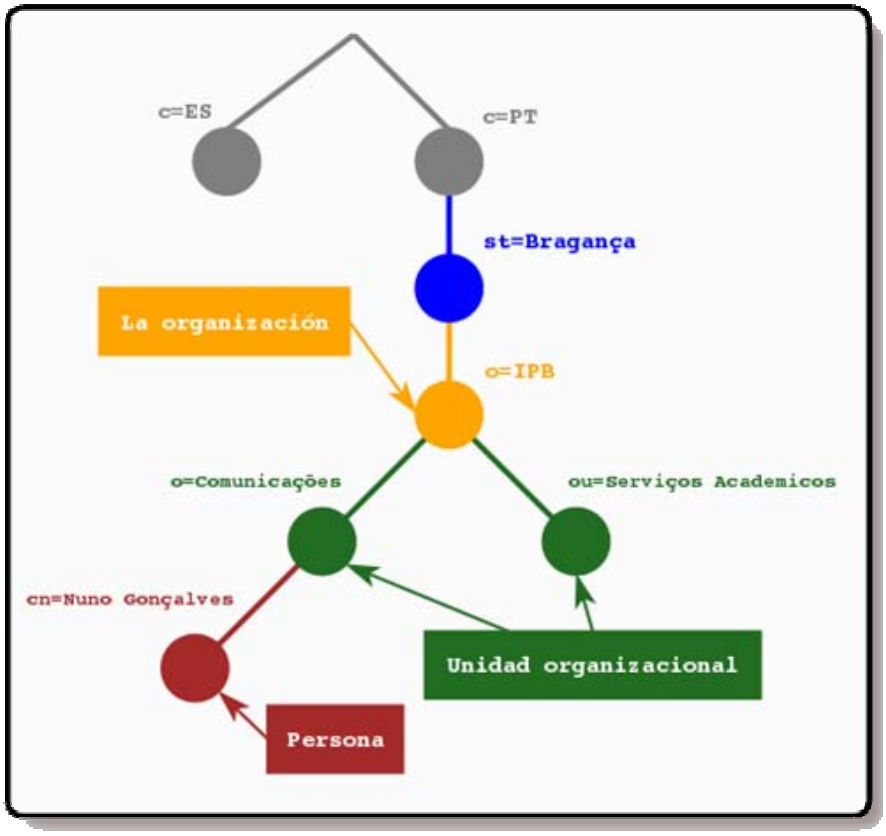


Figura 2.5 Ejemplo de un árbol de directorio LDAP haciendo uso del nombramiento tradicional.

El árbol también se puede organizar basándose en los nombres de dominio de Internet. Este tipo de nombramiento se está volviendo muy popular, ya que permite localizar un servicio de directorio haciendo uso de los DNS.

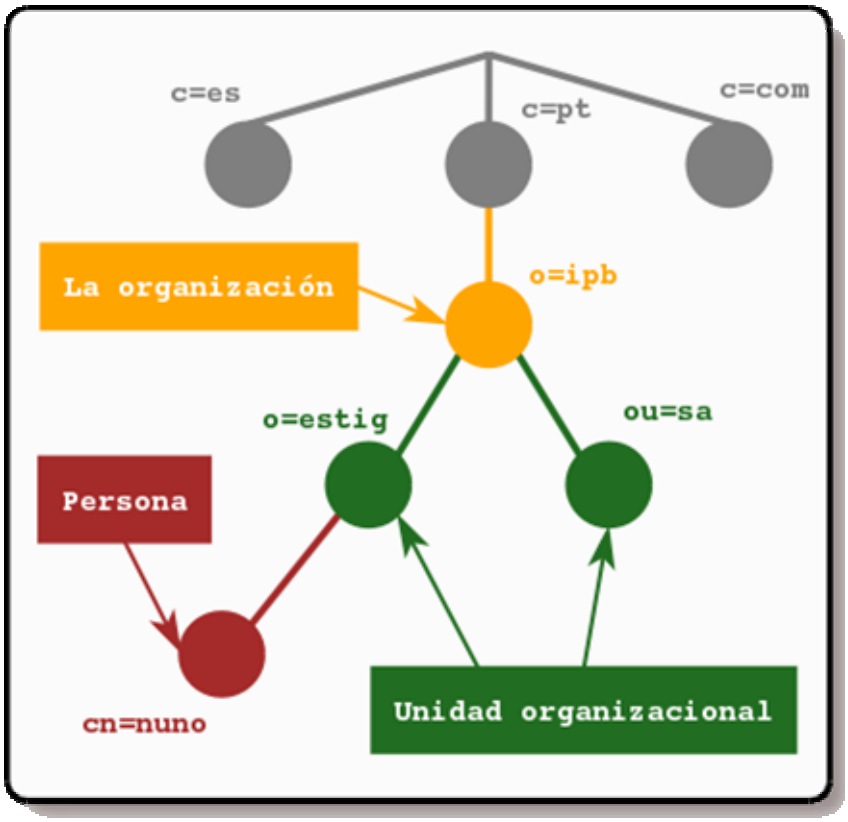


Figura 2.6 Árbol organizado basándose en DNS.

Además, LDAP permite controlar que atributos son requeridos y permitidos en una entrada gracias al uso del atributo denominado objectClass. El valor del atributo objectClass determina que reglas de diseño (schema rules) que ha de seguir la entrada.

3.3 Acceso a la Información en LDAP

Una entrada es referenciada por su nombre distinguido, que es construido por el nombre de la propia entrada (llamado Nombre Relativo Distinguido o RDN) y la concatenación de los nombres de las entradas que le anteceden.

Por ejemplo, la entrada para Nuno Gonçalves en el ejemplo del nombramiento de Internet anterior tiene:

- RDN: uid=nuno
- DN: uid=nuno,ou=estig,dc=ipb,dc=pt.

El formato completo para los DN está descrito en el RFC2253, “Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names.”

LDAP define operaciones para interrogar y actualizar el directorio. Provee operaciones para añadir y borrar entradas del directorio, modificar una entrada existente y cambiar el nombre de una entrada. La mayor parte del tiempo, sin embargo, LDAP se utiliza para buscar información almacenada en el directorio que concuerdan con algún criterio especificado por un filtro de búsqueda.

Por ejemplo, imagínese que quiere buscar en el subárbol del directorio que está por debajo de dc=ipb,dc=pt a personas con el nombre Nuno Gonçalves, obteniendo la dirección de correo electrónico de cada entrada que concuerde.

3.4 Seguridad de la Información

Algunos servicios de directorio no proveen protección, permitiendo a cualquier persona acceder a la información. LDAP provee un mecanismo de autentificación para los clientes, o la confirmación de identidad en un servidor de directorio, facilitando el camino para un control de acceso que proteja la información que el servidor posee. LDAP también soporta los servicios de privacidad e integridad.

3.5 Funcionamiento de LDAP

El servicio de directorio LDAP se basa en un modelo cliente-servidor como se menciono anteriormente. Uno o más servidores LDAP contienen los datos que conforman el árbol de directorio LDAP o base de datos troncal, el cliente LDAP se conecta con el servidor LDAP y le hace una consulta. El servidor contesta con la respuesta correspondiente, o bien con una indicación de donde puede el cliente hallar más información. No importa con que servidor LDAP se conecte el cliente ya que siempre observará la misma vista del directorio; el nombre que se le presenta a un servidor LDAP hace referencia a la misma entrada a la que haría referencia en otro servidor LDAP.

3.6 Características de un servidor LDAP

- **Operaciones de lectura muy rápidas**: Debido a la naturaleza de los datos almacenados en los directorios las lecturas son más comunes que las escrituras.
- **Datos relativamente estáticos:** Los datos almacenados en los directorios no suelen actualizarse con mucha frecuencia.
- Entorno distribuido, fácil replicación
- **Estructura jerárquica:** Los directorios almacenan la información de forma jerárquica de forma nativa.
- **Orientadas a objetos:** El directorio representa a elementos y a objetos. Los objetos son creados como entradas, que representan a una colección de atributos.
- **Esquema Standard:** Los directorios utilizan un sistema standard que pueden usar fácilmente diversas aplicaciones.
- **Atributos multi-valor:** Los atributos pueden almacenar un valor único o varios.
- **Replicación multi-master:** Muchos de los servidores LDAP permiten que se realicen escrituras o actualizaciones en múltiples servidores.

3.7 Ventajas en el uso de LDAP

Un directorio LDAP destaca sobre los demás tipos de bases de datos por las siguientes características:

- Permite replicar el servidor de forma muy sencilla y económica.
- Dispone de un modelo de nombres globales que asegura que todas las entradas son únicas.
- Permite múltiples directorios independientes.
- Funciona sobre TCP/IP y SSL.
- La mayoría de servidores LDAP son fáciles de instalar, mantener y optimizar.

3.8 Usos prácticos de LDAP

Dadas las características de LDAP sus usos más comunes son:

3.8.1 Directorios de información

Por ejemplo bases de datos de empleados organizados por departamentos (siguiendo la estructura organizativa de la empresa) ó cualquier tipo de páginas amarillas.

3.8.2 Sistemas de autenticación/autorización centralizada

Grandes sistemas donde se guarda gran cantidad de registros y se requiere un uso constante de los mismos. Por ejemplo:

- Active Directory Server de Microsoft, para gestionar todas las cuentas de acceso a una red corporativa y mantener centralizada la gestión del acceso a los recursos.
- Sistemas de autenticación para páginas Web, algunos de los gestores de contenidos más conocidos disponen de sistemas de autenticación a través de LDAP.
- Sistemas de control de entradas a edificios, oficinas.

3.8.3 Sistemas de correo electrónico

Cada usuario se identifica por su dirección de correo electrónico, los atributos que se guardan de cada usuario son su contraseña, su límite de almacenamiento (quota), la ruta del disco duro donde se almacenan los mensajes (buzón) y posiblemente atributos adicionales para activar sistemas anti-spam o anti-virus.

Como se puede ver este sistema LDAP recibirá cientos de consultas cada día (una por cada email recibido y una cada vez que el usuario se conecta mediante POP3 o webmail). No obstante el número de modificaciones diarias es muy bajo, ya que solo se puede cambiar la contraseña o dar de baja al usuario, operaciones ambas que no se realizan de forma frecuente.

3.8.4 Sistema de autenticación a una red

Cada usuario se identifica por un nombre de usuario y los atributos asignados son la contraseña, los permisos de acceso, los grupos de trabajo a los que pertenece, la fecha de caducidad de la contraseña.

Este sistema recibirá una consulta cada vez que el usuario acceda a la red y una más cada vez que acceda a los recursos del grupo de trabajo (directorios compartidos, impresoras…) para comprobar los permisos del usuario.

Frente a estos cientos de consultas solo unas pocas veces se cambia la contraseña de un usuario o se le incluye en un nuevo grupo de trabajo.

3.8.5 Otros Usos:

- Sistemas de alojamiento de páginas web y FTP con el repositorio de datos de usuario compartido.
- Grandes sistemas de autenticación basados en RADIUS para el control de accesos de los usuarios a una red de conexión o ISP.
- Servidores de certificados públicos y llaves de seguridad.
- Autenticación única ó “single sign-on” para la personalización de aplicaciones.
- Perfiles de usuarios centralizados, para permitir itinerancia ó “Roaming”
- Libretas de direcciones compartidas.

4. DNS (Servicio de Nombres de Dominio)

4.1 Motivación

De la misma forma que los humanos pueden ser identificados de muchas maneras también lo pueden ser los host de Internet. Un identificador, puede ser su nombre de host. Los nombre de host (como cnn.com, www.yahoo.es, o surf.eurecom.fr) son mnemotécnicos, y muy apreciados por las personas. Sin embargo, proporcionan muy poca información sobre la localización del host. Es más, dado que los nombres de host constan de caracteres alfanuméricos en longitud variable, serían difíciles de procesar por los routers. Por este motivo es que los host también son identificados por una dirección IP.

Una dirección IP es jerárquica, porque leída de izquierda a derecha, se obtiene cada vez información más específica sobre la ubicación de un host en Internet. De igual manera cuando se lee una dirección postal de abajo hacia arriba, se obtiene cada vez, más información específica sobre la localización del destinatario.

4.2 Historia: El archivo HOST

Una vez vista esta necesidad de establecer una relación entre direcciones IP y su nombre de host. Se planteó una primera solución. En la época de ARPANET, la traducción nombre-IP de todas las máquinas conectadas a la red se mantenía en un fichero de texto (HOST).

La información de este fichero era mantenida por el Network Information Center del Stanford Research Institute (SRN-NIC). Cada administrador de red remitía por correo los cambios que ocurrían en la red bajo su autoridad al NIC. Los administradores de redes también se encargaban de descargar periódicamente la última actualización del archivo HOST, para incluir los cambios en sus máquinas.

4.3 Problemas del archivo HOST

Al crecer el número de computadoras, el archivo se hizo demasiado grande y difícil de manejar. Cada vez un número mayor de administradores de red se

conectaban al servidor FTP del SRI-NIC para descargar un archivo que además crecía rápidamente, con lo que se presentaron los siguientes problemas:

- Las instalaciones del SRI-NIC no podían soportar semejante carga.
- Por otro lado, no había un mecanismo eficaz para evitar que aparecieran nombres duplicados. Por lo que pronto se hicieron frecuentes problemas de este tipo.
- También era cada vez más difícil mantener la consistencia del sistema de nombres: los cambios tardaban mucho en hacerse efectivos en todos los host. No todos tenían al mismo tiempo, la misma versión de HOST.

Todo lo anterior ocasionó la desaparición del archivo HOSTS y en ves de este se paso a emplear el método de sistema de nombres de dominios mejor conocido en la actualidad como DNS (*Domain Name Service*).

4.4 Sistema de nombres

Este sistema ante todo debía cumplir con las siguientes características:

- En primer lugar, debía ser capaz de repartir la carga entre varias máquinas: cada una debería mantener información local, pero hacerla accesible globalmente.
- La administración debía ser descentralizada, para que la carga no se concentre en un solo host y de esa manera evitar los cuellos de botella.
- El sistema de nombres debería evitar a toda costa que se pudieran dar nombres duplicados, con lo cual la solución fue, crear un sistema Jerárquico.

Con todo lo anterior a apareció la primera definición del DNS en las RFC's 882 y 883. Las especificaciones actuales, se encuentran en las RFC`s 1034 y 1035.

4.5 Definiciones y descripción

Gracias al DNS, se puede conciliar las diferencias, en la forma de trabajar, que tienen las personas (a través de nombres) y los routers (direcciones IP`s) a la hora de trabajar en Internet. El DNS es:

- Una base de datos distribuida implementada en una jerarquía de servidores de nombres.
- Una aplicación de la capa de aplicación que permite que se comuniquen los host y los servidores de nombres para proporcionar el servicio de traducción.

Los servidores de nombres son habitualmente máquinas UNIX que ejecutan el software de Berkeley Internet Name Domain (BIND). El protocolo DNS se ejecuta sobre UDP, y utiliza el puerto 53.

La estructura empleada es una estructura cliente-servidor que está compuesta por:

- **Servidores de Nombres:** Contiene información sobre fragmentos de la base de datos. Los utilizan para responder a peticiones de los clientes. Saben donde buscar los datos que no administran.

- **Clientes (resolvers):** Programas o librerías de funciones que formulan consultas a los servidores. El proceso por el que el servidor halla la respuesta es transparente para ellos.

4.6 Servicios proporcionados

DNS es empleado comúnmente por otros protocolos de la capa de aplicación (incluyendo HTTP, SMTP y FTP) para traducir los nombres de host proporcionados por los usuarios a direcciones IP's. DNS proporciona otros servicios importantes además de la traducción de nombres de host a direcciones IP:

- **Alias de host:** Un host con un nombre complejo puede tener uno o más nombres de alias. Por ejemplo, el nombre del host relay1.westcoast.enterprise.com podría tener, digamos, dos alias, como enterprise.com y www.enterprise.com. En este caso, el nombre relay1.westcoast.enterprise.com se dice que el nombre canónico. El DNS puede ser invocado por una aplicación para obtener el nombre canónico del host y su IP a partir de un álias.

- **Alias de servidores de correo:** Por razones obvias, es muy recomendable que las direcciones de correo electrónico sean mnemotécnicas. Por ejemplo denis@hotmail.com. Sin embargo, el nombre canónico del servidor de Hotmail podría ser algo así como relay1.westcoast.hotmail.com. El DNS puede ser invocado por una aplicación de correo electrónico para obtener el nombre canónico del host a partir del alias proporcionado, así como la dirección IP del host. De hecho, el registro MX permite que el servidor de correo y el servidor Web de una compañía sean nombres de host idénticos (alias); por ejemplo, el servidor Web y de correo de una compañía pueden ser ambos Hotmail.com.

- **Distribución de carga:** Cada vez más, el DNS es utilizado para realizar una distribución de carga entre servidores replicados, como es el caso de los servidores Web. Por tanto para servidores Web replicados, el nombre canónico de host está asociado con un conjunto de direcciones IP, la cual esta contenida en la base de datos DNS. Cuando un cliente hace una consulta DNS para un nombre que tiene asociado un conjunto de direcciones IP el servidor responde con el conjunto entero de direcciones IP, pero rota el orden de las direcciones en cada respuesta. Dado que el cliente generalmente envía su mensaje HTTP de petición a la primera dirección de la lista, la rotación DNS distribuye el tráfico sobre todos los servidores replicados.

4.7 Modo de funcionamiento

Se presenta ahora, un resumen a alto nivel de cómo trabaja el DNS. La discusión se centrará en el servicio de traducción de nombres de host a direcciones IP.

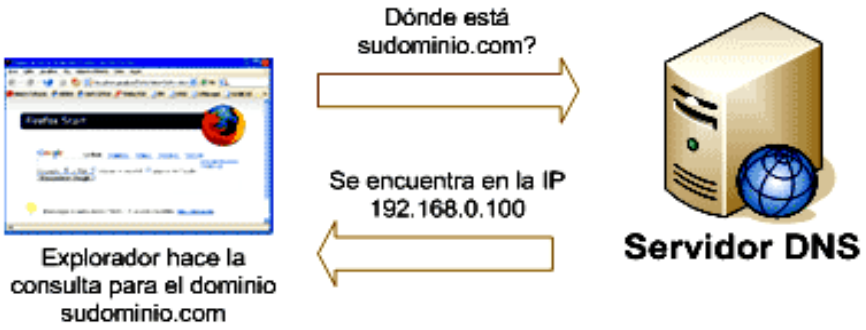


Figura 2.7 Modo de funcionamiento del DNS.

Suponga que una aplicación (como un navegador Web o un lector de correo) que se ejecuta en un host de usuario y que necesita traducir un nombre de host a dirección IP. La aplicación invocará el lado cliente del DNS, especificando el nombre de host que necesita ser traducido. (En muchas máquinas UNIX la función que se llama para realizar la traducción es gethostbyname()). Entonces cuando el lado cliente DNS en el host del usuario entra en funcionamiento, enviando un mensaje de petición a través de la red. Después de un retardo, que puede ir de un milisegundo a decenas de segundos, el DNS cliente recibe un mensaje DNS de respuesta que proporciona la correspondencia deseada. Esta correspondencia es pasada a la aplicación.

4.8 Diseño de la Base de Datos DNS

Desde la perspectiva de la aplicación invocadora en el host del usuario, el DNS es una caja negra que proporciona un servicio sencillo y directo de traducción. Pero en realidad, la caja negra que implemente el servicio es compleja, y consta de un gran número de servidores de nombres distribuidos por el mundo, además de contar de un protocolo de la capa de aplicación que especifica como se comunican los servidores los servidores de nombres y los host peticionarios. Para llevar a cabo toda la funcionalidad del DNS, este se concibe como una base de datos la cual podría tener un diseño centralizado o distribuido

4.8.1 Diseño Centralizado

Un diseño sencillo de DNS tendría un único servidor de nombres de Internet que contuviese todas las correspondencias. En este diseño, los clientes simplemente dirigen todas las consultas al servidor de nombres único y el servidor de nombres responde directamente a los clientes peticionarios.

Problemas del Diseño Centralizado:

A pesar del atractivo de tan simple diseño, este es inapropiado en la Internet actual, con su vasto y creciente número de estaciones. Los problemas de un diseño centralizado son:

- **Un único punto de fallo:** Si el servidor de nombres falla, también lo hace toda la Internet.

- **Volumen de tráfico:** Un único servidor de nombres tendría que gestionar todas las consultas generadas por millones de estaciones en el mundo.

- **Base de datos centralizada distante:** Un único servidor de nombres no puede estar cerca de todos los clientes que lo consultan.

- **Mantenimiento:** Un único servidor debería de registrara todos los host de Internet. No sólo sería gigantesca esta base de datos sino que tendría que ser frecuentemente actualizada para dar cuenta de cada nuevo host. Existe también un problema de autenticación y autorización asociado a permitir que cualquier usuario registre un host en base de datos centralizada.

En resumen, una base de datos centralizada en un único servidor de nombres simplemente no es escalable.

4.8.2 Diseño Distribuido

Consecuentemente el DNS es distribuido por diseño, por lo que constituye un excelente ejemplo de una base de datos distribuida implementada en Internet.

Para poder gestionar el tema de la escala, el DNS utiliza un gran número de servidores de nombres organizados de forma jerárquica y distribuida alrededor del mundo. No existe ningún servidor de nombre que contenga todas las correspondencias para todos los host en Internet, sino que las correspondencias están distribuidas por los servidores de nombres.

Tipos de servidores de nombres:

En una primera aproximación, existen tres tipos de servidores de nombres, que podemos diferenciar por su interacción con el host peticionario:

- **Servidores locales de nombres:** Cada proveedor de servicio de Internet (ISP) tiene un servidor local de nombres (o por defecto). Cuando un host emite un mensaje DNS de consulta, dicho mensaje es enviado, en primer lugar, al servidor local de nombres del host. La dirección IP del servidor local de nombres se configura, por lo general, manualmente.

- **Servidores raíz de nombres:** En Internet hay alrededor de una docena de servidores raíz de nombres, la mayoría de los cuales se encuentran actualmente en Norteamérica. Cuando un servidor de nombres local no puede satisfacer la consulta de un host, el servidor local se comporta como un cliente DNS y hace la consulta a uno de los servidores raíz de nombres.

- **Servidores autorizados de nombres:** Todo host se registra en un servidor autorizado de nombres. Típicamente, el servidor autorizado es un servidor de nombres en el ISP local del host. (En realidad todo host está obligado a tener al menos dos servidores autorizados de nombres para el caso de fallos). Por definición un servidor de nombres está autorizado para un host si siempre tiene un registro DNS que traduzca el nombre del host a su dirección IP.

  Muchos servidores de nombres se comportan al mismo tiempo como servidores locales y como autorizados.

4.9 Estructura de la Base de Datos DNS

4.9.1 Espacio de Nombres

Es una estructura JERÁRQUICA, similar a la de un sistema de ficheros UNIX. Ambos se pueden representar como un árbol invertido. El árbol recibe el nombre de **ESPACIO DE NOMBRES.**

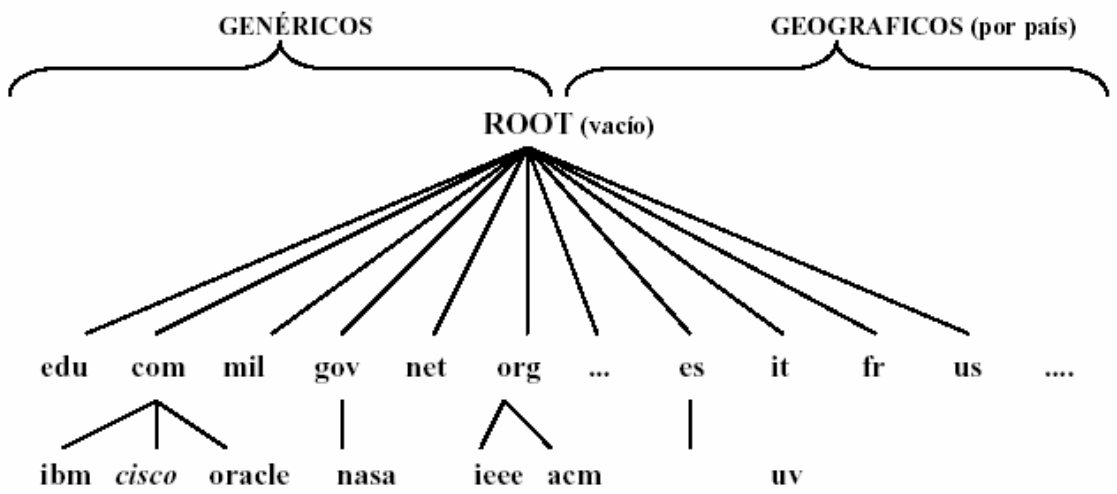


Figura 2.8 Espacio de Nombres DNS.

En DNS la raíz árbol se nombre con una etiqueta nula (“”) y se representa con un punto (“.”). Cada nodo representa una parte de la base de datos y se conocen con el nombre de dominios. Cada dominio se puede dividir en subdominios de hasta 127 niveles. Cada dominio tiene una etiqueta que lo identifica dentro de su dominio padre. Está etiqueta de un máximo de 63 caracteres no puede coincidir con la de ningún otro dominio que tenga el mismo padre. Un dominio tiene también un nombre completo (FQDN) que lo identifica de manera única dentro de un espacio de nombres de dominio. Se construye concatenando las etiquetas de los dominios que forman el camino hasta la raíz, separadas por puntos. Conceptualmente cada dominio tiene un servidor de nombres que resuelve peticiones. En la práctica no es así, hay organizaciones con varios subdominios que están en el mismo servidor de nombres.

4.9.2 Administración de la Base de Datos

La administración de todos los dominios de Internet no es única sino que es distribuida. Una organización encargada de un dominio por ejemplo .edu, puede decidir dividirlo en subdominios, y delegar responsabilidad de su administración en otras organizaciones. Más aún, la división no tiene por qué corresponder con dominios enteros, sino que puede llevarse a cabo de manera más flexible, dando origen a lo que se llaman zonas. Los dominios, además de contener subdominios, pueden contener host.

El espacio de nombres para el que existe una única autoridad se administrativa se conoce como dominio de nombres. Dicha autoridad, ejerce una autoridad sobre los nombres que se pueden lanzar al dominio, aunque puede delegar esta tarea a cada subdominio. Los dominios DNS son colecciones de nombres de dominio donde el nombre de dominio de todas estas es el sufijo que es común a todos ellos. La estructura jerárquica evita nombres duplicados, a la vez que deja libertad a los administradores para elegir los nombres de subdominios y host.

4.10 Servidores de Nombres

4.10.1 Descripción general

Son los programas que se encargan de almacenar información sobre el espacio de nombres de dominio. Un servidor de nombres tendrá información completa

sobre una o varias zonas del espacio de nombres de dominio. Se dice que el servidor mantiene información autorizada sobre dichas zonas.

Para mejorar la confiabilidad, algunos servidores de ciertas zonas pueden colocarse fuera de la zona. El lugar donde se colocan los límites de la zona es decisión del administrador de la zona. Esta decisión se toma en gran medida con base en la cantidad de servidores de nombres y su ubicación.

Si hay delegación de zonas, el servidor almacenará también referencias a los servidores que contiene información autorizada para dichas zonas.

4.10.2 Tipos de servidores

Existe una segunda forma de clasificar los servidores de nombres en relación con zonas concretas, esta es en primarios y secundarios.

- **Primarios:** Obtienen la información sobre zonas de archivos contenidos en la máquina en que corren. Contienen información autorizada para esa zona.

- **Secundarios:** Obtienen la información sobre zonas de otros servidores autorizados para esas zonas.

Cuando un secundario arranca, contacta con los primarios necesarios y descarga de ellos toda la información sobre zonas. Una vez funcionando, el secundario pide periódicamente al primario, información actualizada sobre la zona. Esta redundancia en la información sobre las zonas ayuda a repartir la carga sobre los distintos servidores, y añade seguridad al sistema.

4.11 Clientes DNS

Los resolver son los clientes del sistema DNS. Sus principales funciones son:

- Consultar al servidor.
- Interpretan la respuesta. Si no llega o es incorrecta, posiblemente vuelvan a formular la consulta.
- Devuelven la información al programa que la pidió.

Algunas implementaciones más elaboradas hacen cosas más sofisticadas, como mantener en caché la información ya consultada por si se quiere de nuevo. Normalmente son librerías de funciones que están compiladas en los programas que utilizan los servicios de red.

4.12 Resolución

Es el proceso por el cual se busca en el espacio de nombres de dominio la información correspondiente a un dominio determinado. Cuando un servidor recibe una consulta de un resolver, busca en sus registros la información correspondiente. Si la halla la devuelve sino, depende del tipo de consulta que el cliente haya mandado, que puede ser recursiva e iterativa.

- **Consulta Recursiva:** Un servidor que recibe una consulta recursiva debe responder con la información pedida, o con un código de error. Si sus ficheros no contienen el dato pedido, tendrá que consultar el mismo a otro servidor.

- **Consulta Iterativa:** Ante una consulta iterativa, si el servidor no tiene el dato pedido, puede devolver simplemente la dirección de otro servidor al que consultar.

4.13 Resolución Inversa

4.13.1 Descripción general

El tipo más común de consultas a un servidor de nombres es el número IP a partir de un nombre. Si embargo, también es importante el proceso inverso, el cual consiste en hallar un nombre a partir de una dirección IP. Ejemplo: En los ficheros log, la resolución inversa hace que se puedan grabar nombres, además de IP's: la información resulta más legible y útil. También se utilizan para comprobar autorización de acceso a algunos servicios.

En DNS los datos están indexados por nombre. Para hacer una consulta en principio tendríamos que ir consultando a todos los servidores. Esta solución es poco satisfactoria. Para hacer fiables las consultas inversas, se ha creado una parte del

espacio de nombres que utiliza direcciones como si fueran nombres. Se trata del dominio in-addr.arpa.

4.13.2 El dominio in-addr.arpa

Bajo in-addr.arpa hay 256 posibles dominios que corresponden al primer octeto de las posibles IP's. Bajo cada uno de estos dominios hay otras 256 posibilidades que corresponden al segundo octeto y así hasta completar una estructura que contiene un dominio para cada posible IP (en el nivel 6).

Para cada uno de estos dominios se registra en la base de datos una referencia al nombre de máquina correspondiente. Para leer la IP asociada a cada uno de los dominios in-addr.arpa, debemos hacerlo en orden inverso al que utilizamos en el caso de nombres de dominio: desde la raíz hacia el dominio en cuestión.

Esto se debe a que en un nombre de dominio, porciones del nombre situadas más a la izquierda representan entidades más específicas, mientras que en las direcciones IP es al revés. El dominio in-addr.arpa se ha organizado de esta manera para garantizar la delegación de zonas de dominio inverso que se correspondan exactamente con zonas del dominio de nombres correspondientes.

4.14 Caché DNS

Los servidores DNS utilizan ampliamente la caché DNS para acelerar el proceso de resolución de nombres. Cuando un servidor se ve obligado a pedir información a otros, el proceso está aprendiendo cosas sobre el espacio de nombres de dominio.

Todos los datos que utiliza el servidor para hallar la respuesta final a una consulta así como el dato pedido, se conservan por un espacio de tiempo. Si se recibe otra consulta, el servidor puede tener la respuesta todavía en caché y no tiene que pedirla de nuevo.

Los datos en caché tienen que expirar y ser descartados en algún momento, de lo contrario nunca se harán efectivos los cambios introducidos en los servidores

autorizados. Esto se controla mediante un parámetro de configuración llamado TTL (Time To Live), que puede tener un tiempo máximo de vida de 24 o 48 horas.

Tiene que haber un compromiso entre:

- Optimización del rendimiento (TTL largos).
- Mantenimiento de la consistencia de DNS (TTL cortos).

Como ejemplo, supóngase que ciencias.unanleon.edu.ni consulta el DNS para la dirección de yahoo.com. Es más, supóngase que horas más tarde , otro host de la UNAN (por ejemplo medicina.unanleon.edu.ni) también hace la misma consulta DNS anterior. Por medio de la caché, el servidor local de la UNAN podrá devolver inmediatamente al host peticionario de la dirección IP de yahoo.com sin tener que consultar servidores de nombres en otros países. Cualquier servidor de nombres puede tener correspondencias DNS en la caché.

4.15 Registros del DNS

Los servidores de nombres que, conjuntamente implementan la base de datos distribuida DNS, almacenan registros de recursos (RR) para las correspondencias nombre de host/dirección IP. Cada mensaje de respuesta DNS transporta uno o más registros de recursos.

Los registros de recursos se encuentran formados de 5 elementos que son:

(Nombre, TTL, Clase, Tipo, Valor)

Donde:

- **Nombre:** Indica el dominio al que pertenece este registro. Por lo general existen muchos registros por dominios y cada copia de la base de datos contiene información de muchos dominios. Por tanto, este campo es la clave primaria de búsqueda usada para atender las consultas.

- **Tiempo de vida (TTL):** Es una indicación de la estabilidad del registro. La información altamente estable recibe un valor grande, como la cantidad de segundos en un día (86,400), mientras que la información altamente volátil, recibe un valor pequeño como 69 segundos.

- **Clase:** Para información de Internet, siempre es IN. Para información que no es de Internet se pueden utilizar otros códigos, pero en la práctica, raramente se ven.

- **Valor:** Es el valor del recurso y depende directamente del tipo del recurso. Puede ser un número, un nombre de dominio o una cadena ASCII.

- **Tipo:** Los valores que puede tomar el campo tipo se especifican en la tabla siguiente:

Tipo	Significado	Valor
SOA	Inicio de autoridad	Parámetros para esta zona
A	Dirección IP de un host	Entero de 32 bits
MX	Intercambio de correo	Prioridad, dominio dispuesto a aceptar correo electrónico
NS	Servidor de nombres	Nombre de un servidor para este dominio
CNAME	Nombre canónico	Nombre de dominio
PTR	Apuntador	Alias de una dirección IP
HINFO	Descriptor del host	CPU y SO en ASCII
TXT	Texto	Texto ASCII no interpretado

Figura 2.9 Valores que puede tomar el campo tipo de los registros de recursos.

Si un servidor de nombres está autorizado para un nombre de host particular, entonces el servidor contendrá un registro de tipo A para el nombre del host. (Incluso si el servidor de nombres es no autorizado, podría contener un registro de tipo A en su caché). Si el servidor no está autorizado para el nombre del host, entonces contendrá un registro de tipo NS para el dominio que incluye el nombre del host; también contendrá un registro de tipo A que proporciona la dirección IP del servidor de nombres que aparece en el campo VALOR del registro NS.

Supongamos que un servidor raíz no está autorizado para el host ciencias.unanleon.edu.ni. Entonces el servidor raíz contendrá un registro para un dominio que incluye al host unanleon.edu.ni por ejemplo, (edu.ni, dns.edu.ni, NS). El

servidor raíz también dispondrá de un registro tipo A que establece la dirección IP del servidor de nombres dns.edu.ni; por ejemplo, (dns.edu.ni, 128.119.40.15, A).

4.16 Mensajes DNS

Anteriormente se hizo alusión a los mensajes DNS de consulta y de respuesta. Estos son los dos únicos tipos de mensajes DNS los cuales tienen el mismo formato, como se muestra a continuación:

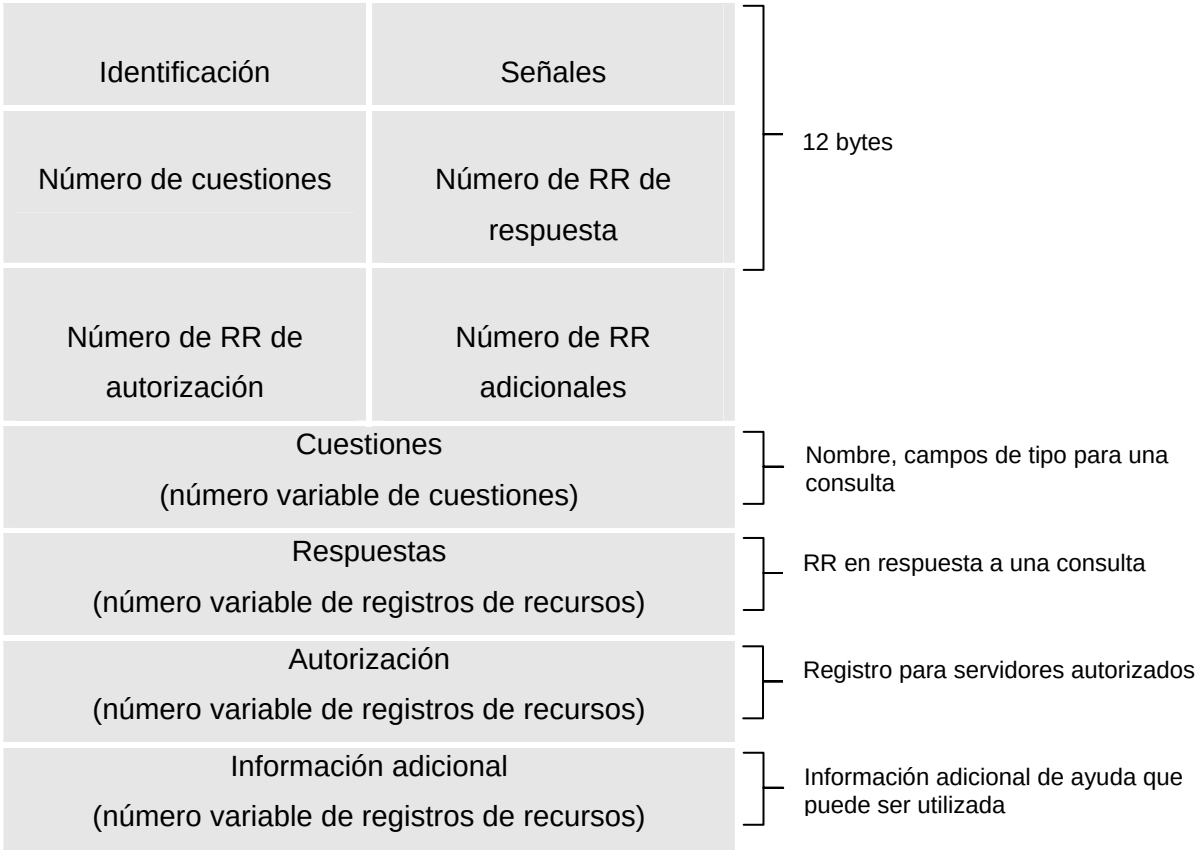


Figura 2.10 Formato de los mensajes DNS.

- **Sección de cabecera:** Son los primeros 12 bytes, que contienen un conjunto de campos. El primer campo es un número de 16 bits que identifica a la consulta. Este identificador se copia en el mensaje de respuesta a la consulta, de forma que el cliente pueda comparar las respuestas recibidas con las consultas enviadas. En el campo de señalización (flag), hay un conjunto de señales. La señal de 1 bit que indica si el mensaje es de consulta (0) o respuesta (1). La señal de 1 bit de autorización es fijada en un mensaje de respuesta cuando un servidor de nombres es un servidor autorizado para un nombre consultado. La señal de 1 bit de

recursión deseada es fijada cuando el cliente (host o servidor de nombres) desea que el servidor de nombres realice recursión cuando no disponga del registro. Una señal de 1 bit de recursión disponible es fijada en una respuesta si el servidor de nombres permite recursión. En la cabecera hay también cuatro campos “Numero de”. Estos campos indican el número de ocurrencias de los cuatro tipos de secciones de datos que siguen a la cabecera.

- **Sección de consultas:** Contiene información sobre la pregunta que se está haciendo. Esta incluye un campo de nombre, que contiene el nombre que está siendo consultado, y un campo tipo que indica el tipo de cuestión preguntada sobre el nombre. Ejemplo, la dirección del host asociada con un nombre (tipo A), o el servidor de correo para un nombre (tipo MX).

- **Sección de respuesta:** Contiene los registros de recursos para el nombre por el que originalmente se pregunto. Recuerdese que en cada registro de recurso está el TIPO (por ejemplo A, NS, CNAME y MX), el VALOR y el TTL. Una respuesta puede devolver múltiples RR en la respuesta, ya que el nombre de host puede tener múltiples direcciones IP (por ejemplo el caso de los servidores Web replicados).

- **Sección de autorización:** Contiene registros de otros servidores de nombres autorizados.

- **Sección adicional:** Contiene otros registros de ayuda. Por ejemplo, el campo de respuesta en una respuesta a una consulta MX contiene el registro de recurso que proporciona el nombre de host canónica de un servidor de correo. La sección adicional contiene un registro tipo A que proporciona la dirección IP para el nombre de host canónico del servidor de correo.

La discusión anterior ha estado centrada en como son recuperados los datos de la base de datos DNS. Se podría hacer la pregunta de cómo los datos son introducidos en la base de datos en primer lugar. Hasta hace poco, los contenidos de los servidores DNS eran configurados estáticamente (por ejemplo, en un archivo de configuración creado por el administrador del sistema). Más recientemente se ha añadido al protocolo DNS una opción UPDATE que permite que los datos sean añadidos o borrados de la base de datos por medio de mensajes DNS.

TEMA 3: WORLD WIDE WEB

Objetivos

- Describir los elementos esenciales que constituyen la arquitectura Web.
- Introducir los conceptos básicos de HTML y la utilización para la creación de páginas Web estáticas.
- Comparar las diferentes técnicas para el desarrollo de aplicaciones Web dinámicas.
- Estudiar el funcionamiento y la estructura de los mensajes que utiliza el protocolo HTTP.
- Exponer diversas técnicas utilizadas en la actualidad para mejorar el desempeño de los servidores Web.

Contenido del Capítulo

- Arquitectura de la Web.
- Documentos Web Estáticos.
- Documentos Web Dinámicos.
- El Protocolo HTTP.
- Mejoras en el Desempeño.

Bibliografía

- Andrews S. Tanenbaum, Redes de Computadores, Cuarta Edición 2003.
- James F. Kurose, Keith W. Ross, Redes de Computadores: Un enfoque descendente basado en Internet, Segunda Edición.

1. Arquitectura de la Web

1.1 Introducción

World Wide Web es un armazón arquitectónico para acceder a documentos vinculados distribuidos en miles de máquinas de toda Internet. Su enorme popularidad se deriva de su interfaz gráfica atractiva que es fácil de usar y proporciona un enorme cúmulo de de información sobre casi cualquier tema.

1.1.1 Inicios de la Web

La Web, comenzó en 1989 en el CERN, el Centro Europeo de Investigación Nuclear. El Web surgió de la necesidad de lograr que los científicos del CERN dispersos internacionalmente colaboren usando un conjunto siempre cambiante de informes, planos, dibujos, fotos y otros documentos.

El primer prototipo basado en texto, estaba en operación 18 meses después. En diciembre de 1991 se hizo una demostración pública en la conferencia Hypertext'91 en San Antonio Texas. Esta demostración llevó a Marc Andreessen de la Universidad de Illinois a desarrollar el primer navegador gráfico, Mosaic y años más tarde desarrollo formó Netscape Communication Corp., para desarrollar clientes, servidores y otro tipo de software Web.

Desde 1996 a 1998 Netscape Navigator y Microsoft Internet Explorer sostuvieron una guerra de navegadores, cada uno tratando frenéticamente de agregar más características (y por ende más errores) que el otro. En 1998, America Online compró Netscape Communication Corp. Por 4,200 millones de dólares, terminando con la breve vida de Netscape como compañía independiente.

En 1994, el CERN y el MIT firmaron un acuerdo para establecer World Wide Web Consortium (W3C), una organización dedicada al desarrollo de Web, la estandarización de protocolos y el fomento de interoperabilidad entre sitios. Desde entonces, cientos de universidades y compañías se han unido al consorcio. La página de inicio del consorcio puede encontrarse en http://www.w3.org.

1.1.2 Panorama de la Arquitectura

Desde el punto de vista del usuario, Web consiste en un enorme conjunto de documentos con vínculos a otras páginas que se ven mediante un navegador que obtiene la página solicitada, interpreta el texto y los comandos de formateo que contienen, y despliega la página, adecuadamente formateada, en la pantalla.

1.2 El Cliente

Un navegador, es un programa que puede desplegar una página Web y atrapar los clics que se hacen en los elementos de la página desplegada. Cuando se selecciona un elemento, el navegador sigue el hipervínculo y obtiene la página seleccionada. Las páginas se nombran utilizando URLs (Localizadores Uniformes de Recursos). Un típico ejemplo sería:

http://www.sitioweb.com/algunapágina.html

1.2.1 Pasos para acceder a una URL

Cuando un usuario hace clic en un hipervínculo o teclea una dirección para acceder a ella, el navegador lleva acabo una serie de pasos para obtener la página que se ha solicitado. Suponga que un usuario que esta navegando desea acceder a la página http://www.aplicaionestelematicas.edu.ni/tema3.html, los pasos ha seguir por el navegador serían:

1. El navegador determina el URL (enviando lo que se seleccionó).
2. El navegador pide al DNS la direccion ip de www.aplicaionestelematicas.edu.ni, que es el nombre de la máquina donde se localiza la página.
3. El DNS le responde con la dirección de la página, (ejemplo 156.123.45.89).
4. El conexión realiza una conexión TCP con el puerto 80 en 156.123.45.89
5. Después envía un mensaje en que solicita el archivo /tema3.html.
6. El servidor www.aplicaionestelematicas.edu.ni envía el archivo tema3.html.
7. Se libera la conexión TCP.
8. El navegador despliega todo el texto de /tema3.html.
9. El navegador obtiene y despliega todas las imágenes del archivo.

Muchos de los navegadores despliegan una línea de estado, de esta manera cuando el desempeño es pobre, el usuario puede ver si se debe a que el DNS o el servidor no están respondiendo, o simplemente hay congestión en la red.

Para permitir que todos los navegadores entiendan todas las páginas Web, éstas se describen en un lenguaje estandarizado llamado HTML el cual las describe.

1.2.2 Visualización de páginas de un tipo MIME

No todas las páginas contienen HTML. Una página puede consistir en un texto en formato PDF, un icono con formato GIF, una fotografía con formato JPEG, una canción con formato MP3 o cualquier otro formato existente. Puesto que un navegador puede vincular cualquiera de estos, el navegador tiene un problema cuando encuentra una página que no puede interpretar.

Cuando un servidor regresa una página, también regresa alguna información adicional acerca de ella. Dicha información incluye el tipo MIME.

Las páginas del tipo text/html se despliegan de manera directa, como las páginas de algunos otros tipos integrados. Si el tipo MIME no es de los integrados, el navegador consulta su tabla de tipos MIME que le indica como desplegar la página. Esta tabla asocia un tipo MIME con un visor. Hay dos posibilidades: pug-ins y aplicaciones auxiliares.

Plug-ins

Un plug-ins (conector) es un módulo de código que el navegador obtiene de un directorio especial en el disco y lo instala como una extensión de sí mismo. Debido a que los plug-ins se ejecutan dentro del navegador, tiene acceso a la página actual y pueden modificar su apariencia.

Cada navegador tiene un conjunto de procedimientos que todos los plug-ins tienen que implementar a fin de que el navegador pueda llamarlos. Los procedimientos típicos de la interfaz del navegador son para asigna y liberar memoria, desplegar un mensaje en la linea de estado del navegador y consultar al navegador sobre los parámetros.

Aplicaciones Auxiliares

Estas son programas completos que se ejecutan como un proceso independiente. Por lo general acepta el nombre de un archivo de trabajo en el que se ha almacenado el archivo de contenido, abre dicho archivo y despliega su contenido. Las aplicaciones auxiliares son independientes del navegador como Acrobat Reader de Adobe o Microsoft Word. Algunos programas tienen un plug-in que invoca a la aplicación auxliar misma.

Muchas aplicaciones auxiliares utilizan el tipo MIME/aplicación (por ejemplo aplicación/pdf, aplicación/msword,etc.). Lo cual indica al navegador, que aplicación desplegar cuando se hace clic sobre el archivo.

Las aplicaciones auxiliares no están restringidas a utilizar el tipo MIME aplicación. Por ejemplo, Adobe Photoshop utiliza imagen/x-photoshop y RealOne Player tiene la capacidad de manejar audio/mp3.

1.2.3 Problemas de Seguridad

Cuando Internet Explorer obtiene un archivo con la extensión exe, sabe que este archivo es un programa ejecutable y, por lo tanto, no tiene una aplicación auxiliar. La acción obvia es ejecutar el programa. Sin embargo, esto podría ser un enorme hoyo de seguridad.

Para evitar huéspedes no deseados como éstos, es posible configurar Internet Explorer para que sea selectivo al ejecutar automáticamente programas desconocidos, pero no todos los usuarios saben como manejar la configuración. En UNIX puede ocurrir algo análogo con las secuencias de comandos de shell, pero eso requiere que el usuario instale de manera conciente el shell como una aplicación auxiliar.

1.3 El servidor

Como vimos anteriormente, cuando el usuario teclea un URL o hace clic en una línea de hipertexto, el navegador lo analiza, e interpreta la parte entre http:// y la siguiente diagonal como un nombre DNS a buscar. Una vez que el navegador obtiene la dirección IP del servidor, establece la conexión TCP con el puerto 80 de ese servidor.

Los pasos que da el servidor en su ciclo principal:

1. Acepta la conexión TCP de un cliente (navegador).
2. Obtiene el nombre del archivo solicitado.
3. Obtiene el archivo (del disco).
4. Regresa el archivo al cliente.
5. Libera la conexión TCP.

Los servidores actuales tienen más características, pero en esencia, esto es lo que hacen. El problema de este diseño es que cada solicitud, requiere un acceso al disco para obtener el archivo. El resultado es que el servidor Web no puede atender más solicitudes por segundo que el número de accesos a disco.

En disco SCSI de alta calidad tiene un tiempo de acceso promedio de aproximadamente 5 mseg, lo que limita al servidor a lo mucho a 200 solicitudes/seg, o menos si se tienen que leer con frecuencia archivos grandes. Para un sitio Web grande esta cifra es muy baja.

1.3.1 Caché en Servidores

Una mejora obvia utilizada por los servidores Web es mantener un caché en la memoria de los n archivos más recientes utilizados. Antes de ir al disco para obtener un archivo, el servidor verifica el caché. Si el archivo está ahí, se puede proporcionar directamente desde memoria con lo que se elimina el acceso a disco. Aunque el almacenamiento en caché es efectivo, requiere una gran cantidad de memoria principal y tiempo de procesamiento extra para verificar el caché y manejar su contenido, el ahorro de tiempo justifica la sobrecarga y costo implícitos.

1.3.2 Módulo Front End

El siguiente paso para construir un servidor más rápido, es hacerlo de múltiples subprocesos. En un diseño, el servidor consiste en un módulo de *front end* que acepta todas las solicitudes entrantes y k módulos de procesamiento. Los k+1 subprocesos pertenecen al mismo proceso por lo que todos los módulos de procesamiento tienen acceso al caché dentro del espacio de direcciones del proceso.

Cuando llega una solicitud sucede lo siguiente:

1. El *front end* la acepta y construye un registro corto que la describe.
2. Entrega el registro a uno de los módulos de procesamiento. En otro diseño posible, se elimina el *front end* y cada módulo de procesamiento, trata de adquirir sus propias solicitudes, pero se necesita un protocolo de bloqueos para evitar conflictos.
3. El módulo de procesamiento primero verifica el caché para ver si el archivo está ahí.
4. De ser así, actualiza el registro para incluir un apuntador al archivo del registro. Si no esta ahí, el módulo de procesamiento inicia una operación de disco para leerlo en el caché.
5. Cuando se ha encontrado el archivo (sea en caché o sea que se haya leído del disco) se regresa al cliente.

Ventajas de este esquema

La ventaja de este esquema es que mientras que uno o más módulos de procesamiento están bloqueados esperando a que termine una operación de disco, otros módulos pueden estar trabajando activamente en otras solicitudes. Por supuesto, para obtener cualquier mejora real sobre el modelo de un solo subproceso, es necesario tener múltiples discos, a fin de que más de un disco pueda estar ocupado al mismo tiempo. Con k módulos de procesamiento y k discos, la velocidad real de transporte puede ser k veces mayor que con el servidor de un solo subproceso y un disco.

1.3.3 Granjas de Servidores

Si llegan demasiadas solicitudes cada segundo, la CPU no será capaz de manejar la carga de procesamiento, sin importar cuantos discos se utilicen en paralelo. La solución es agregar más servidores, posiblemente con discos replicados para evitar que los discos se vuelvan el siguiente cuello de botella. Esto es lo que se conoce como modelo de granja de servidores. Un *front end* aún acepta solicitudes entrantes pero las distribuye en múltiples CPUs en lugar de en múltiples subprocesos para reducir la carga en esa computadora. Las máquinas individuales podrían contar con múltiples subprocesos y canales como en el esquema anterior.

Un problema con las granjas de servidores es que no hay caché compartido debido a que cada nodo de procesamiento tiene su propia máquina (a menos que se utilice un multiprocesador de memoria compartida). Una forma de remediar esta

perdida de rendimiento es hacer que el *front end* registre el lugar a donde envía cada respuesta y que envíe las solicitudes subsecuentes de la misma página al mismo nodo haciendo que cada nodo se vuelva un especialista en ciertas páginas y evitando el desperdicio de caché al tener cada archivo en cada caché.

Otro problema con las granjas de servidores es que la conexión TCP del cliente termine en el *front end.* Para solucionar este problema, algunas veces ese utiliza un truco que se conoce como transferencia TCP (TCP handoff). Con ésta, el punto final de la conexión TCP se pasa al nodo de procesamiento a fin de que pueda contestar de una forma directa y transparente al cliente.

1.3.4 Funciones de los servidores Web modernos

Los servidores Web modernos hacen más que sólo aceptar y regresar nombres de archivos. El *front end* pasa cada solicitud al primer módulo disponible, que después la transporta mediante algunos de los siguientes pasos, dependiendo de los que sean necesarios.

1. Resuelve el nombre de la página Web solicitada.
2. Autentica al cliente (En las páginas que no son para público en general).
3. Realiza control de acceso en el cliente (Verificar si hay restricciones para ese cliente).
4. Realiza control de acceso en la página Web.
5. Verifica el caché.
6. Obtiene del disco la página solicitada.
7. Determina el tipo MIME que se incluirá en la respuesta.
8. Se encarga de diversos detalles.
9. Regresa la respuesta al cliente.
10. Realiza una entrada en el registro del servidor.

1.4 Las URL (Localizadores Uniformes de Recursos)

Cuando se creó Internet, de inmediato fue evidente que para que una página apuntara a otra, se requerían mecanismos para nombrarlas y localizarlas. Había tres preguntas que resolver:

- ¿Cómo se llama la página?

- ¿Dónde está la página?
- ¿Cómo se puede acceder a la página?

La solución escogida identifica a las páginas de manera que resuelve los tres problemas a la vez. A cada página se le asigna un URL (Localizador Uniforme de Recurso) que sirve efectivamente como nombre mundial de la página.

1.4.1 Partes de un URL

Los URLs tienen tres partes que son: el Protocolo, el nombre DNS de la máquina donde reside la página y el nombre local de la página dentro del servidor. Consideremos el siguiente ejemplo:

http://www.aplicacionestelematicas.edu.ni/Tema4/index.html

En una URL, el protocolo es todo aquello que va antes de **://,** en el ejemplo anterior, el protocolo empleado sería http (Protocolo de Transferencia de Hipertexto), sin embargo aunque este es el más común no es el único que existe. La Web soporta diferentes tipos de protocolos los cuales se citan en la tabla siguiente:

Nombre	Usado para	Ejemplo
http	Hipertexto (HTML)	http://www.yahoo.es
ftp	Transferencia de ficheros	ftp://ftp.yahoo.es
File	Archivo local	file:///usr/misprog/prog1.c
News	Grupo de noticias	news:comp.os.minix
News	Artículo	news:AA0123@yahoo.es
Gopher	Gopher	gopher://gopher.tc.umm.edu
Malito	Envío de correo electrónico	mailto:denisjev@yahoo.es
telnet	Inicio de sesión remota	telnet://www.w3.org:80

Figura 3.1 Protocolos soportados por las páginas Web.

A continuación del protocolo, en una URL encontramos el nombre DNS del servidor Web en donde reside la página que queremos abrir. Para el ejemplo anterior seria www.aplicacionestelematicas.edu.ni. Esto nos lleva hasta el servidor web de aplicaciones telemáticas, ahora es necesario indicar que página queremos ver dentro

de este servidor. En nuestro caso, querríamos acceder a la página index.html del directorio Tema4 dentro del servidor.

Si después del nombre del servidor no se indica el nombre de la página que se desea abrir, el navegador buscará por defecto el archivo index.html. Con lo anterior las dos direcciones siguientes serían análogas:

http://www.aplicacionestelematicas.edu.ni/index.html =
http://ww.aplicacionestelematicas.edu.ni

1.4.2 Evolución de las URLs

El problema de las URL es que apuntan a un host específico. En el caso de páginas a las que se hace constante referencia, seria deseable tener varias copias muy distantes, para reducir el tráfico de la red. El problema es que los URL no proporcionan alguna manera de referirse a una página sin decir donde está.

Para resolver este problema y hacer posible la duplicación de páginas, la IETF está trabajando en un sistema URN (Nombre Universal de Recursos). Un URN puede considerarse como un URL generalizado. Este tema aún es el objetivo de la investigación.

1.5 Los Cookies

Como henos visto la Web no tiene estado. No existe el inicio de sesión. El navegador envía una solicitud a un servidor y obtiene un archivo. A continuación el servidor, olvida que ha visto a ese cliente en particular. Esto comenzó a ser un problema cuando surdió la necesidad de registrar clientes, comercio electrónico, páginas de inicio con la información que el usuario desea, etc.

A primera vista pareciera que esto se podría resolver a través de la dirección IP del usuario, pero el problema de la utilización de máquinas compartidas por varias personas o el empleo de NAT por muchos ISPs no lo permitía.

Para resolver este problema, Netscape diseño una técnica muy criticada llamada cookies la cual se formalizó posteriormente en la RFC 2019. Los cookies son similares a un descriptor de archivo en UNIX o un identificador de objetos en Windows.

1.5.1 Descripción de los Cookies

Cuando un cliente solicita una página Web, el servidor puede proporcionar información adicional junto con la página solicitada. Esta información puede incluir una cookie, que es un pequeño archivo o cadena de a lo mucho 4 KB. Los navegadores almacenan los cookies ofrecidas en un directorio de cookies en el disco duro de la máquina a menos que el usuario los haya deshabilitado.

Los cookies son simplemente archivos o cadenas, no programas ejecutables. En principio, una cookies puede contener un virus, pero puesto que las cookies se tratan como datos y no como programas, no hay forma oficial en que los virus se ejecuten y hagan daño.

1.5.2 Campos de los cookies

Un cookies puede contener hasta 5 campos que son:

- **Dominio:** Indica de donde viene el cookie, se supone que los navegadores verifican que los servidores no mientan acerca de su dominio. Cada dominio puede almacenar hasta 20 cookies por cliente.

- **Ruta:** Es la ubicación en la estructura de directorios del servidor que identifica que parte del árbol de archivos del servidor podrían utilizar la cookie. Por lo general es /, lo que significa el árbol completo.

- **Contenido:** Toma la forma nombre=valor. Tanto nombre como valor pueden ser lo que el servidor desee. Este campo es donde se almacena el contenido del cookie.

- **Expira:** Especifica cuando caduca la cookie. Si este campo esta ausente, el navegador descarta la cookie cuando sale. Tal cookie se conoce como cookie no persistente. Si se proporciona una hora y una fecha, se dice que la cookie es persistente y se mantiene hasta que expira. Para eliminar una cookie del disco duro de un cliente, un servidor simplemente le envía nuevamente, pero con una fecha caducada.

- **Seguro:** Puede establecerse para indicar que el navegador podría simplemente regresar la cookie a un servidor seguro. Esta característica se utiliza para comercio electrónico, actividades bancarias y otras aplicaciones seguras.

Las cookies también pueden utilizarse para el beneficio del servidor. Por ejemplo, suponga que un servidor desea llevar el registro de cuántos visitantes únicos ha tenido y cuantas páginas miró cada uno antes de dejar el sitio. Cuando llega la primera solicitud, no hay cookie acompañante, por lo que el servidor regresa una cookie que contiene *Counter=1*. Los clics subsecuentes en ese sitio regresarán la cookie al servidor. Cada vez el contador se incrementa y se regresa al cliente. Al llevar un registro de los contadores, el servidor puede ver cuántas personas salieron del sitio después de ver la primera página, cuantos vieron dos y así sucesivamente.

1.5.3 Problemas de Seguridad en los Cookies

Los cookies también han sido objeto de malos uso ya que los piratas informáticos han aprovechado varios errores de los navegadores para capturar cookies que no son de ellos, en especial, desde que algunos sitios de comercio electrónico colocaron números de tarjetas de créditos en las cookies.

Un uso controversial de las cookies es colectar de manera secreta información sobre los hábitos de navegación de un usuario. Algunos usuarios configuran sus navegadores para que rechacen las cookies. Sin embargo esto puede darles problemas con algunos sitios Web que legítimos que utiliza las cookies.

Para resolver este problema, algunas veces los usuarios instalan software que elimina cookies. Estos son programas especiales que inspeccionan cada cookie entrante al momento del arribo y la aceptan o descartan, dependiendo de las opciones que el usuario haya establecido. Esto le da al usuario un control detallado sobre cuales cookies se aceptan y cuales se rechazan. Los navegadores modernos como Mozilla han elaborado controles de usuario sobre cookies integradas.

2. Documentos Web Estáticos.

La base de Web es la transferencia de páginas Web desde el servidor al cliente. En la forma más simple, las páginas Web son estáticas, es decir, son simplemente archivos que se encuentran en algún servidor esperando a ser recuperados. En este sentido, incluso un video es una página Web estática.

2.1 Qué es HTML

HTML es un lenguaje de marcado que sirve para describir cómo se van a formatear los documentos. Los comando de marcado dentro de HTML y su estandarización, han posible que cualquier navegador Web lea y reformatee cualquier página Web escrita por un usuario.

2.1.1 Etiquetas de HTML

Una página Web se construye utilizando las etiquetas de HTML. Las etiquetas pueden escribirse tanto en mayúsculas como en minúsculas. Por tanto <head> y <HEAD>, significan la misma cosa, pero las nuevas versiones estandar

Toda página Web incluye un encabezado y un cuerpo encerrado entre etiquetas (comandos de formato) <html> </html>, aunque la mayoría de los navegadores no se quejan si faltan estas etiquetas. El encabezado se encuentra delimitado por las etiquetas <head> y </head>, y el cuerpo, por las etiquetas <body> y </body>. Las etiquetas dentro de HTML por lo general la forma <algo> para marcar el comienzo de algo y </algo>, para indicar el final. La mayoría de los navegadores tienen una opción *source view* para ver el código fuente de la página actual.

Etiqueta	Descripción
<html>….</html>	Declara que la página Web está escrita en HTML.
<head>….</head>	Delimita el encabezado de la página Web.
<title>….</title>	Delimita el título (Se ve en la barra de título del navegador).
<body>….</body>	Delimita el cuerpo de la página.
<h1>, <h2>…h6>	Delimita un encabezado de nivel n.

….	Pone …. en negritas.
….	Pone …. en cursivas.
….	Centra …. en la página.
….	Corchetes de una lista desordenada (con viñetas).
….	Corchetes de una lista numerada.
….	Corchetes de un elemento de una lista ordenada o numerada.
 	Obliga salto de línea aquí.
<p>	Inicia un párrafo.
<hr>	Inserta una regla horizontal.
<img src=”…”>	Carga una imagen aquí.
<a ref.=”…”>….	Define un hipervínculo.

Figura 3.2 Etiquetas del lenguaje HTML más utilizadas.

2.1.2 Ejemplo de una página en HTML

Con la lista de etiquetas de la tabla anterior y con un editor de texto sin formato como el bloc de notas de Windows o el vim de Linux podemos crear la página de inicio de la asignatura Aplicaciones Telemáticas.

La página Web que queremos diseñar es la de la Asignatura de Aplicaciones Telemáticas la cual contendrá textos, imágenes, hipervínculos, títulos, listas y viñetas.

Código Fuente en HTML

Para generar la página Web anterior lo hacemos con el siguiente código HTML.

```
<html>
<head><title> Aplicaciones Telemáticas </title></head>
<body>
    <h1><center> Bienvenidos a la página de inicio de <br>Aplicaciones Telemáticas</center>
    </h1><br> <center><img src="logo.jpg" alt=logoAT"></center><br>
    <p>Esta es la página de inicio de <b>Aplicaciones Telemáticas</b>, aquí podrás encontrar toda
    la información necesaria para esta asignatura. También encontradas <i>enlaces</i> que te
    serán de mucha ayuda.
    <p>A continuación, se presentan algunos enlaces que te servirán de mucha ayuda para obtener
    información, registrarte como un usuario de esta página, teléfonos de contacto y dirección de
    correo electrónico para que puedas contactar conmigo.
```

```
<hr>
<h2>Servicios al alumno</h2>
        <ul>
                <li><a href="./registro.html">Crear una cuenta de alumno</a></li>
                <li><a href="./temario.html">Temario de la Asignatura</a></li>
        </ul>
<h2>Información de Contacto</h2>
        <ul>
                <li>Teléfono: 505 311 5519</li>
                <li>E-mail: denisjev@yahoo.es</li>
        </ul>
</body>
</html>
```

Figura 3.3 Código HTML de la página principal de la asignatura.

Presentación en el Navegador

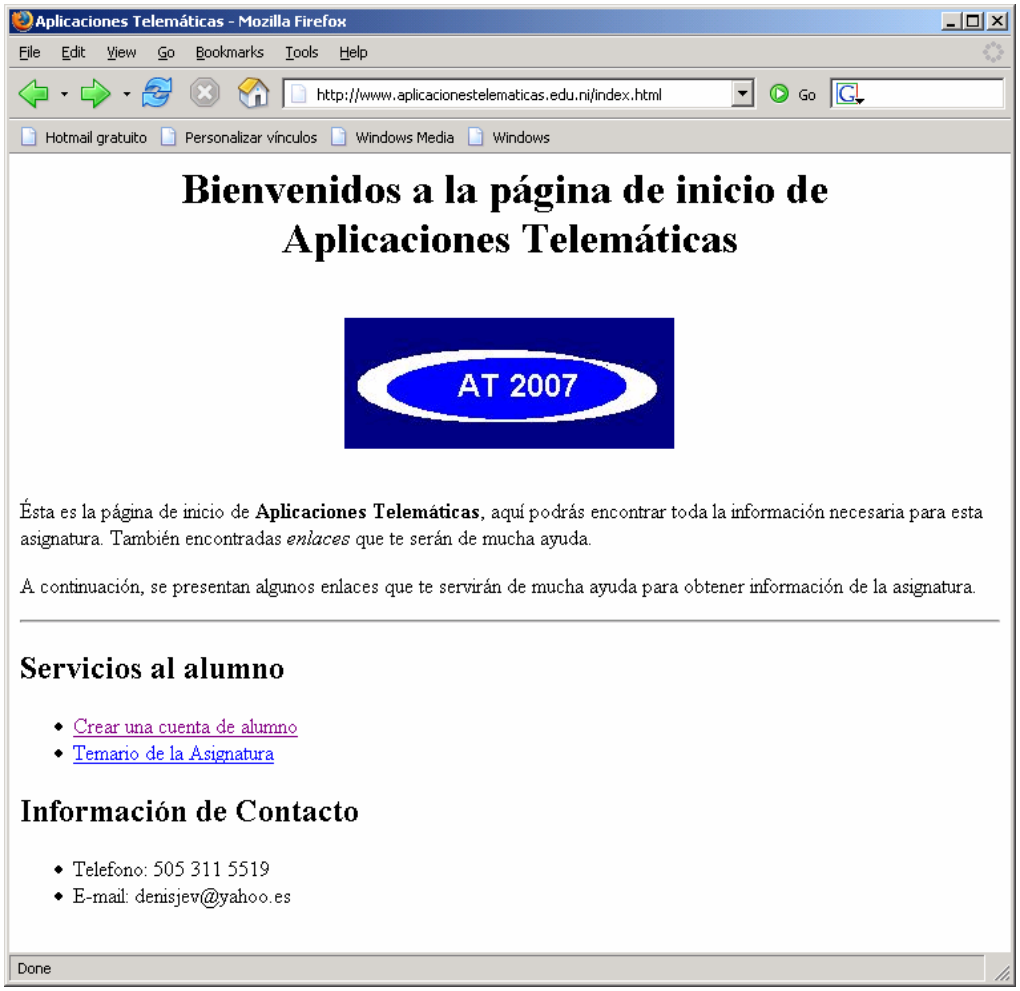


Figura 3.4 Página principal de la asignatura

2.2 HTML 2.0

2.2.1 Nuevas necesidades

HTML 1.0 básicamente era de un solo sentido, es decir, los usuarios podrían bajar información desde algún sitio pero no podían enviársela. Estas demandas condujeron a la inclusión de formularios a partir de HTML 2.0 utilizando la etiqueta <input> para representar las diferentes etiquetas, determinar el tamaño, la naturaleza y el uso del cuadro presentado.

2.2.2 Etiquetas para la Creación de Formularios

Un formulario se encierra entre las etiquetas <form> y </form>. El texto que no se encuentre delimitado por las etiquetas, simplemente se despliega. Todos los elementos son insertados en el formulario empleando la misma etiqueta (<input>), con diferencia en los parámetros. Los principales elementos que peden ser insertados con la etiqueta input son:

- **Cuadro de texto:** Se utilizan para que el usuario pueda introducir un texto.
- **Botones de Opción:** Estos se usan cuando debe tomarse una decisión entre varias alternativas. Los botones de radio se agrupan colocando el mismo valor en el campo name.
- **Los chechbox:** Se emplean cuando para presentar varias opciones dentro de las cuales se pueden seleccionar varias a la vez.
- **Botones:** Sirven para realizar una acción cuando el usuario realiza un clic sobre ellos.

Para dejar más clara la creación de formularios vamos a realizar un formulario sencillo en el cual vamos a permitir el registro de nuevos alumnos a nuestro sitio de Web de Aplicaciones Telemáticas.

Código Fuente en HTML

```
<html>
<head><title>Registro de Alumnos</title></head>
<body>
<h1><center><font color="#0033FF">Página de Registro de Alumnos para el curso de</font></center></h1>
<h1 align="center"> <font color="#0000CC">AT-2007</font></h1>
```

```
<form action="http://aplicacionestelematicas.edu.ni" method="post">
        <p>Nombre:<input name="nombre" type="text" size=100>
        <p>Dirección: <input name="dir" type="text" size="98"></p>
        <p>E-mail: <input name="email" type="text" size="101"></p>
        Sexo: <p>Masculino: <input name="sexo" type="radio" value="masculino">
        Femenino: <input name="sexo" type="radio" value="femenino"> </p>
        <p>Login de usuario: <input name="login" type="text" size=40></p>
        <p>Clave de usuario: <input name="clave" type="password" size=40></p>
<p>Enviar mensajes sobre nuevos eventos <input name="men" type="checkbox"></p>
<p>Enviar información de la asignatura <input name="info" type="checkbox"></p>
<p><input name="enviar" type="submit" value="Enviar Datos">
<input name="limpiar" type="button" value="Limpiar">
<input name="cancelar" type="button" value="Cancelar Envio"></p>
</form>
</body>
</html>
```

Figura 3.5 Código HTML para la creación de un formulario de registro .

Presentación en el Navegador

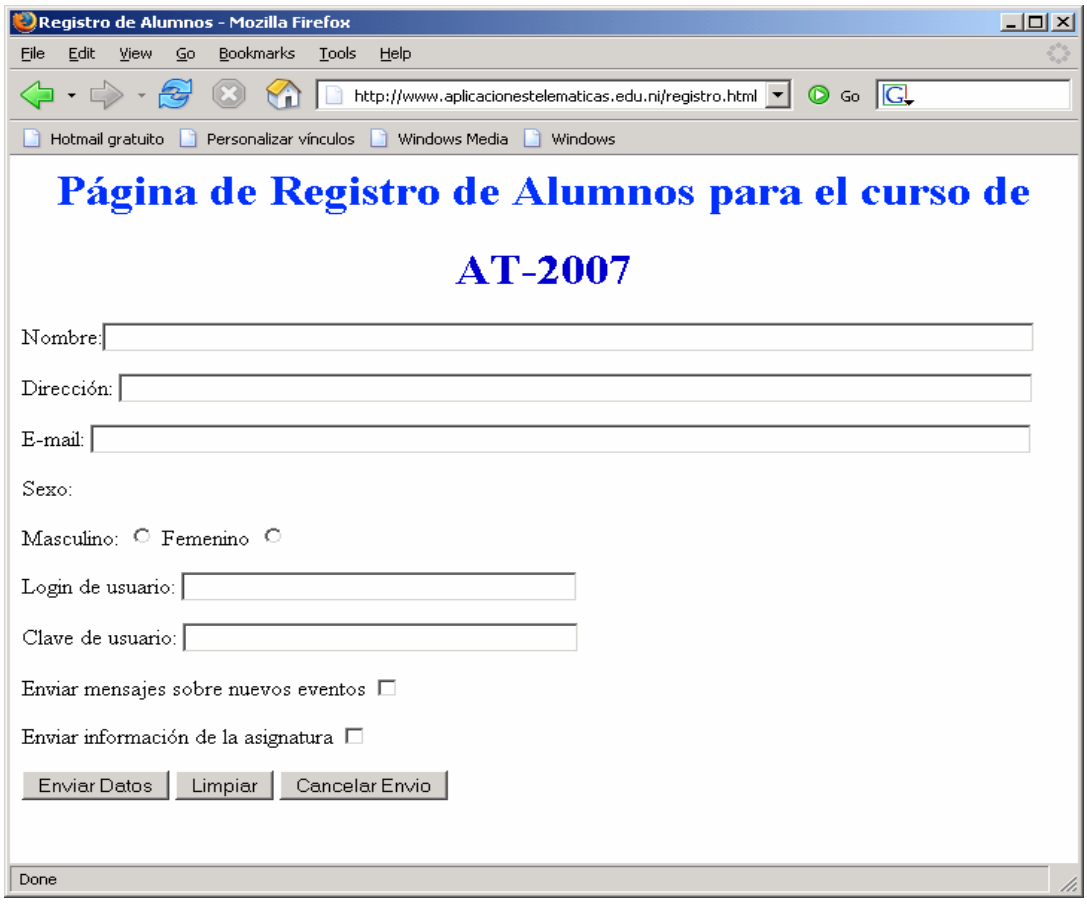


Figura 3.6 Formulario de registro de los alumnos de la asignatura.

Comentaremos brevemente los elementos más importantes para la realización del formulario:

- Todo formulario debe ir encerrado entre las etiquetas <form> y </form>. Dentro de la etiqueta <form> se coloca el parámetro **action** que indica el sitio web con el cual interacciona el formulario. El parámetro **method=POST**, indica que los datos del formulario serán enviados al sitio Web indicados en action con una cadena como la siguiente:

nombre=Denis+Espinoza&dir=Enel+2+cuadras+arriba+75+vrs+al+sur&email=denisjev@yahoo.es&sexo=masculino&login=denis&clave=denis123&men=on.

- Cada línea nueva se antecede con la etiqueta <p> para que se genere una línea nueva dentro del formulario.

- Para insertar un cuadro de texto dentro del formulario como habíamos citado anteriormente utilizamos la etiqueta inptu. El parámetro **name** sirve para asignar al cuadro un nombre por medio del cual podamos luego referirnos, el parámetro **type** sirve para indicar el tipo del elemento, si este parámetro se omite, por defecto es tipo text. El parámetro **size** sirve para establecer el tamaño del cuadro de texto.
- Para insertar botones de radio los cuales pertenezcan a un mismo grupo basta que el parámetro **name** tenga el mismo valor. El parámetro **type** debe de tener el valor radio. En este tipo de elementos, se debe de añadir el campo **value** ya que este es el texto que ha de aparecer al lado del boton.

- Para crear un cuadro de dialogo para una contraseña, se trata como una caja de texto modificando el parámetro **type** por el valor password.

- En el caso de los chckbox solo hay que colocar el campo **type** a checkbox.

- A la hora de insertar los botones tenemos dos tipos. El primero es el **buton,** es un tipo de botón como el empleado en los diferentes lenguajes de programación con interfaces gráficas al cual hay que añadirle algún evento. El otro tipo es **sumit**, este se encarga de recolectar en una línea la información recolectada del formulario y enviarla al servidor descrito en el campo **action** de

la **form**. El & se utiliza para separar campos y + para representar los espacios en blanco. Para el ejemplo de nuestro formulario, si un usuario introdujo los datos siguientes.

Toda esta línea se le enviaría al servidor y es este el que se encargará de interpretar y realizar las operaciones necesarias con esta cadena.

2.3 HTML 3.0

HTML 1.0 y HTML 2.0 no tenían tablas, pero éstas se agregaron en HTML 3.0. Estas pueden contener una gama amplia de material, como texto, figuras, iconos, fotografías e incluso otras tablas. Las tablas se inician mediante la etiqueta <**table**>. Es posible proporcionar información adicional para describir las propiedades generales de la tabla. La etiqueta <**caption**> puede proporcionar el título de una figura.

Cada fila comienza con una etiqueta <**tr**> que sirve para indicar una fila de la tabla. Las celdas individuales se marcan como <**th**> cuando se refiere a los encabezados de la tabla o <**td**> cuando se refiere a los datos de esta tabla. Un elemento de una tabla, puede ser cualquier cosa como se indicó en el caso anterior.

En las tablas también se permiten varios atributos. Incluyen formas de especificar alineaciones de celdas horizontales y verticales, texto justificado dentro de una celda, bordes, grupos de celdas, unidades y más. A continuación, presentaremos un ejemplo para la creación de tablas.

Código HTML

```
<html>
<head><title>Temario de la Asignatura</title></head>
<body>
<h1><center><font color="#0033FF">Temario de la Asignatura AT-2007 </font></center></h1><br>
<p align="justify">En esta sección, se colocará toda la documentación  de la parte teoría y práctica de la asignatura. Se invita al alumno a que este pendiente de la información que se colgará en este sitio Web. Si usted ya se ha inscrito a la página Web, recibirá un correo indicando las actualizaciones que se lleven a cabo.<br>
<center><table width=500 border=1 >
<caption><b><font size=3 face="sans-serif" color="#990000"> Documentación Disponible
<br></font></b></caption>
<tr bgcolor="#3366CC">
```

```
<th width="200"><font face="sans-serif" color="#FFFFFF"> Tema</font>
<th width="200"><font face="sans-serif" color="#FFFFFF">Teoría</font>
<th width="200"><font face="sans-serif" color="#FFFFFF">Práctica</font></tr>
<tr> <td align="left">Aplicaciones Clásicas <td>Descargar <td>Próximamente </tr>
<tr> <td align="left">Servicios de Directorios <td>Descargar <td>Próximamente </tr>
<td align="left">El Correo Electrónico <td>Descargar <td>Próximamente </tr>
<td align="left">El World Wide Web <td>Descargar <td>Proximamente </tr>
<td align="left">Voz sobre IP <td>  <td>Proximamente </tr>
</table></center>
</body>
</html>
```

Figura 3.7 Código HTML para la generación de una tabla.

Presentación en el Navegador

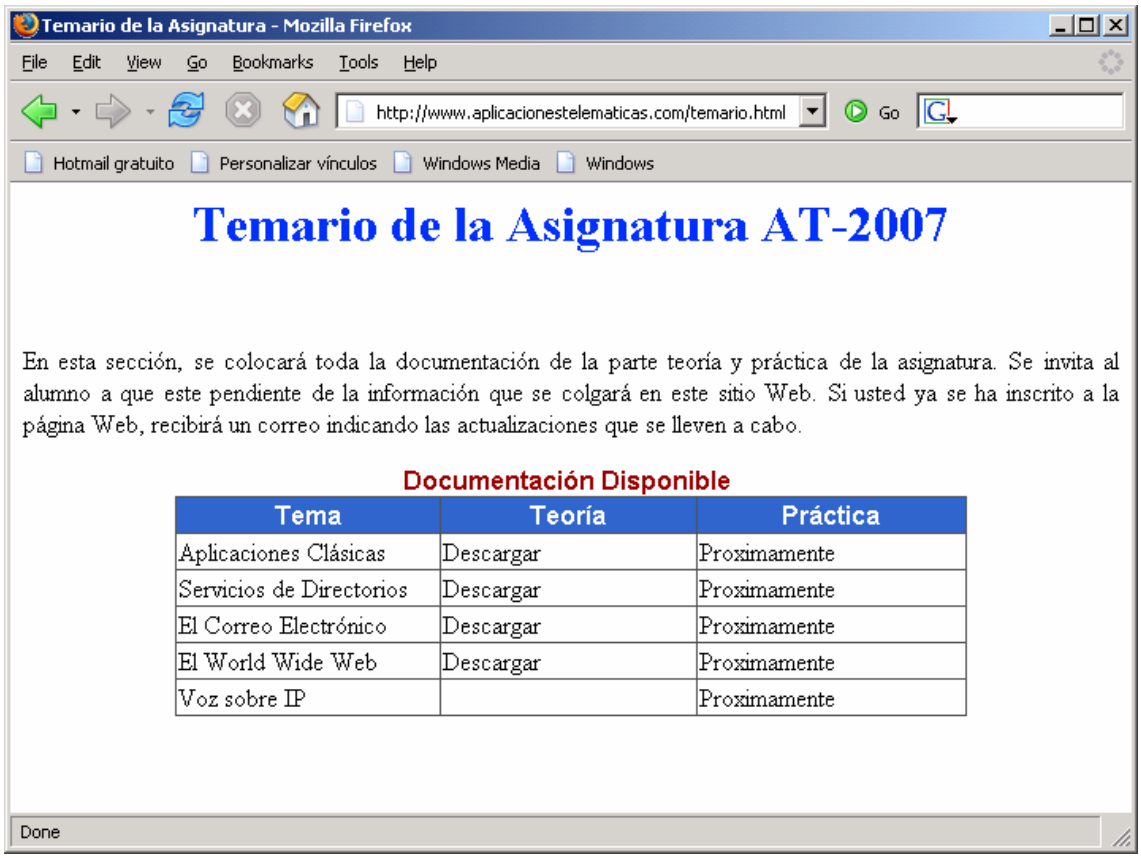


Figura 3.8 Tabla de la documentación de la asignatura.

- Cada tabla siempre se crea entre las sentencias <**table**> y <**/table**>. La etiqueta *table* tiene diversos parámetros entre los cuales tenemos **width**, que nos permite establecer el tamaño de la tabla o el parámetro **border** que nos permite establecer el tamaño de los bordes de dicha tabla.

- Dentro de una tabla, podemos ocupar la etiqueta **caption** para establecer el nombre de la tabla.

- Para crear cada una de las filas de la tabla se utilizan las etiquetas <**tr**> y </**tr**> de las cuales hemos empleado el parámetro **bgcolor** para cambiar el fondo de la primera fila de la tabla.

- Dentro de las etiquetas <**tr**> se pueden insertar etiquetas <**th**> o <**td**>. Las primeras se utilizan para incluir en la fila en que nos encontramos dentro de una columna específica, un valor en forma de encabezado (con lo cual el dato aparece resaltado), mientras que la segunda nos sirve para insertar un dato cualquiera.

- Por último, cuando se desea dejar una celda vacia, lo único que se debe de hacer es insertar después de la etiqueta <td> que referencia a esa celda .

2.4 HTML 4.0

En HTML 4.0 se agregaron nuevas características. Estas incluyen características de accesibilidad para usuarios discapacitados, incrustación de objetos (una generalización de la etiqueta <img> de manera que otros objetos también puedan ser incrustados en las páginas), soporte para lenguajes de secuencias de comandos (para permitir contenido dinámico), y más.

Cuando un sitio Web es complejo, y consiste en muchas páginas creadas por diversos autores que trabajan para la misma compañía, con frecuencia es deseable tener una forma de evitar que páginas de un mismo sitio Web tengan una apariencia distinta. Este problema se puede resolver utilizando hojas de estilo. Con éstas, las páginas individuales ya no utilizan más estilos físicos, como negritas y cursivas.

En su lugar, los autores de páginas utilizan estilos lógicos como <**dn**> (define), <**em**> (énfasis débil), <**strong**> (énfasis fuerte) y <var> (variables de programas). Los estilos lógicos se definen en la hoja de estilo, que se indica al principio de la página, de esta mánera, si el Web master decide cambiar los colores o letras de la página, solo tendrá que cambiar este fichero.

2.5 Nacimiento de las páginas Web estructuradas

HTML con o sin formularios, no proporciona estructura alguna para las páginas Web. Además, mezcla el contenido con el formato, Conforme el comercio electrónico y otras aplicaciones se vuelven más comunes, hay una necesidad cada vez mayor de dar una estructura a las páginas Web y de separar el contenido del formato. Por ejemplo, un programa que busca en la Web el mejor precio de algún libro o CD necesita analizar muchas páginas Web en busca del título y precio del elemento. Con las páginas Web en HTML, es muy difícil que un programa averigüe en dónde están el título y el precio.

Por esta razón el W3C ha mejorado el HTML para permitir que las páginas Web puedan tener estructura para su procesamiento automatizado. Para este propósito se han desarrollado dos nuevos lenguajes, estos son XML, el cual describe el contenido Web de una forma estructura y el segundo XSL, el cual describe el formato independiente del contenido. A continuación abordaremos cada uno de ellos.

2.5.1 XML

XML, sigla en inglés de eXtensible Markup Language (lenguaje de marcas extensible), desarrollado por W3C. Permite definir la gramática de lenguajes específicos para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

Ventajas de XML

- Es un metalenguaje extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas lo que lo hace compatible con las versiones anteriores.

- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje pudiéndose emplear alguno ya disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.

- Es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

Documentos XML bien formados

Del inglés well formed, son los documentos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, ser analizados correctamente por cualquier "parser" (Analizador Sintáctico) que cumpla con la norma.

- Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Las etiquetas deben estar correctamente anidadas.

- Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, sólo puede tener un elemento inicial.

- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.

- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.

- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos entendibles por las personas.

Estructura de un Documento XML

Un documento XML está formado por el prólogo y por el cuerpo del documento.

Prólogo

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

El prólogo contiene:

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

Cuerpo

A diferencia del prólogo, el cuerpo no es opcional en un documento XML, el cuerpo debe contener al menos un elemento raíz, característica indispensable también para que el documento esté bien formado.

Ya que XML y XSL, están intrínsicamente ligados, explicaremos brevemente XSL y posteriormente se presentará un ejemplo que ilustra el uso de ambos.

2.5.2 XSL

Los documentos XML no tienen formato alguno: son simples contenedores de información estructurada. Una vez conseguido el esquema XML, se puede utilizar el lenguaje XSL (Lenguaje de hojas de estilo ampliable) para dar formato y mostrar la información al cliente. Puede definir estilos, elementos de página, el diseño, etc., en un archivo XSL y adjuntarlo a un archivo XML.

El contenido (los datos XML) y la presentación (definida por el archivo XSL) son totalmente independientes, y proporcionan un mayor control sobre el modo en que aparece la información en una página Web. Esencialmente, XSL es una tecnología de presentación para XML, en la que el resultado principal es una página HTML.

La familia XSL está formada por tres lenguajes que son:

- XSLT (siglas de Extensible Stylesheet Language Transformations, lenguaje de hojas extensibles de transformación), que permite convertir documentos XML de una sintaxis a otra (por ejemplo, de un XML a otro o a un documento HTML).

- XSL-FO (lenguaje de hojas extensibles de formateo de objetos), que permite especificar el formato visual con el cual se quiere presentar un documento XML, es usado principalmente para generar documentos PDF.

- XPath, o XML Path Language, es una sintaxis (no basada en XML) para acceder o referirse a porciones de un documento XML.

Estas tres especificaciones son recomendaciones oficiales del W3C. En el 2005 ya son soportadas por algunos navegadores, por ejemplo mozilla o Internet Explorer, aunque, en su lugar, se pueden usar las CSS que son 100% compatibles aunque con una codificación diferente.

2.5.3 Ejemplo de uso de XML y XSL

Supongamos que queremos desarrollar una página web que deseamos que nos muestre por pantalla, los diferentes comentarios que los alumnos envían a la página Web. Primero crearemos el archivo XML que contendrá la estructura de la información a utilizar información a utilizar

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xls" href="comentarios.xls"?>
<lista_comentarios>
        <comentario>
                <asunto>Pagina Web de la Asignatura</asunto>
                <nombre>Juan Antonio Alvarez</nombre>
                <email>juan@yahoo.es</email>
                <fecha>23/10/2007</fecha>
                <texto_com>Le escribia para preguntarle cuando estara lista la pagina Web de la asignatura para descargar los ficheros de la clase siguiente. Gracias</texto_com>
        </comentario>
        <comentario>
                <asunto>Clase de XML</asunto>
                <nombre>Andres Antonio Mendez</nombre>
```

```
        <email>andres@yahoo.es</email>
        <fecha>22/10/2007</fecha>
        <texto_com>Profesor, ya escribi el archivo XML pero me da unos errores un mensaje emergente que se bloqueo, no si es que no se abre el fichero XSLT. Adios</texto_com>
    </comentario>
    <comentario>
        <asunto>Practicas de Laboratorio</asunto>
        <nombre>Ana Maria Lopez</nombre>
        <email>ana@yahoo.es</email>
        <fecha>22/10/2007</fecha>
        <texto_com>Hola, queria preguntarle si la practica se puede entregar la semana siguiente porque no he podido terminarlas. Paso por su oficina a la hora de turorias.</texto_com>
    </comentario>
</lista_comentarios>
```

Figura 3.9 Código XML para la generación de la página de comentarios.

Una vez que tenemos la toda la información estructurada en el archivo XML, realizaremos el archivo XSL, el cual simplemente se encargará de indicar el formato de la página, de esta manera, cuando queramos agregar más comentarios solamente tendremos que modificar el archivo XML y utilizando la plantilla XSL, se creará el archivo html con todo la información y empleando para toda el mismo formato.

```
<?xml version="1.0" encoding="iso-8859-1"?><!-- DWXMLSource="comentarios.xml" -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="iso-8859-1"/>
<xsl:template match="/">
<html>
<head><title>Comentarios y Sugenrencias de la Asignatura</title></head>
<body><h2><center>Lista de Comentarios enviados por los alumnos</center></h2>
<xsl:for-each select="lista_comentarios/comentario">
<center><table border="2" bgcolor="#0066CC" width="600">
    <tr><th align="justify" width="500"><font face="Arial" color="#FFFFFF">
    <xsl:value-of select="asunto" /></font>
    </th><th align="right"><font face="Arial" color="#FFFFFF">
    <xsl:value-of select="fecha" /></font>
    </th></tr>
    <tr>
        <table border="0" width="600">
            <tr><th align="justify" width="100">Nombre:</th><td>
            <xsl:value-of select="nombre" /></td></tr>
            <tr><th align="justify" width="100">Email:</th><td>
```

```
                        <xsl:value-of select="email" /></td></tr>
                </table>
        </tr>
        <tr>
                <table width="600" border="0">
                        <tr><td align="justify">
                        <xsl:value-of select="texto_com" /></td></tr>
                </table>
        </tr>
        </table></center>
</xsl:for-each>
<br />
<center><marquee bgcolor="#0033CC" width="600" height="12"> <font face="arial"
color="#FFFFFF"><b>¡GRACIAS POR SUS TIEMPO! </b></font></marquee></center>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Figura 3.10 Código XSL para la generación de la página de comentarios.

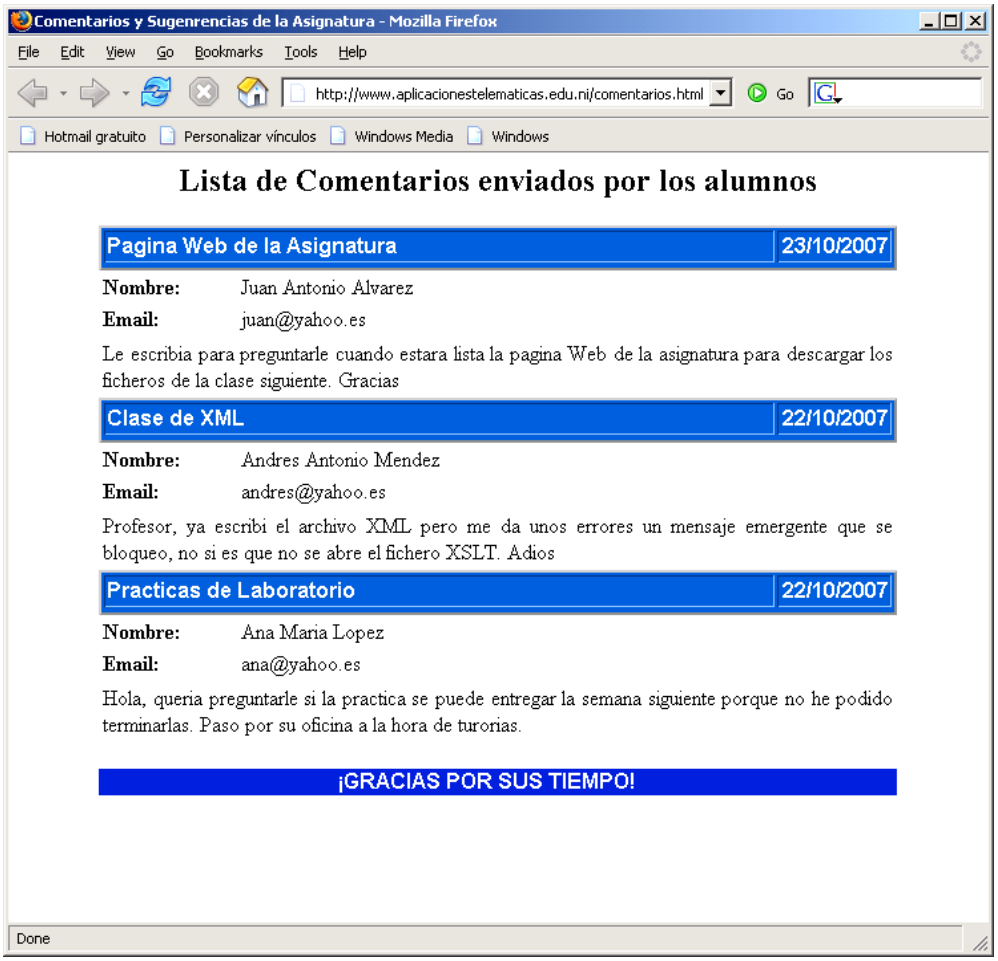


Figura 3.11 Página de comentarios de la asignatura.

Las especificaciones XML y XSL son mucho más estrictas que la especificación HTML. Declaran que es obligatorio rechazar archivos incorrectos sintácticamente. Un navegador que acepta un archivo XML o XSL incorrecto sintácticamente y repara los errores el mismo, no se apega a las especificaciones y se rechazará en una prueba de conformidad, sin embargo este hecho ayuda a tratar con páginas mal diseñadas.

Vale la pena recalcar que las páginas Web en XML y XSL aún son estáticas. Aún no es claro si XSL quitará el lugar a las hojas de estilo tradicionales.

2.6 XHTML

Muchas personas de la industria, sienten que en el futuro, la mayoría de los dispositivos habilitados para Web, no serán PCs, sino, dispositivos de tipo PDA de mano inalámbrico con memoria limitada, llenos de heurística que tratan de alguna manera de lidiar con las páginas Web incorrectas sintácticamente. Por lo tanto, el siguiente paso después de HTML 4 es un lenguaje muy exigente. Se llama XHTML (Lenguaje de Marcado de Hipertexto) y no HTML 5 porque es esencialmente HTML 4 reformulado en XML. Hay seis diferencias principales entre XHTML y HTML 4 las cuales son:

1. Las páginas XHTML y los navegadores deben apegarse estrictamente al estándar, propiedad que se heredo de XML.

2. Todas las etiquetas y los atributos deben estar en minúsculas. Las etiquetas como <HTML> no son válidas en XHTML. De manera similar, <img src=”ima001.jpg”> también está prohibido porque contiene un atributo en mayúscula.

3. Ahora se requiere el uso de etiquetas de cierra, incluso para <p>. Las etiquetas que no tienen cierre natural, como
, <hr> e <img>, deben tener un diagonal antes del paréntesis de cierre “>”, por ejemplo:

4. Los atributos deben de estar encerrados entre comillas. Por ejemplo: ya no se permite. El número 500 debe de estar encerrado entre comillas, de la misma forma que el nombre del archivo JPG, aún cuando sea un número.

5. Las etiquetas deben de estar anidadas de manera apropiada. En el pasado, el anidamiento correcto no era requerido siempre y cuando el estado final fuera el correcto. Por ejemplo: Fotos de vacaciones , era legal. En XHTML no lo es. Las etiquetas deben encerrarse en orden inverso en el que se abrieron.

6. Cada documento debe especificar su tipo de documento, como se vió en el ejemplo del código XML.

Con esto, hemos terminado nuestro estudio de las páginas Web estáticas. Como pudimos notar, con esta tecnología y un poco de creatividad, es posible crear páginas Web muy llamativas. Sin embargo, hoy en día se necesitan páginas Web permitan una interacción con el usuario, con lo cual aparecen las páginas Web dinámicas.

3. Documentos Web Dinámicos

Cada vez más, el contenido se genera de manera dinámica, en lugar de almacenarlo en el disco. Esta generación de contenido puede hacerse ya sea en el lado del cliente o en el del servidor.

3.1 Generación de Páginas Web Dinámicas en el Servidor

Para comprender un poco mejor la generación de código en el lado del servidor considere el uso de formularios los cuales capturan información proporcionada por el usuario para buscar información dentro de una base de datos en el servidor y generar una página HTML personalizada para regresarla al cliente.

3.1.1 Interfaz de Puerta de Enlace Común (CGI)

Es la forma tradicional de manejar formularios y otras páginas Web interactivas. Es una interfaz estandarizada para permitir que los servidores Web hablen con los back-end y las secuencias de comandos puedan aceptar la entrada de datos y generar en respuestas páginas HTML. Por lo general estos back-ends son secuencias de comandos escritos en lenguaje Perl.

Como ejemplo de la forma en que funciona CGI, considere el caso de que un alumno se quiera registrar en nuestra página Web (www.aplicacionestelematicas.edu.ni). Para ello el hipervínculo “Crear una cuenta de alumno” de la página principal, apunta a una secuencia de comando de Perl, digamos www.aplicacionestelematicas.edu.ni/cgi-bin/reg.perl. Cuando esta secuencia de comandos se invoca sin parámetros, regresa una página HTML que contiene el formulario de registro. Cuando el usuario llena el registro y hace clic en el botón de envío, se regresa un mensaje a esta secuencia de comandos que contiene los valores introducidos.

A continuación la secuencia de comandos Perl analiza los parámetros, crea una entrada en la base de datos para el nuevo usuario y regresa una página HTML que confirma que el usuario ha sido agregado. Esta no es la única forma de manejar formularios, pero sí es común.

3.1.2 Preprocesador de Hipertexto (PHP)

Otra forma común es incrustar pequeñas secuencias de comandos dentro de la página HTML y hacer que el servidor mismo sea quien las ejecute para generar la página. Un lenguaje popular para escribir estas secuencias de comandos es PHP.

PHP es especialmente bueno para manejar formularios y es más sencillo que utilizar secuencias de comandos CGI. Como ejemplo de cómo funciona con formularios, considere el código siguiente:

```
<html>
        <body>
        <form action="action.php" method="post">
                <p>Por favor introduzca su nombre: <input type="text" name="name"> </p>
                <p>Por favor introduzca su edad: <input type="text" name="age"></p>
                <input type="submit">
        </form>
        </body>
</html>
```

Figura 3.12 Ejemplo de una página con un formulario que llama a un método PHP.

La página despliega dos cuadros de texto, uno que solicita el nombre y el otro la edad. La primera línea nos indica que se va ha invocar el archivo action.php para manejar los parámetros después de que el usuario ha llenado y emitido el formulario. Una vez que se ha enviado el formulario, el servidor analiza la cadena de tipo regresada, colocando el nombre en la variable *nombre* y la edad en la variable *age*.

Con los datos proporcionados por el usuario, inicia el procesamiento del archivo *action.php* cuyo contenido es el siguiente:

```
<html>
 <body>
        <h1> Respuesta </h1>
        Hola <?php echo $name; ?>
        Predicción: el siguiente año tendrá <?php echo $age + 1; ?>
</body>
</html>
```

Figura 3.13 Código del archivo action.php.

Durante el procesamiento de este archivo, se ejecutan los commandos PHP. Si el usuario introdujo “Bárbara” y “24” en los cuadros, el archivo HTML generado será el siguiente:

```
<html>
        <body>
        <h1> Respuesta </h1>
        Hola Bárbara.
        Predicción: el siguiente año tendrá  25
        </body>
</html>
```

Figura 3.14 Código HTML generado por action.php una vez que se han ejecutado las secuencias de comandos PHP.

Por lo tanto, el manejo de formularios con PHP se vuelve muy sencillo. Aunque PHP es fácil de utilizar, realmente es un lenguaje de programación poderoso diseñado para interactuar entre Web y una base de datos en el servidor. Se diseño para trabajar bien especialmente con Apache, que también es código abierto y en el servidor Web más ampliamente utilizado en el mundo.

3.1.3 Java Server Page (JSP)

Ya vimos dos formas diferentes para generar páginas HTML dinámicas: las secuencias de comandos CGI y el PHP incrustado. También hay una tercera técnica llamada JSP, que es similar a PHP, excepto que la parte dinámica se escribe en el lenguaje de programación Java en lugar de en PHP.

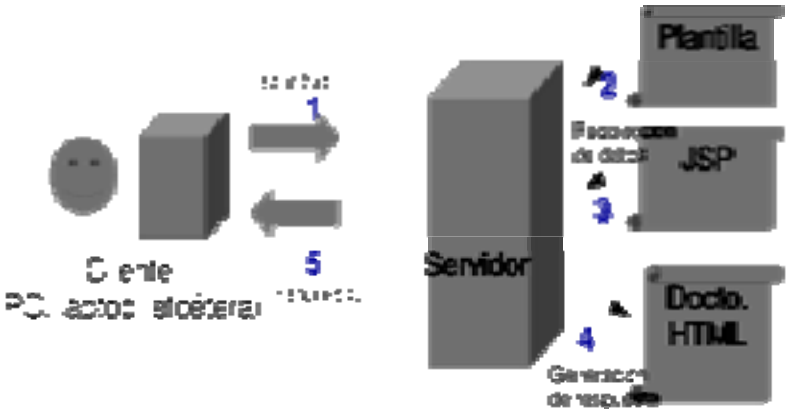


Figura 3.15 Java Server Page dentro de un servidor.

En las JSP se escribe el texto que va a ser devuelto en la salida (normalmente código HTML) incluyendo código java dentro de él para poder modificar o generar contenido dinámicamente. El código java se incluye dentro de las marcas de etiqueta <% y %>, a esto se le denomina scriptlet. Las páginas que utilizan esta técnica tienen la extensión de archivo jsp. La JSP es una tecnología Java que permite a los programadores generar contenido dinámico para Web, en forma de documentos HTML, XML, o de otro tipo. Las JSP's permite al código Java y a algunas acciones predefinidas ser incrustadas en el contenido estático del documento Web.

La principal ventaja de JSP frente a otros lenguajes es que permite integrarse con clases Java (.class) lo que permite separar en niveles las aplicaciones web, almacenando en clases java las partes que consumen más recursos así como las que requieren más seguridad, y dejando la parte encargada de formatear el documento html en el archivo jsp. La idea fundamental detrás de este criterio es el de separar la lógica del negocio de la presentación de la información.

Sin embargo JSP no se puede considerar un script al 100% ya que antes de ejecutarse el servidor web compila el script y genera un servlet, por lo tanto, se puede decir que aunque este proceso sea transparente para el programador no deja de ser una aplicación compilada. La ventaja de esto es algo más de rapidez y disponer del API de Java en su totalidad.

Para ejecutar las páginas JSP, se necesita un servidor Web con un contenedor Web que cumpla con las especificaciones de JSP y de Servlet. Tomcat 5 es una completa implementación de referencia para las especificaciones Java Servlet 2.2 y JSP 1.1.

3.1.4 Active Server Page (ASP)

Una cuarta técnica es ASP, es la versión de Microsoft de PHP y Java Server Pages. Active Server Pages (ASP) es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).

La tecnología ASP está estrechamente relacionada con el modelo tecnológico de su fabricante. Intenta ser solución para un modelo de programación rápida ya que programar en ASP es parecido a programar en VisualBasic.

Lo interesante de este modelo tecnológico es poder utilizar diversos componentes ya desarrollados como algunos controles ActiveX. Otros problemas que han hecho evolucionar esta tecnología es el no disponer de información que oriente a quienes desean aprenderla y resulta muy costosa en tiempo de descubrirla.

En el último ASP clásico, ASP 3.0, hay seis objetos integrados disponibles para el programador, Application, ASPError, Request, Response, Server y Session. Cada objeto tiene un grupo de funcionalidades frecuentemente usadas y útiles para crear páginas web dinámicas.

Las páginas pueden ser generadas mezclando código de scripts del lado del servidor (incluyendo acceso a base de datos) con HTML.

```
<%if x=1 then%>
        <b>X igual a uno</b>
<%else%>
        <b>X diferente a uno</b>
<%end if%>
```

Figura 3.16 Ejemplo de un código en ASP.

Este código trae como resultado en html X igual a uno, cuando se establece la variable del lado del servidor a X=1.

Se facilita la programación de sitios web mediante varios objetos integrados, como por ejemplo un objeto de sesión basada en cookies, que mantiene las variables mientras se pasa de página a página.

Desde 2002, el ASP clásico está siendo reemplazado por ASP.NET, que, entre otras cosas, reemplaza los lenguajes interpretados como VBScript o JScript por lenguajes compilados a código intermedio (llamado MSIL o Microsoft Intermediate Language) como Visual Basic, C#, o cualquier otro lenguaje que soporte la plataforma .NET. El código MSIL se compila con posterioridad a código nativo.

3.1.5 Microsoft .NET

.NET es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones. Basado en esta plataforma, Microsoft intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el Sistema Operativo hasta las herramientas de mercado.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems.

A largo plazo Microsoft pretende reemplazar el API Win32 o Windows API con la plataforma .NET. Esto debido a que el API Win32 o Windows API fue desarrollada sobre la marcha, careciendo de documentación detallada, uniformidad y cohesión entre sus distintos componentes, provocando múltiples problemas en el desarrollo de aplicaciones para el sistema operativo Windows.

Muchas otras empresas e instituciones se han unido a Microsoft en el desarrollo y fortalecimiento de la plataforma .NET, ya sea por medio de la implementación de la plataforma para otros sistemas operativos aparte de Windows (Proyecto Mono de Ximian/Novell para Linux/MacOS X/BSD/Solaris), el desarrollo de lenguajes de programación adicionales para la plataforma (ANSI C de la Universidad de Princeton, NetCOBOL de Fujitsu, Delphi de Borland, entre otros) o la creación de bloques adicionales para la plataforma (como controles, componentes y bibliotecas de clases adicionales); siendo algunas de ellas software libre, distribuibles ciertas bajo la licencia GPL.

Con esta plataforma Microsoft incursiona de lleno en el campo de los Servicios Web y establece el XML como norma en el transporte de información en sus productos y lo promociona como tal en los sistemas desarrollados utilizando sus herramientas.

.NET intenta ofrecer una manera rápida y económica pero a la vez segura y robusta de desarrollar aplicaciones - o como la misma plataforma las denomina, soluciones - permitiendo a su vez una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

3.1.6 Conclusiones

Por lo general, la selección entre PHP, JSP y ASP (o .NET) tiene que ver más con políticas (código abierto vs Sun vs Microsoft) que con la tecnología, puesto que estos tres lenguajes son más o menos similares.

La colección de tecnologías para generar contenido al vuelo algunas veces se llama DHTML (HTML Dinámico).

3.2 Generación de Páginas Web Dinámicas en el Cliente

Las secuencias de comandos de CGI, PHP, JSP y ASP resuelven el problema de manejar formularios e interacciones con bases de datos en el servidor. Lo que ninguno de ellos puede hacer es responder a los movimientos del ratón o interactuar de manera directa con los usuarios.

Para esto, es necesario tener secuencias de comandos incrustadas en páginas HTML que se ejecuten en el lado del cliente y no en el servidor. Comenzando con HTML 4.0, tales comandos son posibles mediante la etiqueta **<script>**. El lenguaje de secuencias de comandos más popular para el cliente es **JavaScript**, por los que lo abordaremos a continuación.

3.2.1 JavaScript

JavaScript es un lenguaje de comandos, inspirado por algunas ideas del lenguaje programación Java. Definitivamente este no es Java. Al igual que en otros lenguajes de secuencias de comandos, es un lenguaje de muy alto nivel. Las características de alto nivel como éstas hacen que JavaScript sea ideal para diseñar páginas Web interactivas. El único problema que presenta es su inestabilidad.

Como ejemplo de un programa en JavaScript considere el código siguiente:

```
<html>
<head>
<title>Procesar formularios con JavaScript</title>
<script language="javascript" type="text/javascript">
```

```
function response(test_form){
        var person = test_form.name.value;
        var year = eval(test_form.age.value) + 1;
        document.open();
        document.writeln("<html> <body>");
        document.writeln("Hola " + person + ".<br>");
        document.writeln("Predicción: El año siguiente usted tendra: " + year + ".");
        document.writeln("</body> </html>");
        document.close();
}
</script>

</head>
<body>
<form>
<p>Por favor, introduzca su nommbre: <input type="text" size="40" name="name" /></p>
<p>Por favor, intoduzca su edad: <input type="text" size="2" name="age" /> </p>
<p><input type="button" value="submit" onclick="response(this.form)"  /> </p>
</form>
</body>
</html>
```

Figura 3.17 Ejemplo de un código con JavaScript.

El código anterior, despliega en pantalla un formulario que pide el nombre y la edad. Este ejemplo es similar al que se realizó con PHP, la diferencia principal es la declaración del botón envío y su instrucción de asignación. Ésta indica al navegador que invoque la secuencia de comandos *response* mediante un clic en el botón y que la pase al formulario como un parámetro.

Lo que es completamente nuevo aquí es la función *response* de JavaScript del encabezado del archivo HTML, un área que reserva normalmente para títulos, colores de fondo, etc. Esta función extrae el valor del campo *name* del formulario y lo almacena como una cadena en la variable *person.* También extrae el valor del campo age, lo convierte en un entero mediante el uso de la función *eval,* le agrega 1 y almacena el resultado en *years.* A continuación abre un documento para salida, utiliza el método *writeln* para realizar 4 escriturasen dicho documento y por último, lo cierra.

Este documento es un archivo HTML, como puede advertirse a partir de las etiquetas HTML. A continuación, el navegador despliega el documento en pantalla.

Todo el trabajo se lleva a cabo localmente en el navegador por lo que es más rápido. PHP (y por ende, JSP y ASP) se utiliza cuando es necesaria la interacción con una base de datos remota. JavaScript se utiliza cuando la interacción es con el usuario de la máquina cliente. Es común tener páginas con PHP y JavaScript, aunque éstas no pueden hacer el mismo trabajo ni poseer, por supuesto, el mismo botón.

3.2.2 Otras alternativas

JavaScript no es la única forma de crear páginas Web altamente interactivas. Otro popular método es a través de subprogramas (*applets*). Éstos son pequeños programas de Java que se han compilado en instrucciones de máquina para una computadora virtual llamada JVM (Máquina Virtual de Java). Los subprogramas pueden incrustarse en páginas HTML (entre <applet> </applet>) en interpretarse mediante navegadores con gran capacidad. Debido a que los subprogramas de Java solo se interpretan y no se ejecutan directamente, el intérprete de Java puede evitar que causen daños, por lo menos en teoría. En la práctica, los escritores de subprogramas han encontrado un flujo infinito de errores en las bibliotecas de E/S de Java de los cuales aprovecharse.

La respuesta de Microsoft a los subprogramas de Java de Sun fue permitir que las páginas Web contuvieran controles **ActiveX,** que son programas compilados para lenguaje de máquina Pentium y ejecutados en el hardware. Estas características los hacen inmensamente rápidos y más flexibles que los subprogramas interpretados de Java porque pueden hacer cualquier cosa que un programa pueda hacer cualquier cosa que un programa puede hacer.

Cuando Internet Explorer ve un control **ActiveX** en una página Web, lo descarga, verifica su identidad y lo ejecuta. Sin embargo esta descarga aumenta los problemas de seguridad.

3.2.3 Comparativas entre JavaScript, Applets y ActiveX

Puesto que casi todos los navegadores pueden interpretar los programas Java y JavaScript, un diseñador que desee crear un página Web altamente interactiva

puede elegir entre al menos estas dos técnicas, y si la portabilidad a múltiples plataformas no es importante, también puede elegir ActiveX.

Como regla general, los programas de JavaScript son fáciles de escribir, los subprogramas de Java se ejecutan muy rápido y los controles ActiveX, se ejecutan todavía más rápido. Además, puesto que todos los navegadores implementan la misma JVM pero no todos implementan la misma versión de JavaScript, los subprogramas de Java, son más portables que los programas de JavaScript.

3.3 Conclusiones Generales

Antes de dejar el tema del contenido Web dinámico, resumamos brevemente lo que hemos visto hasta ahora. Las páginas Web pueden generarse en la marcha mediante una secuencia de comandos en la máquina servidora. Una vez que el navegador las recibe, las trata como páginas HTML normales y simplemente las despliega. Las secuencias de comandos pueden escribirse en Perl, PHP, ASP o JSP. Como se muestra en la figura siguiente.

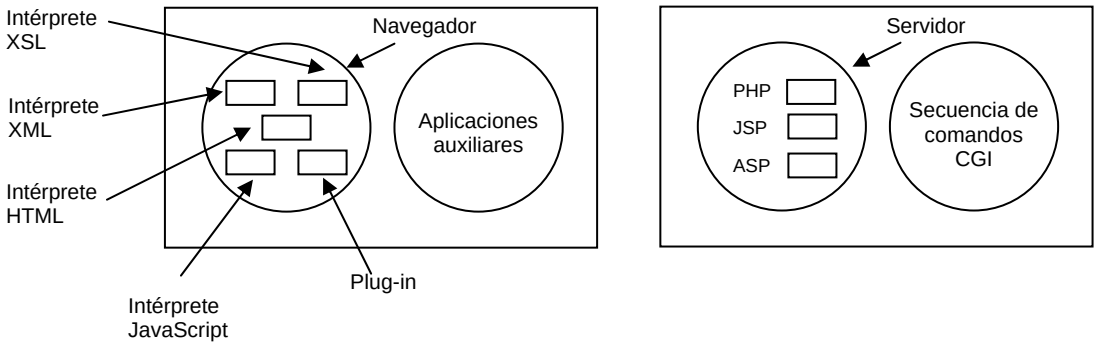


Figura 3.18 Ubicación de las diversas tecnologías dentro de las máquinas de los clientes y los servidores.

La generación de contenido dinámico, también es posible en el lado del cliente. Es posible escribir en XML las páginas Web y después convertirlas en HTML de acuerdo a un archivo XSL. Los programas de JavaScript pueden realizar cálculos arbitrarios. Por último, es posible utilizar los plug-ins y las aplicaciones auxiliares para desplegar contenido en varios formatos.

4. HTTP (Protocolo de Transferencia de Hipertexto).

4.1 Introducción

HTTP significa Hypertext Transfer Protocol. Es el protocolo de red que se utiliza para transferir los archivos (llamados recursos) que forman parte de la World Wide Web. Ya sean estos archivos HTML, imágenes, sonidos, etc.

Normalmente HTTP utiliza TCP como medio de transporte. Y las conexiones que establece pueden ser: persistentes o no persistentes:

- **Persistentes:** Son aquellas en las que los diferentes objetos que integran una página Web son trasladados a través de una única conexión.

- **No persistentes:** Son aquellas en las que los diferentes objetos que forman una página Web son trasladados a través de diferentes conexiones.

4.2 ¿Qué son los recursos?

HTTP se puede utilizar para transferir Recursos, no solo archivos. Un recurso es un trozo de información que puede identificarse a través de una URL. La clase más común de recursos son los archivos, pero también pueden ser datos generados dinámicamente.

4.3 Estructura de las transacciones

HTTP utiliza el modelo cliente/servidor. Un cliente HTTP (un navegador) abre una conexión hacia un servidor HTTP y envía un mensaje de petición (request message), luego el servidor envía un mensaje de respuesta (response message) el cual contiene el recurso solicitado.

Después de enviar la respuesta el servidor cierra la conexión (a no ser que se especifique lo contrario en las cabeceras HTTP). Por ende, el protocolo no mantiene estado (stateless) entre las distintas transacciones de un mismo cliente. Por ello es que se utilizan las cookies.

4.4 Mensajes HTTP

Las solicitudes HTTP pueden ser:

- Solicitudes.
- Respuestas.

Tanto las solicitudes como las respuestas utilizan el formato genérico de e-mails (RFC-822). Ambos tipos de mensajes consisten en:

- Una línea inicial.
- Cero o más encabezados (headers).
- Una línea en blanco.
- Un cuerpo del mensaje (opcional, ej. Datos de una consulta).

Resumiendo el formato de un mensaje HTTP es:

- <línea inicial, distinta para solicitudes y respuestas>
- Encabezado1: valor1
- Encabezado1: valor1
- …<más encabezados>
- Encabezado N: valor N
- <línea en blanco>
- <cuerpo de mensaje opcional, contenidos de un archivo, de una consulta, datos binarios, etc>

4.4.1 Línea Inicial (Solicitud)

La línea inicial de una solicitud tiene tres partes separadas entre sí por un espacio:

- El método (GET, PUT, POST, OPTIONS, TRACE, DELETE,….).
- El identificador del recurso (URL).
- La versión del protocolo HTTP en uso.

Ejemplos:

GET /directorio1/directorio2/index.html HTTP/1.0

4.4.2 Línea Inicial (Respuesta)

La línea inicial de una respuesta (llamada línea de estado) tiene tres partes separadas entre sí por un espacio:

- Versión de HTTP.
- Código de estado.
- Frase explicativa (legible por humanos).

Ejemplos:

HTTP/1.0 200 OK
HTTP/1.0 404 Not Found

Códigos de estado

El código de estado es un entero de 3 dígitos:

- 1XX Informativos.
- 2XX Éxito.
- 3XX Redirección.
- 4XX Error de cliente.
- 5XX Error de servidor.

Los más comunes son:

- 200 OK: Solicitud exitosa, la respuesta se envía en el cuerpo.
- 404 Not Found: El recurso no existe.
- 303 See Other: El recurso se ha movido a otra URL (Dada en el header Location).
- 500 Server Error: Error no esperado en el serviodor.

4.4.3 Línea de Encabezado

Estas líneas proveen información acerca de la solicitud o respuesta. Cada línea de encabezado consiste de un nombre de campo seguido por dos puntos (":") y el valor para ese campo. El orden de los campos no es importante. Ejemplo:

User-Agent: Mozilla/6.0

Content-Type: text/html

HTTP 1.0 define 16 headers (ninguno es obligatorio), mientras que HTTP 1.1 define 46 headers de las cuales sólo Host es obligatoria.

En las solicitudes suelen incluirse los siguientes:

User-Agent: (Identifica el software de cliente y la versión).

From: (La dirección de e-mail de quién envía la solicitud).

En las solicitudes, algunos encabezados comunes son:

Server: (análogo a User-Agent:, ej. Server: Apache/1.3.14).

Last-Modified: (fecha de última modificación del recurso, se utiliza para mantener actualizados los cache. Ej: Last-Modified: Fri, 31 Jan 2000 12:12:12 GMT).

4.4.4 Cuerpo del mensaje

Después de las líneas de encabezado, un mensaje HTTP puede contener un cuerpo (body). En las respuestas el cuerpo es la sección en donde se envía el recurso solicitado. En las solicitudes, el cuerpo se utiliza para subir datos que ingresó el usuario o para transferir archivos hacia el servidor. Las líneas más comunes del encabezado que definen el cuerpo son:

Content-Type: (Da el MIME de los datos del cuerpo, ejemplo: text/html ó image/gif).

4.5 Ejemplo de una sesión HTTP

A continuación se describe lo que ocurre cuando se realiza una conexión HTTP:

- El cliente desea obtener http://www.aplicacionestelematicas.edu.ni/index.html
- El cliente establece una conexión TCP al puerto 80 de www.aplicacionestelematicas.edu.ni y envía la solicitud con el método GET.

  GET /index.html HTTP/1.0

  User-Agent: MiBrowser/2.0

  [línea en blanco]
- El servidor responde por la misma conexión con:

  HTTP/1.0 200 OK

  Date: Sat, 18 Nov 2000 15:18:02 GMT

  Content-Type: text/html

  Content-Length: 52

  [línea en blanco]

  <html><body>

4.6 Métodos de HTTP

4.6.1 Método OPTIONS

Este método representa una petición de información sobre las opciones de comunicación disponibles en la cadena petición-respuesta identificada por la URI de la petición. Esto permite al cliente conocer las opciones y requisitos asociados con un recurso o las capacidades del servidor.

La respuesta sólo debe incluir información sobre las opciones de comunicación.

Si la URL es ”*” entonces la petición se aplica al servidor como un conjunto. Es decir, contesta características opcionales definidas por el servidor, extensiones del protocolo.

4.6.2 Método GET

El método GET requiere la devolución de información al cliente identificada por la URI. Si la URI se refiere a un proceso que produce información, se devuelve la información y no la fuente del proceso. El método GET pasa a ser un GET condicional si la petición incluye las cabeceras If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match o If-Range. Estas cabeceras hacen que el contenido de la respuesta se

transmita sólo si se cumplen unas condiciones determinadas por esas cabeceras. Esto se hico para reducir el tráfico en las redes.

También hay un método GET parcial, con el que se envía sólo parte del contenido del recurso requerido. Esto ocurre cuando la petición tiene una cabecera Range. Al igual que el método GET condicional, el método GET parcial se creó para reducir el tráfico en la red.

4.6.3 Método HEAD

El método HEAD es igual que el método GET, salvo que el servidor no tiene que devolver el contenido, sólo las cabeceras. Estas cabeceras que se devuelven en el método HEAD deberían ser las mismas que las que se devolverían si fuese una petición GET.

Este método se puede usar para obtener información sobre el contenido que se va a devolver en respuesta a la petición. Se suele usar también para chequear la validez de links, accesibilidad y modificaciones recientes.

4.6.4 Método POST

El método POST se usa para hacer peticiones en las que el servidor destino acepta el contenido de la petición como un nuevo subordinado del recurso pedido. El método POST se creó para cubrir funciones como la de enviar un mensaje a grupos de usuarios, dar un bloque de datos como resultado de un formulario a un proceso de datos, añadir nuevos datos a una base de datos.

La función llevada a cabo por el método POST está determinada por el servidor y suele depender de la URI de la petición. El resultado de la acción realizada por el método POST puede ser un recurso que no sea identificable mediante una URI.

4.6.5 Método PUT

El método PUT permite guardar el contenido de la petición en el servidor bajo la URI de la petición. Si esta URI ya existe, entonces el servidor considera que esta

petición proporciona una versión actualizada del recurso. Si la URI indicada no existe y es válida para definir un nuevo recurso, el servidor puede crear el recurso con esa URI. Si se crea un nuevo recurso, debe responder con un código 201 (creado), si se modifica se contesta con un código 200 (OK) o 204 (sin contenido). En caso de que no se pueda crear el recurso se devuelve un mensaje con el código de error apropiado.

La principal diferencia entre POST y PUT se encuentra en el significado de la URI. En el caso del método POST, la URI identifica el recurso que va a manejar en contenido, mientras que en el PUT identifica el contenido. Un recurso puede tener distintas URI.

4.6.6 Método 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 la operación se lleve a cabo aunque la respuesta sea satisfactoria.

4.6.7 Método TRACE

Este método se usa para saber si existe el receptor del mensaje y usar la información para hacer un diagnóstico. En las cabeceras el campo Via sirve para obtener la ruta que sigue el mensaje. Mediante el campo Max-Forwards se limita el número de pasos intermedios que puede tomar. Esto es útil para evitar bucles entre los proxy.

La petición con el método TRACE no tiene contenido.

4.7 Cabeceras generales

Los campos de este tipo de cabeceras se aplican tanto a las peticiones como a las respuestas, pero no al contenido de los mensajes.

Estas cabeceras son:

- **Cache-Control:** Son directivas que se han de tener en cuenta a la hora de mantener el contenido en una caché.

- **Connection:** Permite especificar opciones requeridas para una conexión.
- **Date:** Representa la fecha y la hora a la que se creó el mensaje.
- **Pragma:** Usado para incluir directivas de implementación.
- **Transfer-Encoding:** Indica la codificación aplicada al contenido.
- **Upgrade:** Permite al cliente especificar protocolos que soporta.
- **Via:** Usado por pasarelas y proxies para indicar los pasos seguidos.

4.8 Cabeceras de petición

Este tipo de cabeceras permite al cliente pasar información adicional al servidor sobre la petición y el propio cliente. Estas cabeceras son las siguientes:

- **Accept:** Indican el tipo de respuesta que acepta.
- **Accept-Charset**: Indica los conjuntos de caracteres que acepta.
- **Accept-Encoding:** Que tipo de codificación acepta.
- **Accept-Language:** Tipo de lenguaje de la respuesta que se prefiere.
- **Authorization:** El agente de usuario quiere autentificarse con el servidor.
- **From:** Contiene la dirección de correo que controla en agente de usuario.
- **Host:** Especifica la máquina y el puerto del recurso pedido.
- **If-Modified-Since:** Para el GET condicional.
- **If-Match:** Para el GET condicional.
- **If-None-Match:** Para el GET condicional.
- **If-Range:** Para el GET condicional.
- **If-Unmodified-Since:** Para el GET condicional.
- **Max-Forwards:** Indica el máximo número de elementos por los que pasa.
- **Proxy-Authorization:** Permite que el cliente se identifique a un proxy.
- **Range:** Establece un rango de bytes del contenido.
- **Referer:** Indica la dirección donde obtuvo la URI de la petición.
- **User-Agent:** Información sobre el agente que genera la petición.

4.9 Cabeceras de respuesta

Permiten al servidor pasar información adicional al cliente sobre la respuesta, el propio servidor y el recurso solicitado. Son los campos:

- **Age:** Estimación del tiempo transcurrido desde que se creó la respuesta.
- **Location:** Se usa par a redirigir la petición a otra URI.
- **Proxy-Authenticate:** Ante una respuesta con el código 407 (autentificación proxy requerida), indica el esquema de autentificación.
- **Public:** Da la lista de métodos soportados por el servidor.
- **Retry-After:** Ante un servicio no disponible da una fecha para volver a intentarlo.
- **Server:** Información sobre el servidor que maneja las peticiones.
- **Vary:** Indica que hay varias respuestas y el servidor ha escogido una.
- **Warning:** Usada para aportar información adicional sobre el estado de la respuesta.
- **WWW-Authenticate:** Indica el esquema de autentificación y los parámetros aplicables a la URI.

4.10 Cabeceras de entidad

Como su nombre indica, los campos de este tipo aportan información sobre el contenido del mensaje o si no hay contenido, sobre el recurso al que hace referencia la URI de la petición. Los campos de este tipo son:

- **Allow:** Da los métodos soportados por el recurso designado por la URI.
- **Content-Base:** Indica la URI base para resolver las URI relativas.
- **Content-Encoding:** Indica una codificación adicional aplicada al contenido (a parte de la aplicada por el tipo).
- **Content-Language:** Describe el idioma del contenido.
- **Content-Length:** Indica el tamaño del contenido del mensaje.
- **Content-Location:** Da información sobre la localización del recurso que da el contenido del mensaje.
- **Content-MD5:** Es un resumen en formato MD5 (RFC 1864) para chequear la integridad del contenido.
- **Content-Range:** En un GET parcial, indica la posición del contenido.
- **Content-Type:** Indica el tipo de contenido que es.
- **Etag:** Define una marca para el contenido asociado.
- **Expires:** Indica la fecha a partir de la cual la respuesta deja de ser válida.
- **Last-Modified:** Indica la fecha de la última modificación.

5. Mejoras del Desempeño.

5.1 Introducción

La popularidad de Web ha sido casi su destrucción. Los servidores, enrutadores y las líneas con frecuencia están sobrecargados. Por esta razón, se han realizado diversas técnicas para mejorar el desempeño entre las cuales tenemos: el almacenamiento en caché, la replicación de servidores y las redes de distribución de contenidos.

5.2 Almacenamiento en Caché

Consiste en guardar las páginas que han sido solicitadas en caso de que se utilicen nuevamente. Esta técnica es útil principalmente con páginas que son visitadas frecuentemente y se conoce como almacenamiento en caché. El procedimiento común es que algún proceso, llamado Proxy, mantenga el caché.

Para utilizar el almacenamiento en caché un navegador puede configurarse para que todas las solicitudes de páginas se le hagan a un Proxy en lugar de al servidor real de la página. Si el Proxy tiene la página, la regresará de inmediato. De lo contrario, la obtiene del servidor, la agrega añ caché para su uso posterior y la envía al cliente que la solicitó.

5.3 Replicación del Servidor

El almacenamiento en caché es una técnica en el cliente para mejorar el desempeño, pero también existen técnicas en el servidor. El método más común que los servidores utilizan para mejorar el desempeño es replicar su contenido en múltiples ubicaciones separadas considerablemente. Esta técnica a veces se conoce como espejo (*mirroring*).

Desafortunadamente, Web tiene un fenómeno conocido como *flash crowds* (aglomeración instantánea) en el que un sitio que era un lugar alejado, desconocido y no visitado de repente se vuelve el centro del universo. Lo que se necesita es una forma para que un sitio Web que de repente note un incremento masivo en el tráfico, se clone automáticamente a sí mismo en tantas ubicaciones como sea necesario.

Para tener esta capacidad, un sitio necesita un acuerdo por adelantado con alguna compañía que posea muchos sitios de hospedaje, que indique que puede crear réplicas bajo demanda y que debe pagar por la capacidad que realmente utilice.

5.4 Redes de Entrega de Contenido (CDN)

Los CDN utilizan un modelo de negocio distinto al de las caché web. Para un CDN, los clientes que pagan no son los ISP, sino los proveedores de contenidos. Un proveedor de contenido, (como Yahoo) paga a una compañía de CDN (como Akamai) y hace accesible sus contenidos con retardos menores a los usuarios que los pidan.

Una compañía CDN típicamente proporciona el servicio de distribución de contenido de la siguiente forma:

- La compañía CDN instala cientos de servidores CDN a lo largo de Internet. La compañía CDN típicamente emplaza los servidores CDN en centros de host de Internet. Un centro de host de Internet cuya propiedad y gestión corresponde a un tercero. Los centros de host están en algún ISP de capas bajas, cerca de las redes de acceso ISP.

- CDN replica los contenidos de sus clientes en los servidores CDN. Cuando un cliente actualiza sus contenidos (por ejemplo, modifica una página web), CDN redistribuye el contenido actualizado a los servidores CDN.

- La compañía CDN proporciona un mecanismo para que, cuando un usuario pida un contenido, ésta sea proporcionado por el servidor CDN que pueda entregar más rápidamente el contenido al usuario. Este servidor podría ser el CDN más cercano al usuario (quizá en el mismo ISP que el usuario), o el que tenga un camino no congestionado hasta el usuario.

TEMA 4: CORREO ELECTRÓNICO

Objetivos

- Explicar las características, formato de los mensajes y funciones básicas del correo electrónico.
- Describir la manera en la que se lleva a cabo la comunicación entre los agentes de usuario y los servidores de correo electrónico a través del protocolo SMTP.
- Analizar las diferentes técnicas de codificación y los diferentes formatos empleados en MIME.
- Comparar los protocolos de entrega final POP3 e IMAP.

Contenido del Tema

- El correo electrónico en Internet.
- El protocolo de transporte SMTP.
- Estructura y codificación de mensajes MIME.
- Protocolos de Entrega Final (POP3 e IMAP).

Bibliografía

- Andrews S. Tanenbaum, Redes de Computadores, Cuarta Edición 2003.
- William Stallings, Comunicaciones y Redes de Computadores, Séptima Edición 2004.
- RFC 821, 822
- http://ditec.um.es/laso/docs/tut-tcpip/3376c46.html#smtp
- http://ditec.um.es/laso/docs/tut-tcpip/3376c47.html#mime
- http://es.wikipedia.org/wiki/Post_Office_Protocol
- http://es.wikipedia.org/wiki/Internet_Message_Access_Protocol

1. El Correo Electrónico en Internet

1.1 Introducción

Los primeros sistemas de correos electrónicos simplemente consistían en protocolos de transferencias de archivos, con la convención de que la primera línea de cada mensaje (es decir, archivos) contenía la dirección del destinatario.

A medida que pasó en el tiempo, las limitaciones de este enfoque fueron más evidentes. Las principales desventajas fueron:

- El envío de un mensaje a un grupo de personas era laboriosos.
- Los mensajes no tenían una estructura interna, lo que dificultaba el proceso por computadora.
- El remitente nunca sabía se el mensaje había llegado o no.
- No era fácil la delegación del correo entrante a otra persona.
- No había programas de correo electrónico con editor integrado, por lo que había que escribirlo en otro editor.
- Los mensajes no podían incluir mezclas de texto, dibujos, facsímiles y voz.

A medida que se acumuló experiencia se propusieron sistemas de correos electrónicos más elaborados. En 1982, se publicaron las propuestas de correo electrónico de ARPANET como el RFC 821 (protocolo de transmisión) y el RFC 822 (formato de mensaje). En 1984, el CITT redactó la recomendación X.400. Después de dos décadas de competencia, los sistemas de correo electrónico basados en el RFC 822 se utilizan ampliamente, mientras que aquellos basados en X.400 han desaparecido debido a que estaban pobremente diseñados y era muy complejo.

1.2 Arquitectura del Sistema de Correo

Normalmente el sistema de correo se divide en **dos subsistemas** que son:

- **Los agentes de usuarios:** Permiten a la gente leer y recibir correo electrónico. Son programas locales que proporcionan un método basado en comandos, en

menús o una interfaz gráfica para interactuar con el sistema de correo electrónico.

- **Los agentes de transferencia:** Mueven los mensajes del origen al destino. Por lo común son demonios (*daemons*) del sistema que operan en segundo plano mueven correo electrónico a través del sistema.

Por lo general, los sistemas de correo electrónico desempeñan **cinco funciones básicas** como se describe a continuación:

- **Redacción:** Proceso de crear mensajes y respuestas. Aunque es posible utilizar cualquier editor para cuerpo del mensaje, el propio sistema puede proporcionar uno que incluso nos asista en el rellenado de ciertos campos de la cabecera del mensaje. También puede extraer la dirección de un remitente e insertarla de forma automática en el lugar adecuado de la respuesta.

- **Transferencia:** Se refiere a mover mensajes del remitente al destinatario. En gran medida, esto requiere establecer una conexión con el destino o alguna máquina intermedia, enviar el mensaje y liberar la conexión.

- **Generación de Informes:** Es indicar al remitente lo que ocurrió con el mensaje: ¿se entregó, rechazó o se perdió?

- **Visualización:** Permite leer el correo electrónico. A veces requiere de una conversión o debe invocarse a un visor especial, por ejemplo, si el mensaje es un archivo PostScript o voz digitalizada. Algunas veces también se intentan conversiones y formateo sencillo.

- **Disposición:** Tiene que ver con lo que el destinatario hace con el correo una ves que lo recibe. Las posibilidades incluyen tirarlo antes o después de leerlo, guardarlo, etcétera.

Además de estos servicios básicos, la mayoría de los sistemas de correo, especialmente los corporativos internos proporcionan una gran variedad de características avanzadas. Algunas de estas son:

- Reenvío de correo electrónico por cambio de domicilio, vacaciones, etc.
- Creación de buzones de correo para almacenar el correo entrante.
- Creación de lista de correo, que es una lista de correos electrónicos para poder enviar a todos una copia idéntica del mensaje.

Otras características avanzadas son copias ocultas, correo electrónico de alta prioridad, correo electrónico secreto (es decir, encriptado), destinatarios alternos si el correo primario no esta disponible.

En la actualidad, el correo electrónico se usa ampliamente en la industria para la comunicación entre compañías y para la comunicación entre diferentes personas.

1.3 Direcciones de correo electrónico

Una dirección de correo electrónico es la forma que tenemos de especificar al programa de correo electrónico, el lugar o persona a la que queremos enviar el mensaje en concreto. La dirección de correo electrónico tiene la siguiente forma:

usuario @ maquina.dominio

- **Usuario:** Indica el buzón de correo electrónico correspondiente a la persona a la que va destinado el mensaje. Después se pone un símbolo que se denomina arroba.

- **Maquina:** Nombre del servidor donde tiene cuenta el usuario y por ultimo se pone un punto seguido del tipo de dominio al que pertenece la maquina. Todo lo que va después de la arroba se le suele llamar nombre completo del dominio del ordenador.

Dominio: Hay varios tipos de dominios en Internet. Normalmente suelen tener como máximo 3 letras que los identifican (son los abordados en el DNS).

1.4 Estructura de los mensajes de correo electrónico

Para enviar un correo electrónico se realiza de manera análoga al envío de un correo por sistema normal de correo. Al escribirla se inicia colocando los datos del

remitente y luego el contenido de la carta, ambos son datos que solo le incumben al destinatario.

Posteriormente se guarda en un sobre en el cual se loca la dirección de destino para que la oficina de correo pueda enviarla y nuestra propia dirección en caso de que no se encuentre el destino solicitado.

Como podemos observar, una idea clave de los sistemas de correo electrónico es la distinción entre el sobre y su contenido.

- **El sobre**, encapsula el mensaje; contiene toda la información necesaria para transportarlo, como dirección destino, prioridad, nivel de seguridad, la cual es diferente del mensaje mismo. Los agentes de transporte del mensaje lo utilizan para enrutarlo, al igual que el oficial postal.

- **El mensaje**, dentro del sobre contiene dos partes que son: El encabezado y el cuerpo. El encabezado contiene información de control para los agentes de usuario. El cuerpo es por completo para el destinatario humano.

1.5 Formato de los mensajes

Primero veremos el formato ASCII básico usando el RFC 822 y en el siguiente tema analizaremos la estructura y codificación de las extensiones multimedia del RFC 822 (MIME).

Los mensajes consisten en: un sobre primitivo (descrito en la RFC 821), algunos campos de encabezado, una línea en blanco y el cuerpo del mensaje.

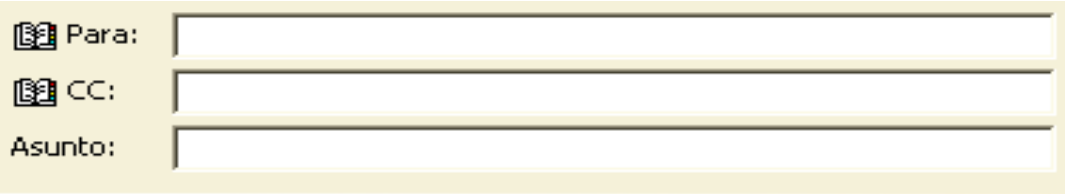


Figura 4.1 Ejemplo de los campos en Outlook Express

Cada campo del encabezado consiste en una sola línea de texto ASCII que contiene el nombre del campo, dos puntos (:) y, para la mayoría de los campos un

valor. A continuación se presentan los distintos campos que conforman la cabecera de un mensaje.

Encabezado	Significado
To:	Direcciones de correo electrónico de los destinatarios.
Cc:	Direcciones de correo electrónico de los destinatarios secundarios.
Bcc:	Direcciones de correo electrónico para las copias ocultas.
From:	Persona(s) que crearon el mensaje.
Sender:	Dirección de correo electrónico del remitente.
Recived:	Línea agregada por cada agente de transferencia en la ruta.
Return-Path:	Puede usarse para identificar una ruta del remitente en la ruta.

Figura 4.2 Campos de encabezado RFC 822 relacionados con el transporte de mensajes

Algunos campos interesantes son:

- **BCC:** Este campo significa copia complementaria invisible, es decir, figuran en dicho campo aquellos destinatarios a los que se le envía una copia del mensaje sin que los usuarios principales y secundarios sepan que estos destinatarios han sido incluidos, ya que no les aparece esta línea en sus mensajes.

  Ej: BCC: invisible@algunsitio.com

- **Sender:** Este campo indica el remitente cuando este remitente no es la persona, sistema o proceso que introdujo el mensaje. Si el remitente no es el creador del mensaje, el remitente tiene que poner una persona como responsable del envío del mensaje, que es la que figura en el campo Sender.

  Ej: From: secretaria@yahoo.es

  Ej: Sender: denisjev@yahoo.es

- **Recived:** Esta línea se agrega por cada agente de transferencia de mensaje en el camino. La línea contiene la identidad del agente, la fecha y la hora de recepción del mensaje, y otra información que puede servir para encontrar fallas en el sistema de enrutamiento.

- **Return-Path:** Es agregado por cada agente de transferencia de mensajes final y estaba destinado para indicar la manera de regresar al remitente. En teoría, esta información puede tomarse de todos los encabezados *Received:* (con excepción del nombre del buzón del remitente), pero pocas veces se llena de esta manera y por lo general solo contiene la dirección del remitente.

Además de los campos anteriormente mencionados se suelen incluir algunos otros usados por los agentes de usuarios o los destinatarios, los cuales se muestran a continuación.

Encabezado	Significado
Date:	Fecha y hora de envía del mensaje.
Reply-To:	Dirección de correo electrónico a la que deben enviarse las contestaciones.
Message-Id:	Número único de referencia posterior a este mensaje.
In-Reply-To:	Identificador del mensaje al que este responde.
Referentes:	Otros identificadores de mensajes pertinentes.
Keywords:	Claves seleccionadas por el usuario.
Subject:	Resumen corto del mensaje para desplegar en una línea.

Figura 4.3 Campos de encabezado RFC 822 relacionados con el agente de usuario o destinatario

- **Date:** Este campo contiene la fecha y hora en la que se creo el mensaje. Este campo es añadido bien por la aplicación cliente o por el servidor de correo. El formato de la fecha en los mensajes de Internet es el siguiente:

Date: [dia,] fecha hora

El día es opcional como podemos ver, pero siempre se suele poner. El día se representa con tres caracteres del día de la semana. La fecha sigue el formato DD MMM AAAA. La hora tiene el formato HH:MM[:SS] zona horaria. La zona horaria se representa con tres caracteres que indican la zona horaria donde se origino la fecha y la hora.

Ej: Date: Mon, 28 Jul 1997 18:28:14 GMT

- **Reply-to:** Este campo indica el buzón o buzones a los que se deben enviar las repuestas. Este campo toma importancia cuando las respuestas se necesiten recibir en buzones distintos de los del remitente.

  Ej: Reply: denisespinoza85@hotmail.com

- **Message-ID:** Este campo del encabezado, contiene un identificador único que es generado por la maquina remitente (el servidor). Este campo es usado por la aplicación encargada de enviar y recibir el mensaje. Este identificador es normalmente una combinación de la fecha, la hora en que se genero el mensaje, nombre de la maquina y posiblemente un generador de números para garantizar su exclusividad.

  Ej: Message-ID: <199707311220.0AA17101@yahoo.es>

- **In-Reply-To:** Este campo contiene todos los ID (identificadores) de mensaje de los mensajes anteriores a los que este mensaje es respuesta. Puede contener varios ID de mensaje separados por comas y el principio y final de los ID de mensaje están delimitados por los símbolos < y >. Este campo es añadido por la aplicación cliente.

  Ej: In-Reply-To: <199707311220.0AA17101@yahoo.es>

- **References:** Este campo contiene todos los ID de mensaje de los mensajes anteriores a los que este mensaje hace referencia. Es un campo generado por la aplicación cliente. Puede contener uno o varios ID de mensaje separados por comas.

  Ej: References: <199707311220.0AA17101@yahoo.es>

- **Keywords:** Este campo contiene palabras clase, incluso frases que identifican el contenido del mensaje. Puede ser una palabra o frase o varias separadas por comas. Este campo es introducido por el usuario.

  Ej: Keywords: proyecto, es de correo electrónico, trabajar duro.

- **Subject:** este campo contiene un resumen de lo que dice el mensaje. Este campo es rellenado normalmente por el remitente del mensaje (persona, proceso o sistema).

  Ej: Subject: Como va el proyecto.

Tras los encabezados viene el cuerpo de los mensajes. Algunos terminan sus mensajes con firmas complicadas, incluidas caricaturas sencillas en ASCII, citas de autoridades mayores y menores, pronunciamientos políticos y renuncias de responsabilidades de todo tipo.

1.6 Los agentes de usuario

Un agente de usuario es normalmente un programa (a veces llamado lector de correo) que acepta una variedad de comandos para redactar, recibir y contestar mensajes, así como para manipular los buzones de correo. Algunos tienen una interfaz elegante para el usuario mientras que otros operan con comandos, aunque funcionalmente ambos son iguales.

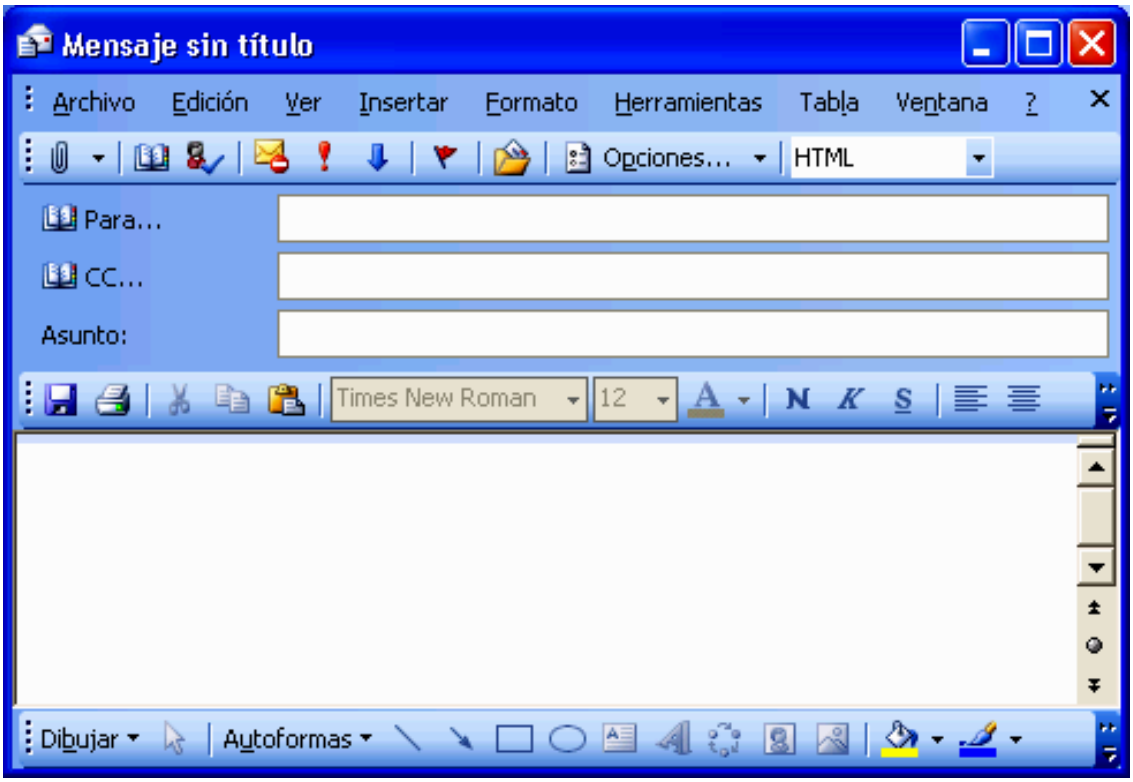


Figura 4.4 Agente de usuario de Microsoft Outlook.

Los agentes de usuario varían de un software a otro aunque en general presentan siempre las mismas funcionalidades. Entre las más básicas tenemos, el envío de correo electrónico a una o varias personas (el caso de las listas) y el de lectura de correos.

2. El Protocolo simple de transferencia de correo SMTP

2.1 Introducción

El significado de las siglas de SMTP es Protocolo Simple de Transmisión de Correo (Simple Mail Transfer Protocol). Los protocolos de correo básicos de correo proporcionan intercambio de correo y mensajes entre hosts TCP/IP hosts; se han añadido servicios para la transmisión de datos que no se pueden representar con texto ASCII de 7 bits.

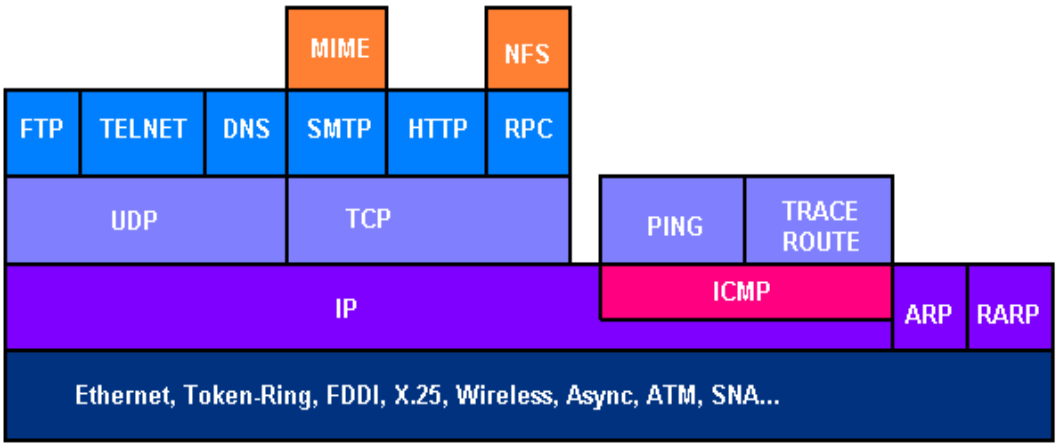


Figura 4.5 Ubicación de SMTP en la torre de protocolos.

Hay tres protocolos estándares que se aplican a este tipo de correo. Todos son recomendados. El término SMTP se emplea con frecuencia para referirse a la combinación de los tres protocolos, por su estrecha interrelación, pero estrictamente hablando, SMTP es sólo uno de los tres. Normalmente, el contexto hace evidente de cuál de los tres se está hablando. Cuando haya ambigüedad, se emplearán los números STD o RFC. Los tres estándares son:

1. Un estándar para el intercambio de correo entre dos ordenadores (STD 10/RFC 821), que especifica el protocolo usado para enviar correo entre hosts TCP/IP. Este estándar es SMTP.

2. Un estándar (STD 11) para el formato de los mensajes de correo, contenido en dos RFCs. El RFC 822 describe la sintaxis de las cabeceras y su interpretación. El RFC 1049 describe como un conjunto de documentos de

tipos diferentes del texto ASCII plano se pueden usar en el cuerpo del correo (los mismos documentos están en ASCII de 7 bits con información de formato embebida: Postscript, Scribe, SGML, TEX, TROFF y DVI aparecen en el estándar). El nombre oficial del protocolo para este estándar es MAIL.

3. Un estándar para el encaminamiento de correo usando el DNS, descrito en el RFC 974. El nombre oficial del protocolo para este estándar es DNS-MX.

2.2 El STD 10/RFC 821

Establece que los datos enviados por SMTP son ASCII de 7-bis, con el bit de orden superior a cero. Esto es adecuado para mensajes en inglés, pero no para otros lenguajes o datos que no sean texto. Hay dos estrategias para superar estas limitaciones:

- **MIME** ("Multipurpose Internet Mail Extensions"), definido en los RFCs 1521 y 1522, que especifica un mecanismo para codificar texto y datos binarios en ASCII de 7 bits en el mensaje RFC 822. MIME se describe en MIME("Multipurpose Internet Mail Extensions").

- **SMTPSE** ("SMTP Service Extensions"), que define un mecanismo para extender las posibilidades de SMTP más allá de las limitaciones impuestas por RFC 821.

2.3 Órdenes y Respuestas en SMTP

El funcionamiento de SMTP se basa en una serie de órdenes y respuestas intercambiadas entre el emisor y el receptor SMTP. La iniciativa la lleva el SMTP emiso, quien establece la conexión TCP. Una vez que se ha establecido la conexión, el emisor SMTP envía órdenes al receptor a través de la conexión. Cada orden genera exactamente una respuesta del receptor.

Nombre	Formato de la Orden	Descripción
HELO	HALO <SP> <dominio> <CRLF>	Envía identificación
MAIL	MAIL <SP> FROM: <camino inverso> <CRLF>	Identifica el origin
RCPT	RCPT <SP> TO: <camino al destino> <CRLF>	Identifica al destino
DATA	DATA <CRLF>	Transfiere texto del mensaje
RSET	RSET <CRLF>	Aborta el correo en curso
NOOP	NOOP <CRLF>	Operación nula
QUIT	QUIT <CRLF>	Cierre de la conexión TCP
SEND	SEND <SP> FROM: <camino inverso> <CRLF>	Envía correo al terminal
SOML	SOML <SP> FROM: <camino inverso> <CRLF>	Envía correo al terminal si es posible en caso contrario lo envía al buzón.
SAML	SAML <SP> FROM: <camino inverso> <CRLF>	Envía correo al terminal y al buzón
VRFY	VRFY <SP> <cadena> <CRLF>	Confirma el nombre del usuario
EXPN	EXPN <SP> <cadena> <CRLF>	Devuelve la lista de miembros de una lista de correo
HELP	HELP [<SP> <cadena>] <CRLF>	Envía documentación específica del sistema.
TURN	TURN <CRLF>	Intercambia el rol del emisor y el receptor.

Figura 4.6 Comandos SMTP.

- <CRLF> = retorno de carro, salto de linea
- <SP> = espacio
- Los corchetes indican elementos opcionales
- Los comandos sombreados son opcionales en implementaciones conformes a SMTP

Cada orden consta de una única línea de texto que comienza con un código de orden de 4 letras, seguido en algunos casos por un campo argumento. La mayoría de las respuestas constan de una sola línea, aunque son posibles respuestas de varias líneas. En la tabla se señalan aquellas órdenes que todos los receptores deben ser

capaces de reconocer. Las otras órdenes son opcionales y pueden ser ignoradas por el receptor.

Cada respuesta comienza con un código de tres dígitos, pudiendo ir seguida por información adicional. El primer dígito indica la categoría de la respuesta:

- **Respuesta de finalización positiva:** La acción solicitada se ha completado satisfactoriamente. Puede iniciarse una nueva solicitud.

- **Respuesta intermedia positiva:** La orden ha sido aceptada, pero la acción solicitada se encuentra suspendida, pendiente de la recepción de información adicional. El emisor SMTP debe enviar otra orden especificando esta información. Esta respuesta se utiliza en grupos de secuencias de órdenes.

- **Respuesta de finalización negativa transitoria:** La orden no se aceptó y la acción solicitada no se llevó a cabo. Sin embargo, la condición de error es temporal y puede solicitarse la acción de nuevo.

- **Respuesta de finalización negativa permanente:** La orden no se acepto y la acción solicitada no se realizó.

2.4 Intercambio de correo

El diseño de SMTP se basa en el modelo de comunicación mostrado en la figura siguiente.

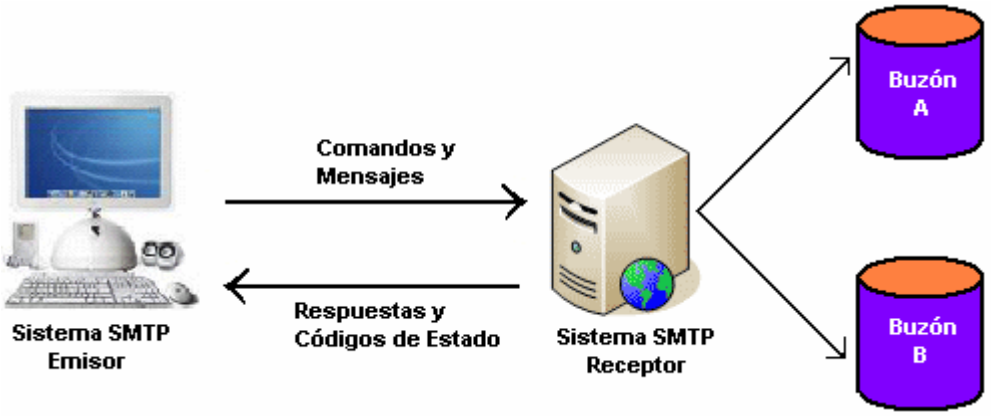


Figura 4.7Modelo empleado en SMTP para la transferencia de correo.

Como resultado de la solicitud de correo de un usuario, el emisor SMTP establece una conexión en los dos sentidos con el receptor SMTP. El receptor puede ser el destinatario final o un intermediario (pasarela de correo). El emisor generará comandos a los que replicará el receptor.

El flujo de transacciones que se lleva a cabo en SMTP se puede ver en el ejemplo siguiente. Para ello las líneas que inician con S corresponden al Servidor y las que comienzan con C al cliente.

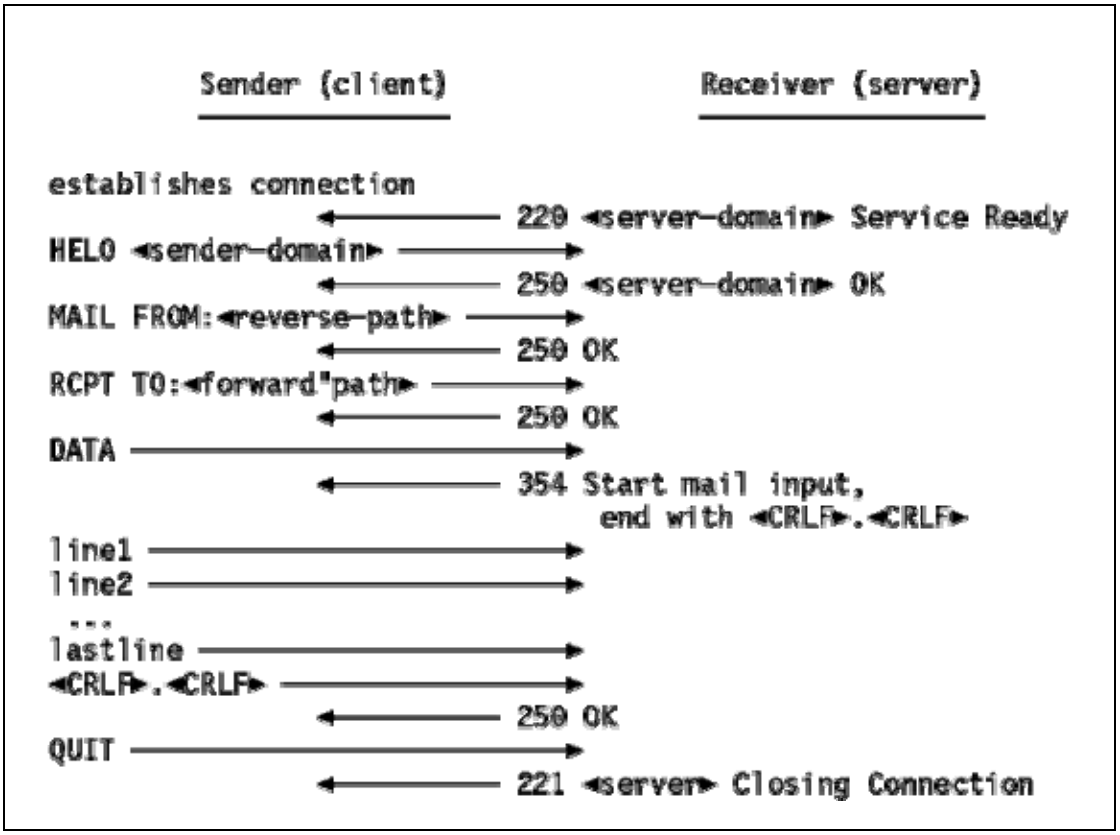


Figura 4.8 Ejemplo de una comunicación a través de comandos SMTP.

Todos los comandos, réplicas o datos intercambiados son líneas de texto, delimitadas por un <CRLF>. Todas las réplicas tienen un código numérico al comienzo de la línea.

El flujo mostrado en la figura anterior se describe a continuación:

1. El emisor SMTP establece una conexión TCP con el SMTP de destino y espera a que el servidor envíe un mensaje "220 Service ready" o "421 Service not available" cuando el destinatario es temporalmente incapaz de responder.

2. Se envía un HELO (abreviatura de "hello"), con el que el receptor se identificará devolviendo su nombre de dominio. El SMTP emisor puede usarlo para verificar si contactó con el SMTP de destino correcto. Si el emisor SMTP soporta las extensiones de SMTP definidas en el RFC 1651, puede sustituir el comando HELO por EHLO. Un receptor SMTP que no soporte las extensiones responderá con un mensaje "500 Syntax error, command unrecognized". El emisor SMTP debería intentarlo de nuevo con HELO, o si no puede retransmitir el mensaje sin extensiones, enviar un mensaje QUIT. Si un receptor SMTP soporta las extensiones de servicio, responde con un mensaje multi-línea 250 OK que incluye una lista de las extensiones de servicio que soporta.

3. El emisor inicia ahora una transacción enviando el comando MAIL al servidor. Este comando contiene la ruta de vuelta al emisor que se puede emplear para informar de errores. Nótese que una ruta puede ser más que el par buzób@nombre de dominio del host. Además, puede contener una lista de los hosts de encaminamiento. Si se acepta, el receptor replica con un "250 OK".

4. El segundo paso del intercambio real de correo consiste en darle al servidor SMTP el destino del mensaje(puede haber más de un receptor). Esto se hace enviando uno o más comandos RCPT TO:<forward-path>. Cada uno de ellos recibirá una respuesta "250 OK" si el servidor conoce el destino, o un "550 No such user here" si no.

5. Cuando se envían todos los comandos rcpt, el emisor envía un comando DATA para notificar al receptor que a continuación se envían los contenidos del mensaje. El servidor replica con "354 Start mail input, end with <CRLF>.<CRLF>". Nótese que se trata de la secuencia de terminación que el emisor debería usar para terminar los datos del mensaje.

6. El cliente envía los datos línea a línea, acabando con la línea <CRLF>. <CRLF> que el servidor reconoce con "250 OK" o el mensaje de error apropiado si cualquier cosa fue mal.

7. **Ahora hay varias acciones posibles:**

    - El emisor no tiene más mensajes que enviar; cerrará la conexión con un comando QUIT, que será respondido con "221 Service closing transmission channel".
    - El emisor no tiene más mensajes que enviar, pero está preparado para recibir mensajes(si los hay) del otro extremo. Mandará el comando TURN. Los dos SMTPs intercambian sus papelees y el emisor que era antes receptor puede enviar ahora mensajes empezando por el paso 3 de arriba.
    - El emisor tiene otro mensaje que enviar, y simplemente vuelve al paso 3 para enviar un nuevo MAIL.

2.5 SMTP y el DNS

Si la red usa el concepto de dominio, un SMTP no puede entregar simplemente correo a TEST.IBM.comando abriendo una conexión TCP con TEST.IBM.comando. Primero debe consultar al servidor de nombres para hallar a que host (en un nombre de dominio) debería entregar el mensaje.

Para la entrega de mensajes, el servidor de nombres almacena los RRs ("resource records") denominados MX RRs. Mapean un nombre de dominio a dos valores:

- **Un valor de preferencia**: Como pueden existir múltiples RRs MX para el mismo nombre de dominio, se les asigna una prioridad. El valor de prioridad más bajo corresponde al registro de mayor preferencia. Esto es útil siempre que el host de mayor preferencia sea inalcanzable; el emisor SMTP intenta conectar con el siguiente host en orden de prioridad.

- **Un nombre de host:** También es posible que el servidor de nombres responda con una lista vacía de RRs MX. Esto significa que el nombre de dominio se halla bajo la autoridad del servidor, pero no tiene ningún MX asignado. En este caso, el emisor SMTP puede intentar establecer la conexión con el mismo nombre del host.

El RFC 974 da una recomendación importante. Recomienda que tras obtener los registros MX, el emisor SMTP debería consultar los registros WKS(Well-Known Services) del host, y chequear que el host referenciado tiene como entrada WKS a SMTP.

Nota: Esto es sólo una opción del protocolo, aunque aparece en numerosas implementaciones.

```
fsc5.stn.mlv.fr. IN    MX 0  fsc5.stn.mlv.fr.
                 IN    MX 2  psfred.stn.mlv.fr.
                 IN    MX 4 mvs.stn.mlv.fr.
                 IN    WKS   152.9.250.150 TCP (SMTP)
```

Figura 4.9 Ejemplo de RRs MX.

En el ejemplo anterior, el correo para fsc5.stn.mlv.fr debería, por prioridad, ser entregado al propio host, pero en caso de que el host sea inalcanzable, el correo también podría ser entregado a psfred.stn.mlv.fr o a mvs.stn.mlv.fr (si psfred.stn.mlv.fr no se pudiera alcanzar tampoco).

3. Estructura y codificación de mensajes MIME

3.1 Introducción

En los primeros días de ARPANET, el correo electrónico consistía exclusivamente en mensajes de texto escritos en inglés y expresados en ASCII. En tal entorno, el RFC 822 hacía todo el trabajo: especificaba los encabezados, pero dejaba el contenido en mano del usuario. Hoy en día, en la red mundial de Internet, este método es inadecuado. Los problemas incluyen envío y recepción de:

- Mensajes en idiomas con acentos (por ejemplo, español, francés y alemán).
- Mensajes en alfabetos no latinos (por ejemplo, hebreo y ruso).
- Mensajes en idiomas sin alfabetos (por ejemplo chino y japonés).
- Mensajes que no contienen texto (por ejemplo, audio y video).

Se propuso una solución en el RFC 1341 y se actualizó en los RFCs 2045-2049. Esta solución, llamada MIME (Extensiones Multipropósito de Correo Intenet), se usa ampliamente.

3.2 Descripción General de MIME

La idea básica de MIME es continuar usando el formato RFC 822, pero agregar una estructura al cuerpo del mensaje y definir reglas de codificación para los mensajes no ASCII. Al no desviarse del 822, los mensajes MIME pueden enviarse usando los programas y protocolos de correo electrónico existentes. Todo lo que tiene que cambiarse son los programas emisores y receptores, lo que pueden hacer los mismo usuarios.

MIME define cinco nuevos encabezados de mensajes como se muestran en la tabla siguiente:

Encabezado	Significado
MIME-Version:	Identifica la versión de MIME
Content-Description:	Cadena de texto que describe el contenido
Content-Id:	Identificador único
Content-Transfer-Encoding:	Comose envuelve el mensaje para su transmisión
Content-Type:	Naturaleza del mensaje

Figura 3.10 Encabezados definidos por MIME.

3.2.1 MIME-Version:

Indica al agente de usuario del receptor, que se esta tratando con un mensaje MIME, así como la versión del MIME que usa. Se considera que cualquier mensaje que no contenga el encabezado MIME-Version: es un mensaje de texto normal en inglés y se procesa como tal.

3.2.2 Content-Description:

Es una cadena ASCII que dice lo que está en el mensaje. Este encabezado es necesario para que el destinatario sepa si vale la pena decodificar y leer el mensaje. Si la cadena dice “Foto de Ronaldo” y la persona que recibe el mensaje no es fanático de Ronaldo, el mensaje probablemente será descartado en lugar de decodificarlo para dar una foto de alta definición.

3.2.3 Content-Id:

Identifica el contenido; usa el mismo formato que el encabezado estándar Message-Id.

3.2.4 Content-Transfer-Encoding:

Indica la manera en que está envuelto el cuerpo para su a través de la red donde podría tener problemas con la mayoría de los caracteres distintos de las letras, números y signos de puntuación. Se proponen cinco esquemas más un escape hacia nuevos esquemas:

1. Codificación 7bits

Codificación 7bit significa que no se ha hecho ninguna codificación y que el cuerpo consiste en líneas de texto ASCII con longitud no mayor de 1000 caracteres.

2. Codificación 8bits

Codificación 8bit implica que las líneas son lo bastante cortas para el transporte SMTP, pero que puede haber caracteres no ASCII. Esta codificación sólo es posible en los agentes SMTP que soporten el SMTP Service Extension for 8bit-MIMEtransport("Extensión SMTP para transporte MIME de 8 bits"), descrito en el RFC 1652. En otro caso, las implementaciones de SMTP deberían poner el bit de orden superior a cero, de modo que la codificación 8bit no sería válida.

3. Codificación Binary

Indica que pueden aparecer caracteres no-ASCII y que las líneas pueden ser demasiado largas para el transporte SMTP(es decir, puede haber secuencias de 999 ó más caracteres sin una secuencia CRLF). En la actualidad no hay estándares para el transporte de datos binarios sin codificar en sistemas de correo de TCP/IP, por lo que el único caso en el que se puede usar codificación "binary" en un mensaje MIME en una red TCP/IP es en la cabecera de una parte externa de un cuerpo. Se podría usar en otros casos si MIME se empleara junto con otros mecanismos de transporte o con una extensión hipotética de SMTP.

4. Codificación Quote-Printable

Es la primera de las dos codificaciones propiamente dichas y su fin es mantener la máxima legibilidad de los ficheros de texto en su forma codificada.

- Representa los caracteres no fiables con la forma hexadecimal de sus valores ASCII.
- Para mantener la longitud de cada línea a 76 caracteres o menos introduce saltos de línea reversibles ("soft").

5. Codificación Base64

Esta codificación se destina a datos que no consisten principalmente en texto. Trata el flujo de entrada como un flujo de bits, reagrupando los bits en bytes más cortos, que luego rellena hasta 8 bits para traducirlos a caracteres fiables. Como se indicó en la sección previa, sólo hay 73 caracteres fiables, por lo que la longitud máxima utilizable de cada byte es de 6 bits, que se pueden representar con sólo 64 caracteres (de aquí el nombre "Base64"). Como tanto la entrada como la salida son flujos de bytes, la codificación se debe hacer en grupos de 24 bits (3 de entrada y cuatro de salida). El proceso se puede ver de la siguiente forma:

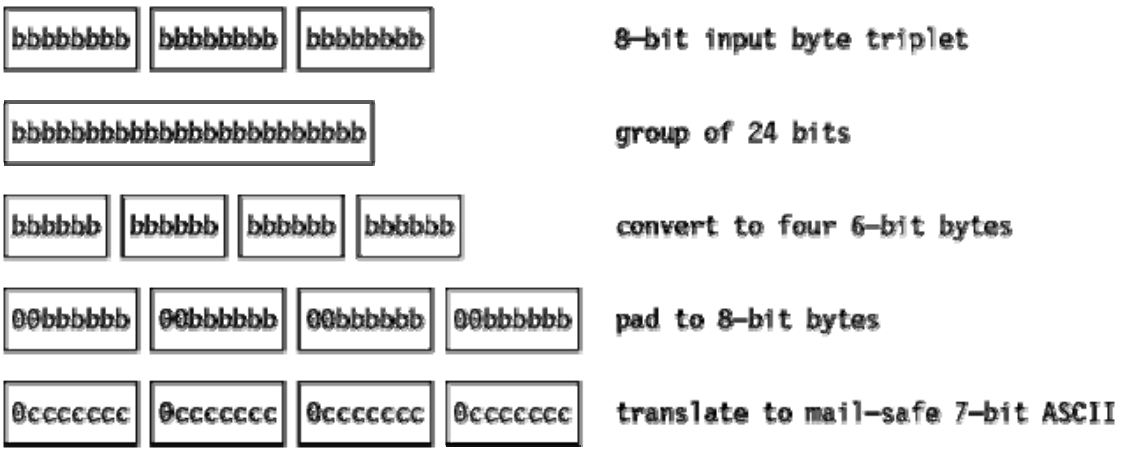


Figura 4.11Traducción empleada en la codificación Base64.

Se necesita un carácter adicional (el "=") para el relleno. Como la entrada es un flujo de bytes que se codifica en grupos de 24 bits, le podrán faltar 0, 8 ó 16 bits, al igual que a la salida. Si la salida tiene la longitud adecuada, no hace falta relleno. Si a la salida le faltan 8 bits, esto se corresponde con una salida de cuarto de dos bytes completos, un "short byte" y un byte faltante. El "short byte" se rellena con los 2 bits de orden inferior a cero. Los dos bytes faltantes se sustituyen con un carácter "=". Si a la salida le faltan 16 bits, esto se corresponde con una salida de cuarto de un byte completo, un "short byte" y dos bytes faltantes.

El "short byte" se rellena con los 6 bits de orden inferior a cero. Los dos bytes faltantes se sustituyen con un carácter "=". Si se utilizaron "cero caracteres", el agente receptor no sería capaz de decir al decodificar el flujo de entrada si X'00' caracteres de cola en la última o en las dos últimas posiciones eran datos orelleno. Con caracteres de relleno, el número de "="s (0, 1 o 2) da la longitud del flujo de entrada en módulo 3(0, 2 ó 1 respectivamente).

En resumen, los datos binarios deben de enviarse codificados en base64 o en forma entrecomillada imprimible.

3.2.5 Content-Type:

Especifica la naturaleza del cuerpo del mensaje. En el RFC hay siete tipos definidos, cada uno de los cuales tiene uno o más subtipos.

En la tabla siguiente pueden verse cada uno de estos tipos y subtipos.

Tipo	Subtipo	Descripción
Texto	Plano	Texto sin formato
	Enriquecido	Texto con comandos de formato sencillo
Imagen	Gif	Imagen fija en formato GIF
	TPEG	Imagen fija en formato JPEG
Audio	Básico	Sonido
Video	Mpeg	Película en formato JPEG
Aplicación	Octet-stream	Secuencia de bytes no interpretada
	Postscript	Documento imprimible en PostScript
Mensaje	Rfc822	Mensaje MIME RFC 822
	Parcial	Mensaje dividido para su transmisión
	Externo	El mensaje mismo debe obtener se de la red
Multipartes	Mezclado	Partes independientes en el orden especificado
	Alternativa	Mismo mensaje en diferentes formatos
	Paralelo	Las partes deben verse en forma simultanea
	Compendio	Cada parte es un RFC 822 completo

Figura 3.12 Tipos y subtipos MIME definidos en el RFC 2045.

Para una mejor comprensión, comentaremos brevemente los tres últimos tipos:

Aplicación

Es un tipo general para los formatos que requieren procesamiento externo no cubierto por ninguno de los otros tipos.

- **Octet-stream:** Este subtipo indica datos binarios generales consistentes en bytes de 8 bits. Es además el subtipo que un lector de correo debería asumir al encontrarse un tipo o subtipo desconocidos. Se permite cualqier parámetro, y el RFC menciona dos: "type=", para informar al receptor del tipo general y "padding=" para indicar un flujo de bits codificado en un flujo de bytes(su valor es el número de ceros añadidos para alinear el flujo a un número entero de bytes).Se recomienda que las implementaciones ofrezcan al usuario la opción de utilizar los datos como entrada a un programa de usuario o de almacenarlos en un fichero(no hay estándar para el nombre por defecto de tal fichero, aunque el RFC 1521 menciona un campo "Content-Disposition:" a ser definido en un RFC posterior.

- **PostScript:** Aunque suele pensarse que PostScript es un formato de impresión, es un lenguaje de programación y el uso de un intérprete de PostScript para procesar el tipo "application/PostScript" es una amenaza potencial a la seguridad. Cualquier lector de correo que interprete automáticamente programas de PostScripts es equivalente, en principio, a uno que ejecute automáticamente los programas ejecutables que recibe. El RFC 1521 perfila las posibles consecuencias.

Mensaje

Permite que un mensaje esté encapsulado por completo dentro de otro. Este esquema es útil para reenviar por ejemplo correo electrónico.

- **rfc822:** El mismo cuerpo es un mensaje encapsulado con la sintaxis de un mensaje RFC 822. Sin embargo, a diferencia de los mensajes de "alto nivel" de RFC 822, no tiene que tener el formato mínimo "From:", "To:" y una cabecera de destino al menos.

- **parcial:** Este tipo se usa para permtir la fragmentación de correos grandes de forma análoga a la fragmentación IP. Debido a que los agentes SMTP pueden

imponer límites superiores al tamaño del correo, puede que sea necesario enviarlos en fragmentos. La finalidad de este campo es que la fragmentación sea transparente al receptor.

- **Externo:** Puede usarse para mensajes muy grandes (por ejemplo, películas en video). En lugar de incluir el archivo MPEG en el mensaje, se da una dirección FTP y el agente de usuario del receptoe puede obtenerlo a través de la red al momento que se requiera. Esta característica es especialmente útil cuando se envía una película a una lista de correo, y solo se espera que unos cuantos la vean.

Multiparte

Permite que un mensaje contenga más de una parte, con el comienzo y el fin de cada parte, claramente delimitados. Posee cuatro subtipos:

- **Mezclado:** Permite que cada parte sea diferente, sin ninguna estructura adicional impuesta, por ejemplo los archivos adjuntos.

- **Alternativa:** Permite que el mensaje se incluya varias veces pero expresado en dos o más medios diferentes. Por ejemplo un mensaje puede enviarse expresado en ASCII, texto enriquecido y como PostScript. El agente de usuario intentará presentarlo en PostScrip de ser posible, la segunda posibilidad sería como texto enriquecido y por último, en ASCII. Las partes deben ordenarse de l más sencilla a la más compleja para ayudar a los receptores con agentes de usuario pre-MIME puedan encontrarle algún sentido al mensaje.

Form: denis@yahoo.es
To: fatima@hotmail.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@yahoo.es>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklñzxcvbnm
Subject: Feliz Cumpleaños

Este es el preambulo. El agente de usuario lo ignora.
--qwertyuiopasdfghjklñzxcvbnm--
Content-Type: text/enriched

Feliz cumpleaños a ti
Feliz cumpleaños a ti
Feliz cumpleaños, querida <bold> Fatima </bold>
Feliz cumpleaños a ti

--qwertyuiopasdfghjklñzxcvbnm--
Content-Type:message/external-body;
access-type=”anon-ftp”
site=”felicidades.yahoo.es”
directory=”pub”
name=”cumpleaños.snd”

Content-Type: audio/basic
Content-Tansfer-Encoding: base64
--qwertyuiopasdfghjklñzxcvbnm--

Figura 3.13 Mensaje multiparte que contiene alternativas de texto enriquecido y audio.

- **Paralelo:** Se usa cuando todas las partes deben verse de manera simultanea. Por ejemplo, las películas con frecuencia tienen una canal de audio y uno de video.

- **Compendio:** Se usa cuando se empacan muchos mensajes juntos en un mensaje compuesto.

4. Protocolos de Entrega Final (POP3 e IMAP)

4.1 Introducción

Hasta ahora hemos supuesto que todos los usuarios trabajan en máquinas capaces de enviar y recibir correo electrónico. Como vimos, el correo electrónico se entrega al hacer que el emisor establezca una conexión TCP con el receptor y después que envíe el correo electrónico a través de ella. Este modelo funcionó bien por décadas cuando todos los hosts ARPANET (y más tarde Internet) se pusieron de hecho, en línea todo el tiempo para aceptar conexiones TCP.

Sin embargo con el advenimiento de personas que acceden a Internet llamando a su ISP por medio de un modem, ese modelo dejó de usarse. El problema aparecía por la imposibilidad de establecer conexiones TCP cuando el receptor no estaba conectado.

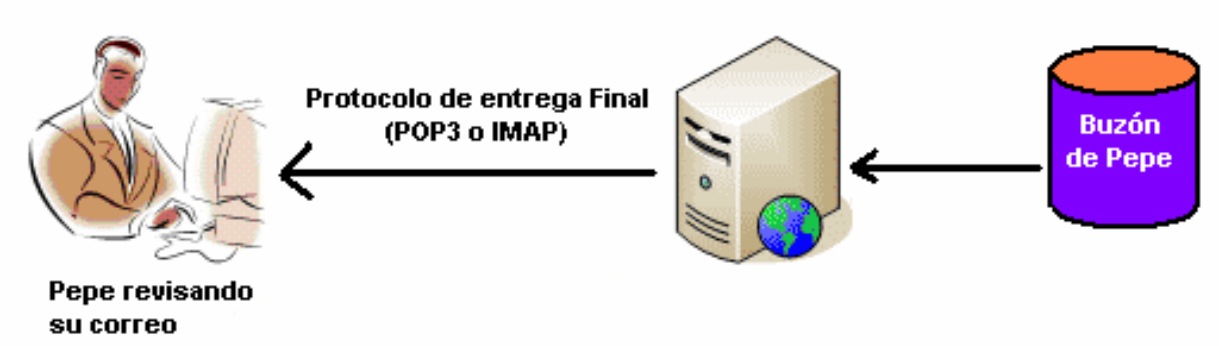


Figura 4.14 Acceso al Servidor de correo empleando POP3 o IMAP.

Una solución es que un agente de transferencia de mensajes en una máquina ISP acepte correo electrónico para sus clientes y lo almacene en sus buzones en una máquina ISP. Puesto que este agente puede estar en línea todo el tiempo, el correo electrónico puede enviarse en cualquier momento.

4.2 POP3 (Post Office Protocolo Versión 3)

4.2.1 Introducción

En informática se utiliza el Post Office Protocol (POP3) en clientes locales de correo para obtener los mensajes de correo electrónico almacenados en un servidor

remoto. La mayoría de los suscriptores de los proveedores de internet acceden a sus correos a través de POP3.

POP3 inicia cuando el usuario arranca el lector de correo. Éste llama al ISP (a menos que ya haya una conexión) y establece una conexión TCP con el agente de transferencia de mensajes ene. Puerto 110. Una vez que se ha establecido la conexión, el protocolo POP3 pasa por tres estados en consecuencia:

- **Autorización:** Tiene que ver con el inicio de sesión por parte del usuario.
- **Transacción:** Se relaciona con el hecho de que el usuario colecte los mensajes de correo electrónico y los marque para eliminación desde el buzón.
- **Actualización:** Se encarga de que los mensajes de correo electrónico se eliminen realmente.

4.2.2 Ordenes

Para establecer una conexión a un servidor POP, el cliente de correo abre una conexión TCP en el puerto 110 del servidor. Cuando la conexión se ha establecido, el servidor POP envía al cliente POP una invitación y después las dos máquinas se envían entre sí otras órdenes y respuestas que se especifican en el protocolo. Como parte de esta comunicación, al cliente POP se le pide que se autentifique (Estado de autenticación), donde el nombre de usuario y la contraseña del usuario se envían al servidor POP.

Si la autenticación es correcta, el cliente POP pasa al Estado de transacción, en este estado se pueden utilizar órdenes LIST, RETR y DELE para mostrar, descargar y eliminar mensajes del servidor, respectivamente. Los mensajes definidos para su eliminación no se quitan realmente del servidor hasta que el cliente POP envía la orden QUIT para terminar la sesión. En ese momento, el servidor POP pasa al Estado de actualización, fase en la que se eliminan los mensajes marcados y se limpian todos los recursos restantes de la sesión.

Puedes conectarte manualmente al servidor POP3 haciendo Telnet al puerto 110. Es muy útil cuando te envían un mensaje con un fichero muy largo que no quieres recibir.

Orden	Significado
USER <nombre>	Identificación de usuario (Solo se realiza una vez).
PASS <password>	Envías la clave del servidor.
STAT	Da el número de mensajes no borrados en el buzón y su longitud total.
LIST	Muestra todo los mensajes no borrados con su longitud.
RETR <número>	Solicita el envío del mensaje especificando el número (no se borra del buzón).
TOP <número> <líneas>	Muestra la cabecera y el número de líneas requerido del mensaje especificando el número.
DELE <número>	Borra el mensaje especificando el número.
RSET	Recupera los mensajes borrados (en la conexión actual).
QUIT	Salir.

Figura 4.14 Comandos empleados en el protocolo POP.

Si bien es cierto el protocolo POP3 soporta la capacidad den descargar un mensaje específico o un conjunto de mensajes y dejarlos en el servidor, la mayoría de los programas de correo electrónico simplemente descarga todo y vacía el buzón, con lo que la única copia queda en el disco duro del usuario.

4.3 IMAP (Internet Message Access Protocol)

4.3.1 Introducción

Para un usuario que tiene una cuenta de correo electrónico con un ISP que siempre se accede desde una PC, POP3 es adecuado y se utiliza ampliamente debido a su sencillez y robustez. Sin embargo, es un axioma de la industria de la computación el de que siempre que algo funciona bien, alguien comienza a pedir más características. Eso también ocurrió con el correo electrónico. Por ejemplo, muchas personas tienen una sola cuenta de correo electrónico en el trabajo o en la escuela y desean accederla desde el trabajo, desde la PC de su casa, desde su computadora portátil cuando están de viajes de negocios y desde algún cibercafé cuando se encuentran de vacaciones.

Aunque POP3 permite esto, debido a que descargas todos los mensajes almacenados en cada contacto, el resultado es que los mensajes de correo electrónico del usuario quedan esparcidos rápidamente en múltiples máquinas, más o menos de manera aleatoria, y algunos de ellos ni siquiera en la máquina del usuario.

Esta desventaja dio lugar a un protocolo de entrega final alternativo, IMAP (Protocolo de Acceso a Mensajes de Internet), que se define en el RFC 2060.

4.3.2 Ventajas sobre POP3

- **Soporte para los modos de operación connected y disconnected:** Al utilizar POP3, los clientes se conectan al servidor de correo brevemente, solamente lo que les tome descargar los nuevos mensajes. Al utilizar IMAP4, los clientes permanecen conectados el tiempo que su interfaz permanezca activa y descargan los mensajes bajo demanda. Esta forma de trabajar de IMAP4 puede dar tiempos de respuesta más rápidos para usuarios que tienen una gran cantidad de mensajes o mensajes grandes.

- **Soporte para la conexión de múltiples clientes simultáneos a un mismo destinatario:** El protocolo POP3 asume que el cliente conectado es el único dueño de una cuenta de correo. En contraste, el protocolo IMAP4 permite accesos simultáneos a múltiples clientes y proporciona ciertos mecanismos a los clientes para que se detecten los cambios hechos a un mailbox por otro cliente concurrentemente conectado.

- **Soporte para acceso a partes MIME de los mensajes y obtención parcial:** Casi todo el correo electrónico de Internet es transmitido en formato MIME. El protocolo IMAP4 le permite a los clientes obtener separadamente cualquier parte MIME individual, así como obtener porciones de las partes individuales o los mensajes completos.

- **Soporte para que la información de estado del mensaje se mantenga en el servidor:** A través de la utilización de banderas definidas en el protocolo IMAP4 de los clientes, se puede vigilar el estado del mensaje, por ejemplo, si el mensaje ha sido o no leído, respondido o eliminado. Estas banderas se almacenan en el servidor, de manera que varios clientes conectados al mismo

correo en diferente tiempo pueden detectar los cambios hechos por otros clientes.

- **Soporte para accesos múltiples a los buzones de correo en el servidor:** Los clientes de IMAP4 pueden crear, renombrar o eliminar correo (por lo general presentado como carpetas al usuario) del servidor, y mover mensajes entre cuentas de correo. El soporte para múltiples buzones de correo también le permite al servidor proporcionar acceso a los directorios públicos y compartidos.

- **Soporte para búsquedas de parte del servidor:** IMAP4 proporciona un mecanismo para que los clientes pidan al servidor que busque mensajes de acuerdo a una cierta variedad de criterios. Este mecanismo evita que los clientes descarguen todos los mensajes de su buzón de correo con el fin de agilizar las búsquedas.

- **Soporte para un mecanismo de extensión definido:** Como reflejo de la experiencia en versiones anteriores de los protocolos de Internet, IMAP define un mecanismo explícito mediante el cual puede ser extendido. Se han propuesto muchas extensiones de IMAP4 y son de uso común.

4.4 Características de la Entrega

Independientemente de si POP3 o IMAP4, muchos sistemas proporcionan ganchos para procesamiento adicional de correo electrónico entrante. Una característica especialmente valiosa para muchos usuarios de correo electrónico es la capacidad de establecer filtros. Éstas son reglas que se verifican cuando llega el correo electrónico o cuando se inicia el agente de usuario.

Cada regla especifica una condición y una acción. Por ejemplo, una regla podría decir que cualquier mensaje del jefe va al buzón número 1, cualquier mensaje de un grupo seleccionado de amigos va al buzón número 2, y cualquier mensaje que contenga en la línea Asunto ciertas palabras objetables se descarta sin ningún comentario.

Algunos ISPs proporcionan un filtro que clasifica de manera automática el correo electrónico como importante o como publicidad no deseada. Otra característica de entrega que por lo general se proporciona la capacidad de reenviar (temporalmente) correo entrante a una dirección diferente, la cual puede ser una computadora que maneja un servicio de localización.

También podría instalarse un demonio de vacaciones. Éste es un programa que examina cada mensaje entrante y envía al emisor una respuesta grabada.

TEMA 5: VOZ SOBRE IP

Objetivos

- Estudiar las principales características de la de la telefonía IP.
- Describir el protocolo de transporte RTP empleado para las comunicaciones en tiempo real.
- Comparar el protocolo H.323 y el protocolo SIP para la señalización de llamadas.

Contenido del Capítulo

- Características de VoIP.
- Transporte de Información en Telefonía IP.
- Señalización de Control de Llamadas.

Bibliografía

- Andrews S. Tanenbaum, Redes de Computadores, Cuarta Edición 2003.
- Davidson, Jonathan, Fundamentos de voz sobre IP, 2001.
- Telefonía IP, estado de arte, D. Manuel Moreno Martín, Revista AHCIET año XIX – Nº 86 Enero/Marzo 2001.
- Una primera aproximación al protocolo SIP, D. Manuel Moreno Martín, Revista AHCIET año XX – Nº 91 Abril/Junio 2002.

1. Características de VoIP

1.1 Introducción

Si bien todavía existen numerosas barreras técnicas que superar hasta alcanzar niveles de calidad de servicio y de seguridad en telefonía IP similares al servicio clásico de telefonía de conmutación de circuitos, las perspectivas que se abren respecto a la incuestionable coexistencia que deben mantener las redes telefónicas ya establecidas, Internet y las numerosas redes corporativas basadas en tecnología IP ya instaladas y que se están instalando y que se están instalando, justifican la investigación y desarrollo de proyectos de trabajos dirigidos hacia la integración de las redes telefónicas (telefonía móvil incluida) y las redes IP, mediante el desarrollo de elementos de interconexión (pasarelas) entre ambas, así como también la propia telefonía IP.

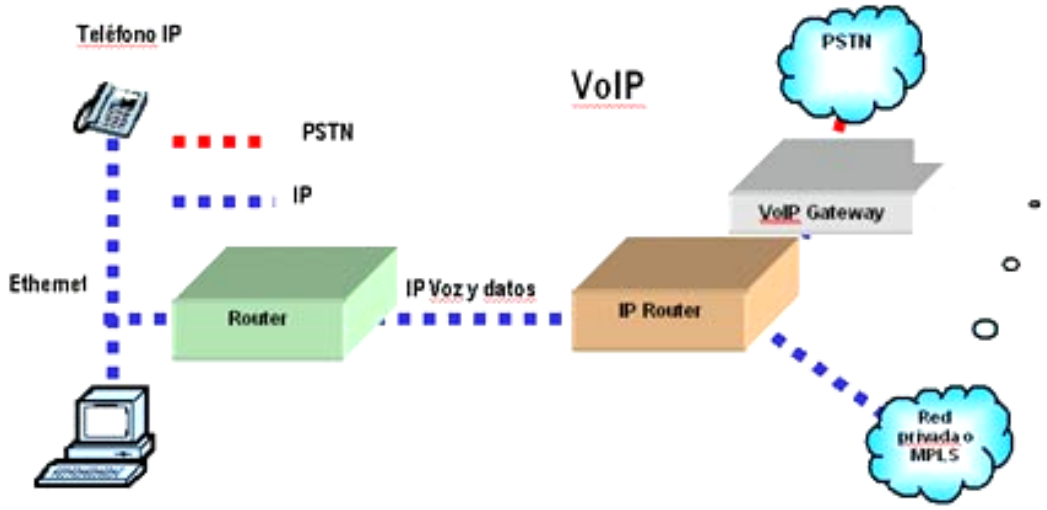


Figura 5.1 Transmisión de datos y voz sobre redes IP.

1.2 Voz sobre IP (VoIP)

Para cursar tráfico de voz a través de redes de paquetes como Internet, se requiere la paquetización de voz previamente digitalizada, por cuanto son los paquetes las unidades de datos que se transportan por las redes IP. También, y dada las limitaciones de recursos de ésta, es necesario reducir el consumo de ancho de banda mediante compresión de datos.

Existen diferentes factores que limitan las prestaciones de las redes para cursar paquetes en tiempo real como son las fluctuaciones de demora, jitter, perdida de paquetes y de secuencia.

1.3 Telefonía IP (IPTEL)

La telefonía IP, denominación abreviada de algo más abarcador como son las comunicaciones multimedia sobre redes de datagramas, implica la presencia de cierto número de “medios” (voz, datos, video, etc.) que comparten los enlaces de comunicación , lo que exige una conveniente codificación de dichos medios en aras de oibtener eficiencia de transmisión y almacenamiento, es decir, una mejor compartición de recursos con calidad aceptable. La pila de protocolos relativos a telefonía IP es:

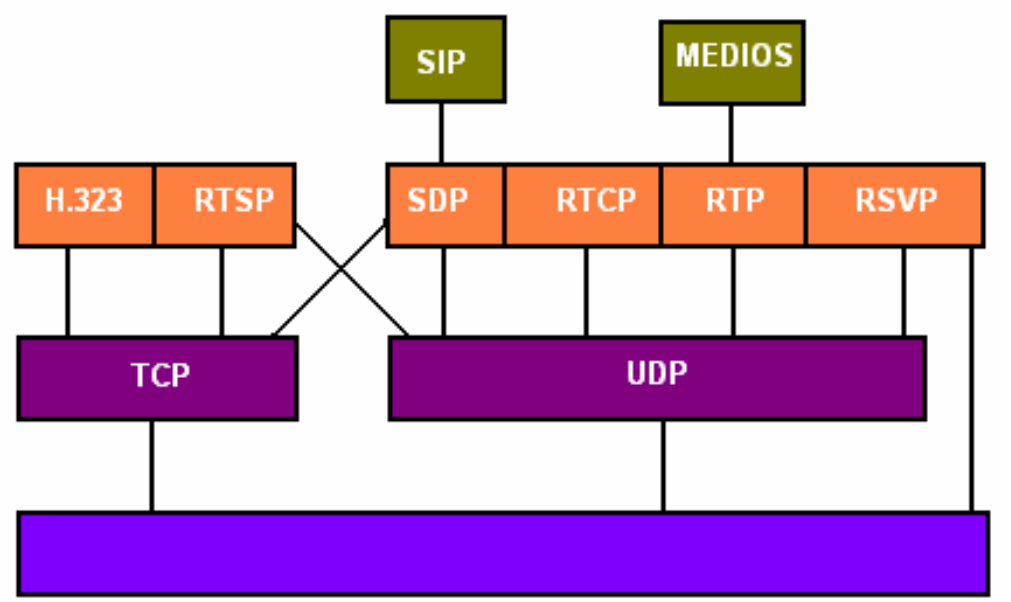


Figura 5.2 Arquitectura de Protocolos de Telefonía IP.

1.4 Codificación de la voz

Las prestaciones de los codificadores de voz (vocoders) dependen de:

- Velocidad de la señal codificada.
- Demora de procesamiento.
- Complejidad del algoritmo de procesamiento, habiendo fuerte interdependencia entre estos factores.

Así es conveniente una baja velocidad resultante de la codificación, por las implicaciones que tiene el ancho de banda y en la compartición de recursos, pero esto se traduce en mayor complejidad en la implementación de los vocoders y mayor demora en el desarrollo de su tarea, a lo que se adiciona inferior calidad en la reproducción de la voz.

Por otra parte, la combinación de otros medios con la voz sugiere tener en cuenta los "silencios" de la conversación, lo que en principio posibilita una mejor explotación de del ancho de banda, así es común el uso de algoritmos de compresión del silencio en la codificación de voz (una mala codificación del silencio, conlleva a un deterioro significativo en la inteligibilidad de la voz). Esto a su vez requiere algoritmos de detección de la actividad de la voz (VAD) y algoritmos de generación de ruido de confort (CNG), posibilitando así que la señal de voz codificada pueda transmitirse a una velocidad no uniforme en función de nivel de actividad que presente la fuente generadora.

En línea con las limitaciones propias de las redes de datagramas para cursar tráfico con requerimientos de tiempo real, se ha hecho necesario el desarrollo de normas o recomendaciones, que permiten enfrentarse satisfactoriamente a este problema. En tal sentido UIT ha publicado tres normas de codificación.

- **G.723.I:** Establece un vocoder para comunicaciones multimedia a 6,3 y 5,3 kbps, con una demora de codificación de 37,5 mseg. Inicialmente fue concebida para videotelefonía de baja velocidad como parte del estándar H.324.

- **G.729:** Establece un vocoder a 8 kbps con una demora de codificación de 15 mseg. Originalmente pensada para entornos inalámbricos, pero es aplicable a entornos IP y comunicaciones multimedia. Presenta nivel de complejidad mayor que G.723.I

- **G. 729A:** Es una versión A de G.729, con menor grado de complejidad y prestaciones que ésta, diseñada para integración de voz y datos. La codificación se hace a una velocidad de 8 kbps con una demora de 15 mseg. Presenta menos requerimientos que G.729 en cuanto a capacidad de procesamiento.

Estas tres recomendaciones presentan, hasta ahora, las mejores características de cara a la telefonía IP.

1.5 Prestaciones de los codificadores

Para evaluar las prestaciones de los codificadores de voz (vocoders) se puede acudir a diferentes parámetros, entre ellos la demora.

La demora en la codificación-decodificación de voz es un parámetro crítico para lograr buenas prestaciones de la comunicación telefónica a través de de redes de almacenamiento y reenvío, como Internet. En la codificación de la voz la demora está delimitada por tres factores:

1.5.1 Demora de Trama:

La demora de trama es consecuencia del modo de operación de los vocoders, pues requieren, para llevar a cabo el procesamiento, una trama completa de muestras de voz, o incluso más de una trama. En consecuencia esto requiere un “buffering”, lo que a su vez implica cierta demora inevitable, independiente de la propia implementación del vocoders.

1.5.2 Demora de Procesamiento:

Es el tiempo que toma el codificador en “analizar” la voz, esto es, las muestras de voz, más el tiempo que consume el codificador en reconstruirla. Es independiente del hardware.

La demora de trama y la demora de procesamiento sumadas corresponden a la demora del vocoder en un sentido de transmisión, cuyo valor debe ser:

Si no hay eco, no más de 400 mseg, pero es preferible que no rebase los 200 mseg. Si hay eco, no más de 20 – 25 mseg, lo que conlleva generalmente el uso de canceladores de eco.

1.5.3 Demora de Puente

Ésta se presenta en los casos en que se llevan a cabo múltiples codificaciones (codificaciones en cascada). Este es el caso, por ejemplo de las MCU (Unidades de

Control Multipunto) utilizadas en la arquitectura funcional de H.323, donde cada "stream" de voz debe ser decodificado en la MCU, se combinan las señales decodificadas, y luego la señal resultante se codifica, lo que da como resultado el incremento de la demora, se dobla en valor, y se reduce la calidad de voz.

1.6 Retardo y Perdida de Paquetes

La calidad de la transferencia de señales con requerimientos de tiempo real, esto es, con tiempo limitado de "viaje" por exigencias de la interactividad como es el caso de las señales de voz a través de redes de paquetes modo datagramas, se ve afectada de manera considerable cuando los recursos de la red no son capaces de responder en tiempo a estas exigencias.

Con el aumento del tráfico y por tanto la carga de la red, las demoras se incrementan, pudiendo rebasar los límites de tolerables para una buena calidad de comunicación de voz. A este factor de aleatorio se suma también la carencia de recursos que la red puede presentar en determinados momentos de alta demanda, provocando la perdida de paquetes contentivos de información de voz.

Al retardo que irremediablemente una red de datagramas se suman los retardos propios de los procesos codificación-decodificación y los retardos variables que las redes de datagramas introducen.

Con vista a reducir las fluctuaciones del retardos de paquetes (jitter), pues necesario obtener una cadencia de reproducción en los receptores similar a la de origen, en estos se emplean tapones (bufferes), lo que a su vez adiciona más demora o retardo a los paquetes.

En consecuencia, el proceso de transferencia de paquetes de voz a través de redes de datagramas lleva implícito un retardo fijo, predecible, más un retardo no predecible, digamos aleatorio que puede estar en el orden de 75 a 300 mseg.

En el caso de redes IP existe la posibilidad de transferencia de paquetes de voz en modo fiable y seguro. Para ello habría que utilizar un protocolo como TCP, pero las demoras que su modo de operación implica no lo hacen apropiado para manejar

tráfico sensible a demoras. En tal sentido, se acude al protocolo UDP, no fiable pero si con menos garantías en cuanto a las necesidades de tiempo real.

Para paliar la no fiabilidad del protocolo UDP, en algunos codificadores es posible admitir cierta tolerancia a las perdidas aleatorias de paquetes, pero no ráfagas de errores, en cuyo caso se acude a la técnica FEC (Forward Error Correction), con la redundancia añadida que esto implica.

A los retardos antes señalados se adicionan los denominados retardos de acceso, asociados a la plataforma a través de la cual se accede a la red IP, y a aquellos elementos funcionales que sirven de “puente” entre ésta y las redes telefónicas convencionales.

2. Transporte de Información en Telefonía IP

2.1 Introducción

Ya antes se ha comentado la necesidad de “preparar” l información de usuario a transportar por la red IP, por ejemplo la voz para Telefonía IP, imágenes en una videoconferencia, etc. Es decir, la codificación, que incluye la compresión y desempaquetado. Una vez que se obtiene el formato adecuado para esto, cabe la siguiente pregunta:

¿Cómo mover por la red IP dichos paquetes contentivos de información sensible a las demoras, a las pérdidas, al desorden?

Es evidente que se requiere un protocolo o conjunto de ellos que se ocupe de de esta tarea y garantice la entrega de los paquetes de “medios” en tiempo real, o casi en tiempo real, siempre y cuando el deterioro de la calidad no sobrepase lo humanamente admisible.

En relación a esto hay que señalar que la red IP no solo mueve a través de ella paquetes de información de usuario, sino que también mueve, paquetes de señalización.

El termino señalización en Telefonía IP tiene una connotación más amplia que el concepto correspondiente en las redes clásicas de Telecomunicaciones modo circuito, pues se trata ahora no solo de la señalización necesaria para el control de llamadas en el sentido clásico (establecimiento, mantenimiento-modificación, terminación), sino también señalización para QoS. Entonces en correspondencia con esto, hay que hacer la distinción entre transporte de de medios, en alusión a la información del usuario y transporte de información de control, es decir, señalización.

2.2 Transporte de Medios

El transporte de medios (voz, video, datos, etc.) por la red IP no solo se traduce en mover por lamisma secuencia de bits empaquetados, sino que también se requiere adicionarle a éstos cierta información de control, por ejemplo:

- Tipo de información transportada.
- Números de secuencia.
- Marcas de tiempo.

De manera que esto supone una forma de formatear los paquetes IP y hacerlos aptos parta el transporte de información con requerimientos de tiempo real. El protocolo RTP (Real Time Transport Protocol) fue concebido para compensar el jiter y la pérdida de secuencia de los paquetes que introducen las redes IP, pudiendo ser empleado para el transporte de streams de datos sensibles al tiempo como son la voz y el video.

Igualmente necesario, con funciones de motorización de la calidad de servicio y otras, está el protocolo RTCP (Real Time Control Protocol), utilizado frecuentemente con RTP.

Los protocolos RTP y RTCP no ejercen ningun tipo de influencia en ls condiciones de la red IP, no controlan la calidad de servicio, solo posibilitan que los receptores puedan resolver apropiadamente las perturbaciones a que son sometidos los paquetes IP con contenidos en tiempo real al atravesar la red.

El transporte de medios exige cumplir una serie de requisitos, a saber:

- Caracterización de la carga útil, su identificación, como ha sido codificad, su formato, cuantas tramas por paquete, etc.

- Información de tiempo que posibilite su reconstrucción temporal en el receptor o los receptores, y la sincronización de diferentes flujos de medios en el tiempo.

- Secuenciamiento de los paquetes que constituyen los streams de medio, para posibilitar su reproducción en orden correcto y la detección de paquetes perdidos.

Todo esto, posibilita la gestión adecuada de los medios en el receptor.

2.3 Real Time Transport Protocol (RTP)

RTP posibilita el transporte de datos isócronos a través de redes de paquetes, ha sido optimizado para transferencias con requisitos en tiempo real, tanto unicast como multicast, soportando servicios extremo a extremo.

RTP no garantiza reserva de recursos ni da garantía de calidad de servicio, tampoco provee ningún mecanismo para garantizar la entrega en tiemp, y asume que la red es segura y entrega los paquetes en secuencia. Su papel fundamental es actuar como una interfaz entre aplicaciones en tiempo real y los protocolos de la capa de transporte, sin establecer que protocolo de ésta usar , aunque típicamente se soporta sobre UDP.

Los paquetes RTP transportan streams de datos sensibles a ls demoras de la red, a sus fluctuaciones, a sus perdidas de paquetes, e información adicional que posibilite su recuperación y entrega adecuada.

2.3.1 Cabecera RTP

Los paquetes RTP están constituidos por dos partes, esto e, una cabecera y un cuerpo. La cabecera a su vez tiene varios campos que están presentes en todo paquete RTP, y un campo eventual. La cabecera RTP transporta información de control relativa al contenido del cuerpo del paquete, es decir referente a la información de usuario (audio, video, etc.) por ejemplo segmentos de conversación de una longitud entre 10 y 30 mseg. Los campos de esta cabecera son:

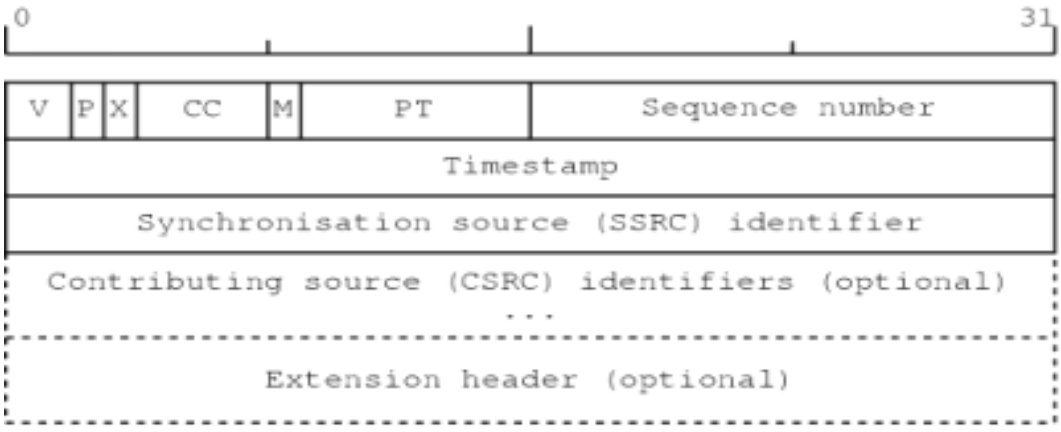


Figura 5.3 Cabecera RTP.

- **V:** 2 bits reservados para identificar la versión del protocolo.

- **P:** 1 bit de relleno. Es un indicador relativo a si el paquete RTP ha sido o no ajustado en un tamaño con proposito de alineamiento, en relación con el campo Payload (carga útil).

- **X:** 1 bit indicador de la extensión de la cabecera del paquete más alla del campo eventual CSRC.

- **CC:** 4 bits de contabilización del número identificadores CSRC que siguen la parte fija de la cabecera. Usualmente es cero.
- **M:** 1 bit de marca. Su uso se define or el perfil RTP. Por ejemplo H.225 plantea que para codificación de audio con supresión de silencio, el bit M debe activarse a 1 lógico en el primer paquete de cada ráfaga de voz después de un intérvalo de silencio.

- **PT:** 7 bits que identifican el tipo de carga útil que transporta el paquete RTP.

- **Nº de secuencia:** 16 bits para identificar cada paquete. Se inicializa con un valor aleatorio, incrementándose en cada paqute de una sesión RTP.

- **Marca de tiempo (timestamp):** 32 bits. Es el valor de reloj de muestreo para el primer bit de la carga útil. La frecuencia de reloj se define para cada tipo de carga útil. El reloj se inicializa con un valor aleatorio.

- **SSRC:** 32 bits. Identificador de fuente de muestreo, identifica la fuente de un stream RTP. Todos los paquetes RTP con SSRC común, tienen referencias de tiempo y secuencias comunes.

- **CSRC:** 32 bits. Identificador de fuentes contribuyentes, uno o varios, y sólo presentes si hay mezcla de flujo de varias fuentes. Representan el listado de los identificadores de las fuentes (SSRC) que contribuyen eventualmente a un stream de datos que tiene su propio SSRC. No se emplea en H.323.

2.4 Real Time Control Protocol (RTCP)

RTCP es un protocolo de control que acompaña al protocolo RTP para la transferencia entre fuentes y destinos streams multimedia. Proporciona a los participantes en una sesión información de calidad de la transmisión en la red, posibilitando así adaptar las fuentes al estado de la red.

RTCP se basa en la transmisión periódica de paquetes de control a todos los participantes en la sesión, empleando el mismo mecanismo de distribución que los paquetes de datos, por lo que se debe proveer multiplexación entre paquetes de datos y paquetes de control RTCP, p.e., mediante el empleo de puertos diferentes el protocolo UDP.

RTCP es imprescindible en sesiones multicast, como realimentación de como reciben los streams de datos los destinatarios posibilitándose así homogeneizar la calidad. En comunicaciones unicast, no es necesario pero si es útil su uso. En resumen, RTCP desarrolla fundamentalmente las funciones siguientes:

Suministro de información respecto a la calidad de la distribución de los datos, esto es, la “realimentación” antes apuntada lo que posibilita a la fuente de los datos:

- Aplicar control de flujo.
- Aplicar control de congestión.
- Aplicar codificación adaptativa.
- Diagnosticar fallos en la distribución multicast.

Controla la velocidad de emisión de paquetes RTP, para que éste sea escalable en cuanto al número de participantes.

Como se infiere, RTCP debe ser utilizado para todos los participantes de una sesión RTP, enviando cada uno de ellos paquetes RTCP periódicamente, y estos períodos deben adaptarse al númerod de fuentes RTCP para evitar una avalancha de este tráfico de control, y evitar así las consecuencias negativas de saturación o congestión de la red. Es común dejar una pequeña parte (típicamente un 5%) del ancho de banda asignado a RTP para RTCP, y se reparte entre todos los participantes.

2.5 Adaptaciones

Si bien RTP y RTCP ayudan a compensar algunas de las imperfecciones de la redes IP para el trasnporte de tráfico en tiempo real, no son suficiente para obtener la calidad de servicio que se necesita en estos casos. Por ello se deben de introducir adaptaciones para contrarrestar esos problemas contra la calidad de servicio como por ejemplo las demoras las cuales, en un sistema se dividen en:

- Demoras introducidas por los dispositivos de red (routers), las colas en los mismos y la demora de procesamiento.

- Demoras en las pasarelas que median entre las redes telefónicas y la redes IP, como resultado da las tareas que estas desempeñan.

- Demoras en los terminales de VoIP En este último caso, en el terminal del emisor se requiere por las propias exigencias de las redes IP, desarrollar tareas tales como codificación, conformación de tramas y añadir redundancia, y en el terminal receptor la contraparte correspondiente, todo lo cuial agrefa más demora extremo a extremo.

En consecuencia, estas demoras, si no se controlan y compensan de alguna manera, dan al traste con la calidad de servicio provocando:

- Perdida de la interactividad en la comunicación.
- Problemas de eco en comunicación de voz, aunque el origen de éste no está precisamente en la demora, aunque ésta lo hace patente cuando se rebasan ciertos límites.

Otros impedimentos que deterioran la calidad de servicio, y contra los que también se toman medidas, o se hacen adaptaciones son el jiter, las perdidas de paquetes y las perdida también de la secuencia de datos.

2.6 Buffers

Las fluctuaciones de la demora de la red IP en la entrega de los paquetes de información de usuario a los destinatarios se compensa utilizando en los receptores

una memoria “tampón” o “buffer”, cuyo llenado se hace al ritmo de llegada de los paquetes RTP, por el vaciado se hace al ritmo de que posibilite compensar las fluctuaciones.

La capacidad de almacenamiento del Buffer anti-jitter incide en la demora total, y también en la recuperación de paquetes, así, buffers más grandes posibilitan absorber mayores fluctuaciones en la recepción de paquetes y mayor cantidad de estos, a expensas de introducir mayor demora extremo a extremo. Este proceso de re-sincronización de paquetes introduce demora, pero es necesario en el marco de VoIP, dado el impredecible comportamiento de la red IP en cuanto a la demora y fluctuaciones de ésta se refiere.

Dadas las condiciones no estables de las redes IP “best effort” ya señaladas , lo ideal es disponer de buffers anti-jitter dinámicos, cuyas capacidades se adapten a als condiciones concretas de la rd en cada momento. Frente a condiciones de red estables indudablemente que buffer anti-jitter estáticos simplifican las cosas. Estos buffers también contribuyen a la solución de otros problemas que se presentan en as redes IP, por ejemplo frente a la pérdida de paquetes.

3. Señalización de control de llamadas

3.1 Introducción

Para soportar el servicio de VolP se requiere, además de los protocolos para el transporte de la información de usuario en tiempo real, también de la señalización correspondiente, es decir; los protocolos necesarios que garanticen el establecimiento, mantenimiento-modificación y terminación de llamadas de voz sobre las redes Ip, esto es, señalización de control de llamadas. También se requiere señalización para: QoS, seguridad, control de medios, etc.

Para ser competitivo y se pueda establecer este servicio comercialmente, es necesario alcanzar en la tecnología de VolP niveles de servicio y calidad de los mismos en correspondencia con los que brindan las redes clásicas de Telecomunicaciones modo circuito, aunque no es descartable que aún con niveles por debajo de éstas se logren establecer por lo económico que resultan. Así, se han desarrollado diferentes soluciones para la problemática de la señalización de control de llamada en sistemas de VolP, a saber:

- Modelo H.323 (solución UIT).
- Modelo SIP (solución lETF),

En definitiva, son tres soluciones diferentes a la misma problemática, la señalización de control de llamada para el servicio de VolP, cada una con una arquitectura funcional y protocolos que las caracterizan.

Por las propias características de las redes Ip, esto es, la no asignación de recursos en exclusiva para cada comunicación, sino una compartición máxima de los mismos, con toda la flexibilidad que esto representa, se hace necesario que los sistemas de VolP requieran señalizar; con los protocolos adecuados, todo el control de la comunicación, por ejemplo:

- Negociar el tipo de codificador a utilizar.
- Negociar los parámetros de empaquetado de la voz (y vídeo).
- Negociar la calidad de servicio con la red (demoras, pérdida de paquetes).

- Intercambiar número de puertos a través de los que se llevará a cabo la comunicación, etc.

La señalización de VoIP, además de garantizar los servicios clásicos que brindan las redes telefónicas, deben sumar la potencialidad que en sí mismas presentan las redes IP de cara a la integración de servicios de voz, datos y vídeo, así como también dar solución a la problemática particular de manejar tráfico con requerimientos de tiempo real sin que la QoS se deteriore más allá de lo admisible.

El flujo de la información de usuario y el flujo de señalización siguen trayectorias diferentes en su paso por las redes IP, pero no solo este es el hecho que debemos destacar; sino también que en la señalización intervienen elementos funcionales que no intervienen en el manejo de los paquetes de la información de usuario. La voz (información de usuario) y la señalización no presentan los mismos requerimientos de transporte por la red, la voz tiene que ser tratada con demora y jitter mínimos, pues pierde "valor" con el tiempo, dados sus requerimientos de tiempo real, en cambio la señalización carece de esto último.

También es de destacar que los "momentos" de la señalización no tienen coincidencia total con los "momentos" de la conversación, en lo fundamental la señalización precede y sucede a la conversación, y según el caso también puede coincidir en el tiempo con ésta, es decir; que durante el trasiego por la red de paquetes de voz también se transfieran paquetes de señalización de una comunicación ya establecida.

En definitiva, el tráfico de información de usuario (voz, etc.) es tratado por la red I P de manera diferente a como lo hace con el tráfico de señalización.

3.2 Modelo H.323

La recomendación H.323 de UIT (versiones 1, 2, 3 y 4) describe terminales, equipos y servicios para comunicaciones multimedia sobre redes de paquetes sin garantía de calidad de servicio, tanto para sesiones multipunto como punto a punto. La comunicación entre terminales H.323 se puede desarrollar en un simple segmento, o a través de múltiples segmentos LAN, incluyendo Internet, aunque esto último puede deteriorar las prestaciones. H.323 también define la señalización necesaria para comunicaciones multimedia sobre redes IP, aunque también puede soportarse sobre

redes IPx/SPX o redes ATM. Para el transporte de medios utiliza los protocolos RTP/RTCP del IETF.

Los terminales y equipos H.323 soportan aplicaciones con requerimientos de tiempo real como son la voz y el vídeo, así como aplicaciones de datos, y combinaciones de ellas (p.e., videotelefonía). Los terminales H.323 pueden ser terminales explícitamente diseñados para ese fin, como es el caso de videoteléfonos, o pueden estar integrados en peso

En H.323 el soporte de voz es obligatorio, no así el vídeo y los datos que son opcionales. Posibilita usar más de un canal de cada tipo al mismo tiempo.

Entre otras, H.323 incluye las siguientes recomendaciones:

- **H.225.0:** Paquetización, sincronización, señalización.
- **H.245:** Control de canal.
- **H.261 Y H.263:** Codificación de vídeo.
- **G.71 1, G.722, G.723. I , G.728, G.729:** Codificación de audio.
- **T.120:** Conferencias de datos en tiempo real, punto a punto y multipunto. Provee interoperabilidad en los niveles de aplicación, de transporte y de red.

El procedimiento según H.245 posibilita que el contenido de cada canal lógico se describa cuando el canal se abre, así transmisor y receptor pueden "dialogar" y "acordar" el modo de comunicación a utilizar.

Otras recomendaciones que también forman parte del estándar H.323 son:

- **H.450.x:** Servicios suplementarios. Define la señalización y los procedimientos utilizados para proveer servicios similares a los telefónicos.
- **H.235:** Define requerimientos de seguridad para proveer autentificación y cifrado en sistemas H.323.
- **H.332:** Para la provisión de conferencias a gran escala basadas en H.323

3.2.1 Elementos Funcionales de la Plataforma H.323

Los elementos funcionales de la plataforma H.323 son:

1. Gatekeepers (GK's)

Los gatekeepers son entidades de control y señalización, siendo los elementos funcionales más complejos en el marco H.323. Proveen aquellos servicios que no pueden ser descentralizados e implementados en los "endpoints" (terminales, pasarelas, MCU 's).

Las funciones que lleva a cabo un gatekeeper son:

- Control de acceso y administración de recursos, autorización de llamadas.
- Traducción (mapping) de direcciones de transporte entre direcciones IP, alias y números E.164.
- Gestión del ancho de banda.
- Control de señalización.
- Gestión de zonas H.323.
- CDR (call details record).
- Gestión de llamadas.

Para desarrollar estas funciones, entre el gatekeeper y un endpoint se emplea el protocolo RAS (Registration/ Admission / Status) sobre UDP. RAS forma parte de H.225.0.

La señalización de control de llamada puede o no estar soportada a través del gatekeeper. Si ésta se soporta, entonces el gatekeeper debe "hablar" el protocolo Q.93 I (sobre TCP) según establece H.225.0. Igualmente, también es posible que el protocolo de control de canal (H.245, sobre TCP) se enrute a través del gatekeeper, pues éste puede actuar como proxy entre endpoints.

Un gatekeeper y sus endpoints definen una zona H.323, de manera que en entornos LAN's es suficiente un gatekeeper; pero en otros entornos como Internet no basta con uno, sino que se requerirán varios, cada uno definiendo una zona H.323. Lógicamente, entre gatekeepers se requerirá comunicación. Actúa como el punto central para todas las llamadas en una zona, comportándose de cierta manera como un conmutador virtual.

Si bien el gatekeeper no es obligatorio su empleo en un entorno H.323, sí posibilita emplear más eficientemente la plataforma H.323, por ejemplo mediante el

enrutamiento de llamadas a su través. Son entidades funcionales lógicamente separadas de los endpoints H.323, pero es posible incluir funcionalidades gatekeepers en entidades físicas como son las gateways y las MCU' s.

Funciones Gatekeeper básicas (obligatorias):

- Traducción de direcciones: entre direcciones de transporte y alias de direcciones.
- Control de admisión: controla el acceso sobre la base de diferentes criterios, p.e. autorización, ancho de banda, etc. Puede no existir; esto es, se admiten todas las solicitudes de acceso.
- Control del ancho de banda: puede basarse en la gestión del ancho de banda. Puede no estar presente, admitiéndose todas las solicitudes de ancho de banda.
- Gestión de Zona H.323: se refiere a las funciones anteriores para los terminales, gateways y MCU 's registrados en su zona de control.

Gateways (GW's, pasarelas)

Las pasarelas (gateways) sirven de "puente" entre redes H.323 y otras redes, por ejemplo con las redes de conmutación de circuitos de telefonía. Desarrollan la "traducción" de la señalización, información de control e información de usuario, según requerimientos de las redes que interconecta, posibilitando así la interoperabilidad entre redes, terminales y servicios, haciendo viable la integración de servicios aún con plataformas dispares (p.e., PSTN y redes IP).

A diferencia de los Gatekeepers, las Gateways si cursan información de usuario, soportada ésta en RTP/UDP/IP. Desarrollan las funciones siguientes:

- Traducción de formatos de transmisión.
- Traducción de procedimientos de comunicación (p.e., entre H.245 y H.242).
- Transcodificación de audio y vídeo.
- Establecimiento y liberación de llamadas.

En general, las gateways, "vistas" desde cada una de las redes que interconecta, se comportan como un elemento más de las mismas, es decir; p.e., vista

desde la red telefónica se comporta como una central de conmutación más de ésta, y vista desde una red IP como un nodo más de ésta.

Evidentemente, dada su funcionalidad, las gateways son un elemento funcional opcional en entornos H.323, sólo son necesarias cuando se requiere interconexión entre entornos H.323 y entornos no H.323:

- Gateways H.320, para redes N-ISDN.
- Gateways H.321, para redes B-ISDN.
- Gateways H.322, para redes LAN con calidad de servicio.
- Gateways H.324, para redes GSTN e inalámbricas, posibilitándose así la interoperabilidad entre terminales multimedia H.323 y terminales multimedia asociados a otras redes.

Transporte de medios

H.323 emplea el protocolo RTP para transportar los medios, y como ya ha sido apuntado, es un mecanismo para el soporte integrado de voz, vídeo y datos, proporcionando además información de control necesaria para la recuperación en tiempo real de los medios en el receptor (secuencia y fechado correcto de los paquetes). RTP no garantiza reserva de recursos ni da garantía de calidad de servicio.

Su papel fundamental es actuar como un interface entre aplicaciones de tiempo real y los protocolos de la capa de transporte, sin establecer qué protocolo de ésta usar; aunque típicamente se soporta sobre UDP.

Con los protocolos RTP y RTCP se tiene un mecanismo de transferencia de medios con requerimientos de tiempo real a través de redes de paquetes.

Protocolos de señalización:

Para propósitos de señalización H.323 dispone de los mecanismos necesarios para el manejo de comunicaciones multimedia en redes de datagramas IP, a saber:

- Localización de usuarios, establecimiento de sesión.
- Negociación de sesión.
- Gestión de participantes, servicios suplementarios.

Para desarrollar toda esta señalización y las prestaciones que se derivan, H.323 incorpora los protocolos enmarcados en las recomendaciones H.225.0 (RAS, Q,93 1) Y H.245. El protocolo RAS solo se utiliza si está presente el gatekeeper.

2. Terminales H.323 (figura 3)

Un terminal H.323 posibilita comunicaciones bidireccionales en tiempo real de voz, vídeo y datos. La recomendación establece que el audio es obligatorio, no así el video y los datos que son opcionales. H.323 especifica los modos de operación requeridos para que los terminales de audio, vídeo y/o datos trabajen conjuntamente. Las funciones de control propiamente que implementan se establecen en las especificaciones:

- H.245 para negociación del canal.
- H.225.0 (Q.93 1) para señalización y control de llamada.
- H.225.0 (RAS) para comuncación con el Gatekeeper.

También implementan los protocolos RTP/RTCP para el manejo de los flujos de audio y vídeo. Deben soportar el estándar G.711 para codificación y compresión de voz, cualquier otro es opcional (G.722, G.723, G.728, G.729). Si soporta vídeo (opcional), entonces debe implementar la codificación según la norma H.261 (H.263 es opcional). Igualmente, la capacidad de manejar conferencias de datos es opcional, pero si se soporta es sobre la base de la especificación T. 120 de UIT

Dado el jitter que sufren los paquetes IP en su viaje por la red, y las consecuencias negativas de esto para el tráfico de audio y video, en el terminal H.323 se requiere un buffer de recepción para absorber; en la medida de lo posible, estas fluctuaciones en la demora de los paquetes IP, anulando o reduciendo el efecto negativo que el jitter puede producir en flujos de información de usuario con requerimientos de tiempo real.

3.2.2 Arquitectura de protocolos

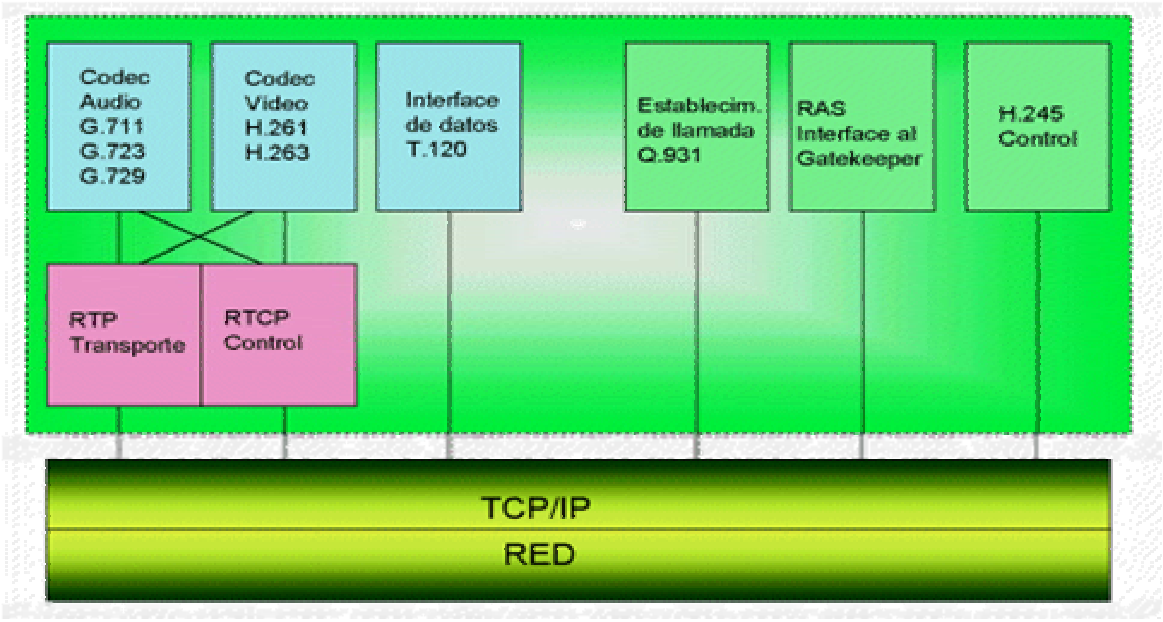


Figura 5.4 Arquitectura de protocolos del nivel de aplicación de H.323.

En la figura 5.4 se muestra el conjunto de protocolos del nivel de aplicación asociados a la Recomendación H.323, así como los estándares de codificación de medios contemplados en la misma.

H.323 se soporta en la suite del modelo Internet, por tanto los protocolos que comprende para comunicaciones multimedia se sustentan en TCP/IP y UDP/IP. Los medios con requerimientos de tiempo real que precisan no solo una entrega correcta en el destino, sino también una entrega a tiempo (pierden su valor con el tiempo), se encapsulan dentro del protocolo UDP, evitando así las demoras y pérdidas del throughput que puede introducir la solución TCP a pesar de ser más segura.

La codificación de estos medios se lleva a cabo según los estándares G.7xx para el audio (G.71 I al menos, y opcionalmente G.722, G.723, G.728, G.729) Y H.26x para el vídeo (H.261, Y opcionalmente H.263). Tanto uno como otro de estos medios se encapsulan en el protocolo RTP, concebido fuera del ámbito UIT para el transporte de flujos de estas características. Asociado a RTP está el protocolo de control RTCP Tanto RTP (información de medios) como RTCP (información de control de medios) se encapsulan en datagramas UDP

Para los medios de datos H.323 emplea a nivel de aplicación el protocolo T 120, Y éste se encapsula en el protocolo de nivel de transporte TCP, pues estos medios no presentan requerimientos de tiempo real, posibilitándose así aprovechar el factor de seguridad que adiciona TCP en relación con UDP

Los protocolos de control comprendidos en H.323, unos se encapsulan en UDP, es el caso del protocolo H.225.0 (RAS, Registration-Admission-Status) que se desarrolla entre el gatekeeper y los endpoints H.323, y otros sobre TCP como son H.225.0 (Q.93 1) para el control de llamada y H.245 para el control de canal.

3.2.3 Llamadas H.323

El establecimiento de una llamada H.323 se lleva a cabo en tres fases:

- **Fase RAS:** Intercambio de mensajes entre gatekeeper y endpoint, para traducción de direcciones, autorización (o rechazo) de llamadas y gestión de ancho de banda, básicamente.

- **Fase Q.93 1:** Intercambio de mensajes entre endpoints para el establecimiento de conexiones lógicas

- **Fase H.245:** Intercambio de mensajes entre endpoints para "acordar" el intercambio de información de usuario.

Considerando el papel que juegue el gatekeeper en las llamadas H.323, se puede hablar de dos modelos:

- Modelo de llamada H.323 directa (direct routed model).
- Modelo de llamada H.323 indirecta (gatekeeper routed model).

En el primer caso la entidad funcional gatekeeper solo interviene en las fase RAS (inicial, intermedia si es necesaria, y final), esto es, en la fase de registro y admisión ("alta" de la llamada) que necesariamente precede toda llamada, y en la fase opuesta a ésta al final de la llamada ("baja" de la llamada). Eventualmente, durante el transcurso de una llamada se puede desarrollar una actualización o reasignación de

recursos para la misma, lo que requiere un intercambio de mensajes RAS entre endpoints y gatekeeper.

En el modelo de llamada indirecta, el gatekeeper no solo interviene en las fase RAS, sino también en la fase Q.931.

La fase H.245 normalmente se desarrolla sin la intervención del gatekeeper, aunque pudiera actuar también como proxy para esta fase.

A continuación de estas tres fases de establecimiento de llamada, se lleva a cabo la transferencia de información de usuario (audio, vídeo, datos, según el caso) por medio de los protocolos RTP/RTCP, según lo acordado en la fase H.245, previa apertura de los canales lógicos entre los endpoints. Estos canales lógicos son unidireccionales, por lo que para una comunicación bidireccional se requiere abrir uno en cada dirección de transmisión. Incluso, es posible la apertura de más de un canal lógico al mismo tiempo. En la transferencia de medios no interviene el gatekeeper, pues sólo es una entidad de señalización, sino que se lleva a cabo directamente entre los endpoints.

A continuación, se muestra, de forma resumida, el intercambio de mensajes correspondientes a cada una de las fases comentadas en una llamada punto a punto.

ARQ y ACF son mensajes RAS relativos a la solicitud de llamada:

- ARQ: admission request.
- ACF: admission confirm.

Otros mensajes RAS son:

- GRQ: gatekeeper request.
- GCF: gatekeeper confirm.
- RRQ: registration request.
- RCF: registration confirm.
- LRQ: location request.
- BRQ: bandwith request.
- BCF: bandwith confirm.

3.3 Modelo SIP (Sesion Initiation Protocol)

En este artículo se presenta una breve aproximación al protocolo de señalización SIP (Session Initiation Protocol) del IETF, mucho más sencillo en su concepción y funcionamiento que la solución ITU equivalente, pero también con menor grado de normalización y penetración en el mercado que ésta.

Con SIP es posible implementar, sobre redes IP, servicios telefónicos básicos y avanzados. Su sencillez resulta muy atractiva, y con un futuro prometedor, en tanto que se vislumbra su utilización para el soporte de otros tipos de servicios y también en los sistemas de Tercera Generación de Comunicaciones Móviles.

3.3.1 Aspectos generales

SIP (RFC 2543) es un protocolo de control del nivel de aplicación, concebido para la señalización y el control de llamadas, esto es, para el control del establecimiento, modificación y terminación de sesiones o llamadas multimedia.

SIP forma parte de las especificaciones del IETF para comunicaciones multimedia, conjuntamente con otros protocolos como RSVP (Resource reServation Protocol), RTP (Real Time Protocoll, SDP (Session Description Protocoll, etc., pero su funcionalidad no depende de ninguno de éstos. Soporta comunicaciones entre usuarios de redes IP, y también, con el empleo de pasarelas, con usuarios de otras redes, por ejemplo, con terminales de las redes telefónicas convencionales.

Múltiples transacciones SIP pueden ser soportadas en una simple conexión TCP, o en una asociación SCTP posiblemente. En cambio, un simple datagrama UDP solo porta un mensaje SIP. Los datagramas UDP, incluyendo todas las cabeceras, no deben ser mayores que la MTU (Media Transmisión Unit) del trayecto, si es que ésta se conoce, o no mayores de 1 500 bytes si no se conoce.

SIP es un protocolo basado en texto, utiliza el juego de caracteres ISO 10646 con codificación UTF-G RFC 2279), lo que posibilita una fácil implementación y depuración , lo hace flexible y extensible. El sobre encabezamiento que implica usar un protocolo basado en texto no tiene aquí mayor trascendencia, pues SIP es un protocolo de señalización, no es un protocolo para el trasiego de datos de usuario.

SIP soporta, entre otras cosas:

- Correspondencia ("mapping") de nombres y servicios de re direccionamiento.
- Movilidad de usuarios y de terminales.
- Llamadas con múltiples interlocutores a través de MCU's (Multipoint Control Unit).
- Servicios suplementarios y de red inteligente.
- Identificación de usuarios con URI's (Uniform Resource Identifier), posibilitando a éstos iniciar una llamada haciendo un simple "cIick" sobre un enlace web.
- Operación stateless o stateful.
- Cualquier sintaxis en el cuerpo de sus mensajes.
- Ser utilizado conjuntamente con otros protocolos de señalización,
- Sus propios mecanismos de seguridad.

Genéricamente, las comunicaciones multimedia con el protocolo SIP comprenden:

- Localización del usuario.
- Determinación de la disponibilidad del usuario para la comunicación.
- Determinación de los medios para la comunicación, esto es, las capacidades de usuario.
- Establecimiento de los parámetros de la comunicación entre las partes involucradas.
- Manipulación de las llamadas, es decir, establecimiento, modificación y terminación de llamadas.

3.3.2 Elementos Funcionales

Los elementos funcionales en la arquitectura SIP son:

- Agentes de Usuario (UA).
- Servidores de red.

Los Agentes de Usuario son aplicaciones que residen en las estaciones terminales SIP, y contienen dos componentes: Agentes de Usuario Clientes (User Agent Client, UAC) y Agentes de Usuario Servidores (User Agent Server, UAS).

Los UAC originan las solicitudes SIP (asociados al extremo que origina la llamada), y los UAS responden a estas solicitudes, es decir, originan respuestas SIP (asociados al extremo que recibe la llamada). Los UA's deben implementar el transporte tanto sobre TCP como sobre UDP, y quizás también sobre SCTP.

Los UACs y UAS's pueden, por si solos y sin los servidores de red, ser capaces de soportar una comunicación básica (modelo de llamada básico, directamente entre endpoints). No obstante, la potencialidad de SIP se aprovecha con el empleo de los servidores de red.

Los servidores de red se clasifican, desde un punto de vista lógico, de la manera siguiente:

- Servidores de redirección.
- Servidores Proxy.
- Servidores de registro

Servidores de redirección (Redirect Server):

Procesan mensajes INVITE, que son solicitudes SIP, y retornan la dirección (o direcciones) de la parte llamada, esto es, el SIP - URL (Uniform Resource Locator) de la parte llamada, o cómo contactar con ella (respuesta 3xx). De lo contrario rechaza la llamada, enviando una respuesta de error (error de cliente 4xx o error de servidor 5xx). Desarrollan una funcionalidad similar al Gatekeeper H.323 cuando en la solución ITU se emplea el modelo de llamada directo.

Servidores proxy (Proxy Server):

Corren un programa "intermediario" que actúa como servidor y como cliente, respecto al llamante se comporta como servidor, y respecto al llamado como cliente. Un servidor proxy puede re-enviar solicitudes hasta el destino final sin efectuar cambio alguno en ellas, o cambiar alguno de sus parámetros si se requiere, por ejemplo, en el caso de las cabeceras "Vio" "Record Route".

Los servidores proxy desarrollan el "routing" de los mensajes de solicitudes y respuestas SIP. Pueden ser "stateful" o "stateless"

Los servidores proxy stateful retienen información de la llamada durante el tiempo que dure el establecimiento de ésta, no así los servidores proxy stateless, los que procesan un mensaje SIP y entonces "olvidan" todo lo referente a la llamada hasta que vuelven a recibir otro mensaje SIP asociado a la misma. Esto se refiere al "estado" de la llamada, sin embargo, pueden mantener un "estado" para una simple transacción SIP, lo que es denominado "minimal state".

La implementación stateless provee buena escalabilidad, pues los servidores no requieren mantener información referente al estado de la llamada una vez que la transacción ha sido procesada. Además, esta solución es muy robusta dado que el servidor no necesita "recordar" nada en relación con una llamada.

Sin embargo, no todas las funcionalidades pueden ser implementadas en un servidor proxy stateless, por eiemplo, las funcionalidades relativas a la contabilización y facturación de las llamadas puede requerir funcionalidades proxy stateful, de manera que se le pueda "seguir el rastro" a todos los mensajes y estados de una comunicación.

Servidores de registro (Registrar Server):

Registran las direcciones SIP (SIP-URL) y sus direcciones IP asociadas, es decir, garantizan el "mapping" entre direcciones SIP y direcciones IP. Típicamente están localizados con servidores Proxy o servidores de redirección.

Es un servidor que acepta solo mensaies de solicitud REGISTER, posibilitando el registro correspondiente a la localización actual de los usuarios, esto es, "seguir el rastro" de los usuarios pues por diferentes razones (conexión vía ISP, usuarios móviles, conexión vía LAN con DHCP) las direcciones IP de éstos puede cambiar.

También se les denomina servidores de localización (Location Server), pues son utilizados por los servidores proxy y de redirección para obtener información respecto a la localización o localizaciones posibles de lo parte llamada.

Ahora bien, en rigor, los Location Server (LS) no son servidores SIP, ni entidades SIP, sino bases de datos, que pueden formar parte de arquitecturas de comunicaciones que utilicen SIP. Entre un LS y un servidor SIP no se utiliza el

protocolo SIP, por ejemplo, en ocasiones se emplea entre éstos el protocolo LDAP (Lightweight Directory Access Protocol).

La información registrada en los servidores de registro, esto es, el registro del mapping de direcciones SIP correspondiente a un usuario, no es permanente, requiere ser "refrescado" periódicamente, de lo contrario, vencido un "time out" (por defecto, una hora), el registro correspondiente será borrado. Este valor por defecto del "time out" puede ser modificado según valor que se especifique en la cabecera "Expires" de un mensaje de solicitud REGISTER. En consecuencia, para mantener la información de registro, el terminal (o el usuario) necesita refrescar lo periódicamente. Igualmente, un registro vigente puede ser cancelado y/o renovado por el usuario.

Usualmente, un servidor de red SIP implemento uno combinación de los diferentes tipos de servidores SIP ya comentados: servidor proxy + servidor de registro y/o servidor de redirección + servidor de registro. En cualquier caso deben implementar el transporte sobre TCP y UDP.

3.3.3 Mensajes

Los mensajes SIP, solicitudes (métodos) y respuestas (códigos de estado), emplean el formato de mensaje genérico establecido en la RFC 822, esto es:

- Una línea de inicio.
- Uno o más campos de cabeceras (header).
- Una línea vacía (indica final del campo de cabeceras) • Cuerpo del mensaje (opcional).

1. Solicitudes (Métodos) SIP:

El formato de los mensajes de solicitud es como sigue:
Request-Li ne
* (general-header I request-header
I entity-header)
CRLF
[message-body]

La línea "Request-Line" tiene la estructura siguiente: **Method** SP **Request-URL** SP **SIP- Version** CRLF, donde "SP" es el carácter "espacio" y "CRLF" es la secuencia "retorno del carro" y "nueva línea". Veamos cada uno de estos tres componentes de la línea "Request-Line".

Método (Method):

Son seis los métodos básicos definidos en SIP, en su núcleo (RFC 2543), a saber: INVITE, ACK, OPTIONS, BYE, CANCEL, REGISTER. Otros métodos, ya normalizados o en fase draft, están más allá de la RFC 2543, es decir, son extensiones al núcleo básico de SIP, y no se presentan aquí.

- **INVITE:** Invita a un usuario, o servicio, a participar en una sesión. El cuerpo del mensaje contiene, generalmente, una descripción de la sesión.

- **ACK:** Confirma que el cliente solicitante ha recibido una respuesta final desde un servidor a una solicitud INVITE, reconociendo la respuesta como adecuada. Solo para reconocer solicitudes INVITE, y no otros mensajes de solicitud.

- **OPTIONS:** Posibilita "descubrir" las capacidades del receptor.

- **BYE:** Finaliza una llamada, o una solicitud de llamada. Puede ser enviado por el agente llamante o por el agente llamado.

- **CANCEL:** Cancela una solicitud pendiente, pero no afecta una solicitud ya completada. Este método finaliza una solicitud de llamada incompleta.

- **REGISTER:** Se utiliza este método como un servicio de localización que registra la localización actual de un usuario.

Los métodos que no sean soportados por servidores, proxy o de redirección, son tratados por éstos como si se tratase de un método OPTION, y en consecuencia reenviados. Los métodos que no sean soportados por los servidores UAS o Registrar, provocan el mensaje de respuesta 501, "no implementado".

Request-URI:

Es un SIP-URL o un URI (Uniform Resource Identifier). Indica el usuario o servicio al que la solicitud está siendo dirigida. A diferencia de la cabecera "To", el "Request-URI" puede ser reescrito por servidores proxy.

Típicamente, el UAC establece el "Request-URI" y la cabecera "To" al mismo SIP-URL, pero puede no ser así. Los servidores proxy o de redirección pueden usar la información en los campos Request-URI y cabeceras de solicitud para manejar la solicitud, y posiblemente re-escribir el Request-URI. Los servidores proxy son responsables del procesamiento del Request-URI.

SIP-Version:

Da cuenta de la versión del protocolo SIP en uso, y se incluye tanto en mensajes de solicitud (métodos) como en mensajes de respuesta (códigos de estado).

2. Respuestas (Códigos de estado) SIP:

Después que se recibe e interpreta un mensaje de solicitud SIP, el receptor del mismo (servidor SIP) responde con un mensaje (o varios) de respuesta (código de estado) El formato de los mensajes de respuesta es como sigue:

Status-Une
(general-header
I response-header
I entity-header)
CRLF
[message-body]

La línea "Status-Line" comprende:

SIP-Versión SP Status-Code SP Reason-phrase, CRLF donde "SP" es el carácter "espacio" y "CRLF" es la secuencia CRLF.

SIP-Version: Versión del protocolo SIP.

Status-Code: Código de tres enteros para ser interpretado por máquinas. Indica el resultado de comprender y satisfacer o no una solicitud. Hay seis tipos diferentes de Status-Code:

- **1xx:** Informativo. Solicitud recibida, se continua para procesar la solicitud. Por ejemplo, 180, RINGING.

- **2xx:** Solicitud exitosa. La solicitud (acción) fue recibida de forma adecuada, comprendida y aceptada. Por ejemplo, 200, OK.

- **3xx:** Redireccionado. Más acciones deben ser consideradas para completar la solicitud. Por ejemplo, 302, MOVED TEMPORARILY.

- **4xx:** Error de cliente. La solicitud contiene malla sintaxis o no puede ser resuelta en este servidor. Por ejemplo, 404, NOT FOUND.

- **5xx:** Error de servidor. El servidor ha errado en la resolución de una solicitud aparentemente válida. Por ejemplo, 501, NOT IMPLEMENTED.

- **6xx:** Fallo global. La solicitud no puede ser resuelta en servidor alguno. Por ejemplo, 600. BUSY EVERYWHERE.

Los mensajes respuestas 2xx, 3xx, 4xx, 5xx y 6xx son "respuestas finales", y terminan lo transacción SIP. En cambio, los mensajes de respuestas 1xx's. son "respuestas provisionales", y no terminan la transacción SIP.

Reason-Phrase: Explicación textual muy breve del StatusCode, para ser interpretada por humanos.

3.3.4 Cabeceras

Las cabeceras SIP son similares a las cabeceras utilizadas en el protocolo HTTP (Hyper Text Transfer Protocol), tanto en la sintaxis como en la semántica. Especifican aspectos referentes a los participantes, trayectos, etc.

Generalmente, el orden en que aparecen las cabeceras no tiene mayor importancia, siempre que se cumpla que las cabeceras del tipo "salto a salto" (hop-by-hop) deben aparecer antes que cualquier cabecera del tipo "extremo a extremo" (end-to-end). Las primeras pueden ser modificadas o añadidas por los servidores proxy, en cambio las segundas deben ser transmitidas por éstos sin modificación alguna.

Los servidores proxy no deben reordenar las cabeceras, pero si pueden adicionar ciertos tipos de cabeceras, como por ejemplo la cabecera "Vio", así como otras del tipo "salto a salto" (hop-by-hop). Si bien pueden modificar ciertas cabeceras, no deben alterar los campos que son autentificados.

Determinadas cabeceras están presentes en todos los mensajes, otras no, solo en algunos. Igualmente, una aplicación que contenga el protocolo SIP no requiere necesariamente tener que comprender todas las cabeceras, aunque si es deseable. En el mismo sentido, si un participante SIP no entiende una cabecera, la ignora. Las cabeceras no especificadas deben ser ignoradas por los servidores.

El número total de cabeceras SIP definidas hasta ahora es 46, que rebasa el número de 37 cabeceras de la especificación inicial de SIP (RFC 2543), pudiendo ser agrupadas de la manera siguiente:

Cabeceras generales (general headers)

Se utilizan tanto en los mensajes de solicitud como en los mensajes de respuesta. Estas son: Call-ID, Contact, CSeq, Date, Encryption, From, Organization, Retry-After, Subject, Supported, Timestamp, To, User Agent y Vio.

Cabeceras de "entidad" (entity headers)

Dan información adicional referente al cuerpo del mensaje, o si éste no está presente, entonces a cerca del recurso identificado por la solicitud. Son las siguientes: Allow, Content-Encoding, ContentLength, Con tent-Type, ContentDisposition, Expires y MIME-Version.

Cabeceras de solicitud (request headers)

Posibilitan que el cliente pase información adicional al servidor referente a la solicitud, y también a cerca del propio cliente. Actúan como un modificador de solicitud. Son las siguientes: Accept, AcceptEncoding, Accept-Language, Accept-Contact, Authorization, Hide, In-Reply-To, Max-Forwards, Priority, Proxy-Authorization, Proxy-Require, RecordRoute, Reject-Contact, RequestDisposition, Require, ResponseKey, Route, Rack y Session-Expires.

Cabeceras de respuesta (response headers)

Posibilitan al servidor pasar información adicional en relación a la respuesta, información que no puede situar en el campo "Status-Une". Da información del servidor y también referente al recurso identificado por "Request-URI". Son éstas: Proxy-Authenticate, Server, Unsupported, Warning, WWW-Athenticatey Rseq.

A continuación se comentan las cinco cabeceras generales más significativas, que no deben faltar en un mensaje SIP:

Call-ID: Identifica, unívocamente, una llamada. Se utiliza para diferentes propósitos, por ejemplo:

- Para hacer corresponder respuestas con las correspondientes solicitudes.
- Para detectar duplicados de solicitudes INVITE.
- Para cambiar dinámicamente parámetros de una sesión.

Cseq: Identifica cada solicitud. Está compuesta por un número decimal y el nombre del método. En una sesión SIP Cseq se incremento por cada nueva solicitud, excepto en solicitudes ACK y CANCEL. La respuesta a una determinada solicitud lleva el mismo Cseq que dicha solicitud, esto es, el UAS copia el valor del Cseq de la solicitud en la correspondiente respuesta, por lo que la pareia relacionada solicitud-respuesta tiene el mismo valor de cabecera CSeq.

From: Identifica el origen de la solicitud en la pareja relacionada solicitud-respuesta, por tanto siempre se refiere al origen de la solicitud. Contiene la dirección del origen y, opcionalmente, el nombre de éste. Está presente en todas las solicitudes y respuestas

To: Identifica el destino de una solicitud. Debe estar presente en todos los mensajes de solicitudes y respuestas.

Via: Se emplea para registrar la ruta de una solicitud, en aras de posibilitar a los servidores SIP que intervienen reenviar las respuestas por la misma ruta seguida por la solicitud. En consecuencia, cada proxy adiciona una nueva cabecera "Vio". Esta cabecera es una de las potencialidades más importantes del protocolo SIP, mostrando que éste ha sido diseñado teniendo muy en cuenta la interconexión IP.

3.3.5 Cuerpo de mensajes

Los mensajes SIP, solicitudes y respuestas, opcionalmente pueden contener un cuerpo de mensaje, como ya sido indicado. Generalmente éste es una descripción de sesión con SDP, pero puede ser cualquier otro contenido, en forma "clara" o cifrado.

El contenido del cuerpo de mensaje solo es de interés para los UA s, no para los servidores de red, si de funcionalidades SIP se trata, pues éstos para encaminar los mensajes SIP solo necesitan conocer los contenidos de la línea de solicitud o de la línea de estado, según el caso, y de las cabeceras.

3.3.6 Modo de operación

Ya antes se planteó que los UAs pueden, por si solos y sin los servidores de red, ser capaces de soportar una comunicación básica, esto es, directamente entre endpoints. Pero también se establecido que la potencialidad de SIP se aprovecha con el empleo de los servidores de red.

Entonces, la señalización SIP puede desarrollarse por intermedio de servidores proxy, o servidores de redirección, a través de los cuales se llevan a efecto las transacciones SIP. De manera muy resumida, la operatividad SIP es la siguiente:

- Un UAC emite una solicitud, • un servidor proxy interviene en la localización de la parte llamada.
- Un UAS acepta (o rechaza) la llamada.

- Un UAC emite una solicitud.
- Un servidor de redirección notifica la dirección de la parte llamada.
- Un UAS acepta (o rechazo) la llamada.

Por lo que se requiere la localización de servidores y usuarios SIP, invitaciones SIP, servicios de registro SIP, transacciones SIP, direcciones SIP, etc.

Direcciones SIP: los "objetos" direccionados a través del protocolo SIP son usuarios en hosts, que se identifican mediante SIP - URL' s del tipo user@host, donde:

- user: nombre de usuario o número telefónico.
- host: nombre de dominio o dirección numérica de red.

Los SIP- URL' s son utilizados en los mensajes SIP para indicar, en relación con una solicitud: origen (from), destino actual (request-URI) y destino final (to). También para especificar direcciones de "redirección".

Por otra parte, los campos de cabeceras SIP pueden no contener SIP-URL' s, por ejemplo, en el caso que una llamada desde un teléfono convencional se trasvase por Internet con el empleo del protocolo SIP, entonces la cabecera "from" puede contener un URL telefónico.

Localización de un servidor SIP: cuando un cliente SIP desea enviar una solicitud, debe:

- Enviar ésta a un servidor proxy configurado localmente.
- Enviar dicha solicitud a la dirección IP y puerto que corresponda.

Por lo que es evidente que el software cliente debe disponer de una, o de un conjunto de direcciones SIP relativas a servidores SIP.

Transacciones SIP: una vez efectuada la localización de un servidor SIP, tienen lugar las transacciones, entendiendo como tal el conjunto de solicitudes (o solicitud) enviadas por el cliente al servidor, y las respuestas (o respuesta) retornadas por éste al cliente. Se trata de solicitudes y respuestas relacionadas, o lo que es lo mismo, con determinados parámetros idénticos (por ejemplo, call-ID, Cseq, to, from).

Si se emplea TCP como protocolo de transporte, la solicitud (o solicitudes) y la respuesta (o respuestas) de una transacción usan la misma conexión TCP. Varias solicitudes desde el mismo cliente y al mismo servidor pueden emplear la misma conexión TCP, o para cada transacción una conexión TCP diferente. Esto es, una conexión TCP puede soportar una o varias transacciones entre un cliente y un servidor.

Invitaciones SIP: una invitación SIP consta de dos solicitudes, por tanto corre a cargo de un cliente. Estas parejas de solicitudes son INVITE y ACK, o INVITE y BYE, según el caso.

La solicitud INVITE contiene, generalmente, la descripción de la sesión, descrita con formato SDP. Esta descripción informa a la parte llamada qué tipo de medios, el solicitante, puede aceptar, y dónde éste desea le sean enviados los datos SDP. Esta descripción informa a la parte llamada qué tipo de medios, el solicitante puede aceptar, y dónde éste desea le sean enviados los datos

Localización de usuarios SIP

Dado que el usuario llamado puede desplazarse entre diferentes terminales, con un Servidor de Localización estas localizaciones pueden registrarse dinámicamente. Para esto puede usarse uno de diferentes protocolos, por eiemplo, finger, LDAP, etc.

Servicio de registro SIP

Posibilita que el cliente, mediante una solicitud REGISTER, notifique a un servidor, proxy o de redirección, su ubicación. Ya se ha comentado antes que usualmente en la práctica un servidor de localización o registro acompaña a los servidores proxy y redirección. Nótese que, tanto uno como otro, debe consultar a un servidor de localización para poder desarrollar su tarea.

Un Servidor de Localización puede retornar varias localizaciones, bien porque el usuario esté "loggeado" en diferentes hosts al mismo tiempo, o porque el Servidor de Localización tenga temporalmente información no actualizada

Con esta información, y según el tipo de servidor SIP, proxy o de redirección, se procede así:

Servidor de redirección

Retorna dicha información de dirección al cliente solicitante, y éste procede a direccionar a la parte llamada.

Servidor Proxy

Secuencialmente o en paralelo, intenta éste direccionar a la parte llamada.

Servicio de registro SIP:

Posibilita que el cliente, mediante una solicitud REGISTER, notifique a un servidor, proxy o de redirección, su ubicación. Ya se ha comentado antes que usualmente en la práctica un servidor de localización o registro acompaña a los servidores proxy y redirección. Nótese que, tanto uno como otro, debe consultar a un servidor de localización para poder desarrollar su tarea.

Una implementación minima de SIP bebe cumplir, en relación con los elementos funcionales clientes y servidores, lo siguiente:

Clientes: Deben ser capaces de generar las solicitudes INVITE y ACK, así como las cabeceras Call-Id, Content-Length, Content-Type, Cseq, Require, From y To. También deben "entender" el protocolo SDP y ser capaces de reconocer las clases 1 hasta la 6 de los status codeo

Servidores: Deben "entender" las solicitudes INVITE, ACK, OPTIONS y BYE. De tratarse de servidores proxy, también la solicitud CANCEL. También deben ser capaces de generar de manera apropiada las cabeceras Call-Id, ContentLenght, Content-Type, CSeq, Expires, From, Max-Forwards, Require, To y Vio.

X. PRÁCTICAS DE LABORATORIO

PRÁCTICA 0: ADMINISTRACIÓN DE SERVICIOS EN LINUX

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

Objetivo:

- Presentar la temática a seguir durante las prácticas de laboratorio.
- Familiarizar al estudiante con los conceptos básicos de administración de servicios en Linux que son necesarios para el desarrollo de las prácticas.

Contenido

- Descripción general de las prácticas.
- Administración de usuarios y grupos en Linux.
- Instalación mediante apt-get en Debian.
- El sistemas de archivos de Linux (directorios y ficheros importantes).

Metodología de Evaluación

- Se explicará al alumno la metodología de evaluación empleada durante el desarrollo de las prácticas.

Tutorías

- Todos los estudiantes podrán hacer consultas al profesor fuera del laboratorio en un determinado horario, por lo tanto se debe brindar el horario para estas tutorias.

PRÁCTICA 1: ACCESO REMOTO A UNA MÁQUINA CON SSH

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción:

Con esta práctica, iniciamos nuestro proceso de conocer de una manera más profunda algunas de las aplicaciones y servicios más utilizados dentro de Internet. En esta primera práctica vamos a conocer como podremos acceder a una máquina desde otra, este es el punto de partida para la administración centralizada de una serie de servidores. La idea básica es poder administrar los diferentes servicios que vamos a montar en las siguientes prácticas desde una sola a través de SSH.

Las terminales virtuales son de las aplicaciones que desde siempre han sido muy utilizadas para el acceso remoto a máquinas. En esta práctica, utilizaremos una aplicación de red que nos permitirá obtener una shell segura gracias al uso de cifrado en las comunicaciones. Para esto emplearemos el protocolo ssh (Secure Shell), en la implementación Open SHH.

SSH nos ofrece una gran cantidad de alternativas a la hora de emplear cifrado y autenticación en las comunicaciones, así como para la creación de túneles. Sin embargo estas características son tema de estudio de otra asignatura por lo cual no las abordaremos.

2. Objetivo de la Práctica

Tiene como principal objetivo estudiar el uso básico del servicio de red SSH para la conexión a una máquina remota y la ejecución de aplicaciones en otra máquina que puedan ser vistas en la nuestra.

3. Instalación de Open SSH

El servidor SSH viene incluido ya en los discos de instalación de las distribuciones actuales de Linux. Si no se tiene el disco, puede descargarse de la

página de la http://www.openssh.org el fichero XXXXXX.tar.gz y seguir los pasos del archivo README para la Instalación.

Para instalarlo directamente desde Internet, lo podemos hacer como en la práctica anterior:

```
$ apt-get install openssh-server
```

4. Arrancar, Detener y Reiniciar Servidor SSH

SSH es un servicio de red que se arranca al iniciar la máquina, por lo cual se utiliza como los demás servicios del sistema:

1. Para arrancar el servidor:
   ```
   $ /etc/init.d/ssh start
   ```
2. Para detener el servidor:
   ```
   $ /etc/init.d/ssh stop
   ```
3. Para reiniciar el servidor:
   ```
   $ /etc/init.d/ssh restart
   ```

5. Iniciando sesión contra un Servidor SSH

Para iniciar sesión contra un servidor SSH se hace básicamente con la orden:

```
$ ssh maquina_remota
```

Con esta sintaxis lo que haríamos es intentar iniciar sesión ssh en la máquina remota empleando como login de acceso el nombre del usuario con el cual estamos trabajando en nuestro sistema si es que hay un usuario en la máquina remota con ese nombre. En caso de que no existiese un usuario con el mismo login que el que nosotros estamos empleando, el servidor nos preguntaría por un login y un password. Ejemplo:

```
denis$ ssh 192.168.0.3
```

Intentará acceder al usuario denis de la máquina 192.168.0.3. Si queremos especificar un usuario podemos hacerlo de dos maneras. Supongamos que desde nuestra máquina en la que estamos con la cuenta denis deseamos acceder a la

cuenta del usuario pepe de la máquina 192.168.0.3, esto lo podríamo hacer de dos formas:

```
denis$ ssh -l pepe 192.168.0.3
denis$ ssh pepe@192.168.0.3
```

Ambas órdenes producen iguales resultados. En la primera ocupamos el modificador –l para indicar el nombre del usuario con el que queremos iniciar sesión en la máquina remota y en la segunda, lo hacemos indicando el nombre del usuario antes de la arroba.

Existen diferentes parámetros que nos darán más opciones a la hora de iniciar sesión contra un servidor SSH. Para obtener una lista completa recomiendo revisar el manual de ssh.

6. Ejecutar aplicaciones en una máquina remota

Una vez conectados a la máquina remota, podemos realizar en ella cualquier cosa como si estuviésemos en nuestra máquina local. Sin embargo, a veces es interesante poder ejecutar una aplicación en otra máquina pero obtener la salida de la aplicación en nuestra máquina local. Para ello colocamos al final de la orden de conexión, entre comillas simples la aplicación que queremos que se ejecute en la máquina remota. Ejemplo:

```
$ ssh pepe@192.168.0.3 'ping 19.168.0.9'
```

Con esta orden ejecutamos desde la máquina 192.168.0.3 como el usuario pepe un ping a la máquina 192.168.0.9 y la respuesta del ping la vemos en nuestra máquina.

Podemos redireccionar también las aplicaciones en modo gráfico, para ello necesitamos añadir el parámetro -X a la orden. Ejemplo:

```
$ ssh pepe@192.168.0.3 'firefox'
```

Con esto, arrancamos el proceso firefox el cual se ejecuta en la máquina 192.168.0.3 pero se visualiza en nuestra máquina. Con ello lo que se gana es que no

estamos ocupando recursos de nuestra máquina sino de la máquina a la cual estamos accediendo. Esto resulta útil cuando la máquina que tenemos no es muy potente pero tenemos alguna otra máquina más potente en la cual podríamos ejecutar algunas aplicaciones.

Si deseamos que esto ocurre siempre para cualquier aplicación gráfica sin necesidad de poner siempre el parámetro -X, debemos abrir el fichero de configuración del cliente en `/etc/ssh/ssh_config`, y cambiar:

```
ForwardX11 NO por YES
```

7. Añadiendo un poco de seguridad

Debemos de recordar que la idea de aprender SSH es que podamos manejar de manera remota los diferentes servicios que vamos a instalar a lo largo de estas prácticas. Por esta razón, es importante que tengamos bastante seguridad en las máquinas a las cuales vamos a acceder de manera remota porque en esencia pueden ser servidores que deben de estar bastante seguros.

Por esto es importante que aunque podamos acceder de manera remota a las máquinas (servidores) también es importante restringir lo más posible el acceso a estas máquinas. Por esta razón, es importante tener en cuenta las siguientes consideraciones:

1. No permitir que nadie pueda establecer conexión a nuestro servidor SSH como usuario root. Para ello abrimos el fichero de configuración de `/etc/ssh/sshd_config` y cambiamos la linea:

```
PermitRootLogin YES por NO
```

2. Permitir la conexión solo a ciertos usuarios del sistema. Para ello en el fichero de configuración añadimos una línea que contenga a los usuarios autorizados:

```
AllowUsers denis pepe
```

Esto permite que sólo los usuarios denis y pepe puedan iniciar sesión en esta máquina.

3. Permitir que solo se pueda iniciar sesión desde una determinada IP. Para ello nos auxiliaremos del fichero del sistema hosts.allow situado en /etc/, y especifican los host que estan permitidos para una determinada aplicación. Para lograr nuestro objetivo añadimos en el fichero /etc/hosts.allow:

```
sshd: 192.168.0.2/255.255.255.0
```

Con esto, sólo permitimos iniciar sesión desde la máquina 192.168.0.2 con máscara de red 255.255.255.0. Si probamos conexión desde cualquier otra máquina, no nos lo permitirá.

4. Cambiar el puerto en el cual escucha el servidor ssh de manera que solo el usuario que conozca dicho puerto pueda iniciar una sesión contra nuestro servidor. Para ello debemos de cambiar en el fichero de configuración /etc/ssh/sshd_config la linea:

port 22 (que es el valor que tiene por defecto) a port 2000 por ejemplo.

Ahora para conectarnos al servidor ssh lo haremos indicando el puerto al que deseamos conectarnos. Esto lo hacemos con la opción -p:

```
$ssh -p 2000 192.168.0.3
```

8. Copia de archivos a una máquina remota

Usando el protocolo SSH podemos copiar archivos de nuestra máquina local a la remota o viceversa de una manera segura a través del comando scp. La sintaxis de este comando es:

```
$ scp archivo_origen archivo_destino
```

Por ejemplos, si queremos copiar el archivo prueba.txt de nuestra máquina local al directorio home del usuario pepe en la máquina 192.168.0.3 con el nombre remoto.txt la orden a emplear sería:

```
denis$ scp prueba.txt
pepe@192.168.0.3:/home/pepe/remoto.txt
```

Si por el contrario, lo que queremos es traer el archivo viceversa.txt que esta en el directorio tmp de la máquina 192.168.0.3 accediendo a esta como el usuario juan a nuestro directorio home (/home/denis) sería:

```
denis$ scp juan@192.168.0.3:/tmp/viceversa.txt   /home/pepe
```

En cualquiera de estos casos se nos pedirá la contraseña del usuario al cual estemos intentando acceder en la máquina remota, pero la sesión finalizará inmediatamente después de que termine la transferencia de el (los) fichero (s). También se pueda hacer la copia de directorios completos con el contenido de los subdirectorios dentro de este, para lo cual necesitaríamos indicar que la copia se realice de una manera recursiva. Esto lo indicamos con el parámetro -r.

Por ejemplo, para copiar todo el contenido del directorio /home del usuario juan de la máquina 192.168.0.3 con sus carpetas y subcarpetas a la carpeta /tmp de nuestra máquina local sería:

```
denis$ scp -r juan@192.168.0.3:/home/juan/*/tmp
```

9. Notas Finales

1. Prueba diferentes conexiones a diferentes máquinas del laboratorio y ejecuta aplicaciones en otras máquinas. Además de varias aspectos en la seguridad.
2. Siempre que realizamos algún cambio en el archivo de configuración del demonio SSH, debemos reiniciar el servidor.
3. Existen otros parámetros de configuración pero estos son los más usados. Su uso depende de la seguridad y los requerimientos de cada caso.
4. Para una lista más detallada de estos parámetros, revisar el manual de SSH.

10. Ejercicio Práctico

Piensa que eres un administrador de la Red en una empresa ANONIMA S.A. y tu jefe te llama para decirte que necesita poder acceder desde la computadora en su oficina a la computadora de su casa. Para lo cual le dices que lo mejor es usar una

conexión SSH hasta su computadora. Para poder darle seguridad a la conexión debes garantizar que las conexiones solo puedan hacerse de la IP de su máquina en la oficina, a través de un puerto que solo el conozca y debe poder acceder como root.

Para ello escoge una pareja en el grupo de laboratorio y configura tu servidor ssh como si fuese la máquina de la casa de tu jefe y tu compañero debe poder conectarse a tu máquina como si fuera la oficina de la máquina de tu jefe y viceversa.

11. Referencias

- http://www.vensign.com/2007/02/22/ssh-secure-shell/
- http://www.techtear.com/2007/04/08/trucos-y-consejos-para-asegurar-ssh-en-linux/

PRÁCTICA 2: CONFIGURACIÓN DE UN SERVIDOR NFS

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción

El protocolo NFS nos permitirá compartir sistemas de ficheros a través de una red de manera transparente. La idea básica de esto es poder disponer de directorios que se encuentran instalados en una máquina remota como si estuviesen en nuestra propia máquina.

El implementar este servidor es bastante sencillo pero la utilidad que se puede tener es bastante potente sobre todo en entornos en los que se requiere que sin importar la máquina de la red en la que un usuario inicie sesión, su información siempre este presente.

2. Objetivo de la Práctica

Tiene como principal objetivo el configurar un servidor NFS utilizando en nuestro caso el páquete nfs-utils-XYZ y de esta manera lograr compartir ficheros a través de nuestra red local de manera transparente.

3. Requerimientos de la Instalación

Para poder instalar el paquete nfs-utils se nescesita antes que todo tener instalado:

- **tcp_wrappers:** El paquete tcpwrappers proporciona programas envoltorios para demonios, que informan sobre el nombre del cliente que solicita servicios de red y el servicio solicitado. Ya viene instalado por defecto.

- **portmap:** Pormap se utiliza para reenviar peticiones RPC a demonios RPC como NFS.

Ambos paquetes, podemos encontrarlos en los discos de instalación de Linux (instalado por defecto en algunas distribuciones), o se pueden bajar los paquetes de la páqina de la asignatura.

4. Instalación desde los Fuentes de nfs-utils-XYZ

1. Obtener el fichero nfs-utils-XYZ que para nuestro caso esta disponible en el disco de instalación de Linux o en Internet y lo copiamos en nuestro directorio home. (Si lo hemos instalado desde el cd de Linux no se requieren los siguientes pasos).

2. Una vez que hemos obtenido el paquete, lo desempaquetamos con la orden:
   ```
   $ tar -xvzf  nfs-utils-XYZ.tar.gz
   ```
3. Cambiarse al directorio que se crea:
   ```
   $ cd nfs-utils-1.0.7
   ```
4. Configurar el paquete con las siguientres opciones:
   ```
   $ ./configure –prefix=/usr –sysconfdir=/etc –disable-nfsv4
     –disable-gss
   ```
5. Compilamos los ficheros del archivo:
   ```
   $ make
   ```
6. Instalamos el programa
   ```
   $ make install
   ```

5. Instalación desde Internet

De igual manera que en las practicas anteriores, en Debian, se puede instalar desde Internet haciendo:

```
$ apt-get install portmap
$ apt-get install nfs-common nfs-kernel-server
```

6. Arrancar, Detener y Reiniciar Servidor NFS

NFS es un servicio de red que para lanzarlo se necesitan de varios demonios, por la tanto no vamos a lanzar uno de ellos. Éstos son:

El demonio statd que se arranca con:
```
$ /etc/init.d/nfs-common { start | restart | stop }
```

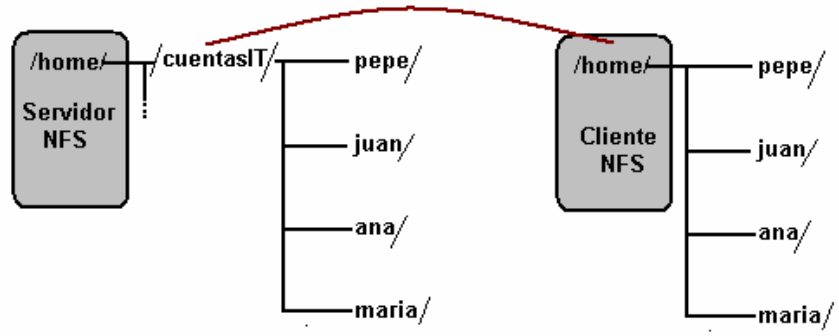
Los demonios nfsd y mountd que se arrancan con:
```
$ /etc/init.d/nfs-kernel-server { start | restart | stop }
```

El demonio portmap que se arranca con:
```
$ /etc/init.d/portmap { start | restart | stop }
```

7. Configuración del Servidor NFS

Antes que todo, presentaremos el escenario que se quiere instalar. La idea es colocar un servidor NFS en el laboratorio de Aplicaciones Telemáticas la cual será una máquina Linux que en su directorio HOME contendrá un directorio **cuentasIT** que contendrá una carpeta por cada usuario que hay en el Laboratorio. Con esto lo que vamos a hacer es que cada máquina en vez de usar el directorio home que tiene en su disco, monte encima el directorio **cuentasIT** del servidor con lo cual un usuario podrá acceder a sus datos desde cualquier máquina del laboratorio ya que cada vez que guarde no guardará en la máquina local en la que trabaje sino en el Servidor NFS.

Además la forma de trabajo del usuario seguirá siendo la misma ya que el siempre tendrá su directorio home en la misma ubicación, desde cualquier máquina. El directorio home del servidor y que a su vez será replicado por todas las máquinas del laboratorio será:



Para configurar el servidor NFS se necesita añadir en el fichero /etc/exports una línea por cada directorio que deseemos compartir. La estructura de dicha línea es:

```
directorio equipo1(opcion11,opcion12) equipo2(opcion21,opcion22)
```

donde:

directorio: Es el directorio a compartir, que en nuestro caso es **/home/cuentasIT**

equipo#: Clientes que tendrán acceso al directorio compartido. Estos equipos se pueden indicar por su dirección IP o su nombre DNS.

opcionXX: Son las opciones que nos permiten tener acceso a esos directorios con determinados privilegios:

1. ro | rw: Con ro se obtiene acceso de solo lectura, esta es la opcion por defecto. Mientras que con rw tenemos acceso tanto de lectura como de escritura.

- sync | async: sync es la opción recomendada, ya que se ha de respetar el protocolo NFS, es decir, no se responden a las peticiones antes de que los cambios realizados sean escritos al disco. Con la opción async se permite mejorar el rendimiento y agilizar el funcionamiento global, pero supone un riesgo de corrupción de archivos o del sistemas de ficheros en casos de caidas del servidor y/o errores de éste.

- root_squash | no_root_squash | all_squash : root_squash indica que un cliente identificado como root tendrá acceso al directorio con privilegios de un usuario anónimo. Si seleccionamos la opción no_root_squash evitaremos esto, y si indicamos all_squash, entonces aplicaremos esto último a todos los usuarios, no sólo root.

En nuestro caso para exportar cuentasIT a todas las máquinas del laboratorio , suponiendo que todas están en la red 192.168.0.0 deberemos incluir en /etc/exports

```
/home/cuentasIT/   192.168.0.0(rw,sync,no_root_squash)
```

Con este fichero indicaremos que queremos compartir los directorios /home/cuentasIT a los hosts de la red 192.168.0.0; tanto para lectura como para escritura. Se respetará el protocolo NFS, ya que no se responderán a las peticiones que se hagan antes de que los cambios se hayan escrito en disco. Si un usuario root(en el cliente) accede al directorio exportado, sus privilegios son los mismos del root.

8. Restringiendo el acceso a nuestro Servidor

Al igual que en la práctica anterior, vamos a restringir el acceso a los directorios especificados en exports a través de dos ficheros de acceso a la máquina como son: /etc/host.deny y /etc/host.allow.

El archivo /etc/hosts.deny

En este fichero pondremos todas las restricciones posibles para hacer mas seguro el sistema. Para ello denegaremos el acceso a portmap, ya que si se deniega portmap, aunque permitas nfs, no se podrá compartir porque éste depende de portmap. Por lo que solo se tendrá acceso a portmap por aquellos equipos que estén definidos en el fichero /etc/hosts.allow. El fichero /etc/hosts.deny quedará:

```
portmap:ALL
```

El archivo /etc/hosts.allow

En este fichero debe indicar a quienes permitimos el acceso al servicio de nfs y portmap. Se pueden indicar hosts individuales o una red. En este caso indicaremo nuestra red local que tiene máscara 24:

```
portmap:192.168.0.0/255.255.255.0
nfs:192.168.0.0/255.255.255.0
```

Con esto solo permitiremos que puedan montar el sistema de archivo las máquinas pertenecientes a la red 192.168.0.0 con máscara de subred 255.255.255.0. Que serán las de nuestra red. Con esto, y reiniciando el servidor, estamos listos para compartir los directorios /home/cuentasIT nuestro servidor cuya IP es la 192.168.0.1.

9. Configuración del Cliente NFS

Para poder montar el cliente en NFS, necesitamos que instalar el paquete nfs-common en nuestra máquina y el portmap (ambos incluidos en el disco de instalación de Linux) y arrancar ambos servicios a como se arrancaron en el servidor.

```
$ /etc/init.d/nfs-common start
$ /etc/init.d/portmap start
```

Montando el sistema Manualmente

Una vez hecho esto, para montar el directorio compartido utilizamos la orden mount. Cuya estructura genérica es:

```
$ mount -t nfs -o opcion[:usuario] dir_remoto dir_local
```

Las opciones son ro, rw, root_squash, no_root_squash, entre otras.

Si ahora desde cualquier máquina del laboratorio que pertenece a la red 192.168.0.0/24 queremos montar sobre su directorio HOME el contenido del directorio /home/cuentasIT que esta en el servidor NFS (ip 192.168.0.1) la orden sería:

```
$ mount -t nfs 192.168.0.1:/home/cuentasIT /home/
```

Para desmontar la carpeta lo hacemos con la orden umount y el nombre de la carpeta.

```
$ umount /home/
```

Montando el sistema desde el Arranque

Si queremos que desde que se encienda la máquina se monte la carpeta a como lo hicimos con la orden mount, debemos añadir la línea siguiente al fichero /etc/fstab:

```
192.168.0.1:/home/cuentasIT     /home/     nfs   defaults   0   0
```

Si entrar en detalle, el archive fstab indica el origen, el punto de montaje, el tipo y algunas opciones de los diferentes sistemas de archivos que se montan cuando la máquina arranca. Para más información consultar el manual de fstab.Con esto al arrancar nuestra máquina se montará la carpeta compartida cada máquina del laboratorio.

10. Notas Finales

1. Siempre que realizamos algún cambio en la configuración, debemos de detener el demonio y volverlo a ejecutar.
2. Existen otros parámetros de configuración pero estos son los más usados. Su uso depende de la seguridad y los requerimientos de cada caso.
3. Para mayores detalles sobre las ordenes mount, umount y sobre los ficheros hosts.allow, host.deny y fstab, revisar el manual de Linux.

11. Ejercicio Práctico

Continuando con la práctica anterior, tu jefe te vuelve a llamar para decirte que no le gusta tener que acceder a través de una SHELL a su máquina y que el quiere tener en su computadora de la oficina, una carpeta en su directorio home llamada Mi_casa (/home/oficinaIMi_casa) con los documentos que el tiene en el directorio /home/jefe de la computadora de su casa.

Para ello tu le dices que lo que se puede hacer es montar en su máquina un sistema de ficheros en red NFS, y exporta el directorio /home/jefe de su computadora de casa y montarla en la carpeta /home/oficina/Mi_casa. Tu jefe acepta la propuesta pero te dice que necesita que lo hagas ya mismo.

Tu misión es conectarte mediante SSH a la computadora de la casa de tu jefe instalar y configurar el servidor NFS, exportar el fichero y lograr que en la carpeta /home/oficina/Mi_casa este el mismo contenido que en /home/jefe..

Para la realización de esta práctica, debes continuar trabajando con tu compañero de la práctica anterior. El directorio /home/jefe será el directorio home de tu compañero y el /home/oficina, será tu directorio home. Deberás primero, crear el directorio Mi_home y luego deberás conectarte mediante SSH a la máquina de tu compañero y montar el servidor que se te pide.

12. Referencias

- http://revartm.wordpress.com/2007/03/06/servidor-nfs-bajo-ubuntu/
- http://www.escomposlinux.org/lfs-es/blfs-es-6.0/basicnet/nfs-util

PRÁCTICA 3: CONFIGURACIÓN DE UN SERVIDOR FTP

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción

El protocolo FTP fue uno de los primeros protocolos que aparecieron en Internet, a lo largo de los años han venido apareciendo diferentes implementaciones con el fin de hacerlo cada vez más seguro. Uno de esas implementaciones (vsftp) es precisamente la que utilizaremos para el desarrollo de esta práctica.

Debes tener en cuenta que muchas veces, la mejor forma de realizar actualizaciones o subir paquetes a un sitio web es precisamente a través de FTP. No sólo eso, también puede ser usado para guardar en el, información de respaldo o programas, videos u otros tipos de archivos que por ser bastante pesados no es muy recomendable que se descarguen a través de HTTP o que se transmitan a través del correo.

2. Objetivo de la Practica

Esta práctica tiene como principal objetivo el configurar el servidor vsftp que es un servidor FTP seguro, para que el alumno pueda afianzar a través de la práctica, los conocimientos que sobre este protocolo se adquirieron en la teoría.

Se indicarán, distintas políticas que pueden tomarse con el objetivo de hacer más seguro nuestro servidor y al mismo tiempo garantizar la seguridad de nuestro sistema.

3. Instalación desde los Fuentes de vsftpd

Para poder llevar a cabo la instalación de nuestro servidor, lo haremos de la manera tradicional que es:

1. Obtener el fichero vsftpd-XYZ.tar.gz que para nuestro caso esta disponible en la página Web de vsftpd y lo copiamos en nuestro directorio home.

2. Una vez que hemos obtenido el paquete, lo desempaquetemos con la orden:

```
$ tar -xvzf  vsftpd-XYZ.tar.gz
```

3. Cambiarse al directorio que se crea:

```
$ cd  vsftpd-XYZ
```

4. Crear el archivo binario:

```
$ make
```

5. Comprobar que se ha creado el archivo. Para ello ejecutaos el comando ls y debe de generanos el siguiente resultado:

```
$ ls -l vsftpd
-rwxrwxr-x 1     denis denis 61748 Sep 27 00:26 vsftpd
```

Ajustes para un correcto funcionamiento

Para continuar con el proceso de configuración, necesitamos verificar algunos elementos para el correcto funcionamiento:

1. Tener en nuestra máquina al usuario “nobody”. Para ello debemos visualizar (con la orden cat) el archivo /etc/passwd y buscar dentro del archivo (con la orden grep) el usuario nobody:

```
$ cat /etc/passwd | grep nobody
```

Si no existe (es decir, si la orden anterior no produce ningún resultado) lo añadimos con la orden useradd:

```
$ useradd nobody
```

2. A continuación, necesitamos verificar si existe un directorio vacio que se encuentra en /usr/share/empty:

```
$ ls -l /etc/share/empty
```

Si no existe, lo creamos con la orden mkdir:

```
$ mkdir /etc/share/empty
```

3. Como todo servidor FTP, se debe de tener un usuario anónimo (anonymous), para ello debemos de crear el usuario ftp el cual debe de tener un directorio home valido en /var/ftp:

```
# Creamos el directorio para el usuario anonimo:
$ mkdir /home/ftp
```

Creamos el usuario ftp y con directorio home /home/ftp:

```
$ useradd -d /home/ftp ftp
```

Cambiamos el propietario y el grupo del directorio `/var/ftp` para que solo el usuario root pueda hacer modificaciones y el usuario anónimo sólo pueda leer de este:

```
$ chown root.root /var/ftp
$ chmod 755 /var/ftp
```

4. La instalación requiere que nosotros mismos copiamos en su lugar cada uno de los fichero que incluyen el paquete:

   `$ cp vsftpd /usr/local/sbin/vsftpd` (Fichero ejecutable)

   `$ cp vsftpd.conf /etc/` (Fichero de configuración)

   `$ cp vsftpd.conf.5 /usr/local/man/man5` (Ayuda)

   `$ cp vsftpd.5 /usr/local/man/man8` (Ayuda)

5. Para arrancar el servidor aparte de los demás servicios de red (modo standalone) debemos abrir el fichero de configuración:

   ```
   $ vi /etc/vsftpd.conf
   ```

   Y añadimos al final del archivo la siguiente línea al final del fichero:

listen=YES

4. Instalar desde Internet en Debian

Una forma más sencilla de instalar en debian es ejecutar como root:

```
$ apt-get install vsftpd
```

Para el resto de la práctica, se ha de suponer que se instaló desde Internet ya que esto genera menos errores. Esta instalación nos ha de dejar una configuración básica con la cual podemos probar ya nuestro servidor. Debemos de recordar que

durante la instalación se nos habrá creado el usuario ftp cuyo directorio home será /home/ftp.

5. Arrancar, Probar y Detener el Servidor FTP

Antes de arrancar el servidor debemos de copiar algún fichero (de cualquier tipo) dentro del directorio que hemos asignado al usuario “nobody” (/var/ftp), para que al conectarnos intentemos descargarlo.

1. Para arrancar el servidor lo haremos lanzándolo en segundo plano utilizamos la orden:

   ```
   $ /etc/init.d/vsftpd start
   ```

2. Para probar que nuestro servidor esta funcionando de forma correcta abrimos otra terminal y nos conectamos al servidor FTP que tenemos corriendo en nuestra máquina:

   ```
   $ ftp localhost (ó ftp nuestra_ip)
   ```

   Una vez dentro nos pedirá un login al cual debemos de contestar con **ftp** o con **anonymous** dejando la contraseña vacia. Ya dentro podremos hacer un **ls** y deberá de visualizarse el contenido que colocamos en /home/ftp. Para probar que funciona podemos hacer un:

   ```
   get nombre_archivo_remoto nombre_destino
   ```

   Con lo cual nos traeríamos el archivo remoto a nuestra máquina local. Si lo que deseamos es poner un archivo en el ftp lo que debemos usar el método put:

   ```
   put nombre_local nombre_en_el_destino
   ```

   Con lo cual nos traeríamos el archivo local esta en nuestra máquina y debemos de indicar con que nombre queremos que se cree en la máquina remota. En este caso no se podría porque el usuario anónimo sólo pueda descargar archivos. Para ver que comandos podemos emplear dentro del ftp podemo teclear: ?. Para salirnos del servidor ftp empleamos la orden exit.

3. Finalmente, si queremos detener nuestro servidor lo hacemos con la orden:

   ```
   $ /etc/init.d/vsftpd stop
   ```

6. Configuración del Servidor FTP

A continuación se comentarán algunos parámetros de configuración que encontramos dentro del archivo vsftpd.conf que son muy útiles a la hora de configurar el servidor de una manera segura. Muchas de estas opciones ya vienen en el fichero y solo hay que descomentarlas, sin embargo, otras hay que incluirlas en el fichero por nuestra propia cuenta.

```
# Profesor: Indicamos que en nuestro servidor permitimos usuarios
# anónimos, si solo queremos usuarios autenticados la ponemos a NO.
anonymous_enable=YES
# Profesor: Indicamos que los usuarios registrados en Linux puedan
# iniciar sesion en nuestra máquina. En este caso, los usuarios al
# conectarse acceden a su directorio HOME.
local_enable=YES

# Profesor: Se utiliza para enjaular a los usuarios dentro de su
# directorio home para que no puedan navegar por todo el sistema.
chroot_local_user=YES

# Profesor: Permite que se suban ficheros al servidor.
write_enable=YES

# Profesor: Permisos con los que se crean los archivos subidos.
local_umask=022

# Numero de clientes simultaneos
max_clients=10
```

7. Añadiendo seguridad al Servidor FTP

Enjaular usuarios de forma selectiva:

Con la opción `chroot_local_user=YES`, enjaulamos a todos los usuarios dentro de su directorio home, sin embargo no siempre queremos que esto suceda. En ocaciones deseamos enjaular a ciertos usuarios pero que otros usuarios puedan seguir accediendo a todo el sistema, un caso concreto puede ser el del usuarios root que deseamos que tenga acceso mediante FTP a todo el sistema (aunque lo mejor es hacerlo con SSH) para lograr eso, añadimos al fichero de configuración las opciones:

```
# Profesor: Habilitar la listas de usuarios enjaulados
chroot_list_enable=YES
# Profesor: Indicar la ruta donde se encuentra la lista de
usuarios
# enjaulados.
Chroot_list_file=/etc/vsftpd.chroot_list
```

El archivo vsftpd.chroot_list debe ser creado en /etc/, y debemos de añadir a este archivo en una linea diferente cada uno de los usuarios que deseamos que no sean enjaulados, ejemplo: Si tuviesemos el sistema los usuarios (pepe, juan, maria, ana). Si creamo el archivo

```
$ vi /etc/vsftpd.chroot_list
```

y añadimos dentro de este:

```
Pepe
Juan
```

Si ahora arrancamos nuevamente el servidor, si entramos con la cuenta de pepe o la de juan, podremos navegar por todo el sistema de archivos del servidor, mientras que con la de maria y la de ana, solo podremos estar dentro de nuestro directorio home.

Nota: Si se comenta la linea `chroot_local_user=YES` y se deja intacta la configuración actual, entonces el archivo funciona al revés, ahora este contiene los usuarios que quedan enjaulados en su directorio home y los que aquí no aparecen, están libres. Esta forma de trabajo es muy insegura ya es que mejor enjaular a todos e indicar expresamente a quienes se deja libre.

Permitir acceso a usuarios del sistema de forma selectiva:

Esta es otra forma de proporcionar seguridad. Ahora podemos restringir todavía más el acceso a nuestro servidor FTP, seleccionando que usuarios del sistema deseamos que puedan iniciar sesión en nuestro servidor con la cuenta de usuario que tengan en la máquina. Para ello activamos la opción `userlist_enable`.

```
userlist_enable=YES
userlist_deny=YES/NO
```

Esta opción nos permite usar un archivo para los usuarios del sistema a los cuales se les permitirá o denegará el acceso. Igual que con chroot_list_enable, se

debe de crear un fichero en /etc/ llamado vsftpd.user_list que contenga a los usuarios a ser afectados. Ejemplo:

```
$ vi /etc/vsftpd.chroot_list
```

Podemos añadir a los mismos del caso anterior:

```
Pepe
Juan
```

Ya tenemos creado el archivo. Aquí entra en juego la segunda opción que es userlist_deny. Esta opción (al igual que las demás) puede tomar el valor YES ó NO. Si se pone NO, le denegamos el acceso a los usuarios que no estén en el archivo vsftpd.user_list. En nuestro caso si intentamos usar como login para entrar al servidor ftp maria o ana, el servidor nos denegará el acceso.

Si por el contrario, ponemos YES, denegaremos el acceso a los usuarios que estén en el archivo vsftpd.user_list. Si ahora intentamos entrar con el usuario pepe o juan no podremos, pero si lo lograremos con los usuarios maria o ana.

Debe tenerse cuidado con estas opciones porque pueden ocasionar un agujero de seguridad si no se emplean de manera adecuada.

8. Creación de usuarios Virtuales

Cuando la cantidad de usuarios a los cuales deseamos darles servicios es muy grande, es poco recomendable tener una cuenta, por ello lo que haremos es crear usuarios virtuales que utilizaran al usuario ftp como usuario real del sistema para entrar a sus cuentas. Usaremos db3 (Berkereley db) para almacenar los nombres de usuarios y contraseñas, y PAM (Pluggable Authentication Modules) como mecanismo de autenticación.

Debemos instalar los paquetes libdb3, libdb3-util, para lo cual hacemos:

```
$ apt-get install libdb3-util
```

Como root creamos una carpeta por ejemplo en /etc/ llamada vsftpd-config en la cual guardaremos todas las modificaciones que vamos a crear:

```
$ mkdir /etc/vsftpd-config
```

Creamos otro directorio llamado usuarios en /etc/vsftpd-config donde guardaremos la configuración de los usuarios y nuestra base de datos con los nombres y contraseñas:

```
$ mkdir /etc/vsftpd-config/usuarios
```

Ahora tendremos que crear un archivo de texto con cualquier editor de texto (nano, vi, etc) donde su contenido serán los nombres de los usuarios y sus respectivas contraseñas.

```
$ vi /etc/vsftpd-config/usuarios/users
```

Y el contenido del archivo será este:

Andres andres123 sofia sofia123 (Se terminara el contenido con una linea vacía)

Ahora lo vamos a convertir con db3 a una base de datos para después poder usarlos con PAM y luego le cambiaremos los permisos a la base de datos creada:

```
$db3_load -T -t hash -f /etc/vsftpd-config/ususarios/users
                        /etc/vsftpd_login.db
$ chmod 600 /etc/vsftpd_login.db
```

Ya tenemos nuestra base de datos, a continuación cambiaremos el contenido del archivo PAM ftp (vsftpd en Debian) ubicado en el directorio /etc/pam.d/ pero antes realizaremos una copia de seguridad del mismo.

```
$ cp /etc/pam.d/vsftpd /etc/pam.d/vsftpd.old
```

Modificamos el archivo con el comando cat

```
$ cat >/etc/pam.d/vsftpd<<EOF
>auth required /lib/security/pam_userdb.so db=/etc/vsftpd_login
>account required /lib/security/pam_userdb.so \
db=/etc/vsftpd_login
>
>EOF
```

Una vez modificado el archivo PAM modificaremos el archivo de configuración vsftpd.conf en el directorio /etc/. A continuación se muestran las líneas que deben modificarse o agregarse:

```
# Especifica el archivo PAM modificado anteriormente.
pam_service_name=vsftpd
# Esto permite el acceso a usuarios virtuales.
# "guest" se refiere todo el acceso no-anónimo.
chroot_local_user=YESguest_enable=YES
# Mapea el acceso no-anónimo al usuario ftp del sistema.
guest_username=ftp
# Especifica la ruta de los archivos de configuración de cada
usuario.
user_config_dir=/etc/vsftpd-config/usuarios
```

En este ejemplo se permite el acceso a usuarios anónimos donde su directorio raíz (por defecto) es /home/ftp/ especificado como "home" del usuario ftp del sistema. Dentro de este directorio vamos a crear los directorios raíz de los demás usuarios, entonces deberemos modificar el directorio "home" del usuario ftp del sistema, ya que como vimos anteriormente al acceder como usuario anónimo se podría ver el contenido de los demás usuarios.

Ahora crearemos en el directorio /home/ftp/ los directorios raíz de cada usuario. En nuestro ejemplo para **andres** y **sofia**. También le cambiaremos el propietario y grupo, para que tenga los permisos del usuario ftp:

```
$ mkdir /home/ftp/andres /home/ftp/sofia
$ chown ftp:nogroup /home/ftp/andres /home/ftp/sofia
```

Luego habrá que crear dos archivos en el directorio /etc/vsftpd-config/usuarios con las configuraciones de los usuarios. El nombre de los archivos de configuración de permisos y otros aspectos deberá llevar el mismo nombre que el usuario en cuestión. Para el usuario andres crearemos un archivo llamado andres con el siguiente contenido:

```
dirlist_enable=YES
download_enable=YES
local_root=/home/ftp/andres
anon_upload_enable=YES
anon_world_readable_only=NO
write_enable=YES
anon_mkdir_write_enable=YES
```

Esto hace que tenga permisos de escritura, pueda descargar archivos, subir archivos crear directorios. La línea local_root=/home/ftp/andres hace que el usuario solo pueda moverse dentro de ese directorio y sus sub-carpetas. Para el usuario sofia crearemos un archivo llamado sofia con el siguiente contenido:

```
dirlist_enable=YES
download_enable=YES
local_root=/home/ftp/frikiuser02
```

Este usuario solo puede listar el contenido y descargar siempre dentro del directorio /home/ftp/sofia y sub-directorios.

Ahora solo nos falta reiniciar nuestro servidor o recargar la configuración para que actualice todos los cambios.

```
$ /etc/init.d/vsftpd restart
```

Un problema con el usuario Anónimo:

Como sabemos, el usuario anonymous al conectarse, entra a la carpeta /home/ftp con lo cual puede entrar en todos los subdirectorios que están allí entre ellos, los de los usuarios virtuales. Para evitar esto tenemos dos opciones:

- **No permitir usuarios anónimos:**

Para desactivar el acceso de usuarios anonimos solo debe ser especificado en el archivo de configuración vsftpd.conf:

```
anonymous_enable=NO
```

- **Enjaular al usuario anónimos en otro directorio**

Para solucionar esto agregamos una linea al archivo vsftpd.conf:

```
anon_root=/home/ftp/anonimos
```

Y luego creamos el directorio:

```
$ mkdir /home/ftp/anonimos
```

Ahora ya enjaulamos al usuario anonymous en otro directorio y no podrá acceder a los directorios de los usuarios virtuales.

9. Indicaciones Finales

1. Siempre que realizamos algún cambio en la configuración, debemos de detener el servidor y volverlo a ejecutar.
2. Intentar acceder desde Window con una interfaz gráfica a nuestro servidor FTP. Puede usarse FileZilla_2_2_32_setup.exe, que es un Cliente FTP con interfaz gráfica.
3. Existen otros parámetros de configuración pero estos son los más usados. Para mayores detalles, revisar el manual de vsftpd.

10. Ejercicio Práctico

El departamento de computación de la UNAN-León ha decidido ofrecer a sus alumnos y profesores un servicio de FTP. Para no sobrecargar el servidor, se ha decidido que se crearán usuarios virtuales. Se desea que se configure un servidor vsftp que tenga dentro de /home/ftp/ los directorios **profesores** (que contendrá dentro de él los directorios de cada uno de los profesores) y **alumnos** que contendrá dentro de él un directorio por cada año de la carrera de Ingeniería en Telemática.

Además, todos los profesores tendrán permisos de crear directorios, subir y bajar archivos, listar y escribir en sus directorios. Los alumnos solo podrán descargar ficheros ya que la idea es que allí se pongan toda la información que se necesite descargar.

Nota: Suponga nombres aleatorios para los profesores (3 mínimo) y las carpetas de los alumnos se llamaran (SegundoIT, TerceroIT, etc).

11. Referencias

- http://www.linuxparatodos.net/portal/staticpages/index.php?page=09-como-vsftpd
- http://www.frikilinux.com.ar/wiki/Tutoriales/Vsftpd
- http://www.esdebian.org/staticpages/index.php?page=20050424180613347

PRÁCTICA 4: CONFIGURACION DE UN SERVICIO DE DIRECTORIOS LDAP

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción:

Con esta práctica iniciamos la consolidación de los aspectos teóricos del segundo bloque de la asignatura que son los “Servicios de Directorios”. Para ello emplearemos dos prácticas que serán la **Configuración de un Servicios de Directorios con LDAP** y la **Configuración de un Servidor DNS.**

2. Objetivo:

El objetivo de ésta práctica es, configurar un servicio de directorio basado en LDAP estableciendo diferentes políticas de acceso en dependencia de la posición del usuario dentro de la jerarquía del árbol.

3. Qué son los Directorios:

Cuando hablamos de directorios debemos olvidarnos de los directorios (o carpetas) que conocemos de toda la vida y empezar a pensar en el directorio como el de la guía telefónica, solo que con más información sobre cada miembro o empresa que solo el número de teléfono. Un directorio es una base de datos algo particular ya que a diferencia de las base de datos relacionales:

- Están diseñados para agilizar las búsquedas y las lecturas.
- Su contenido suele modificarse poco o nada.
- Tienen escaso soporte para transacciones complejas.

Otra de sus principales diferencia con respecto a una base de datos relacional es el hecho de que el modelo se organiza de manera jerárquica, lo que permite que la información se encuentre distribuida. Además de lo anterior, también tienen una

característica más que los diferencia de las bases de datos y es que estos directorios permiten inconsistencias.

A medida que se avance en la lectura de este documento se irán aclarando un poco más estos conceptos, de momento basta con saber que un servicio de directorio permite guardar los datos de una organización de forma jerárquica y luego hacer consultas sobre su contenido.

4. ¿Por qué usar servicios de directorios?

Estamos bastante acostumbrados a que siempre que escuchamos hablar de guardar información se nos viene a la mente una base de datos, sin embargo, no siempre necesitamos realizar transacciones. En muchas ocasiones la información tiene carácter estático y lo que se requiere más bien es realizar sobre ella una gran cantidad de consultas. Además, un modelo jerárquico se adapta más al mundo real y permiten además tener un punto de administración centralizada.

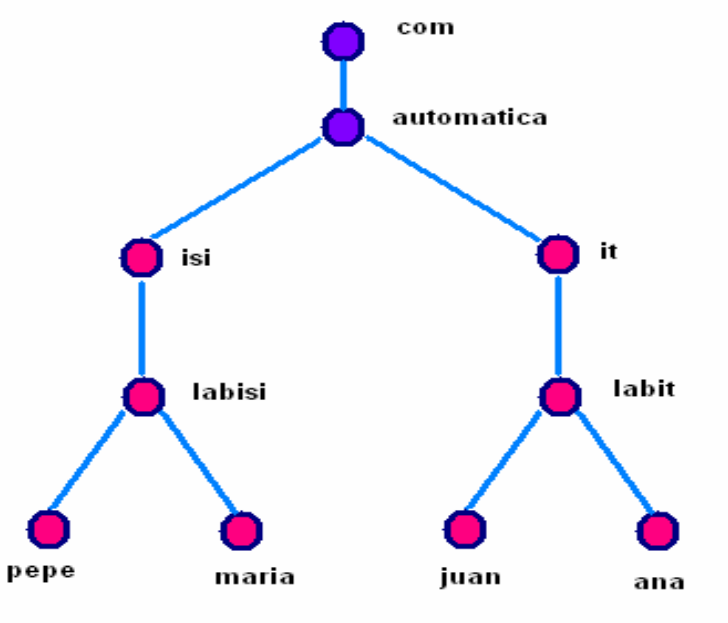
5. ¿Qué es LDAP?

LDAP (Lightweight Directory Acces Protocol), es un protocolo de acceso a directorio que se encuentra basado en el estándar X.500 de OSI y que hoy en día está muy difundido por su fácil implementación y su ligereza a diferencia de X.500. LDAP se encuentra basado en entradas u objetos. Cada objeto tiene asignado un DN (Nombre Distintivo) que depende de la posición que el objeto tenga dentro del árbol. Los objetos tienen atributos que se encuentran definidos según el o los esquemas a los que el objeto pertenezca.

En LDAP existen dos formas de nombrar a un objeto que son, mediante su DN (Nombre distintivo) por ejemplo uid=denis,dc=labit,dc=itdc=automatica,dc=com o su RDN (Relativo Nombre Distintivo), por ejemplo uid=denis.

6. Un ejemplo de un Directorio en LDAP

Para aclarar los conceptos acerca de LDAP y su uso, desarrollaremos un ejemplo de un árbol de directorios. A continuación se muestra el esquema que se desea desarrollar:



7. Planteamiento del problema

El departamento de automática de la unan león que cuenta de dos áreas de enseñanza que son Ingeniería en Sistemas de Información (ISI) e Ingeniería en Telemática (IT) desea poder almacenar la información de los alumnos que pertenecen a cada uno de los laboratorios de manera que puedan guardar para cada uno de ellos datos sobre sus cuentas en Linux (uid, gid, directorio home, etc).

La propuesta es crear un directorio LDAP para almacenar dicha información ya que se desea dejar abierta la posibilidad de que más adelante pueda añadirse también información de los profesores y directores. Además la información de los alumnos, no va a variar al menos durante todo el semestre.

8. Seleccionando e Instalándolo OpenLDAP

Para instalar nuestro directorio emplearemos OpenLDAP que es una implementación de LDAP libre. Podemos adquirir la última versión de este software descargándola desde la página web de OpenLDAP http://www.openldap.org/software/download/.

- Descomprimiremos el fichero con las fuentes en un directorio con la orden:

```
$ tar xvfz openldap-XXXXXX.tgz
```

- El configure lo haremos con las siguientes opciones:

```
$./configure --enable-debug --enable-cache --enable-
referrals --with-threads --enable-slapd
```

Puedes ver la documentación para otros tipos de opciones, pero con estas son suficientes para lo que tenemos pensado montar. Luego pasamos a compilar el software, testearlo e instalarlo en el sistema.

```
$ make depend
$ make
$ cd tests
$ make tests
$ cd ..
$ su
# make install
```

Si todo ha ido bien y no ha fallado ni la compilación ni los testeos, tendremos instalados el binario en /usr/local/libexec/slapd y los ficheros de configuración en /usr/local/etc/openldap/.

9. Editando el archivo de Configuración sladpd.conf

Ya tenemos instalado el software en nuestro equipo ahora debemos editar el fichero de configuración del demonio slapd que se encuentra en /usr/local/etc/openldap que se llama slapd.conf para adecuarlo al sistema de directorio que deseamos crear.

A continuación se presenta el archivo de configuración que usaríamos para iniciar a trabajar y se explica cada una de las líneas necesarias para nuestro caso. Este archivo puede que este presente luego de la instalación, sino es así podemos crearlo siguiendo algún ejemplo.

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
# Profesor: Iniciamos añadiendo los esquemas a utilizar, como los
# archivo de cabecera en C. Ya que los esquemas definen los diferentes
```

```
# tipos de objetos que se pueden declarar y sus atributos.
# Hay que revisar que estén en la ruta que se indica. Sino están allí,
# se debe de cambiar a su localización real. Estos que hemos incluido
# son los mas usuales pero no son los únicos.

include         /usr/local/etc/openldap/schema/core.schema
include         /usr/local/etc/openldap/schema/cosine.schema
include         /usr/local/etc/openldap/schema/inetorgperson.schema
include         /usr/local/etc/openldap/schema/nis.schema

schemacheck     on

# Profesor: Indica el lugar en el que se almacenaran los archivos que
# guardan el PID y la lista de argumentos que se le pasan al programa
# en mi caso los he modificado para guardarlos en una carpeta que cree
# en mi directorio personal

pidfile    /home/denis/ldap-file/slapd.pid
argsfile   /home/denis/ldap-file/slapd.args

########################################################################
# BDB database definitions
########################################################################
# Profesor: Definimos el tipo de base de datos a utilizar
database   bdb

# Profesor: Indicamos el directorio en el cual se creara la base de
# datos (El directorio debe existir).
directory  "/home/alcala/ldap-file/data"

# Profesor: Guarda la última modificación
lastmod on
index objectClass     eq

# Profesor: El suffix indica donde comienza nuestro directorio, es
# decir la raíz. Para nuestro caso será dc=automatica, dc=com, ya que
# como se puede ver en la figura es de allí de donde ha de colgar toda
```

```
# la información que deseamos agregar.
suffix          "dc=automatica,dc=com"

# Profesor: Identificador del administrador de la base de datos que
# para nuestro caso se llama manager y debe de colgar de la raíz.
rootdn          "cn=Manager,dc=automatica,dc=com"

# Profesor: Clave del administrador en texto claro, mas adelante la
# explicaremos como podemos cifrar la clave
rootpw          secret

# Profesor: Reglas de acceso al directorio, volveremos mas adelante
# sobre ellas
access to attrs=userPassword
     by dn.regex="cn=Manager,dc=automatica,dc=com" write
          by anonymous auth
          by self write
          by * none
access to *
     by dn="cn=Manager,dc=automatica,dc=com" write
          by * read
access to dn.regex=".*,ou=Roaming,o=morsnet"
     by dn="cn=Manager,dc=automatica,dc=com" write
          by dnattr=owner write
```

10. Arrancando y deteniendo nuestro servidor

Con la configuración anterior es suficiente en la mayoría de los casos (salvo por ciertas reglas de acceso) ahora podemos lanzar nuestro servidor que utilizara este fichero para arrancar. Para arrancar lo hacemos indicando las rutas tanto del demonio como del fichero de configuración con la opción –f:

```
$ /usr/local/libexec/slapd –f
/usr/local/etc/openldap/sldap.conf
```

Si deseásemos arrancar el servidor en un puerto diferente al que viene por defecto lo hacemos con la opción –h ldap://:puerto/. Si en nuestro caso quisiésemos lanzar el servidor en el puerto 5001 la sentencia completa sería:

```
$ /usr/local/libexec/slapd –f /usr/local/etc/openldap/sldap.conf -h ldap//:5001
```

Para detenerlo, matamos al demonio lapd, con la orden:

```
$ kill -9 `ps -ax | grep slapd | cut –c 1-5`
```

11. Creando los ficheros LDIF

Para poder agregar objetos a nuestro árbol, debemos de escribirlo en un fichero con extención ldif y luego añadirlo al directorio. Dentro de este archivo estarán el o los objetos que deseamos ingresar a nuestro directorio y estos objetos se definen escogiendo su tipo de unos de los objetos que vienen descritos en los ficheros de esquemas.

Analizando nuestro caso, los objetos que deseamos crear serían:

- Un objeto principal que encabece nuestra organización. (Para automatica).
- Varios objetos que nos permitan crear unidades funcionales dentro de la organización. (Para isi. it, labisi y labit)
- Objetos que nos permitan guardar los datos de usuarios de Linux (pepe, maria, Juan y ana).

Una que tenemos claro debemos de buscar los objetos que se adecuen a esto. Buscando dentro de los esquemas que están en /usr/local/etc/openldap/schema/ podemos encontrar los apropiados. Para nuestro caso hemos encontrado dentro de /usr/local/etc/openldap/schema/core.schema dos objetos que nos sirven que son:

Estas son **organization**:

```
objectclass ( 2.5.6.4 NAME 'organization'
     DESC 'RFC2256: an organization'
     SUP top STRUCTURAL
     MUST o
     MAY ( userPassword $ searchGuide $ seeAlso $ businessCategory $
```

```
          x121Address $ registeredAddress $ destinationIndicator $
          preferredDeliveryMethod $ telexNumber $
teletexTerminalIdentifier $
          telephoneNumber $ internationaliSDNNumber $
          facsimileTelephoneNumber $ street $ postOfficeBox $ postalCode
$
          postalAddress $ physicalDeliveryOfficeName $ st $ l $
description ) )
```

Y **organizationalUnit**:

```
objectclass ( 2.5.6.5 NAME 'organizationalUnit'
     DESC 'RFC2256: an organizational unit'
     SUP top STRUCTURAL
     MUST ou
     MAY ( userPassword $ searchGuide $ seeAlso $ businessCategory $
          x121Address $ registeredAddress $ destinationIndicator $
          preferredDeliveryMethod $ telexNumber $
teletexTerminalIdentifier $
          telephoneNumber $ internationaliSDNNumber $
          facsimileTelephoneNumber $ street $ postOfficeBox $
postalCode $
          postalAddress $ physicalDeliveryOfficeName $ st $ l $
description ) )
```

Y dentro de /usr/local/etc/openldap/schema/nis.schema encontramos uno que se adecua a los que necesitamos para las cuentas de los usuarios que es **posixAccount**:

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount'
     DESC 'Abstraction of an account with POSIX attributes'
     SUP top AUXILIARY
     MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
     MAY ( userPassword $ loginShell $ gecos $ description )
)
```

De primera impresión, no queda muy claro como se puede crear un sistema de directorio con esto pero describiremos ahora a groso modo lo que son cada objeto. Primero, cada archivo *.schema contiene primero la definición de una serie de atributos como por ejemplo el que se usa para representar el nombre de la ciudad que es 'c' ó contryName, ambos igual de válidos:

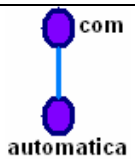
```
attributetype ( 2.5.4.6 NAME ( 'c' 'countryName' )
      DESC 'RFC2256: ISO-3166 country 2-letter code'
      SUP name SINGLE-VALUE )
```

Luego encontramos definidos una serie de objetos que ocupan esos atributos. Cada objeto posee:

- **NAME:** que es el nombre del objeto. Lo usaremos a la hora de crear nuestros elementos.
- **DESC:** Es una descripción del objeto.
- **SUP:** Indica de quien se deriva el objeto. Si un objeto se deriva de otro, debemos de poner en la declaración de un elemento de ese tipo, el objeto padre.
- **STRUCTURAL ó AUXILIARY:** Indica si el objeto puede ser usado como un nodo del árbol (si es que es estructural) o solo como hoja (si es auxiliary).
- **MUST:** Indica cuales atributos son obligatorios poner a la hora de declarar un objeto. Por ejemplo en la clase organization, el atributo 'o' que es el nombre de la organización es obligatorio.
- **MAY:** Son los atributos que son opcionales. Hay que recordar que los atributos que se indican en cada objeto están definidos en el archivo .schema al que pertenece el objeto.

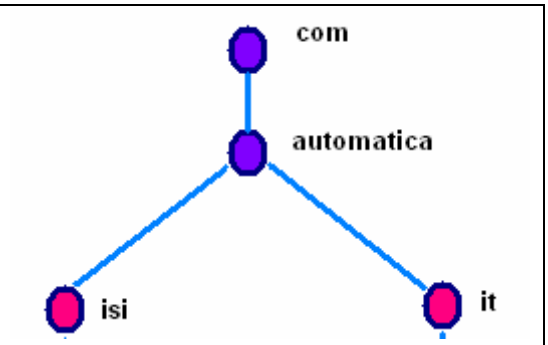
Ahora solo nos hace falta crear el fichero que contendrá toda la información, haciendo uso de los objetos que hemos seleccionado. Iniciamos creando una entrada para la raíz.

```
dn: dc=automatica,dc=com
objectclass: dcObject
objectclass: organizational
dc: isi
o: Departamento de Automatica
```



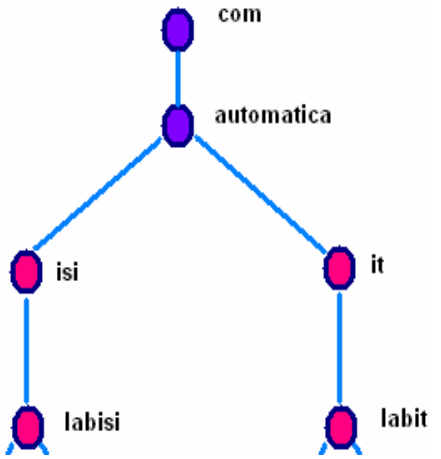
Es conveniente que si vamos a utilizar el objeto como nodo, lo declaremos tambien del tipo dcObject, para usemos el atributo 'dc' para indicar su 'dn' y el atributo 'o' para describir lo que es. Luego añadimos los nodos que cuelgan de dc=automatica, dc=com. Empleando el tipo dc igual que en el caso anterior.

```
dn: dc=isi,dc=automatica,dc=com
objectclass: dcObject
objectclass: organizationalUnit
dc: isi
ou: Area de Ingenieria en
Sistemas

dn: dc=it,dc=automatica,dc=com
objectclass: dcObject
objectclass: organizationalUnit
dc: it
ou: Area de Ingenieria en
Telematica
```



Ahora agregamos los laboratorios de cada una de las áreas:

```
dn:
dc=labit,dc=it,dc=automatica,dc=com
objectclass: dcObject
objectclass: organizationalUnit
dc: labit
ou: Laboratorio IT

dn:
dc=labisi,dc=it,dc=automatica,dc=com
objectclass: dcObject
objectclass: organizationalUnit
dc: labisi
ou: Laboratorio ISI
```



Y por último debemos añadir los objetos que han de identificar a los usuarios de los diferentes laboratorios:

```
dn: uid=pepe,dc=labisi,dc=isi,dc=automatica,dc=com
uid: pepe
cn: Pepe Mendez
objectClass: account
objectClass: posixAccount
objectClass: top
userPassword: pepe123
loginShell: /bin/bash
uidNumber: 1010
gidNumber: 1010
homeDirectory: /home/pepe

dn: uid=maria,dc=labisi,dc=isi,dc=automatica,dc=com
uid: maria
cn: Maria Hernandez
objectClass: account
objectClass: posixAccount
objectClass: top
userPassword: maria123
loginShell: /bin/bash
uidNumber: 1011
gidNumber: 1011
homeDirectory: /home/maria

dn: uid=juan,dc=labit,dc=it,dc=automatica,dc=com
uid: juan
cn: Juan Vega
objectClass: account
objectClass: posixAccount
objectClass: top
userPassword: juan123
loginShell: /bin/bash
uidNumber: 1012
gidNumber: 1012
homeDirectory: /home/juan

dn: uid=ana,dc=labit,dc=it,dc=automatica,dc=com
```

```
uid: ana
cn: Ana Conde
objectClass: account
objectClass: posixAccount
objectClass: top
userPassword: ana123
loginShell: /bin/bash
uidNumber: 1013
gidNumber: 1013
homeDirectory: /home/ana
```

Con esto tendríamos lista ya la estructura necesaria para crear nuestro directorio con información de estos alumnos. En el siguiente apartado mostrare como añadir los datos al directorio, de momento es importante recordar lo siguiente:

- Existen mucha clase de objetos declarados en los esquemas para poder crear la estructura que deseemos.
- Los datos deben de irse introduciendo a medida que se crea el árbol iniciando por la raíz hasta llegar a las hojas.
- Si un objeto va a contener hojas debe de ser de tipo estructura.
- Todo el contenido de nuestro árbol puede escribirse en un mismo archivo o en diferentes siempre que tengan la extensión .ldif. Para nuestro caso. Colocaremos todo el contenido en un mismo archivo y lo llamaremos usuarios.ldif

12. Trabajando sobre la base de datos

En estos momentos tenemos arrancado el servidor y tenemos creado el archivo usuarios.ldif en el cual hemos definido los diferentes elementos para formar nuestro directorio. Ahora lo que nos queda es saber como operar sobre el directorio que hemos creado. A continuación describiré como realizar operaciones de añadir, modificar, borrar y buscar en LDAP.

Añadir elementos al directorio:

Para añadir el contenido de un archivo ldif al directorio lo hacemos con la orden:

```
# ldapadd –x –H ldap://nombre_o_ipdehost:puerto –f
 nombredelFichero –D rootdn –w password
```

La opción –x nos ayuda a indicar que estamos haciendo autenticación simple. –H es para indicar la máquina y el puerto al cual deseamos conectarnos. La opción –f nos sirve para indicar el fichero que deseamos añadir. Con la opción –D, decimos como que usuario deseamos que se ejecute la orden y con –w añadimos la contraseña para ese usuario. Si empleamos una –W (w mayúscula) no debemos introducir la contraseña en la orden ya que se nos pedirá luego de pulsar enter. En nuestro caso, para añadir el fichero usuarios.ldif que esta en mi directorio home haríamos:

```
# ldapdadd –x –H ldap://localhost:5001 –f /home/denis/
  usuarios.ldif  –D  “cn=Manager,dc=automatica,dc=com”  –w
secret
```

Una vez añadido el fichero a la base de datos, podemos eliminar el fichero usuarios.ldif ya que su contenido ahora forma parte del directorio.

Buscar elementos en el directorio

Para poder realizar búsquedas o consultas sobre un directorio los hacemos a través de la orden ldapsearch. La sintaxis básica para realizar las consultas es

```
#    ldapsearch    –x    -LLL    –b    ‘base_de_busqueda’
‘condicion_a_buscar’
  –H    ldap://nombre_o_ipdehost:puerto    [atributo1]    …
[atributon]
```

Las opciones –x y –H tienen los mismos usos que en ldapadd. Igual que en ese caso se puede emplear la opción –D y –W(o -w) para indicar en nombre de que usuario deseamos hacer la consulta. Por defecto se pueden omitir porque cualquier usuario puede leer la base de datos (luego deberemos modificar eso). Aquí lo nuevo

es -LLL que nos visualiza solo el contenido del fiechero sin información del directorio; –b que nos sirve para indicar desde que nodo del árbol hacia abajo deseamos realizar la búsqueda, si se omite la búsqueda se realiza desde la base que en nuestro caso sería “dc=automatica,dc=com”.

La condición_a_buscar, es el parámetro que deseamos que cumpla aquello buscado. Aquí se puede comparar con tipos de objetos o atributos. Para mayor claridad haremos algunas búsquedas sobre nuestro directorio:

```
$ ldapsearch –x –LLL –b ‘dc=labit,dc=it,dc=automatica,dc=com’
  ‘uid=denis’ cn loginShell uidNumber homeDirectory
```

Con la orden anterior buscaríamos dentro de labit aquel elemento que tenga un atributo uid que sea igual a denis. Y si lo encuentra, queremos que nos imprima por pantalla su cn, loginShell, uidNumber y homeDirectory. Si no se especifican los que atributos que deseamos visualizar, se visualizará todo el contenido del objeto. Otro ejemplo sería.

```
$ ldapsearch –x –LLL ‘objectclass=posixAcount’
```

Esto buscaría desde la raíz del directorio todos los objetos que sean de tipo posixAcount (todas las cuentas de los usuarios) y visualizaría todo su contenido.

Otros ejemplos serían de condiciones serían:

Condición	Significado
`dc=isi`	Aquellos objetos cuyo elemento dc sea isi.
`Objectclass=*`	Los objetos de cualquier tipo.
`(!(sn=Garcia))`	Los objetos cuyo sn (Apellido) no sea Garcia.
`(|(cn=andres)(cn=pedro))`	Los objetos cuyo cn sea andres o pedro.
`telephoneNumber=31155*`	Los objetos que incluyan el atributo telephoneNumber y que inicien con 31155
`uid~=Juan`	Los objetos cuyo uid se asemeje a Juan.

Modificar y Borrar elementos del directorio

Para modificar o borrar los datos de nuestro directorio lo haremos a través de la orden ldapmodify, por ello, explicaremos su sintaxis y luego indicaremos como realizar los borrados y las modificaciones en con esta orden.

La sintaxis de esta orden es:

```
$ ldapmodify –x –H ldap://nombre_host:puerto –D rootdn –w
  passwd –f nombre_fichero
```

Como se puede observar su sintaxis es prácticamente idéntica a la de ldapadd y al igual que en esta, es necesario que añadamos el nombre del root y su clave porque este es el único que puede hacer cambios sobre la estructura del directorio. Los cambios que se desean realizar se especifican en un fichero de tipo ldif y es este fichero el que ha de indicar si la operación que se desea realizar es un borrado o una modificación.

Si lo que deseamos hacer es eliminar por ejemplo a Ana que es un usuario del laboratorio de ISI, crearíamos un fichero que su borrar.ldif que contendría:

```
dn: uid=ana,dc=labisi,dc=isi,dc=automatica,dc=com
changetype: delete
```

Y ejecutaríamos la orden:

```
$ldapmodify –x –H ldap://localhost:5001 –D “cn=Manager,
dc=autonmatica dc=com” –w secret –f /home/denis/borrar.ldif
```

Si por el contrario lo que queremos es modificar el contenido del usuario Juan, la sintaxis genérica sería la siguiente:

```
dn: dn de la entrada a modifica
changetype: modify
tipoDeModificacion: atributoAModificar
[atributoAModificar: valor]
```

Se puede indicar una sola modificación o varias separándolas con un guión. Los tipos de modificaciones que se pueden hacer sobre los atributos de un determinado objeto son add, delete y replace. Un ejemplo de esto sería:

```
dn: uid=juan,dc=labit,dc=it,dc=automatica,dc=com
changetype: modify
replace: cn
cn: Juan Montalvo
-
delete: loginShell
loginShell: /bin/bash
-
add: description
description: Esta entrada fue modificada
```

En este ejemplo lo que hemos hecho cambiar el apellido de Juan de Vega a Montalvo, borrar el atributo loginShell de este y añadir un atributo llamado description. Ahora solo hay que guardar esto en un fichero y utilizar ldapmodify como en el caso anterior para hacer efectivos los cambios en el directorio.

13. Modificando la Estructura del directorio

Hasta este momento lo único que hemos hecho es modificar los datos para un nodo concreto, quedando la estructura del árbol intacta. Si lo que queremos es efectuar cambios que afecten a las estructuras del árbol hay que usar un nuevo valor de *changetype*, llamado modrdn. En este caso el modelo general es el siguiente:

```
dn: dn de la entrada a modifica
changetype: modrdn
[newrdn: nuevo RDN del elemento]
[deleteoldrdn: (0 | 1)]
[newsuperior: dn del nuevo nodo padre]
```

Dentro de un DN se conoce como RDN al *Relative Distinguished Name*, es decir, el valor del atributo que diferencia a un elemento de los elementos del mismo nivel que cuelgan del mismo nodo padre. Así dentro de los alumnos que están dentro de labisi, su RDN sería su uid, en nuestro ejemplo tendríamos el RDN Juan y el RDN maria.

Supongamos que tenemos el DN

```
dn: uid=juan, dc=labisi, dc=isi,dc=automatica,dc=com
```

Y queremso cambiar juan por juancito, el fichero ldif que deberiamos crear sería:

```
dn: uid=juan, dc=labisi, dc=isi,dc=automatica,dc=com
changetype: modrdn
newrdn: uid=juancito
deleteoldrdn: 1
```

Si hubiésemos puesto deleteoldrdn: 0, lo que habríamos hecho es conservar también el valor anterior de cn. Es decir, el nodo tendría dos valores diferentes para el atributo uid pero solo uno sería el RDN.

Con lo anterior, solo hemos cambiado el RDN del elemento pero aún se mantienen la estructura intacta, si lo que queremos es cambiar la ubicación del elemento, por ejemplo, cambiar a Juan de labisi a labit deberiamos hacer:

```
dn: uid=juan, dc=labisi, dc=isi, dc=automatica, dc=com
changetype: modrdn
newrdn: uid=juancito
deleteoldrdn: 1
newsuperior: dc=labit, dc=it, dc=automatica, dc=com
```

14. Cifrar la clave del Administrador

Antes de continuar, vamos a cifrar la clave del administrador que en el fichero slapd.conf es secret pero que esta en texto claro. Para ello emplearemos una herramienta que viene con ldap que es slappasswd. La sintaxis sería:

```
$ slappasswd –s clave
```

Donde clave es el valor de nuestra clave de administrador definida en rootpw. Debemos poner este resultado en el fichero de configuración slapd.conf en lugar donde habiamos escrito nuestra clave que era secret. Aunque nuestra clave este cifrada, a la hora de escribirla la seguiremos escribiendo como siempre es decir, escribiremos secret.

15. Control de Acceso al árbol de directorio

Uno de los aspectos que dejamos para después cuando se estudió el fichero de configuración de ldap fueron las directivas access. Esta directivas son las que nos van a permitir gestionar el acceso de usuarios a nuestro servidor. Iniciaremos recordando las directivas access que había en nuestro fichero de configuración:

```
Access to attrs=userPassword
     by dn.regex="cn=Manager,dc=automatica,dc=com" write
          by anonymous auth
          by self write
          by * none
access to *
     by dn="cn=Manager,dc=automatica,dc=com" write
          by * read
access to dn.regex=".*,ou=Roaming,o=morsnet"
     by dn="cn=Manager,dc=automatica,dc=com" write
          by dnattr=owner write
```

Como se puede observar, el patrón que sigue una directiva de acceso es:

1. Indicar el recurso al que se da acceso. Con la directiva **access to …**
2. Indicar quién(es) tiene(n) acceso a ese recurso (by dn=…).
3. Indicar que clase de acceso (lectura, escritura))tiene el usuario mencionado (dy dn=… write).

Indicar a qué recurso se da acceso

Esta es la primera parte de la directiva. La forma de expresarlo es a través de **access to** más una expresión que indique uno o varios usuarios del sistema. El caso más general, en el que se da acceso a todo, es el asterisco (access to *). Sin embargo, se puede ser más específico, para ello contamos con dos posibilidades:

1. **dn[.basic-style]=<regex>:** Permite indicar un conjunto de recursos mediante una expresión regular que englobe sus diferentes DNs. El valor basic-style puede ser:

- regex, de modo que acontinuación venga una expresión regular que englobe un conjunto de recursos. Por ejemplo: Para indicar que se da acceso a todos los recursos que están dentro de it sería:

  dn.regex="*,dc=it, dc=automatica, dc=com"

  Con esto indicamos todos los recursos dentro de it que son labit y los usuarios dentro de labit.
- exact, para indicar un recurso en concreto, sin poder acceder a sus nodos hijos. Ejemplo:

  dn.exact="dc=labit,dc=it, dc=automatica, dc=com"

  Con esto estamos endicando únicamente el recurso labit y nada más. Sino se indica si regex o exact, por defecto se asume que regex.

2. **dn.<scope-style>=<DN>:** Permite indicar un subárbol a partir de un elemento. Los diferentes valores que puede tomar scope-style son:

Valor de dn	Ejemplo	Significado
dn.base=…	dn.base=" dc=automatica,dc=com"	Referencia únicamente al nodo en concreto, es decir, a dc=automatica,dc=com"
dn.one=…	dn.one=" dc=automatica,dc=com"	Referencia a los nodos que están por debajo del indicado en nuestro caso serían dc=isi y dc=it.
dn.subtree=…	dn.subtree=" dc=automatica,dc=com"	Referencia al propio nodo y a todos los que se encuentren debajo de él, en nuestro caso haría referencia a todo el árbol excepto a dc=com.
dn.children=…	dn.children=" dc=automatica,dc=com"	Semejante al anterior solo que no incluye al propio nodo, en este caso no se estaría haciendo referencia a dc=automatica, dc=com

Con esto podemos seleccionar objetos dentro del árbol para indicar distintas políticas de acceso, también pueden emplearse filtros para hacer restricciones más refinadas pero nosotros no entraremos en detalle.

También pueden hacerse restricciones sobre algunos atributos en concreto, por ejemplo, en nuestro código, no permitimos que la clave de cualquier usuario, sea vista mas que por el administrador:

```
Access to attrs=userPassword
      by dn.regex="cn=Manager,dc=automatica,dc=com" write
            by anonymous auth
            by self write
            by * none
```

Para imponer restricciones sobre los atributos, se coloca la palabra **attrs** y la lista de atributos separados por coma (en el ejemplo solo hay uno) o alguna expresión regular. Para los siguientes dos apartados seguiremos utilizando como ejemplo este caso.

Indicar a quién se le concede el acceso

Una vez que hemos indicado sobre que recurso se desea implantar una política de acceso, debemos de indicar quien tiene acceso a ese recurso. Esto se hace colocando la palabra by, seguido de un palabra reservada o una expresión que identifica al conjunto sobre el cual se especifica el permiso. En el ejemplo que se acaba de mostrar se especifican cuatro conjuntos de usuarios para los que se va a conceder un permiso en concreto estos son: `dn.regex="cn=Manager,dc=automatica,dc=com", anonymous, self, *`.

La forma de especificar los usuarios aparecen resumidos en la tabla siguiente:

Especificaciones	Significado
*	Todos los usuarios.
Anonymous	Usuarios anónimos (sin autenticar).
Users	Usuarios autenticados.
Self	Usuario asociado a la entrada sobre la que se aplica la directiva.
dn[.<basic-style>]=<regex>	Usuarios que encajan con la expresión regular que se indique.
dn.<scope-style>=<DN>	Usuarios bajo el ámbito de un DN completo.

Indicar que clase de acceso se concede

Una vez que hemos definido quién va a acceder al recurso debemos de indicar la clase de acceso que se concede a los usuarios. Continuando con el ejemplo

anterior, los permisos que se expresan aquí sobre los usuarios son `write, auth y none`. Los posibles tipos de acceso aparecen reflejados en la tabla siguiente:

Nivel	Privilegios	Descripción
None	=0	Sin acceso.
Auth	.=x	Necesario para acceder al sistema.
compare	=cx	Necesario para comparar.
search	=scx	Necesario para efectuar búsquedas.
read	=rscx	Necesario para leer.
write	=wrscx	Necesario para escribir o modificar.

En la tabla los permisos se encuentran ordenados del mas restrictivo al menos restrictivos de manera que cualquiera de ellos incluye a los que están por encima de él en la tabla (excepto none, evidentemente).

El orden de declaración de los permisos es muy importante ya que cuando se va a hacer una operación el servidor consulta los permisos en busca de la primera directiva de acceso que encaje con el recurso solicitado y con el usuario que requiere el permiso, y es ésa la que aplica, independientemente de que haya más adelante en el fichero de configuración otra directiva que sea más restrictiva o más flexible.

Ejemplo del uso de las reglas de acceso

Supongamos que ahora para refinar nuestro directorio deseamos que los alumnos sólo puedan ver información de aquellos usuarios que pertenecen a su mismo laboratorio. Es, decir que los alumnos del laboratorio de labit, sólo puedan obtener información de los otros miembros de labit pero no de labisi ni de ningún otra parte de nuestro directorio (lo mismo para los de labisi). En este caso, el archivo de configuración en la zona de directivas de acceso quedaría de la siguiente manera:

```
# Profesor: Esto ya estaba antes
access to attrs=userPassword
      by dn.regex="cn=Manager,dc=automatica,dc=com" write
            by anonymous auth
            by self write
            by * none

# Profesor: Con esto especificamos que damos acceso un nivel
por
```

```
# debajo del recurso labisi
access to dn.one="dc=labisi,dc=isi,dc=automatica,dc=com"
# Profesor: A todos los usuarios que esten un nivel por debajo
de
# labis y el acceso que se permite es de solo lectura
          by dn.one="dc=labisi,dc=isi,dc=automatica,dc=com"
read

#Profesor sucede lo mismo que para labisi
access to dn.one="dc=labit,dc=it,dc=automatica,dc=com"
          by dn.one="dc=labit,dc=it,dc=automatica,dc=com" read
access to *
     by dn="cn=Manager,dc=automatica,dc=com" write
          by * read

access to dn.regex=".*,ou=Roaming,o=morsnet"
     by dn="cn=Manager,dc=automatica,dc=com" write
          by dnattr=owner write
```

Como ahora ya hay más reglas de acceso, cuando deseemos hacer una consulta, deberemos de indicar como que usuario queremos hacer la consulta y nuestra clave. Supongamos que el usuario con uid=juan (de labisi) desea conocer la información de los demás miembros del laboratorio, entonces la orden sería:

```
$ ldapsearch –x –LLL –D ‘uid=juan, dc=labisi, dc=isi,
  dc=automatica, dc= com’ –w juan123 ‘uid=*, dc=labisi,
dc=isi,
  dc=automatica, dc= com’
```

Esto mostraría todos datos de los usuarios que esten en labisi, si por el contrario el cambiase **‘uid=*, dc=labisi, dc=isi, dc=automatica, dc= com’** por **‘uid=*, dc=labit, dc=it, dc=automatica, dc= com’**, la consulta no tendría ningún efecto.

16. Conclusiones

LDAP es un sistema muy potente que puede ser empleado para muchas cosas que a lo largo de las practicas iremos descubriendo, por ahora podemos darnos cuenta que nos ha de ser bastante útil para llevar la información de los miembros de la organización de una empresa.

17. Empleando una interfaz más amigable con LDAP

Bueno, hasta ahora ha sido bastante tedioso el estar añadiendo o consultando cosas en nuestro directorio LDAP. Existe una forma gráfica para trabajar con LDAP y es a través de una interfaz Web, sin embargo esto requiere tener instalado Apache con PHP lo cual haremos en la práctica 7, por lo cual, de momento trabajaremos con LDAP a través de la línea de comandos.

18. Ejercicio Práctico

Tu jefe de la empresa ANONIMOS S.A., te llama para presentarte un nuevo problema. Su necesidad es poder tener a mano la información de los trabajadores de su empresa, sean teléfonos, dirección y otros datos, pero que no quiere tener que ingresar los mismos datos para todos los usuarios. Tú le dices que la mejor forma es instalar un servicio de directorios LDAP, dentro del cual podrá dividir los usuarios en dos grupos que serían, los de **ejecutivos** y **empleados**.

Tu misión será crear un servicio de directorio que tendrá como raíz **anonimos.ni** dentro de el deberás crear las dos sub-organizaciones que serán ejecutivos y empleados, y deberás añadir al menos dos personas a cada grupo (puedes usar el objeto Person). Debes garantizar que los ejecutivos puedan ver la información tanto de los demás ejecutivos como de todos los empleados, pero los empleados solo podrán ver información de los demás empleados.

Selecciona tu mismo los datos y la información de cada persona que agregues al directorio LDAP.

19. Referencias

- idefix.eup.uva.es/Manuales/OpenLdap/openldap.pdf
- http://www.it.uc3m.es/aat/
- http://dns.bdat.net/bind-9/DNS-HOWTO-9-es/

PRÁCTICA 5: CONFIGURACION DE UN SERVIDOR DNS

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción

Una vez que hemos terminado de instalar un servicio de directorio como fue LDAP, nos corresponde ahora iniciar la instalación de otro servicio de directorio que esta aún más difundido y que diariamente es utilizado por millones de personas en todo el mundo. Este es el DNS (Servicio de Nombres de Dominio).

2. Objetivo

El objetivo de ésta práctica es configurar un servidor DNS tanto de cacheo, autorizado y secundario.

3. ¿Qué es el DNS?

Ya en la parte teórica hemos abordado de una manera bastante amplia lo que es el DNS y su forma de funcionamiento, por lo cual nos centraremos en decir que un DNS se encarga básicamente en traducir una dirección canónica (p. e. www.yahoo.es) en su dirección IP y viceversa. Antes de entrar a ver en definitiva como configurar este servicio, es bueno aclarar que esta relación (Nombre - IP) no tiene porque ser única, como se verá en el ejemplo.

4. Instalación de BIND

Entremos en materia, existen varios servidores DNS que se pueden instalar, sin embargo, nosotros emplearemos BIND (Berkeley Internet Name Domain), en su última versión. Lo primero será obtener este software y para ello podemos descargarlo de Internet desde la página de BIND que es http://www.isc.org/products/BIND/ e instalarlo a como ya hemos hecho en las prácticas anteriores que sería:

```
$ ./configure
$ make
$ make install
```

O una manera aún más simple y rápida que es desde una consola en la cual estemos como root hacer:

```
$ apt-get install bind9 bind9-doc dnsutils
```

Con esto deberíamos tener ya instalado BIND y funcionando con la configuración mínima. Si lo hacemos de la primera forma (./configure …..) normalmente no crea el archivo de configuración pero eso no es ningún problema porque nosotros vamos a crear nuestros propios archivos de configuración.

La instalación de BIND deberá de dejarnos instalados tres programas que son:

1. **named:** Que es el demonio DNS.
2. **named-checkconf:** Que es un analizador de sintaxis que nos permitirá revisar si nuestro archivo de configuración está bien diseñado.
3. **named-checkzone:** Que es un analizador sintáctico de las zonas que creamos.

5. El archivo de configuración de BIND9

Una vez que tenemos instalado BIND pasamos a ver el archivo de configuración del demonio DNS **named** que es named.conf. Este archivo puede estar en /etc/named.conf o no estar. Si no se encuentra lo podemos crear nosotros mismos adecuándolo a las necesidades que tengamos. Para conocer como crear nuestro propio archivo de configuración describiremos como viene estructurado:

```
# Profesor: Archivo de configuración del demonio named. Este archivo
# posee varias secciones, cada una con diversas opciones que pueden
# ser vistas en el manual, aquí sólo comentaremos las más interesantes
# para estudiar el funcionamiento de BIND.

# Definición de listas de acceso. Consisten en definir agrupaciones de
# de direcciones bajo un solo nombre. En el ejemplo siguiente siempre
```

```
# que escribamos slaves, nos estaremos refiriendo a la máquina
# 193.168.0.1. Esto puede contener más elementos u otro nombre.
Pueden
# definirse varias listas de acceso.

acl “slaves” {
     193.168.0.2;
};

# Área de opciones común a todas las zonas siguientes
options {
# Lugar donde se guardará el fichero que contiene el PID del
proceso
pid-file “/dir/file.pid”;
# Indica el tamaño de la pila en este caso 30 Megabytes
stacksize 30M;
# Máximo tamaño de la zona de datos en este caso de 20
Megabytes
datasize 20M;
# Permite la transferencia en paralela de las zonas a los
esclavos
transfer-format many-servers;
# Acota globalmente las transferencias de zonas a los
servidores
# esclavos definidos en el acl
allow-transfer { slaves; };
};

# A continuación, se define cada una de las zonas que se desee.
Cada
# una de las zonas puede contener diversas opciones que
comentamos a
# continuación.
# Nombre de la zona mizonaejemplo.com
zone “mizonaejemplo.com” {
# Indica que nuestro servidor esta primario, maestro o
autorizado para
```

```
# esta zona. Al configurar un servidor secundario este valor se
pone
# a slave que es diferente del slaves que definimos en acl.
type master;
# Fichero con la configuración de la zona. La ruta es absoluta.
file “/dir/db.mizonaejemplo.com”;
# Indica que se permiten consultas externas a la zona.
allow-query { any; };
# Posibilita la transferencia de información a los servidores
secundarios de las zonas.
allow-transfer { slaves; };
};

# Siguiente zona……
# También deben de añadirse las zonas inversas de las redes
sobre las
# que se tiene dominio aunque esto no es obligatorio.
```

Sino a quedado clara la forma en que debe ser utilizado el servidor DNS no te preocupes porque luego haremos varios ejemplos con los que espero te queden más claros los diferentes conceptos.

Por ahora nos vamos a enfocar en la opción **file** de **zone**. Debemos de tener claro lo siguiente, siempre que para una zona que definamos seamos autorizados (es decir **types master**), debemos de definir la configuración de esa zona en un archivo en el que se especifica la configuración de dicha zona.

¿Qué configuración es la que se debe especificar por cada zona?

Antes que todo vamos a decir que un archivo de zona está compuesto por diferentes tipos de registros entre otros pueden ser:

Tipo	Nombre	Función
SOA	Stara Of Authority	Define una zona representativa del DNS.
NS	Name Server	Identifica los servidores de zona, delega subdominios.

A	Dirección IPv4	Traducción de nombre a dirección IPv4.
AAAA	Dirección Ipv6	Traduce de nombre a dirección IPv6.
MX	Mail eXchanger	Controla el enrutado del correo.
CNAME	Canonical Name	Nick o alias para un dominio.
PTR	Puntero	Traducción de dirección a nombre
LOC	Localización	Localización geográfica y extensión.
TXT	Texto	Comentarios o información sin cifrar.

La idea básica es que dentro de una zona coma la definida anteriormente que era “mizonaejemplo.com” puedo querer tener una página web que sería www.mizonaejemplo.com o un servidor ftp que puede ser ftp.mizonaejemplo.com, un servidor de correo que puede ser smtp.mizonaejemplo.com u otras páginas situadas en otros servidores como puede ser leon.mizonaejemplo.com y Managua.mizonaejemplo.com. Entonces lo que necesitamos es que cuando un usuario pregunte por cualquiera de estos recursos, el servidor DNS sepa responder a que dirección IP exacta debe ir el cliente para obtener dicho recurso.

Puesto que todos estos recursos están dentro la zona (mizonaejemplo.com) el archivo que se crea para esta zona, en el servidor que esta autorizado sobre ella, debe saber decir en que dirección IP están todos estos recursos asociados a esta zona, que no tiene porque estar en la misma máquina. Por así decirlo, el servidor que es autorizado para una zona debe dar respuesta a todas las peticiones que se hagan sobre un recurso que este dentro de esa zona. Esta información la guarda en el archivo de la zona.

6. Configurando un servidor de solo cacheo

Un servidor de solo cacheo es aquel que no tiene autoridad más que sobre el mismo pero esto nos permite agilizar las consultas DNS ya que cada vez que alguien hace una consulta sobre un recurso determinado, queda almacenada la respuesta en el servidor. Con esto, si después alguien hace una consulta sobre el mismo recurso, el servidor mismo podrá responder a esa petición sin necesidad de consultar al servidor autorizado para ese recurso. Incluso, continuará respondiendo sobre ese recurso aún cuando se caiga el servidor autorizado.

El archivo de configuración **named.conf** para un servidor de cacheo que podríamos instalar en nuestro laboratorio o incluso en nuestra casa sería:

```
/*Configuración del demonio DNS named para un servidor de
cacheo*/

options {
      # Indica que cualquier petición debe enviarse a los
servidores
      # de la lista.
      forward first;

      # Lista de servidores a los cuales redireccionamos.
Pueden ser
      # los de nuestro ISP o cualquier otro conocido. Para la
practica
      # usaremos el servidor DNS configurado por el profesor
para esta
      # practica.
      forwarders {
           172.29.20.46;
      };
};

# Definimos la zona inverse para nuestra máquina para la cual
somos
# maestros evidentemente y cuyo fichero de configuración lo
tenemos
# en /home/denis/dns/db.cacheo
zone "0.0.127.in-addr.arpa" {
      type master;
      file "/home/denis/dns/db.cacheo";
};
```

Ahora que tenemos listo nuestro fichero de configuración para nuestro servidor de cacheo, debemos de crear el archivo para el cual somos autorizados. En este caso, sólo somos maestros para nuestra zona inversa con lo que solo deberemos de crear el archivo **db.cacheo** en **/home/denis/dns**. El contenido de este archivo sería:

```
$TTL 3D
@ IN SOA ns.it.automatica.com. hostmaster.it.automatica.com. (
        2007060101      ; Serial
                8H      ; Refresh
                2H      ; Retry
                4W      ; Expire
                1D)     ; Minimum TTL

      NS      ns.it.automatica.com.
1     PTR     localhost.
```

Este pequeño ejemplo nos servirá para aclarar algunas cosas acerca de la configuración de los archivos asociados a cada zona. Primero, todo lo que se encuentre después de un ';' (punto y coma) es un comentario.

- `$TTL 3D`: Esta directiva es obligatorio a partir de la versión 9 de bind, indica el tiempo de vida (o de validez) de la información del fichero. Pasado ese tiempo, la información deberá refrescarse. Por defecto se usan segundos 604800 segundos (equivale a siete horas), pero pueden usarse también semanas ($TTL 1w), días ($TTL 7d), horas ($TTL 168h) y minutos ($TTL 10080m). Estas mismas abreviaturas se utilizan en el registro SOA.

- `@ IN SOA ns.it.automatica.com. hostmaster.it.automatica.com.`: El registro SOA proclama información relevante sobre la autoridad de un dominio al servidor de nombres. Es siempre el primer recurso en el fichero de zona. El símbolo @ equivale al nombre de la zona. Este sería el esqueleto de este registro:

```
@ IN SOA <primary-name-server> <hostmaster-email> (
<serial-number>
<time-to-refresh>
<time-to-retry>
<time-to-expire>
<minimum-TTL>)
```

El servidor de zonas que es representativo de este dominio `<primary-name-server>` y el correo electrónico de la persona a contactar sobre este espacio de nombres `<hostmaster-email>` (Esta debe ser una dirección de correo válida). Ambas deben de poner como direcciones absolutas es decir, deben terminar en punto (.).

El campo `<serial-number>` es un número que se incrementa cada vez que se modifica el fichero. Es común usar la fecha del día más dos dígitos que indican el número de cambios. En el ejemplo anterior se dice que esta es la primera versión del 01 de Junio del 2007. Este valor es usado por los servidores esclavos para determinar si tiene información anticuada que debe actualizarse. El campo `<time-to-refresh>`

indica a los servidores secundarios, cuanto tiempo deben de esperar para preguntar al servidor maestro si ha hecho algún cambio en la zona.

El campo `<time-to-retry>` especifica a los servidores esclavos cuanto el intervalo de tiempo a esperar antes de solicitar una actualización en el caso de que el servidor de nombres principal no esta respondiendo. Si el servidor maestro no ha respondido a la petición antes de que expire el tiempo del campo `<time-to-expire>`, el esclavo dejará de actuar como servidor representativo ese espacio de nombres (zona).

El campo `<minimum-TTL>` solicita a otros servidores de dominio que almacenen en su caché la información de esta zona, al menos por la cantidad de tiempo especificada.

- `NS ns.it.automatica.com.`: Indica que servidor de nombres tiene autoridad sobre este dominio. Debe quedar claro que cada dato que se escriba a la izquierda o la derecha del indicador del tipo de registro (en este caso NS y PTR en el siguiente) puede escribirse con su nombre absoluto, en cuyo caso debe de finalizar con un punto (como es el caso de ns.it.automatica.com.) o con su nombre relativo a la zona que pertenece, es decir escribir únicamente la parte del nombre que lo identifica dentro de la zona. Por ejemplo si tenemos el recurso www.mizonaejemplo.com y la zona se llama mizonaejemplo.com el nombre relativo del recurso sería www y sería BIND el que se encargue de añadir el resto. Esto BIND lo hace con todos aquellos que no finalicen con punto (.).

  En el ejemplo `NS ns.it.automatica.com.` al no haber nada a la izquierda de NS BIND coloca allí el valor del @ (que es el nombre de la zona) que para el ejemplo es 127.0.0

- `1 PTR localhost.`: Recordemos que el registro PTR indica conversión de IP a nombre de dominio. Nótese que en este caso si hay algo a la izquierda del tipo PTR. Como tampoco termina en punto, BIND le añadirá el nombre de la zona (valor de @) con lo que este registro quedará: `127.0.0.1 PTR localhost.` Local host si tiene un punto al final para evitar que BIND agregue el nombre de la zona.

7. Arrancando el servidor y haciendo consultas

Con las configuraciones hechas hasta ahora, podemos arrancar nuestro servidor y debería de funcionar perfectamente. Para ejecutarlo vamos a lanzarlo como cualquier otro demonio en Linux:

```
$ /etc/init.d/bind9 start # stop para detenerlo
```

Ahora debemos de configurar nuestra máquina para que todas las consultas las realice a nuestro servidor DNS. Para ello añadimos nuestra dirección de loopback al fichero /etc/resolv.conf el cual contiene en orden de uso, la lista de los servidores DNS a usar por nuestra máquina. Este archivo debe de quedar así:

```
nameserver   127.0.0.1
```

Con esto, nuestro servidor esta listo y funcionan. Para probarlo, naveguemos por Internet un tiempo, luego el profesor apagará el servidor DNS del laboratorio y todas las máquinas deberán seguir navegar por Internet sólo sobre aquellas páginas que ya se hayan visitado con anterioridad.

Con esto, tendremos listo ya un servidor DNS de cacheo que normalmente son bastante usados en empresas pequeñas e incluso en tu propia casa si tienes conexión a Internet puedes montar uno y las cosas habrán de ir un poco más rápido.

8. Configurando nuestra propia zona

Hasta ahora, nuestro servidor DNS solo da respuestas de otras zonas ya que el no posee ninguna. Ahora vamos a añadir una zona para le preguntemos a él sobre esa zona y sea el quien nos responda. En nuestro caso vamos a crear una zona llamada unanleon.edu.ni y haremos que nuestro servidor responda a algunas peticiones sobre esa zona. Si bien es cierto esta zona ya existe, pero como el primer servidor DNS que se consultará es el nuestro, al tener nosotros este registro seremos nosotros quienes contestemos sobre las preguntas hechas por esta zona.

Para este ejemplo, hemos supuesto que la UNAN-León poseo un rango direcciones públicas clase C que es la red 193.168.0.0. El fichero de configuración quedaría de esta manera:

```
/*Configuración del demonio DNS named para un servidor
Primario*/
# Opciones generales

acl “slaves” {
     193.168.0.2;
};

options {
      forward first;
       forwarders {
            172.29.20.46;
      };
};

# Declaracion de la zona de unanleon.edu.ni
zone "unanleon.edu.ni" {
      type master;
      file "/home/denis/dns/db.unanleon.edu.ni";
      allow-query { any; };
      allow-transfer { slaves; };
};

# Declaracion de la zona inversa de unanleon.edu.ni
zone "0.168.193.in-addr.arpa" {
      type master;
      file "/home/alcala/dns/pz/193.168.0";
      allow-transfer { slaves; };
};

# Zona inversa de localhost
zone "0.0.127.in-addr.arpa" {
      type master;
      file "/home/denis/dns/db.cacheo";
```

En general debería quedar claro lo que hemos agregado pero, por cualquier duda, comentaré algunos elementos. Somos autorizados para la zona unanleon.edu.ni ya que esta en nuestro servidor (`type master`) el archivo de configuración esta en `"/home/denis/dns/db.unanleon.edu.ni"` permite consultas de todos y y solo permitimos transferencias a los esclavos que para nuestro caso solo puede ser la máquina 193.168.0.2

Igualmente definimos una zona inversa de la cual también somos representantes que es la red 193.168.0.0. Los archivos de las zonas inversas las guardamos dentro de una carpeta que hemos creado llamada pz para diferenciarlos de los archivos de las zonas directas.

Ahora debemos crear y configurar los archivos para estos registros. Primero, el archivo db.unanleon.edu.ni quedaría:

```
;Archivo de datos de la zona externa de unanleon.edu.ni
$TTL 604800
@ IN SOA unanleon.edu.ni. hostmaster.unanleon.edu.ni. (
     2007052501
     30
     1m
     4m
     5m )
          NS    ns1.unanleon.edu.ni.
          NS    ns2.unanleon.edu.ni.
          MX    1 ns1.unanleon.edu.ni.
          MX    2 ns2.unanleon.edu.ni.

Localhost. A    127.0.0.1
unanleon.edu.ni. A   193.168.0.1
ns1       A     193.168.0.1
ns2       A     193.168.0.2

www       A     193.168.0.3
ftp       A     193.168.0.3
smtp      A     193.168.0.4
pop3      A     193.168.0.4
```

Como podemos ver, de este fichero podemos estraer la siguiente información:

- El tiempo de vida de este registro es de tres días.
- El servidor primario es unanleon.edu.ni. y el correo del administrador es hostmaster.unanleon.edu.ni. Esta dirección es entendida por el programa ya que enviaría el correo al administrador de BIND 9.
- Fue creado el 25 de Junio del 2007 y es la 1 versión.
- Tiene un tiempo de refresco de 30 segundos. Tiempo de solicitud de retransmisión, 1 minuto, tiempo en el que expira el registro 4 minutos y tiempo del registro en cache 5 minutos.
- Los dos registros NS indican dos servidores (el primario y el secundario) en los cuales se encuentra registrada la zona. Los otros 2 registros MX indican con un número el servidor primario y secundario para transferencia de correo.

- Los demás son traducciones de nombres a direcciones IP. Por ejemplo, ftp (que equivale a ftp.unanleon.edu.ni) tiene la dirección 193.168.0.3.

Ahora crearemos el archivo de nuestra zona inversa. Este archivo estará dentro la carpeta pz. El contenido del archivo db.193.168.0 será:

```
;Archivo de datos de la zona automatica.com
$TTL 604800
@ IN SOA unanleon.edu.ni. hostmaster.unanleon.edu.ni (
      2007052501
      30
      1m
      4m
      5m )

      IN    NS    ns1.unanleon.edu.ni.
      IN    NS    ns2.unanleon.edu.ni.

1     IN    PTR   unan.leon.edu.ni.
1     IN    PTR   ns1.unan.leon.edu.ni.
2     IN    PTR   ns2.unan.leon.edu.ni.
3     IN    PTR   www.unanleon.edu.ni.
3     IN    PTR   ftp.unanleon.edu.ni.
4     IN    PTR   smtp.unanleon.edu.ni.
4     IN    PTR   pop3.unanleon.edu.ni.
```

Como podemos obsevar, el contenido de este fichero es bastante similar al del ejemplo anterior, solo que ahora las resoluciones se hacen a la inverse.

9. Verificando que nuestra zona funciona bien

Antes que todo, reiniciaremos nuestro servidor:

```
$ /etc/init.d/bind9 restart
```

Para comprobar que nuestro servidor DNS está resolviendo bien la zona que hemos creado podemos auxiliarnos de tres herramientas que con nslookup, dig y host. Las tres sirven para preguntar por la dirección IP de una zona. La diferencia es que unas brindan más información que otras pero cualquiera nos sirve para hacer las comprobaciones. La forma de usuarlas es:

```
$ dig www.unanleon.edu.ni
$ host ftp.unanleon.edu.ni
$ nslookup smtp.unanleon.edu.ni
```

Debemos de comprobar que las respuestas que nos envían son las que hemos configurado en el fichero de zonas. Para comprobar las resoluciones inversas, haremos:

```
$ dig –x 193.168.0.3
```

A lo cual deberá respondernos con el nombre de la máquina que para este caso serán: www.unanleon.edu.ni y ftp.unanleon.edu.ni.

Ahora, cada uno en su puesto deberá de configurar su servidor primario para una zona que el decida crearse y hacer consultas para ver si responde.

10. Configurando un servidor DNS Secundario

En el mundo de Internet, siempre es importante tener un servidor secundario ya que si uno cae, se sigue ofreciendo el servicio ya que toda la información esta respaldada, además, permite dividir la carga de trabajo ya que no solo es un servidor el que está respondiendo a las peticiones sino al menos dos.

Para poder instalar un servidor secundario necesitamos instalar BIND en otra máquina. Si lo que deseamos es crear un servidor secundario para la zona unanleon.edu.ni debemos recordar que ya en el servidor primario indicamos que el único servidor secundario es el de la IP 193.168.0.2, por tanto es en esa máquina en la que debemos de configurar el servidor secundario, además el fichero para esta zona, indica que el servidor secundario es esa IP la que se debe de usar.

El fichero named.conf del servidor secundario sería:

```
/*Archivo de configuracion del demonio DNS named configuración
de nuestro servidor DNS Secundario*/

#Declaracion de la zona unanleon.edu.ni
zone "unanleon.edu.ni" {
      type slave;
      file "sec.ul.ni";
      allow-query { any; };
      masters { 193.168.0.1; };
};

#Declaracion de la zona inversa unanleon.edu.ni
zone "0.168.193.in-addr.arpa" {
      type slave;
```

```
        file "sec.inv";
        masters { 193.168.0.1; };
};
```

Si analizamos, espracticamente igual (a nivel de zonas) que el del servidor primario. Las características principales de este archivo son:

- El tipo ahora es slave.
- El archivo se guarda en una ruta por defecto que es en /var/cahe/bind con lo que solamente hay que indicar el nombre que queremos darle. No hay que crear los archivos, solo hay que indicar con que nombre se crearán ya que al iniciar el servidor secundario, él los creará.
- Definimos quién es el maestro para cada zona que en este caso es la máquina 193.168.0.1.

11. Poniendo a prueba el servidor secundario

Para poner a prueba nuestro servidor secundario debemos, en la máquina que actuará como servidor secundario poner en el fichero /etc/resolv.conf lo mismo que hicimos cuando configuramos el servidor de cacheo.

Ahora hacemos una consulta preguntando (a nuestro servidor secundario siempre) por la IP de www.unanleon.edu.ni lógicamente no nos responderá. Arrancamos el servidor secundario y volvemos a consultar. Probablemente tampoco nos responda, pero si esperamos un poco más y volvemos a hacer la misma consulta si nos sabrá responder porque ya habrá descargado del servidor primario, toda la información de las zonas para las cuales es esclavo.

Crea un archivo de configuración llamado named.conf.sec y configura en él un servidor secundario para la zona que tu compañero es primario. Para arrancar el servidor con otro fichero solo debes de hacer:

```
$ named –f –c /etc/named.conf.sec
```

La opción –f permite que el demonio se ejecute en primer plano con lo que deberás abrir otra terminal para hacer las consultas. La opción –c permite indicar que fichero de configuración usará el demonio.

Para detener el demonio simplemente en la Terminal en que lo ejecutaste pulsa CTRL + C.

12. Las vistas en Bind9

Las vistas son una nueva características de Bind9 que permite mostrar a las máquinas internas una visión distinta de la jerarquía de nombres de DNS de la que se ve desde el exterior. Básicamente nos permite responder de una manera a las peticiones que vengan desde nuestra red interna y de otra forma a las peticiones que vengan desde la red externa.

Este tipo de configuración se esta haciendo cada vez más común, anteriormente esto se implementaba teniendo que montar dos servidores DNS. Gracias a la sentencia view podemos hcer esto en el mismo archivo de configuración.

La sentencia view empaqueta una lista de acceso que controla quien ve la vista, algunas opciones que se aplican a todas las zonas de la vista y, finalmente, las propias zonas. La sintaxis es:

```
View “nombre-de-la-vista” {
      match-clients { address_match:list; };
      opcion-de-vista;
      sentencia de zona;
};
```

La clausula controla quien puede ver la vista. Las vistas son procesadas en orden secuencial, por lo que las más restrictivas deben de ir primero. Las zonas en distintas vistas pueden tener el mismo nombre. Las vistas son una proposición de todo o nada; si las usas, todas las sentencias zone en tu fichero named.conf deben de aparecer dentro del contexto de una vista.

13. Un ejemplo empleando Vistas

Ahora lo que vamos a hacer es simular que nosotros internamente pertenecemos a la red 172.16.2.0 y tenemos replicas de los servidores web, ftp, smtp y pop3 dentro de esta red. La idea es que si alguna máquina que pertenece a esta red

pregunta por algunos de los subdominios de unanleon.edu.ni (www,ftp…) le contestaremos con una IP de la red 172.16.2.0, si por el contrario, el que pregunta por algunos de los subdominios, no pertenece a nuestra red interna, le contestaremos con la IP pública de esa máquina que esta en la red 193.168.0.0.

Para eso debemos de crear dos vistas, una interna a la cual pertenezca únicamente la red 172.26.2.0 y otra externa para cualquiera. Hay que aclarar que si se crea una vista es para dar respuesta distintas sobre al menos una zona, por tanto, se deberán de crear 2 ficheros de zonas para la zona de la cual se quiere dar una respuesta diferente dependiendo del caso. El fichero de configuración quedaría de la siguiente manera:

```
/*Archivo de configuración del demonio DNS named
Opciones de nuestro servidor DNS Primario empleando vistas*/

#Redes que componen la red interna de la Universidad
acl "lan" {
     172.16.2.0/16;
};
#Declaramos el o los servidores que actuaran como esclavos
acl "slaves" {
     193.168.0.1;
};

#Vista para la Red interna de la Universidad
view "internal" {
     match-clients { lan; }; #Cliente que utilizan esta vista
     recursion yes;

     #Declaracion de la zona de unanleon.edu.ni
     zone "unanleon.edu.ni" {
          type master;
          file "/home/alcala/DNS/unan-
zonas/db.unanleon.edu.ni.in";
          allow-query { any; };
     };

     #Declaracion de la zona inversa de la red interna
     zone "2.16.172.in-addr.arpa" {
          type master;
          file "/home/alcala/DNS/unan-zonas/pz/192.168.0";
     };

     #Declaracion de la zona inversa de la red pública
     zone "0.168.193.in-addr.arpa" {
          type master;
          file "/home/alcala/DNS/unan-zonas/pz/193.168.0";
     };
};
```

```
#Vista para la Red externa de la Universidad
view "external" {
     match-clients { any; };
     recursion no;

     zone "unanleon.edu.ni" {
          type master;
          file "/home/alcala/DNS/unan-
zonas/db.unanleon.edu.ni.ex";
          allow-query { any; };
          allow-transfer { slaves; };
     };

     zone "0.168.193.in-addr.arpa" {
          type master;
          file "/home/alcala/DNS/unan-zonas/pz/193.168.0";
          allow-transfer { slaves; };
     };
};
```

En este ejemplo podemos ver que solo es necesario crear dos ficheros para la zona unanleon.edu.ni ya que la zona inversa 0.168.193.in-addr.arpa es accesible desde cualquiera de las dos vistas por igual y la zona 2.16.172.in-addr.arpa solo es accesible desde la vista interna. Como dijimos antes, para la zona unanleon.edu.ni se crearon dos ficheros que son unanleon.edu.ni.in (vista interna) y unanleon.edu.ni.ex (vista externa) cuyos contenidos son:

```
;Archivo de datos de la zona interna de unanleon.edu.ni
$TTL 604800
@ IN SOA unanleon.edu.ni. hostmaster.unanleon.edu.ni. (
     2007052501
     30
     1m
     1m
     1m )
          NS   ns1.unanleon.edu.ni.
          NS   ns2.unanleon.edu.ni.
          MX   1 ns1.unanleon.edu.ni.
          MX   2 ns2.unanleon.edu.ni.

localhost  A    127.0.0.1
unanleon.edu.ni A    172.16.2.1
ns1        A    172.16.2.1
ns2        A    193.168.0.2
www        A    172.16.2.2
ftp        A    172.16.2.3
smtp       A    172.16.2.4
pop3       A    172.16.2.5
```

```
;Archivo de datos de la zona externa de unanleon.edu.ni
$TTL 604800
@ IN SOA unanleon.edu.ni. hostmaster.unanleon.edu.ni. (
      2007052501
      30
      1m
      1m
      1m )
            NS    ns1.unanleon.edu.ni.
            NS    ns2.unanleon.edu.ni.
            MX    1 ns1.unanleon.edu.ni.
            MX    2 ns2.unanleon.edu.ni.

localhost  A     127.0.0.1
unanleon.edu.ni A     193.168.0.1
ns1        A     193.168.0.1
ns2        A     193.168.0.2

www        A     193.168.0.3
ftp        A     193.168.0.3
smtp       A     193.168.0.4
pop3       A     193.168.0.4
```

Como se puede ver, aún mantenemos el servidor DNS esclavo funcionando pero solamente le transmitimos las zonas externas. Ahora bien, si queremos probar si nuestro servidor funciona, podemos probar cambiando nuestra IP y poniéndonos una que pertenezca a la red 172.16.2.0/16 y hacer consultas a nuestro servidor. Luego cambiar nuestra IP por cualquier otra, hacerle consultas nuevamente a nuestro servido y comparar que las respuestas son diferentes.

14. Notas adicionales

Supongamos ahora que deseamos montar el subdominio automatica.unanleon.edu.ni y que queremos que la resolución de nombres para este subdominio sea hecha desde otra máquina que es la 80.12.78.90 (supongamos). Lo que debemos de hacer es delegar la administración de este subdomnio a la máquina que se indica. En este caso, nosotros pasaríamos a actuar de esclavos de ese dominio y debemos indicar quien es el maestro. Para lograr esto, bastará con agregar al archivo named.conf:

```
zone "automatica.unanleon.edu.ni" {
      type slaves;
      file "db.automatica";
      master {
             80.12.78.90;
       };
};
```

En la máquina 80.12.78.90 deberá de configurarse un servidor DNS que contenga la zona automatica.unanleon.edu.ni y que contenga el fichero para esa zona.

15. Ejercicio Práctico

Piensa que eres el dueño de la empresa miempresa.ni (sustituye “miempresa” por un nombre único para tu empresa) ha decidido ampliarse y entrar en el mundo del Internet para lo cual a comprado una dirección ip pública (que en este caso será la IP de tu computadora) para poder instalar un servidor Web, un servidor de correo a través de la Web (webmail), un servidor imap y un servidor smtp, todos ellos se corresponderán con la misma dirección IP.

En esta ocasión se te pide que configures un servidor DNS que administre la zona anonimos.ni, dentro de la cual estarán los servicios www, smtp, imap y webmail. Para añadir mayor eficiencia deberás configurar el servidor empleando vistas para lo cual la vista externa será la red del laboratorio y para la vista interna añade una dirección virtual a la interfaz eth0 (puede ser la red que desees).

Para comprobar que los servidores de tus compañeros están en correcto funcionamiento, deberás configurar tu computadora para que haga las consultas al servidor DNS de otro de tus compañeros, acerca de sus empresas.

Recuerda que las consultas puedes hacerlas a través de dig, host y nslookup. Procura que el servidor funcione correctamente ya que te será de gran utilidad para las prácticas siguiente.

16. Referencias

- http://www.it.uc3m.es/aat/
- idefix.eup.uva.es/Manuales/dns/bind9.2.1.pdf
- http://http://dns.bdat.net/bind-9/DNS-HOWTO-9-es/dns.bdat.net/bind-9/DNS-HOWTO-9-es/

PRÁCTICA 6: CONFIGURACIÓN DE LAMP (LINUX + APACHE + MYSQL + PHP)

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción

La utilización conjunta de Linux, Apache, MySQL y PHP es conocida comúnmente como LAMP y permite la instalación de un servidor Web Dinámico y muy potente.

2. Objetivo

Los objetivos de esta práctica serán:

- Instalar un servidor Web Apache con PHP y Mysql, que permita montar las bases necesarias para la práctica siguiente en la cual se realizarán ejemplos de páginas Web dinámicas (PHP) con acceso a bases de datos (Mysql).
- Conocer las principales directivas y zonas utilizadas en los archivos de configuración de Apache.
- Configurar el servidor Web apache para que permita redirección de peticiones y páginas Web para los usuarios del sistema.
- Realizar autenticación y control de acceso a páginas Web o directorios concretos.
- Crear servidores virtuales que pueden estar basados en IP, puerto o nombre.

3. Referencias

- http://www.ubuntu-es.org/index.php?q=node/6304
- http://ciruelo.uninorte.edu.co/pdf/ingenieria_desarrollo/12/instalacion_y_configuracion_de_apache.pdf
- http://iagorubio.com/component/option,com_docman/task,cat_view/gid,35/Itemid,48/

PRÁCTICA 7: EJERCICIOS DE PÁGINAS WEB ESTÁTICAS Y DINÁMICAS

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción

Las páginas Web, como se observo en la teoría, pueden ser estáticas o dinámicas. En esta práctica se afianzarán los conceptos expuestos en la teoría a través de la realización de páginas web de ambos tipos. El objetivo no es que el alumno aprenda a fondo a relaizar páginas web sino que se familiarice con la estructura y semántica que sigue su diseño así como las ventajas y flexibilidad que ofrece el trabajar con PHP y Mysql a la hora de realizar páginas web con gran interactividad.

2. Objetivo

Los objetivos de esta práctica son:

- Realizar una página Web sencilla empleando HTML puro en la cual se presenten imágenes, enlaces, tablas.
- Diseñar una página web separando los datos del estilo a través del uso de XML y XSL.
- Crear una página Web que contenga un formulario y que permita introducir los datos de los usuarios y almacenarlos en una base de datos.

3. Referencias

- http://es.wikipedia.org/wiki/HTML
- http://www.desarrolloweb.com/manuales/21/
- http://bulma.net/body.phtml?nIdNoticia=628
- http://www.php-es.com/

PRÁCTICA 8: CONFIGURACIÓN DE UN SERVIDOR DE CORREO ELECTRÓNICO

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción

En esta práctica configuraremos un servidor de correo electrónico a través de la utilización de diferente software de libre distribución. El MTA que utilizaremos para enviar y recibir correo a otros servidores será Postfix. Una vez que el correo llegue a nuestro servidor srá depositado en nuestros buzones. La gestión de los buzones y el acceso a ellos lo haremos a través de IMAP.

El software que implementa este protocolo es Cyrus IMAP. Las cuentas de los usuarios las gestionaremos a través de Cyrus SASL. Para comunicar a IMAP con el MTA emplearemos LMTP. Para la creación de listas de distribución emplearemos Mailman y para poder acceder al correo a través de la red utilizaremos Webmail.

1.1 Postfix

El MTA (Mail Transportation Agent) Postfix pretende ser rápido, fácil de administrar y seguro. A diferencia de Sendmail (otro MTA), Postfix no es un programa monolítico, sino una combinación de pequeños programas, cada uno de los cuales lleva a cabo una función especializada. En nuestro caso, la idea básica es utilizar este MTA en conjunción con Cyrus IMAP de tal manera que sea Cyrus IMAP el que gestione las cuentas de los usuarios y tenga los buzones asociados a cada uno de ellos y Postfix sea nuestra MTA que envía y recibe el correo de nuestro servidor.

1.2 Cyrus IMAP

Cyrus IMAP (Internet Message Access Protocol) es desarrollado y mantenido por el Andrew Systems Group de la Carnegie Mellon University. A diferencia de otros servidores IMAP, Cyrus usa su propio método para almacenar el correo de los usuarios. Cada mensaje es almacenado en su propio fichero. El beneficio de usar ficheros separados es una mayor fiabilidad ya que sólo un mensaje se pierde en caso

de error del sistema de ficheros. Los metadatos, tales como el estado de un mensaje (leído, etc.) se almacenan en una base de datos. Además, los mensajes son indexados para mejorar el rendimiento de Cyrus, especialmente con muchos usuarios e ingentes cantidades de mensajes. No hay nada tan rápido como el servidor IMAP Cyrus.

Otra característica muy importante es que no son necesarias cuentas locales de Linux para cada usuario. Todos los usuarios son autenticados por el servidor IMAP. Esto lo convierte en una magnífica solución cuando se tiene una gran cantidad de usuarios.

La administración es llevada a cabo mediante comandos especiales de IMAP. Esto le permite usar tanto la interfaz de línea de comandos como los interfaces web. Desde la versión 2.1 de Cyrus, se usa la versión 2 de la librería SASL para la autenticación. En la configuración descrita en esta práctica, se implementa una autenticación de tres capas. Cyrus se autentica con saslauthdaemon, quien redirige la petición al mecanismo que le hayamos definido, por ejemplo sasldb, que buscará la información del usuario en una base de datos Berkeley DB.

1.3 Cyrus SASL

SASL son las siglas de Simple Authentication and Security Layer, un método para añadir soporte para la autenticación a protocolos basados en la conexión que ha sido estandarizado por la IETF (Internet Engineering Task Force). Se usa en servidores (en este caso Cyrus IMAP) para manejar las peticiones de autenticación de los clientes. Para ello, el protocolo incluye un comando para identificar y autenticar un usuario contra un servidor y para, opcionalmente, negociar la protección de las subsiguientes interacciones del protocolo. Si se negocia su uso, una capa de seguridad es añadida entre el protocolo y la conexión.

1.4 LMTP

SMTP (Simple Mail Transfer Protocol) y sus extensiones ESMTP (SMTP Service Extensions) proporcionan un mecanismo para transferir correo fiable y eficientemente. El diseño del protocolo SMTP requiere que el servidor maneje colas de envío de correo.

En ciertas circunstancias, fuera del área que engloba el intercambio entre hosts independientes en redes públicas, es deseable implementar un sistema donde el receptor del correo no maneje colas, como es el caso de un MDA (Mail Delivery Agent). Esto es precisamente lo que hace el protocolo LMTP (Local Mail Transfer Protocol).

Aunque LMTP es una alternativa al protocolo ESMTP, usa (con algunos pequeños cambios) la sintaxis y la semántica de ESMTP. Este diseño permite al LMTP utilizar las extensiones definidas para el ESMTP. LMTP no debería ser nunca usado en el puerto 25.

1.5 Mailman

Mailman es un software libre que permite gestionar listas de distribución, noticias y correo electrónicos. Mailman está integrado con la web, permitiendo a sus usuarios una fácil administración de sus cuentas, así como a sus propietarios administrar las listas. Mailman incluye soporte para crear archivos de correos, procesamiento automático de correo rechazado, filtrado de contenido, envío en modo compendio o resumen, filtros de spam, etc.

1.6 SquirrelMail

SquirrelMail es un paquete de correo por web basado en estándares y escrito en PHP4. Incorpora soporte PHP para los protocolos IMAP y SMTP, y todas sus páginas se crean en puro HTML 4.0, de modo que se garantice la máxima compatibilidad entre navegadores. Tiene muy pocos requerimientos y es muy fácil de instalar y configurar. SquirrelMail tiene toda la funcionalidad que se espera de un cliente de correo electrónico, incluyendo soporte de MIME, agendas de contactos y gestión de carpetas.

Avelsieve (SIEVE Mail Filters Plugin for SquirrelMail) es un plugin para SquirrelMail que permite crear scripts hechos con Sieve en un servidor Cyrus IMAP que tenga habilitado el soporte para dicho lenguaje (Tim's SIEVE daemon). Avelsieve es parte de Cyrusmaster, una herramienta de administración de Cyrus basada en web. Debería proporcionar una interfaz similar a la de los filtros de usuario a los administradores y personal de soporte técnico.

2. Objetivos

El objetivo es instalar y configurar el MTA Postfix y el servidor Cyrus IMA y conseguir un sistema de correo electrónico totalmente funcional y de alto rendimiento. Al final de la práctica conseguiremos tener un servidor de correo electrónico con:

- Independencia de los usuarios de sistema y las cuentas de correo electrónico.
- Un dominio principal donde se crean cuentas de correo.
- Acceso al correo a través de la Web.
- Listas de correo.

3. Instalación y configuración de SASL

La instalación de Cyrus es muy sencilla en Debian. Los paquetes necesarios para instalar Cyrus SASL son libsasl2, sasl2-bin y libsasl2-modules. Empezaremos por las librerías SASL y el servicio saslautd. Por lo tanto, como root:

```
$ apt-get install libsasl2 sasl2-bin libsasl2-modules
```

Editamos el fichero /etc/default/saslauthd y modificamos el parámetro MECHANISMS, de modo que el resultado final sea:

```
START=yes
MECHANISMS="sasldb"
```

Iniciamos el demonio saslauthd con el comando:

```
$ /etc/init.d/saslauthd start
```

Los scripts de instalación del paquete sasl2-bin crean una base de datos de usuarios vacía, en formato Berkeley Database, que se encuentra en /etc/sasldb2. Este archivo pertenece al usuario root y al grupo sasl, por lo que todos los usuarios que quieran acceder a ella sin pasar por el servicio saslauthd tendrán que pertenecer a ese grupo. En la actualidad este servicio sólo soporta el método de autenticación PLAIN.

El paquete sasl2-bin proporciona los ejecutables necesarios para la administración de la base de datos de usuarios, entre otros. Con el comando sasldblistusers2 podemos listar los usuarios existentes, mientras que con el comando

saslpasswd2 podemos añadir y borrar usuarios. Los parámetros más utilizados son -c y -d, para añadir y borrar, respectivamente. Por ejemplo:

```
$ saslpasswd2 -c denis
```

Crearía el usuario denis, tal y como podemos ver usando sasldblistusers2 (maquina es el hostname):

```
$ sasldblistusers2
denis@maquina: userPassword
```

Para eliminar el usuario ejecutaríamos el comando:

```
$ saslpasswd2 -d denis
```

La opción -u nos permite asociar usuarios a dominios. Por ejemplo:

```
$ saslpasswd2 -c denis -u espinozahernandez.ni
```

Añadiría el usuario denis@espinozahernandez.ni a la base de datos.

```
$ sasldblistusers2
denis@maquina: userPassword
denis@espinozahernandez.ni: userPassword
```

4. Instalación y configuración de Cyrus IMAP

El siguiente paso es instalar el servidor Cyrus IMAP Cyrus IMAP son cyrus21-admin, cyrus21-common, cyrus21-doc, cyrus21-imapd y cyrus-clients y para ello ejecutaremos el siguiente comando:

```
$ apt-get install cyrus21-admin cyrus21-common cyrus21-doc
cyrus21-imapd cyrus21-clients
```

El paquete cyrus21-clients, proporciona la herramienta imtest, la cuál nos será útil para comprobar el buen funcionamiento de nuestro servidor a medida que vayamos avanzando en la configuración y el uso de los diversos protocolos. Al instalar estos paquetes se crea una jerarquía de directorios en /var/spool/cyrus/mail que almacenará los buzones de correo. Todos estos buzones tendrán como propietario al usuario cyrus y al grupo mail. Existen únicamente dos ficheros de configuración para toda la plataforma Cyrus:

- **/etc/cyrus.conf:** Donde se determinan los servicios que se ejecutarán al arrancar el sistema Cyrus.
- **/etc/imapd.conf**: Donde se especifican los parámetros de configuración del servicio IMAP de Cyrus.

Nótese también que la instalación de la plataforma Cyrus nos obliga a instalar el paquete postfix, lo cual no es ningún inconveniente. Para una mayor información de las particularidades del sistema Cyrus, se recomienda leer, detenidamente y en varias ocasiones, la documentación disponible en /usr/share/doc/cyrus21-doc. En estos ficheros se encuentra la mayoría de la información necesaria para configurar tanto Cyrus IMAP como Postfix.. Los documentos más relevantes son README.Debian.gz, README.Debian.simpleinstall.gz y README.postfix.gz.

Antes de continuar es conveniente asegurarse de que el grupo sasl tiene permisos de lectura sobre la base de datos de usuarios, /etc/sasldb2, así como que el usuario cyrus pertenece a este grupo. Los scripts del paquete cyrus21-common automáticamente nos dejan una configuración así, pero mejor cerciorarse al principio que provocar un tedioso proceso de depuración cuando haya múltiples elementos en juego.

4.1 El fichero /etc/cyrus.conf

Este fichero de configuración consta de tres partes claramente diferenciadas:

- **START:** Esta sección lista los scripts que se ejecutarán antes de que se arranquen los servicios. Su uso más característico es inicializar las bases de datos y lanzar los servicios de larga ejecución.

- **SERVICES:** Esta sección es el corazón del fichero /etc/cyrus.conf, pues describe los procesos que deberán lanzarse para atender las conexiones que los clientes hagan a ciertos sockets, bien sean tipo TCP o UNIX.
- **EVENTS:** Esta sección lista los procesos que deberían ejecutarse a intervalos específicos, de modo similar a los trabajos del cron. Típicamente se usa para llevar a cabo tareas programadas de limpieza y mantenimiento.

Es suficiente con realizar un único cambio sobre la configuración estándar que nos deja el mantenedor del paquete de GNU/Debian Linux, y es comentar la línea

donde se declara la ejecución del servicio pop3. Por lo tanto, nos quedará un fichero de configuración tal que:

```
START {
      recover    cmd="/usr/sbin/ctl_cyrusdb -r"
      delprune   cmd="/usr/sbin/ctl_deliver -E 3"
      tlsprune   cmd="/usr/sbin/tls_prune"
}
SERVICES {
      imap    cmd="imapd    -U    30"    listen="imap"    prefork=0
maxchild=100
      lmtpunix cmd="lmtpd" listen="/var/run/cyrus/socket/lmtp"
                 prefork=0 maxchild=20
      sieve cmd="timsieved" listen="localhost:sieve" prefork=0
                 maxchild=100
      notify cmd="notifyd" listen="/var/run/cyrus/socket/notify"
                   proto="udp" prefork=1
}
EVENTS {
      checkpoint cmd="/usr/sbin/ctl_cyrusdb -c" period=30
      delprune   cmd="/usr/sbin/ctl_deliver -E 3" at=0401
      tlsprune   cmd="/usr/sbin/tls_prune" at=0401
}
```

En estos momentos entra en escena una cuestión importante en la configuración: usar sockets TCP o sockets UNIX. En la configuración que se acaba de presentar se ha optado por la segunda opción debido a que se considera que van a ejecutarse todos los servicios en la misma máquina. En un caso como éste, muy habitual, es mejor usar sockets UNIX debido, principalmente, al mejor rendimiento que ofrecen y a que simplifican la configuración en general. En cambio, en un contexto donde los servidores Postfix y Cyrus IMAP estén en máquinas diferentes, será necesario usar sockets TCP.

Tras cualquier cambio en el fichero /etc/cyrus.conf es preciso reiniciar el servidor, tarea que podemos llevar a cabo cómodamente con el comando /etc/init.d/cyrus21 restart.

4.2 El fichero /etc/imapd.conf

/etc/imapd.conf es el fichero de configuración del servidor Cyrus IMAP y en él se definen los parámetros locales para IMAP. Cada una de las líneas de tiene el formato *opción: valor*, donde opción es el nombre de la opción a configurar y valor el valor al cual se está estableciendo esa opción. Las líneas en blanco o que empiecen por # son ignoradas. A continuación se detallan algunas de las opciones más relevantes y sus valores recomendados:

- **altnamespace:** Esta opción viene por defecto con el valor no, forzando que las subcarpetas de usuario se creen debajo de inbox; si se cambia a yes, las subcarpetas del usuario se crearán a la misma altura que inbox. Esta opción está documentada en el fichero /usr/share/doc/cyrus21-doc/html/altnamespace.html. En este artículo se usará el valor por defecto a no.

- **lmtp_downcase_rcpt:** Esta opción, que sirve para forzar que el nombre de usuario se convierta a minúsculas, viene por defecto comentada, es decir, con valor no. Debido a que Cyrus diferencia mayúsculas y minúsculas, es una buena idea trabajar con los nombres de usuario siempre en minúsculas (el valor por defecto asume que el usuario es consciente de lo que está haciendo). En este artículo se usará el valor a yes (basta con descomentar la línea).

- **admins:** Esta opción permite definir los usuarios que tendrán permisos de administrador (flag a de la ACL de un buzón) sobre todos los buzones del sistema. El usuario cyrus, y únicamente él, es la opción más recomendable, por lo que bastará con descomentar la línea del fichero. Este usuario se va a autenticar mediante el método SASL, por lo que debe añadirse a la base de datos /etc/sasldb2 mediante el comando saslpasswd2, tal que $saslpasswd2 -c cyrus.

- **lmtp_admins:** Esta opción permite especificar una lista de usuarios, separados por espacios, que tendrán la categoría de administradores LMTP, es decir, que podrán enviar correo a través de LMTP por TCP/IP (además de aquellos definidos en la opción admin anterior). Si se van a usar sockets UNIX no es necesario descomentar esta opción, pues el usuario postman (valor por defecto) es autenticado automáticamente.

- **allowanonymouslogin:** Esta opción permite el acceso anónimo a los buzones en cuyas ACLs se haya añadido al usuario anonymous. Carece de sentido a menos que se quieran implementar grupos de noticias, por lo que se dejará su valor por defecto no.

- **umask:** Esta opción permite definir los permisos con los cuáles se guardarán los ficheros y subdirectorios dentro de /var/spool/cyrus/mail. Por defecto tiene el valor 077 (lectura y escritura para el propietario, nada para el resto), pero es conveniente permitir que el grupo (mail por defecto) tenga también permisos de lectura, pues de ese modo otras aplicaciones podrán leer el contenido de los emails, por ejemplo amavisd-new (bastará con que añadamos al usuario con el cuál se ejecutan a ese grupo).
- **allowplaintext:** Mediante esta opción decidimos si vamos a permitir uso del mecanismo de autentificación sasl plain. Es recomendable mantener el valor por defecto yes hasta que tengamos todo el sistema base configurado y funcionando. Más adelante en este artículo se explicará como usar otros métodos de autenticación más seguros.

- **sasl_mech_list:** Esta es la lista de los mecanismos de autenticación que se van a soportar. Es útil para evitar que se prueben todos los plugings existentes y para definir el orden de los mismos.

- **sasl_minimum_layer:** El SSF (del inglés, security strength factor) mínimo que el servidor permitirá negociar al cliente. Un valor igual a 1 requiere protección de integridad; un valor más algo requiere algún tipo de cifrado. Se recomienda empezar con un valor de 0 (valor por defecto), que permite login de texto plano, en primera instancia. Una vez esté todo funcionando será más sencillo aumentar la seguridad. Más adelante en este artículo se cambiará este valor.

- **sasl_maximum_layer:** Valor máximo del SSF que el servidor permitirá negociar al cliente. Es pertinente dejar el valor por defecto, 256 (no descomentar la línea).

- **sasl_auxprop_plugin:** Esta opción nos permite especificar los plugins del auxpropd que deseamos cargar, en el caso de estar usando

sasl_pwcheck_method: auxprop. Es necesario descomentar esta línea para que use sasldb.

- **lmtpsocket, idlesocket y notifysocket:** Estas tres opciones especifican las rutas para los tres sockets UNIX que utiliza Cyrus. Los valores por defecto son correctos. Cabe destacar que el valor de lmtpsocket (por defecto /var/run/cyrus/socket/lmtp) debe estar en consonancia con el especificado en el fichero /etc/cyrus.conf.

El fichero /etc/imapd.conf quedará como el que se presenta a continuación:

```
configdirectory: /var/lib/cyrus
defaultpartition: default
partition-default: /var/spool/cyrus/mail
partition-news: /var/spool/cyrus/news
newsspool: /var/spool/news
altnamespace: no
unixhierarchysep: no
lmtp_downcase_rcpt: yes
admins: cyrus
allowanonymouslogin: no
popminpoll: 1
autocreatequota: 0
umask: 027
sieveusehomedir: false
sievedir: /var/spool/sieve
hashimapspool: true
allowplaintext: yes
sasl_mech_list: PLAIN
sasl_minimum_layer: 0
sasl_pwcheck_method: saslauthd
sasl_auxprop_plugin: sasldb
sasl_auto_transition: no
tls_ca_path: /etc/ssl/certs
tls_session_timeout: 1440
tls_cipher_list:
 TLSv1:SSLv3:SSLv2:!NULL:!EXPORT:!DES:!LOW:@STRENGTH
lmtpsocket: /var/run/cyrus/socket/lmtp
idlesocket: /var/run/cyrus/socket/idle
notifysocket: /var/run/cyrus/socket/notify
```

Tras cualquier cambio en el fichero /etc/imapd.conf es necesario indicarle al servidor Cyrus que relea su contenido. Tenemos dos opciones para este cometido: /etc/init.d/cyrus21 restart o /etc/init.d/cyrus21 reload.

4.3 Conceptos de administración del servidor Cyrus IMAP

Podemos acceder a la interfaz de comandos para la gestión del servidor Cyrus IMAP mediante una llamada a cyradm, tal que:

```
$ cyradm --user cyrus localhost
Password:
localhost>
```

Pero antes, aclararemos ciertos conceptos:

Espacio de nombres de los buzones de correo.

El servidor Cyrus IMAP presenta los buzones de correo usando la convención netnews. Si bien en los nombres de los buzones no se hace distinción entre mayúsculas y minúsculas, se mantiene el formato en el cual se creó. Un nombre de buzón no puede comenzar o terminar con punto, ni tampoco puede contener dos puntos seguidos. Todos los buzones de correo del usuario denis comienzan con la cadena user.denis. El buzón user.denis es un caso especial. Desde el punto de vista del usuario denis, el buzón user.denis es lo que se conoce como su inbox o buzón de entrada. Si la lista de control de acceso del buzón de correo user.denis permite a otros usuarios ver dicho buzón, aparecerá a esos otros usuarios como user.denis. En este documento se hace referencia al buzón de correo user.denis como el inbox de denis.

El administrador crea y borra usuarios al crear y borrar los inbox de esos usuarios. Si un usuario tiene inbox, entonces puede suscribirse a los buzones de correo. Sólo los usuarios sin puntos en su identificador de usuario poseen la capacidad de tener un inbox (un usuario con un punto en su identificador podría hacer login pero no sería capaz de recibir correo). Cuando un administrador borra el inbox de un usuario, todos los buzones de correo personales de ese usuario se borran también.

Con la notable excepción del inbox, todos los nombres de buzones de correo pertenecen al sistema, independientemente del sistema. Son las listas de control de acceso las que determinan qué usuarios tienen qué permisos en qué buzones.

Listas de control de acceso

El acceso a cada buzón de correo se controla desde la lista de control de acceso de cada buzón. Las ACLs (del inglés, Access Control List), proporcionan un poderoso mecanismo para especificar los usuarios o grupos de usuarios que tienen permisos para acceder a los buzones de correo.

Una ACL es una lista de cero o más entradas. Cada entrada tiene un identificador y una serie de permisos. El identificador especifica el usuario o grupo de usuarios para el cuál la se aplica la entrada. El conjunto de permisos está formado por una o más letras o digítos, cada uno de ellos otorgando un privilegio en particular.

Los permisos definidos son:

- **l (lookup):** El usuario puede ver que el buzón de correo existe.
- **r (read):** El usuario puede leer el buzón de correo. El usuario puede seleccionar el buzón, leer los datos contenidos, llevar a cabo búsquedas y copiar mensajes de ese buzón.
- **s (seen):** Mantiene el estado leído por usuario. Se preservan los flags Seen y Recent para cada usuario.
- **w (write):** El usuario puede modificar los flags excepto Seen y Deleted (los cuales son controlados por otros permisos).
- **i (insert):** El usuario puede insertar mensajes en el buzón de correo.
- **p (post):** El usuario puede mandar correo a la dirección de envío del buzón. Este permiso difiere del permiso i en que el sistema de envío inserta información de seguimiento en los mensajes envíados;
- **c (create):** El usuario puede crear subcarpetas en el buzón;
- **d (delete):** El usuario puede alterar el flag Deleted, expirar correos y borrar o renombrar el buzón.
- **a (administer):** El usuario puede cambiar la ACL del buzón.

El identificador es la parte en una ACL que especifica el usuario o grupo para el cuál se aplica. El significado del identificador habitualmente depende del mecanismo de autorización que se use. Con cualquiera que sea el mecanismo de autorización,

existen siempre dos identificadores especiales. El identificador anonymous hace referencia al usuario anónimo, no autenticado.

El identificador anyone hace referencia a cualquier usuario, incluyendo al usuario anónimo. En la versión 2.1.16 existen únicamente dos métodos de autorización, Kerberos y grupos UNIX (/etc/group). Para una cantidad pequeña de usuarios y grupos, el uso del fichero de grupos de UNIX es una opción adecuada, aún haciendo referencia a cuentas de usuario UNIX que no existen.

Independientemente de la ACL de un buzón, los usuarios administradores (es decir, los que aparecen en la opción admins del fichero /etc/imapd.conf) siempre tienen los permisos l y a sobre todos los buzones de manera implícita. Cuando se crea un buzón de correo, su ACL inicial es una copia de la ACL de su buzón padre. Cuando se crea un usuario, la ACL del inbox de ese usuario contiene una única entrada con todos los permisos garantizados para el propietario. Cuando se crea un buzón de correo no asociado a un usuario que no tiene padre, su ACL se inicializa a valor de la opción defaultacl del fichero /etc/imapd.conf.

Los administradores globales se especifican con un identificador de usuario no cualificado en la opción admins y tienen acceso sobre cualquier buzón del servidor. Debido a que los administradores globales usan identificadores de usuario no cualificados, pertenecen al dominio por defecto, definido en la variable defaultdomain del mismo fichero /etc/imapd.conf. Como resultado de esto, no pueden tenerse administradores globales sin especificar el defaultdomain. Nótese que, al tratar de hacer login como administrador global en un servidor multi-home desde una máquina remota, posiblemente sea necesario cualificar completamente el identificador de usuario con el defaultdomain. Los administradores globales deben usar una sintaxis tipo buzon@dominio.tld al especificar buzones fuera del defaultdomain.

Ejemplos al usar cyradm:

Para crear un inbox para el usuario denis en el defaultdomain:

```
$ cm user.denis
```

Para crear un inbox para el usuario denis en el dominio espinozahernandez.ni:

```
$ cm user.denis@espinozahernandez.ni
```

Para listar todos los buzones del dominio espinozahernandez.ni:

```
$ lm *@espinozahernandez.ni
```

Cuotas de usuario

El servidor Cyrus IMAP soporta cuotas de almacenamiento, lo que se define como el número de bytes de los mensajes RFC-822, en kilobytes. Cada mensaje se cuenta independientemente, incluso si el servidor es capaz de ahorrar espacio con el uso de enlaces duros a los ficheros de mensajes. El espacio adicional requerido para almacenar el índice del buzón y la caché de los ficheros no se tiene en cuenta para calcular la cuota.

Las cuotas se aplican a las cuotas raíz, que pueden estar en cualquier nivel de la jerarquía de buzones. Las cuotas raíz no es necesario que sean buzones. Las cuotas en una cuota raíz se aplican a la suma del uso de todo buzón que se encuentre al mismo nivel o por debajo en la jerarquía y que no esté bajo otra cuota raíz en la subjerarquía. Esto significa que cada buzón se ve limitado por una cuota raíz como máximo.

Por ejemplo, dados los buzones:

```
user.denis
user.denis.lists.universidad
user.denis.lists.postfix-users
user.denis.todo
user.denis.work
user.denis.work.customers
user.denis.work.suppliers
```

Y cuotas raíz en:

```
user.denis
user.denis.lists
user.denis.work
```

Entonces, la cuota raíz user.jsabater se aplicaría a los buzones user.denis y user.denis.todo, la cuota raíz user.denis.lists tendría efecto sobre los buzones user.denis.lists, user.denis.lists.universidad y user.denis.lists.postfix-users y la cuota raíz user.denis.work se aplicaría sobre los buzones user.denis.work, user.denis.work.customers y ususer.denis.suppliers.

Las cuotas raíz se crean mediante el comando setquota de la utilidad de consola cyradm. Las cuotas raíz no se pueden eliminar mediante comandos, sino que deben borrarse los ficheros pertinentes, tal y como se explica más adelante en este documento.

Normalmente, para que un mensaje pueda insertarse en un buzón, la cuota raíz de ese buzón debe tener suficiente espacio sin usar para que la inserción de dicho mensaje no provoque que la cuota se sobrepase. Pero el envío de correo es un caso especial. Para que un mensaje pueda ser entregado en un buzón, el uso de la cuota raíz del buzón no debe superior a lo permitido. Si el uso no sobrepasa el límite, entonces un mensaje puede ser entregado independientemente de su tamaño.

Esto provoca que el buzón sobrepase la cuota, causando que el usuario sea informado de dicho problema y dándole la oportunidad de que lo arregle. Si no se permitiera la entrega en estos casos, el usuario no tendría manera práctica de saber que hubo correo que no pudo ser entregado. Si el uso está por encima del límite, entonces el envío del correo fallará, retornando un error temporal.

Esto hará que el sistema de envío reintente la entrega durante un par de días (permitiendo al usuario darse cuenta del problema y corregirlo) y luego devolverá el correo al remitente.

Logging

El subdirectorio log bajo el directorio de configuración permite a los administradores mantener logs por usuario. Si existe un subdirectorio de log con el nombre de un usuario, entonces los servidores IMAP y POP3 mantendrán un log de las sesiones que se han autenticado como ese usuario. El log de telemetría se guarda en el subdirectorio con el nombre de fichero del identificador de proceso del servidor y empieza con el primer comando tras la autenticación. El servidor IMAP Cyrus envía mensajes de log al syslog local. Los niveles de gravedad usados son:

- **CRIT:** Errores críticos que probablemente requieren acciones administrativas en la línea de comandos.
- **ERR:** Errores de entrada y salida (E/S), incluyendo fallos al actualizar el uso de la cuota. El mensaje al syslog incluyen el error Unix y el fichero específicos.

- **WARNING:** Fallos de mecanismos de protección, timeouts por inactividad del cliente.
- **NOTICE:** autenticaciones, correctas o incorrectas.
- **INFO:** aperturas de buzones de correo.

El directorio proc

El subdirectorio proc dentro del directorio de configuración contiene un fichero por cada proceso activo del servidor. El nombre de fichero es la representación ASCII del identificador de proceso y cada fichero contiene los siguientes campos, separados por tabuladores:

nombre del host del cliente;
login del usuario, si está conectado;
buzón seleccionado, si hay alguno.

Normalmente el subdirectorio proc es purgado al reiniciar el servidor.

4.4 Configuración de buzones de correo

La administración de los buzones de correo se realiza mediante el programa cyradm. Deberá usarse uno de los administradores definidos en el fichero /etc/imapd.conf para conectarse, tal que /usr/bin/cyradm --user cyrus localhost. El par usuario/contraseña será el definido en la base de datos de usuarios /etc/sasldb2 mediante el programa sasldbpasswd2, tal y como se explica en el punto 4 de este artículo.

Se recomienda usar el usuario cyrus como administrador, por lo que procederemos a darlo de alta:

```
saslpasswd2 -c cyrus
```

En estos momentos, nuestra base de datos de usuarios contiene al usuario jsabater y al usuario cyrus:

```
$sasldblistusers2
```

```
denis@maquina: userPassword
cyrus@maquina: userPassword
denis@espinozahernandez.ni: userPassword
```

Ahora ya se puede acceder a la administración de los buzones de Cyrus, mediante /usr/bin/cyradm --user cyrus localhost. Una vez dentro, el comando help nos mostrará una descripción de los comandos disponibles y sus alias. De entre todos, los de uso más frecuente se comentan acto seguido:

Comando	Alias	Función	Sintaxis	Ejemplos
Createmailbox	cm	Crear buzones de correo	cm <buzon>	cm user.denis
Deletemailbox	dm	Borrar buzones de correo	dm <buzon>	dm user.denis
Listacl	lam	Listar las ACL de un buzón	lam <buzon>	lam user.denis
Setacl	sam	Establecer las ACL en un buzón	sam <buzon> <usuario> <permisos>	sam user.denis denis all
Deleteacl	dam	Borrar las ACL de un buzón	dam <buzon> <usuario>	dam user.denis denis

La palabra clave all agrupa todos los permisos. Para asignar permisos a un grupo UNIX se usa la forma group:nombre_grupo. Un grupo UNIX debe existir en el fichero /etc/group. Los usuarios que añadamos a ese grupo (editando manualmente el fichero) serán buzones de correo creados con el programa cyradm, no usuarios de sistema (aunque puede existir correspondencia, pero no tendrá relación alguna).

Pese a que el usuario cyrus tiene permisos implícitos de administración sobre todos los buzones, es preciso hacerlos explícitos para ejecutar algunos comandos sobre ellos (por ejemplo, para borrarlos). Un comando del estilo sam user.buzon cyrus all solucionaría el problema, pues luego se heredarían los permisos en los subbuzones.
En este punto, podemos crear un buzón10 cualquiera con propósitos de prueba, tal que:

```
$ cyradm --user cyrus localhost
Password:
localhost> cm user.denis
```

```
sam user.buzon cyrus all
localhost> lam user.denis
denis lrswipcda
cyrus lrswipcda
```

En este momento deberíamos ser capaces de dar de alta una cuenta de correo usando el protocolo IMAP en nuestro cliente de correo favorito y conectarnos al servidor, usando el usuario definido con saslpasswd2, al buzón del mismo nombre creado con cyradm. Asimismo, también podemos usar la herramienta imtest (/usr/bin/imtest), incluida en el paquete cyrus21-clients para comprobar su correcto funcionamiento.

```
$ imtest -a jsabater -w <contraseña> -m login localhost
S: * OK genma Cyrus IMAP4 v2.1.16-IPv6-Debian-2.1.16-6 server
10ready
C: C01 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4rev1 ACL QUOTA LITERAL+ MAILBOX-
REFERRALS NAMESPACE UIDPLUS ID NO_ATOMIC_RENAME UNSELECT
CHILDREN MULTIAPPEND SORT THREAD=ORDEREDSUBJECT
THREAD=REFERENCES IDLE AUTH=NTLM AUTH=DIGEST-MD5 AUTH=CRAM-MD5
LISTEXT LIST-SUBSCRIBED ANNOTATEMORE
S: C01 OK Completed
C: L01 LOGIN jsabater {8}
S: + go ahead
C: <omitted>
S: L01 OK User logged in
Authenticated.
Security strength factor: 0
```

Pulsando Ctrl+C abandonaremos el programa de pruebas:

```
C: Q01 LOGOUT
Connection closed.
```

5. Instalación y configuración de Postfix.

Es muy posible que tengamos ya instalado el paquete postfix, pues el paquete cyrus21-common depende de él o de mail-transport-agent. En cualquier caso, ejecutaremos el siguiente comando como root:

```
$ apt-get install postfix postfix-tls postfix-doc postfix-
pcre mime-codecs
```

Esto nos dejará instalados en el sistema todo lo necesario para la configuración posterior de Postfix. En el caso de que no tuviésemos el paquete postfix anteriormente, el script de postinstalación de este paquete nos mostrará unas pantallas que permiten configurar de un modo básico el servidor. Si ya estaba instalado el paquete, entonces deberemos llamar a ese asistente de modo manual mediante el comando:

```
$ dpkg-reconfigure postfix
```

No es imprescindible usarlo, pues en el artículo se incluyen los ficheros completos de configuración, pero es un buen punto de partida. Estas son las respuestas usadas para el servidor de este artículo:

General type of configuration? Internet Site
Where should mail for root go? denis
Mail name? espinozahernandez.ni
Append .domain to simple addresses? No
Other destinations to accept mail for? espinozahernandez.ni, maquina.espinozahernandez.ni,localhost.espinozahernandez.ni, localhost
Force synchronous updates on mail queue? No
Local networks? 127.0.0.0/8
Use procmail for local delivery? No
Mailbox size limit? 0
Local address extension character? +

5.1 El fichero /etc/postfix/master.cf

Usaremos el protocolo LMTP para la comunicación entre el MTA Postfix y Cyrus, pues es condición sinequa non para aprovechar las ventajas de Sieve. No es posible usar el transporte cyrus ni tampoco procmail y Sieve a la vez y, además, LMTP ofrece un rendimiento muy superior al uso directo de cyrdeliver para entregar el correo. Usaremos el socket /var/run/cyrus/socket/lmtp, por lo que debemos asegurarnos de que el servicio lmtpunix está habilitado en el fichero /etc/cyrus.conf y que Postfix tiene acceso a ese fichero (un socket, a efectos de permisos, funciona igual que un fichero). Asimismo, Cyrus requiere que las entregas por LMTP estén autenticadas, y asume que las que se hagan a través del socket Unix son de confianza y las preautentica

como si vinieran del usuario postman (ficticio). Por lo tanto, nos aseguraremos de que el fichero /etc/postfix/master.cf contenga esta línea:

```
# service type  private unpriv  chroot  wakeup  maxproc command + args
#               (yes)   (yes)   (yes)   (never) (100)
# ==========================================================================
   lmtp    unix    -       -        n        -        -       lmtp
```

Nótese que el programa no se ejecuta en un chroot. En el caso de querer ejecutarlo en una jaula, el socket /var/run/cyrus/socket/lmtp debería ser accesible desde dentro de la jaula, bien creando un enlace duro (hard link), bien modificando la ruta en el fichero de configuración /etc/cyrus.conf.

5.2 El fichero /etc/postfix/main.cf

Para conseguir que Postfix entregue los correos a Cyrus a través de LMTP deberemos configurar un transporte en el primero. No es aconsejable usar cyrdeliver. La configuración del transporte en Postfix puede hacerse de diversas maneras (default_transport, transport_maps o mailbox_transport). En este artículo se usará mailbox_transport. Debido a que Postfix 2.x no pasa a minúsculas los destinatarios en las entregas por LMTP, es aconsejable usar la opción lmtp_downcase_rcpt: yes en el fichero /etc/imapd.conf. Para el uso de sockets Unix, el transporte de Postfix se especifica como lmtp:unix:/var/run/cyrus/socket/lmtp (en este ejemplo se usa la localización por defecto del socket de Cyrus en Debian, que se define en /etc/cyrus.conf).

Se necesita también un servicio lmtpd de Cyrus escuchando en ese socket, luego es conveniente asegurarse de que exista una línea como la siguiente en la sección SERVICES del fichero /etc/cyrus.conf:

```
lmtpunix   cmd="lmtpd"       listen="/var/run/cyrus/socket/lmtp"
prefork=0 maxchild=20
```

Asimismo, es imprescindible asegurarse de que tanto Cyrus como Postfix pueden hablarse a través de ese socket. Los sockets Unix funcionan igual que los ficheros, por lo que esto se traduce en que tanto el usuario cyrus como el usuario postfix pueden leer y escribir en ese fichero. Aviso: debido a que Cyrus preautentica

cualquier cosa que proceda del socket Unix, cualquiera que pueda escribir en él será capaz de inyectar correo directamente en Cyrus.

Úsese dpkg-statoverride para asegurarse que la configuración de los permisos del socket no es sobreescrita por los paquetes de Cyrus. Recuerde que Postfix ejecuta el transporte LMTP con el usuario definido en /etc/postfix/master.cf, por defecto postfix) y que si se quiere ejecutar dicho transporte en una jaula, el socket deberá encontrarse dentro de esa jaula. Para ello:

Cree un grupo llamado lmtp:

```
$ addgroup lmtp
```

Agregue el usuario postfix a ese grupo:

```
$ adduser postfix lmtp
```

Corrija los permisos del directorio del socket

```
$ dpkg-statoverride --force --update --add cyrus lmtp 750
  /var/run/cyrus/socket
```

Reinicie Postfix y Cyrus:

```
$ /etc/init.d/postfix restart
$ /etc/init.d/cyrus21 restart
```

Las modificaciones mínimas necesarias en Postfix nos dejan un /etc/postfix/main.cf como éste:

```
setgid_group = postdrop
smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no
append_dot_mydomain = no
delay_warning_time = 4h
command_directory = /usr/sbin
daemon_directory = /usr/lib/postfix
program_directory = /usr/lib/postfix
myhostname = webmail.linuxsilo.net
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydestination = $myhostname, $mydomain, localhost.$mydomain, localhost
myorigin = $mydomain
mynetworks = 127.0.0.0/8
mailbox_size_limit = 0
recipient_delimiter = +
local_recipient_maps =        mailbox_transport                                =
lmtp:unix:/var/run/cyrus/socket/lmtp
```

Donde /etc/mailname es:

```
espinozahernandez.ni
```

Donde /etc/hostname es:

```
maquina
```

Donde /etc/hosts contiene:

```
127.0.0.1           localhost    localhost.localdomain
172.29.21.15        maquina maquina.espinozahernandez.ni
                    webmail.espinozahernandez.ni
```

Y donde /etc/resolv.conf contiene:

```
search espinozahernandez.ni
nameserver 172.29.21.15
```

Nótese que la directiva local_recipient_maps = evita que Postfix busque el nombre de usuario y su contraseña en los usuarios de sistema, pues su valor por defecto es proxy:unix:passwd.byname $alias_maps.

Para comprobar el correcto funcionamiento de lo que tenemos hecho hasta el momento, primero nos aseguramos de que los servicios estén escuchando en los puertos correspondientes:

```
$ netstat -an|grep LISTEN
tcp    0    0   0.0.0.0:143     0.0.0.0:*     LISTEN
tcp    0    0   127.0.0.1:2000  0.0.0.0:*     LISTEN
tcp    0    0 0.0.0.0:25        0.0.0.0:*     LISTEN
```

El puerto 25 es de smtp, el 143 es de imap y el 2000 de sieve. Las correspondencias entre servicios y puertos pueden consultarse en el fichero /etc/services. Acto seguido podemos mandar un correo electrónico a un buzón local desde de la propia máquina local o desde una exterior (en negrita los comandos que tecleamos nosotros, genma es el hostname):

$ telnet localhost 25

Trying 127.0.0.1...

Connected to localhost.

Escape character is '^]'.

220 webmail.espinozahernandez.ni ESMTP Postfix (Debian/GNU)

HELO localhost

250 webmail.espinozahernandez.ni

MAIL FROM: <denis@espinozahernandez.ni>

250 Ok

RCPT TO: <denis@espinozahernandez.ni>

250 Ok

DATA

354 End data with <CR><LF>.<CR><LF>

Este es un mensaje de pruebas enviado a traves de telnet.

.

250 Ok: queued as 5F496BEE9

QUIT

221 Bye

Connection closed by foreign host.

En estos momentos podemos ver que el correo ha sido enviado al buzón denis@espinozahernandez.ni ejecutando el comando:

```
$ ls /var/spool/cyrus/mail/d/user/denis/
1.    cyrus.cache          cyrus.header    cyrus.index
```

A continuación vamos a activar el uso de SASL en Postfix. El objetivo es autenticar los clientes smtp para que puedan hacer relay a través del servidor de correo. Para ello se modifica la opción smtpd_recipient_restrictions del fichero /etc/postfix/main.cf:

```
smtp_sasl_auth_enable = no
smtpd_sasl_auth_enable = yes
smtpd_sasl_local_domain = genma
smtpd_recipient_restrictions =
        permit_mynetworks,
        permit_sasl_authenticated,
        reject_unauth_destination
smtpd_sasl_security_options = noanonymous
```

De este modo, se permite hacer relay a los clientes sin autenticar que pertenezcan a las redes indicadas en mynetworks (habitualmente la red local) y a los clientes autenticados mediante el método SASL. Se rechazarán los que están sin autenticar. Opcionalmente, mediante la opción smtpd_sasl_local_domain=maquina se indica que compruebe el usuario entrando en /etc/sasldb2 con usuario@maquina.

Para que Postfix sepa qué mecanismo usar a través de SASL hay que crear el fichero /etc/postfix/sasl/smtpd.conf con las siguientes líneas:

```
pwcheck_method: saslauthd
mech_list: plain login
```

Debido a que Postfix se ejecuta por defecto en una jaula (chrooted), localizada en /var/spool/postfix, no puede acceder al socket y demás ficheros del demonio de autenticación SASL, que se encuentran en /var/run/saslauthd. Por lo tanto, Postfix va a tratar de encontrarlos en /var/spool/postfix/var/run/saslauthd/. Para ello, usaremos enlaces duros. Siempre que el fichero original y el enlazado estén en la misma partición, podemos crear un hard link entre /var/run/saslauthd y /var/spool/postfix/var/run/saslauthd.

Pero, debido a que estos ficheros son sobreescritos cada vez que se inicia el demonio saslauthd, debemos asegurarnos de que se vuelven a enlazar. Para ello, modificaremos el script de arranque en /etc/init.d/saslauthd y lo dejaremos como el que sigue (en negrita los cambios realizados):

```
     #!/bin/sh –e
     NAME=saslauthd
     DAEMON="/usr/sbin/${NAME}"
     DESC="SASL Authentication Daemon"
     DEFAULTS=/etc/default/saslauthd
     PWDIR=/var/run/saslauthd
     PIDFILE="/var/run/${NAME}/saslauthd.pid"

     mklinks(){
     sleep 1
          cd /var/spool/postfix/var/run/saslauthd/
          ln /var/run/saslauthd/* .
          echo "Links inside the Postfix jail have been
created."
     }

     rmlinks() {
          rm -f /var/spool/postfix/var/run/saslauthd/*
          echo "Links inside the Postfix jail have been
removed."
     }

     createdir() {
     # $1 = user
     # $2 = group
     # $3 = permissions (octal)
     # $4 = path to directory
```

```
            [ -d "$4" ] || mkdir -p "$4"
            chown -c -h "$1:$2" "$4"
            chmod -c "$3" "$4"
      }

      test -f "${DAEMON}" || exit 0

      # Source defaults file; edit that file to configure this
       # script.
      if [ -e "${DEFAULTS}" ]; then
      . "${DEFAULTS}"
      fi

      # If we're not to start the daemon, simply exit
      if [ "${START}" != "yes" ]; then
            exit 0
      fi

      # If we have no mechanisms defined
      if [ "x${MECHANISMS}" = "x" ]; then
            echo "You need to configure ${DEFAULTS} with
mechanisms
                  to be used"
            exit 0
      fi

      # Add our mechanimsms with the necessary flag
      PARAMS="${PARAMS} -a ${MECHANISMS}"

      START="--start --quiet --pidfile ${PIDFILE} --startas
${DAEMON} --name ${NAME} -- ${PARAMS}"

      # Consider our options
      case "${1}" in
            start)
                  echo -n "Starting ${DESC}: "
                  dir=`dpkg-statoverride --list $PWDIR`
                  test -z "$dir" || createdir $dir
                  if start-stop-daemon ${START} >/dev/null 2>&1;
then
                        echo "${NAME}."
                  else
                  if start-stop-daemon --test ${START} >/dev/null
                      2>&1; then
                        echo "(failed)."
                        exit 1
                  else
                        echo "${DAEMON} already running."
                        exit 0
                  fi
            fi
            mklinks
            ;;

            stop)
                  echo -n "Stopping ${DESC}: "
```

```
                if start-stop-daemon --stop --quiet --pidfile
                    "${PIDFILE}" \--startas ${DAEMON} --retry
10 –
                    name ${NAME} \ >/dev/null 2>&1 ; then
                      echo "${NAME}."
                else
                if start-stop-daemon --test ${START} >/dev/null
                    2>&1; then
                      echo "(not running)."
                      exit 0
                else
                      echo "(failed)."
                      exit 1
                fi
                fi
          rmlinks
          ;;
          restart|force-reload)
          $0 stop
          exec $0 start
          ;;
          *)
          echo "Usage: /etc/init.d/${NAME} {start|stop|restart
                |force-reload}" >&2
          exit 1
          ;;
          esac
exit 0
```

De este modo los enlaces se alterarán adecuadamente cada vez que se pare, inicie o reinicie el demonio. Aunque antes es preciso crear el directorio donde se harán los enlaces en el interior de la jaula de Postfix:

```
$ mkdir -p /var/spool/postfix/var/run/saslauthd
$ chown root.sasl /var/spool/postfix/var/run/saslauthd
```

Tras reiniciar el servidor de autenticación (/etc/init.d/saslauthd restart) y el de correo (/etc/init.d/postfix restart) ya en funcionamiento de la autenticación SASL.

6. Listas de correo con Mailman.

Mailman tiene dependencia de apache2 por lo cual, este y otros paquetes se instalaran en el momento en que ejecutemos apt-get install. Antes de ejecutar el comando también conviene asegurarse de que tenemos un alias para root en el /etc/aliases apuntando a un usuario que tenga buzón de correo (p.e. denis).

Tras la instalación del paquete, es conveniente leer la documentación disponible en /usr/share/doc/mailman, principalmente en `/usr/share/doc/mailman/README.Debian.gz` y en `/usr/share/doc/mailman/README.POSTFIX.gz`. En ella se detallan las modificaciones necesarias en Apache para el correcto funcionamiento de la interfaz web, así como la forma de integrar Postfix y Mailman. Acto seguido se va a configurar mailman, integrándolo con Postfix, para el dominio local. Primero procederemos a instalar el paquete Debian de Mailman mediante el comando:

```
$ apt-get install mailman
```

En el diálogo de configuración que nos presenta el script de postinstalación elegimos los idiomas que vamos a utilizar en Mailman y, de entre ellos, el que será elegido por defecto. A continuación debemos modificar ligeramente el fichero /etc/mailman/mm_cfg.py a fin de que Mailman sepa que se está trabajando con Postfix como Mail Transport Agent:

```
MTA = 'Postfix'
```

La configuración por defecto de Postfix nos deja la directiva alias_maps apuntando a /etc/aliases. Ya que no nos interesa estar modificando este fichero y ejecutando el comando newaliases de Postfix cada vez que creemos o borremos una lista, utilizaremos el fichero de alias propio de Mailman, que es automáticamente actualizado por los comandos newlist y rmlist. El primer paso será generarlo:

```
$ cd /var/lib/mailman
$ bin/genaliases
```

A continuación añadiremos ese fichero de alias a la directiva alias_maps del /etc/postfix/main.cf, además de otras directivas necesarias, tal que:

```
alias_maps = hash:/etc/aliases,
hash:/var/lib/mailman/data/aliases
mailman_destination_recipient_limit = 1
unknown_local_recipient_reject_code = 550
owner_request_special = no
recipient_delimiter = +
```

Y solicitaremos a Postfix que recargue la configuración:

```
$ /etc/init.d/postfix reload
```

El tercer paso de la instalación de Mailman nos avisa de que es necesario crear una site list llamada mailman y que hasta que no la creemos el demonio del Mailman no arrancará. Ahora es el momento de crearla y, para ello, ejecutamos el siguiente comando:

```
$ newlist mailman
Enter the email of the person running the list:
denis@espinozahernandez.ni
Initial mailman password:
Hit enter to notify mailman owner...
```

Nótese que Mailman no nos muestra la lista de alias que nos requiere que añadamos a nuestro fichero de alias. Esto es debido a la configuración realizada más arriba, eliminándose de esta manera este tedioso paso. Podemos, por lo tanto, pulsar enter y pasar a iniciar el demonio de Mailman mediante el comando /etc/init.d/mailman start. Una vez iniciado el servicio, recibiremos el correo que nos notifica la creación de la lista en la dirección de correo que hayamos especificado (denis@espinozahernandez.ni en este ejemplo).

Las listas que creemos en el futuro tampoco nos solicitarán que añadamos manualmente la lista de alias. Cuando se añada o quite una lista, el fichero /var/lib/mailman/data/aliases.db será automáticamente actualizado, pero no se ejecutará automáticamente un /etc/init.d/postfix reload. Esto es debido a que es necesario ser root para ejecutar este comando y los scripts suid-root no son seguros. El único efecto de esto es que le llevará aproximadamente un minuto a Postfix darse cuenta de los cambios y actualizar sus tablas, si bien considero esto una inconveniencia menor.

Finalmente, para poder acceder a la interfaz web de Mailman, deberemos llevar a cabo unas simples modificaciones en Apache, versión 2.0.50 en este artículo:
Activar el módulo cgi, por ejemplo mediante el comando. Este paso es obligatorio.

```
$2enmod cgi
```

Añadir algunos alias que nos permitan acceder mediante URLs más cortas. En el default virtual host o en el del dominio que queramos usar para acceder a la interfaz, por ejemplo lists.espinozahernandez.ni, añadiremos las siguientes directivas:

```
ScriptAlias /mailman/ /usr/lib/cgi-bin/mailman/
Alias /pipermail/ /var/lib/mailman/archives/public/
Alias /images/mailman/ /usr/share/images/mailman/
```

Este paso es opcional. Una manera más sencilla de realizar estas modificaciones y que consigue que afecten a todos los hosts es crear un fichero llamado mailman dentro del directorio /etc/apache2/conf.d/ que contenga esas tres líneas.

Ahora reiniciaremos apache con:

```
$ apache2ctl graceful
```

Ahora podemos acceder a la interfaz web de Mailman a través de la URL http://webmail.espinozahernandez.ni/mailman/listinfo (o con el dominio que hayamos definido).

7. Correo a través de web con SquirrelMail.

De nuevo, en Debian GNU/Linux la instalación del software es tan sencilla como ejecutar un simple comando como root:

```
$ apt-get install squirrelmail
```

Tal y como se nos indica al final de la instalación, para configurar SquirrelMail disponemos de la herramienta /usr/sbin/squirrelmail-configure. De las opciones que nos presenta a lo largo y ancho de los menús, estas son las que personalmente he considerado que eran merecedoras de modificación:

```
$ squirrelmail-configure
Organization Preferences: Organization Name:
espinozahernandez.ni
Message of the Day (MOTD): Edit the MOTD: Bienvenido al Correo
de espinozahernandez.ni
```

Grabamos los cambios con la opción Save data (S) y ya tan sólo nos queda modificar la configuración de Apache 2 para tener acceso a la interfaz web. Para ello, el camino más corto y sencillo es, tal y como se documenta en /usr/share/doc/squirrelmail/README.Debian, modificar el fichero de configuración de Apache para SquirrelMail, que hallaremos en /etc/squirrelmail/apache.conf, adaptándolo a nuestras necesidades. Finalmente, tan sólo deberemos crear un enlace débil en /etc/apache2/conf.d/, tal que:

```
$ln –s /etc/squirrelmail/apache.conf \
/etc/apache2/conf.d/squirrelmail.conf
```

En el fichero /etc/squirrelmail/apache.conf tan sólo deberemos descomentar el último apartado, que activa el acceso por https de manera automática (en el caso de encontrarse todos los plugins necesarios) y descomentar el apartado central para definir nuestro virtual host, tal que:

```
# users will prefer a simple URL like http://webmail.example.com
<VirtualHost 172.29.21.15:80>
        DocumentRoot /usr/share/squirrelmail
        ServerName webmail.linuxsilo.net
</VirtualHost>
```

Por supuesto necesitaremos una entrada en nuestro servidor de DNS que dirija los accesos a webmail.espinozahernandez.ni a la dirección IP de nuestra máquina. Si no deseamos tener un virtualhost webmail.example.com específico. La configuración por defecto de SquirrelMail trata de acceder al servidor IMAP a través del puerto 143 del localhost, por lo que deberemos tener activado el servidor en ese puerto. Si se han seguido todos los pasos, nuestro servidor IMAP sólo acepta conexiones sobre la interfaz local, por lo cual deberemos modificar el fichero /etc/cyrus.conf:

```
imap    cmd="imapd -U 30" listen="localhost:imap" prefork=0 maxchild=100
```

Se debe reiniciar el servidor Cyrus tras el cambio: /etc/init.d/cyrus21 restart. Para comprobar la correcta instalación de Squirrelmail podemos usar el script configtest.php, localizable en http://webmail.linuxsilo.net/src/configtest.php.

8. Conclusiones

Con esto nuestro servidor Web estará listo y funcionando, para comprobarlo, accede a través de tu navegador favorito a webmail.espinozahernandez.ni y podras acceder a la cuenta del usuario denis.

9. Ejercicios Práctico

Antes que todo recuerda que tu servidor DNS debe de estar funcionando para que tu máquina sea capaz de resolver los nombres de tu empresa. Ahora, como parte de las ampliaciones que le estas haciendo a tu empresa deberás configurar un servidor de correo electrónico como el explicado en la práctica. Deberás añadir al menos a tres usuarios y cada uno de esos usuarios deberá de tener dentro de su inbox, buzones con los nombres: **trabajo, amigos, otros**. Prueba mandar correos a otros servidores de tus compañeros, recuerda que para ello deberás de cambiar el DNS que ocupas por el de tu compañero al que deseas enviarle el correo.

10. Referencias

- http://linuxsilo.net/articles/postfix.html

PRÁCTICA 9: CONFIGURACIÓN DE UNA CENTRALITA DE VOZ SOBRE IP

Autor: Denis Leopoldo Espinoza Hernández

UNAN – León

1. Introducción

En esta práctica instalaremos y configuraremos una centralita de voz sobre IP. Para ello instalaremos un servidor de voz sobre IP de libre distribución llamado Asterisk el cual permite crear y gestionar nuestra propia centralita, gestionar extensiones y efectuar llamadas internas sin pasar por el operador telefónico, para dar servicios de telefonía inteligente mediante reconocimiento de voz, para locuciones informativas automáticas, incluso poner nuestra canción favorita como tono de llamada en espera.

2. Objetivo

Los objetivos para ésta práctica son:

- Instalar asterisk y describirlos principales elementos de la configuración.
- Crear usuarios y lograr la comunicación entre los mismos.
- Instalar un Sofphone (X-Lite) para hacer llamdas entre las PC`s de los estudiantes.
- Conectar varias centrales de voz sobre IP de las configuradas en el Laboratorio.

3. Referencias

- http://www.slackware.cl/files/slackware-asterisk.pdf
- http://www.imaginar.org/iicd/tus_archivos/TUS9/3_manual_asterisk.pdf
- http://www.voipnovatos.es
- http://www.sinologic.net
- http://linux.softpedia.com/get/Communications/Internet-Phone/X-Lite-5595.shtml

XI. BIBLIOGRAFÍA

Bibliografía Básica:

- James F. Kurose, Keith W. Ross, Redes de Computadores: Un enfoque descendente basado en Internet, Segunda Edición.
- Andrews S. Tanenbaum, Redes de Computadores, Cuarta Edición 2003.
- Fred Halsall, Redes de Computadores e Internet, Quinta Edición 2006.
- William Stallings, Comunicaciones y Redes de Computadores, Séptima Edición 2004.

Referencias de Internet:

Las referencias a Internet se encuentran al inicio de cada uno de los capítulos del temario y al final de cada una de las prácticas de laboratorio.