UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERIA

EXPERIENCIAS EN LA INSTALACION Y ADMINISTRACION DE REDES BASADAS EN TCP/IP EN LA FACULTAD DE INGENIERIA

ING. FREDY JARA D.

JULIO 1994

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERIA
INSTITUTO DE MATEMATICA Y CALCULO APLICADO

Trabajo de ascenso presentado por el Profesor Fredy Jara Diaz Ante el Ilustre Consejo de la Facultad de Ingeniería, para ascender en el escalafón del personal Docente y de Investigación de la Universidad de Carabobo a la categoría de Profesor Asociado.

JULIO 1994

Indice

CAPITULO I

CAPITULO II

CAPITULO III

CAPITULO IV

Introducción

Las redes de computadoras han revolucionado el mundo moderno; su uso ya se ha escapado del ámbito de los centros educativos, empresas y entidades gubernamentales, para llegar al común de los hogares; este fenómeno ha ocurrido en un corto período de tiempo, constituyéndose en una revolución que lejos de mostrar signos de estabilizarse o detener su crecimiento, se desarrolla cada vez con mayor fuerza y a pasos cada vez mas grandes.

Inicialmente, en las primeras etapas del desarrollo de redes de computadoras, fueron concebidos muchos enfoques que permitían la interconexión de varios computadores; se desarrollaron también una gran cantidad de protocolos que permitían que dos computadoras se comunicaran e intercambiaran información; estos protocolos, desarrollados por empresas comerciales, generalmente fueron concebidos con el propósito de satisfacer los requerimientos de una aplicación particular; cada aplicación desarrollada para trabajar en la red, requería también el desarrollo de un conjunto de protocolos que le permitían a dicha aplicación operar en la red, ocultándole al usuario de la misma, los pormenores de esta operación, que desde el punto de vista del usuario parecía como transparente. Muchos de estos protocolos desarrollados para una aplicación particular, tenían los mismos propósitos que los desarrollados para otras aplicaciones, pero debido a que no estaban estandarizados, no podían ser usados en la otra aplicación.

Algunas empresas comerciales resolvieron este problema de duplicidad de esfuerzo, incorporando en sus sistemas operativos un conjunto de protocolos que pudieran ser compartidos por todas las aplicaciones que se desarrollasen para trabajar en redes (Digital desarrolló DECnet; IBM desarrolló SNA etc.), sin embargo, esas soluciones tenían el inconveniente de que eran propietarias de las empresas que las

desarrollaron y generalmente solo funcionaban en un hardware particular, que por supuesto era desarrollado por la misma empresa.

Si bien es cierto que entidades independientes al adquirir los equipos y programas a una empresa particular podían establecer sus redes de computadoras con aplicaciones importantes, la necesidad de intercambiar información con otras entidades que también tenían sus redes pero desarrolladas por otra empresa comercial, conducía generalmente al mismo problema: los protocolos no estaban estandarizados y por tanto no podían ser compartidos por las aplicaciones; mas grave aun, los protocolos se desarrollaban para un hardware específico, por lo que no eran fácilmente trasladables a otro y por lo general también se desconocían los detalles de su implementación .

Varios intentos de desarrollar estándares abiertos (que no pertenecen a una empresa particular y cuyas especificaciones son públicas) comenzaron a realizarse, con el propósito de, en primer lugar, hacer posible la interconexión de redes fisicas diferentes y en segundo lugar lograr que estas redes operen como una unidad coordinada en la cual los equipos que la integran funcionan cooperativamente. En la actualidad se han hecho muchos avances en ese sentido, sin embargo, la única aproximación a un estándar universal de protocolos, que opere en cualquier plataforma lo constituye el conjunto de protocolos conocidos como TCP/IP, los cuales han sido probados en gran escala en una red de alcances mundiales, integrada por casi dos millones de computadores. Esta red tuvo su base en una red creada por el Departamento de Defensa de los Estados Unidos (DARPA, Defense Advanced Research Projects Agency) y en la actualidad interconecta los principales Centros de Investigación de ese país y es conocida con el nombre de Internet.

En nuestro país, el CONICIT estableció una red con el propósito de distribuir información a la comunidad científica; esta red conocida como SAICYT (Sistema Automatizado de Información Científica y Tecnológica) está conectada a la red Internet y aun cuando inicialmente no funcionaba bajo los protocolos TCP/IP, en la

actualidad está evolucionando hacia esos protocolos y la Universidad de Carabobo, específicamente en la Facultad de Ingeniería, desde hace un año dispone de un punto de acceso a la red SAICYT y a través de ésta, a la red Internet. Este punto de conexión o nodo fue instalado, configurado y administrado por nosotros y hemos creído conveniente presentar en este trabajo nuestras experiencias, de manera que las mismas puedan ser aprovechadas por cualquier otra dependencia o Facultad y en general por la Universidad.

El trabajo consta de una parte teórica que describe los protocolos y servicios que posibilita el TCP/IP (capítulos I, II y III), y una parte que describe la forma en que se configuran y administran algunos de estos servicios en la plataforma en la que nosotros hemos trabajado, Unix System V de SCO(capítulo IV). Los apéndices A y B, contienen los comandos básicos del sistema operativo Unix y del editor **vi** que viene con ese sistema, porque sabemos que en la Universidad el uso de este sistema operativo no está difundido y la mayoría de las personas desconoce las potencialidades del mismo, nosotros nos vimos en la necesidad de usarlo porque es en este sistema en donde se han desarrollado la mayoría de los servicios y el único que en la actualidad garantiza proveer todos los servicios que se mencionan, ello se debe básicamente a que fue Unix el primer sistema operativo que incluyó como parte integral, los servicios de red de los protocolos TCP/IP que estaban siendo experimentados en la red del Departamento de Defensa de los EEUU y que ahora ha evolucionado al Internet que conocemos actualmente.

Capitulo I

Lo que en la actualidad se conoce como Internet es una red que conecta las redes de los principales centros de investigación en los Estados Unidos a través de canales de comunicación de alta velocidad que constituyen la columna vertebral o backbone de la red Internet. Los principales integrantes de esta red son: La red de la Fundación Nacional de Ciencia(National Science Fundation, NSFnet), la red del Departamento de Energía(Department of Energy, DOEnet), la red del Departamento de Defensa (Department of Defense, DODnet), la red de la Agencia de Servicios Humanos y Salud (Health and Human Services Agency, HHSnet) y la red de la Agencia Nacional para el Espacio y Aeronautica(National Aeronautics and Space Agency, NASAnet). Esta red, constituida por este conjunto de redes está fundamentada en una serie de protocolos que han demostrado en la práctica la posibilidad cierta de desarrollar protocolos universales, con una filosofia de sistema abierto, y especificaciones públicamente disponibles, por lo cual pueden ser desarrollados por cualquier empresa, Institución, Centro de Investigación o inclusive por una iniciativa personal.

Al conjunto de protocolos usados en la red Internet se les conoce mundialmente como TCP/IP y a continuación se hará una revisión de sus características principales; en capítulos posteriores se profundizará en los aspectos mas importantes de esos protocolos.

1.1 Que es TCP/IP?

El nombre de TCP/IP se usa para referirse a un conjunto de protocolos desarrollados para permitir que un grupo de computadoras compartan recursos de

índole variada, sin importar, la localización fisica, la topología de la red, la arquitectura ni el fabricante de los equipos que se interconectan. El nombre de TCP/IP proviene de dos de los protocolos mas importantes de este conjunto de protocolos, el TCP(Transmision Control Protocol) y el IP (Internet Protocol), cuyo uso será descrito en detalle en secciones posteriores. Quizás TCP/IP no sea el nombre mas apropiado para estos protocolos pues puede darse el caso de aplicaciones que no están basadas en TCP, sino que usan un protocolo alterno que pertenece también al grupo de protocolos de Internet(UDP, por ejemplo); muchas personas prefieren llamar a estos protocolos como el conjunto de protocolos de Internet, sin embargo, puesto que el uso del nombre TCP/IP está generalizado, se usará en este trabajo ese nombre para designar al conjunto de protocolos y cuando haya necesidad de referirse a un caso particular de protocolo se dirá explícitamente. En la siguiente sección se va a discutir acerca de la arquitectura de redes, de manera de proveer una base para la discusión propiamente dicha de los protocolos TCP/IP.

1.2. Arquitectura de Redes

La arquitectura de redes se refiere a la forma en la que los distintos elementos que conforman la red se relacionan e interactuan para lograr un propósito específico; no debe ser confundida con la topología que se refiere al diseño fisico de la red lo que viene determinado por la forma en que los cables (o cualquier otro medio fisico que se utilice) interconectan los equipos integrados a la red.

Las redes modernas están diseñadas en una forma altamente estructurada en donde las tareas necesarias para realizar todas las actividades relacionadas con la comunicación entre dos máquinas se dividen y agrupan por algún criterio; cada uno de esos grupos de actividades constituyen un nivel o capa que maneja una parte específica del problema. Esta estructuración en capas se sustenta en las siguientes premisas:

a) Que cada capa provee un servicio a la capa inmediatamente superior (entendiendo el servicio como una serie de primitivas u operaciones que esta capa está en capacidad de proveer a sus usuarios, la capa superior).

b) Que la relación entre una capa superior y otra inferior esté gobernada por un conjunto de reglas o formatos de intercambio de información y solicitudes de servicios denominada interfase.

c) Que la apariencia funcional (el llamado modelo conceptual) sea apreciada como que si la comunicación se hace entre capas iguales que operan en máquinas distintas; estas capas iguales operando en máquinas distintas, intercambian información, cuyo formato e interpretación está gobernado por un conjunto de reglas que constituyen los llamados protocolos.

Los protocolos permiten que uno especifique o comprenda la comunicación sin conocer los detalles de un vendedor particular de hardware de red y proporcionan un nivel de abstracción similar al que proveen los lenguajes de programación. La estructuración en capas por otra parte permite manejar un problema de gran complejidad, como lo es el establecimiento de una conexión entre dos o mas sistemas, como un conjunto de problemas de naturaleza mas simple (subproblemas) que pueden ser desarrollados en forma independiente. El gran desarrollo alcanzado por las redes se debe en gran parte al hecho de haber estructurado la solución al problema en la forma antes mencionada y en donde los conceptos de interfase y protocolos han sido determinantes. La naturaleza tan variada del tipo de problemas que presenta una conexión de máquinas que requieren un trabajo cooperativo, hace dificil sino casi imposible el establecimiento de un solo protocolo para manejar esas tareas; es posible sin embargo, agrupar un conjunto de tareas para que se ocupen de una parte específica del problema.

Conceptualmente el modelo estructurado en capas permite que uno piense que el problema de comunicación de dos máquinas es resuelto colocando en cada máquina una serie de módulos de software (protocolos), organizados verticalmente en niveles o capas; a cada capa o nivel le corresponde el tratamiento de una parte específica del

problema de comunicación. Si un módulo en una máquina A, ubicado en un nivel superior en el conjunto de capas, desea enviar una información a su equivalente en una máquina B, debe transferir esa información verticalmente hacia abajo a través de todos los módulos que están por debajo de él, hasta llegar finalmente al medio fisico que permite que las dos máquinas se comuniquen. En la máquina B, la información es recibida y debe viajar verticalmente hacia arriba a través de todas las capas de protocolos, hasta llegar al módulo destino. Es importante destacar que en el viaje que hace la información a través de las capas, estas realmente no conocen la estructura interna ni la interpretación de los datos que le entrega la capa superior; cada capa inferior se limita a proveer un determinado servicio, que es solicitado por una capa superior siguiendo ciertas reglas (estas reglas constituyen la interfase). Cada capa puede requerir agregar cierta información a la información recibida de la capa superior, con la finalidad de cumplir con su función; en la máquina receptora, cada capa va interpretando la información que corresponde a su nivel, según las reglas establecidas para la comunicación (estas reglas constituyen los protocolos). La figura 1.1 ilustra la funcionalidad del modelo conceptual.

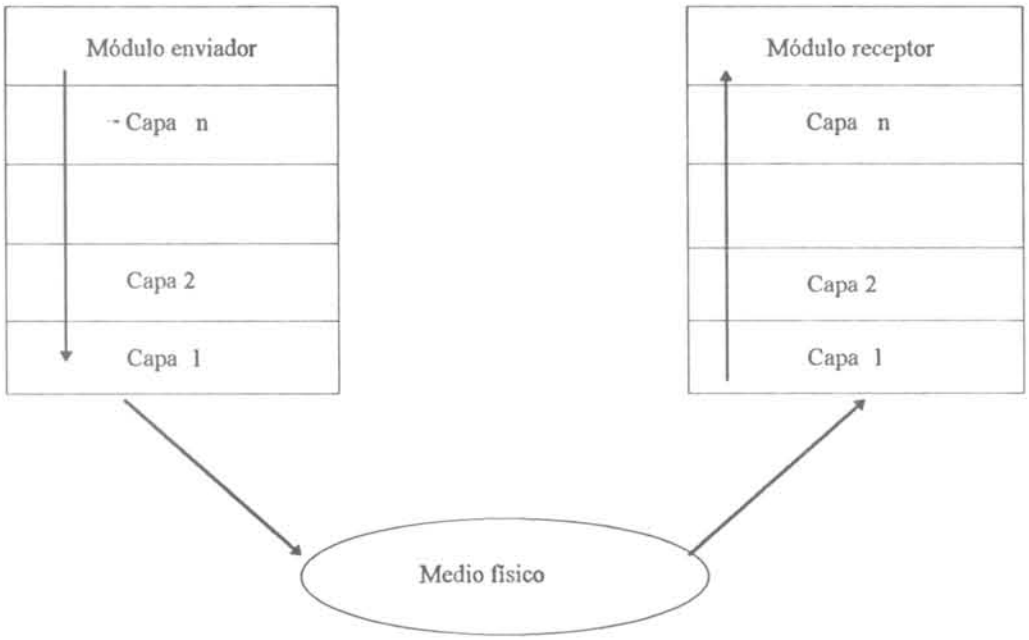


Figura 1.1. Modelo conceptual basado en una estructuración por niveles o capas

En este modelo conceptual que se ha presentado, hay muchos aspectos que tienen que ser definidos al momento de llevarlos a la práctica; el mas importante en esta concepción de capas es la forma en la que el problema global de comunicación se divide en subproblemas, lo cual conlleva a definir el número de capas o módulos de protocolos que tendrá el modelo. Es fundamental para que dos máquinas se comuniquen que estén sustentadas en el mismo modelo pues como se mencionó anteriormente una de las premisas es que la comunicación de las máquinas se lleva a cabo entre capas o módulos de software (protocolos) del mismo nivel, operando en máquinas distintas. Es también necesario aclarar que en una misma capa pueden convivir varios protocolos que manejen una determinada situación con un enfoque diferente.

En la actualidad dos modelos acaparan el mundo de las redes: El modelo de 7 capas presentado por la Organización Internacional para la Estandarización (International Standarization Organization, ISO), conocido como el modelo de referencia para la interconexión de sistemas abiertos(Open System Interconection, OSI) y el modelo Internet de 4 capas conocido popularmente como TCP/IP o mas formalmente conjunto de protocolos de Internet. El modelo OSI es un estándar, sin embargo, no ha alcanzado la difusión que se esperaba, básicamente porque muchas de los esquemas que propone el modelo son dificiles de implementar eficientemente en la práctica; el segundo modelo, el TCP/IP, si bien no es un estándar en el sentido de que no fue presentado por uno de esos comités que nombran para el establecimiento de un estándar, en la práctica se ha convertido en lo que se conoce como un estándar de facto y es quizás el único que en la actualidad garantiza el establecimiento de redes heterogéneas de cualquier tipo y naturaleza, pues en los momentos actuales existe en todas las plataformas de Sistemas Operativos alguna versión de TCP/IP que le permite integrarse a redes basadas en este modelo; a diferencia del modelo de referencia de ISO, los protocolos y la estructuración del modelo TCP/IP se han ido desarrollando a partir de experiencias prácticas que una vez probadas se adoptan como estándar; la filosofia de Internet siempre ha sido la de estar abierta a las sugerencias de los usuarios para las mejoras y actualizaciones de los protocolos que se usan.

1.3. Descripción general del modelo de capas de TCP/IP

El modelo conceptual de TCP/IP es organizado en cuatro capas de software que se construyen sobre una quinta capa de hardware; por esta razón se dice que el TCP/IP es un modelo de cuatro capas (4 niveles de protocolos), en otras palabras el problema de comunicación ha sido dividido en cuatro subproblemas y para cada uno de estos subproblemas se han desarrollado los protocolos correspondientes al tratamiento del problema específico; este esquema conceptual se muestra en la figura 1.2.

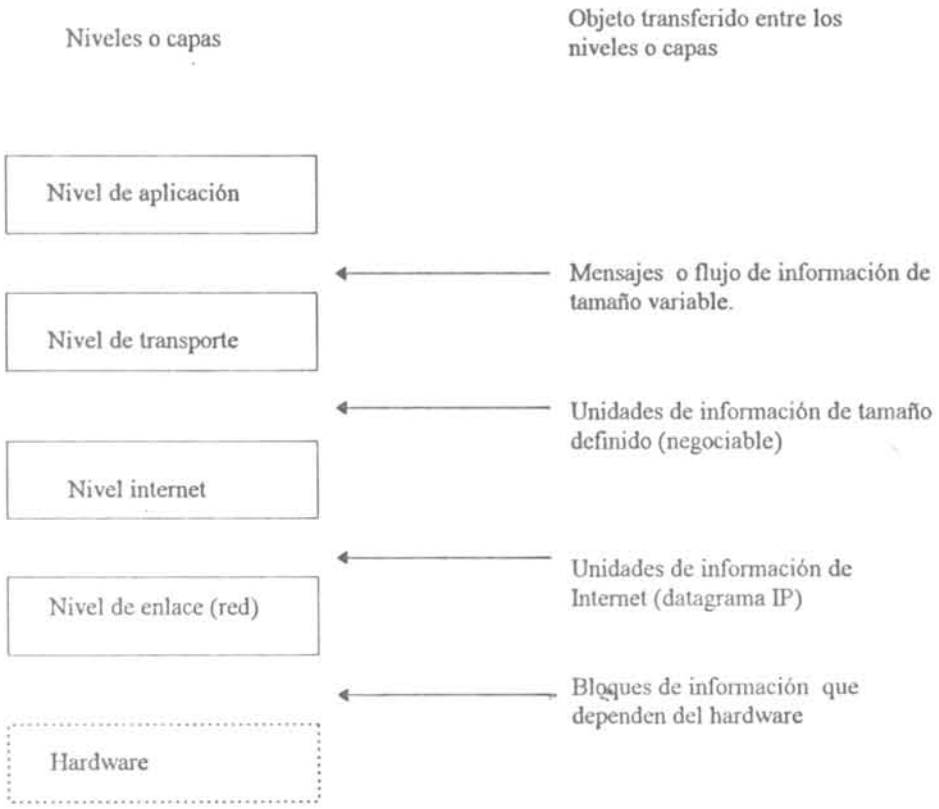


Figura 1.2 Modelo conceptual de cuatro capas de TCP/IP

1.3.1. Nivel o capa de aplicación

Este nivel o capa tiene que ver con el desarrollo de aplicaciones que puedan acceder o proveer un servicio a través de la red. Un usuario que va a desarrollar una aplicación define primero un protocolo para esa aplicación, que por supuesto reflejará la actividad que se espera cumplir con esa aplicación (por ejemplo una aplicación como correo electrónico, requerirá un destinatario, un remitente, el texto o contenido del mensaje etc.), La definición del protocolo de la aplicación conlleva al establecimiento del conjunto de comandos y al formato que la aplicación usará para comunicarse con otra aplicación similar en otra máquina integrada a la red. Desde el punto de vista del que desarrolla la aplicación el no tiene que preocuparse de como la información será trasladada hasta la máquina destino, puesto que este problema es manejado a otro nivel inferior. Lo único que la aplicación debe conocer es la forma en la que debe solicitar el servicio al nivel inferior (en el caso del modelo Internet el nivel siguiente es la capa de transporte).

1.3.2. Nivel o capa de transporte

Cuando se desea enviar información entre dos máquinas a través de un medio surgen muchos problemas que es necesario considerar y manejar. En primer lugar está el hecho de la capacidad finita de manejo de información que tienen los equipos involucrados, lo que plantea la necesidad de disponer de un control de flujo de información que permita regular tanto el tamaño o cantidad de información que pueda ser enviada como una sola unidad, así como también la rata o número de unidades de información que pueden ser enviados en un tiempo determinado.

El que desarrolla la aplicación no tiene que conocer de esta limitante y por eso el entrega su bloques de información en un tamaño y a una rata que no necesariamente se compaginan con los soportados por los elementos de la red; la capa de transporte se encargará de manejar la situación para lo cual divide el bloque de información recibida de la aplicación en bloques de un tamaño apropiado a la

capacidad de los elementos involucrados. Al hacer esta división la capa de transporte debe proveer la información necesaria para que el nivel de transporte de la máquina destino pueda reensamblar estas unidades y reconstruir la información de manera de hacerle llegar a la aplicación destino la información tal y como fue entregada a la capa de transporte de la máquina fuente.

En segundo lugar, hay que considerar la posibilidad de que ocurran errores de transmisión que pueden hacer que la información llegue con errores al destino o que inclusive unidades o bloques completos de información sean perdidos. El nivel de transporte debe garantizar también que la información llegue al destino, libre de errores y en la secuencia requerida.

En tercer lugar se supone que esta capa va a prestar sus servicios a una serie de aplicaciones(correo electrónico, transferencia de archivos, ejecución remota de comandos y procedimientos, etc.) y por tanto debe ser capaz de aceptar peticiones de servicios desde varias aplicaciones en la máquina origen y estar en capacidad de entregarlas a las respectivas aplicaciones en las máquinas destino.

En el modelo Internet la información es transferida en unidades denominadas datagramas. Un datagrama no es mas que una colección de data de determinada longitud, que es enviada como un solo bloque (en el modelo OSI, el equivalente es el paquete, packet). La información recibida de la capa de aplicación es dividida (en caso de ser necesario) en bloques de información ; cada uno de estos bloques de información constituyen un datagrama , y son enviados a la red (capas inferiores), en forma individual. El nivel de transporte le agrega a cada datagrama un encabezado o header que le permite resolver entre otras cosas los tres problemas que se acaban de plantear. En el modelo Internet no existe un solo protocolo que cumpla con las labores de la capa de transporte, sino que existen varias alternativas, esto con la finalidad de dar una mayor flexibilidad y posibilitar que las aplicaciones escojan el nivel de transporte que maneje en la forma mas eficiente posible sus necesidades.

Básicamente existen dos protocolos en el nivel de transporte del modelo Internet disponibles para las aplicaciones normales(hay además otros con propósitos mas específicos, que se discutirán brevemente mas adelante): el TCP (Transmission Control Protocol) y el UDP (User Datagram Protocol)

1.3.2.1. Transporte basado en el protocolo TCP

Este es el protocolo mas usado en Internet como protocolo de transporte, tanto que se identifica al conjunto de protocolos con su nombre. Este protocolo es capaz de proveer todos los servicios deseados en un nivel de transporte como son: el dividir, en caso de ser necesario, los bloques de datos provenientes de la aplicación en bloques mas pequeños, ajustados al tamaño soportado por los elementos de la red; garantizar que estos bloques podrán ser ensamblados para reconstruir la información original entregada por la aplicación; regular la rata o flujo de información y por último posibilitar la interacción con diferentes aplicaciones. Para cumplir con estos propósitos el protocolo TCP, debe agregar a los distintos bloques en que ha partido la información proveniente de la aplicación, un encabezado cuyo formato se muestra en la figura 1.3.

<table>
<tr><td colspan="3">puerto fuente</td><td>puerto destino</td></tr>
<tr><td colspan="4">número de secuencia (número de octeto)</td></tr>
<tr><td colspan="4">número de reconocimiento</td></tr>
<tr><td>corrimiento
de la data</td><td>reservada</td><td>U A P R S F
R C S S Y Y
G K H T N N</td><td>ventana</td></tr>
<tr><td colspan="3">chequeo de redundancia (checsum)</td><td>indicador de urgente</td></tr>
<tr><td colspan="4">parte de la data proveniente de la aplicación</td></tr>
</table>

Figura 1.3. Estructura de un datagrama TCP (header TCP y data de la aplicación).

recibir la máquina fuente un reconocimiento positivo de haber sido recibido correctamente el datagrama..

El campo número de reconocimiento es parte del mecanismo que se usa para tener certeza de que un datagrama llegó a su destino libre de errores. El mecanismo trabaja en conjunto con el número de secuencia y está basado en un esquema de reconocimiento acumulativo, pues lo que en realidad reporta es la cantidad de octetos que ha logrado acumular en forma correcta (sin errores y en secuencia). Este esquema tiene ventajas y desventajas; la principal ventaja es que es muy fácil de generar y no da lugar a ambigüedades. La principal desventaja es su ineficiencia como consecuencia de que el reconocimiento es por bloques de datos que ha logrado ensamblar correctamente y no por los datagramas recibidos sin errores, lo cual puede dar lugar a que por ejemplo, si en una transmisión de un grupo de datagramas el segundo de ellos fue recibido con error y todos los demás arribaron correctamente, se enviará reconocimiento positivo solo por el primer datagrama y no por el resto; en otras palabras, una vez que un datagrama ubicado en una cierta posición de secuencia es recibido con error o se pierde, todos los datagramas que siguen en la secuencia deberán ser transmitidos de nuevo, a pesar de que hayan sido recibidos correctamente. Algo que es importante destacar es que lo que realmente se escribe en el campo que corresponde al reconocimiento positivo, es la ubicación del próximo octeto que el receptor espera recibir, lo que en definitiva es interpretado por el emisor, como que el receptor ha logrado ensamblar correctamente hasta el octeto que antecede al indicado por el campo de reconocimiento positivo.

13.2.2. Transporte basado en el protocolo UDP

El otro protocolo de que se dispone en la capa o nivel de transporte en el modelo Internet es el UDP (User Datagram Protocol). Este es un protocolo mas simple que el TCP que fue provisto para soportar el nivel de transporte de aplicaciones menos exigentes, en las cuales la información que ellas intercambian es lo suficientemente pequeña como para garantizar que no será necesario dividirla en

varios bloques y en consecuencia tampoco será requerido el control de secuencia que se usa para garantizar el ensamblaje correcto de los distintos datagramas (se supone que la información recibida de la aplicación cabe en un solo datagrama). El servicio de transporte que este protocolo provee al no incluir la división de la data en bloques mas pequeños ni el control de secuencia es obviamente mas sencillo de implementar, tiene menos cosas que chequear y su encabezado es mas pequeño que el del TCP, lo que en definitiva se traduce en una mayor eficiencia; sin embargo, esto requerirá un mayor esfuerzo en la capa de aplicación, la cual deberá asumir (en caso de ser necesario) las labores de división de los bloques de manera de adaptarlos a la capacidad de los elementos de la red y el reensamblado de los mismos.

UDP no provee un esquema de reconocimiento que asegure que los mensajes llegan a su destino; tampoco provee un mecanismo para controlar la rata a la cual los mensajes fluyen entre las máquinas, en consecuencia es posible perder mensajes, recibir mensajes duplicados o fuera de orden y además es posible que la información esté llegando mas rápido de lo que la aplicación que la recibe es capaz de procesarla; de manera que las aplicaciones que usan UDP como medio de transporte, tienen la responsabilidad ellas mismas de proveer los mecanismos de control necesarios para garantizar la funcionalidad de las mismas. La idea de disponer de un protocolo de transporte tan elemental y poco confiable, es que los desarrolladores de aplicaciones, considerando las exigencias particulares de las mismas, establezcan el mecanismo que necesiten, ajustado exactamente a sus exigencias, lo que en general se traducirá en una mayor eficiencia en la operación del protocolo, pues los controles necesarios para esas aplicaciones son muy elementales de implementar y en muchas de ellas ni siquiera se hacen necesarios.

El encabezado o header de un datagrama UDP es mucho mas sencillo que el del TCP y se muestra en la figura 1.4, en la cual se puede apreciar que consta de 8 octetos: los dos primeros octetos corresponden al número del puerto que identifica la aplicación fuente; los dos siguientes identifican el número de puerto de la aplicación destino. Al igual que en el protocolo TCP, esto permite que el protocolo UDP distinga

entre múltiples aplicaciones destinos que él les esté dando servicio dentro de un computador particular. Es obligatorio especificar la aplicación destino(puerto destino) del datagrama, mas no la fuente, la cual, es opcional; si se especifica el puerto fuente, éste será la aplicación a la cual le serán enviadas las respuestas; si no se especifica puerto fuente, se considerará como cero. UDP provee un mecanismo mínimo de seguridad para garantizar la integridad de los datos, para lo cual, como parte del encabezado se proporciona un campo(dos octetos) para almacenar la longitud total del datagrama (en octetos) y un campo(dos octetos) para chequeo de redundancia cíclica (checksum). Después de estos 8 octetos de encabezado viene la data proveniente de la aplicación; Un datagrama no puede tener nunca menos de 8 octetos (lo que correspondería a un datagrama constituido por el puro encabezado, sin datos de la aplicación).

Puerto fuente	Puerto destino
Longitud del mensaje (datagrama)	chequeo de redundancia(checksum)
Data . . .	

Figura 1.4. Formato de un datagrama UDP (encabezado y data)

1.3.3. Nivel o capa internet (IP)

En el modelo Internet el nivel o capa que sigue a la de transporte es el nivel internet o IP. El nivel de transporte (cualquiera sea el protocolo que se use para ese nivel) entrega al nivel IP la información proveniente de la capa de aplicación, una vez que le ha agregado el encabezado del nivel de transporte que le corresponda según el protocolo. Es importante observar que cualquier información proveniente de una

aplicación tiene dos niveles de destinatario: Un nivel, que corresponde a la aplicación destino y el otro nivel que corresponde al computador o host donde esa aplicación se ejecuta. El nivel de transporte se encarga de todo lo referente al primer destino, o sea la aplicación destino y el nivel IP se encarga del segundo destino o sea, el host destino.

El protocolo IP se sustenta en un servicio conocido en el ambiente de redes como no confiable y sin conexión. El servicio se dice que es no confiable porque la liberación del paquete no está garantizada; los paquetes pueden ser perdidos, duplicados o liberados fuera de orden y el protocolo no detectará esas condiciones por lo que no informará ni al origen ni al destino de tal situación. Se dice que es sin conexión porque cada paquete(datagrama) es tratado en forma totalmente independiente de los otras; de hecho el nivel IP desconoce si los paquetes tienen alguna relación y si van dirigidos a la misma aplicación; el solo conoce a que protocolo del nivel de transporte debe ser entregado el paquete y es en ese nivel en donde se identifica la aplicación destino.

El protocolo IP es quizás el mas importante de los protocolos de Internet; el provee tres importantes definiciones para el mundo Internet: a) Define la unidad básica de transferencia de datos y especifica el formato exacto de todos los datos que fluirán a través de la red. b) Define y realiza la función de enrutamiento, la cual consiste en escoger o determinar una vía a través de la cual los paquetes serán enviados y c) establece el conjunto de reglas que enmarcan el servicio no confiable de liberación de paquetes, entre las que destacan la forma en que los hosts procesarán los paquetes, como y cuando se generarán mensajes de error y las condiciones bajo las cuales los paquetes serán desechados.

El protocolo IP define un formato para el datagrama IP (datagrama Internet), que al igual que como se vio para TCP y UDP en el nivel de transporte, consta de un encabezado y una data; en este caso la data proviene de la capa de transporte y en ella se encuentra imbuido el encabezado que colocó el nivel de transporte junto con la data proveniente de la aplicación. El datagrama IP contiene como parte fundamental de su

encabezado las direcciones (denominadas direcciones IP) que identifican los hosts fuente y destino. Mas adelante se explicará en detalle en que consiste una dirección IP y la forma en que se acostumbra a representarla. Las figuras 1.5 y 1.6 muestran el formato general del datagrama internet (datagrama IP) y el detalle del encabezado respectivamente y cuyos campos serán discutidos a continuación.

Figura 1.5. Formato general de un datagrama IP

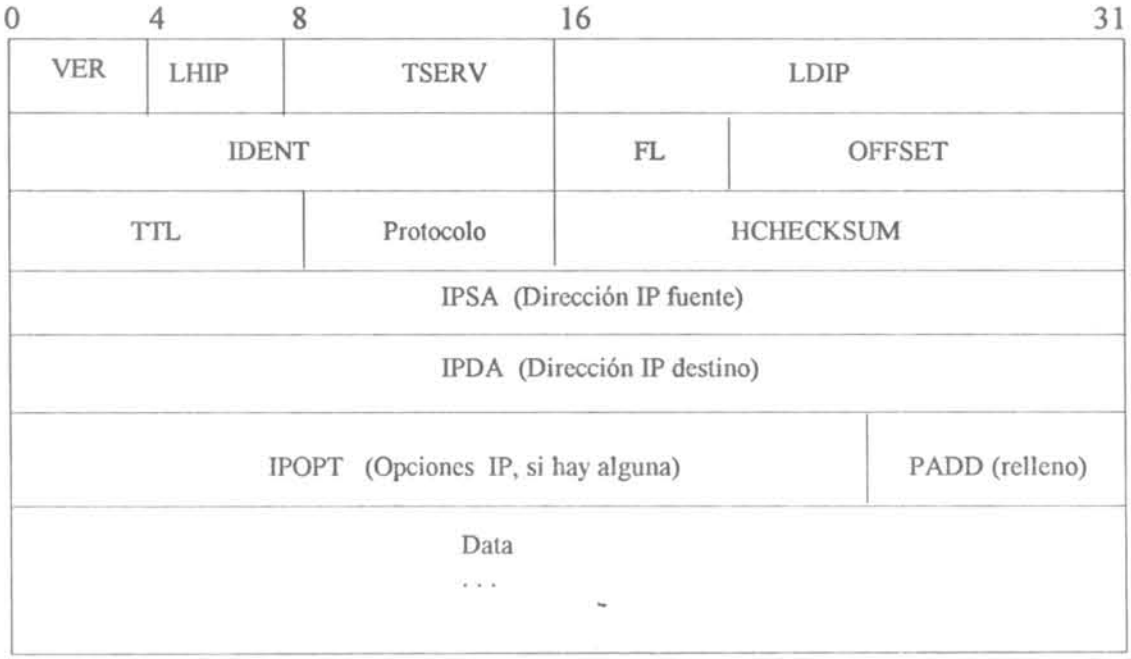


Figura 1.6. Diagrama detallado del encabezado IP

El encabezado de IP tiene la particularidad de no tener una longitud fija; está constituido como mínimo por 20 octetos (5 palabras de 32 bits c/u). A continuación se describirán brevemente los distintos campos del encabezado IP.

El campo identificado en la figura 1.6 como VER(bits 0-3 de la primera palabra del encabezado), tiene un ancho de 4 bits, se usa para indicar la versión del protocolo IP que se usó para crear el datagrama; el propósito del mismo es poder garantizar que todas las máquinas por donde requiera pasar el datagrama, interpreten en forma consistente el encabezado, el cual ha venido evolucionando y no existe garantía de que todas las máquinas en el Internet tengan la misma versión de protocolo IP. Si se conoce la versión es posible tomar las previsiones en el protocolo, de manera tal que mantenga la compatibilidad con versiones anteriores o sencillamente se rechacen los datagramas que no se correspondan con la versión que esté en ejecución en la máquina que lo recibe.

El campo LHIP (bits4-7 de la primera palabra del encabezado) se usa para especificar la longitud del encabezado IP (solo del encabezado), esto es necesario pues en el caso del protocolo IP, la longitud del encabezado no es fija y depende del hecho de que se incluyan o no las opciones y sus correspondientes rellenos(padding); la longitud del encabezado se especifica en palabras de 32 bits y puede tener como mínimo 5 palabras (que corresponde a un encabezado sin opciones) y como máximo 15.

El campo TSERV (bits 8-15 de la primera palabra del encabezado), está previsto para usar en un mecanismo de control de flujo; la idea básica es en primer lugar, categorizar los datagramas que se envían según su importancia y en segundo lugar, hacer que el software de esta capa, en todas las máquinas, respete esta categorización; de esta forma sería posible implementar algoritmos de control de congestión que no se verían afectados por las congestiones de la red, pues los datagramas con propósitos de control, tendrían la máxima prioridad y en consecuencia se liberarían primero. Desafortunadamente, casi todo el software que se ha

desarrollado hasta ahora ignora este campo del datagrama IP, sin embargo, el aspecto importante está en el hecho de que ya el protocolo tiene una previsión para el manejo de una situación conflictiva de la red. Realmente existen en este campo previsiones que van mas allá de la simple categorización del datagrama, previéndose la posibilidad de que se especifiquen aspectos referentes a las características deseables en las rutas que se elijan para llegar a un destino cualquiera(rutas de bajo retardo, rutas de alta velocidad o rutas de alta confiabilidad); en general, los algoritmos de este nivel de protocolos, eventualmente, podrían usar esta información como un criterio para escoger entre varias alternativas, sin embargo, debido al hecho de la heterogeneidad de las redes y de los medios que se usan para conectarlas, es imposible garantizar que siempre se podrá cumplir con las peticiones que en este aspecto indique la máquina fuente

El campo LDIP (bits 16- 31 de la primera palabra del encabezado), se usa para especificar la longitud completa (encabezado mas la data propiamente dicha) del datagrama IP, expresada en octetos; puesto que se disponen de 16 bits, la longitud máxima posible de un datagrama IP, sería de 65535 octetos. En la práctica, los canales involucrados en la conexión de dos máquinas cualesquiera, pueden tener diferentes capacidades; con la finalidad de hacer la comunicación lo mas eficiente posible, la longitud del datagrama generalmente es un parámetro que se negocia al momento de establecer la comunicación, para llegar a un compromiso que saque el mayor provecho a los medios disponibles; el protocolo IP provee un medio de hacer esto automáticamente.

Los próximos tres campos en el encabezado IP tienen que ver con algo que se conoce como encapsulamiento y sus consecuencias (fragmentación). El tamaño de un datagrama es manipulado por el software y en consecuencia el diseñador del protocolo puede escoger cualquier tamaño(en el caso del nivel actual del protocolo IP el datagrama tiene un tamaño máximo de 65535 octetos); sin embargo, cuando un datagrama viaja de una máquina a otra debe pasar por elementos fisicos que imponen restricciones especificas en cuanto a la longitud fisica máxima del bloque de datos que

manejan (frame) y que está definida por la máxima unidad de transferencia (MTU) del medio usado. Idealmente, por razones de eficiencia y simplicidad del protocolo es deseable que un datagrama pueda ser transportado por un frame de red fisica. La idea del encapsulamiento es precisamente hacer que un datagrama sea transportado en un bloque de datos (frame) de los manejados por la red fisica(de tamaño MTU). La solución del problema no es sencilla, pues si bien es cierto que los diseñadores del protocolo pudieron optar por escoger un tamaño de datagrama lo suficientemente pequeño como para garantizar que funciona sobre una amplia variedad de redes fisicas, no es menos cierto que esa solución puede ser ineficiente al desaprovechar las potencialidades de algunas de las redes fisicas y por otra parte no tiene la generalidad deseada en un protocolo que se pretende opere sobre prácticamente cualquier red fisica.

La solución adoptada en los protocolos Internet y que se desarrolla en este nivel IP, es general y trata de aprovechar en forma eficiente las potencialidades de las redes fisicas sobre las que opera, manteniendo como siempre la idea de que todos estos manejos sean transparentes al usuario, es posible que muchos de los esquemas adoptados sean discutibles, pero está comprobado que funcionan y lo hacen eficientemente. A continuación los puntos mas importantes del enfoque dado en el protocolo IP.

a) Desde el punto de vista del usuario, el tiene libertad para escoger el tamaño del datagrama a usar en su aplicación, sin considerar las restricciones impuestas por el hardware de la red.

b) El protocolo IP provee un mecanismo que permite en primer lugar detectar la necesidad de hacer encapsulamiento y en segundo lugar proporciona el mecanismo para dividir automáticamente un datagrama (de un tamaño cualquiera) en pedazos que pueden ser transportados cada uno en un frame de la red fisica. Los pequeños pedazos en los que se divide el datagrama son denominados **fragmentos**, y el proceso en sí se denomina **fragmentación**.

c) Se define el proceso inverso denominado **reensamblaje** y se especifica que el mismo será realizado en el destino, lo que significa que una vez que un datagrama en su paso a través de

distintos elementos físicos en la ruta hacia su destino final es fragmentado en uno de estos elementos, de allí en adelante cada fragmento será tratado como un datagrama independiente; las máquinas intermedias por donde pasa lo encaminan hacia su destino sin conocer su relación con los otros pedazos o fragmentos; solo al llegar a su destino final, los fragmentos comienzan a ser agrupados hasta formar el datagrama original que será entregado al nivel superior. Este esquema tiene sus ventajas y desventajas pues si bien es cierto que el hecho de no tener que ensamblar los fragmentos en los receptores intermedios es mas eficiente, también es cierto que se puede desaprovechar la potencialidad de una red física de mayor capacidad que se encuentre entre el destino final y un punto intermedio. Adicionalmente un esquema como el planteado tiene los mismos inconvenientes que se mencionaron cuando se discutió el protocolo TCP en el nivel de transporte en referencia a la pérdida de fragmentos completos o en caso de errores en algún fragmento.

d) Por último vale la pena mencionar el hecho de que el protocolo IP en referencia al proceso de fragmentación estipula unas exigencias mínimas a los elementos físicos de la red; en primer lugar estipula una unidad mínima de transferencia que debe ser manejada por cualquier red física (mínimo MTU) sobre la que opere el protocolo, esta MTU mínima es 576 octetos(en la práctica esto significa que un datagrama con tamaño 576 octetos no será fragmentado pues todos los elementos soportan ese tamaño como mínimo). En segundo lugar el protocolo establece que los hosts deben tener capacidad para manejar hasta el máximo MTU de la red física a la que se conectan(o sea que si un host se conecta a una red a través de una interfase ethernet, cuyo MTU es 1500 octetos, ese host debe estar en capacidad de manejar datagramas de 1500 octetos).

Los campos IDENT, FL y OFFSET en el encabezado del datagrama IP tienen que ver con el proceso de fragmentación, su control y el reensamblaje. El campo IDENT (bits 0-15 de la segunda palabra del encabezado IP), se usa para identificar el datagrama al cual el fragmento pertenece (si fue fragmentado). El host destino usa el campo IDENT junto con la dirección fuente del datagrama (fragmento) para identificarlo unívocamente. Todos los fragmentos tienen el mismo formato que el datagrama original desde el punto de vista del encabezado, pero se provee un campo que permite conocer a que porción del datagrama original corresponde la data que transporta el datagrama, el OFFSET (bits 19-31 de la segunda palabra del encabezado IP); en este campo se especifica el desplazamiento de la data transportada por el datagrama (fragmento) respecto del origen del datagrama original, expresada en

unidades de octetos; el primer fragmento le corresponde un OFFSET de cero y de allí en adelante, el valor de OFFSET dependerá del fragmento que corresponda y del tamaño de los mismos. El campo FL (bits 16-18 de la segunda palabra del encabezado IP) permite controlar la fragmentación y ayuda en el proceso de ensamblaje de los fragmentos en el destino. A través de este campo se provee un mecanismo que permite determinar si un datagrama recibido es un fragmento de un datagrama mayor o es un datagrama no fragmentado; por medio de este campo es posible también detectar el arribo del último fragmento y determinar la longitud del datagrama original, para lo cual el destino hace uso adicionalmente de los campos OFFSET y LDIP (longitud del datagrama IP).

El campo TTL (Time To Live, bits 0 -7 de la tercera palabra del encabezado IP), especifica la longitud en segundos que se permitirá al datagrama mantenerse en el sistema Internet. La idea con este campo es prevenir que un datagrama se mantenga indefinidamente circulando en la red por alguna falla en los mecanismos de enrutamiento, lo cual ocasionaría un tráfico innecesario. El esquema planteado es sencillo y consiste fundamentalmente en lo siguiente: Cuando una máquina coloca un datagrama en la red (como originario del mismo), le establece un tiempo de vigencia el cual coloca en el campo TTL. En su ruta hacia el destino final, el datagrama pasará a través de varios hosts y redes, cada host por el que pase decrementa en uno el campo TTL del datagrama y chequea su valor; si ha alcanzado el valor de cero, el datagrama es desechado (no se envía a ninguna parte) y se le envía un mensaje de error al origen del mismo.

El campo Protocolo (bits 8-15 de la tercera palabra del encabezado IP) especifica el tipo de protocolo empleado para crear el mensaje que está transportando el área de datos del datagrama y define en consecuencia el formato del área de datos; este campo tiene como propósito permitir que varios protocolos diferentes sean usados en la misma red y usen los servicios del protocolo IP (TCP, UDP ICMP, etc.). Cada uno de los protocolos existentes tiene un número asignado que es el que se

coloca en este campo; dicho número es asignado por una autoridad central para garantizar que todos los protocolos tengan el mismo identificador en todo el Internet.

El campo HCHECSUM (bits 16- 31 de la tercera palabra del encabezado IP) es provisto para asegurar la integridad de los valores que constituyen el encabezado. Este chequeo está limitado exclusivamente al encabezado, lo cual lo hace muy fácil de calcular pues el encabezado está constituido por unos pocos octetos. Este esquema es muy eficiente pero presenta la desventaja de que los protocolos de las capas superiores(como ya se vio cuando se describió TCP y UDP), deben proveer su propio mecanismo de chequeo de integridad de la data ya que el protocolo IP no lo garantiza.

Los campos IPSA (dirección IP de la fuente) e IPDA(dirección IP del destino) son los campos provistos para identificar el elemento en la red que colocó el datagrama y el elemento al que el datagrama está dirigido. Aun cuando un datagrama en su ruta hacia su destino final puede pasar a través de varios hosts intermedios, los cuales reenvían el datagrama, las direcciones fuente y destino se mantienen inalteradas de manera que reflejen siempre quien fue el que originó inicialmente el datagrama y hacia quien va dirigido.

Una dirección en el Internet está constituida por una palabra de 32 bits y debido a los alcances mundiales que esta red tiene, cada elemento que esté integrado a esta red debe tener una dirección única. Conceptualmente una dirección IP está formada por dos partes: una parte que identifica la red y otra parte que identifica al host dentro de esa red. Las direcciones IP se acostumbra representarlas como cuatro números separados unos de otros por un punto, como por ejemplo 150.186.32.1; cada número representa el valor de cada uno de los octetos que constituyen la palabra de dirección (32 bits, 4 octetos). Los detalles del esquema de direcciones de Internet se verán en el capítulo II.

Los campos identificados en el encabezado como IPOPT y PADD no siempre están presentes (son opcionales) y pueden ocupar mas de una palabra en el

objetivo es permitir especificar algunas opciones que pudieran facilitar entre otras cosas el monitoreo de la red y la detección de problemas funcionales. Las opciones se especifican con un código de opción y existen varias de ellas a disposición de los desarrolladores de aplicaciones, una de las mas conocidas es la que permite hacer un seguimiento de todos los elementos por donde el datagrama ha pasado para alcanzar su destino final; con esta opción el originario del datagrama puede crear una lista vacía de direcciones IP en el encabezado la cual va siendo llenada por cada host por donde el datagrama pase, al final en el destino aparecerán todas las direcciones IP de los elementos por donde pasó el datagrama. La longitud o espacio necesario en el encabezado para las opciones no es fijo y depende del tipo de opción.

El campo de data en el datagrama IP, incluye la data proveniente de la aplicación mas el encabezado agregado en la capa de transporte (depende del protocolo de transporte usado); el protocolo IP no conoce de la organización y estructuración de esa data pues no es de su competencia la interpretación de la misma. Una función muy importante que realiza el protocolo IP además de lo que hemos visto hasta ahora es el enrutamiento, el cual consiste en la determinación de la vía apropiada para liberar el datagrama de manera que sea encaminado a su destino final. Hay que tener presente que un datagrama puede estar dirigido a un host con la cual la red que lo genera no tiene una conexión directa ni un conocimiento específico de su ubicación, por lo cual esta no es una tarea fácil. El protocolo IP se apoya para cumplir esta tarea en una serie de protocolos que mantienen unas tablas llamadas tablas de enrutamiento a través de las cuales el protocolo IP determina la ruta por la que debe encaminar el datagrama. Los detalles de operación de los esquemas de enrutamiento se discutirán en el capitulo II.

1.3.4. Nivel de Enlace (Capa de red)

En esta capa del modelo se encuentran los protocolos que manejan directamente el medio fisico que permite la interconexión entre los elementos de la red. Estos protocolos realmente no forman parte del conjunto de protocolos de

Internet y están por supuesto íntimamente ligados con el medio físico que manejan. Como una característica fundamental de estos protocolos se puede decir que ellos desconocen los esquemas de direccionamiento del Internet, ellos en general manejan otro esquema de direccionamiento (dependiente del hardware) en el cual aparecen las llamadas direcciones físicas. Las direcciones físicas en caso de existir no tienen ninguna relación directa con las direcciones Internet, por lo cual en general es necesario disponer de alguna forma que permita asociar una dirección física con una dirección IP; dependiendo de la interfase que se use en la conexión a la red se proveerán los protocolos necesarios que permiten hacer esta asociación entre ambas direcciones. Obsérvese sin embargo que de la capa internet (capa IP) hacia arriba solo se manejan direcciones IP y de esa capa hacia abajo solo se manejan direcciones físicas dependientes del hardware. Una de las interfaces mas comunes sobre las que se montan los protocolos TCP/IP es la **ethernet,** la cual es una interfase que opera sobre una red tipo bus de amplia difusión (todos los elementos de la red reciben simultáneamente la información colocada en la red por cualquiera de los integrantes, sin embargo, solo el elemento direccionado, mediante una dirección física llamada dirección ethernet, responde a los requerimientos hechos).

En el caso de las interfases ethernet existe un formato específico para la transferencia de datos y el protocolo de la misma define una estructura para el transporte de esos datos (frame), que está constituido por un encabezado y una data. La data en el caso de los protocolos TCP/IP (recuerde que la interfase ethernet y su protocolo pueden ser usados por otros protocolos que no pertenezcan a la familia Internet), la constituye la información proveniente de la capa IP (que como ya se dijo está constituida por el encabezado IP, el encabezado de la capa de transporte y la data de la aplicación), y por supuesto el protocolo de la ethernet desconoce la estructuración de la data que transporta. Básicamente el encabezado ethernet consta de las direcciones fuentes y destinos de las interfases involucradas (direcciones físicas), cada una de estas direcciones están definidas por un número de 48 bits, estipulado por el fabricante de la misma y que es único dada la naturaleza de difusión amplia de este tipo de red (los fabricantes de estas interfases se aseguran que no existirán dos tarjetas

con la misma dirección ethernet no importa quien las fabrique); un identificador del tipo de protocolo al que pertenece la data (la interfase puede prestar servicio en una misma red a distintos protocolos, entre los que podría estar IP de Internet, IPX de Netware Novell, Decnet de Digital etc.); la data propiamente dicha y por último un campo para incluir la información de un checksum que garantice la integridad de la data transportada. La figura 1.7 muestra el esquema o formato de un frame de la interfase ethernet, en la cual se puede apreciar que a diferencia de como se vio en el caso de los protocolos propios de Internet, el checsum no forma parte del encabezado y la data transportada se encuentra entre el encabezado del frame y el checsum. Se insiste en resaltar el hecho de que el protocolo de la interfase ethernet no pertenece a la familia de protocolos de Internet.

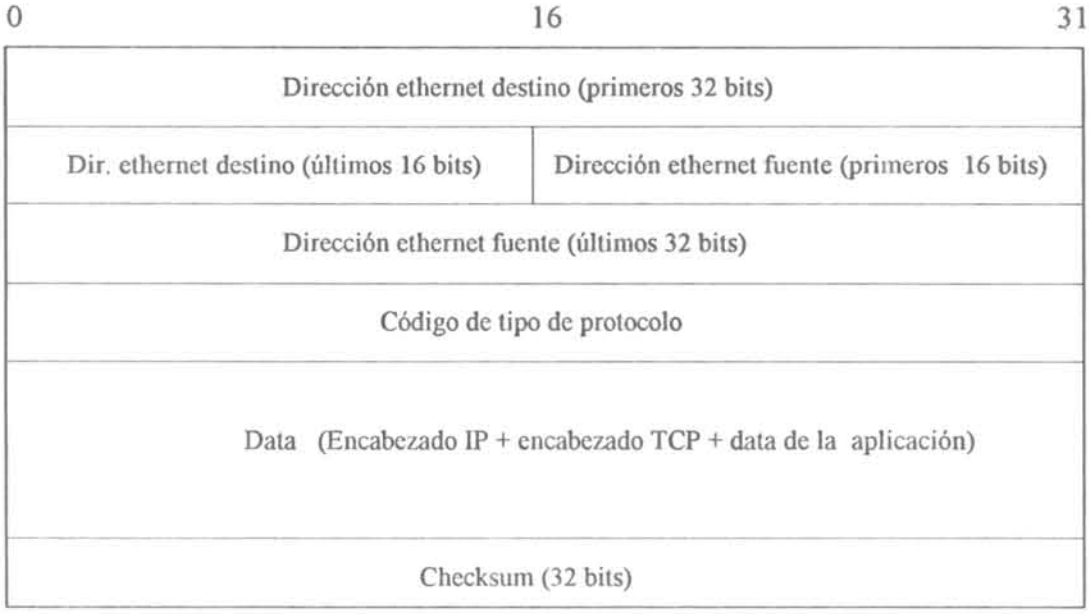


Figura 1.7 Estructura de un bloque de datos (frame) ethernet

1.4. Conexión a nivel de capa de aplicación(proveedores de servicio).

Cuando alguien desarrolla una aplicación, esto es, desarrolla un protocolo a nivel de capa de aplicación, puede hacer abstracción de todos los detalles de la red,

pues los mismos son manejados por las capas inferiores (transporte e internet), sin embargo, hay todavía un detalle por aclarar que terminará de definir la operación del modelo TCP/IP. Si se desarrolla por ejemplo un programa para transferir archivos de una máquina a otra obviamente existirá un programa que interactua con el usuario y acepta los comandos dados por el a través del teclado; esos comandos en definitiva tendrán como destino final un programa(aplicación) en la máquina desde donde se desean transferir los archivos. Este programa debe ser capaz de recibir comandos a través de la red y responder a ellos apropiadamente. El primer programa (el que interactua con el usuario directamente) es llamado el cliente y el segundo programa el servidor.

Los programas en general son procesos los cuales al ser ejecutados se les asigna un número por parte del sistema operativo con el cual identifica la tarea, dicho número no es fijo y es en general aleatoriamente asignado por el sistema operativo y es diferente de ejecución en ejecución. Para identificar las aplicaciones se ha introducido el concepto de puerto, el cual es un número asignado para identificar las aplicaciones y los protocolos de la capa de transporte los usan como un medio para identificar a que aplicación deben entregar el datagrama recibido y de que aplicación proviene el mismo. Desde el punto de vista del programa cliente, carece de importancia el hecho de que entre ejecución y ejecución el número del puerto sea diferente, pues la asociación entre la fuente de los comandos (el terminal) y la aplicación (el programa cliente) es directa. El problema se presenta cuando se desea contactar el programa servidor de la aplicación; ¿Como saber cual número lo identifica en la otra máquina?. Recuerde que cuando se presentó el encabezado de los protocolos de transporte (TCP y UDP), las aplicaciones se identificaban con un número de puerto; de manera que el programa cliente debe conocer ese número para contactar su servidor. Pero ¿como conocerlo si ese número cambia de una máquina a otra o entre una ejecución y otra se le asignan números diferentes?. La solución que se ha dado a este problema es simple y eficiente y consiste en asignar un número de puerto fijo a las aplicaciones servidoras. Ese número de puerto es el que usarán las aplicaciones clientes para contactar las aplicaciones servidoras. En el Internet existen muchas aplicaciones las cuales ya tienen

números de puertos asignados (nótese que estos números tienen que ser únicos de manera de mantener la universalidad de los protocolos), estos números son conocidos como **WKS** (**W**ell **K**nown **S**ockets) y cualquier persona que desarrolle una nueva aplicación debe definir para la misma un número de puerto que identifique al servidor de la aplicación, dicho número no puede coincidir con los ya asignados en el Internet ni con los que se mantienen en reserva para aplicaciones previstas y que aun no están operativas (ver apendice C para una lista de las asignaciones existentes).

Obsérvese que cualquier conexión entre una aplicación cliente y un servidor de esa aplicación viene identificada por un conjunto de cuatro (4) números: los números IP de la fuente y el destino (o sea la dirección IP del host donde se ejecuta el programa cliente y la dirección IP del host donde se ejecuta el programa servidor) y los números de puertos de esos programas (el cliente tendrá cualquier número asignado por el sistema operativo y el servidor tendrá el numero que corresponda a el servidor de esa aplicación, el cual por supuesto es fijo). La conclusión importante de esta sección debe ser el hecho de que en el Internet (modelo TCP/IP) las aplicaciones o programas desarrollados para prestar un servicio a través de la red tienen números de puertos fijos ante el sistema operativo, lo que permite que cualquier programa cliente lo pueda contactar fácilmente a través de la red. El esquema adoptado para este modelo cliente-servidor no es el único posible pero lo que si es cierto es que es extremadamente sencillo y su uso ha sido extendido fuera del ámbito del mundo Internet.

Capitulo II

El diseño de una arquitectura para una red de alcances universales como Internet se sustentó en varios principios que son los que en definitiva le han permitido alcanzar esa aceptación tan amplia que tiene. Es importante conocer esos principios pues ello ayuda a entender muchas de las decisiones de arquitectura y operativas de Internet. En primer lugar se trata por todos los medios de que ni el desarrollador de una aplicación ni el usuario de la misma requiera conocer los detalles de las interconexiones de hardware; en segundo lugar no se quiere forzar el uso de una topología específica; en tercer lugar agregar una nueva red a Internet no debe implicar la conexión a un punto central específico ni tiene que significar tampoco agregar conexiones físicas directas entre la nueva red y todas las existentes; en cuarto lugar debe ser posible enviar datos a redes que no estén directamente conectadas, recurriendo para ello a redes intermedias; en quinto lugar, puesto que se quiere universalidad, se debe proveer un esquema universal de identificación compartido por toda máquina que se conecte a la red; y por último se quiere una independencia tanto de la tecnología usada para la conexión a la red como de la máquina destino. En la próxima sección se discutirá acerca de la arquitectura de Internet.

2.1. Arquitectura de Internet

Desde el punto de vista de un usuario cualquiera que tenga acceso al Internet por algún medio, apreciará la misma como una gran red física a la cual los computadores se conectan tal como se muestra en la figura 2.1, en donde la nube simboliza la red Internet. Los protocolos TCP/IP proporcionan una abstracción de red que oculta los detalles de la red física, sin embargo, es obvio que existe una red física que soporta esta abstracción de red que percibe el usuario y que la misma debe estar

organizada de alguna forma y sustentada en algún elemento fisico que le da esa funcionalidad.

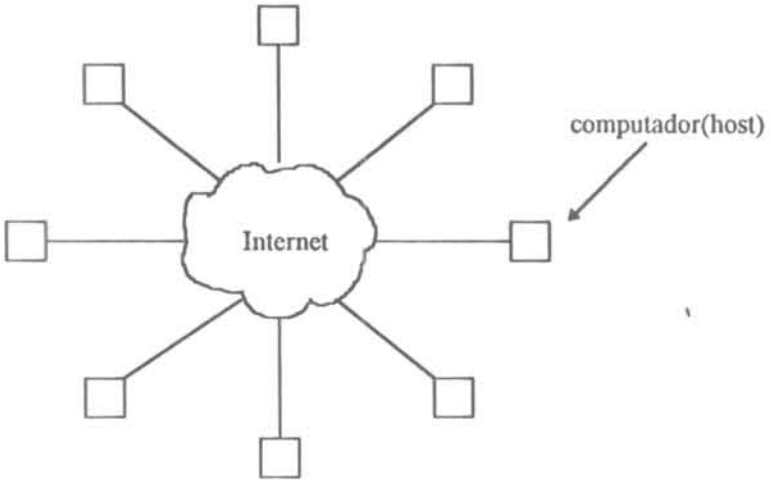


Figura 2.1. Apreciación del Internet que tiene el usuario

El elemento clave sobre el que se sustenta el modelo Internet y a través del cual teóricamente se conectan los distintos elementos que la constituyen es el llamado **enrutador(router)** también conocido como **gateway Internet**. Un enrutador es en esencia un computador dedicado a conectar **redes**. Para entender la función del enrutador supóngase que se desea interconectar dos redes individuales de manera que trabajen cooperativamente. Como primera condición el elemento que conecta las dos redes debe, en principio, estar fisicamente conectado a las dos redes (ver figura 2.2). Es claro que la simple conexión fisica no garantiza que las dos redes estén interconectadas, para que esto suceda, el elemento que las interconecta(el enrutador) debe ser capaz de trasladar información (paquetes) de una red a la otra; en un esquema tan simple como el mostrado en la figura 2.2, el trabajo del enrutador es relativamente sencillo y pudiera darse con varios niveles de inteligencia. En principio, en el caso mas limitado, se pudiera pensar de este elemento enrutador como un dispositivo que coloca todos los paquetes de la red 1 en la red 2 y viceversa, una alternativa poco inteligente. Otra alternativa es que el enrutador tenga la suficiente inteligencia como para saber cuando un paquete producido por algún elemento (host) en la red 1, tiene como

destinatario un elemento en la red 2 y solo en ese caso, lo traslade para la misma; de esta forma solo son trasladados de una red a la otra los paquetes que lo requieran. Obsérvese que con este criterio, el enrutador le bastaría con disponer de una tabla en la cual aparezcan todos los hosts de las redes a los cuales el está fisicamente conectado y la red a la cual pertenecen, de manera que al revisar el destinatario del paquete, el enrutador determina si el paquete requiere su traslado de una a otra red. El tamaño de dicha tabla dependerá de la cantidad de hosts que estén conectados a ambas redes.

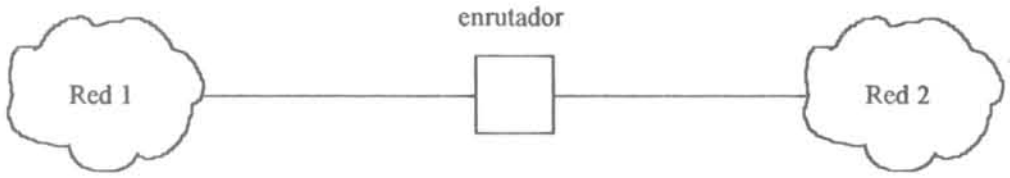


Figura 2.2 Conexión de dos redes a través de un enrutador Internet(R).

En la medida en que el número de interconexiones se hacen mas complejas, la tarea del enrutador se hace mas complicada, sobre todo si se desea que el trabajo que realice sea hecho con cierta inteligencia. Supóngase un esquema de conexión de tres redes como el mostrado en la figura 2.3, en la cual se están usando dos enrutadores R1 y R2 para conectar tres redes (red 1, red 2 y red 3). Con el esquema planteado anteriormente los paquetes en la red 1 destinados a algún elemento de la red 2 (y viceversa), los traslada el enrutador R1; lo mismo ocurre entre la red 2 y red 3, pero en este caso el trabajo de interconexión lo realiza el enrutador R2. Obviamente si se desea que por ejemplo, el enrutador R1 pueda extender su ámbito de acción de manera de que sea capaz de saber que cuando un paquete producido por un elemento de la red 1, vaya destinado a un elemento de la red 3, debe ser enviado primero a la red 2, para que a través de esta llegue a su destinatario en la red 3, el conocimiento que el enrutador debe tener de la red para realizar su labor debe ser extendido mas allá de la simple información que corresponde a las redes a las que está fisicamente conectado.

La solución de la tabla simple, en la cual aparecerían los hosts junto con la red en la que se encuentran, ahora en el caso del ejemplo, debería incluir los hosts de las redes 1, 2 y 3. Es evidente que esta solución con un enrutamiento orientado a los elementos que conforman cada una de las redes(hosts) a las cuales se les desea llegar, conducirá a tablas que pudieran alcanzar gran tamaño en la medida que se aumente el número de redes interconectadas y el número de elementos que cada una de estas redes tenga; adicionalmente al crecer las tablas, el acceso a las mismas para conseguir la información buscada se hará mas ineficiente.

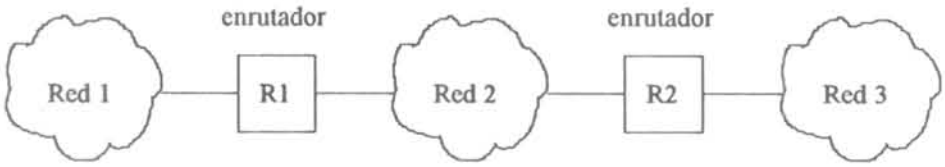


Figura 2.3. Interconexión de tres redes vía enrutadores.

La idea de los enrutadores de Internet, no es lo que se ha planteado hasta ahora (enrutamiento orientado a elementos) sino un enrutamiento orientado a redes, en otras palabras, los enrutadores realizan su trabajo basándose en la red destino y no en el host destino; como una consecuencia inmediata de esta concepción está el hecho de que el tamaño de las tablas para almacenar la información necesaria para realizar su tarea, es proporcional al número de redes y no al número de máquinas (hosts), y en consecuencia, puesto que en una red normalmente hay muchas máquinas, con poca información es posible tener un gran conocimiento del entorno, lo que en definitiva se traduce en eficiencia.

El modelo Internet supone pues que todas las redes físicas se conectan a través de enrutadores; estos realizan su labor (determinar como hacer llegar un paquete a un destinatario) basándose en un esquema de enrutamiento orientado a redes. Es necesario sin embargo, que el enrutador tenga un conocimiento acerca de la topología

de la red que está mas allá de las redes a las cuales el está directamente conectado, pero como se verá mas adelante en la sección 2.5, que trata del enrutamiento, existen protocolos especiales que permiten a los enrutadores aprender ellos mismos acerca de la topología de la red y adaptarse automáticamente a los cambios de la misma. La red Internet que aprecia el usuario (figura 2.1), está en realidad internamente organizada en una forma similar a como muestra la figura 2.4(esta figura solo es ilustrativa del esquema de conexión y no corresponde a la estructura real).

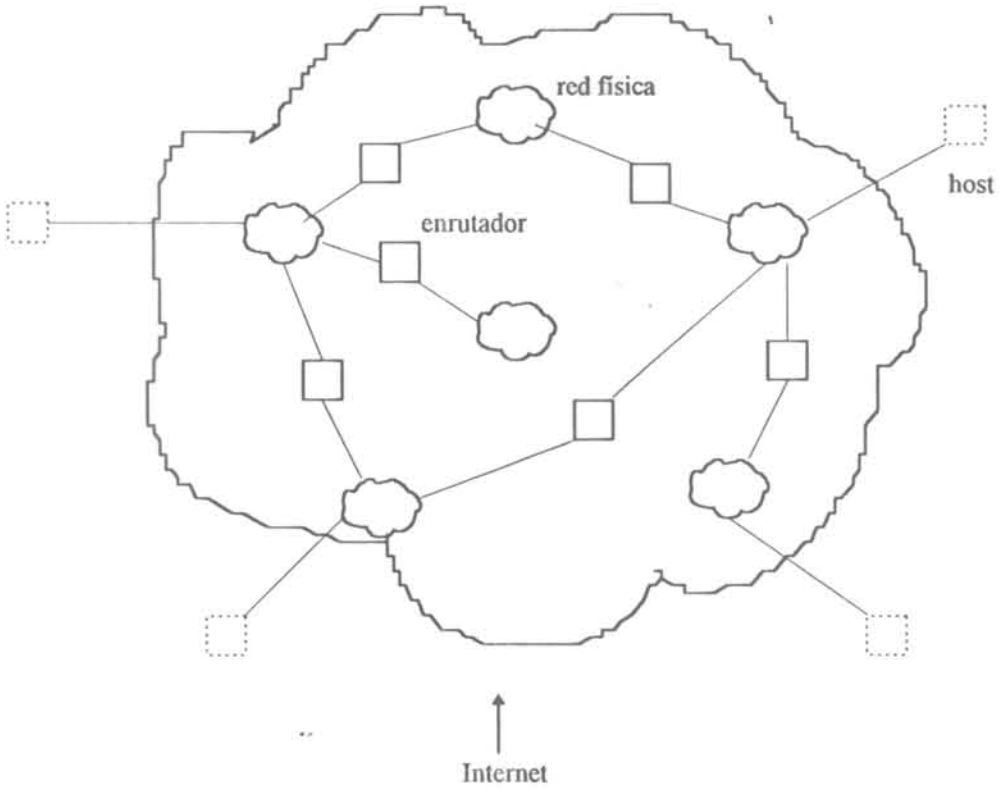


Figura 2.4 Estructura de la red física que provee la interconexión en Internet

2.2. Identificación de elementos en la red

Los elementos que conforman una red deben tener una identificación de manera que puedan ser individualmente contactados para la realización de alguna actividad específica; esta identificación es lo que constituye su dirección. Hay muchos esquemas de direccionamiento, algunos basados en direcciones físicas (determinadas

por un hardware) y otros basados en direcciones lógicas (basadas en el software). En el mundo Internet y en general en cualquier red que use los protocolos TCP/IP de Internet, se usa un esquema de direccionamiento basado exclusivamente en software y cuyo objetivo principal es que la tarea de enrutamiento sea realizada de manera muy eficiente, puesto que fue concebido para la interconexión de redes en donde el enrutamiento es una actividad muy frecuente y fundamental.

Puesto que la idea del Internet es suministrar un servicio de comunicación universal, en donde cualquier host se pueda comunicar con cualquier otro, independientemente de su localización física, el esquema de direccionamiento debe ser globalmente aceptado por todo aquel que se pretenda unir a Internet de manera de garantizar que no existen en la misma dos elementos con la misma dirección. Es necesario aclarar que cualquiera puede instalar una red basada en los protocolos de Internet (TCP/IP) y no estar interconectado a Internet; en estos casos el esquema de direccionamiento aun cuando sigue siendo el mismo usado en el Internet puede, usar cualquier dirección valida según lo exigen los protocolos TCP/IP, sin importar que esa dirección esté asignada a algún equipo de la red Internet; pero si desea conectarse al Internet, entonces no solo debe especificar las direcciones en el formato que definen los protocolos sino también que debe respetar las regulaciones que para tal efecto se han establecido para garantizar la existencia de direcciones únicas de todos los elementos interconectados. La conclusión importante es que la necesidad de identificadores universales lo determina el hecho de interconectarse a una red de alcances universales y no el hecho de usar TCP/IP en redes individuales no interconectadas a Internet.

Una dirección Internet es un número entero de 32 bits, que se asigna a un host y que permite identificarlo ante todos los demás integrantes de la red. Conceptualmente una dirección Internet (y en general una dirección basada en los protocolos TCP/IP) está constituida por un par ordenado (idred, idhost), en donde **idred** identifica la red e **idhost** identifica el host en aquella red. El aspecto importante de este esquema es el hecho de que la dirección no solo está destinada a identificar un

elemento específico en una red, sino también en identificar la red en donde ese elemento se encuentra y como consecuencia de esto se tiene que la dirección Internet realmente no identifica al host en si sino su conexión a una red y por tanto si un host está conectado a dos redes tendrá dos direcciones Internet que lo identifican, una para cada red; esto parece entrar en contradicción con lo que se había venido planteando de la necesidad de identificadores únicos para cada elemento y es si se quiere una de las desventajas que presenta el esquema de direccionamiento adoptado por los protocolos de Internet, sin embargo, como ya se había mencionado su objetivo principal está orientado a facilitar la labor de enrutamiento de manera que se realice en forma muy eficiente, como se verá mas adelante en este mismo capítulo. La conclusión es que con el esquema de direccionamiento de Internet un host tendrá tantas direcciones como conexiones tenga a redes diferentes ya que lo que en realidad se identifica es la conexión a la red y no el host y si un host se traslada fisicamente de una red a otra, su dirección debe ser cambiada de manera que la parte de la dirección que identifica la red refleje el cambio de red.

2.3. Clases de direcciones en Internet

El esquema de direccionamiento de los protocolos TCP/IP contempla la existencia de tres tipos o clases de direcciones que definen tres clases de redes, basándose en la cantidad de hosts que pueden estar conectados a esas redes: Las redes clase A; las redes clase B y las redes clase C, las cuales se describen a continuación.

Redes clase A: Son redes concebidas para tener una gran cantidad de host(mas de 65536 hosts). La dirección Internet de tales redes es lógicamente dividida en tres partes: una parte, que corresponde al bit mas significativo, identifica el tipo de red, en este caso el bit mas significativo en cero determina la clase A. Los próximos 7 bits de la dirección identifican la red, de manera que solo hay 127 posibles redes clase A; los restantes bits de la dirección Internet(24 bits) identifican los hosts dentro de cada una de las posibles redes(ver figura 2.5).

Redes clase B: Son redes concebidas para conectar una cantidad mediana de hosts(entre 256 y 65535 hosts por red). La dirección Internet de tales redes está igualmente dividida en tres partes(ver figura 2.5); los dos bits mas significativos de la dirección, el mas significativo en uno y el siguiente en cero, definen la clase B. Los próximos 14 bits de la dirección se usan para identificar la red, por lo que pueden existir solo 16383 redes clase B; los últimos 16 bits de la dirección Internet, identifican los hosts que pueden existir en cada una de estas redes (hasta 65535 hosts por cada red clase B).

Redes clase C: Son redes concebidas para conectar pocos hosts (menos de 255). La dirección Internet de tales redes está dividida lógicamente en tres partes (ver figura 2.5): Los tres bits mas significativos de la dirección, el mas significativo en uno, el que sigue en uno y el que sigue en cero, definen la clase C. Los siguientes 21 bits de la dirección se usan para identificar la red, por lo que pueden existir 2.097.151 redes clase C diferentes (dos millones noventa y siete mil ciento cincuenta y uno); los últimos 8 bits de la dirección se usan para identificar los hosts dentro de cada una de esas distintas redes (hasta 255 hosts).

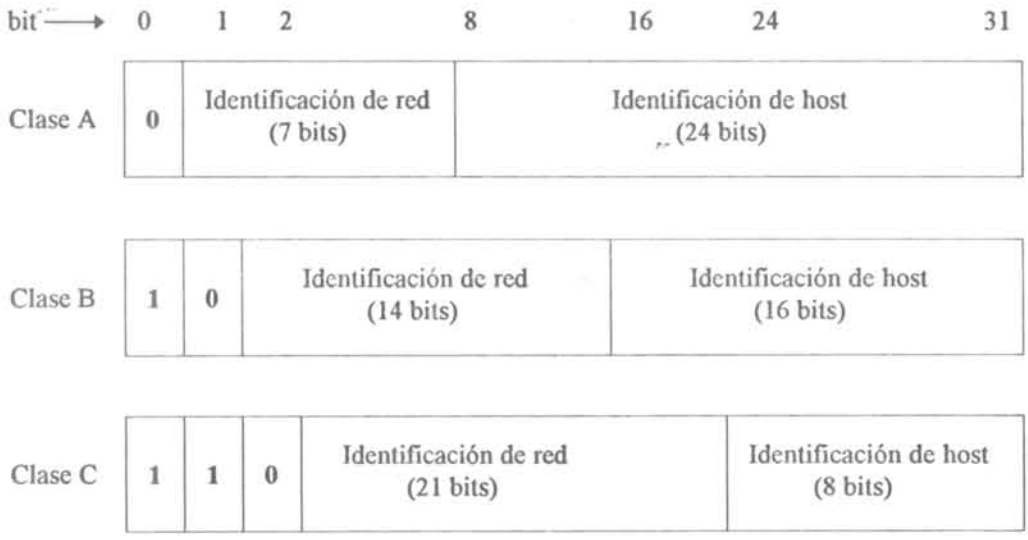


Figura 2.5. Estructuración lógica de una dirección Internet según su clase.

Independientemente de su clase, las direcciones se acostumbra especificarlas como cuatro números enteros, separados unos de otros por un punto; cada número entero corresponde a la interpretación decimal de cada uno de los cuatro bytes (octetos) que conforman la dirección. De manera que una dirección Internet se especifica como 10.32.44.56 (equivale a la red clase A número 10 y al hosts 32.44.56 de esa red); 150.186.32.1 (equivale a la red clase B número 150.186 y al host 32.1 de esa red); 199.87.15.68 (equivale a la red clase C número 199.87.15 y al host 68 de esa red). La siguiente tabla es un resumen que permite identificar las distintas posibles redes de cada clase y el rango de números posibles de hosts en cada una de ellas.

Clase	Identificador de red (primer byte dirección)	Rango de direcciones posibles de la clase (mas baja - mas alta)
A	1 - 127	1.0.0.1 - 127.255.255.254
B	128 - 191	128.0.0.1 - 191.255.255.254
C	192 - 223	192.0.0.1 - 223.255.255.254

Tabla 2.1 Clases de redes y sus rangos de direcciones

2.4 Interpretaciones especiales de algunas direcciones

Hay en el Internet direcciones reservadas para propósitos específicos. Por convención, en una red de cualquier clase, nunca se asigna a un host específico el identificador cero (por ejemplo en una red clase C cuyo número sea 195.23.54, el rango disponibles de direcciones para hosts sería 195.23.54.0 hasta 195.23.54.254, no se le asignará a ningún host el número 195.23.54.0; en una red clase B cuyo número sea 150.186, no se asignará a ningún host específico la dirección 150.186.0.0 y en una red clase A cuyo número sea 10, no se asignará a ningún hosts específico la dirección 10.0.0.0), pues el mismo está reservado para identificar la red en sí. Por convención

entonces se ha establecido que una dirección Internet con todos los bits que corresponden a la parte de host en cero se usa para referirse a la red en si y no a un host particular, esto es un aspecto clave en el establecimiento de subredes, una de las técnicas mas usadas para reorganizar la distribución de los bits de direcciones establecidas para las tres clases fundamentales de redes(A, B y C) y que será estudiada mas adelante en este mismo capítulo.

En muchos casos se desea hacer un requerimiento que sea atendido por todos los hosts conectados a una red particular; los protocolos Internet han previsto por convención que una dirección Internet valida, que contenga puros unos(1's) en los bits que corresponde a la identificación del host, es una dirección especial de amplia difusión (llamada dirección broadcast) que se refiere a todos los hosts de la red especificada y cualquier paquete con esa dirección, será procesado por cada uno de los hosts conectados a la misma. Por ejemplo, la dirección broadcast de la red 150.186 será 150.186.255.255; la de la red 195.23.54 será 195.23.54.255 y la de la red 10 será 10.255.255.255. Esto como se verá mas adelante puede cambiar cuando se use el esquema de subredes, sin embargo, el concepto fundamental de la dirección broadcast, o sea una dirección de red valida con todos los bits que corresponden a la identificación del host en uno, se mantiene con en el esquema de subredes.

En general en Internet se ha adoptado la filosofia de que si en un campo de la dirección aparecen puros unos (bien sea en la parte de hosts o de red) se refiere a todos (todas las redes o todos los hosts, según sea el caso) y si en un campo aparecen todos los bits en cero (bien sea en la parte de red o hosts) se refiere a esa red o a ese hosts en particular según sea el caso. Por otra parte, los protocolos Internet tratan por todos los medios de restringir la extensión de las direcciones de amplia difusión (broadcast) al menor número de máquinas posibles.

La dirección Internet consistente de puros unos en todos los campos (red y hosts), que corresponde a 255.255.255.255 es usada como una dirección dirigida a todos los hosts de la red en la cual se está produciendo el paquete con esa dirección, o

sea es una dirección similar a la broadcast que se vio anteriormente, pero de alcances limitados, por lo cual se le llama dirección broadcast limitada, obsérvese que en este caso no existe una dirección de red valida, de manera que el emisor del paquete no necesita conocer el número de la red en la que se encuentra para lanzar a la red un paquete de amplia difusión (todos los hosts) y esto tiene su utilidad en algunas aplicaciones y protocolos.

Otra dirección reservada para uso especial es la dirección 127.0.0.0 (llamada dirección *loopback*) designada para pruebas y comunicación entre procesos sobre la máquina local. Si un programa usa la dirección *loopback* para enviar los paquetes, el software de la red entiende que ese paquete está dirigido a un proceso que se está ejecutando (o que se va a ejecutar) en la misma máquina y no envía ese paquete a la red sino que se lo pasa internamente al proceso al cual está dirigido(comunicación entre procesos locales). De manera que en la práctica esto significa que en una red nunca aparecerá un paquete dirigido a la red 127 y los protocolos en los enrutadores nunca propagan información referente a esta red pues la misma no es considerada una red valida. Generalmente se usa como dirección loopback 127.0.0.1, pero en realidad cualquier dirección de host que aparezca acompañando la dirección de red 127 es considerada como una dirección loopback, o sea dirigida a un proceso ejecutado localmente.

2.5. Otros esquemas de direccionamiento (clases secundarias)

El esquema de direccionamiento de Internet, en el cual está prevista la existencia de tres clases de redes, dependiendo de la cantidad de hosts, tiene como propósito fundamental el manejo eficiente del enrutamiento, lo cual se consigue en parte, minimizando la cantidad de información requerida por los enrutadores para tener un conocimiento de la topología de la red. Esa es la razón fundamental de haber incluido como parte de la dirección Internet una porción que identifica la red, lo que se traduce en que todos los hosts pertenecientes a una misma red física comparten un prefijo común que corresponde a la identificación de la red física a la que están

conectados y para enrutar información a cualquiera de esos hosts basta con conocer la red en la cual se encuentran (enrutamiento orientado a red).

Para la época en la que se inicia el diseño de la Internet, la idea prevalente era la de unas pocas redes(decenas de ellas), girando alrededor de grandes computadores(costosos mainframes) con cientos de hosts conectados a esas grandes redes; el mundo actual no está configurado de esa forma sino que está constituido por decenas de miles de pequeñas redes, constituidas básicamente por computadores personales y una pequeña estación de trabajo o servidor proveyendo servicios básicos. Si bien es cierto que con la clase C, se prevé en Internet la existencia de una gran cantidad de estas pequeñas redes (menos de 255 hosts), no es menos cierto que la aparición o puesta en operación de esta gran cantidad de pequeñas redes le quitarían en parte una de las grandes bondades al esquema de direccionamiento del Internet, pues mientras mas redes distintas existan, mas grandes serán las tablas de los elementos enrutadores y la idea era precisamente mantener esas tablas pequeñas(es obvio que el esquema sigue siendo mas eficiente que si el direccionamiento estuviese orientado a hosts en vez de redes).

Por otra parte, en general una misma empresa o institución tiene varios departamentos o núcleos que pueden estar geográficamente dispersos y que poseen sus propias redes locales; si esa empresa o institución se conecta al Internet, con el esquema de direccionamiento planteado, cada una de estas redes deberá tener un número de red distinto(un número por cada red fisica), lo que se reflejará en los elementos enrutadores como una entrada mas en su tabla de enrutamento, lo cual no es deseable, sobre todo si se considera que corresponden a la misma empresa o institución. En definitiva , en tanto sea posible es deseable mantener el menor número de identificadores de redes distintos, porque esto contribuye a mantener tablas de enrutamiento pequeñas, lo cual no solo es importante por lo que se ha venido diciendo de la eficiencia en el acceso, sino también porque como se verá mas adelante, los enrutadores en general intercambian información con los enrutadores adyacentes, de manera de conocer la topología de la red que está mas allá de las redes a las cuales el

está directamente conectado y esta información es actualizada cada cierto tiempo; mientras mayor sea el número de redes, mayor será la información que hay que intercambiar y mayor será el tráfico en la red con el solo propósito de mantener información para el enrutamiento, lo cual por supuesto va en detrimento de la transferencia de información para las aplicaciones destinadas a proporcionar servicios a los usuarios de la red.

Se han propuesto y puesto en operación esquemas alternos de direccionamiento, tendientes a minimizar el número de identificadores de redes, manteniendo todavía el esquema original de clases; la idea de todos estos nuevos esquemas es que un mismo prefijo o identificador de red pueda ser compartido por múltiples redes fisicas(lo cual no es posible en el esquema original), pero en una forma tal que desde el punto de vista del mundo exterior (resto de las redes interconectadas a Internet), sigue apareciendo como una sola red. La implantación de estos nuevos esquemas ha requerido la modificación de algunos algoritmos, en especial los encargados del enrutamiento, de manera de manejar apropiadamente estas extensiones al esquema original de direcciones de Internet. En las próximas secciones se presentarán tres de estos esquemas; el último de ellos (subredes), se ha convertido en un estándar que permite redefinir la interpretación de la estructura de direccionamiento de Internet. Estos tres esquemas son: Uso de enrutadores transparentes, uso del protocolo Proxy ARP y por último el esquema de sub redes.

2.5.1. Enrutadores o gateways transparentes

La idea de este esquema es multiplexar varias conexiones a través de un solo puerto(un solo punto de conexión a la red). Su funcionamiento está basado en un enrutador especial (conocido como enrutador transparente o gateway transparente); este enrutador tiene una interfase conectada a la red Internet (o cualquier red de área amplia, WAN) y una interfase conectada a una red de área local, LAN. La presencia de este enrutador especial no es percibida por los elementos de la red WAN y de hecho, ellos desconocen su existencia ya que su operación es tal que para los efectos

de esa red (WAN), los hosts que están en la red local aparecen como si estuvieran conectados directamente a la red WAN. El enrutador se encarga de demultiplexar los datagramas provenientes de la red WAN y se los hace llegar al host apropiado en la red local, usando para ello una tabla de direcciones o por decodificación de alguna parte de la dirección IP que no se use. Con el esquema del enrutador transparente, la red local no tiene su propio prefijo de red (identificador de red fisica) ya que usa el mismo prefijo de red que el de la red WAN. La figura 2.6 ilustra el esquema de conexión de una red local LAN con una red WAN mediante el enrutador transparente; obsérvese que el enrutador no tiene dirección Internet, en ninguno de sus puertos(ni el de la WAN ni el de la LAN) y que las direcciones Internet de los hosts (en este caso se refiere a una red clase A cuyo identificador de red es 15) tienen todas el mismo identificador de red (15) por lo cual lucen como si estuviesen colocadas en la misma red fisica a pesar de que están conectados a una red fisicamente independiente.

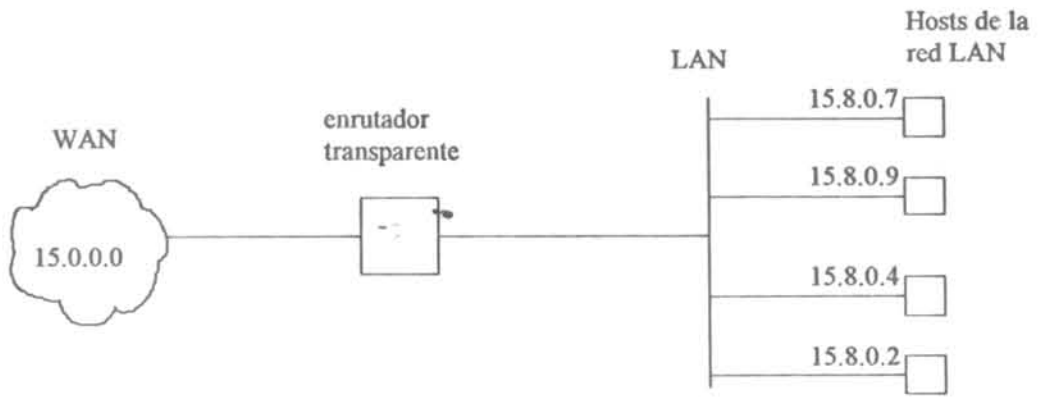


Figura 2.6 Esquema de conexión mediante enrutador transparente
(los números de los hosts son solo para ilustración)

2.5.2 Protocolo *Proxy ARP*

Este es otro de los enfoques que permiten asignar un mismo identificador de red a múltiples redes fisicas; se puede usar solamente en redes que usen el protocolo ARP (Address Resolution Protocol), el cual es un protocolo desarrollado a nivel de capa fisica, usado comúnmente en las redes ethernet con la finalidad de determinar la

asociación entre una dirección Internet (software) y una dirección fisica (la dirección fisica de la tarjeta ethernet, por ejemplo). Cualquier datagrama generado en una capa superior (aplicación, transporte o internet), usa como ya se ha explicado direcciones Internet para especificar las fuentes y los destinos de esos datagramas; sin embargo, cuando esos datagramas van a ser enviados al medio fisico en el que están conectadas todas las interfaces ethernet de los hosts de la red(cable coaxial, por ejemplo), las cuales entienden solo direcciones fisicas (única para cada interfase), es necesario determinar que dirección fisica tiene el hosts cuya dirección Internet aparece en el datagrama y es allí donde entra en operación el protocolo ARP.

Básicamente el protocolo ARP funciona de la siguiente forma: Cuando la capa fisica necesita conocer que dirección fisica tiene un host con una determinada dirección Internet, envía un requerimiento ARP, en un formato específico que todas las interfaces entienden, y que traducido al lenguaje normal esencialmente dice "necesito la dirección ethernet del host cuya dirección IP es <Dir IP>"; todas las interfases oyen el mensaje (ethernet es una red de difusión amplia), pero solo la interfase cuya dirección Internet aparece en el requerimiento responde al mismo;usa para ello un formato preestablecido en el protocolo que traducido al lenguaje normal dice "<Dir IP> esta asociado con la interfase <Dir Ethernet>". El protocolo ARP se encarga adicionalmente de guardar en unas tablas las respuestas obtenidas de manera que la próxima vez que tenga que referirse al mismo host ya sabe cual es la dirección fisica que debe usar.

El esquema Proxy ARP, es una extensión del protocolo ARP para permitir, mediante un pequeño truco, que dos redes fisicamente diferentes compartan el mismo identificador de red de manera que los hosts en estas redes sean vistos desde el punto de vista de los protocolos Internet, como que si estuviesen en una misma red. El truco consiste en interconectar las redes mediante un enrutador que en vez de ejecutar el protocolo estándar ARP, ejecute Proxy ARP.

El enrutador que usa proxy ARP, tiene una interfase ethernet en cada una de las redes fisicas que interconecta; cada interfase tiene una dirección física y una dirección Internet asociadas con ella. Como parte de la operación rutinaria del protocolo ARP, el enrutador proxy recibe, como todos los elementos conectados a las respectivas redes, los requerimientos ARP de cualquiera de los elementos y cuando detecta que un requerimiento ARP en una de las redes va dirigido a un host en la otra red (ese host no puede responder porque fisicamente está en lotra red y no le llega la petición ARP, pues los requerimientos ARP, tienen difusión local; al enrutador le llegan las peticiones de cualquiera de las redes porque el tiene interfase en ambas redes), responde diciendo que el es ese host, de manera que cuando el host que hizo el requerimiento ARP recibe la respuesta le envía la información al enrutador que ejecuta el Proxy ARP, y este al recibirlo lo traslada a la otra red, previa identificación del verdadero host en la otra red (usando un requerimiento ARP o si ya lo tiene en las tablas, usando la información almacenada en las mismas).

Desde el punto de vista de los hosts de la red, la única cosa que les parecería como extraña es el hecho de que puesto que el enrutador que está ejecutando proxy ARP responde todas las peticiones ARP de los hosts que están en la otra red, en las tablas que lleva el protocolo ARP de cada host, aparecerá que una misma dirección ethernet (dirección fisica) le corresponde a varias direcciones Internet, sin embargo, el protocolo ARP no prohibe esto y por tanto normalmente no hay problema; en algunas implementaciones de ARP, por razones de seguridad prohiben que una misma dirección fisica tenga varias direcciones Internet asociadas, en estos casos este esquema no puede ser usado.

La figura 2.7, ilustra el esquema proxy ARP(los números de redes son ilustrativos solamente); en dicha figura el enrutador R debe estar ejecutando proxy ARP; en el resto de los hosts se ejecuta el estándar ARP, obsérvese que en las tablas ARP de los hosts H2 y H3 los hosts (o interfases) que pertenecen a la otra red aparecen con la misma dirección fisica (la dirección fisica de la interfase del enrutador en esa red).

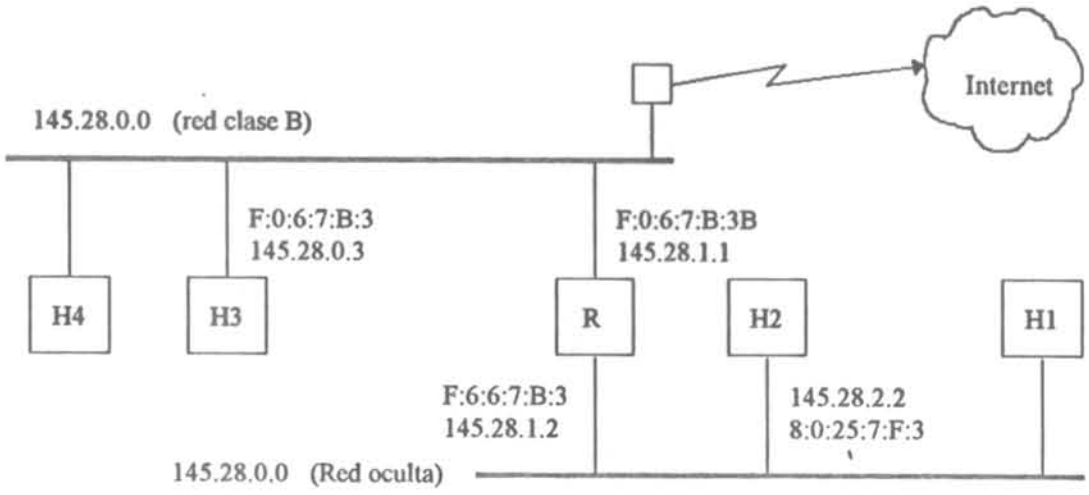


Tabla ARP (H3)

Dir IP	Dir Ethernet
145.28.1.1	F:0:6:7:B:3B
145.28.1.2	F:0:6:7:B:3B
145.28.2.2	F:0:6:7:B:3B
...	

Tabla ARP (H2)

Dir IP	Dir Ethernet
145.28.1.1	F:6:6:7:B:3
145.28.1.2	F:6:6:7:B:3
145.28.0.2	F:6:6:7:B:3
...	

Figura 2.7. Esquema proxy para extensión de los esquemas de direccionamiento de Internet

2.5.3. Subredes

Esta es la técnica mas ampliamente usada para la extensión del esquema de direccionamiento del Internet y se puede considerar ya como un estándar en el mundo Internet. El objetivo de este esquema persigue el mismo propósito de los vistos anteriormente, evitar que el número de redes fisicas manejados por los enrutadores de Internet aumenten desproporcionadamente. El esquema de subredes puede ser completamente implementado en software y representa un cambio muy sutil del esquema original del Internet. Conceptualmente el esquema de subredes lo que hace es un cambio en la forma de interpretación de los 32 bits que corresponden a la dirección Internet, la cual aparece dividida en dos porciones; una porción que corresponde a la identificación de la red y otra porción que corresponde a la identificación del host dentro de esa red; cada clase de red define la cantidad de bits de la dirección que

corresponden a identificación de red y la que corresponde a identificación de host. El esquema de subredes permite redefinir esta interpretación, pero desde un punto de vista local esto es, esta reinterpretación no trasciende al mundo Internet. La gran aceptación del esquema de subredes se debe a su flexibilidad y al hecho de que mantiene una completa compatibilidad con el sistema básico del Internet. Con el esquema de subredes los 32 bits de la dirección se consideran divididos en una porción Internet, que corresponde a lo que el esquema básico simple identifica como red; y una porción local, que corresponde a lo que el esquema básico identifica como host, pero que ahora con el esquema de subredes será reinterpretado localmente de manera que una parte identifique la red fisica y la otra los hosts dentro de esas diferentes redes fisicas, tal como muestra la figura 2.8.

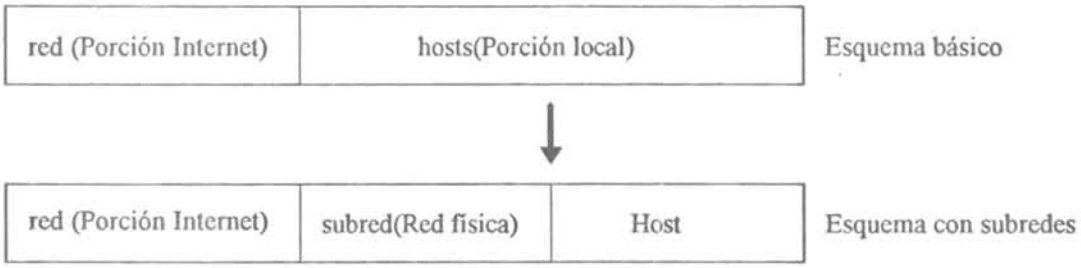


Figura 2.8. Interpretación conceptual de las subredes

La implementación de las subredes es relativamente sencilla, y básicamente consiste en escoger como reinterpretar la porción que corresponde al identificador del host en el esquema básico; dicho de otra forma, definir la forma en la que se interpretará la porción local de manera que permita definir que bits se usarán para identificar las diferentes redes fisicas (subredes) y que bits se usarán para identificar los hosts que pertenecen a cada una de ellas. El mecanismo consiste en definir una palabra de 32 bits, llamada mascara, cuyos bits definirán la interpretación que se le dará a la porción local; todo bit que se coloque en uno (1) en la mascara, se considerará como parte de la identificación de la red fisica (subred) y todo el que aparezcan en cero (0) se considera como parte de la identificación del host. El algoritmo que usan los protocolos para extraer la porción efectiva que identifica la red (Internet + subred) y

el host, es extremadamente simple; para extraer la porción que identifica la red basta con realizar un AND entre la mascara y la dirección Internet asignada a la interfase; para extraer la porción que identifica el host, basta con hacer un Xor (or exclusivo) entre el resultado del AND anterior y la dirección Internet asignada a la interfase. Esto se ilustra mejor con un ejemplo: supóngase que se tiene una red clase B asignada identificada con el número 150.186 (dirección internet); en el esquema básico los otros 16 bits de la dirección identifican el host; obsérvese que esto equivale a decir que el esquema básico para una red clase B define por defecto una mascara de 255.255.0.0 (recuerde la convención un 1 en la mascara define la parte que se interpretará como red y un cero la que se interpretará como host). Si se quieren reinterpretar esos 16 bits de manera que solo los últimos 6 bits de la dirección Internet sirvan para identificar hosts, eso se expresaría con una submascara de 255.255.255.224. Si un host perteneciente a una de estas redes se le asigna la dirección 150.186.32.5, con la mascara anterior se puede determinar que ese host pertenece a la red física 150.186.32.0, subred 32, (255.255.255.224 AND 150.186.32.5) y es el host número 5 de esa red física (150.186.32.0 XOR 150.186.32.5), tal como se detalla a continuación:

IP host:	150.186.032.005	10010110.10111011.00100000.00000101	
mascara:	255.255.255.224	11111111.11111111.11111111.11100000	← And
red física:	**150.186.032.000**	**10010110.10111011.00100000.00000000**	
mascara :	255.255.255.224	11111111.11111111.11111111.11100000	← Xor
Host	**: 000.000.000.005**	**00000000.00000000.00000000.00000101**	

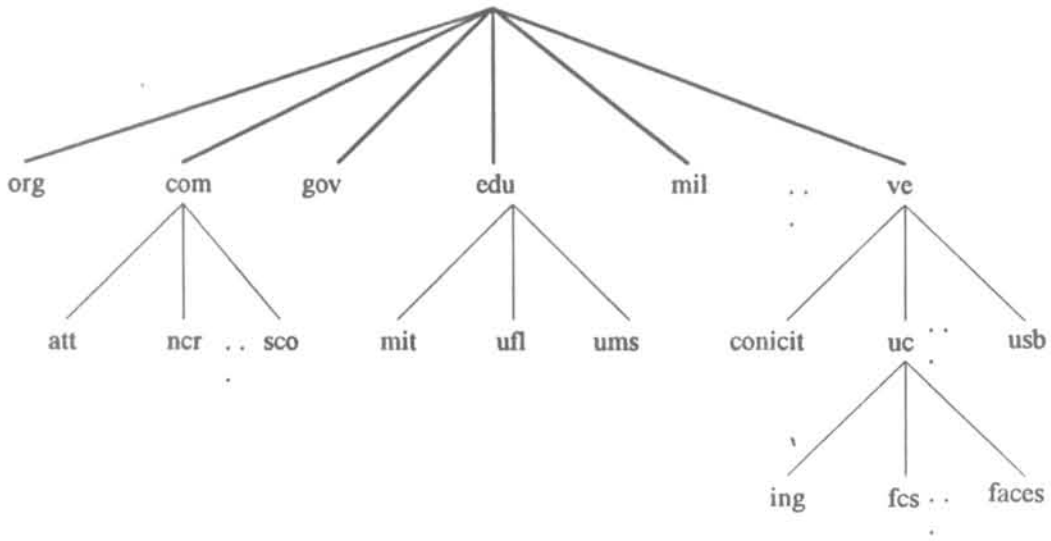
Obsérvese en el ejemplo anterior que a pesar de que el número de red corresponde a una red clase B, en la cual solo los primeros 16 bits de la dirección se usan para identificar la red, al definir una submascara de 255.255.255.224, ahora en vez de 16 bits para identificar las redes se tienen 27 bits para ello y solo 6 bits para los hosts, de manera que se podrían tener hasta 2047 redes físicas distintas (subredes), cada una de las cuales puede tener hasta 30 hosts(recuerde que hay algunas direcciones con uso especial que normalmente no se le asignan a ningún host); sin embargo, esta es una interpretación que no trasciende al resto del internet sino que es

acceder a los servicios prestados por el host que especifica la dirección IP; sin embargo, a las personas les es difícil recordar números y en general prefieren los nombres, sobre todo si los mismos tienen alguna característica o condición que lo asocie con algo conocido; por esta razón en el Internet se ha creado un servicio conocido como Sistema de Dominio de Nombre (**D**omain **N**ame **S**ystem, **DNS**), el cual en forma sencilla se puede decir que es un sistema que permite trasladar un nombre que los humanos pueden recordar y pronunciar fácilmente, en una dirección IP (número) que necesitan las aplicaciones para comunicarse. Realmente el sistema es mucho mas complejo; el DNS es una compleja base de datos distribuida que además de contener registros que permiten asociar un nombre con una dirección, poseen registros que especifican los servicios que un determinado host está en capacidad de prestar, permite definir alias para los hosts, almacenar información acerca de usuarios, listas de correo u otros objetos. Esta información puede ser consultada para un host específico en forma parcial o total. Existe un estándar en Internet que define la operación del DNS así como también de los protocolos usados para hacer las consultas a esta base de datos.

El DNS está organizado jerárquicamente en forma de una estructura de árbol invertido similar al del sistema de archivos; opera en una forma descentralizada. La idea es organizar el espacio de nombres en áreas lógicas llamados dominios, en cada dominio hay una autoridad que conoce de ese dominio (conoce los nombres y números de los servidores de los subdominios dependientes de él ó sabe los nombres y los números asociados con las máquinas de ese dominio); cada dominio es particionado por algún criterio, por ejemplo el dominio de una Universidad se podría pensar en particionarlo en Facultades; en las Facultades existen Escuelas y estas podrían constituir otro nivel; a su vez las Escuelas tienen departamentos que podrían constituir otro nivel. Cada nivel constituye un subdominio en el cual debe haber alguien que sabe acerca de los nombres de ese subnivel; solo cuando en un nivel determinado no es posible resolver una consulta se recurrirá al nivel superior, que tiene un conocimiento mas amplio de todo el dominio que está por debajo de el. Siguiendo con el ejemplo de

la Universidad supóngase que la Universidad de Carabobo tiene un dominio identificado como **uc**; este constituiría el nivel de mayor jerarquía que conoce acerca de toda la Universidad. Si el dominio **uc** se particiona por Facultades, a cada Facultad se le creará un subdominio, el cual tendrá conocimiento acerca de todas las máquinas de esa Facultad. Supóngase por ejemplo que para Ingeniería se escoge **ing**, para Ciencias Económicas, **faces**, etc.; estos subdominios constituyen el segundo nivel. Si dentro de la Facultad de Ingeniería cada Escuela es a su vez considerada un subdominio podría agregarse otro nivel constituido por las Escuelas de Eléctrica, Mecánica, Química, Civil e Industrial (**ee, em, eq, ec e ei** respectivamente). Con este esquema descrito habrá una autoridad que conoce acerca de las máquinas de la Escuela de Eléctrica (**ee.ing.uc**) otra que conoce acerca de las de la Escuela de Mecánica (**em.ing.uc**) y así sucesivamente a nivel de todas las Escuelas. El nivel anterior con un conocimiento mas amplio en este caso estaría constituido por el dominio de la Facultad (**ing.uc**); este dominio tiene conocimiento acerca de todos los servidores de nombres que hay en la Facultad de Ingeniería y por último a un nivel mayor estaría el dominio de la Universidad (**uc**), el cual tendrá conocimiento acerca de todos los servidores de nombre que hay en el dominio uc. En el ejemplo que se acaba de presentar se ha introducido sin decirlo explícitamente la sintaxis que se usa para especificar los dominios y el nivel de autoridad que cada uno de los dominios tiene: los dominios se identifican con nombres que constituyen a su vez niveles de autoridad; cada nivel está separado del otro por un punto (.) y la mayor autoridad corresponde al nivel que se encuentre mas a la derecha.

El Internet ha escogido un esquema de jerarquización que permite realmente dos arboles de jerarquía completamente diferentes; uno basado en la actividad organizacional y el otro basado en la ubicación geográfica. En general dentro de los Estados Unidos se sigue el dominio jerarquizado por actividad organizacional y fuera de los Estados Unidos se acostumbra a usar el de ubicación geográfica. La figura 2.10 muestra la jerarquización de los dominios en el Internet, encontrándose en el tope de la autoridad el dominio llamado raíz que tiene autoridad sobre todo el Internet.



Leyenda:			
com:	Organizaciones comerciales	ve:	Dominio Venezuela
org:	Otras organizaciones	uc.ve:	Dominio UC Venezuela
edu:	Instituciones educativas	ing.uc.ve:	Dominio Ingeniería UC Venezuela
gov:	Instituciones del gobierno	ufl.edu:	Universidad de Florida educación
mil:	Grupos militares	uml.edu:	Universidad de Lowell educación

Figura 2.10 Jerarquización de los dominios en Internet

El sistema de dominio de nombres es distribuido, lo que significa que existen un conjunto de servidores (llamados servidores de nombres) que trabajan cooperativamente para resolver las consultas planteadas. El mecanismo es muy eficiente en varios aspectos, en principio la mayoría de las consultas se resuelven localmente (usando el servidor de nombre local) y cuando es necesario recurrir a un nivel superior la respuesta obtenida es "recordada" de manera que si se le vuelve en otro momento a plantear la misma consulta, usa el resultado obtenido en alguna consulta previa (estas respuestas tienen en general un tiempo de vigencia, que el protocolo DNS permite especificar). La explicación de la operación del sistema de dominio de nombres se puede entender fácilmente si se considera la estructura organizativa de la figura 2.10. El servidor en el tope (raíz) conoce cuales servidores de nombre existen para cada uno de los dominios topes(edu, com, org, gov, mil, ve, etc.). Los servidores a este nivel conocen que servidores están en capacidad de resolver las consultas de los subdominios que dependen de el (en el ejemplo de la figura 2.10, mit,

ufl,edu ó conicit, usb y uc). El árbol puede seguir extendiéndose creándose subdominios adicionales o sencillamente a este nivel los servidores estarán en capacidad de resolver las consultas, proveyendo la traslación de nombres a números. Es necesario aclarar que los servidores de un dominio no tienen necesariamente que estar ubicados fisicamente en la localización geográfica de el dominio al que sirven.

2.7. Enrutamiento

El enrutamiento es el proceso que permite escoger una vía sobre la cual enviar los datagramas dirigidos a un host. El enrutamiento es realizado por un host que se conoce con el nombre de enrutador o gateway. Todo los datagramas son pasados de una red a otra por estos enrutadores o gateways, cuando un datagrama es dirigido desde una red local a una red remota. El datagrama va pasando de enrutador en enrutador hasta alcanzar su destino final. En cada enrutador se ejecuta un protocolo cuyo propósito es determinar, basándose en la dirección destino del datagrama, a quien debe dirigir el mismo de manera de encaminarlo hacia la vía en donde eventualmente encontrará su destino. Existen muchos protocolos desarrollados para cumplir la labor de enrutamiento, antes de revisar algunos de los mas usados es necesario comprender el modelo sobre el cual Internet (y especialmente el protocolo IP) está basado.

En primer lugar el protocolo IP (capa 2), supone que los hosts están conectados a alguna red local y ese host puede enviar datagramas a cualquier host que se encuentre en la misma red local sin necesidad de ningún algoritmo de enrutamiento; pero si el datagrama va dirigido a una red distinta a la red local el datagrama debe ser enviado a un enrutador o gateway, el cual no es mas que un computador que tiene conexiones con dos o mas redes, en el cual se encuentran los protocolos que permiten realizar las labores de enrutamiento. El enrutamiento está basado exclusivamente en el número de red del destino (no en el host); cada hosts de la red tiene una tabla(llamada tabla de enrutamiento IP), en la cual aparecen los números de las redes que el conoce y para cada una de estas redes, se le asocia un enrutador o gateway que permite

alcanzarla. Cuando un host quiere enviar un datagrama, primero chequea a ver si la dirección destino pertenece a la misma red local en donde el se encuentra, si es así se lo envía directamente. Si ese no es el caso el consulta la tabla de redes para ver a que enrutador el debe enviar el datagrama de manera que tome la vía para alcanzar el destino final. Las tablas de enrutamiento se pueden hacer muy grandes (a pesar de que como se ha dicho el enrutamiento es por redes y no por hosts), por lo cual como una alternativa en vez de especificar una ruta para cada red se acostumbra especificar una ruta por defecto, la cual no es mas que una ruta que será usada cuando no se tenga información precisa de como alcanzar una red determinada; es posible inclusive especificar solo una ruta por defecto lo cual en general funciona bien cuando solo existe una posible vía de conexión al resto del Internet. En los enrutadores los protocolos que se encargan del enrutamiento en general se encargan de mantener actualizadas las tablas de enrutamiento, para lo cual intercambian información con los enrutadores con los que se interconectan y de esa forma van "aprendiendo" la forma de llegar a las diferentes redes del Internet, existen varios protocolos que se encargan de llevar esta actualización de las tablas en una forma consistente que se revisaran brevemente a continuación.

Existen básicamente dos enfoques para la propagación de información referente al enrutamiento: uno basado en el vector-distancia y el otro basado en los algoritmos conocidos como de mas corta vía primero (Shortest Path First, **SPF**). En el caso del vector distancia la idea es muy simple; se asume que todos los enrutadores comienzan con una tabla de enrutamiento básica en las cuales aparecen todas las redes a las cuales el se conecta; cada entrada identifica una red y una distancia a esa red medida en saltos. Las redes a las cuales el enrutador está directamente conectado tienen una distancia 0. Periódicamente cada enrutador envía sus tablas de enrutamiento a cualquier otro enrutador que sea directamente alcanzable; cuando un enrutador **X** recibe información proveniente de un enrutador **Y**, la examina para ver las redes destinos que incluye y las distancias a cada una de ellas, si **Y** conoce una vía mas corta para alcanzar una determinada red o si el reporte de **Y** incluye una red para la cual **X** no tenia ruta, el enrutador **X** actualiza su tabla con la información enviada por **Y** para

esas redes. La información enviada por cada enrutador básicamente consiste de un destino (llamado el vector) y la distancia a ese destino desde el punto de vista del enrutador que envía el reporte; la tabla que lleva cada enrutador consiste de un destino, una distancia y la ruta a seguir (la cual es precisamente el enrutador que envía el reporte). El problema con los algoritmos basados en este enfoque es que la información a intercambiar crece a medida que aumenta el número de redes, por lo cual se ha propuesto la alternativa de los algoritmos basados en SPF. Estos algoritmos en vez de enviar reportes que contengan las listas de destinos y sus distancias hacen dos cosas: primero chequean el estatus de todos los enrutadores vecinos (entendiendo que dos enrutadores son vecinos si ellos se conectan a una red común); en segundo lugar la información referente al estatus del enlace se propaga periódicamente a todos los otros enrutadores. El cálculo de las rutas es hecho localmente por cada enrutador y no se propaga a los otros enrutadores pues cada uno de ellos los determina independientemente basado en la información de los estatus de los enlaces.

La complejidad del enrutamiento hace ineficiente cualquier esquema que pretenda centralizar dichas labores, pero por otra parte la difusión amplia no garantiza la consistencia de la información que se propaga, por lo cual se ha introducido el concepto de sistema autónomo y se han establecidos protocolos que permiten regular en primer lugar la información que se intercambiará entre estos sistemas autónomos y en segundo lugar la información que se intercambiará dentro de estos sistemas autónomos, dando origen a los llamados protocolos EGP (Exterior Gateway Protocol), que son protolos para intercambio de información de enrutamiento entre sistemas autónomos e IGP (Interior Gateway Protocol), que son protocolos para el intercambio de información de enrutamiento dentro de un sistema autónomo.

Un sistema autónomo no es mas que un grupo de redes y enrutadores controlados por una simple autoridad administrativa que garantiza que las rutas internas a ese sistema se mantienen en forma consistente y viable. Los enrutadores dentro del sistema autónomo están en libertad de escoger el mecanismo por el cual descubrirán, propagarán, validarán y chequearan la consistencia de las rutas.

Adicionalmente el sistema autónomo debe proveer uno o varios enrutadores que colecten toda la información referente a la alcanzabilidad de las redes que el comprende(información SPF) y esta información debe ser entregada a otros sistemas autónomos. Cada sistema autónomo le es asignado un número por la misma autoridad central que asigna las direcciones Internet y este número lo identifica ante el resto de los enrutadores cuando intercambia la información de alcanzabilidad de las redes que comprende.

Dentro de los protocolos disponibles dentro de un sistema autónomo para internamente mantener actualizado sus tablas de enrutamiento(familia de protocolos IGP, Interior Gateway Protocol) se tiene el **RIP**(**R**outing **I**nformation **P**rotocol) el cual es quizás el mas difundido; fue desarrollado originalmente en la Universidad de California en Berkeley. El HELLO, el cual está basado en un esquema de vector-distancia que no usa la cuenta de saltos (el esquema típico de los protocolos basados en vector-distancia) sino que usa un esquema basado en los retardos. El OSPF(Open SPF), el cual está basado en el esquema SPF(Shortest Path First), es un protocolo propuesto por un grupo de ingenieros de la fuerza de trabajo de Internet cuya literatura está públicamente disponible por lo cual no es necesario pagar licencia por su implementación, basado en el estatus de los enlaces y se espera que eventualmente sustituirá a los protocolos propietarios corrientemente usados.

Capitulo III

En los capítulos precedentes se estudió el modelo general sobre el que se sustenta el conjunto de protocolos de Internet y la operación de los protocolos TCP/IP a nivel de las capas 2 y 3 (capa internet y capa de transporte, respectivamente); en este capítulo se presentarán los protocolos de la capa 4 o capa de aplicación, que constituyen el aspecto mas atractivo, pues son estos protocolos los que proveen los servicios que el usuario usa cotidianamente y los que permiten sacar provecho de la interconexión a la red Internet.

Existen en el Internet una serie de servicios que pudieran llamarse tradicionales entre los cuales se encuentran:

- Acceso a computadores remotos, para lo cual existen dos protocolos a nivel de capa de aplicación el **rlogin** (usado en ambientes unix) y el **telnet**, que es mas general y puede funcionar en otras plataformas además de unix.

- Transferencia de archivos, para lo cual existen tres protocolos a nivel de capa de aplicación el **ftp** (**f**iel **t**ransfer **p**rotocol) el **tftp** (**t**rivial **f**ile **t**ransfer **p**rotocol) y el **rcp** (**r**emote **cp**y), este último para plataformas basadas en unix.

- Ejecución remota de comandos y procedimientos, para lo cual existen dos protocolos a nivel de capa de aplicación, el **rpc** (**r**emote **p**rocedure **c**all) y el **rsh** (**r**emote **sh**ell); el primero permite ejecutar procedimientos en un computador remoto y vía la red obtener los resultados producidos por la ejecución del procedimiento en la máquina remota. El segundo (**rsh**), está orientado a la ejecución de comandos del sistema operativo o del interpretador o shell, en vez de un procedimiento.

- Impresión remota, **rlp** (**r**emote **l**ine **p**rinter), es un protocolo a nivel de capa de aplicación que permite accesar a impresoras conectadas a diferentes hosts de la red como si ellos estuviesen conectados directamente.

- Sistema de archivos distribuido, **NFS** (**N**etwork **F**ile **S**ystem), es un protocolo de la capa de aplicación que permite que los sistemas de archivos de varios hosts aparezcan ante el usuario como si estuviesen en su máquina local, constituyéndose desde el punto de vista del usuario como un solo sistema de archivos integrado; esta integración se hace vía la red y en una forma que es prácticamente transparente al usuario.

- Correo electrónico (**e-mail**), el cual es un protocolo que implementa un sistema de despacho y recepción de correspondencia por medios electrónicos a través de la red y que funciona en forma muy similar al sistema de correo tradicional.

Aparte de los servicios tradicionales que se acaban de mencionar existen otra serie de servicios que se han desarrollado y puesto en operación prestando un invalorable servicio a los usuarios; entre estos servicios destacan:

-**finger** el cual es un protocolo desarrollado para facilitar la búsqueda de direcciones electrónicas de usuarios de Internet.

- **archie** este es un protocolo a nivel de capa de aplicación desarrollado para localizar software en el mundo Internet. Existen en Internet muchos hosts que mantienen software de dominio público para miles de aplicaciones; es posible con **archie** localizar los hosts en donde se encuentra un determinado software (del cual conocemos alguna característica, por ejemplo un nombre); una vez localizado es posible transferirlo (vía ftp, por ejemplo) hasta una máquina local.

- **gopher** este es uno de los mas importantes protocolos desarrollados actualmente en la capa de aplicación; tiene el propósito de proveer una forma

consistente e integral de acceder(o incorporar) servicios de información de(al) Internet. En Internet ya se ha dicho existen infinidad de aplicaciones que permiten proveer algún servicio; para cada una de estas se han desarrollado protocolos que permiten que un cliente disponga de esos servicios en un determinado host, pero en general, cada uno de estos protocolos tiene una interfase y unos comandos particulares y se limitan a proveer el servicio de una forma si se quiere inflexible; con el **gopher** se pretende proveer una herramienta que permita fácilmente a un administrador integrar los servicios de información de una instalación particular, al Internet, de una forma extremadamente simple y consistente.

En este capítulo se estudiarán algunas de las aplicaciones mencionadas de una forma que se pretende sirva como una especie de guía a los nuevos usuarios, de manera que puedan sacar provecho de la misma. No se estudiará el detalle de su implementación sino su parte operacional. Es de aclarar acá que si bien la interfase que la aplicación particular presenta al usuario es consistente e independiente de la plataforma de sistema operativo en donde se instale, la forma en la cual esa aplicación es activada o ejecutada, en general si depende del sistema operativo y del software que implementa los protocolos TCP/IP. En las próximas secciones se estudiarán las aplicaciones que se han seleccionado por considerarlas de interés fundamental para el usuario.

3.1. Aplicaciones tradicionales

3.1.1 Acceso a computadores remotos (telnet, rlogin)

Una aplicación muy atractiva de usar disponible en TCP/IP es el **telnet**; ésta fue desarrollada para permitir ganar acceso a un computador remoto (hacer login o

establecer una sesión de trabajo). **Básicamente permite que un computador(el cliente telnet)** se comporte como un terminal remoto del host en el cual se hace login **(servidor telnet)**. En general se hace login en sistemas operativos multiusuarios, multitareas, que tienen la capacidad para permitir que varios usuarios ejecuten concurrentemente varias tareas (programas), como por ejemplo el Unix.. Como casi todos los protocolos de la capa de aplicación, el **telnet** usa la filosofia del modelo cliente-servidor (recuerde que las aplicaciones servidoras tienen un número de puerto fijo, para la aplicación servidora telnet es el 23), y usa como servicio de transporte (capa 3 del modelo) el TCP, el cual hace posible una conexión interactiva con la máquina remota de forma tal que el usuario (cliente) tiene acceso a todos los comandos disponibles en el computador remoto(servidor).

Para que un usuario use esta aplicación básicamente requiere dos cosas: primero debe conocer el nombre o la dirección IP del host al cual desea contactar (por ejemplo, dino.conicit.ve ó thor.uc.ve para el caso de usar los nombres o 150.188.1.10 ó 150.186.32.2 en el caso de usar los números IP) y segundo, debe tener una cuenta o login en ese host. Los sistemas operativos multiusuarios controlan el acceso a sus recursos por medio de una identificación que el usuario debe suministrar cada vez que desee conectarse a ese host; esta identificación es personal y básicamente consiste de un conjunto de letras (típicamente constituido por 8 caracteres, la inicial del nombre mas el apellido del usuario, por ejemplo rperez o dgomez etc.); adicionalmente, cada usuario tiene una palabra clave secreta llamada el password que solo el conoce y con lo cual se le garantiza la privacidad de sus trabajos (archivos de datos, programas etc.) esto es lo que constituye una cuenta en un host(el nombre de la cuenta es el login y la palabra secreta que le da privacidad a la cuenta es el password).

Las cuentas son personales y por host; de manera que si Ud. le asignan una cuenta en un equipo, esa cuenta le permite o da acceso a los recursos de ese host (software y hardware) y no al de todos los hosts que estén en la red. El formato general para usar telnet (se supone que Ud. está instalado en un computador que posee un software que le proporciona el comando telnet) es el siguiente:

telnet nombre del host remoto

ó

telnet dirección IP del host

A continuación se transcribe una sesión telnet hecha por el usuario, **rgomez** en un host llamado **dino.conicit.ve (150.188.1.10)**; las respuestas escritas en letra pequeña provienen del host al que se desea contactar. Para hacer la sesión se usa el comando telnet en respuesta al prompt (el prompt, es el símbolo usado por el sistema operativo o el interpretador de comandos para indicar que el computador está en espera de comandos; en este caso se está suponiendo que el prompt es el símbolo $):

```
$ telnet dino.conicit.ve
Trying ...
Connected to dino.conicit.ve
Escape character is '^]'.
login: rgomez
password: ??????
SunOS UNIX (dino)
Last login: Fri Jun 15 18:15:10 from thor.uc.ve
$
```

En esencia, el comando **telnet** trata de contactar a un computador llamado **dino.conicit.ve** (puesto que este es un nombre y como se ha dicho, en Internet los nombres son para las personas, pero los protocolos usan es el número IP, posiblemente como parte de la localización del host, está la conversión vía DNS (**D**omain **N**ame **S**ystem), de nombre a el número IP que corresponde a ese nombre; después que el host ha sido localizado, el software cliente telnet (que fue invocado cuando en respuesta al prompt local se escribió el comando **telnet ...**), contacta la aplicación servidora telnet (vía el número de puerto de esa aplicación que se dijo es 23) y siguiendo el protocolo de la aplicación le informa que hay un usuario en el host local que desea abrir una sesión remota; como parte del protocolo de la aplicación telnet, se le solicita la identificación del usuario para determinar si tiene cuenta en ese

host (o sea para determinar si tiene autorización para acceder a los servicios del host remoto); se permite la sesión remota, si el usuario suministra una cuenta válida y adicionalmente suministra el password que esa cuenta tiene. Una vez que la sesión es establecida el usuario puede ordenar la ejecución de cualquier comando válido en el sistema operativo en el cual esté siendo atendido en el host remoto (en el caso del ejemplo el sistema operativo es UNIX y cualquier comando valido en UNIX puede ser ordenado). Para terminar la sesión remota, el usuario en respuesta al prompt, puede usar **exit** o **CTRL D** (control simultáneamente con la letra D, representado como ^D).

La aplicación servidora siempre supone que el cliente está conectado a través de un tipo de terminal (generalmente este terminal es emulado por el software que implementa el comando telnet); algunas veces como parte del protocolo se le solicita al usuario que suministre el tipo de terminal (esto no siempre es verdad pues desafortunadamente las aplicaciones servidoras no presentan una interfase uniforme en todos los aspectos); debe existir correspondencia entre el terminal que la aplicación servidora supone y la que efectivamente provee el cliente, pues de lo contrario algunos caracteres pueden interpretarse incorrectamente o sencillamente no producen el efecto esperado (en especial las teclas que corresponden a las flechas, las funciones y controles del teclado, son los que típicamente presentan problemas de interpretación). En general el software que implementa el telnet posibilita la emulación de varios tipos de terminales; el terminal VT100 es un estándar que casi todas emulan; otros terminales que comúnmente están disponibles son: ansi, VT220, VT51, wyse60, etc.; si Ud. desconoce el tipo de terminal que su software emula, una buena escogencia es VT100. Si la aplicación servidora se ejecuta en una plataforma Unix, es posible escoger el terminal que se adapte al emulado por la aplicación cliente, colocando en la variable de ambiente **TERM**, el nombre del terminal apropiado (por ejemplo, TERM=ansi, si el terminal emulado por el cliente es el ansi).

Hay un aspecto importante que vale la pena mencionar, la aplicación telnet hace que cualquier caracter que sea escrito por el teclado sea enviado al host remoto para su interpretación por la aplicación telnet servidora; sin embargo, existe un

caracter especial (el que cuando comienza la sesión telnet se informa que es el caracter ESC; en el caso del ejemplo es "^]"), que permite entrar en el llamado modo comando (en este modo aparece como prompt **telnet>**), en el cual, la aplicación está en espera de comandos, los cuales son interpretados por el telnet local. Existe un conjunto particular de comandos que esta aplicación reconoce, los mas usados se listan a continuación con una pequeña explicación (se puede obtener una lista de todos los comandos disponibles, escribiendo el caracter **?** en respuesta al prompt telnet>).

close		Termina la conexión telnet que esté establecida; sigue en el modo comando, o sea no termina la aplicación cliente.
open	nombre	Intenta establecer una sesión telnet con el host "nombre"
quit		Termina la aplicación telnet (cliente).
^z		Suspende temporalmente la sesión telnet y activa un sub shell (caso sistema operativo unix), que permite que el usuario ejecute cualquier comando de ese sistema y cuando termine ese subshell reasume la operación el comando telnet.

Para salir del modo comando basta con dar "Return" o "Enter" en respuesta al prompt. Es también importante resaltar el hecho que una vez en el modo comando uno puede abrir una sesión telnet en otro host por medio del uso del comando open.

Existe una aplicación alterna que permite establecer conexiones remotas, pero que solo funciona entre dos hosts que ejecutan el sistema operativo Unix, es la aplicación **rlogin**; esta aplicación fue desarrollada por la Universidad de Berkeley y forma parte de la versión distribuida de ese Unix(BSD UNIX), que incluye una serie de comandos que son los mismos que se encuentran en un ambiente Unix, pero que se les antepone la letra **r** para indicar remoto; por ejemplo para el **cp**, comando copy de Unix, existe el **rcp**; para el **lp**, comando para imprimir en Unix; existe el **rlp**; para el **login**, comando para hacer sesión, existe el **rlogin**; en general casi todas las versiones de Unix System V, traen implementado estos comandos. El rlogin posee ciertas ventajas, producto precisamente del hecho de que la aplicación funciona en una

plataforma uniforme (el mismo sistema operativo) y por tanto ellas se comunican mejor y proveen mayor flexibilidad al usuario, pero lamentablemente está restringido a máquinas que estén ejecutando Unix. El formato para usar el comando **rlogin** es:

$rlogin nombre del host remoto (puede usar la dirección IP)

3.1.2 Transferencia de archivos (ftp, rcp)

La aplicación **ftp** permite mover archivos de un computador a otro. Su operación es independiente de la localización de los computadores, de la forma en la que están conectados e inclusive del sistema operativo que se ejecute en las máquinas involucradas. Usa como protocolo de transporte (capa 3) TCP y la aplicación servidora tiene asignado como número de puerto el 21(realmente ftp tiene un esquema de operación tan complejo que tiene dos números de puertos el 21 y el 20; el primero es la aplicación servidora normal de cualquier aplicación y el segundo es asignado a la transferencia efectiva del archivo, ftp-data). Esta es una aplicación que aun cuando pudiera parecer sencilla es realmente un complejo programa debido que debe proveer una gran flexibilidad de manera de adaptarse a la gran variedad de formas y estructuras de datos que manipula. La aplicación **ftp** provee básicamente las siguientes facilidades:

- Acceso interactivo: Provee una interfase interactiva al usuario, para lo cual hay una serie de comandos que le permiten al usuario conocer por ejemplo, en que directorio está posicionado, posicionarse en cualquier directorio que le esté permitido, listar el contenido de los directorios donde esté, etc.

- Especificación de formato: El cliente tiene la posibilidad de especificar el tipo y el formato de la información a transferir; por ejemplo, puede especificar si el archivo es texto(contiene solo caracteres imprimibles) o es binario (contiene cualquier caracter) y si la representación o código empleado para los caracteres es ASCII o EBCDIC .

- Control de autenticación: Es necesario que el usuario disponga de una cuenta en el host que desea contactar, para poder ganar acceso al mismo y transferir

algún archivo. En muchos hosts existe una cuenta genérica a través de la cual es posible acceder a algunos servidores y transferir archivos de dominio público, como se verá mas adelante.

- Transferencia bidireccional: Una conexión **ftp** permite que se transfieran archivos hacia o desde uno cualquiera de los computadores involucrados.

Para que un usuario use esta aplicación básicamente requiere dos cosas: primero debe conocer el nombre o la dirección IP del host al cual desea contactar (por ejemplo, dino.conicit.ve ó thor.uc.ve para el caso de usar los nombres o 150.188.1.10 ó 150.186.32.2 en el caso de usar los números IP) y segundo debe tener una cuenta o login en ese host. . El formato general para usar **ftp** (se supone que Ud. está instalado en un computador que posee un software que le proporciona el comando **ftp**) es el siguiente:

ftp nombre del host remoto

ó

ftp dirección IP del host remoto

A continuación se transcribe la forma de hacer una sesión **ftp** en un host llamado **dino.conicit.ve (150.188.1.10)** por parte de un usuario cuyo login es **rgomez**; las respuestas escritas en letra pequeña provienen del host al que se desea contactar. Para hacer la sesión se usa el comando **ftp** en respuesta al prompt (se está suponiendo que el prompt es el símbolo $):

```
$ ftp dino.conicit.ve
Connected to dino.conicit.ve
220 dino FTP server (SunoOS  Unix)  ready.
name: rgomez
331 password required for  rgomez
Password:
230 User rgomez logged in
ftp>
```

Obsérvese que el comportamiento de la aplicación **ftp** es similar a la del telnet en el sentido de que es necesario suministrar una cuenta; en el caso del **ftp**, en respuesta a "name" el usuario debe suministrar su cuenta (en el ejemplo rgomez); después se le pide al usuario que suministre el password para esa cuenta; cuando el usuario suministra el password lo que el escriba no se ve en la pantalla esto es con fines de seguridad de manera que si hay alguna persona alrededor no vea el password, pues como se dijo anteriormente su propósito es precisamente garantizar la privacidad de los archivos pertenecientes al usuario. Si se suministra un nombre y un password correcto se le autoriza acceso en el host especificado y el usuario puede comenzar a transferir archivos de uno a otro computador (esto es indicado por el prompt **ftp>**). El usuario dispone de dos comandos básicos **get** y **put**. El primero permite traer un archivo del host remoto al local (host local es el que inició el ftp o sea el cliente) y el segundo permite copiar un archivo del host local al host remoto (el servidor). Los archivos que se copian se pueden traer con el mismo nombre (por defecto es así) o el usuario puede especificar el nombre con el que desea la copia. El formato general de los comandos **get** y **put** se da a continuación:

ftp> **get archivo-fuente [archivo -destino]**
ftp> **put archivo-fuente [archivo-destino]**

Si se especifica "**archivo-destino**" el archivo fuente("**archivo-fuente**") es copiado en el destino con el nombre "**archivo-destino**"; si no se especifica "**archivo-destino**", el archivo es copiado con el mismo nombre que tenía en la fuente ("**archivo-fuente**").

Desde el punto de vista del usuario el **ftp** es una aplicación interactiva y el usuario dispone de una serie de comandos, además de **get** y **put**, los mas importantes se describen a continuación (el usuario puede obtener una lista completa de todos los comandos disponibles tipeando **help** cuando le sea presentado el prompt **ftp>**)

ascii	Selecciona el modo ASCII, para transferir archivos tipo texto.
binary	Selecciona modo binario, para transferir archivos binarios.
cd dirname	Selecciona como directorio de trabajo en el host remoto a **"dirname"**.
delete filename	Borra el archivo **"filename"** en el host remoto.
dir	Lista por pantalla todos los archivos que se encuentren en el directorio actual de trabajo en el host remoto. Es posible especificar un archivo particular o un conjunto de archivos usando los conocidos caracteres de sustitución * e **?** (por ejemplo **dir *.txt** ó **dir ?file**). También es posible especificar un destino para el resultado del comando (por defecto el terminal) el cual puede ser un archivo o un comando de los conocidos típicamente como filtros.
help	Proporciona una lista de los comandos disponibles. Es posible obtener una pequeña información de un comando particular especificando help seguido del comando del cual se desea obtener información; por ejemplo **help dir**.
lcd dirname	Cambia el directorio de trabajo en el host local a **"dirname"**.
mget lista	Permite transferir desde el host remoto hasta el host local múltiples archivos (indicados acá por **lista**). Se pueden usar los caracteres de sustitución * e **?** para generar automáticamente la lista de archivos a transferir. El comando **mget** supone que todos los argumentos que siguen al comando son archivos a transferir, de manera que no es posible especificar un recipiente (por ejemplo un directorio) en donde colocar el grupo de archivos a transferir; si se desea que el grupo de archivos sea colocado en un directorio particular el usuario debe posicionarse en ese directorio, antes de ejecutar el **mget**.
mput lista	Permite transferir desde el host local hasta el host remoto múltiples archivos (indicados acá por **lista**). Se pueden usar los caracteres de sustitución * e **?** para generar automáticamente la lista de archivos a transferir. Lo que se aclaró para el **mget** es también válido para el **mput**.
open host	Intenta abrir una conexión **ftp** con el host especificado.
pwd	Permite averiguar en que directorio se está posicionado en el host remoto.
quit	Permite cerrar cualquier conexión que esté abierta y sale de **ftp.**

Como se ha mencionado, el comando **ftp** exige que el usuario tenga una cuenta en el host al cual desea contactar como una medida de seguridad. Para permitir el acceso a archivos públicos muchos hosts en Internet permiten el uso de una cuenta especial llamada **anónimos,** por convención, que o bien no necesita password o el password que se usa es **guest** ó el login en el host local (el cliente); en otros sitios puede que le pidan como password su dirección electrónica; lo importante es que lo que sea que requiera el sistema servidor como password, generalmente lo dice como parte del intercambio de información para establecer la sesión y el usuario debe estar pendiente de cual información se le solicita ya que desafortunadamente no existe uniformidad al respecto, pues en general esto depende del sitio y de las políticas del administrador del sistema. Con esta posibilidad un usuario que no tenga cuenta en un host puede todavía hacer una conexión **ftp anonymous** y transferir archivos desde o hacia ese host. A continuación la transcripción de una sesión ftp anonymous a un host llamado **nic.sura.net.**

```
$ftp nic.sura.net
Connected to nic.sura.net.
220 nic.sura.net FTP server (Versión 6.9  Jun 15 1994) ready
Name: anonymous
331 Guest login ok, send e-mail address as password.
Password: fjara@thor.uc.ve
ftp> cd pub/nic
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
Total 4096
-rw-rw-r---    1  root   2500  Jan 10  10:50  acceptable.use.policy
-rw-rw-r--     1  root   8790 May 11  17:25  archie.manual
...
226 Transfer complete
1772 bytes received in 1.2 seconds (1.4 Kbytes/s)
ftp> get archie.manual
200 PORT command successful
Opening ASCII mode conection for archie.manual (8790 bytes).
226 Transfer complete.
8790 bytes received en 2 seconds (4.9 Kbytes/sec)
ftp> quit
221 Goodbye
$
```

En el ejemplo anterior se hace una conexión ftp anonymous a un host llamado nic.sura.net, el cual pide la dirección electrónica como password; se hace un cambio de directorio con el comando cd, para posicionarse en **pub/nic**; una vez en ese directorio se le pide un listado de todos los archivos que se encuentran en el, con el comando **dir**; el host informa que hay 4096 archivos en ese directorio y lista los archivos en un formato típico de Unix (acá solo se han mostrado dos archivos por razones de espacio), en el cual la primera parte de cada línea corresponde a los permisos del archivo, después aparece el dueño del archivo(root), el tamaño del archivo, la fecha y hora de creación y por último el nombre del archivo. Luego con el comando **get** el archivo es transferido hasta el host local. El ftp siempre avisa cuantos bytes transfirió y la rata efectiva de transferencia.

El comando **rcp** también permite transferir archivos entre hosts a través de la red pero funciona entre equipos que estén operando bajo el sistema operativo Unix, o sea los dos equipos involucrados en la transferencia deben estar corriendo Unix.

3.1.3. Correo electrónico (e-mail)

El correo electrónico es quizás una de las aplicaciones mas usadas en Internet; debe su popularidad al hecho de que ofrece un método rápido y conveniente de transferir información, la cual puede estar constituida por unas pocas líneas, una voluminosa correspondencia e inclusive un archivo; en todos estos casos el mecanismo es muy simple. El correo electrónico difiere de las otras aplicaciones que se han estudiado, en virtud de que no es un servicio usuario final a usuario final (end to end). La máquina que recibe y la que envía no necesitan comunicarse directamente, y el mecanismo de operación de este servicio es conocido como un servicio de "store and forward", pues el correo se va pasando de una máquina a otra y puede que en el trayecto sea almacenado en una de estas máquinas intermedias hasta que pueda alcanzar su destino final, en una operación parecida al servicio postal convencional. Al igual que todas las aplicaciones existe un puerto fijo para la aplicación servidora del

correo electrónico, es el 25 y corresponde al protocolo **SMTP** (Simple Mail Transport Protocol).

Existen muchos manejadores de correo electrónico (cada uno con una interfase particular ante el usuario), pero quizás los mas extendidos son el **mail**, que acompaña a todas las versiones del sistema operativo Unix y el **elm** que es en la actualidad el mas usado. Este es un software de dominio público que puede ser conseguido vía ftp anonymous en algunos servidores de Internet y que debe ser compilado (para lo cual necesita el compilador C del sistema de desarrollo de Unix) y configurado en el host particular en donde se vaya a instalar. El programa básico que viene con Unix, **mail**, también provee casi los mismos servicios que permite el **elm**, pero es menos amigable.

Cualquiera sea el manejador de correo que se use es importante conocer las direcciones electrónicas para poder enviar un correo en Internet. La dirección electrónica de una persona consiste de su cuenta (solo el login, sin el password), seguido del símbolo **@** , luego el nombre del host en donde está definida la cuenta y el dominio. Por ejemplo para un usuario que tenga una cuenta en un equipo llamado **thor.uc.ve** y que le hayan asignado como login **rgomez**, su dirección electrónica sería **rgomez@thor.uc.ve**. En este caso **thor** es el nombre de la máquina y **uc.ve** es el nombre del dominio. Muchas veces en algunas instalaciones ocultan el nombre de la máquina y usan un mecanismo interno para distribuir el correo en su dominio de acuerdo a una tabla de alias. Por ejemplo, el usuario **ftorres** que tiene una cuenta en el host **dino.conicit.ve**, pudiera tener una dirección electrónica como **ftorres@conicit.ve**.

La dirección electrónica tiene el formato que se acaba de explicar, sin embargo, si el destinatario del correo es un usuario en el mismo equipo desde donde se está mandando el correo, no es necesario especificar la dirección en el formato descrito anteriormente sino que basta con especificar solo el login del usuario; en otras palabras si alguien que tenga una cuenta en **thor** desea enviar un correo al usuario **rgomez**, puede especificar como destinatario **rgomez** puesto que están en la misma máquina

(thor); es de observar que el formato completo también es valido cuando se esté en la misma máquina.

Es posible encontrar direcciones electrónicas que no tengan el formato descrito, lo cual en general se debe a que Internet está conectado a otras redes externas que tienen otros formatos, de manera que es difícil indicar un formato estándar para la dirección electrónica, sin embargo, afortunadamente los programas que efectivamente se encargan del envío del mail (no el que interactua con el usuario) en general, se pueden configurar de manera que se encarguen de hacer los ajustes necesarios en las direcciones para adaptarlos al formato que la aplicación maneja.

Para arrancar la ejecución del manejador de correo **elm** basta con escribir en respuesta al prompt el siguiente comando:

$elm

Una vez que se ha dado este comando se arranca la ejecución del manejador de correo **elm** el cual muestra una pantalla en donde aparecerán todos los correos recibidos, con una indicación de su estatus y en la parte inferior mostrará una lista de comandos disponibles, tal como se muestra en la siguiente transcripción de una sesión con elm.

```
Reading in /usr/spool/mail/fjara, message:
Mailbox is '/usr/spool/mail/fjara' with 8 messages      ELM 2.4 PL21

->      1  Jun 29 archie-errors@arch (565)  archie [prog archie] part 1 of 1
        2  Jun 28 Luis           (49)  Nervios.
        3  Jun 20 Proyecto Monitoreo (74)  cursos andinos
        4  Jun 15 yanoff@csd4.csd.uw (915)  Updated Internet Services List
        5  Jun 14 Luis           (43)  Recordatorios.
        6  Jun 9  Sergio Cimirro    (136)  Re: Tarea 2
        7  Jun 7  Eddy Carrasco     (222)  Jornadas
        8  Jun 1  yanoff@csd4.csd.uw (955)  Updated Internet Services List

    |=pipe, !=shell, ?=help, <n>=set current to n, /=search pattern
a)lias, C)opy, c)hange folder, d)elete, e)dit, f)orward, g)roup reply, m)ail,
   n)ext, o)ptions, p)rint, q)uit, r)eply, s)ave, t)ag, u)ndelete, or e(x)it

Command:
```

El **elm**, posee cierto nivel de ayuda la cual se obtiene con el caracter **?**. En general los nombres de los comandos fueron escogidos de forma que casi se explican por si solos. Es posible entre otras cosas, enviar un mail a cualquier usuario local o externo(**m**), crear una lista personal de alias de las direcciones mas frecuentemente usadas(**a**), enviar una copia a otra persona de un correo recibido(**b** o **f**), imprimir(**p**) o guardar(**s**) en un archivo un correo recibido etc., todo esto de una forma relativamente amigable. Cuando se va a enviar un correo a alguien es necesario primero conocer su dirección electrónica (**To**); segundo especificar un sujeto (**Subject**) que sintetice el contenido o asunto de lo que trata el correo y tercero indicar si se le desea enviar copia (**Cc**) a otros usuarios (se pueden enviar copias a cualquier cantidad de usuarios, remotos o locales). El cuerpo del mensaje se escribe con un editor que el usuario puede seleccionar de los que generalmente se encuentran en una instalación Unix; en el caso específico de la instalación de la Universidad de Carabobo, el editor usado es el **vi**, que es muy poderoso pero poco amigable (en los anexos, se incluye un resumen de los comandos de edición disponibles en **vi**). Una alternativa al editor **vi** es el **emacs**, pero el mismo todavía no está disponible en nuestra instalación; el **emacs** es mas amigable que el **vi** y tiene funciones que van mas allá de las de un simple editor.

3.2. Otros servicios

3.2.1. Búsqueda de personas y direcciones en Internet (finger, whois, fred)

El Internet es tan grande que se han desarrollado aplicaciones para localizar usuarios y direcciones electrónicas, que cumplen un propósito similar al de las guías telefónicas. Estas aplicaciones, cuya labor en apariencia sencilla, se ven limitadas básicamente por la carencia de un directorio unificado de toda Internet. Muchas razones han privado para no disponer aun en la actualidad de un directorio unificado de Internet pero básicamente son tres las mas importantes: La movilidad de los usuarios, las direcciones electrónicas pertenecen a hosts específicos y estos hosts pertenecen a organizaciones, de manera que, al cambiar de trabajo o aun el simple

cambio de departamento dentro de una organización puede implicar un cambio de dirección electrónica; el retardo en establecer un estándar para la creación y manipulación de los directorios, lo que condujo a que muchas organizaciones desarrollaran sus propios esquemas y ahora una vez que ya se dispone del estándar es dificil que esas organizaciones, que ya tienen un sistema funcionando que les presta un servicio adecuado, adopten un estándar que si se quiere todavía en la práctica es experimental; por último, factores de seguridad y preservación de la privacidad que en algunos países es materia regulada severamente por las leyes. A pesar de todos estos factores se ha avanzado mucho en los últimos tiempos en esta materia y hay varias aplicaciones y servidores dedicados a proveer un directorio que permita localizar personas y direcciones electrónicas de las personas en Internet. A continuación se estudiarán tres de estas aplicaciones que se han considerado representativas.

3.2.1.1 Búsqueda de un usuario en un sistema específico (finger)

El **finger** es una vieja aplicación disponible en el sistema operativo Unix, que tiene la capacidad de examinar el archivo **/etc./passwd** de un host. El sistema operativo Unix mantiene una lista de todas las cuentas autorizadas para acceder sus recursos, en un archivo llamado **/etc./passwd,** en el cual además del nombre de la cuenta, aparece cierta información personal como el nombre de la persona dueña de la cuenta, el directorio home y la dirección electrónica.

La aplicación **finger** no solo se limita a leer el archivo /etc./passwd sino que informa si el usuario consultado está conectado actualmente a ese host o cuando fue la última vez que se conectó; es posible, y esto depende de la instalación, que suministre información adicional como por ejemplo, los planes del usuario consultado lo cual lo hace listando dos archivos especiales llamados **.plan** y **.project**; estos archivos los debe crear el usuario en su directorio y darle permisos para que cualquier persona los lea (caso sistema operativo Unix usar **chmod 555 .plan .project**); es capaz de mostrar el estatus del correo del usuario consultado(si tiene correo sin leer). Si bien la aplicación **finger** es una facilidad provista primariamente en los ambientes Unix, ya

existen en otras plataformas aplicaciones similares que permiten hacer consultas a otros hosts en una forma similar al finger de Unix.

El **finger** está orientado a realizar consultas por hosts y no por organizaciones. El usuario de **finger** debe conocer por lo menos el nombre del host al que desea consultar. El formato general del comando **finger** es:

$ finger nombre@nombre-host

"**nombre**" es opcional y puede ser omitido (luego se explicará lo que hace finger cuando se omite); si se suministra, especifica el nombre que se quiere buscar en la máquina remota. La aplicación finger cliente contacta la aplicación finger servidora (puerto 79) en el host "**nombre-host**" y le pide que le envíe información de todas las cuentas que aparecen en el archivo **/etc./passwd** del servidor, en las cuales aparezca la palabra "**nombre**", en los campos que corresponda a nombre de la cuenta (login name) o nombre de usuario (In real life), de ese archivo (el archivo /etc./passwd, contiene varios campos con nombres específicos entre los cuales están "login name" y "In real life"; el primero contiene el nombre de la cuenta asignada al usuario y el segundo en general corresponde al nombre verdadero, nombre y apellido, del usuario dueño de la cuenta). Uno debe suministrar el primer nombre o el apellido de la persona que se desea localizar o puede suministrar también el nombre de la cuenta(lo cual es en general lo que uno desconoce), cualquiera sea lo que suministre debe estar completo de manera que uno no puede dar una parte del nombre (por ejemplo si uno busca un rodriguez no puede dar rodrig).

"**nombre-host**", es el nombre de la máquina donde se desea hacer la consulta. Si la consulta se quiere hacer en el host desde el cual se está ejecutando el **finger**, se puede omitir la porción **@nombre-host**, con lo cual la consulta se hace en el archivo /etc./passwd del host local.

Si se omite la porción "**nombre**", **finger** lista los usuarios que estén actualmente en sesión en el computador especificado; la sintaxis del comando obliga

en este caso a incluir el caracter @ antes del nombre del computador de manera que el comando sería **$ finger @nombre-host**. Es posible inclusive omitir tanto "**nombre**" como "**nombre-host**", en esta caso **finger** lista todos los usuarios que estén en sesión actualmente en el host local. A continuación las transcripciones de algunos ejemplos de uso de **finger**.

Ejemplo 1

$finger zapata@cs.uml.edu
[cs.uml.edu]

Login name: mzapata In real life: Margarita
Office: WL 234, (508) 934-36-24 Home phone: (---) --- -- --
Directory: /usr/u/staff/mzapata Shell: /bin/csh
Last login Tue Jun 28 15:19 on ttyq7 from altair.uml.edu
Plan: No plan.

Ejemplo 2

$finger @cs.uml.edu
[cs.uml.edu]

Login	Name	TTY	Idle	When	Office
dpourgha	Dara Pourghasemi	01		Tue 18:34	
pchen	Philip Chen	02	2	Tue 18:35	
srujipat	Sirichai Rujipattana	03	6	Tue 18:37	(508) 934-3184
mtaheri	Mandeuchehr Taheri	05	1:19	Tue 17:24	
cjones	Christopher B. Jones	08	2	Tue 17:47	
mtaheri	Mandeuchehr Taheri	10	1:23	Tue 17:19	
sdamiani	Stephen R. Damiani	11		Tue 18:22	
tmuzeral	Tricia A Muzerall	16	1:07	Tue 17:37	
ale	An Le	19		Tue 16:40	

. . .

Es evidente que **finger** provee una ayuda limitada sobre todo porque uno debe realmente conocer información muy específica para poder sacar provecho de esta aplicación y para el caso de buscar una dirección electrónica de alguien del cual apenas conocemos el nombre o del que solo sabemos la organización donde trabaja no es muy útil. Sin embargo el **finger** provee cierta ayuda y aprovechando una de sus facilidades que es el de listar los archivos **.plan** y **.project** de la cuenta consultada, en muchas instalaciones se usa esta aplicación como un medio para distribuir información

limitada acerca de algún tópico, para lo cual se crean cuentas específicas y en los archivos **.plan** y **.project** de esas cuentas, se escribe la información que se desea distribuir (los permisos de esos archivos y del directorio donde se encuentren, deben ser amplios para todos los usuarios, por lo menos de lectura en el caso de **.plan** y **.project** y de lectura y ejecución para el directorio); cuando algún usuario consulta vía **finger** esa cuenta, el finger responde suministrando la información que se vio respecto del usuario y luego lista los contenidos de los archivos mencionados. Por ejemplo, en la cuenta **quake** en el host **geophys.washington.edu**, alguien mantiene en el archivo **.plan**, una lista de los recientes temblores ocurridos en el planeta. Ud. podría obtener esta información ejecutando el comando:

$finger quake@geophys.washington.edu.

Hay otras organizaciones en el Internet que proveen algunos servicios de información vía **finger**.

3.2.1.2. Directorio de páginas blancas (whois)

whois, es el nombre de una aplicación y el nombre de una base de datos mantenida por el Centro de Información de Red del DDN (DDN NIC, Network Information Center). **whois** constituye uno de los primeros intentos en Internet de crear un directorio o servicio de páginas blancas y muchas instalaciones lo han usado como modelo para la creación de sus propios directorios; estos nuevos directorios en general pueden ser accedidos de la misma forma que la base original pues mantienen compatibilidad. En esa base de datos existen cerca de 70000 registros de personas que han realizado algún trabajo en Internet o cuya área de investigación son las redes. Existen tres formas de acceder esta base de datos: usando el comando **whois**, el comando **telnet**, o a través del correo electrónico (e-mail).

Para usar el comando **whois** basta con suministrar como argumento de dicho comando el apellido de la persona de la cual se quiere conseguir información. Por

ejemplo si se desea conseguir información acerca de un investigador en el área de redes cuyo apellido es **krol** se deberá dar el comando siguiente:

$ whois krol

La aplicación contactará al servidor de la base de datos whois (**nic.ddn.mil**) y le pedirá la información del usuario especificado (**krol**). De encontrarse esta persona en el directorio de esa base de datos toda la información importante referida a este usuario será regresada; esta información incluye su nombre completo, su dirección electrónica actual, la organización donde trabaja, dirección y la fecha de la última actualización de esta información. La especificación del nombre es un poco mas flexible que con el finger pues se puede escribir una parte del nombre, para lo cual basta con terminarlo con un punto. Por ejemplo si en el caso del ejemplo anterior no se está seguro de si el apellido es kroll o krol pudiera darse el comando de la siguiente forma: **$ whois kro.** En este caso el servidor de whois busca todos los nombres que comiencen por **kro** que aparezcan registrados en la base de datos; la información que regresa no es detallada como en el caso de una búsqueda específica, pues el propósito es que el usuario identifique cual es la persona precisa de la que requiere información para luego hacer una nueva consulta, pero ya en forma específica y no genérica. Es importante tener presente la sintaxis de la forma genérica de busca, en donde el punto es necesario y es lo que realmente indica a la aplicación que la búsqueda es genérica y debe buscar todos los registros que contengan nombres que comiencen por la cadena que precede al punto.

El **whois** puede ser usado a través de la aplicación **telnet,** para lo cual es necesario hacer una sesión en el servidor **whois** (hay varios servidores, el principal es **nic.ddn.mil**). Se comienza por hacer la sesión telnet en el servidor para lo cual se da el comando **$ telnet nic.ddn.mil**. El servidor una vez contactado mostrará un promtp @ y el usuario debe dar el comando **whois**; cuando el sistema responda con el prompt **whois:** el usuario puede comenzar a trabajar en una forma interactiva con la base de datos, disponiendo de una ayuda (limitada) con el caracter **?** y puede realizar

varias búsquedas hasta que decida terminar la sesión lo cual se hace respondiendo con un ^D al prompt **whois:**. Luego de terminada la sesión whois es necesario terminar la sesión telnet lo cual se hace también con ^D .

Es posible acceder a los servicios whois vía correo electrónico para lo cual es necesario mandar un correo a la dirección electrónica **service@nic.ddn.mil**; el sujeto (Subject) que se coloque no importa y en el cuerpo del mensaje se escribe el comando whois seguido del nombre de la persona que se desea localizar(por ejemplo **whois kro.**), en forma similar a como se hace con el comando whois. Posteriormente le llegara un correo a su cuenta con la información solicitada. La base de datos whois además de contener información acerca de personas provee también información acerca de dominios y redes de esos dominios. Por ejemplo se podría averiguar información acerca de las redes de la Universidad de Rutgers, enviando un correo a **service@nic.ddn.mil** y colocando en el cuerpo del mensaje:

whois University Of Rutgers.

Si se envía este correo llegará información a la cuenta que lo envía, en donde aparecerán las redes que esa Universidad tiene y los servidores que existen en sus diferentes áreas.

3.2.1.3. Directorio estándar de Internet X.500 (usando fred)

En Internet ya existe un estándar para proveer los servicios de directorios de usuarios y otra información importante acerca de la red, sin embargo no está muy difundido, primero por su aparición tardía y segundo porque su interfase no es muy amigable y requiere que el usuario tenga una noción de su filosofía para poder sacar provecho. Realmente no existen muchos servidores en Internet que provean este servicio sin embargo, NYSERnet y PSI desarrollaron una aplicación que provee una interfase mas amigable para acceder los servicios de los servidores de directorios

basados en el estándar X.500 y la llamaron **fred**, esta aplicación forma parte de un proyecto piloto para el establecimiento de un servidor X.500 en PSI.

La aplicación **fred** está disponible vía telnet (usando el comando **telnet wp.psi.com**) o por medio de correo electrónico escribiendo a la dirección electrónica **whitepages@wp.psi.com** y colocando como sujeto del mensaje lo que se desea buscar en una forma similar a como se hace con whois; de hecho el comando para ordenar la búsqueda es **whois** y se pueden especificar patrones por medio del caracter *. En estos casos * opera en forma diferente a como son interpretados por el Unix y realmente lo que quieren significar es la existencia de ningún o mas de un caracter. Esta es una aplicación experimental y no existen en el mundo Internet muchos servidores que la soporten y mantengan, sin embargo, es muy poderosa sobre todo si la búsqueda involucra una consulta en una base de datos con una gran cantidad de registros (mas de 70000).

3.2.2 Buscando software en Internet (archie)

En Internet hay mas de 1000 servidores en los cuales se puede hacer un ftp anónimo para traer software de dominio público.Contactar en forma individual cada uno de estos servidores para averiguar si contiene entre sus archivos de dominio público algún archivo específico, sería una tarea muy laboriosa y por supuesto impráctica. Por otra parte si realmente no se conoce el nombre del software que se busca sino que se tiene alguna palabra clave que se sabe debe estar en alguno de los archivos, es todavía mas difícil la localización del software buscado. **Archie** es un sistema desarrollado en la Universidad de McGill, Canadá, que permite la búsqueda por índice de aquellos archivos que se encuentran en servidores públicos en Internet.

La idea detrás de **archie** es si se quiere sencilla, básicamente consiste en disponer de unos servidores (**servidores archie**) que se encargan de contactar todos los servidores de ftp anonymous conocidos en Internet y solicitarle que le envíen un directorio de todos los archivos que están disponibles en el; el servidor **archie** recibe

la información proveniente de cada servidor y basándose en esa información construye un directorio de todos los archivos disponibles en los servidores públicos de ftp. El usuario **archie** contacta al servidor archie y a través de este hace las consultas en el directorio previamente construido. Para realizar las consultas existe un protocolo que soporta una serie de comandos que se explicaran mas adelante.

Para usar **archie** es necesario dirigirse a uno de los **servidores archie** que existen en el Internet. Existen varios servidores archie que tienen una cierta área de influencia de las cuales se suponen vendrán la mayoría de las solicitudes, sin embargo, realmente uno puede contactar a cualquiera de estos servidores independientemente de que esté o no dentro de su supuesta área de influencia, pero en general uno debe escoger el servidor que geográficamente esté mas cerca pues casi siempre es el que mejor rendimiento presenta, recuerde que si bien es cierto que independientemente de la localización fisica es posible contactar cualquier host, también es cierto que mientras mas alejado esté dicho host, los paquetes tendrán que pasar a través de mas enrutadores o gateways y estos introducen siempre algún retardo que a la larga se refleja en la velocidad a la cual se obtienen las respuestas. A continuación una lista de los servidores **archie** disponibles en Internet y su área de influencia.

Nombre del servidor archie	Area de influencia
archie.rutgers.edu	**Noroeste de USA**
archie.sura.net	**Sureste de USA**
archie.unl.edu	**Oeste de USA**
archie.ans.net	**Provedores de servicios de Internet (ANS)**
archie.mcgill.ca	**Canada**
archie.au	**Australia y la Cuenca del Pacífico**
archie.funet.fi	**Europa**
archie.doc.ic.ac.uk	**Reino Unido**

A los servidores **archie** se les puede acceder de tres formas: Usando telnet para contactar directamente a uno de los servidores disponibles; usando un programa

archie cliente que está disponible públicamente y por último usando el correo electrónico.

La forma mas común de usar **archie** es estableciendo una sesión telnet en uno de los servidores archie disponibles en Internet (**telnet "servidor archie"**). El servidor escogido responderá pidiendo la identificación (login:); se debe responder escribiendo **archie**; no pedirá password y al activarse la aplicación en el servidor, aparecerá el prompt de la aplicación archie (**archie>**), lo que indica que está disponible para procesar los comandos que se le suministren.

El usuario tiene la posibilidad de hacer búsquedas por nombre de archivos (usando el comando **prog**) o puede hacer búsquedas por un índice descriptivo (usando el comando **whatis**). El formato del comando **prog** es el siguiente:

archie> **prog** texto-buscado

Con el comando anterior se le está pidiendo a **archie** que busque en su base de datos todos los registros en donde aparezca "texto-buscado"; es importante aclarar acá que la forma en que **archie** interpretará "texto-buscado" depende del tipo de búsqueda que se haya definido. Existen en **archie** varias formas de búsqueda; el usuario puede escoger la que considere conveniente usando el comando **set** para colocar en la variable **search**, el tipo de búsqueda deseada. Hay cuatro valores posibles que se le pueden asignar a la variable **search**, que definen cuatro formas diferentes de interpretar "texto-buscado", las cuales se describen a continuación:

exact	"texto-buscado" debe coincidir exactamente con el nombre de un archivo.
regex	"texto-buscado" es tratado como una expresión regular tipo Unix; se buscan en la base de datos todos los archivos cuyos nombres satisfagan la expresión regular.
sub	"texto-buscado" es interpretado como un sub string. Se buscaran en la base de datos todos los archivos cuyos nombres contengan el substring "texto-buscado", sin diferenciar mayúsculas de minúsculas. Este es quizás el tipo de búsqueda mas útil.

subcase	**La interpretación es igual que con sub pero ahora si se diferencia mayúsculas de minúsculas.**

Si no se especifica ningún valor para la variable **search**, las búsquedas que se ordenen usan como criterio de búsqueda el que le haya definido como valor por defecto el administrador; no existe un criterio uniforme entre los distintos servidores como valor por defecto, de manera que lo mas conveniente es definir como paso previo a la realización de cualquier búsqueda, el criterio para la interpretación del argumento dado a **prog**. Por ejemplo si se desea que **archie** interprete los argumentos suministrados a **prog** como una expresión regular tipo Unix, se deberá dar el siguiente comando:

archie> **set search regex**

Una vez definido un criterio, el mismo se mantiene durante toda la sesión **archie** a menos que explícitamente sea cambiado usando de nuevo el comando anterior.

Es posible la búsqueda de software a través de **archie** sin conocer el nombre del archivo sino solamente alguna palabra clave que se sabe tiene relación con ese software. Los administradores de los servidores públicos de archivos, cada vez que colocan un software en su directorio de distribución pública, incluyen una lista de palabras claves que tienen alguna afinidad con el nombre del archivo, esta información es usada en **archie** para crear una base de datos descriptiva del software.

La búsqueda de software usando un índice descriptivo se hace en **archie** por medio del comando **whatis**, el cual tiene el siguiente formato

archie> **whatis** texto

En este caso **archie** busca "texto" en la base de datos descriptiva de software y lista el nombre de todos los archivos que la contengan como palabra clave en su descripción. Con el comando **whatis** existe cierta incertidumbre en cuanto a que los archivos que reporte existan realmente en los servidores públicos de ftp, pues como se mencionó la tarea de incorporar un índice descriptivo depende del administrador del sistema y no hay garantía de que el lo mantenga actualizado. Con el comando **prog** no

ocurre lo mismo, pues la base de datos es actualizada cada 30 días en forma automática.

Adicionalmente **archie** provee una serie de comandos que pueden ser de utilidad; una lista completa de todos los comandos disponibles se puede obtener escribiendo **help** en respuesta al prompt **archie**. Es posible obtener una ayuda para cada comando específico escribiendo en respuesta el prompt archie la palabra help seguida del comando del cual se quiere ayuda; por ejemplo para obtener ayuda acerca del comando set se deberá dar **help set.** Algunos de los comandos disponibles se listan a continuación:

set variable valor	Permite darle algún valor a las variables que controlan la sesión archie. En este caso a "variable" se le asigna "valor". Las variables disponibles son: **mailto, pager, maxhits, sort, search y term**. Ya se explicó para que se usa la variable **search**; use **help set variable** para obtener una explicación del uso de "variable".
mail destino	Envía, por correo electrónico, el resultado de la última búsqueda, a la dirección electrónica especificada por "destino".
show variable	Muestra el valor que tenga definido "variable"; si no se especifica ninguna variable, **archie** muestra todas las variables con los valores que tengan actualmente asignados.
site nombre-host	Da una lista de todos los archivos que se encuentran disponibles en el servidor de ftp "nombre-host".

Es posible acceder a los servicios de archie vía correo electrónico para lo cual es necesario enviar un correo electrónico a **archie** en cualquiera de los servidores archie que se dieron anteriormente (por ejemplo **archie@archie.sura.net**); el cuerpo del correo contendrá los comandos que se desee dar a **archie**; pueden suministrarse varios comandos en mismo mensaje y es posible especificar que los resultados sean enviados a una dirección particular(comando **path** dirección-electrónica; por defecto

archie usa la que aparezca en el **From:** del correo) y que los resultados los envíe comprimidos (comando **compress**), lo cual puede ser útil si los resultados de la búsqueda ordenada son muy extensos. No es necesario especificar ningún sujeto para el correo (Subject), pero si se coloca alguno es ignorado por **archie.** Cualquier comando que aparezca en el cuerpo del correo y que archie no reconozca, es interpretado como un **help** y como parte de la respuesta le será enviado la información que corresponde al help. Los comandos **prog** y **whatis** están igualmente disponibles, el criterio de búsqueda es el de expresión regular de Unix.

3.2.3 Sistema de información en línea de Internet (gopher)

Gopher es un sistema de información que organiza el acceso a los recursos de Internet; utiliza una interfase orientada a menús. Se puede ir navegando a través de los menús, y al conseguir algo de interés, se puede accesar o leer a través de gopher, sin preocuparse por direcciones IP, dominios de nombres o el protocolo necesario para acceder esa información, pues de eso precisamente se ocupa el gopher. Gopher fue desarrollado por la Universidad de Minnesota y realmente todavía está bajo desarrollo, por lo cual las opciones disponibles cambian frecuentemente para proporcionar mejoras a las existentes o para incorporar alguna nueva facilidad.Gopher se ha expandido vertiginosamente en todo el mundo y cada dia aparecen mas servidores gopher integrados al Internet, pues provee una forma sencilla que permite a cualquier institución mostrar al resto del mundo alguna información que se genere internamente.

Para accesar el sistema **gopher** es necesario disponer de un programa **cliente gopher**, que permita contactar a uno de los múltiples **servidores gopher** que existen en el Internet. Obviamente para poder ejecutar el programa cliente es necesario tener cuenta en algún host conectado al Internet y el programa cliente debe estar disponible en ese host. Hay algunos sitios donde es posible acceder **gopher** públicamente por medio de telnet, usando para ello la cuenta **gopher;** a continuación una lista algunos servidores públicos de **gopher**.

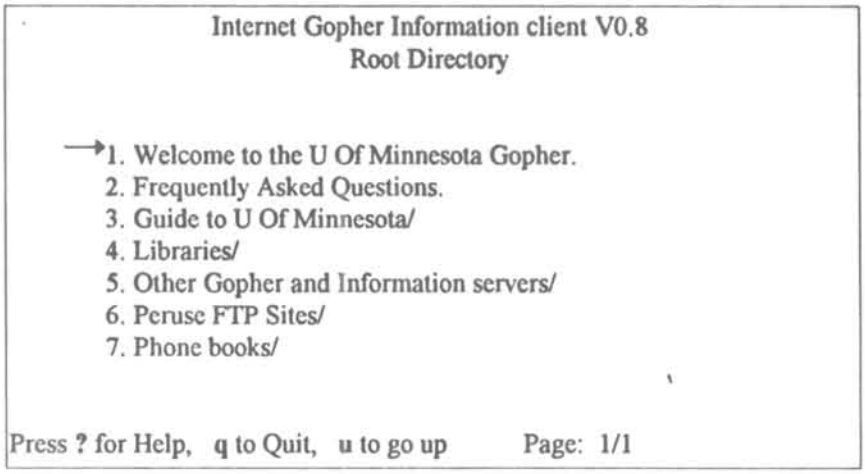
consultant.micro.umn.edu	login: **gopher**
gopher.uiuc.edu	login: **gopher**
info.anu.edu.au	login: **info**
panda.uiowa.edu	login: **panda**
gopher.uwp.edu	login: **gopher**

Cuando se arranca un **gopher cliente**, este contacta a un servidor cercano (en realidad el administrador al instalar la aplicación cliente, le define a que servidor gopher debe contactar primero, en la Universidad de Carabobo siempre se contacta primero el servidor gopher de la Universidad de Minnesota) y le pregunta por su menú principal, el servidor envía el menú principal y el cliente lo envía al computador que se esté usando como terminal. El usuario puede seleccionar cualquier ítem del menú, al hacerlo el **cliente gopher**, le solicita información adicional acerca del ítem seleccionado, al **servidor gopher;** el servidor le dice al cliente que tipo de información representa la selección hecha (es decir, archivo de texto, un directorio, un host, un servidor de páginas blancas, etc.) y adicionalmente le suministra la dirección IP del servidor de ese ítem, el número de puerto a usar o la trayectoria (path) en caso de ser un archivo. Esta información es usada por el **cliente gopher** para escoger la forma de acceder al recurso seleccionado y desde el punto de vista del usuario todo este proceso es transparente, pues el **cliente gopher** escogerá la herramienta apropiada; si el ítem es un archivo, el cliente gopher hace un ftp para traer el archivo hasta el cliente; si el recurso es un host, el cliente contacta a ese host vía una sesión telnet; si el ítem seleccionado es un servicio archie o de páginas blancas, el gopher cliente contactará el servidor que corresponda; de manera que el usuario no tiene que conocer la forma de acceder a ninguno de los recursos que aparezcan en el menú, y ni siquiera el tipo de recurso, pues todo eso lo hace el **gopher cliente**.

Suponiendo que en un servidor Unix existe instalado el cliente gopher, para arrancar el mismo basta con escribir el siguiente comando en respuesta al prompt:

$ gopher

El cliente contacta al servidor gopher que se le haya programado y presenta un menú parecido al que se muestra a continuación:



En este caso, el **cliente gopher,** está programado para contactar como entrada a los servidores gopher al servidor de la Universidad de Minnesota; realmente carece de importancia el lugar por donde se inicie el contacto con el sistema gopher, pues una vez que se está conectado a uno de ellos, se podrá contactar a cualquier otro servidor gopher que esté registrado en el Internet, en cualquier parte del mundo, para lo cual basta con escoger en el menú principal la opción 5 (Other Gopher and Information Servers). Los ítems que aparezcan en el menú terminados en un punto corresponde a archivos de texto, que de ser seleccionados serán transferidos vía ftp hasta el computador donde se esté ejecutando el gopher cliente, para luego ser mostrados por la pantalla del computador o terminal que se esté usando. Los ítems que aparecen terminados en "/" (slash) conducen a otro menú y de ser alguno de ellos seleccionados, conducirá al despliegue del próximo nivel bajo ese menú(submenú). La flecha que aparece a la izquierda de algún ítem (en el ejemplo aparece en el ítem 1), muestra el ítem en el cual se está posicionado y será la selección escogida si se da "enter" o "return"; el usuario puede escoger un ítem distinto al que aparece indicado por la flecha escribiendo, el número del ítem que desea seleccionar. En la parte inferior de la pantalla aparece siempre una pequeña ayuda; una ayuda adicional acerca de todas las opciones disponibles se obtiene con el caracter **?**; la opción **u** permite regresar un nivel en la estructura del menú(regresa al menú anterior).

Describir todas las posibilidades de gopher es muy difícil ya que cada servidor gopher presenta un menú particular, la forma mas fácil de aprender acerca de **gopher** es iniciando una sesión de gopher y comenzar a explorar las distintas opciones que le presente; la opción de ayuda (?) siempre está disponible y en general las opciones y menús se han escogido de manera que intuitivamente se tenga una noción del para que sirven. Una de las opciones mas importantes que siempre está presente en los servidores gopher es la que permite acceso a librerías públicas; por medio de esta opción, el usuario tiene la posibilidad de revisar miles de centros de información bibliográfica dispersos en todo el mundo; muchas veces es posible transferir vía correo electrónico algún resumen de alguna publicación y en muchos casos la publicación completa, sin embargo, esto depende de las políticas que al respecto tenga el proveedor de la información bibliográfica que se esté contactando. Generalmente el acceso es gratuito pero hay algunas librerías cuyo acceso tiene algún costo; es posible que solo sea necesario pagar algunos servicios muy especiales, en todo caso, lo que hay es que experimentar con el gopher y extraer el mayor provecho de las facilidades que presenta.

Otra posibilidad que presentan los gopher en su menú es la de posibilitar la transferencia de archivos vía ftp anónimo; por supuesto que se puede usar directamente la aplicación ftp, sin usar el gopher, pero la organización por menú que posibilita el gopher lo hace mas atractivo, pues uno en general tiene la posibilidad de ir explorando varios servidores los cuales pueden estar organizados por algún criterio en el menú y basta con seleccionar el servidor deseado y el gopher se encarga de contactarlo (vía ftp) y uno puede realizar las transferencias que desee.

Capitulo IV

El presente capítulo tiene por finalidad presentar un resumen de la actividad cumplida para poner en operación un nodo Internet en la Facultad de Ingeniería a través de la red SAICYT, que el CONICIT ha establecido en nuestro país. En los capítulos anteriores se discutió acerca de los protocolos que se usan en el Internet y de las aplicaciones mas importantes que están disponibles en esta red; en este se discutirá acerca de los aspectos relevantes que tienen que ver con la configuración y establecimiento de un nodo Internet, entre estos aspectos se tienen: selección del hardware necesario para el establecimiento del nodo; escogencia de un sistema operativo o plataforma sobre la cual instalar esos protocolos; configurar el software que permite el establecimiento de los servicios básicos discutidos en los capítulos precedentes; conexión del nodo al mundo exterior; definir una "arquitectura" para la red que determinará la forma de asignar direcciones a los hosts adicionales que se interconecten, la forma de hacer enrutamiento y la forma en que los diferentes hosts interactuarán con la red. Básicamente este capítulo comprende los aspectos que se acaban de mencionar.

4.1. Selección del hardware y Sistema Operativo

La escogencia del hardware tiene mucho que ver con la plataforma de software que se ejecutará sobre ese hardware y de los servicios que se aspire proveer. TCP/IP existe para todas las plataformas de software y hardware, sin embargo no todos los servicios pueden ser obtenidos en todas las plataformas. Por ejemplo para MS-DOS y Windows(ambos de Microsoft Corporation), existe software que implementa los protolos TCP/IP, pero básicamente como un cliente y para un solo usuario; hay varios servicios que no pueden darse, precisamente por la naturaleza del sistema operativo

sobre el que se sustenta y algunos solo pueden operar parcialmente. Los servidores de redes LAN, como Netware (de Novell), ahora están incorporando los protocolos TCP/IP, sin embargo, también en forma limitada. La plataforma de software que permite proveer todos los servicios de TCP/IP es Unix, de manera que en vista de que nuestro interés era instalar un nodo capaz de proveer todos los servicios de TCP/IP escogimos como sistema operativo Unix. Unix es soportado por casi cualquier hardware existente; en el Internet es común conseguir que la mayoría de los servidores usan Unix como sistema operativo y estaciones de trabajo SUN como plataformas de hardware, sin embargo, en el caso nuestro no fue posible esta combinación básicamente por limitaciones económicas. Nuestra instalación usa como plataforma de software, Unix (Unix System V, en una versión desarrollada por SCO, Santa Cruz Operation, Inc) y como plataforma de hardware un computador IBM compatible (486).

Unix es un sistema operativo multitarea, multiusuario que nos da la posibilidad de extender todos los servicios de la red a una mayor cantidad de usuarios a un costo relativamente bajo; adicionalmente los servidores Unix son enrutadores (gateways) naturales y esto es un factor importante pues permite el establecimiento de varias redes fisicas con el servidor como medio de enlace entre las diferentes redes.

4.2. Esquema de direccionamiento

Cualquier persona que desee conectar sus redes a Internet debe solicitar ante las autoridades que administran Internet un número para su red. Ya se vio en el capítulo II que existen varios tipos de redes (clase A, B y C), cuya categorización está basada en el número de hosts que esa red tiene capacidad de soportar. En Venezuela el CONICIT tramitó ante las autoridades de Internet la asignación de unos números de redes para el establecimiento de la red SAICYT. Uno de esos números corresponde a la red 150.186 (una red clase B) que tiene capacidad para soportar 65535 hosts. El CONICIT valiéndose del esquema de subredes que se estudió en el capítulo II, le ha asignado a los diferentes Centros de Educación del país una porción de ese espacio

disponible, de manera que estas Instituciones puedan desarrollar redes e integrar las mismas al Internet.

A la UC, se le ha asignado un espacio de direcciones IP, dentro esa red clase B (150.186); la conexión al SAICYT se hace a través de un enrutador CISCO que posee dos interfases: una ethernet, para el desarrollo de la red local y una interfase serial a través de la cual se hace la conexión al SAICYT, usando una línea muerta provista por CANTV que nos conecta permanentemente al nodo Barquisimeto de SAICYT y que es en definitiva nuestro punto de entrada a Internet. Un enrutador tiene conexión por lo menos a dos redes diferentes; el nuestro tiene una de sus interfases conectada a la red 150.186 (la interfase ethernet, hacia el lado interno de la UC) y la otra interfase(la serial, hacia el nodo Barquisimeto), a la red 150.189 (otra red clase B, usada por el CONICIT para enlazar todos los enrutadores que conforman la plataforma comunicacional del SAICYT).

Se había mencionado que los servidores Unix podían servir como enrutadores o gateways naturales, sin embargo, para la instalación del nodo fue necesario la adquisición de un enrutador de propósito específico, en previsión de que el tráfico que se producirá en la medida en que se vaya expandiendo la red amerita un equipo dedicado exclusivamente a las labores de enrutamiento; por otra parte, la comunicación se hace en la modalidad sincrónica bajo un protocolo de bajo nivel conocido como HDLC (High-level Data Link Control) y en general las tarjetas que permiten la comunicación serial en los equipos compatibles son asincrónicas y no soportan el protocolo HDLC. La conexión al SAICYT se ha establecido a través de una línea muerta con Barquisimeto, punto donde el CONICIT tiene instalado un nodo, esto se hizo así en virtud de que para el momento en que se instaló el nodo en la UC, todavía el CONICIT no había instalado el nodo en Valencia (ahora se dispone de un nodo en Valencia, el cual está fisicamente ubicado en las instalaciones del CID), en un futuro la conexión se hará a través del nodo Valencia. La siguiente gráfica ilustra la forma como la UC está conectada al SAICYT.

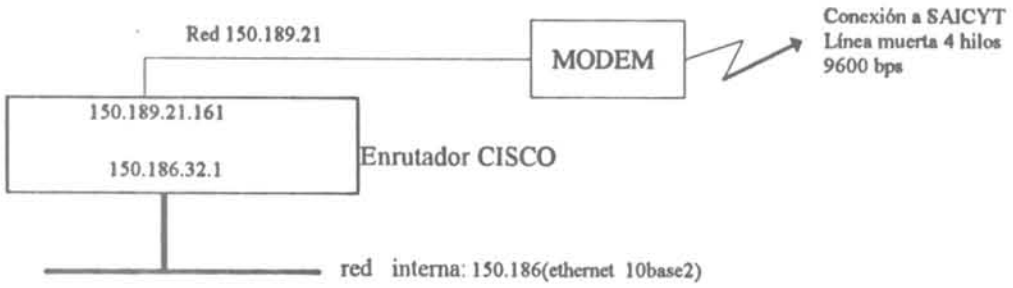


Figura 4.1. Conexión al SAICYT a través del nodo Barquisimeto

Los administradores de la red SAICYT nos asignaron una parte del espacio de direcciones de la red 150.186, para lo cual usaron una submascara de 255.255.255.0 y nos autorizaron a usar números IP comprendidos entre 150.186.32 hasta 150.186.63 (ver figura 4.2); esto puede ser visto como que si nos hubiesen asignado 32 redes clase C, cada una de las cuales puede tener hasta 256 hosts (realmente 254 pues como se vio en el capítulo II hay números con propósitos específicos que no pueden ser asignados a ningún host). Si bien es cierto que la red 150.186 corresponde a una red clase B, lo que significa que los dos últimos bytes de la dirección están destinados para identificar hosts, por medio de la redefinición del espacio de direcciones establecido a través de la submascara, se ha definido que solo el último byte se usará para identificar hosts, pues el penúltimo, al colocar 255 en esa posición en la submascara, será interpretado como parte de la identificación de la red, lo que permite que una misma dirección de red desde el punto de vista del Internet (150.186), sea interpretada internamente (en el dominio Venezuela, ve) como varias redes, en una forma que emula la red clase C de Internet, pero con la ventaja que se había mencionado de que las tablas de los enrutadores de Internet la consideran como una sola red, lo cual es fundamental para mantener un nivel de eficiencia adecuado en los algoritmos encargados de satisfacer las exigencias de la labor de enrutamiento.

Red clase B: 150.186
mascara: 255.255.255.0

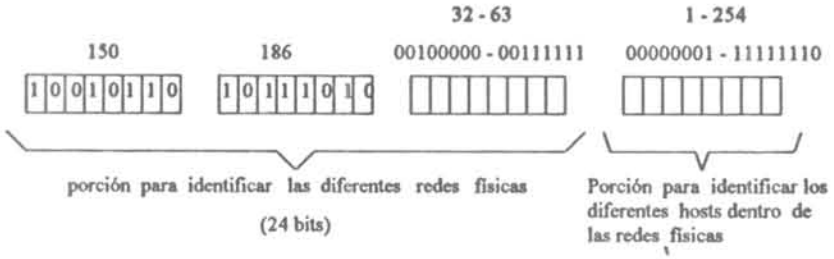
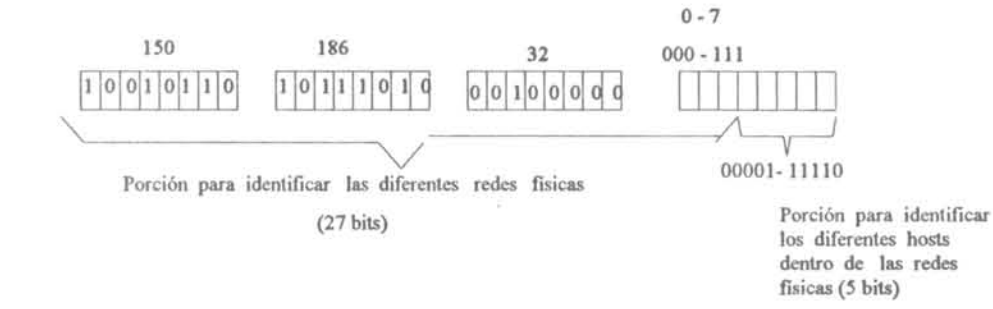


Figura 4.2. Espacio de direcciones asignado a la UC

En resumen tenemos la posibilidad de asignar 8192 números IP entre nuestros computadores, que para efectos del ente asignador (en nuestro caso el CONICIT) estarán distribuidos en 32 redes, cada una de las cuales tendrá hasta 254 hosts. Se puede mantener ese esquema y asignar en principio a cada Facultad una de las subredes asignadas, eso permitiría que cada una de estas Facultades puede establecer una red que podría tener hasta 254 computadores; sin embargo lo mas probable es que cada Facultad desee redefinir el espacio asignado, mediante la misma técnica de la submascara, y adaptarlo a su estructura organizativa.

En la Facultad de Ingeniería se ha redefinido el espacio, para lo cual se ha usado una submascara de 255.255.255.224, esto nos permite disponer de 8 subredes cada una con la capacidad de soportar hasta 30 hosts. La idea es que a cada Escuela se le asigne una de estas subredes (dependiendo de los requerimientos se le podrá asignar otras adicionales); esto se ha pensado así porque en la mayoría de las Escuelas es posible que se vaya a instalar una red local tipo ethernet, bajo Netware (Novell) o cualquier otro sistema operativo para red local; estas redes típicamente soportan hasta 30 computadores por segmento y en general un servidor de red local soporta hasta 4 segmentos; cada segmento se constituiría en una subred, sin embargo, se está suponiendo que inicialmente existirá un solo segmento de red, pero existe la potencialidad de crecer hasta 4 segmentos. A pesar de que como ya se dijo lo mas

probable es que cada Escuela instale una red local como Netware, sería recomendable que se pensara en la posibilidad de instalar un servidor Unix por Escuela de manera de que cada una maneje y administre los servicios que prestará la red, sobre todo lo referente a la asignación de cuentas (login name) y la recepción de correo electrónico, ya que al estar distribuido entre varios servidores la prestación de los servicios, los requerimientos para el mismo serán menores y la confiabilidad del sistema será mayor. La siguiente figura muestra la distribución del espacio de redes que se ha planificado para la Facultad de Ingeniería.



Red clase B: 150.186
mascara: 255.255.255.0
Subred: 150.186.32.0
Mascara: 255.255.255.224
subred: 150.186.32.0 (Conexión SAICYT)
subred: 150.186.32.32 (Estación de Promoción y Des.)
subred: 150.186.32.64 (Eléctrica segmento I)
subred: 150.186.32.96 (Eléctrica segmento II)
subred: 150.186.32.128 (Mecánica)
subred: 150.186.32.160 (CIMBUC)
subred: 150.186.32.192 (Control de Estudio)
subred: 150.186.32.224 (Industrial)

Subred: 150.186.33.0
Mascara: 255.255.255.224
subred: 150.186.33.0 (IMYCA)
subred: 150.186.33.32 (Química)
subred: 150.186.33.64 (Civil)
subred: 150.186.33.96 (Dpto. Comp.)
subred: 150.186.33.128
subred: 150.186.33.160
subred: 150.186.33.192
subred: 150.186.33.224

Figura 4.3 Distribución de las subredes 150.186.32 y 33 en la Facultad de Ingeniería.

Como se puede apreciar en la figura, en la Facultad de Ingeniería se ha requerido tomar dos de las subredes asignadas (la 150.186.32 y la 150.186.33) y se ha dividido el espacio de manera de asignarle a cada Escuela una subred. La Escuela de Eléctrica, que ya tenía instalada una red local basada en un servidor Netware, con una infraestructura instalada también para cubrir toda el área de la Escuela, con dos segmentos, se le asignaron dos subredes, la 150.186.32.64 y la 150.186.32.96, una para cada segmento. En la mayoría de las Escuelas todavía no están instaladas las

redes locales previstas; Química está en proceso de instalación de su red local (basada en Netware) y Mecánica, que actualmente tiene un solo equipo conectado a la red, posiblemente desarrolle su red interna en corto plazo. La figura 4.4 muestra la estructura actual de la red de Ingeniería, en la cual se puede apreciar la existencia de dos enrutadores, uno comprado específicamente para esos propósitos (Router CISCO) y el otro es un PC XT (8088), corriendo un programa(el PCroute) que le permite comportarse como un enrutador. En el Imyca está instalado el servidor Unix (thor.uc.ve) encargado de proveer los servicios de correo electrónico, transferencia de archivos gopher etc. Disponibles en la red; ese equipo además de prestar los servicios mencionados está funcionando también como enrutador (gateway) pues como se aprecia en la figura posee interfases conectadas en tres redes distintas (150.186.32.0, 150.186.32.32 y 150.186.33.0), en cada una de las cuales tiene una dirección IP(150.186.32.2, 150.186.32.34 y 150.186.33.2). Esta estructura va a cambiar cuando se desarrolle completamente el proyecto de la red de Ingeniería, de manera de darle mayor confiabilidad y segmentar el tráfico de la red para evitar su congestionamiento.

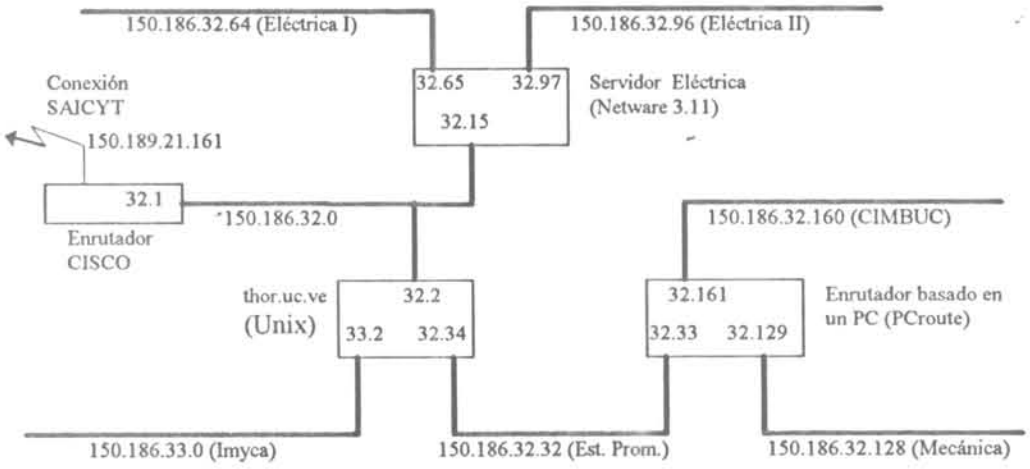


Figura 4.4 Extensión de la red Internet en Ingeniería

4.3. Configuración de algunos servicios

El sistema operativo Unix tiene la potencialidad de soportar todos los servicios disponibles en la red Internet, pero algunos de ellos deben ser configurados; especial consideración debe darse a la configuración del servidor de dominio de nombre (DNS), debido a que el mismo es fundamental para la operación del correo electrónico. El correo electrónico debe ser configurado después del dominio de nombre y existen dos niveles de configuración: uno que corresponde al nivel del agente encargado de liberar el correo hasta sus destinos(**MTA**, **M**ail **T**ransport **A**gent) para lo cual se ha escogido en nuestra instalación el **MMDF** (**M**ultichannel **M**emorandum **D**istribution **F**acility); y el otro que corresponde al llamado agente del usuario (**MUA**, **M**ail **U**ser **A**gent), para lo cual se ha escogido el **elm**.

4.3.1. Configuración del Servidor de dominio de nombre (DNS)

La principal tarea del servidor de dominio de nombre es responder consultas acerca de nombres y direcciones de hosts. Existen varios tipos de servidores de dominio de nombre, dependiendo de su autoridad y autonomía en el dominio en el cual es servidor. A continuación una breve descripción de los tipos de servidores disponibles

- Servidores maestros: Corresponde a la máxima autoridad para el dominio del cual es servidor y se supone que mantiene toda la información correspondiente a su dominio. Existen dos tipos de servidores maestros: el primario, el cual carga o mantiene toda la información del dominio en un archivo en su propio disco duro y el servidor secundario, el cual recibe la información del dominio de un servidor maestro primario; básicamente este tipo de servidor lo que hace es que al momento de arrancar (hacer boot), solicita los datos del dominio a un servidor maestro primario y después periódicamente chequea para ver si requiere actualizar su data.

- Servidores remotos: Este tipo de servidor permite que estaciones de trabajo o equipos con baja capacidad presten el servicio de nombre al resto de software de red que se esté ejecutando en ellas y que requieran el uso del servicio. En este caso todas las consultas son servidas por un servidor de nombre que está corriéndose en otra máquina de la red.
- Servidores esclavos: Un servidor esclavo está en capacidad de responder ciertas consultas acerca del dominio pero que cuando hay alguna que no puede satisfacer solo puede interactuar con una lista predefinida de servidores, en vez de consultar directamente a los servidores maestros del dominio.

Como parte de la configuración del servicio de nombre está el definir el tipo de servidor que se usará. Existe un archivo fundamental para el establecimiento del servicio de nombre; este archivo por defecto se supone que está ubicado en el directorio **/etc.** (este es el directorio donde Unix normalmente guarda los archivos que tienen que ver con la administración del sistema y los servicios en general) en un archivo llamado **named.boot**; existen adicionalmente otros archivos que necesita el servidor de nombre cuya ubicación es precisamente definida en el archivo **named.boot**; en este archivo **named.boot** se define también el tipo de servidor (primario, secundario, esclavo o remoto). Los archivos que necesita el servidor de nombre son los siguientes (la ubicación exacta de estos archivos en el árbol de directorios, se define en el archivo **named.boot)**:

- **named.local**: En este archivo se especifica la dirección de la interfase del host local (también llamada loopback; en el capítulo 2 se vio que el número IP 127.0.0.1 se usaba generalmente para estos propósitos). El nombre de este archivo puede ser redefinido en **named.boot**.
- **named.hosts**: En este archivo se colocan todos los datos referentes a las máquinas del dominio. Este nombre puede ser redefinido en el archivo **named.boot**.
- **named.rev:** En este archivo se especifican las direcciones de los hosts del dominio en la forma pedida por un dominio especial llamado IN-ADDR.ARPA.

Este dominio fue creado con el propósito de permitir la conversión inversa, o sea de número a nombre. Este nombre puede ser redefinido en el archivo **named.boot,**

A continuación se presenta el contenido de estos archivos en la instalación actual de la Facultad de Ingeniería. El archivo **named.boot** que por defecto debiera estar ubicado en el directorio **/etc.**, se le hizo un enlace simbólico para ubicarlo en el directorio **/var/named** (para hacer este enlace simbólico, se crèan los directorios **/var** y **/var/named**, luego con el editor se escribe el archivo **named.boot**, y luego se ejecuta el comando **ls -s /var/named/named.boot /etc./named.boot**; de esta forma el archivo **/etc./named.boot**, el cual lo necesita el servidor de nombre para arrancar, es fisicamente el mismo que el archivo **/var/named/named.boot)**

/var/named/named.boot (es el mismo archivo /etc./named.boot

```
:Boot file for Primary Name Server

directory       /var/named
;
; type          domain                  source file or host
;
cache           .                       named.cache
primary         uc.ve                   uc.ve.info
primary         186.150.in-addr.arpa    ucveinfo.rev
primary         loopback                named.local
primary         127.in-addr.arpa        named.locrev
;
```

Obsérvese que la primera línea del archivo define que todos los archivos relacionados con el servicio de dominio de nombre se encuentran en el directorio **/var/named** (directory /var/named). Luego se define el dominio para el cual es servidor y el tipo de servidor; se define como servidor maestro primario para el dominio **uc.ve** y como el servidor primario requiere un archivo de donde sacar los datos se le especifica que el archivo **uc.ve.info**, contiene la información del dominio (este nombre es relativo al directorio especificado con la instrucción directory, de manera que en este caso el nombre absoluto del archivo será **/var/named/uc.ve.info**). También se definen el

nombre del archivo **named.rev** declarándose que el mismo se llamará realmente **ucveinfo.rev**.

/var/named/uc.ve.info (equivale a named.hosts)

```
;       @(#)uc.ve.info  1.1     (UC)    13/10/93
;
@       IN      SOA     thor.uc.ve.    postmaster.thor.uc.ve. (
                        931225  ; serial number - UPDATE AFTER EVERY CHANGE
                        3600    ; refresh (60 minutes)
                        1800    ; retry  (30 minutes)
                        3600000         ; expire
                        3600 )  ; minimum
                        IN      NS      thor.uc.ve.
                        IN      NS      dino.conicit.ve.
                        IN      NS      nisc.jvnc.net.
nisc.jvnc.net.          IN      A       128.121.50.7
dino.conicit.ve.        IN      A       150.188.1.10
uc-gateway              IN      A       150.186.32.1
estcprom.uc.ve.         IN      A       150.186.34.1
localhost               IN      A       127.0.0.1
uc.ve.                  IN      MX      10      thor.uc.ve.
thor.uc.ve.             IN      A       150.186.32.2
                        IN      A       150.186.33.2
                        IN      A       150.186.32.34
2.32.186.150            IN      PTR     thor.uc.ve.
2.33.186.150            IN      PTR     thor.uc.ve.
34.32.186.150           IN      PTR     thor.uc.ve.
meca01.uc.ve.           IN      A       150.186.32.130
meca01.                 IN      CNAME           meca01.uc.ve
imyca.uc.ve.            IN      A       150.186.33.1
Imyca.                  IN      CNAME           imyca.uc.ve.
1.33.186.150            IN      PTR             imyca.uc.ve.
```

En este archivo se declara en principio un registro de recursos (SOA, Start of Autority) que define el inicio de la autoridad para el dominio y el hosts dentro del dominio en donde residen los datos; en este caso para el dominio **uc.ve** se especifica que los datos están en el host **thor.uc.ve**. En este registro también se especifica la dirección electrónica de la persona responsable del dominio; en este caso se dice que es **postmaster@thor.uc.ve**. Se define el tiempo de validez de la información suministrada, la frecuencia del refrescamiento y el intervalo para reintentos después de una transferencia fallida, toda esta información tiene que ver con la existencia de

servidores secundarios para el dominio. También se define el tiempo mínimo para el TTL (tiempo de vida) de los registros de recursos.

Después del registro de autoridad (SOA), las primeras cosas que se deben declara son los servidores de nombre conocidos; esto se hace con un registro de autoridad definido como NS (name server); en este caso se están declarando como servidores de nombre a **thor.uc.ve**, **dino.conicit.ve** y **nisc.jvnc.net**. Se define también a **thor.uc.ve** como un hosts capaz de servir de intercambiador de correo en el dominio **uc.ve** (esto es fundamental para el manejador de correo electrónico), lo cual se hace con el registro **uc.ve MX 10 thor.uc.ve** (el número 10 tiene importancia solo cuando existen varios equipos con la capacidad de servir como intercambiadores de correo, es un valor para definir preferencias entre los distintos intercambiadores de correo). Los siguientes registros del archivo básicamente permiten declarar los nombres de los hosts y las direcciones IP que los mismos tienen asociados; se definen también las direcciones en el formato IN-ADDR.ARPA, lo cual se hace con los registros identificados como del tipo PTR. Obsérvese que los hosts que tienen mas de una interfase (caso de thor.uc.ve), por cada interfase aparece un registro del tipo dirección (A), que corresponde a la definición de los distintos números IP que corresponden a cada una de estas interfase conectadas a diferentes redes. Los registros del tipo CNAME se usan para definir alias para los hosts; por ejemplo el host meca01.uc.ve se le define como alias meca01 y al host imyca.uc.ve se le define como alias Imyca. Un host puede tener varios alias y un mismo nombre puede corresponder a distintas direcciones IP; es el caso por ejemplo del equipo thor.uc.ve, que aparece tres veces con tres direcciones distintas, lo cual significa que si bien el equipo posee tres números IP distintos para cada interfase, el nombre es el mismo para las tres interfases. Es importante la terminación en punto de los nombres de los alias porque si no se hace normalmente el software de dominio de nombre automáticamente le agrega el dominio (en este caso uc.ve) y en general eso no es lo que uno desea que ocurra cuando declara un alias, pues lo que se busca es acortar los nombres. En este archivo se pueden definir también, listas de correos de los servicios disponibles en el servidor,

esto último se hace con un tipo de registro identificado como **WKS**, pero en el ejemplo no aparece ninguno de estos registros.

4.3.2. Configuración del enrutador en el servidor Unix

La versión de Unix que se tiene provee dos programs diseñados par el intercambio de información referente al enrutamiento y al mantenimiento de las tablas de enrutamiento del host y que son los que permiten que el servidor Unix además de los servicios que presta se convierta en un enrutador. Estos programas son excluyentes esto es si se usa uno no puede usarse el otro. El primero de ellos implementa el protocolo RIP, el cual es un protocolo que se usa exclusivamente para intercambio de información de enrutamiento en el interior de sistemas autónomos(el programa que lo implementa se llama **routed**), basado en el vector distancia; el otro protocolo es mas general y permite el intercambio de información de enrutamiento tanto con los enrutadores pertenecientes al sistema autónomo como los que pertenecen a otros sistemas autónomos. Nuestra instalación básicamente requiere de un enrutamiento interior y los intercambios de información con los otros sistemas autónomos se hacen a través del enrutador especializado de que disponemos (router CISCO) y no con el servidor de la red; de manera, que se habilitará el programa que implementa el RIP. Para hacer esto basta con asegurarse que **no** exista el archivo **/etc./gated.conf**, pues de existir este archivo, el enrutamiento se hará automáticamente con el otro protocolo(programa **gated**). El algoritmo de enrutamiento permite que se le definan rutas estáticas para lo cual provee el comando **route**; una ruta muy importante de declarar es la ruta por defecto, esta es la ruta que se escogerá si no se tiene información de como llegar a una red específica. Esta ruta puede ser declarada con el comando **route**, pero como es deseable que cada vez que el sistema arranque tenga definido una ruta por defecto, es conveniente incluir esta ruta en el archivo que configura el **TCP (/etc/tcp)**, lo cual se puede hacer editando dicho archivo y colocando la siguiente línea, justo después de que se haya determinado que se usará el protocolo RIP (el archivo /etc/tcp, es un programa escrito en un lenguaje interpretado,

que entre otras cosas, determina el tipo de algoritmo de enrutamiento que usará, para lo cual hace un chequeo para ver si existe el archivo /etc/gated.conf; si existe este archivo **no** usa RIP):

route add default 150.186.32.1 0

Con esta instrucción se le está especificando al algoritmo de enrutamiento que cuando no tenga conocimiento de como alcanzar una red específica, envíe la información al host 150.186.32.1, el cual en nuestro caso es el enrutador CISCO. El cero que aparece al final de la instrucción es la distancia en saltos, a la cual está ese host de la interfase que corresponde a el servidor, un cero significa que está en la misma red (en efecto el servidor tiene una interfase cuya dirección IP es 150.186.32.2, lo que corresponde a la misma red; recuerde que RIP es un protocolo basado en el vector distancia, y esta se mide en saltos o sea redes que hay que atravesar para llegar a un determinado destino).

4.3.3. Configuración del correo electrónico

La configuración del correo electrónico que se tratará acá será la referente al agente deliverador de correo (que en la instalación nuestra es el MMDF); el sistema provee un programa utilitario que facilita la configuración y básicamente lo que hay que hacer es prepara una lista de las cosas que ese programa pedirá como datos y el mismo se encarga automáticamente de realizar todos los ajustes necesarios en los archivos de configuración de manera de particularizarlos a las condiciones de la instalación sobre la que funcionará. Es importante resaltar que este programa **no** debe ser ejecutado sin haber configurado primero el servidor de nombre, pues parte de su actividad se apoya en este servicio. A continuación una lista de la información requerida para la configuración del servicio de correo a nivel de agente deliverador:

- **Nombre del host**: esto corresponde al nombre que se le dará al equipo en donde se ejecutará el servicio de correo; se refiere exclusivamente a la parte

del nombre sin incluir el dominio o subdominio. En nuestra instalación el servidor se llama **thor**.

- **Dominio:** Esto corresponde al nombre del dominio. En nuestro caso ya se ha dicho que el dominio es **uc.ve**.

- **Nombre Completo (Fully-qualified host name)**: Esto se refiere al nombre del host incluyendo la parte que corresponde al dominio o sea en nuestro caso sería: **thor.uc.ve**

- Los siguientes datos tienen que ver con la forma funcional del sistema de correo. Por ejemplo en algunas instalaciones acostumbran a que las direcciones de correos no incluyan el nombre del host a la cual la cuenta pertenece; si esto se hace habrá que tener un archivo de alias completo en el cual se determine a partir de la cuenta, el host a la que pertenece. Nosotros preferimos por los momentos no ocultar los nombres de los hosts en la dirección pues en la actualidad el único equipo que provee los servicios es thor. También se acostumbra a redireccionar el correo electrónico dirigido al super usuario (root) o administrador del sistema a otro código, así como también el correo dirigido al usuario administrador del correo (cuenta mmdf); esto si no se desea sencillamente no se especifica ningún nombre a quien redirigir el correo de estos usuarios. Existe por convención un usuario ficticio llamado postmaster, al momento de configurar el correo es necesario asociar este usuario a algún usuario real.

- **Smart hosts(badhosts):** Es posible especificar al agente deliverador de correo un host al cual dirigir los correos cuyas máquinas destinos sean desconocidas; este es el llamado smart host. En nuestra instalación se definió como smart host a **dino.conicit.ve**.

- **smart hosts (baduser)**: Es posible especificar también un host al cual enviar los correos recibidos para usuarios que sean desconocidos para el servidor (no tengan una cuenta definida); en este caso se ha definido a

dino.conicit.ve como el host al cual enviar los correos de los usuarios desconocidos.

Para configurar el correo hay que entrar en sesión (hacer login) con la cuenta del administrador del correo (**mmdf**) y ejecutar el comando **mkdev mmdf**, con lo cual se arranca el programa configurador y comenzará a pedir los datos que se acaban de ver; al finalizar de introducir los datos el servicio de correo a nivel de agente deliverador queda configurado.

4.4. Configuración de otros servicios

Los otros servicios como el **gopher** el **elm** etc., traen programas para configurarlos, que van solicitando la información requerida. El problema con estas aplicaciones de dominio público es en la compilación. Este software generalmente se desarrolla para estaciones de trabajo como SUN e IBM y los programas preparados para la compilación del software han sido probados en esas máquinas y es posible que al tratar de instalar esas en una plataforma Unix distinta, como es el caso nuestro, haya que hacerle algunos pequeños ajustes, que en general se limitan a definir algunos encabezados o los directorios donde se encuentran las librerías requeridas por la aplicación; en cualquier caso es difícil predecir lo que haya que modificar y por eso en este trabajo no se ha incluido nada al respecto. Ya varios de estas aplicaciones se han logrado compilar e instalar en el servidor, específicamente el **gopher** (en sus dos versiones, cliente y servidor) y el correo electrónico **elm**.

Bibliografía

COMER, Douglas E. Internetworking With TCP/IP, Vol I, Prentice Hall, 1991.

CLARK D. D. Subnetwork Addressing Scheme, rfc0932, Jun. 1985

CROKER S. D. Computer Network, rfc016, May. 1971.

NCR Corporation. NCR System 3000 WIN-TCP Administrator's Guide, NCR Corporation, 1992.

POSTEL, J. B. User Datagram Protocol, rfc0788, Aug. 1980.

----------. Internet Protocol, rfc0791, Sep. 1981.

----------. Transmission Control Protocol, rfc0793, Sep. 1981.

----------. File Transfer Protocol, rfc0959, Oct. 1985.

PLUMMER, D. C. Ethernet Address Resolution Protocol, rfc0826, Nov. 1982.

SOCOLOFSKY, T. J. and Kale, C. J. TCP/IP Tutorial, rfc1180, Jan. 1991.

STALLINGS, William. Handbook of Computer Communications Standards, Vol 2, second edition, Howard W. SAMS & Company, 1989.

----------. Handbook of Computer Communications Standards, Vol 3, second edition, Howard W. SAMS & Company, 1989.

THE SANTA CRUZ OPERATION Inc. SCO TCP/IP Runtime System for SCO UNIX Systems User's and Administrator's Guide, SCO, 1992.

----------. SCO UNIX Operating System System Administrator's Guide, SCO, 1992.

APENDICE A

Comandos básicos de Unix

Caracter	Función
* ? [ ]	Metacaracteres o caracteres de sustitución que proveen una forma corta de especificar patrones para los nombres de un conjunto de archivos.
&	Le indica al shell que el comando que se le está dando lo debe ejecutar en background. El & debe ser el último caracter en la línea de comando, inclusive despues de los diferentes redireccionamientos.
\	Anula el significado de caracter especial, si es que tiene alguno, del caracter que sigue al slash invertido (\). Los caracteres que son anulados por el slash son: * ? [] & ; > < >> |
'... '	Anula el significado de todos los caracteres especiales que se encuentren dentro de las comillas simples. Anula inclusive el significado de separador de argumentos que se le atribuye al espacio.
"..."	Anula el significado especial del espacio y de todos los caracteres especiales que se encuentren dentro de las dobles comillas con la excepción del $ y la comilla invertida (`).
>	Permite redireccionar la salida de un comando a un archivo. Por defecto, casi todos los comandos envian su salida a la pantalla y toman sus datos del teclado, el simbolo > se usa para indicar al shell que la salida del comando va a ser dirigida al archivo que se especifique despues del simbolo >.
<	Permite redireccionar la entrada a un comando. Con este simbolo se le avisa al shell que los datos que el comando esperaba por el teclado los tomará del archivo que sigue al simbolo <.
>>	Redirecciona la salida de un comando a un archivo (similar al >), pero en este caso se le avisa al shell que la salida del comando se va a agregar al final de un archivo existente. Si el archivo especificado no existe el comportamiento del shell es identico al obtenido cuando se usa >.
|	Permite encadenar o entubar comandos de manera que la salida de uno sirva como entrada al otro.
`...`	Permite que la salida de un comando, en este caso representado por los tres puntos, pueda ser usada como argumento para otro comando. La secuencia seguida por el shell es mas o menos la siguiente: Ejecuta el o los comandos que estan entre los acentos graves y los resultados del comando, son colocados en el sitio donde estaba el comando.
;	Permite separar multiples comandos en una línea de comando.

Caracteres con significado especial para el Shell

Comando	Opción(s)	Argumento(s)
mkdir	ninguna	nombre(s) de directorio(s)

Descripción: Permite crear un nuevo directorio. El nombre del directorio puede ser especificado en forma absoluta, lo que permite que el subdirectorio sea creado en cualquier lugar del file system, distinto al directorio de trabajo, sin tener que cambiarse de directorio. El usuario debe tener permiso(de escritura), para poder crear un subdirectorio en el directorio que se especifique.

Ejemplos:

$mkdir letras	Crea el directorio "letras" en el directorio actual de trabajo.
$mkdir /bin/letras	Crea el directorio "letras", dentro del directorio /bin. Se supone que /bin ya existia.
$mkdir letras bin1 /bin/bin11	Crea los directorio "letras"," bin1" y" /bin/bin1".

Comando	Opción(s)	Argumento(s)
rmdir	ninguna	nombre(s) de directorio(s)

Descripción: Remueve el directorio o directorios especificados como argumentos. Los directorios deben estar vacios (sin ningun archivo o subdirectorio adscrito a el). Para poder remover un directorio el usuario debe tener permiso para hacerlo (con la excepción del super usuario root).

Ejemplos:

$rmdir letras	Remueve el directorio "letras" del directorio de trabajo actual.
$rmdir /bin/bin11	Remueve el directorio "bin1" del directorio "/bin".
$rmdir letras bin1 /bin/bin11	Remueve los directorios "letras", "bin1" y "bin11", del directorio de trabajo y del directorio /bin respectivamente.

Comando	Opción(s)	Argumento(s)
cd	ninguna	nombre de directorio (opcional)

Descripción: Permite ubicarse en cualquier posición en el file system o sea permite cambiar el directorio de trabajo. Si no se especifica el nombre del directorio a donde se desea ir, el comando asume que el usuario se desea ubicar en el directorio HOME corrientemente asignado. Para ubicarse en un determinado directorio, el usuario debe tener permiso para hacerlo.

Ejemplos:

$cd letras	Se posiciona en el directorio letras. Es equivalente a: $cd ./letras
$cd ../bin1	Se posiciona en el directorio bin1, el cual se encuentra en el directorio del corriente.
$cd /bin/bin11	Se pociona en el directorio /bin/bin. (direccion absoluta).
$cd	Se posiciona en el directorio HOME del usuario que ejecuto el comando.
$cd /	Se posiciona en el directorio RAIZ.

Comando	Opción(s)	Argumento(s)
ls	-a -l y otras(ver manual)	nombre(s) de directorio

Descripción: Lista los nombres de los archivos y subdirectorios que pertenecen al directorio especificado. Si no se especifica argumento se lista el contenido del directorio de trabajo.

opciones:

-a	Permite listar todos los archivos, incluyendo los que comienzan por . (punto).
-l	Lista el contenido del directorio en formato largo, mostrando caracteristicas del archivo o directorio tales como: permisos, tamano, fecha ultima actualizacion etc.

Ejemplos:

$ls	Muestra los archivos del directorio de trabajo, menos los que comienzan por . (punto).
$ls -a /	Muestra todos los archivos del directorio RAIZ, hasta los que comienzan por punto.
$ls -al	Muestra todos los archivos del directorio de trabajo, con todos sus atributos. (Permisos, dueno, grupo, tamano, fecha creacion ,fecha ultima modificacion).

Comando	Opción(s)	Argumento(s)
lc, lx	-r -t -u -q -b y otras(ver manual)	nombre(s) de directorio(s)

Descripción: Los comandos lc y lx son sinonimos del comando ls. Realmente son el mismo comando ls con una opcion especifica activada. lc lista los archivos que se encuentran en el directorio especificado, en un formato multicolumna, cada una ordenada verticalmente(es equivalente a un ls -C). lx lista los archivos bajo el o los directorios especificados, en un formato multicolumna, con las salidas ordenadas horizontalmente en vez de por columna (es equivalente a un ls -x). Las opciones son validas para el ls.

Opciones

- -r Invierte el orden del ordenamiento de los archivos listados. Por defecto el orden es descendente, esta opcion lo hace ascendente.
- -t Muestra los archivos ordenados por fecha de creacion, segun el criterio fijado (ascendente o descendente).
- -u Muestra los archivos ordenados por fecha de ultimo acceso, segun el criterio solicitado
- -q Muestra cada caracter no visible presente en el o los nombres de los archivos, como interrogaciones (?).
- -b Muestra cada caracter no visible presente en el o los nombres de los archivos, en forma de un numero octal, equivalente a la representacion ASCII del caracter (formato \ddd).

Comando	Opción(s)	Argumento(s)
pwd	ninguna	ninguno

Descripción: Muestra el nombre completo (ubicacion absoluta dentro del file system) del directorio en el cual se este corrientemente trabajando.

Ejemplo:

$pwd Muestra por pantalla el directorio actual de trabajo.

Comando	Opción(s)	Argumento(s)
echo	ninguna	cadena de caracteres

Descripción: Muestra los argumentos separados por espacios en blanco; al final termina con un new line. Existen algunos caracteres que se interpretan en forma especial cuando aparecen como argumento:

\b backspace	\t tab
\c No escribe new line al final del mensaje.	\v tab vertical.
\f form feed (salto de pagina)	\\ slash invertido
\n new line	\0n n es un numero octál que representa el caracter que se desea enviar.
\r carriage return	

Ejemplo:

$echo "Esta linea deje el cursor al final de este texto (no new line) \c"

Comando	Opción(s)	Argumento(s)
more	-r y otras (ver manual)	nombre(s) de archivo(s)

Descripción: Permite listar, en forma paginada, el contenido de un archivo. Despues de cada pagina el listado del archivo se suspende hasta que el usuario pulse un caracter, algunos de los cuales tienen significado especial para el more. Por ejemplo, el RETURN hace que se muestre solo una linea mas del archivo; la barra espaciadora hace que se muestre una pantalla mas; q o Q indica que el usuario desea suspender la ejecucion del more, etc. El comando va indicando el porcentaje que ha listado del archivo(en caracteres).

Opciones:

- -r Permite que los caracteres de control contenidos en el archivo sean listados como ^ caracter.

Ejemplos: $more cap[123] Lista del directorio actual, los archivos "cap1", "cap2" y "cap3".

$more -r /etc/gettydefs Lista "/etc/gettydefs", mostrando los caracteres de control que existan en el archivo.

Comando	Opción(s)	Argumento(s)
cat	-v -t -e y otras (ver manual)	nombre(s) de archivo(s)

Descripción: Permite listar el contenido del archivo o archivos que se le pasan como argumento. Como casi todos los comandos en Unix, por defecto, la entrada se asume por teclado y la salida por la pantalla. El archivo o archivos especificados son listados sin ningún formateo. La salida puede ser redireccionada a un archivo o entubada (pipe) a otro comando.

Opciones:

- -v Muestra los caracteres especiales(control) con la excepción de tab, new line y form feed.
- -t Muestra los tab como ^I y los form feed como ^L.
- -e Muestra un caracter $ al final de cada linea.

Nota: Las opciones -t y -e deben ser usadas en conjunto con la opción -v por que si no no tienen ningún efecto (son ignoradas).

Ejemplos:

$cat -vet doc1	Lista por pantalla el archivo doc1. Los caracteres especiales (control) son mostrados, los tab los muestra como ^I y al final de cada línea muestra un $.
$cat doc1 > doc1.old	Lista el archivo doc1 hacia el archivo doc1.old. En este caso se comporta como un copy
$cat doc1 doc1.old	Lista los archivos doc1 y doc2 por pantalla.
$cat doc1 doc2 > sumdoc	Concatena los archivos doc1 y doc2 en el archivo sumdoc.
$cat > copycon.msdos	Copia todo lo que se introduzca por el teclado al archivo copycon.msdos, hasta que se tipee un ^D.

Comando	Opción(s)	Argumento(s)
pg	-p y otras (ver manual)	nombre(s) de archivo(s)

Descripción: Muestra en forma paginada, el contenido del o los archivos especificados. Despues de mostrar cada página, se queda esperando por comandos del usuario; existe la posibilidad de retroceder páginas, avanzar páginas, encontrar un patrón específico en las páginas que faltan o en las ya listadas, etc. A continuación una lista de los comandos reconocidos por pg y su correspondiente función:

Comando	función	Comando	función
h	Muestra los comandos disponibles.	$	Lista la última página del archivo que se esté listando.
q o Q	Termina el comando.	/patrón	Busca hacia adelante, en el archivo corriente, el patrón de caracteres que se especifique.
l	Lista la próxima línea de texto.		
d o ^d	Lista media página siguiente.		
. o ^l	Repite la página corriente.		
f	Salta la próxima página, continúa el listado en la que sigue a esta.	?patron	Busca hacia atras, en el archivo corriente, el patrón que se especifique.
n	Comienza a listar el próximo archivo que se especificó.	!coman.	Ejecuta cualquier comando de Unix, inclusive el mismo pg.
p	Lista el archivo anterior al que se esta mostrando.		

Opcion: -p string Hace que pg use como prompt el string especificado con la opción. (%d hace que se use como prompt, el numero de la página que se está presentando)

Ejemplos

$pg cap1	Lista por pantalla el archivo "cap1" paginado.
$pg cap[1357]	Lista por pantalla los archivos "cap1", "cap3", "cap5" y "cap7" en forma paginada.
$pg cap[!1357]	Lista por pantalla los archivos "cap" menos cap1, cap3, cap5 y cap7.
$pg *	Lista por pantalla todos los archivos del directorio actual de trabajo, paginados.

Comando	Opción(s)	Argumento(s)
cp	ninguna	arch1 arch2 (arch(s) directorio)

Descripción: Permite hacer una copia de arch1 y llamarla arch2; arch1 permanece inalterado. Permite tambien copiar uno o mas archivos a un directorio. Con cp no se pueden copiar directorios. Si el archivo arch2 ya existe, el mismo es borrado y luego se hace la copia (sobreescribe); esto lo hace sin advertirle al usuario, de manera que es necesario tener mucho cuidado al especificar el nombre del archivo hacia donde se va a copiar.

Ejemplos:

$cp datos1 datos1.old	Copia el archivo "datos1" al archivo "datos1.old". "datos1.old" es creado.
$cp cap*r ../revisados	Copia del directorio actual al directorio "revisados", todos los archivos que comiencen por "cap y terminen en r.. "revisados" es un subdirectorio del directorio actual(existia).
$cp cap*[!r] ../pendientes	Copia del directorio actual al directorio pendientes, los archivos que comiencen por "cap" y no terminen en "r".

Comando	Opción(s)	Argumento(s)
mv	ninguna	arch1 arch2 (arch(s) directorio)

Descripción: Permite cambiar el nombre de un archivo; arch1 se le pone como nombre arch2. Puede ser usado para mover un archivo o archivos de un directorio a otro. Si existe un archivo que ya tenía el nombre especificado como arch2, el mismo es sobreescrito sin preguntarle al usuario.

Ejemplos:

$mv cap1 cap2	Mueve "cap1" a "cap2". "cap1" desaparece del directorio de trabajo.
$mv cap*r ../revisados	Mueve del directorio actual al directorio revisados, todos los archivos que comiencen por "cap" y terminen en "r". "revisados" es un directorio adscrito al padre del actual.
$mv cap*[!r] ../pendientes	Mueve del directio actual al directorio "pendientes"(ubicado en el padre del actual), todos los archivos que comiencen por "cap" y no terminen en "r".

Comando	Opcion(s)	Argumento(s)
copy	-a -o -m -r -n -v	fuente(s) destino

Descripción: Permite copiar grupos de archivos. Se diferencia del comando cp, en que con el se pueden copiar directorios y subdirectorios, con todos los archivos que estos contengan. Mediante la invocacion del comando con las opciones apropiadas, el usuario puede controlar completamente la forma en que se hara la copia. Este comando es valido solo para el Unix de SCO.

Opciones:

- -a El usuario es interrogado antes de realizar el copy; Si la respuesta no comienza por "y" el copy no se realiza.
- -o Forza a que los archivos se copien con los mismos permisos y modo de la fuente. Si no se especifica esta opcion, los archivos copiados tendran como dueño al que ejecuta el comando (requiere que el usuario tenga permiso para leer la fuente).
- -r Forza a que cada directorio(subdirectorio) en la fuente, sea explorado recursivamente para copiarlo al destino(creando los subdirectorios o directorios necesarios).
- -v Forza a que todas las acciones que esté realizando el comando sean informadas por la salida estandard.
- -n Forza a que el archivo destino sea nuevo(no exista en el destino); si el archivo existe en el destino se deja inalterado (no se copia).

Ejemplos:

$copy -r dir1 dupdir1	Copia dir1(y todo su contenido incluyendo subdirectorios) en el directorio dupdir1.
$copy -rnv dir1 dupdir1	Copia dir1(y todo su contenido) en dupdir1. Si algún archivo existía previamente en dupdir1, no lo copia. Todas las acciones son informadas por la salida estandard.

Comando	Opción(s)	Argumento(s)
diskcp	-f -d -r -48ds9 -96ds9 -96ds15 -135ds9 -135ds18	ninguno

Descripción: Permite duplicar un diskette. diskcp se usa para hacer una imagen exacta de un diskette fuente. La fuente y el destino no necesariamente tienen que ser del mismo tipo. En máquinas con dos unidades de floppy, diskcp coloca en forma inmediata el contenido de la fuente en el destino; en máquinas con un solo floppy (puede que fisicamente posea los dos floppy, pero no se le especificó la opción -d), el contenido de la fuente es copiado temporalmente en el disco duro (en el directorio /tmp) y vaciado luego en el diskette destino.

Opciones:

-f	Se usa para indicar a diskcp que debe formatear el diskette destino antes de hacer el copy. En este caso es obligatorio especificar tambien la densidad a la que se formateara.
-d	Se usa para especificar a diskcp que use los dos floppy que tiene el computador. Por defecto se copia del drive0 al drive1.
-r	Se usa para indicar a diskcp que el disco fuente esta ubicado en el drive1 y no en el cero, como por defecto, se supone.
-48ds9	Se usa para especificar diskette de 5 1/4" baja densidad (360Kb).
-96ds9	Se usa para especificar diskette de 5 1/4" doble densidad (720 Kb).
-96ds15	Se usa para especificar diskette de 5 1/4" alta densidad (1.22Mb).
-135ds9	Se usa para especificar diskette de 3 1/2" baja densidad (720 Kb).
-135ds18	Se usa para especificar diskette de 3 1/2" alta densidad (1.44 Mb).

Ejemplos:

$diskcp -df -96ds15	Copia del drive0 al drive1. Formatea el diskette de 5 1/4", para alta densidad, antes de copiar.
$diskcp -f -96ds15	El diskette fuente y el destino se colocaran en el mismo drive, el 0; diskcp avisa cuando colocar en ese drive el diskette fuente y el destino. Se formatea el destino para alta densidad (5 1/4").
$diskcp -rf -135ds18	El diskette fuente y el destino se colocarán en el mismo drive, en este caso el 1 (por la opción -r). El destino se formatea para alta densidad (diskette 3 1/2").
$diskcp -rd	Copia del drive1 al drive0. Se supone que el diskette destino ha sido previamente formateado.

Comando	Opción(s)	Argumento(s)
format	-n -v -f -q -i interleave <device>	ninguno

Descripción: El comando format permite formatear diskettes para ser usados en unix, como un medio de almacenamiento, usando los comandos disponibles para tal efecto: tar, cpio, dd. Si no se especifica ninguna opción, el diskette se formateará según las especificaciones que aparezcan en el archivo /etc/default/format, en el cual se define el dispositivo a usar (que tiene implicito ya la densidad a la que se formateará) y si se verifica o no el diskette despues de formateado. Generalmente se usa como defecto el drive0, en alta densidad.

Los usuario de MSDOS, acostumbrados a usar comandos como copy o xcopy, rename, etc, notarán que no pueden usar los comandos unix equivalentes para hacer copias de archivos a diskettes(cp y copy); esto se debe a que en unix el format solo se ocupa de demarcar los sectores y tracks en la superficie magnética, (formateo a bajo nivel), en cambio en MSDOS, el comando format en un diskette, ademas de demarcar los sectores y tracks en la superficie magnética, escribe en el mismo la estructura de archivos (file system) del sistema operativo DOS. En Unix es necesario usar otro comando para preparar el diskette con la estructura de archivos del sistema unix(file system), y usarlo en una manera similar a la que se usa bajo MSDOS; ademas, es necesario "montar" el file system del diskette.

Opciones:

<device>	Especifica el dispositivo a ser formateado. El nombre del dispositivo ya tiene implicito la densidad y el drive. Solo se pueden formatear los dispositivos row (no buferizados); por ejemplo un dispositivo valido sería /dev/rfd096ds15, que corresponde a un diskett de alta densidad 5 1/4"(96ds15), ubicado en el drive 0 (rfd0).
-v	Se usa para especificar verificación del diskette despues del formateado. La opción -n, especifica que no se haga la verificación despues del formateo (unix mo acepta diskettes con sectores malos).

Ejemplos:

$format /dev/rfd1135ds18	Formatea un diskette de 3 1/2", ubicado en el drive1, para alta densidad (1.44Mb).
$format /dev/rfd0135ds9	Formatea un diskette de 3 1/2", ubicado en el drive0, para baja densidad(720Kb).
$format /dev/rfd096ds15	Formatea un diskette de 5 1/4", ubicado en el drive0, para alta densidad(1.22Mb).

Comando	Opción(s)	Argumento(s)
rm	-f -i -r	archivo(s) o directorios

Descripción: Permite remover uno o mas archivos. Los archivos removidos no pueden ser recuperados (por lo menos hasta el momento), por lo que el usuario debe ser muy cuidadoso con este comando.

Opciones

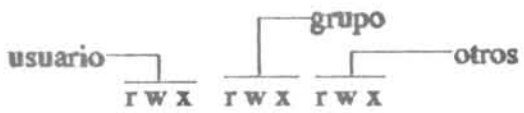
- -f Causa la remoción de todos los archivos especificados aun cuando tengan protección contra escritura, sin avisar de ello al usuario(esto último lo puede hacer el dueño o el super usuario).
- -i Avisa al usuario antes de borrar un archivo especificado que tenga protección contra escritura.
- -r Esta opción hace que recursivamente se borre el contenido completo de cualquier directorio especificado, incluyendo cualquier subdirectorio(vacio o no). Hay que ser extremadamente cuidadoso al usar esta opción, pues es posible borrar un directorio, a pesar de que no esté vacio, sin informar o pedir permiso al usuario para hacerlo.

Ejemplos:

$rm pendientes/cap*	Remueve todos los archivos que comiencen por "cap" en el directorio pendientes
$rm -i *	Remueve todos los archivos del directorio de trabajo, pregunta para remover los protegidos.
$rm -f *.bak	Remueve incondicionalmente del directorio actual todos los archivos que terminen en ".bak".
$rm -r dir1 dir2	Remueve los directorios dir1 y dir2 del directorio actual de trabajo. Si los directorios dir1 y dir2 contienen archivos o subdirectorios, los mismos serán recursivamente borrados.

Comando	Opción(s)	Argumento(s)
chmod		nombre(s) de archivo(s) o direct.

Descripción: Permite cambiar los permisos a los archivos o directorios. Unix tiene permisos separados para el usuario (u), el grupo (g) y otros(o). Los permisos son para leer, escribir o ejecutar un determinado archivo o directorio. Los permisos se pueden especificar en forma absoluta o en forma simbólica. Los permisos son descritos por tres secuencias de caracteres para cada categoría tal como se muestra a continuación:



Categorías: usuario o dueño (u), grupo (g), otros usuarios (o)

Numeración octal

0 = 000	1 = 001
2 = 010	3 = 011
4 = 100	5 = 101
6 = 110	7 = 111

El primer campo de cada categoria corresponde al permiso de lectura para esa categoría; la presencia de una r en dicho campo indica autorización para leer el archivo. Un - en cualquiera de los campos indica negación del permiso correspondiente. El segundo campo de cada categoría corresponde al permiso de escritura; una w en dicho campo indica autorización de escritura para la categoria donde esté. El tercer campo de cada categoría corresponde al permiso de ejecución; una x en dicho campo indica autorización para ejecutar ese archivo por parte de la categoria donde aparezca.

La forma simbólica de usar chmod es utilizando los símbolos asignados a cada categoría (u, g, o) y los símbolos usados para especificar el tipo de permiso (r, w, x) precedidos de un signo - si se desea negar dicho permiso.

Un método alterno es el llamado octal. En este caso se usan tres números octales (van de 0 a 7) para especificar las distintas autorizaciones. El primer numero octal es para el usuario, el segundo es para el grupo y el tercero para otros. La representacion binaria del numero octal permite negar o dar la autorización: un uno (1) en una determinada posición autoriza, un cero (0) lo niega.

Ejemplos:

$chmod 744 miarchivo	Este comando le da permiso de lectura, escritura y ejecucion para el dueno de "miarchivo"; Para el grupo y los otros usuarios les da solo permiso de lectura.
$chmod go-wx miarchivo	En este caso se le niega el permiso de escritura y ejecucion a los del grupo y al resto de los usuarios.
$chmod 511 miarchivo	Prohibe la escritura a todos(dueno, grupo y otros), permite que todos lo ejecuten. Solo el dueno lo puede leer.

Comando	Opción(s)	Argumento(s)
lpstat	-t -u -o y otras (ver manual)	depende de la opción

Descripción: Permite averiguar informacion acerca del estatus del servicio de impresoras. Si no se suministra ninguna opcion, el comando retorna el estatus de todos los requerimientos de impresion hechos por el usuario mediante un comando lp.

Opciones

-t Muestra toda la información de estatus del sistema de impresión.

-u [lista] Muestra el estatus de todas las salidas solicitadas por los usuarios especificados en lista (lista es un conjunto de códigos de usuarios).

-o [lista] Muestra el estatus de las salidas solicitadas. Lista es un conjunto de nombres de printers y números asignados por el sistema de impresión, cuando se da un comando lp.

Ejemplo: $lpstat -u "user1 user2 " Muestra los requerimientos de impresion de los usuarios "user1" y "user2".

Comando	Opción(s)	Argumento(s)
lp	-d -m -n -title -w -o	archivo(s)

Descripción: Permite sacar un listado de un archivo o archivos por la impresora (copia en papel).

Opciones:

-d <destino> Permite que se especifique la impresora por donde se desea el listado (destino es el nombre de una impresora del sistema; ver lpstat para determinar el nombre de las impresoras disponibles). Si no se usa esta opción la salida es por la impresora declarada como default.

-m Envia un mensaje via el correo de Unix (mail), para avisar que ya la impresión está completa.

-n <numero.> Imprime tantas copias como especifique número. Por defecto número es 1.

-title Escribe en la página inicial del listado (la que corresponde al banner), el título del archivo que se mandó a imprimir. Por defecto no se imprime el título.

-w Esta opción hace que se envie un mensaje al terminal, para avisar que el archivo ya se imprimió.

-o Esta opción permite especificar otra serie de opciones. La sintaxis es especificar la opción -o seguida de las subopciones de la misma. Estan disponibles las siguientes subopciones:

nobanner No se imprime el banner. nofilebreak No al salto de página entre archivos. width=número Fija el ancho de página a "número". lpi=número Fija el numero de líneas por pulgada a "número". cpi=número Fija el número de caracteres por pulgada a "número".

Ejemplos:

$lp cap1 Por la impresora definida como default, imprime el archivo "cap1".

$lp -m revisados/cap[123] Imprime los archivos "cap1, cap2 y cap3", avisando por mail al terminar la impresión.

$lp -d hplaser revisados/cap* Imprime, por la impresora "hplaser", los archivos especificados.

$lp -o nobanner cpi=12 arch1 Imprime el archivo "arch1" del directorio actual, por la impresora default; No imprime el banner y usa una densidad de caracteres de 12 por pulgada.

$lp -title VER-2 arch1 Imprime "arch1" por la impresora default. Se imprime el título especificado (VER-2), en forma de banner, en la página inicial del listado.

Comando	Opción(s)	Argumento(s)
cancel	ninguna	id o un nombre de printer

Descripción: Permite cancelar un requerimiento de impresión hecho con un comando lp. El argumento puede ser un número identificador de un requerimiento de impresión, id (id lo regresa el comando lp al aceptar una solicitud de impresión de uno o mas archivos) o puede ser el nombre de una impresora (use lpstat -s para averiguar los nombres de las impresoras de que dispone el sistema). Si el argumento es un id, la impresión es cancelada aun cuando se esté imprimiendo. Si el argumento un nombre de impresora, se cancela la impresión que se esté sacando al momento de ejecutarse el cancel, sin importar cual era su id.

Ejemplos: $cancel lp-23 Elimina de la cola de impresión la solicitud identificada por el número lp-23.

$cancel lpt 1 Cancela la impresión que se esté realizando en la impresora lpt1.

Comando	Opción(s)	Argumento(s)
ps	-e -a -t -p -u -f	depende de la opcion

Descripción: Permite obtener información de los procesos activos. Sin opciones, se muestra una lista de los procesos asociados al terminal de donde se ejecuta el comando, mostrando un reporte que incluye el PID, el terminal, el tiempo de ejecución acumulado y el nombre del comando. Las opciones aceptan argumentos, los cuales pueden ser suministrados separando unos de otros por comas o encerrandolos entre comillas dobles y dentro de ellas, separados unos de otros por espacios en blanco o comas.

Opciones:

-e	Muestra información acerca de todos los procesos que se están ejecutando.
-a	Muestra información acerca de todos los procesos mas frecuentemente requeridos, excepto aquellos no asociados con un terminal.
-t <lista>	Muestra información de los procesos asociados con el o los terminales dados en lista. Lista es un conjunto de terminales (tty001, tty1a, tty1b, etc).
-p <lista>	Muestra información acerca de los procesos cuyos PID esten incluidos en lista. Lista es un conjunto de identificadores de procesos (PID).
-u <lista>	Muestra información acerca de los procesos en ejecución perteneciente a los códigos de usuarios especificados en lista. Lista es un conjunto de códigos de usuarios o nombres de login.
-f	Produce un listado completo (full listing) con toda la información asociada a un proceso(UID, PID, PPID, C, STIME, TTY, TIME, COMMAND).

Ejemplos:

ps	Muestra las tareas activas del usuario que da el comando.
ps -u "user1 user2"	Muestra las tareas activas de los usuarios "user1" y "user2".
ps -t "tty01 tty1a tty1b"	Muestra las tareas activas en los terminales "tty01", "tty1a" y "tty1b".
ps -fe	Muestra en forma detallada todos los procesos que están en ejecución.
ps -p 103,210,330	Muestra información de los procesos "103", "210" y "330".
ps -fa	Muestra información detallada de todos los procesos asociados a terminales.

Comando	Opción(s)	Argumento(s)
kill	-senal	PID

Descripción: Este comando se utiliza para enviar a los procesos, señales tales como: fin de programa, perdida de suministro, errores, etc. Las señales que se pueden enviar, dependen de la implementación y las mismas se encuentran en un archivo llamado /usr/include/signal.h; muchas de ellas solo pueden ser enviadas si se es super usuario. La señal -9 siempre termina con el proceso al que se le mande.

Ejemplo:

$kill -9 234	Mata el proceso 234 (PID =234), perteneciente al usuario que da el comando.
$kill -15 133	Le manda una señal tipo 15 al proceso 133 (PID = 133).
#kill -9 0	Mata todos los procesos, incluyendo el sistema operativo. Solo la puede ejecutar un super usuario.

Comando	Opción(s)	Argumento(s)
nohup	ninguna	línea de comandos

Descripción: Permite que se ejecuten comandos ignorando desconexiones y logoff. Los procesos hijos normalmente mueren si sus padres mueren; al apagar un terminal se produce una señal de hangup que mata el getty asociado a dicho terminal. Por otra parte si uno se sale de sesión (logoff) todos los procesos iniciados bajo dicha sesión mueren (aun los que se ejecutan en background). nohup permite que un proceso se ejecute aun cuando su padre muera.

Ejemplo:

$nohup find /etc -size +100 -print &	Ejecuta en backgrond el comando find(por el & al final); si se muere el shell o se sale de la sesión, el comando continúa ejecutandose (resultados van a $HOME/nohup.out)

Comando	Opción(s)	Argumento(s)
find	-name -perm -size -user -group -atime -mtime -ctime -exec -ok -type -print	depende de la opción o condición

Descripción: Encuentra archivos que satisfagan una determinada condición, descendiendo recursivamente por cada path-name-list. El formato general del find es: find path-name-list expresión. Para los efectos de su ejecución, la busqueda comienza en el o los directorio(s) especificado(s) en path-name-list y va revisando cualquier subdirectorio que dependa directa o indirectamente de dicho directorio, para determinar si dentro de ellos existen archivos que satisfagan la condición indicada por expresión.

Algunas expresiones involucran un argumento "n", que corresponde a un número entero. Ese número puede estar precedido o no de un signo mas (+) o de un signo menos(-); el signo mas expresa la condición mayor que "n"; el signo menos expresa la condición menor que "n"; sin signo expresa la condición igual que "n".

Las opciones o condiciones primarias pueden ser combinadas usando los operadores " !" y "-o" (operaciones equivalentes a not y or respectivamente). ! es la negacion y -o es el Or de dos expresiones primarias. La funcion and está implícita y no necesita operador; la opción and tiene mayor prioridad que el or(-o) pero menor que el not(!). Cuando se use la opción -o las expresiones deben estar encerradas entre "\(" y "\)".

Opciones o condiciones primarias:

-name <arch>	La condición es conseguir el(los) archivo(s) que tenga(n) por nombre "arch".
-size n[c]	La condición es conseguir los archivos que tengan n bloques de longitud(512 bytes por bloque) si n es seguido por una c, el tamaño es en caracteres.
-perm <onum>	La condición es, archivos o directorios cuyo set de permisos se corresponda con "onum", el cual representa un número octal de tres dígitos, que define los permisos(ver chmod).
-user <uname>	La condición es, archivos cuyo dueño sea "uname".
-group <grp>	La condición es, archivos cuyo dueño(s) pertenezcan al grupo "grp".
-atime n	La condición es, archivos que hayan sido accesados hace "n" dias.
-mtime n	La condición es, archivos que hayan sido modificados hace "n" dias.
-ctime n	La condición es, archivos cuyo último cambio fue hace "n" dias(cambio implica creación o modificación).
-exec cmd	Ejecuta el comando shell "cmd" (cmd es cualquier comando). El argumento que se le coloca a "cmd" es {}, el cual es reemplazado automaticamente, al momento de su ejecución, por la salida producida por el find, producto de las condiciones de busqueda especificadas. El comando "cmd" debe ser terminado por un punto y coma, desprovisto de significado especial (el punto y coma se usa en el shell como un separador, que permite multiples comandos en una línea; para obviar esta interpretación, es necesario precederlo de algún caracter que anule este significado, lo que puede hacerse con la secuencia \;)
-ok cmd	Esta opcion opera en forma similar a -exec; se diferencia de este en que antes de ejecutar "cmd", se pide confirmación al usuario, el cual respondera con "y" si desea que "cmd" se ejecute.
-type x	La condición es, archivos cuyo tipo sea el especificado por "x". Los posibles valores de "x" son: bb para archivo especial de bloque; c, para archivo especial de caracter; d para directorio; f para archivo corriente y l para enlace simbólico.
-print	Hace que se muestre por el terminal, todos los nombres absolutos, de los archivos que satisfacen la condicion dada. Si no se especifica esta opcion, el find actua silenciosamente y no informa nada al usuario.

Ejemplos:

$find / -name termcap -print

A partir del directorio raiz (/), busca los archivos con nombre "termcap". Muestra por el terminal la ubicación absoluta de esos archivos.

$find /u/msince -name "*.dbf" -size +100 -print

Busca, a partir del directorio /u/msince (recursivamente), los archivos que terminen en ".dbf" y tengan un tamaño mayor de 100 bloques(50 KB). Se muestra por el terminal, los nombres absolutos de los archivos que satisfacen esas condiciones.

$find /u/msince -type d ! -perm 770 -exec chmod 770 {} \;

Busca, a partir del directorio /u/msince(recursivamente), todos los archivos cuyo tipo sea directorio y que no tengan el set de permisos 770 (lectura, escritura y ejecución para el dueno y el grupo). Los archivos encontrados, se les aplica el comando "chmod 770", el cual le establece un nuevo set de permisos; esto se hace sin preguntar al usuario.

$find / \(-name core -o -name "*.bak"\) -ctime +2 -ok rm {} \;

Busca, a partir del directorio raiz, los archivos cuyo nombre sea "core" o terminen en ".bak" y que no hayan sido cambiados en los últimos dos dias. Los que satisfagan esas condiciones serán removidos, previa autorización del usuario.

Comando	Opción(s)	Argumento(s)
wc	-l -w -c	nombre(s) de archivo(s)

Descripción: Permite contar los caracteres, las palabras y las líneas en el o los archivos que se le dan como argumento. Cuando no se suministra ninguna opción, el comando nos reporta las tres cuentas (caracteres, palabras y líneas).

Opciones:

- -l Cuenta el número de lineas en el o los archivos especificados.
- -w Cuenta el número de palabras en el o los archivos especificados.
- -c Cuenta el número de caracteres en el o los archivos especificados.

Ejemplos:

$wc -wc memo1	Cuenta cuantas palabras y cuantos caracteres hay en el archivo "memo1".
$ls | wc -l	Cuenta cuantos archivos hay en el directorio de trabajo.
$wc memo2	Cuenta el número de lineas, palabras y caracteres que hay en el archivo memo2.

Comando	Opción(s)	Argumento(s)
cut	-c<lista> ; -f<lista> [-d][-s]	nombre de archivo

Descripción: Permite extraer columnas de una tabla o extraer campos de un conjunto de líneas presentes en un archivo. Los campos que se especifiquen pueden ser de longitud fija (opción -c) o pueden variar de longitud de línea a linea y estar definidos por un delimitador de campo(opción -f).

Opciones:

- -c<lista> Donde lista debe estar pegado de la c y define el rango de donde se hará la extracción; por ejemplo -c1-40, para extraer los primeros 40 caracteres de cada linea.
- -f<lista> Donde lista es un conjunto de campos separados por el caracter que se de como separador con la opción -d(por defecto el separador es el tab).

Ejemplos:

$cut -c1,4-10,30-45 datos1	Extrae el primer caracter, los caracteres entre 4 y 10 y entre 30 y 45 de cada línea del archivo datos 1.
$cut -d: -f1,5 /etc/passwd	Muestra los campos 1 y 5 del archivo /etc/passwd. El caracter dos puntos (:) es el separador de campo (la opción -d lo define).

Comando	Opción(s)	Argumento(s)
grep	-b -c -i -n y otras	patron y nombre(s) de archivo(s)

Descripción: Busca dentro de los archivos pasados como argumentos, las lineas que contengan el patrón especificado. Si se especifica mas de un archivo, el nombre del archivo donde aparezca el patrón tambien es reportado.

Opciones:

- -b Muestra cada línea que contenga el patrón, precedida por el número de bloque.
- -c Muestra solamente una cuenta de las líneas que contienen el patrón.
- -i Ignora la distinción entre las minúsculas y las mayúsculas.
- -n Precede cada línea que contenga el patrón, con su número de línea dentro del archivo.

Ejemplos:

$grep -n 3546578 dependencia	Busca el patrón dado (3546578) en el archivo "dependencia". Indica la línea o líneas donde se encuentra el patrón dentro del archivo.
$grep -in "pedro perez" julio	Busca el patrón (pedro perez) en el archivo "julio", sin distinguir minúsculas de mayúsculas. Muestra el número de línea donde localiza el patrón en el archivo.

APENDICE B

Comandos del editor vi de Unix

Last line	Reading input for : / ? or !; terminate by typing a carriage return; an interrupt cancels termination.

COMMAND SUMMARY

In the descriptions, <cr> stands for carriage return and <ESC> stands for the escape key.

Sample commands

← ↓ ↑ →	arrow keys move the cursor
h j k l	same as arrow keys
i*text*<ESC>	insert *text*
cw*new*<ESC>	change word to *new*
ea*s*<ESC>	pluralize word (end of word; append **s**; escape from input state)
x	delete a character
dw	delete a word
dd	delete a line
3dd	delete 3 lines
u	undo previous change
ZZ	exit *vi*, saving changes
:q!<cr>	quit, discarding changes
/text<*cr*>	search for *text*
^U ^D	scroll up or down
:*cmd*<cr>	any *ex* or *ed* command

Counts before vi commands

Numbers may be typed as a prefix to some commands. They are interpreted in one of the following ways.

line/column number	**z G |**
scroll amount	**^D ^U**
repeat effect	most of the rest

Interrupting, canceling

<ESC>	end insert or incomplete cmd
<Del>	(delete or rubout) interrupts
^L	reprint screen if <Del> scrambles it
^R	reprint screen if ^L is → key

File manipulation

ZZ	if file modified, write and exit; otherwise, exit
:w<cr>	write back changes
:w!<cr>	forced write, if permission originally not valid
:q<cr>	quit
:q!<cr>	quit, discard changes
:e *name*<cr>	edit file *name*
:e!<cr>	reedit, discard changes
:e + *name*<cr>	edit, starting at end
:e +*n*<cr>	edit starting at line *n*
:e #<cr>	edit alternate file
:e! #<cr>	edit alternate file, discard changes
:w *name*<cr>	write file *name*
:w! *name*<cr>	overwrite file *name*
:sh<cr>	run shell, then return
:! *cmd*<cr>	run *cmd*, then return
:n<cr>	edit next file in arglist
:n *args*<cr>	specify new arglist
^G	show current file and line
:ta *tag*<cr>	position cursor to *tag*

In general, any *ex* or *ed* command (such as *substitute* or *global*) may be typed, preceded by a colon and followed by a carriage return.

Positioning within file

^F	forward screen
^B	backward screen
^D	scroll down half screen
^U	scroll up half screen
*n***G**	go to the beginning of the specified line (end default), where *n* is a line number
/*pat*	next line matching *pat*
?*pat*	previous line matching *pat*
n	repeat last / or ? command
N	reverse last / or ? command
/*pat*/+*n*	*n*th line after *pat*
?*pat*?–*n*	*n*th line before *pat*
]]	next section/function
[[	previous section/function

(	beginning of sentence
)	end of sentence
{	beginning of paragraph
}	end of paragraph
%	find matching ( ) { or }

Adjusting the screen

^L	clear and redraw window
^R	clear and redraw window if ^L is → key
z<cr>	redraw screen with current line at top of window
z-<cr>	redraw screen with current line at bottom of window
z.<cr>	redraw screen with current line at center of window
/*pat*/z–<cr>	move *pat* line to bottom of window
z*n* .<cr>	use *n*-line window
^E	scroll window down 1 line
^Y	scroll window up 1 line

Marking and returning

``	move cursor to previous context
''	move cursor to first non-white space in line
m*x*	mark current position with the ASCII lower-case letter *x*
`*x*	move cursor to mark *x*
'*x*	move cursor to first non-white space in line marked by *x*

Line positioning

H	top line on screen
L	last line on screen
M	middle line on screen
+	next line, at first non-white
–	previous line, at first non-white
<cr>	return, same as +
↓ or j	next line, same column
↑ or k	previous line, same column

Character positioning

^	first not white-space character
0	beginning of line
$	end of line
h or →	forward
l or ←	backward
^H	same as ← (backspace)
space	same as → (space bar)
f*x*	find next *x*
F*x*	find previous **x**
t*x*	move to character prior to next *x*
T*x*	move to character following previous *x*
;	repeat last **f F t** or **T**
,	repeat inverse of last **f F t** or **T**
n|	move to column *n*
%	find matching ({) or }

Words, sentences, paragraphs

w	forward a word
b	back a word
e	end of word
)	to next sentence
}	to next paragraph
(	back a sentence
{	back a paragraph
W	forward a blank-delimited word
B	back a blank-delimited word
E	end of a blank-delimited word

Corrections during insert

^H	erase last character (backspace)
^W	erase last word
erase	your erase character, same as **^H** (backspace)
kill	your kill character, erase this line of input
****	quotes your erase and kill characters
<ESC>	ends insertion, back to command mode
<Del>	interrupt, terminates insert mode
^D	backtab one character; reset left margin of *autoindent*
^^D	caret (^) followed by control-d (^D); backtab to beginning of line; do not reset left margin of *autoindent*
0^D	backtab to beginning of line; reset left margin of *autoindent*
^V	quote non-printable character

Insert and replace

a	append after cursor
A	append at end of line
i	insert before cursor
I	insert before first non-blank
o	open line below
O	open above
r*x*	replace single char with *x*
R*text*<ESC>	replace characters

Operators

Operators are followed by a cursor motion, and affect all text that would have been moved over. For example, since **w** moves over a word, **dw** deletes the word that would be moved over. Double the operator, e.g., **dd** to affect whole lines.

d	delete
c	change
y	yank lines to buffer
<	left shift
>	right shift
!	filter through command

Miscellaneous Operations

C	change rest of line (**c$**)
D	delete rest of line (**d$**)
s	substitute chars (**cl**)
S	substitute lines (**cc**)
J	join lines
x	delete characters (**dl**)
X	delete characters before cursor (**dh**)
Y	yank lines (**yy**)

Yank and Put

Put inserts the text most recently deleted or yanked; however, if a buffer is named (using the ASCII lowercase letters **a** - **z**), the text in that buffer is put instead.

3yy	yank 3 lines
3yl	yank 3 characters
p	put back text after cursor
P	put back text before cursor
"*x***p**	put from buffer *x*
"*x***y**	yank to buffer *x*
"*x***d**	delete into buffer *x*

Undo, Redo, Retrieve

u	undo last change
U	restore current line
.	repeat last change
"*d***p**	retrieve *d*th last delete

APENDICE C

Números de puertos asignados en Internet
(Well Known Ports)

PORT NUMBERS

Ports are used in the TCP [45,106] to name the ends of logical connections which carry long term conversations. For the purpose of providing services to unknown callers, a service contact port is defined. This list specifies the port used by the server process as its contact port. The contact port is sometimes called the "well-known port".

To the extent possible, these same port assignments are used with the UDP [46,104].

To the extent possible, these same port assignments are used with the ISO-TP4 [64].

The assigned ports use a small portion of the possible port numbers. The assigned ports have all except the low order eight bits cleared to zero. The low order eight bits are specified here.

Port Assignments:

Decimal	Keyword	Description
0	Reserved	
1		TCPMUX TCP Port Service Multiplexer
2-4	Unassigned	
5	RJE	Remote Job Entry
7	ECHO	Echo
9	DISCARD	Discard
11	USERS	Active Users
13	DAYTIME	Daytime
15	Unassigned	
17	QUOTE	Quote of the Day
19	CHARGEN	Character Generator
20	FTP-DATA	File Transfer [Default Data]
21	FTP	File Transfer [Control]
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer
27	NSW-FE	NSW User System FE
29	MSG-ICP	MSG ICP
31	MSG-AUTH	MSG Authentication
33	DSP	Display Support Protocol
35		any private printer server
37	TIME	Time
39	RLP	Resource Location Protocol
41	GRAPHICS	Graphics
42	NAMESERVER	Host Name Server
43	NICNAME	Who Is
44	MPM-FLAGS	MPM FLAGS Protocol
45	MPM	Message Processing Module [recv]
46	MPM-SND	MPM [default send]
47	NI-FTP	NI FTP
49	LOGIN	Login Host Protocol
51	LA-MAINT	IMP Logical Address Maintenance
53	DOMAIN	Domain Name Server

55	ISI-GL	ISI Graphics Language
57		any private terminal access
59		any private file service
61	NI-MAIL	NI MAIL
63	VIA-FTP	VIA Systems - FTP
65	TACACS-DS	TACACS-Database Service
67	BOOTPS	Bootstrap Protocol Server
68	BOOTPC	Bootstrap Protocol Client
69	TFTP	Trivial File Transfer
71	NETRJS-1	Remote Job Service
72	NETRJS-2	Remote Job Service
73	NETRJS-3	Remote Job Service
74	NETRJS-4	Remote Job Service
75		any private dial out service
77		any private RJE service
79	FINGER	Finger
81	HOSTS2-NS	HOSTS2 Name Server
83	MIT-ML-DEV	MIT ML Device
85	MIT-ML-DEV	MIT ML Device
87		any private terminal link
89	SU-MIT-TG	SU/MIT Telnet Gateway
91	MIT-DOV	MIT Dover Spooler
93	DCP	Device Control Protocol
95	SUPDUP	SUPDUP
97	SWIFT-RVF	Swift Remote Vitural File Protocol
98	TACNEWS	TAC News
99	METAGRAM	Metagram Relay
101	HOSTNAME	NIC Host Name Server
102	ISO-TSAP	ISO-TSAP
103	X400	X400
104	X400-SND	X400-SND
105	CSNET-NS	Mailbox Name Nameserver
107	RTELNET	Remote Telnet Service
109	POP2	Post Office Protocol - Version 2
110	POP3	Post Office Protocol - Version 3]
111	SUNRPC	SUN Remote Procedure Call
113	AUTH	Authentication Service
115	SFTP	Simple File Transfer Protocol
117	UUCP-PATH	UUCP Path Service
119	NNTP	Network News Transfer Protocol
121	ERPC	Encore Expedited Remote Proc. Call
123	NTP	Network Time Protocol
125	LOCUS-MAP	Locus PC-Interface Net Map Server
127	LOCUS-CON	Locus PC-Interface Conn Server
129	PWDGEN	Password Generator Protocol
130	CISCO-FNA	CISCO FNATIVE
131	CISCO-TNA	CISCO TNATIVE
132	CISCO-SYS	CISCO SYSMAINT
133	STATSRV	Statistics Service
134	INGRES-NET	INGRES-NET Service
135	LOC-SRV	Location Service
136	PROFILE	PROFILE Naming System
137	NETBIOS-NS	NETBIOS Name Service
138	NETBIOS-DGM	NETBIOS Datagram Service
139	NETBIOS-SSN	NETBIOS Session Service

140	EMFIS-DATA	EMFIS Data Service
141	EMFIS-CNTL	EMFIS Control Service
142	BL-IDM	Britton-Lee IDM
143	IMAP2	Interim Mail Access Protocol v2
144	NEWS	NewS
145	UAAC	UAAC Protocol
146	ISO-TP0	ISO-IP0
147	ISO-IP	ISO-IP
148	CRONUS	CRONUS-SUPPORT
149	AED-512	AED 512 Emulation Service
150	SQL-NET	SQL-NET
151	HEMS	HEMS
152	BFTP	Background File Transfer Program
153	SGMP	SGMP
154	NETSC-PROD	NETSC
155	NETSC-DEV	NETSC
156	SQLSRV	SQL Service
157	KNET-CMP	KNET/VM Command/Message Protocol
158	PCMail-SRV	PCMail Server
159	NSS-Routing	NSS-Routing
160	SGMP-TRAPS	SGMP-TRAPS
161	SNMP	SNMP
162	SNMPTRAP	SNMPTRAP
163	CMIP-Manage	CMIP/TCP Manager
164	CMIP-Agent	CMIP/TCP Agent
165	XNS-Courier	Xerox
166	S-Net	Sirius Systems
167	NAMP	NAMP
168	RSVD	RSVD
169	SEND	SEND
170	Print-SRV	Network PostScript
171	Multiplex	Network Innovations Multiplex
172	CL/1	Network Innovations CL/1
173	Xyplex-MUX	Xyplex
174	MAILQ	MAILQ
175	VMNET	VMNET
176	GENRAD-MUX	GENRAD-MUX
177	XDMCP	X Display Manager Control Protocol
178	NextStep	NextStep Window Server
179	BGP	Border Gateway Protocol
180	RIS	Intergraph
181	Unify	Unify
182	Unisys-Cam	Unisys-Cam
183	OCBinder	OCBinder
184	OCServer	OCServer
185	Remote-KIS	Remote-KIS
186	KIS	KIS Protocol
187	ACI	Application Communication Interface
188	MUMPS	MUMPS
189	QFT	Queued File Transport
190	GACP	Gateway Access Control Protocol
191	Prospero	Prospero
192	OSU-NMS	OSU Network Monitoring System
193	SRMP	Spider Remote Monitoring Protocol
194	IRC	Internet Relay Chat Protocol

195	DN6-NLM-AUD	DNSIX Network Level Module Audit
196	DN6-SMM-RED	DNSIX Session Mgt Module Audit Redirect
197	DLS	Directory Location Service
198	DLS-Mon	Directory Location Service Monitor
198-200	Unassigned	
201	AT-RMTP	AppleTalk Routing Maintenance
202	AT-NBP	AppleTalk Name Binding
203	AT-3	AppleTalk Unused
204	AT-ECHO	AppleTalk Echo
205	AT-5	AppleTalk Unused
206	AT-ZIS	AppleTalk Zone Information
207	AT-7	AppleTalk Unused
208	AT-8	AppleTalk Unused
209-223	Unassigned	
224-241	Reserved	
243	SUR-MEAS	Survey Measurement
245	LINK	LINK
246	DSP3270	Display Systems Protocol
247-255	Reserved	