55
24.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
A R A G O N
INGENIERIA EN COMPUTACION

INTERCONECTIVIDAD CON EL SISTEMA OPERATIVO UNIX

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

MAURICIO ROCHA PERDOMO

ASESOR DE TESIS: ING. JIMENEZ VAZQUEZ DONACIANO

NOVIEMBRE DE 1997

VNIVERSIDAD NACIONAL
AVPENOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DEMÉXICO
CAMPUS ARAGÓN
CARRERA DE INGENIERÍA EN COMPUTACIÓN

ING. DONACIANO JIMENEZ VAZQUEZ

ING. ERNESTO PEÑALOZA ROMERO

ING. ITSMAEL MANZO SALAZAR

ING. DAVID MOISES TERAN PEREZ

ING. LILIA ENCISO GARCIA

Informamos a ustedes de la autorización que se le concedió a el alumno MAURICIO BOCHA FERDOMO para desarrollar el trabajo de tesis titulado "INTERCONECTIVIDAD CON EL SISTEMA OPERATIVO UNIX", dirigida por el Ing. DONACIANO JIMENEZ VAZQUEZ solicitando a ustedes, sean tan amables de revisar el avance del mismo y hacer las observaciones que consideren pertinentes, o en su caso, indicar a el alumno si dicha revisión se hará a la conclusión del trabajo de tesis.

Sin otro en particular, me es grato enviarles un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPÍRITU"
San Juan de Aragón, Edo. de Méx., Octubre 31 de 1996.
EL JEFE DE CARRERA

ING. JUAN GASTALDI PEREZ

JGP/srr.

AGRADECIMIENTOS

Doy gracias a Dios, por haberme permitido terminar este trabajo y logrado un objetivo más de mi vida, agradezco así mismo a mi Mamá Ana y a mi Papá Zeferino por todo el apoyo, motivación y comprensión que me han brindaron en todo momento. Doy también las gracias a mis hermanos Marisol y Jaime por toda su ayuda , apoyo y por las porras que me han echado a lo largo de toda mi vida, asi mismo quiero agradecer a Juana sus valiosos puntos de vista para llevar a cabo este trabajo. También quiero agradecer a todos mis tíos y primos por sus valiosos consejos.

Agradezco también el apoyo, comprensión y el tiempo invertido en mi al Ingeniero Donaciano Jimenéz Vázquez, para la terminación del presente trabajo

Igualmente doy las gracias a todos mis amigos (no los menciono, por temor a omitir alguno) que en todo momento me han apoyado, y por todas las cosas que hemos pasado juntos, y seguiremos pasando, gracias por su amistad.

También quiero agradecer al Departamento de Servicios de Computo de la Compañía Red Uno que me apoyo y brindó su ayuda para terminar el presente trabajo.

INTERCONECTIVIDAD CON EL SISTEMA OPERATIVO UNIX

Alumno: Mauricio Rocha Perdomo.
Director: Donaciano Jimenéz Vazquez.

INTRODUCCIÓN

Interconectividad significa la comunicación de una manera transparente entre plataformas y ambientes operativos completamente distintos, llegando al concepto de interoperabilidad, donde los servicios de red, se proporcionan no importando la plataforma que realiza la petición o el servicio.

El presente trabajo tiene como objetivo mostrar como ciertos servicios del sistema operativo UNIX, han trascendido a otras plataformas debido a la creciente explosión que han tenido las redes de computadoras, especialmente la Internet.

UNIX ha estado muy ligado a la suite de Protocolos TCP/IP, y debido a que bajo este último esta basada la comunicación en Internet, podemos observar que muchos de los servicios de red implementados en UNIX, ha sido implementados en otras plataformas, para llegar a formar servicios de red homógeneos.

Asi, actualmente podemos ver que bajo el Sistema Operativo *Windows*, o bien bajo *Mac*, si dichos Sistemas estan conectados en una Red que tiene acceso a la Internet, se les tiene que configurar a dichos sistemas, el DNS por default el cual se encarga de la traduccion de nombres a direcciones IP, las cuales representan el esquema direccionamiento de Internet, los Sistemas DNS, por lo general se encuentran corriendo en Servidores UNIX.

Otro de los Sistemas que ha transcendido, es la comparticion de archivos en red, ya sea linea o mediante transferencia de los mismos, para el primero se emplea NFS, un estándar abierto, desarrollado bajo un ambiente UNIX, y debido a que es un estándar, NFS ha sido portado a otras plataformas, permitiendo el acceso y ejecución de archivos que se encuentran contenidos en diferentes plataformas, a diferencia por ejemplo de los Grupos de Trabajo de Windows, los cuales requieren de dicha plataforma en especifico para tener acceso a los recursos de dichas máquinas. Para el segundo caso, se pueden transferir archivos desde cualquier punto de una red, no importando en que plataforma esten contenidos los archivos, para este caso empleamos FTP. Cabe mencionar que para ambos casos no importando en que plataforma se almacenen los archivos, estos siempre conservaran el significado para la plataforma en la cual fueron generados, así una maquina UNIX puede almacenar documentos realizados en Word, y estos documentos

pueden ser accesados y modificados desde una maquina corriendo Windows ,vía NFS de una manera transparente.

La comunicación entre los usuarios de la red, ha sido muy importante, así UNIX desde sus orígenes UNIX ha implementado servicios de correo electrónico basado en SMTP, y debido a que desde sus orígenes la Internet estuvo conformada por plataformas UNIX, las máquinas de plataformas distintas que se integraron posteriormente a la red de redes, tuvieron que implementar sus servicios de correo basados en el estándar de SMTP, así, actualmente casi la mayoría de la comunicación en Internet de correo electrónico esta basado en SMTP, y la mayoría de los servidores de correo están corriendo en plataformas UNIX.

Uno de los servicios que ocasiono el creciente crecimiento de Internet es el World Wide Web (WWW), dicho servicio de red, permite el acceso a grandes cantidades de información, de una manera mas amigable e interactivas, mediante graficas, sonidos, empleando hypertexto (palabras, que tienen ligas, a otros documentos WEB, relacionados con dicha palabra), dicho servicio fue desarrollado sobre una máquina corriendo NEXT, una variante de UNIX. Actualmente el 56% de los servidores WEB en Internet, se encuentran corriendo en alguna plataforma UNIX.

Así, en el presente trabajo, se comienza con los conceptos básicos de redes, que son fundamentales para entender la filosofía de TCP/IP, después se hace una descripción más detalla de Ethernet, la tecnología de red predominante, sobre la cual, ciertos conceptos de TCP/IP son entendibles mejor bajo el enfoque de dicha tecnología. Después se explica mas a detalle los conceptos de la suite de protocolos de TCP/IP, y como han llegado a ser uno de los protocolos de red mas populares, debido a que pueden ser implementados bajo cualquier tecnología de red. Seguimos con una breve descripción de los conceptos fundamentales del Sistema Operativo UNIX, para después pasar a ser una breve descripción de la administración de los servicios de red de UNIX que tienen una relación especial con TCP/IP, no se entro mas a detalle con la administracion de UNIX en general, ya que esto puede ametirar otro trabajo en particular, debido a lo extenso del tema.

Se explican a continuacion los servicios de UNIX, que han sido portados a otra plataformas, así como su configuración y puesta en funcionamiento, se trato de dar una

explicación general para todas las plataformas UNIX, pero muchas de estas, están enfocadas al Sistema Operativo Solaris 2.x, que es una de las versiones de UNIX, lider en el mercado. Dichos servicios de UNIX, son muy importantes para la comunicación y trasnsferencia de información a traves de Internet, sin los cuales dicha red no hubiera cobrado la importancia que actualmente tiene.

Por último se explican brevemente, las interfaces de programación que hacen factible la programación de aplicaciones de red basadas en TCP/IP, los cuales se les denomina, programación de *sockets*, que fueron proporcionadas por vez primera en la versión 4.2 del UNIX BSD, dichas especificaciones de funciones de comunicación en red, fueron portadas a otros Sistema Operativos, claro, con sus pequeñas variantes, pero todas basadas en el estándar BSD, asi para escribir aplicaciones de red para TCP/IP en Windows, uno puede emplear los *Winsocks*, aunque cabe mencionar que muchos productos de programación para Windows, han facilitado mucho la programación de esta clase de aplicaciones, asi vemos que muchas de los conceptos de las aplicaciones de UNIX, han sido portadas a otras plataformas para poder realizar una comunicación transparente entre diferentes arquitecturas de red.

I. INTRODUCCIÓN A REDES

1.1 CARACTERÍSTICAS DE UNA RED.

En nuestra sociedad, mas que ninguna otra sociedad anterior, estamos viviendo una época en que la información se ha convertido en algo de vital importancia. Toda nuestra vida y lo que nos rodea depende de un modo u otro de la información que podamos obtener y procesar.

La cantidad de información a la que cada uno de nosotros estamos expuestos es enorme, y aumenta día a día.

Las primeras redes comerciales se empezaron a instalar a finales de los años setenta (aunque de forma bastante restringida) y cada día se están haciendo más populares debido a las muchas ventajas que ofrecen, entre otras: el aumento de la productividad, la economía en cuanto a recursos de hardware y software, la optimizacion de los sistemas instalados, y sobre todo, la reducción de los precios de los sistemas y a la mejora de la facilidad de los mismos.

Al igual que todos los campos de la tecnología , la ciencia o las artes, se usa una terminología especifica, a continuación veremos algunos de los conceptos empleados.

Una red de computadoras, es un conjunto de sistemas conectados entre si, a través de medios de comunicación (líneas telefónicas, cable coaxial, fibra óptica y microondas) y sus objetivos fundamentales son:

1. Compartir información.
2. Compartir usuarios.
3. Tener flexibilidad en el manejo de la información.

Existe no obstante una definición oficial, la del Comité IEEE 802, quien define una red local de la siguiente manera: Una red local es un sistema de comunicaciones que permite que un número de dispositivos independientes se comuniquen entre si.

Una definición mas completa y actual seria: Un sistema de comunicaciones capaz de facilitar el intercambio de datos de datos informativos, voz, facsímil, video, conferencias, difusión de video, telemetría y cualquier otra forma de comunicación electrónica.

Por el tipo de propietarios de la red se clasifican en :

1. Redes privadas. Estas son las mas comunes y normalmente pertenecen a Universidades, bancos, empresas publicas y privadas. Se caracterizan porque un solo grupo reducido de personas tienen acceso a este tipo de red.

2. Redes Comerciales. Estas rentan sus servicios a personas interesadas a tener acceso a la información de la red.

3. Redes Publicas. Estas son administradas generalmente por el gobierno en países subdesarrollados y por grandes consorcios en países capitalistas. Utilizan la infraestructura de la red telefónica y ofrecen sus servicios a cualquier organización que se suscriba a la red y los servicios de transmisión que ofrece son en extremo económicos debido a que se comparte canales de comunicación entre gran cantidad de usuarios.

Por su distribución se dividen en:

1. Redes LAN (Local Area Network: Redes de Area Local). Estas redes estan confinadas a un espacio físico restringido y comparten gran cantidad de periféricos (impresoras, graficadores, etc) . No existe un parámetro que indique la longitud máxima de una LAN pero estan confinadas a un edificio o a un conjunto de edificios.

2. Redes MAN (Metropolitan Area Network: Redes de Area Metropolitana). Son redes que estan confinadas desde un gran número de edificios hasta abarcar una ciudad, y son redes híbridas, de diferentes equipos y topologías, las cuales comparten recursos entre sí.

3. Redes WAN (Wide Area Network: Redes de Area Amplia): Estas redes son aquellas en la que es necesario conectar equipos de comunicación remota para poder comunicar a las redes que estan integradas, que pueden ser redes de diferentes equipos y topologías. La extensión geografica que abarca una red WAN, puede ir desde una pequeña ciudad hasta cubrir en su totalidad el territorio de un país o incluso continentes.

Cuatro características han hecho importante la descripción de la implementacion de una arquitectura particular de red.

Esas características se describirán a lo largo del capitulo y son:

- Medios de transmisión.
- Técnicas de transmisión.
- Topología de la red.

- Métodos de control de acceso.

1.2 MEDIOS DE TRANSMISIÓN

Las características físicas de una red involucran la transmisión de bits a través de un medio físico. Esas características físicas pueden ser divididas en dos categorías principales: el medio físico empleado para la transmisión y la técnica de transmisión empleada para transmitir los datos sobre el medio físico.

El medio de transmisión es el camino o vía por donde viajan las señales en una transmisión.

Aunque muchos de los medios convencionales empleados en telecomunicaciones puede ser empleado en la construcción de una red, tres medios, son frecuentemente usados en la implementacion de una red: par trenzado, cable coaxial y fibra óptica.

1.2.1 Par Trenzado.

Este es, por mucho, el medio más común de transmisión, tanto para transmisión de datos analógicos o digitales. Existen dos tipos básicos de par trenzado: el par trenzado sin blindar (UTP: Unshielded Twisted Pair) y el blindado (TP: Twisted Pair). Ambos tipos de par trenzado consisten en dos alambres de cobre, cada uno cubierto con un aislamiento de PVC, en un arreglo de patron espiral. El cobre proporciona una buena conductividad, en tanto que el acero se emplea para darle dureza. El trenzado de los pares individuales minimiza la interferencia electromagnética entre los pares, conocida como crosstalk, y ayuda a proteger contra la interferencia eléctrica exterior. Los alambres en los pares tienen grosores promedio de 0.4064 mm (0.016 in) a 0.9144 mm (0.036 in), o bien, tienen un grososr de entre 20 AWG (American Wire Gauge: Patron de medida para los cables, a mayor número corresponde un diámetro menor en el cable, y viceversa) y 26 AWG.

A diferencia de UTP, los pares trenzados en STP están individualmente protegidos con una capa de aluminio o una malla de alambre. Debido a que cada par en un cable multipar tiene su propia cubierta, el crosstalk no es un problema en STP:

El estándar Ethernet que determina el cableado y la conectorización con par trenzado es el 10BaseT donde, 10 indica 10 Mbps, Base, indica que las señales que se están manejando en banda base (BaseBand), T, Par trenzado (Twisted Pair). El par trenzado tiene las siguientes características:

1) Características de Transmisión: Par trenzado puede ser empleado para transmitir señales analógicas o digitales y presenta una impedancia característica de entre 90 ohms y 110 ohms. Para señales analógicas, los amplificadores son requeridos cada 5 ó 6 kilómetros. Para señales analógicas, los repetidores son requeridos cada 100 metros.

En banda base hay determinadas categoría en las que cambian sus características de transmisión para señales digitales. Son cinco niveles de UTP:

Categoría1: Se emplea en transmisión analógica o digital de voz.

Categoría2: Se emplea en transmisión de voz.

Categoría3: Se emplea en transmisión de voz y aplicaciones de transmisión de datos a velocidad intermedia (20 mbps).

Categoría4: Se emplea en LANs de alta velocidad.

Categoría5: Se usa en LANs de alta velocidad de hasta 100 Mbps.

2) Conectividad: El par trenzado puede ser empleado para aplicaciones punto a punto o multipunto.

3) Area Geográfica: El par trenzado es empleado para redes locales dentro de mismo edificio o edificios que se encuentren muy próximos, ya que la máxima distancia es de 100 m. para UTP y 300 para STP.

4) Inmunidad al ruido: El medio es completamente susceptible a la interferencia y al ruido debido a que es f cilmente afectado por campos electromagnéticos. Señales en pares de cables adyacentes pueden interferir el uno con el otro, fenómeno conocido como crosstalk.

5) Costo: El costo de par trenzado es mucho menor que el cable coaxial o de fibra óptica en términos de costo por metro. Conociendo más este medio, podemos hablar de las siguientes ventajas y desventajas que presenta:

6) Ventajas: Barato, accesorios y conectores relativamente económicos; fácil maniobrabilidad y tecnología bien establecida para manejarlo.

7) Desventajas: limitaciones de distancia en lo que a transmisión de datos se refiere y mayor susceptibilidad a la interferencia magnética y al crosstalk que otros medios.

1.2.2 Cable Coaxial

El cable coaxial consiste en dos conductores, cuyas características de construcción les permiten operar en un rango amplio de frecuencias. Consiste en un conductor hueco cilíndrico en cuyo interior contiene otro conductor metálico; el cable interior puede ser sólido o retorcido y el conductor exterior puede ser sólido o trenzado. El conductor interior esta recubierto por un material sólido dialéctico, en tanto que el otro conductor tiene una cubierta para protección. La malla protectora aísla de la transferencia de información de interferencias electromagnéticas y del fenómeno llamado crosstalk. Además de poseer un ancho de banda amplio permite su utilización para distintas aplicaciones como enlaces de voz, datos o vídeo.

Existen dos tipos principales de cable coaxial para aplicaciones de redes: el cable de 75 ohms, usado para señalización analógica con FDM (llamado banda ancha) y el cable de 50 ohms empleado para señalización digital (llamado de banda base). Se emplea el cable de 50 ohms que se presenta en dos diferentes estándares para Ethernet.
1. Segmento Ethernet de cable coaxial grueso(Cable 10Base5).

Se le denomina segmento Ethernet debido a que para redes Ethernet la topología usada en un principio fue la de bus, y fue el primer estándar que libero Ethernet. Este medio presenta las siguientes características:

- El backbone utiliza cable coaxial grueso de 500 mts de largo como máxima longitud.
- En cada extremo se conectan terminadores de 50 ohms para aterrizar las armónicas.
- Para distancias mayores de 500 m. se conecta un repetidor.
- La distancia mínima entre hosts es de 2.5 m.
- Como máximo soporta 100 transceivers por segmento.
- Se puede emplear el cable con especificación AWG 12.

- Se recomienda para dar red a un edificio.

2. Cable Ethernet de coaxial delgado (10Base2).

Sus principales caracteristicas son:

- El backbone soporta como un máximo 185 m. de extremo a extremo.
- En cada extremo se conecta terminadores de 50 ohms.
- Como máximo soporta 30 transceivers por segmento.
- Se recomienda por su fácil manejo para la instalación de redes de computo pequeñas.

A Continuación se ennumeran las caracteristicas más importantes del cable coaxial:

1.) Caracteristicas de transmisión: El cable de 50 ohms es empleado exclusivamente para transmisión digital, empleando una codificación Manchester en forma tipica.

2) Conectividad: Este tipo de cable se utiliza para configuraciones punto a punto o multipunto.

3) Area Geográfica: La distancia máxima para un cable de banda base esta limitado a unos cientos de metros. El cable de banda ancha puede ser empleado en redes de varios kilómetros.

4)Inmunidad al ruido: La inmunidad al ruido del cable coaxial depende de la aplicación y la implementacion. Por lo general, esta es superior a la del par trenzado, para altas frecuencias.

5) Costo: El costo de instalación del cable coaxial se encuentra entre el costo del par trenzado y de la fibra óptica.

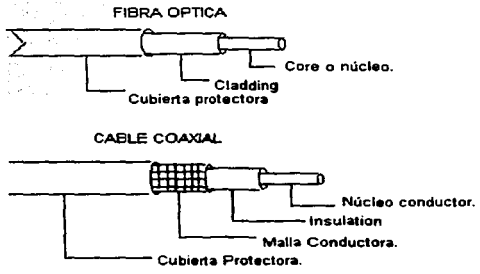


Figura 1.1 Representación esquematica de Cable coaxial y Fibra optica

1.2.3 Fibra Optica.

Un fibra óptica es un medio delgado (2 a 125 micras)capaz de conducir un rayo de luz. Varios tipos de plástico y cristales pueden ser usados para la fabricación de fibras.

Un cable de fibra óptica tiene una forma cilíndrica y consiste en tres secciones concéntricas: el núcleo (core), el revestimiento (cladding) y la cubierta exterior (jacket). La parte central consiste en una o mas fibras muy delgadas. Cada fibra viene dentro de su propio revestimiento de cristales o plástico cuyas propiedades ópticas son diferentes a la de la parte central.

1) Caracterisiticas de Transmisión: Las fibras ópticas transmiten una señal codificada en un rayo de luz por medio de una reflexión interna total. Esta reflexión puede ocurrir en un medio transparente que tiene un mayor índice de reflexión que la cubierta. La fibra óptica actúa como una guía de onda para frecuencias en el rango de 1014 a 1015 Hz en la parte del espectro electromagnético correspondiente a la luz visible y parte a la zona de rayos infrarrojos.

Existen tres modos de transmisión en las fibras ópticas, donde entendemos por modo, la variedad de ángulos con que la luz es reflejada en las paredes de la cubierta, donde la luz de la fuente entra al núcleo central y los rayos que tienen un ángulo de

incidencia adecuado se reflejan en las paredes de la cubierta y son propagados a lo largo de la fibra, en tanto que los demás rayos son absorbidos por el material de la cubierta. La segunda forma de propagación se presenta cuando se reduce el radio de la parte central de la fibra , hasta valores muy pequeños, con lo que solo un ángulo sencillo o un modo puede pasar, siendo este el rayo axial. Finalmente, el tercer tipo de transmisión se presenta cuando en la parte central el índice de reflexión se varia, conocido este tipo como fibra mutimodo de índice gradual la que es un tipo intermedio entre las características de los tipos anteriores. El índice de reflexión variable tiene un efecto de transmisión de los rayos en formas mas eficiente que el modo multimodo ordinario, pero menor que el del modo sencillo.

Existen dos tipos diferentes de luz empleadas en sistemas de fibra ópticas: los LED's (light-emmitting diode) y el diodo láser de inyección (ILD). El LED es un dispositivo de estado sólido que emite luz cuando una corriente es aplicada. El ILD es un dispositivo de estado sólido que opera basado en el principio del láser en el cual los efectos cuánticos electrónicos son estimulados para producir un rayo superradiante de un ancho de banda muy reducido. El detector usado en la recepción y para convertir la luz es el fotodiodo. La técnica de ASK es la mas comúnmente usada para transmitir datos digitales a través de fibras ópticas.

1) Conectividad: El empleo mas común de las fibras ópticas es para enlaces punto a punto.

2) Area Geográfica: La tecnología actual soporta transmisión de datos en distancias de 6 a 8 kilómetros sin repetidores.

4) Inmunidad al ruido: La fibra óptica no se ve afectada por la interferencia electromagnética o ruido. Esta característica permite altas velocidades de transmisión a largas distancias y proporciona una seguridad excelente.

5) Costo: Las fibras ópticas son mas caras que el par trenzado y que el cable coaxial en cuanto al costo por metro y los componentes requeridos para el sistema (transmisores, receptores y conectores).

1.3 TÉCNICAS DE TRANSMISIÓN.

Las técnicas de transmisión determinan como el medio físico es empleado para una comunicación de datos.

Dos técnicas pueden ser usadas para la transmisión de señales sobre un medio físico de comunicación: banda base y banda ancha. Las transmisiones en banda base emplean señalización digital, y transmisiones en banda ancha emplean señalización analógica. Los equipos pueden ser diseñados para transmitir ya sea señales digitales o analógicas sobre algún medio físico de transmisión empleado en las telecomunicaciones.

1.4 TRANSMISIONES EN BANDA BASE

Con las transmisiones en banda base, las señales de datos son transportadas sobre el medio físico de comunicación en forma de pulsos eléctricos discretos o bien ya sea mediante luz. Con esta forma de transmisión, un dispositivo emisor envía pulsos directamente sobre el canal de comunicación, y el dispositivo receptor los detecta.

Con la transmisión en banda base, la capacidad entera del canal es empleada para transmitir una señal simple de datos. Múltiples dispositivos conectados a una red, empleando transmisión en banda base, comparten el canal de comunicación empleando Multiplexaje por división de tiempo(MDT). Con MTD los dispositivos toman turnos en la transmisión, y solamente un dispositivo transmite a la vez. Los datos de los diferentes dispositivos están intercalados sobre el canal de comunicación, como se ilustra en la fig 1.2. Como solo una estación puede transmitir a la vez, debe de haber una forma de determinar cual estación esta permitida para transmitir. Las redes emplean una amplia variedad de técnicas llamadas métodos de control de acceso, para controlar el acceso al medio de transmisión.

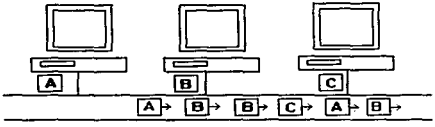


Figura 1.2 Multiplexaje por división de Tiempo.

1.5 TRANSMISIÓN EN BANDA ANCHA.

Las transmisiones en banda ancha emplean típicamente transmisión analógica usando una amplio rango de frecuencias que las transmisiones en banda base. Con las transmisiones analógicas , las señales empleadas son continuas y no discretas. El flujo de señales cruzan el medio de transmisión en la forma de ondas electromagnéticas.

Las ondas electromagnéticas tienen tres características que son útiles en telecomunicaciones: amplitud, frecuencia, y fase. En un cable eléctrico , una amplitud de onda esta asociada con el nivel de voltaje transportado sobre el cable; en una fibra óptica, la amplitud de onda esta relacionada con el flujo de intensidad de luz. La frecuencia de una onda esta relacionada con el número de ciclos u oscilaciones que l hace por segundo. La unidad es el hertz es el número de ciclos que se realizan en un segundo.

En general, la mas alta frecuencia de una señal portadora, es la máxima cantidad de información que puede transportar. En telecomunicaciones, el término ancho de banda es usado frecuentemente para referirse a la capacidad de un canal de comunicación. El ancho de banda de un canal es la diferencia entre la frecuencia mas alta y la más baja que es transportada sobre un canal. El ancho de banda de un canal tiene una relación directa con el con su flujo de datos, o el número de bits por segundo que pueden ser transportados sobre el.

1.6 TOPOLOGÍA DE RED

Una configuración de red se denomina *topología de red.* Por tanto, la topología establece la forma (en cuanto a conectividad física) de la red.

La topología de una red identifica la forma del cableado cuando es empleado para interconectar los dispositivos de red, en otras palabras, la topología de una red concierne a la configuración física de los dispositivos y al cable que los conecta. Tres principales topología son empleadas para las redes de área local: estrella, bus y anillo. La elección de una topología en particular tiene como objetivos: proporcionar la máxima fiabilidad a la hora de establecer el tráfico, encaminar el tráfico utilizando la vía de menor coste entre los dispositivos y por último proporcionar al usuario el rendimiento óptimo, y el

tiempo de respuesta mínimo. Examinaremos ahora las principales topologías que podemos encontrar en las redes de área local.

1.6.1 Topología de Estrella

Una configuración de estrella ,mostrada en la figura 1.3, tiene un controlador central, al cual todos los nodos están directamente conectados. Todas las transmisiones de una estación a otra pasan a través del controlador central, el cual es responsable de la administración y control de todas las comunicaciones. Frecuentemente el controlador central actúa como un conmutador, cuando un nodo desea comunicarse con algún otro, el controlador central, en este caso un conmutador (switch), establece un circuito, o ruta dedicada, entre los dos nodos que desean comunicarse. Esta topología ha sido muy empleada en sistema telefónicos.

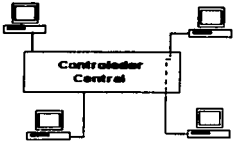


Figura 1.3 Topología de estrella

1.6.2 Topología de Bus

En la topología de bus , mostrada en la figura 1.3, cada estación esta directamente conectada a un canal de comunicación. Las señales son transmitidas sobre el canal, como cada mensaje pasa a lo largo del canal, cada estación lo recibe. Cada estación , basada sobre una dirección contenida en el mensaje, decide cuando lo acepta y procesa o simplemente lo ignora.

1.6.3 Topología de Anillo.

La topología de anillo es ilustrada en la figura 1.4, aquí el cable forma un ciclo, con las estaciones conectadas en intervalos alrededor del anillo. Las señales son transmitidas a lo largo del anillo, los mensajes son recibidos por cada estación en turnos. Como en la topología de bus, una estación determina, sobre una dirección contenida en

el mensaje, cuando lo acepta y procesa. Sin embargo , después de haber recibido un mensaje, cada estación actúa como un repetidor, retransmitiendo el mensaje en su forma original.

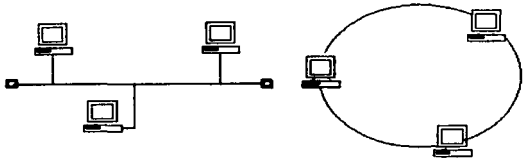


Figura 1.4 Topologías de bus y anillo.

1.7 MÉTODOS DE CONTROL DE ACCESO.

Una característica común de todas las redes de área local es que múltiples dispositivos , llamados típicamente estaciones, deben compartir el acceso a un simple medio físico de transmisión. Varios métodos pueden ser empleados para controlar como se comparte el acceso a un medio de transmisión. Una característica por las cuales las redes locales pueden ser clasificadas es por el método de control de acceso empleado.

Los métodos de control de acceso describen el método por el cual las estaciones controlan su acceso al medio de transmisión. Los dispositivos sobre una red de área local comparten el sistema de cableado que los conecta, por lo que una red de área local generalmente permite que solamente una estación transmita a un tiempo. A continuación se explicaran los métodos de acceso para controlar cuando cada estación puede usar el medio de transmisión.

Un método de control de acceso puede emplear las siguientes formas de control de transmisión.

Control Aleatorio: Con el controla leatorio, cualquier estación puede transmitir, y un permiso especifico no es requerido. Una estación puede checar al medio para ver si esta libre antes de empezar a transmitir. Entre los que tenemos:

- CSMA/CD (Carrier sense multiple access with collision detection).

- Slotted ring.
- Insersion de registro.
- Control Distribuido.

Control Distribuido: Con el control distribuido, solamente una estación a la vez, tiene el derecho de transmitir, y este derecho es pasado de estación a estación. De este método de acceso tenemos:

- Token passing: token ring, token bus.
- CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance).

Control Centralizado: Con el control centralizado, una estación controla a la totalidad de la red, y las otras estaciones deben recibir permiso de la estación de control en orden para transmitir. De este método de acceso tenemos:

- Polling.
- Conmutación de circuitos.
- Multiple acceso por división de tiempo (TDMA).

De esas técnicas, dos son de principal importancia , porque son las bases de los estándares del IEEE para redes de área local:

- *CSMA/CD* (Carrier Sense with Multiple Acess with Colision Detection: Acceso Múltiple con Detección de Portadora y Detección de Colisiones). Este es el protocolo usado por Ethernet.
- Token Ring. Esta es la base para la arquitectura predominare de IBM.

1.8 CSMA/CD (Carrier Sense Multiple Access with Collision Detection).

Bajo *CSMA/CD*, antes de que cada estación transmita, primero "escucha" en el medio para determinar si una estación esta transmitiendo. Si el medio de transmisión esta "quieto", la estación envía el mensaje. El término detección de portadora indica que una estación escucha antes de transmitir.

Cuando un mensaje es transmitido, este les llega a todas las estaciones sobre la red. Cada estación examina entonces la dirección contenida en el mensaje. Si la

dirección aplica a la estación , la estación recibe y procesa el mensaje, en otro caso, es destruido.

Con CSMA/CD, ocasionalmente sucede que dos (o mas) estaciones envían los mensajes simultáneamente, resultando en la distorsión de la transmisión conocida como colisión. Todas las estaciones sobre la red, incluyendo a las estaciones transmisoras, escuchan al medio de transmisión y están capacitadas para detectar una colisión. Y en el caso de que ocurra una, las estaciones que están transmitiendo se detienen inmediatamente después de que detectan la colisión. Después de la colisión, cada estación espera por un periodo diferente para reintentar transmitir.

Un ventaja del método CSMA/CD es que el acceso al medio físico es muy rápido cuando el trafico no es pesado. Sin embargo, bajo cargas de tráfico pesadas, el numero de colisiones se incrementa, y el tiempo gastado debido a las colisiones y retransmisiones puede causar deterioro en el desempeño de la red.

1.9 TOKEN RING

Con redes que emplean una topología de anillo, el método de acceso comúnmente usado es token passing.

Con token passing, un pequeño mensaje, llamado *token*, circula constantemente alrededor del anillo. Si el *token* esta marcado como libre, la estación que lo recibe puede transmitir un mensaje. Cuando transmite un mensaje , marca al token como ocupado, y agrega el token al mensaje. El mensaje, junto con el token de ocupado, circula alrededor del anillo, pasando de una estación a otra. Cada estación que lo recibe lo copia y le coloca unos bits que indican si se ha recibido apropiadamente. Cuando el mensaje llega de nuevo a la estación que originalmente lo envío, la estación cambia de nuevo el token a libre y remueve el mensaje. El token libre, entonces el token circula de nuevo alrededor del anillo, hasta que otra estación desee transmitir.

Una de las estaciones del anillo es designado como monitor y es responsable de detectar tokens viejos en la red, otras estaciones son designadas como monitores pasivos. Estas son responsables de monitorear el estado del monitor activo, tomando su actividad, si el monitor activo esta incapacitado para realizar su función.

El diseño de token ring tiene la ventaja de permitir mayor control sobre la transmisión de las estaciones sobre el anillo. El método permite asignar diferentes prioridades a las estaciones en el anillo, y estaciones con alta prioridad pueden tener la oportunidad de transmitir antes que las estaciones con baja prioridad. A una estación puede permitírsele enviar múltiples mensajes mientras tiene el token. En este caso, hay un tiempo límite en la cual una estación puede continuar transmitiendo mensajes. La principal desventaja con la técnica de token ring es la complejidad involucrada en el los procesamientos del token y del monitoreo.

1.10 ETHERNET

Ethernet es la tecnología de red mas usada, se puede elegir entre topología bus y estrella y cableado coaxial, de par trenzado, o de fibra óptica. .

El método de acceso que emplea Ethernet es CSMA/CD (Carrier Sense Múltiple Acces with Collision Detection). Las dos posibles topologías para Ethernet son bus o estrella. La topología bus es la mas sencilla y tradicional. Ethernet Estándar (10BASE5) y Thin Ethernet (10BASE2), ambas basadas en sistemas de cable coaxial.

En esta red, todas las estaciones de trabajo están conectadas en serie (un convenio "bus") con un solo cable. Todas las transmisiones van a todas las estaciones de trabajo conectadas. Cada estación de trabajo selecciona después las transmisiones que recibirá, basa en la información de dirección contenida en la transmisión.

En una topología estrella, todas las estaciones de trabajo anexas son canalizadas eléctricamente a una conexión central, que establece, mantiene y corta conexiones entre ellas. La desventaja es que si la conexión en el panel de conexiones no funciona todo el sistema esta comprometido.

El Ethernet de par Trenzado (10BASET) basado en UTP, como Ethernet de Fibra Óptica (FOIRL y 10BASEFL), basado en cable de fibra óptica, usa la estrella.

1.11 ARQUITECTURAS DE RED

Parte del poder de las redes es su habilidad para soportar una amplia variedad de dispositivos. El soporte de varios dispositivos puede sin embargo presentar problemas de

incompatibilidad. Para una amplia variedad de dispositivos a hacer enlazados juntos, el hardware y software de esos dispositivos necesita ser compatible, o sino interfaces complejas deben ser construidas para que una comunicación se lleve a cabo. Para facilitar esta compatibilidad, las arquitecturas de red están siendo desarrolladas para permitir que redes complejas sean construidas empleando una gran variedad de equipos.

Una arquitectura de red define protocolos, formatos de los mensajes, y estándares en los cuales el software y las maquinas deben conformar para lograr los objetivos dados. Cuando nuevos productos son creados dentro de la arquitectura, estos deberán ser compatibles - es decir pueden ser empleados en redes e interconectarse con el software y dispositivos existentes.

Debido a la importancia de las arquitecturas de red, varios tipos de organizaciones han estado envueltas en el desarrollo de estándares, incluyendo organizaciones de estándares, carriers (compañías telefónicas), y fabricantes de computadoras. Las arquitecturas diseñadas por esas organizaciones tienen muchas características en común. Todas ellas definen las reglas de una red y como los componentes de una red pueden interactuar. Pero además son diferentes.

Entre las organizaciones de estándares tenemos a la CCITT (Comité Internacional Consultivo sobre Telefonía y Telegrafía), la cual ha desarrollado estándares para transmisiones telefónicas y de datos. El ISO (Organización Internacional de Estándares), ha desarrollado un "modelo de referencia" para redes de computadoras llamado OSI (Interconexión de Sistemas abiertos). El IEEE (Instituto de Ingenieros Eléctricos y en Electrónica) ha desarrollado estándares para redes de área local mejor conocidos como el proyecto IEEE 802.

Entre los carriers tenemos a AT&T, MCI, o Western Union. Mientras que del lado de los fabricantes de computo tenemos a IBM (SNA) a DEC (DECnet) son muy importantes sus desarrollos ya que son las bases de muchas redes corporativas.

Las arquitecturas de red que han sido desarrolladas comparten muchas características entre ellas. Una es que todas ellas tienen en común un conjunto de objetivos de alto nivel:

- Conectividad. Permitir a diverso software y hardware ser conectado para formar un sistema unificado de red.
- Modularidad. Permite el uso de un pequeño conjunto de bloques de propósito general en una amplia variedad de dispositivos de red.
- Fácil de Implementar. Proveer una solución general a redes de comunicaciones que pueden ser fácilmente instaladas en una variedad de configuraciones.
- Fácil de Usar. Proveer Facilidades de comunicación de red a los usuarios en una forma que los libera del conocimiento de la estructura o implementacion de la red.
- Confiabilidad. Proveer facilidades de detección y corrección de errores.
- Fácil de Modificar. Permitir a la red de ser fácilmente modificable conforme las necesidades de los usuarios cambien o que nuevas tecnologías sean disponibles.

Otra característica que tienen en común las diferentes arquitecturas es el uso de un diseño estratificado. Una arquitectura de red comprende un amplio rango de funciones que deben ser realizadas. Esas funciones son organizadas en grupos, los cuales son acomodados en varias capas funcionales. Una capa es responsable de ejecutar un conjunto especifico de funciones y proveer un conjunto especifico de servicios.

Una arquitectura de red puede ser definida entonces en términos de los servicios provistos por cada capa y las interfaces entre dichas capas. Los protocolos definen los servicios ofrecidos a través de la interface de una capa y las reglas que sigue en el procesamiento ejecutado como parte de un servicio. Los formatos de los datos para los datos intercambiados a través de una interface son además definidos como parte de la arquitectura.

1.12 EL MODELO OSI

Como se menciono anteriormente, ISO ha estado muy activo en el desarrollo de un modelo generalizado de interconexión de sistemas, conocido como el Modelo de Referencia de Interconexión de Sistemas Abiertos (mejor conocido como modelo de referencia OSI). El modelo ha sido aceptado por otras organizaciones de estándares y es usado para coordinar todos sus esfuerzos de estándares. El propósito primario del modelo OSI es proveer las bases para el desarrollo coordinado de estándares

relacionados para interconexión flexible de sistemas usando las facilidades de las comunicaciones de datos. En la terminología OSI , un sistema esta definido como sigue:

Un conjunto de una ó mas computadoras y el software asociado con ellas, así como periféricos, terminales, operadores, procesos físicos, transferencias, etc. que forman un todo autónomo capaz de ejecutar procesamiento de información y/o transferencia de información.

El modelo OSI concierne con la interconexión entre sistemas - la forma en que ellos intercambian información - y no con las funciones internas que son realizadas por un sistema dado. El modelo OSI provee una vista generalizada de una arquitectura estratificada.

El modelo OSI usa un diseño estratificado, donde un conjunto de funciones han sido colocadas en diferentes capas. A continuación se explican brevemente las siete capas del modelo OSI:

Capa Física. La capa física es la responsable de la transmisión del flujo de bits a través de un medio particular de transmisión. Es el nivel mas básico e involucra un conjunto de reglas que especifican las conexiones eléctricas y físicas entre los dispositivos. Este nivel especifica las conexiones del cableado y las reglas eléctricas necesarias para transferir datos entre dispositivos.

Capa de Enlace de datos. La capa de enlace de datos es responsable de proveer transmisión de datos confiable de un nodo a otro y para separar a las capas superiores de lago relacionado con el medio de transmisión. Tiene que ver que la transmisión este libre de errores, en este nivel a la información se le denomina frames de datos.

En la arquitectura IEEE 802, la capa de enlace de datos es dividida en las siguientes subcapas:

Control Lógico de Enlace (LLC). Esta subcapa es responsable de funciones de enlace de datos independientes del medio. Esto permite a la capa superior de red accesar los servicios de la red de área local sin preocuparse como la red este implementada.

Control de Acceso al Medio (MAC). Esta subcapa esta relacionada con el método de control de acceso, que determina como es controlado el uso del medio de transmisión.

Capa de Red. La capa de red se encarga con el ruteo o encaminamiento de los datos entre una red a otra. Es responsable de establecer, mantener y terminar la conexión de red entre dos usuarios y para la transferencia de datos a lo largo de esa conexión.

Capa de Transporte. La capa de transporte es responsable de proveer transporte de datos entre dos usuarios con cierto nivel de calidad. Cuando una conexión es establecida entre dos usuarios, la capa de transporte es responsable de seleccionar una clase particular de servicio a ser usado, de monitorear transmisiones para asegurar que la calidad del servicio sea mantenida, y para notificar a los usuarios si esto no es logrado.

Capa de Sesión. La capa de sesión se enfoca en proveer servicios de organización y sincronizar el diálogo que toma lugar entre usuarios y para administrar el intercambio de datos. Un tarea principal de la capa de sesión es controlar cuando los usuarios puedan enviar y recibir, basado en cuando ellos pueden enviar y recibir concurrentemente o alternadamente.

Capa de Presentación. La capa de presentación es responsable de la presentación de la información en una forma que esta tenga significado a los aplicaciones de red. Esta puede incluir traducción de códigos, conversión de datos, o compresión y expansión de datos.

Capa de Aplicación. La capa de aplicación provee un significado a los procesos de las aplicaciones que accesan al sistema de interconexión para intercambiar información. Esta capa incluye servicios para establecer y terminar las conexiones entre aplicaciones y monitorear, administrar que los sistemas están interconectados y los recursos que ellos emplearan.

1.13 CONCLUSIONES.

En este capítulo se presento una breve explicación de los conceptos más importantes que estan envueltos en una red, y que son importantes para comprender los capitulos subsecuentes. Por lo que podemos concluir que las redes han cambiado la computación de escritorio, de una simple máquina a un rico ambiente , con una gran cantidad de información y recursos que las personas pueden accesar, la necesidad de

comunicarse, para enviar mensajes, compartir datos, acessar a recursos de computo y compartir dispositivos periféricos es lo que ha contribuido al rápido desarrollo y crecimiento de las redes de computo.

II CARACTERÍSTICAS ETHERNET

2.1 INTRODUCCIÓN

El desarrollo actual de Ethernet ocurrió en el Centro de Investigación de Xerox en Palo Alto, CA. Un equipo de desarrollo lidereado por el Dr. Robert Metcalfe conecto a mas de 100 computadoras sobre un 1 K.M. de cable. El sistema resultante, el cual operaba a 2.94 Mbs usando el protocolo de acceso CSMA/CD, fue referido como Ethernet en un memorándum publicado por Metcalfe.

De la Alianza con Digital Equipment Corporation e Intel Corporation resulto en el desarrollo de una red Ethernet de 10 Mbps.

2.2 ELEMENTOS DE ETHERNET

2.2.1 Transceivers

Los tranceivers se usan para conectar nodos a los distintos medios Ethernet. Los tranceivers son también llamados MAUs (Media Attachment Unit) permiten la conexión al cable Ethernet y proporcionan una Interface o conector AUI (Application User Interface) para la computadora.

2.2.2 Repetidores

Los repetidores se usan para conectar 2 o mas segmentos Ethernet de cualquier tipo de medio. A medida que los segmentos exceden su número máximo de nodos o longitud máxima, la calidad de la señal empieza a deteriorarse. Los repetidores amplifican la señal y realizan la temporización necesaria para conectar segmentos. La división de un segmento en dos o mas segmentos con un repetidor permite que una red continúe creciendo. Una conexión con un repetidor cuenta con el limite total de nodos de cada segmento.

Los repetidores también monitorean las características básicas para Ethernet pueda correr correctamente en todos los segmentos conectados. Cuando estas condiciones no se satisfacen en un segmento en particular, por ejemplo, cuando ocurre una ruptura en el bus, todos los segmentos podrían volverse inoperables. Los repetidores limitan el efecto de estos problemas a la sección del cable que tiene la falla,

desconectando el segmento problema y permitiendo que los segmentos no afectados funcionen correctamente.

De la misma manera como los diferentes medios Ethernet tienen limitaciones de segmentos, las ampliaciones a las redes Ethernet que se crean con repetidores y múltiples segmentos también tienen restricciones. Estas restricciones generalmente tienen que ver con limitaciones de sincronía. Aunque las señales eléctricas dentro del medio Ethernet viajan a una velocidad cercana a la de la luz, a una señal le toma cierto tiempo llegar de un extremo a otro de un segmento grande Ethernet. El estándar Ethernet asume que a una señal no le tomara más de una cierta cantidad de tiempo su propagación a los extremos de la red. Si la red es muy grande esta asunción no se cumplirá y la red no funcionara correctamente. Cuando el estándar Ethernet es violado, los paquetes se pierden, el desempeño de la red se decrementa y las aplicaciones se vuelven lentas y pueden fallar. Las redes que violan las reglas podrían seguir siendo funcionales, pero están expuestas a fallas esporádicas o problemas frecuentes de naturaleza indeterminada. Además el uso de repetidores simplemente extiende el tamaño de la red y a medida que va creciendo , el ancho de banda de la red podría llegar a ser un problema . En este caso, se pueden usar switches, bridges para partir una red grande en muchos segmentos pequeños y mas eficientes.

2.2.3 Bridges

La función de un bridge es conectar redes diferentes. La conexión se puede hacer sobre un solo enlace remoto de datos soportado entre dos bridges o se puede realizar en un solo bridge que una dos o mas redes; el objetivo es lograr que los usuarios de la red no noten el uso de estos dos dispositivos. Los bridges mapean las direcciones Ethernet de los nodos que residen en cada segmento de la red y entonces solo el trafico necesario pasara a través de el. Cuando se recibe un paquete, el bridge determina los segmentos fuente y destino. Si ambos segmentos son el mismo, el paquete es rechazado ("filtrado"); si los segmentos son distintos, entonces el paquete se acepta y pasa al otro lado. Además los bridges evitan la propagación de un segmento a otro de paquetes erróneos y también los filtra. Los bridges son llamados dispositivos de "almacenamiento y aceptación" porque exploran todo el frame Ethernet antes de tomar la decisión de filtrado o de aceptación.

2.4 DIRECCIONAMIENTO

Las comunicaciones entre usuarios a través de redes requieren varias formas de direccionamiento. Típicamente, dos o tres direccionamientos son requeridos: un direccionamiento físico, un direccionamiento de enlace de datos, y un direccionamiento de red. Un diseño mas común, es el uso de solamente del direccionamiento físico y de red. Prácticamente hablando, otros direccionamientos son requeridos para comunicaciones no ambiguas entre dos usuarios, tal como nombres en las capas superiores y números de puertos, que se veran en los capítulos posteriores.

2.5 DIRECCIONAMIENTO FÍSICO

Cada dispositivo (tal como una computadora o una estación de trabajo) sobre un enlace de comunicación o red es identificado con una dirección física. Esta dirección es frecuentemente llamada *dirección hardware ó MAC*. Muchos fabricantes colocan la dirección física sobre una tarjeta lógica dentro del dispositivo o en una unidad de interface conectada directamente al dispositivo. Dos direcciones físicas son empleadas en una comunicación : una dirección que identifica al remitente(fuente) y la otra que identifica al destino(receptor). El tamaño de la dirección física varia: muchos sistemas usan 2 direcciones de 48 bits, pero también otros tamaños de direcciones son empleadas. La estructura de 48 bits de direccionamiento es considerada demasiado grande por algunos diseñadores, pero los protocolos Ethernet y IEEE los usan, así que es ampliamente usado. Este direccionamiento es llamado dirección de *Control de Acceso al Medio* (MAC).

Desde el contexto del modelo de capas de las comunicaciones, la dirección física es empleada en las capa física o de enlace de datos. El dispositivo receptor examina la dirección destino de un unidad de datos (denominada *frame*) que llega. Si la dirección coincide con la dirección física del dispositivo, este es pasado a las capas superiores. Si la dirección no coincide, el frame es ignorado. Así, la detección de direcciones en las capas inferiores previene que los datos que no van dirigidos a la maquina sean pasados hasta las capas superiores y ocupen tiempo de procesamiento.

La IEEE asumió la tarea de la asignación universal de las direcciones de capa física y los identificadores de protocolo de las LANs. Previamente, este trabajo fue

realizado por Xerox, quien administraba que fueran conocidos como identificadores de bloque (bloque ID) para direccionamiento Ethernet. La oficina de Administración Ethernet de Xerox asignaba esos valores, los cuales fueron de tres octetos (24 bits) en tamaño. La organización que recibía esta dirección podía emplear los 24 bits restantes de la dirección Ethernet, en la forma que decidiera.

Así la IEEE asumió la tarea de la asignación de los identificadores universales para todas las LANs, no solamente para las redes Ethernet. Así cada Identificador único para una organización, provee una organización de espacio de 24 bits, aunque realmente el espacio verdadero de direcciones es de 22 bits porque los dos primeros son empleados para propósito de control. Así, el espacio de direcciones es 2^{22}.

El formato para el Identificador Único de una Organización (IUO) es mostrado en la Figura 2.2. El bit menos significativo del espacio de direcciones corresponde al bit de dirección individual/grupal (I/G).

El bit de dirección I/G, si es colocado a 0, significa que el campo de dirección identifica una dirección individual. Si el valor es colocado a 1, el campo de direcciones identifica un grupo de direcciones usado para identificar más de una estacion conectada a la LAN. Si todo el Identificador Unico de la Organización (IUO) es colocado a 1, significa una dirección de broadcast, la cual identifica a todas las estaciones de la red.

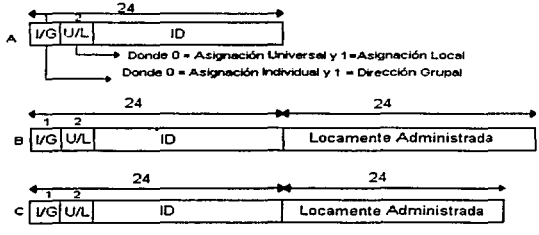


Figura 2.1 Ids y Direccionamiento Universal (a) ID de Organización unica (Bloque ID) (b) Direccionamiento Universal MAC (C) ID de protocolo.

El segundo bit del espacio de direcciones es conocido como el bit local o universal (U/L). Cuando es colocado a 0 tiene una asignación universal de importancia. Por ejemplo, del IEEE, si es colocado a 1, tiene una dirección localmente asignada. Este bit debe estar siempre colocado en 0, si es administrado por el IEEE.

El IUO es extendido para incluir una dirección LAN universal de 48 bits (designada como la dirección MAC), mostrada en la figura 2.1. Los 24 bits del espacio de direcciones son los mismos como los designados para el IUO por el IEEE. La segunda parte del espacio de direcciones, consiste de un restante de 24 bits, éste es administrado localmente y puede ser colocado a algún valor que la organización escoga, este rango, normalmente es normalmente asignado por los fabricantes de tarjetas de red.

Los 24 bits administrados localmente permiten a una organización asignar aproximadamente 16 millones de direcciones únicas.. Si este espacio de direccciones es terminado, el IEEE asigna un IUO adicional, pero no se asigna un adicional IUO hasta que una organización emplee todos los valores en el espacio de 24 bits asignados. Se tienen aproximadamente 2^{48} posibles valores, los cuales pueden identificar aproximadamente 281,475 trillones de direcciones unicas, las cuales deberan ser suficientes por un tiempo.

El proyecto IEEE 802 ademas administra un *identificador de protocolo*. Este valor no es una dirección física, pero es discutida aqui, debido a su relación con el esquema de direccionamiento del IEEE. El formato para el identificador es mostrado en la figura 3.2c. Los primeros 24 bits son para el IUO discutido previamente. Los 16 bits restantes son administrados localmente por una organización; sin embargo, en algunas instancias, estos valores estan reservados para protocolos bien conocidos.

2.6 FUNCIONES CSMA/CD

El estándar define un modelo que comprende seis funciones, como se muestra en la figura 2.2 .Tres de esas funciones están asociadas con la transmisión de datos, y tres funciones paralelas están relacionadas con la recepción de datos. La encapusación/desencapsulación de datos y administración de acceso al medio mostradas son ejecutadas por la misma subcapa de control de acceso al medio; la función de

codificación/decodificación de datos es realizada por la capa física, la cual opera debajo de la subcapa MAC.

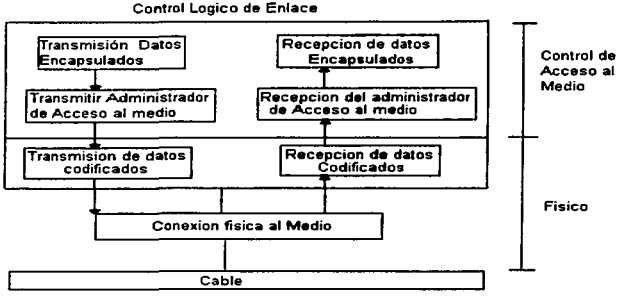


Figura 2.2 Funciones CSMA/CD

2.7 ENCAPSULACIÓN/DESENCAPSULACIÓN DE DATOS

La encapsulación de datos aplica a una estación emisora y agrega información al comienzo y principio de la unidad de datos a ser transmitida, después la transmisión del frame es recibida desde el control lógico de enlace. Esta información es usado para realizar las siguientes tareas:

- Sincroniza a la estación receptora con la señal.
- Delimita el comienzo y final del frame.
- Identifica el direccionamiento de las estaciones receptoras y emisoras.
- Detecta la transmisión de errores.

Cuando es recibido un frame , una función en la estación receptora es responsable de reconocer la dirección destino, determinado si coincide con la dirección de la estación. realizando chequeo de errores, y removiendo entonces la información de control que fue

agregada por la función de encapsulación de datos en la estación emisora, antes de pasar al frame a la subcapa de LLC.

2.8 ADMINISTRACIÓN DE ACCESO AL MEDIO

En una estación receptora , la función de administración de acceso al medio es responsable de determinar cuando el medio de transmisión esta disponible para su uso, así como para la inicialización de la transmisión cuando se requiera. Esta función, además determina las acciones que deberán ser tomadas cuando es detectada una colisión y cuando deberá darse una retransmisión. En una estación receptora, la administración de acceso al medio ejecuta chequeos de validación sobre los frames recibidos antes de pasarlos a la función de desencapsulación de datos.

2.9 DECODIFICACIÓN DE DATOS.

La decodificación de datos ejecutados en la capa física, es responsable de la transformación de los bits que están siendo transmitidos a señales eléctricas para ser enviadas a través del medio de transmisión. Para CSMA/CD, la codificación Manchester es empleada para transladar el flujo de bits a señales eléctricas. Cuando la señal es recibida, la codificación de datos lo transforma de señales eléctricas al flujo de bits que esas señales representan. La función de codificación de datos es además responsable de escuchar al medio de transmisión y para notificar al administrador de acceso al medio cuando el medio de transporte esta libre u ocupado y cuando una colisión ha sido detectada.

Además de la codificación/decodificación de datos, la capa física incluye funciones relacionadas a la estación conectada a un medio particular de transmisión. Esas funciones son generalmente ejecutadas en un dispositivo físico separado llamado Unidad de Conexión al Medio (Medium Attachment Unit, MAU) que es empleada para conectar una estación de red al cable físico de transmisión.

2.10 FRAMES ETHERNET.

En esta sección , se centrara nuestra atención sobre la composición de los diferentes tipos de frames Ethernet, los frames son las denominaciones a las unidades de datos en la capa fisica del modelo OSI. En realidad solamente hay un *frame Ethernet*, mientras que el formato de frame CSMA/CD estandarizado por el IEEE es técnicamente referido como un *frame 802.3*. Una área donde Ethernet y el IEEE 802.3 difieren es en su formato de frame.

2.11 COMPOSICIÓN DEL FRAME

La figura ilustra la composición general del frame Ethernet y del 802.3. Las diferencias son mínimas . Un frame Ethernet contienen un campo de preámbulo(preamble) de 8 bytes seguido por un campo de un byte de delimitador de frame. Una segunda diferencia entre la composición de los frames Ethernet y 802.3 concierne al campo de Ethernet de dos bytes type. Este campo es empleado por Ethernet para especificar el protocolo transportado en el frame , habilitando a varios protocolos a ser transportados independientemente de algún otro. Bajo el formato del frame 802.3, el cual especifica el numero de bytes que siguen a ese campo como datos.

Las diferencias entre Ethernet y 802.3, los hace incompatibles uno del otro. Esto significa que la red debe contener ya sea todas interfaces de red compatibles con Ethernet o todas las NICs compatibles con el IEEE 802.3. Afortunadamente, el hecho que el formato representa un estándar significa que muchos vendedores actualmente venden hardware y software compatible con 802.3. A continuación se presenta una explicación de los campos de Ethernet y 802.3.

2.12 FORMATOS DE LOS FRAMES .

El formato de los frames se muestra en la figura.

Preamble	Direccion Destino	Direccion Fuente	Campo de tipo	Campo de datos	Chequeo de secuencia frame
8 Bytes	6 Bytes	6 Bytes	2 Bytes	46-1500 bytes	4 bytes

Preamble	Delimitador comienzo	Direccion Destino	Direccion Fuente	Tamaño	Campo de datos	Bytes relleno	Chequeo de secuencia frame
7 Bytes	1 Byte	2-6 Bytes	2 o 6 Bytes	2 Bytes	0-n Bytes	0-p Bytes	4 Bytes

Figura 2.3 Formato del frame Ethernet y del IEEE 802.3

Preamble o Preámbulo. El campo de preamble consiste de ocho(Ethernet) o siete bytes(IEEE 802.3) bytes alternado los bits en 1 y 0. El propósito de este campo es para anunciar al frame y habilitar a todos los receptores sobre la red de sincronizarse por ellos mismos para recibir el frame. En adición, este campo por si mismo(bajo Ethernet) o en un conjunción con el campo delimitador del comienzo del frame(bajo el estándar IEEE 802.3) asegura que hay un mínimo espacio de 9.6 ms entre frames para detección de errores y recuperación de operaciones. El preámbulo es empleado para sincronización.

Delimitador de Comienzo del frame. Este campo solo es aplicable para el estándar IEEE 802.3 y puede ser visto como una continuación del preamble,consiste de una secuencia de bits de 10101011. El delimitador de comienzo del frame indica el comienzo de un frame de datos.

Campos de Direcciones. El frame incluye las direcciones fuentes y destino. Ethernet especifica el uso de una direccion de 48 bits, IEEE 802.3 permite direcciones de 16 y 48 bits.

Longitud. El campo de longitud es un campo de 2 bytes que indica el tamaño del campo de datos que sigue. Este campo es empleado cuando un campo de relleno es incluido en el frame.

Campo de Tipo. Ethernet no soporta el empleo de un campo de longitud, los dos bytes son empleados para contener un campo de tipo. El valor especificado en el campo de tipo solo tiene significado para las capas de red superiores y no esta definido como parte de la especificacion Ethernet. Para IP es $0800_{(16)}$, para ARP es $0806_{(16)}$.

Campo de Información. El campo de información contiene la unidad de protocolo de datos que fue pasado desde la subcapa de control lógico de enlace.
Ethernet define un tamaño mínimo de frame de 72 bytes y un maximo de 1526 bytes, incluyendo el preámbulo. Si los datos a ser enviados son más grandes o pequeños que esos tamaños, es responsabilidad de las capas superiores rellenarlo o dividirlo en paquetes individuales.

Campo de relleno. Para detectar las colisiones apropiadamente, el frame que es transmitido debe contener un número mínimo de bytes. Si un frame que esta siendo ensamblado no conoce esta longitud mínima, un campo de relleno es agregado para llevarlo a esa longitud.

Chequeo de Secuencia del Frame. El frame término con un secuencia de chequeo del frame. Cuando la estación emisora ensambla un frame, ejecuta un calculo de código de redundancia ciclica (CRC) sobre los bits de ese frame. El algoritmo especifico que es empleado esta descrito en la documentación del IEEE y siempre da como resultado un valor de 4 bytes. La estación emisora almacena este valor en el campo de chequeo de secuencia del frame y entonces transmite el frame. Cuando la estación receptora recibe el frame, ejecuta el mismo CRC y compara el resultado con el valor contenido en ese campo. Si los valores no coinciden, la estación receptora asume que un error de transmisión ha ocurrido y entonces puede pedir que el frame sea retransmitido.

2.13 ADMINISTRACIÓN DEL CONTROL DE ACCESO AL MEDIO CSMA/CD

Un función principal de la subcapa de control de acceso al medio es administrar la compartisión del medio de transmison entre las diferentes estaciones de la red. En CSMA/CD, esto es conocido como administrador de acceso al medio. La función de administración de acceso al medio, recibe un frame de la función de encapsulación de datos, una vez que la información necesaria de control ha sido agregada. El administrador del Acceso al medio es responsable entonces por ver que los datos sean

físicamente transmitidos. El diseño usado por CSMA/CD involucra detección de portadora ó escuchar por un medio de transmisión libre, antes de transmitir.

El administrador de acceso al medio, determina a través de los servicios de la capa física, cuando el medio de transmisión esta ocupado. Si no lo esta, se pasa al frame al administrador de acceso al medio para su transmisión. Si la portadora esta ocupada, el administrador de acceso al medio continua monitoreando a la portadora hasta que ninguna otra estación esté transmitiendo. El administrador de acceso al medio entonces espera un tiempo especifico, para permitir a la red limpiar por señales de ruido y entonces comenzar la transmisión.

2.14 DETECCIÓN DE COLISIONES

El administrador de control de acceso al medio continua monitoreando a la portadora después de que la transmisión del frame empieza. Si dos estaciones habuieran comenzado a transmitir al mismo tiempo, sus señales debieron tener una colisión.

La figura ilustra el peor caso de una detección de una colisión sobre una red de banda base. Las estaciones A y B son dos estaciones sobre la red. La estación A comienza la transmisión y en ese mismo instante el nodo B comienza la transmisión. La colisión ocurre cerca de la estación B, causando una señal que debe viajar en sentido opuesto de la estación A. El frame que la estación A esta transmitiendo deben ser largo para asegurar que la estación A esta aún transmitiendo cuando detecta la colisión con la transmisión de la estación B.

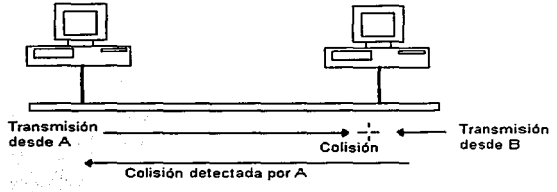


Figura 2.4 Colisión en banda Base.

El *tiempo de propagación*, es la cantidad de tiempo que le toma a una señal para llegar al final de la red. Así que el tiempo máximo que le toma detectar una colisión es dos veces el tiempo de propagación de la estación A. Esto representa el tiempo que le toma a la estación A alcanzar el final mas el tiempo que le toma a la señal de colisión viajar a través de la red hasta alcanzar a la estación A.

2.15 BACKOFF DESPUÉS DE UNA COLISIÓN

Cuando ocurre una colisión , todas las estaciones que están transmitiendo dejan de hacerlo, esperan una cierta cantidad de tiempo, y entonces, si la portadora esta libre, empiezan la transmisión de nuevo. Si todas las estaciones esperaran la misma cantidad de tiempo antes de checar a la portadora y comienzan la transmisión de nuevo, otra colisión puede ocurrir. Para prevenir esto, cada estación genera un número aleatorio que determina la cantidad de tiempo que la estación debe esperar antes de probar a la portadora. Este periodo de tiempo es conocido como retardo backoff de la estación. El retardo backoff es calculado en términos de múltiplos de ranuras de tiempo , el cual es el tiempo que le toma a una señal viajar desde un extremo de la red al otro extremo, y regresar de nuevo.

Cada estación genera un número aleatorio que cae dentro de un rango específico de valores . Entonces espera ese número de espacios o ranuras de tiempo antes de reintentar la transmisión. Cabe mencionar que si todas tienen el mismo rango de valores es muy probable que en la retransmisión, pueda ocurrir otra colisión.

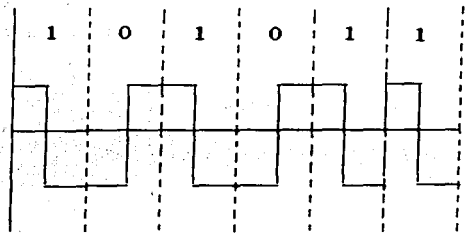
Para balancear estas consideraciones, el estándar CSMA/CD usa un diseño conocido como backoff binario exponencial. El rango de números esta definido como $0 \leq r < 2^n$, donde n refleja el número de intentos de retransmisión que la estación ha hecho. Para los primeros diez intentos, el rango de n va de 1a 10. Para subsecuentes intentos, n continua teniendo un valor de 10. Esto significa que para el primer intento de retransmisión, el rango es de 0-1; para el segundo intento, 0-3; para el tercero, 0-7, y así sucesivamente. Si repetidas colisiones ocurren, el rango continua creciendo hasta que la estación transmite exitosamente sin colisiones. Si una estación no transmite después de 16 intentos, se reporta una condición de error. Cuando el tráfico es ligero , el backoff exponencial binario resulta en un retardo antes de la retransmisión . Cuando el trafico es

alto, colisiones repetidas deberán causar que el numero de rango se incremente. Por supuesto, cuando el tráfico es extremadamente alto, colisiones repetidas deberán comenzar a generar condiciones de error.

2.16 FUNCIONES DE SEÑALIZACIÓN FÍSICA.

Las funciones primarias de la subcapa de señalización física son las siguientes:
Acepta un bit de la subcapa MAC, lo codifica, y lo transmite.
Recibe una señal de bit desde la unidad conectada al medio, lo decodifica, y lo pasa a la subcapa de MAC.
Provee a la subcapa MAC de la información sobre el estado de la portadora , la ocurrencia de colisiones, y el éxito ó error de las transmisiones.

La codificación y decodificación involucra la conversión entre el flujo de bits y las señales binarias. El esquema de codificación Manchester usado con CSMA/CD es ilustrado en la figura .Para propósitos eléctricos, es deseable en muchas redes de área local implementaciones en las que las transiciones de positivo a negativo y viceversa ocurran con una regularidad predicible. La codificación Manchester produce el número deseado de transiciones y es usado en muchas redes de área local. Para una implementación típica de la codificación Manchester, un voltaje negativo para la primera mitad de tiempo de bit seguido por un voltaje positivo para la segunda mitad del tiempo de bit representa el valor de 1; un voltaje positivo seguido por una transición a un voltaje negativo, representa el valor de 0. Así con la codificación Manchester, una transición de negativo a positivo o viceversa ocurre en cada tiempo de bit.



CODIGO MANCHESTER

Figura 2.5 Codificación Manchester

Con la codificación Manchester, los tiempos de bit en los cuales la señal es cambiada de positivo a negativo y viceversa son empleadas para representar algo más que el valor de un bit, como por ejemplo, el comienzo o término de un bloque de transmisión . Este tipo de señalización permite a los datos y a las señales de reloj ser combinadas dentro de una transmisión simple , desde la cual la estación receptora puede emplear el cambio de estado que ocurre durante cada tiempo de bit, para permanecer sincronizada.

En resumen las funciones CSMA/CD incluyen encapsulación/desencapsulación de datos, administración de acceso al medio, y codificación/descodificación de datos. Un frame CSMA/CD o Ethernet contiene los campos para que los datos puedan llegar a su destino.. La administración de acceso al medio esta basada en que las estaciones que escuchan por una medio libre antes de transmitir. Si dos o mas estaciones comienzan a transmitir al mismo tiempo, una colisión ocurre. Las estaciones transmisoras detienen su transmisión, esperan un periodo de tiempo, y entonces intentan retransmitir de nuevo.

CSMA/CD emplea codificación Manchester y soporta varias formas de transmison. Las transmisiones en banda base pueden ser usadas en velocidades desde 1 Mbps hasta 100 Mbps.

2.17 TECNOLOGIAS ETHERNET.

2.17.1 ETHERNET CONMUTADA.

En la búsqueda de mejores velocidades de filtrado y de aceptación, surgió la idea de que el tiempo requerido para determinar si un paquete debería ser filtrado o aceptado ("latencia") se podría reducir si solo se examinara la información de direcciones contenida al inicio del frame Ethernet. Los dispositivos que empezaron a usar esta tecnología fueron llamados originalmente switches "cut-through". Actualmente el termino "Switch Ethernet" (conmutador) se emplea para referirse a cualquier dispositivo de puertos múltiples que sea capaz de filtrar y aceptar paquetes a la misma velocidad de propagación de Ethernet sin importar la técnica empleada. El termino "Bridge" se usa para referirse a los dispositivos de dos puertos de técnica de almacenamiento y aceptación.

La Ethernet conmutada es un nueva tecnología que cuenta con interruptores de entradas múltiples centralizados para proveer un enlace físico entre segmentos múltiples de red. Dentro de cada interruptor inteligente, circuitos de alta velocidad soportan conexiones virtuales entre todos los segmentos, para máxima asignación de longitud de banda a la demanda. La adición de nuevos segmentos a un interruptor aumenta la velocidad de la red, en tanto que disminuye la sobrecarga general, de manera que Ethernet conmutado provee flexibilidad de configuración superior. También le da una excelente vía de migración de 10 a 100 Mbps Ethernet, puesto que ambos segmentos pueden operar mediante el mismo interruptor.

2.17.2 FAST ETHERNET (100BASET ; IEEE 802.3u)

100BASET conserva la técnica familiar de acceso al servicio informatico CSMA/CD usada en redes Ethernet de 10 Mbps. También soporta un amplio margen de opciones de cableado: dos estándares para par trenzado , y uno para fibra. 100BASETX soporta cable de categoría 5 de dos pares o Tipo 1 STP. 100BASET4 usa 4 pares de cable categorías 3 o 4. Y 100BASEFX permite enlaces de fibra óptica mediante cable de fibra mutimodo dúplex.

2.18 CONCLUSIONES

Para finalizar, actualmente Ethernet es la tecnología de LAN mas popular. Otros tipos de LAN son Token Ring, FDDI (Fiber Distributed Data Interface) y LocalTalk. Cada una tiene sus ventajas y desventajas. Ethernet ofrece un buen balance entre velocidad, precio y facilidad de transmisión. Estos puntos fuertes, combinados con la amplia aceptación en el mercado de computo y su habilidad para soportar virtualmente todos los protocolos populares de red hacen a Ethernet la tecnología ideal para muchos usuarios de computadoras actualmente. El estándar 802.3 de la IEEE define reglas para la configuración de Ethernet y especifica como los elementos de una red deben de interactuar entre si. En los últimos años han ido surgiendo otros estándares a mayores velocidades , uno de los mas importantes es Fast Ethernet o 100BaseT; la tabla muestra la comparación entre este estándar y 10BaseT.

	Ethernet 10BaseT	Ethernet 100BaseT
Distancia Máxima entre Repetidores	100 m.	5 m.
Número máximo de repetidores en un segmento	4*	2
Distancia Total Máxima entre dos estaciones de trabajo	500 m.	205 m.
Tiempo máximo para detectar una colisión	Tiempo para transmitir 64 bits	Tiempo para transmitir 46 bits.

* Aunque puede variar de acuerdo al fabricante del equipo

III. PROTOCOLOS TCP/IP

3.1 INTRODUCCION

En los primeros días de la computacion las computadoras intercambiaban información con los dispositivos que tenian directamente conectados a ellas, tal como tarjetas e impresoras. El uso interactivo de las computadoras requirio primero conexiónes locales y luego remotas, de terminales de usuarios. Cuando una organizacion adquiria varias computadoras, frecuentemente habia la necesidad de transferir datos entre las computadoras o permitir a los usuarios conectados a una computadora acessar a otra.

Los vendedores de computadoras respondieron a esos requerimientos desarrollando hardware y sofware de comunicaciones. Desafortunadamente, este hardware y software:

- Fue propietario, es decir trabajaba solamente con el equipo de los vendedores.
- Soportaba solamente un número limitado de tipos de redes de area local y amplia.
- Algunas veces era extremadamente complejo, requiriendose de diferentes dialectos en el software para cada dispositivo y para cada aplicación.
- Faltaba la flexibilidad que deberia habilitar a redes independientes a ser conectadas entre ellas fácilmente y con una baja inversión.

En los 90s la conectividad se ha hecho urgente. Las organizaciones necesitan combinar computadoras de escritorio, servidores, dentro de comunidades de redes de area local. Se desea conectar LANs a otras LANs y a redes de Area Amplia (WANs). También se desea tomar ventaja de redes de tecnologias nuevas y de costo apropiado tan rápido como sea posible.

Esos requerimientos son similares a los de 1970, que se tenian en los Estados Unidos en el Agencia de Investigacion de Proyectos Avnazados de la Defensa, o DARPA (Defense Advanced Research Projects Agency). TCP/IP fue diseñado por esas necesidades, y desarrollado bajo el patrocinio de DARPA.

3.2 TERMINOLOGIA

3.2.1 Big Endians y Little Endians

Algunas computadoras almacenan datos con el primer byte significativo primero. Esto es llamado estilo de representación big endian. Algunas computadoras almacenan los datos con el byte menos significativo primero, en un estilo Little Endian.

Similarmente, hay estandar de comunicaciones de datos Big Endian que representan a los datos transmitidos con el bit y byte mas significativos primero. Los estandares de protocolos de Internet son Big Endians.

3.2.2 Protocolos, Stacks y Suites.

Un protocolo es un conjunto de reglas gobernando la operación de algunas funciones de la comunicación. Por ejemplo, IP consiste de un conjunto de reglas para el ruteo o encaminamiento de datos, y TCP incluye las reglas para la llegada confiable y en secuencia de los datos.

Un stack de protocolos es un conjunto de protocolos estratificados que trabajan en conjunto para proveer la comunicación entre aplicaciones. Por ejemplo, TCP, IP y Ethernet hacen un stack de protocolos.

Una suite de protocolos es una familia de protocolos que trabajan juntos de una manera(estilo) consistente.La suite de protocolos TCP/IP envuelve un gran número de funciones, que van desde la obtención dinámica de la dirección física sobre una tarjeta de red a un servicio de directorio que revela como el correo electrónico debera ser encaminado.

3.2.3 Hosts, Routers y Gateways.

Un host es una computadora con uno o más usuarios. Un host que soporta TCP/IP puede actuar como el punto final de una comunicación.

Un router es un sistema que encamina los datos a travez de la red.

Gateway es un sistema que realiza alguna clase de translaciónde protocolos.

Finalmente, elemento de red o nodo debera ser usado para referirse a una entidad de comunicación en la red, sin especificar si éste es un host, un router, o algun otro dispositivo, como un bridge.

3.3 HISTORIA Y CONCEPTO DE TCP/IP

A los finales de los 60s la Agencia de Investigacion de Proyectos Avanzados de los E.U. o ARPA(por sus siglas en ingles, posteriormente cambiado a DARPA) promobio una asociacion con Universidades y otras organizaciones de Investigacion para investigar nuevas tecnologias de comunicacion de datos.

Juntos construyeron la ARPANET, la primera red de conmutación de paquetes. Una versión experimental de cuatro nodos de la ARPANET fue puesta en operación en 1969. El experimento fue un éxito, y después se extendió en los E.U. de costa a costa. En 1975, la Agencia de Comunicación de la Defensa(DCA) asumio la responsabilidad para operar la red, la cual fue aun considerada una red de investigación.

Los primeros protocolos de ARPANET fueron lentos y sujetos a frecuentes errores. Por 1974, el diseño de un nuevo de núcleo de protocolos, fue propuesto, por Vynton G. Cerf y Robert E. Kahn.

El diseño *Cerf/Kahn* sento las bases para el desarrollo subsecuente del Protocolo Internet(IP) y el Protoclo de Control de Transmisin(TCP). Tomo tres años convertir los hosts ARPANET a la nueva suite de protocolos.

En 1983, la ARPANET incluia a mas de trescientas computadoras. En 1984, la ARPANET fue divida en dos partes. Una, llamada ARPANET, fue dedicada a la investigación y desarrollo. La otra llamada MILNET, fue una red militar no clasificada. La ARPANET original, fue finalmente disuelta en 1990.

La arquitectura TCP/IP aglutina racimos de redes, creando una gran red llamada una internet. Para un usuario una Internet, simplemente aparece como una red simple , compuesta de todos los hosts conectados a alguna de las redes constituyentes.

El protocolo TCP/IP fue diseñado para ser independiente del hardware y del Sistema Operativo, así como del medio de transmisión y de las tecnologías de enlace de datos.

En 1983, el Departamento de Defensa, adopto la suite de protocolos TCP/IP como su estándar.

El Departamento estimulo la disponibilidad de TCP/IP mediante la implementación de TCP/IP en UNIX por Bolt, Beranek y Newman (BBN) y asi como también a la Universiad de California en Berkeley a incoporar el código BBN, dentro de su sistema

operaitvo UNIX BSD (Berkeley Software Foundation) 4.2. Posteriormente, fue agregado al UNIX System V de AT&T.

La familia de protocolos TCP/IP es usado ampliamente por su habilidad de aglutinar redes de area amplia y local.

3.4 ARQUITECTURA DE PROTOCOLOS EN NIVELES.

Para lograr un intercambio confiable de datos entre computadoras, hay muchos procedimientos separados que deben ser tomados en cuenta:

- Formatear los datos.
- Empaquetar los datos.
- Determinar la ruta que los datos debera seguir.
- Regular la cantidad de datos transferidos acorde al ancho de banda disponible y la capacidad del receptor de para procesar los datos.
- Transmitir los datos sobre el medio fisico.
- Ensamblar los datos para que esten en sequencia y no haya piezas pérdidas.
- Checar los datos recibidos, por piezas duplicadas.
- Notificar al remitente cuantos datos han sido recibidos correctamente.
- Enviar los datos a la aplicación correcta.
- Manejar errores o problemas.

El resultado es que la comunicación es demasiado compleja. Una motivación del uso del protocolo por niveles es dar al software de comunicaciones una estructura que sea racional, simple y fácil de modificar.

El servicio de internet mas fundamental consiste de un sistema de envío de paquetes. Tecnicamente, el servicio esta definido como un servicio de envío de paquetes connectionless, que no es confiable y que hace su mejor esfuerzo, análogo a un servicio provisto por un hardware de red que opera bajo el paradigma del mejor esfuerzo. El servicio es llamado no confiable debido a que el envío no esta garantizado. El paquete puede ser perdido, duplicado, retardado, o enviado en desorden, debido a que el servicio no detecta tales condiciones, éste no lo informa al emisor y al receptor. El servicio es llamado *connectionless*, porque cada paquete es tratado independientemente de todos los otros. Una secuencia de paquetes enviados de una máquina a otra pueden viajar

sobre rutas diferentes, o algunos pueden perderse, mientras otros son enviados. Finalmente, se dice que el servicio realiza su mejor esfuerzo porque la internet no descarta paquetes caprichosamente; es decir solamente cuando los recursos son agotados o las redes inferiores fallan.

TCP/IP es un modelo estratificado de 5 capas, a diferencia del modelo OSI. La figura 3.1 muestra la unidad de información de cada capa; en la capa 1 es un *bit*, en la capa 2 es un *frame*, en la capa 3 es un *datagrama*, en la capa 4 un *segmento*, y en la capa 5 un *mensaje*.

Las capas 1 y 2 del modelo no están definidas actualmente por TCP/IP, ya que éste esta diseñado para ser independiente del medio físico.

En la capa 3 esta el Protocolo Internet (IP), ICMP, ARP y RARP. Esta capa provee un servicio de datagramas básico, esto es, IP mueve datos realizando su mejor esfuerzo pero no garantiza su recepción. Un servicio llamado ICMP es normalmente provisto en la transmisión de datagramas y permite a los sistemas, adaptarse a los condiciones actuales de una red. Además, contenida dentro de la capa de IP, están ARP y RARP, estos son usados para el mapeo del direcciónamiento IP al direcciónamiento MAC.

En la capa 4 hay dos posibles opciones, UDP y TCP:

UDP extiende el servicio no orientado a conexión de IP (connectionless) a las aplicaciones que no requieren confiabilidad. Los datagramas UDP pueden ser enviados a una red, sin un exceso de cabeceras (conocido como *overhead*) que son necesarios para crear y mantener una conexión. En algunas situaciones, esto es mas confiable.

TCP provee un servicio confiable con corrección de errores y control de flujo. El costo de proveer un servicio confiable es un exceso de overhead en el cierre y apertura de una conexión, el poder de procesamiento para la corrección de errores y la transmisión de errores, pero algunas aplicaciones requieren de confiabilidad no importando el costo.

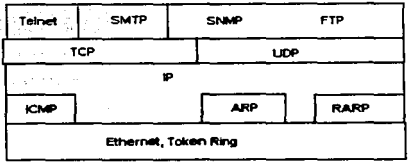


Figura 3.1. El Modelo TCP/IP

3.5 INTERNET PROTOCOL (IP).

IP es la piedra angular de TCP/IP y esta especificado en el RFC (Request For Comments: Un conjunto de documentos que contiene protocolos de la Internet y discusiones de temas relacionados) 791 y MIL-STD 1777.

IP ofrece tres definiciones importantes: Primero, el protocolo IP define la unidad básica de transferencia de datos usada a través de una Internet TCP/IP. Asi, especifica el formato exacto de todos los datos, a medida que vayan pasando por la Internet TCP/IP. Segundo, el software IP realiza las funciones de enrutamiento, escogiendo una ruta sobre la que seran enviados los datos. Técnicamente, el servicio se define como un sistema de entrega de paquetes sin conexión (connectionless), no confiable, que opera en base al paradigma de entrega del mejor esfuerzo. El servicio se dice no confiable, porque no grarantiza la entrega. Este servicio, se llama sin conexión, porque cada paquete es tratado de manera independiente de los demás. Una secuencia de paquetes enviada de una máquina a otra, podrian viajar por diferentes caminos, y algunos paquetes podrian perderse por el camino.

El Protocolo Internet llama a su unidad básica de transferencia *datagrama Internet*, o simplemente *datagrama*.

Una Internet, se define como un conjunto de redes conectadas por enrutadores. El Protocolo Internet (IP) es un protocolo de la capa de red, que encamina los datos a través de una Internet. IP ofrece las siguientes caracteristicas:

- Implementar el uso de hosts construidos por diferentes vendedores.

- Implementar el uso de enrutadores construidos por diferentes vendedores.
- Englobar una creciente variedad de redes de diferentes tipos.
- Ser adaptable para el crecimiento y modificacion de la redes.
- Interrelacionar viejas tecnologías con las nuevas.
- Soporte para servicios orientados a conexión y connectionless.
- Fragmentación de los datos, si es necesario.

3.6 DIRECCIÓNES IP

Las direcciónes en el nivel de capa de red, deben de contar con la característica de ser universales para la red o interred sobre la que estan trabajando. Esto evitar que existan ambigüedades en la elección del camino a seguir para las unidades de protocolo que están viajando debido al intercambio de información entre equipos. El esquema de direccionamiento en una red como la Internet, es forzozamente jerárquico.

La primera jerarquía que podemos decir, esta establecida formalmente, es la que divide a los tipos de direcciónes en cinco clases: *Clase A,B,C,D y E*. Todas las direcciónes IP estan conformadas por 32 bits o sea 4 bytes u octetos.

Las direcciones clase A se distinguen porque comienzan con un cero. En ellas los primeros ocho bits especifican el número de red, y los siguientes, el número de host dentro de esa red. Las direcciónes clase B se distinguen porque comienzan con los bits 10, y en ellas, 16 bits identifican a la red, y 16 al host. Las direcciónes clase C comienzan con 110 , y asignan 24 bits al número de red, los restantes 8 son para el host.

Podemos encontrar un tipo especial de dirección, la clase D, que comienza con el patrón 1110, que especifica direcciónes multicast. Todo esto lo podemos resumir en los siguientes puntos:

- Las direcciónes de Clase A comienzan con un número entre 0 y 127.
- Las direcciónes de Clase B comienzan con un número entre 128 y 191.
- Las direcciónes de Clase C comienzan con un número entre 192 y 223.
- Las direcciónes de Clase D comienzan con un número entre 224 y 239. Esta tipo de direcciónes es empleada para Multicasting.

Las direcciónes de Clase E comienzan con un número entre 240 y 255. Este Clase de direcciónes esta reservada para uso experimental.

Las direcciónes no solamente permiten especificar hosts especificos, sino redes. Si por ejemplo, hablamos de la dirección 132.248.0.0 estamos hablando de una dirección que especifica una red. La dirección nula permite hacer referencia a la red o a la máquina actual (0.0.0.0). La dirección donde todos los bits son iguales a 1 se utiliza para direccionar un mensaje a todas las máquinas de la red (*broadcast*), por ejemplo, la dirección 132.248.255.255, serviría para enviar un mensaje a todos los host dentro de esa red, esto se llama *broadcast dirigido*. Por otra parte la dirección 255.255.255.255 se denomina *broadcast limitado*, y especifica un mensaje para todos los hosts dentro del contexto de una red. Cuando una parte de la dirección contiene 0s, se da por supuesto que se trata de "este", así, si una computadora envía un mensaje a la dirección 0.0.57.5, se enviara un mensaje al host 57.5 dentro de "esta" red.

La dirección 127 es una dirección especial, que sirve para probar el software de red (127.0.0.1). Nunca deber existir tráfico con esta dirección, ya que los paquetes jamas salen del host que los envío. Esta dirección se denomina *loopback*.

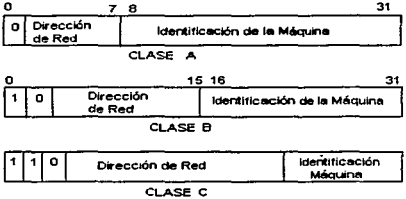


Fig. 3.2. Clases de Red comunmente utilizadas.

¿Quién asigna las direcciones de las que estamos hablando? Existe una autoridad central llamada *Network Information Center(NIC)*, este es el que asigna las direcciones de la clase indicada para cada entidad. Para el caso de México las asigna el NIC México.

3.7 SUBREDES.

Una organizacion que tiene una red de clase A o B es una red muy compleja conformada de varias LANs y WANs. De aqui que es necesario particionar el espacio de direcciónes en una forma que coincida la estructura de la red como una familia de

subredes. Para hacer esto, la parte local de la dirección es dividida en dos piezas en una forma conveniente, como se muestra en la figura 3.3:

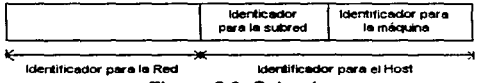


Figura 3.3. Subredes

El tamaño de la parte de la subred de una dirección y la asignación de los números a las subredes es responsabilidad de la organización *propietaria* de esa parte de direcciónes.

El direcciónamiento de una subred es frecuentemente hecho en el límite de un byte. Una organización con una dirección de Clase B tal como 132.248 puede ser usado su tercer byte para identificar subredes. Por ejemplo:

132.248.1

132.248.2

132.248.3

Entonces el cuarto byte entonces puede ser usado para identificar hosts individuales sobre una subred.

3.8 MÁSCARAS DE SUBREDES.

El tráfico es encaminado a un host buscando en la parte de la red y de la subred de su dirección IP. Es fácil decir cual es parte de la red ya que esta estrictamente definida como una dirección Clase A, B ó C.

Las organizaciones escogen su propio tamaño del campo de la subred, así ¿ Cómo pueden los hosts reconocer este campo?. La respuesta es, un parámetro de configuración llamado la *máscara de la red.* Esta es, un secuencia de 32 bits de 1s. Los bits cubren la parte de la red y la subred de una dirección.

Por ejemplo, si el administrador de la red de Clase B 132.248 ha escogido usar el tercer octeto de todas las direcciónes para identificar a las subredes, entonces la máscara de la red es:

Una máscara para red de Clase B con Redes de 8 bits.

111111111 11111111 11111111 00000000

Las máscaras de la subred, son expresadas en forma decimal, separados por puntos. La máscara anterior puede ser escrita como 255.255.255.0.

La máscara puede ser escrita en notación hexadecimal como:

ffffff00

Los ruteadores directamente conectados a una subred son configurados con la máscara para la subred.

Para ver como trabaja la máscara, supongamos que nuestro host esta conectado a una LAN y tiene una dirección IP 132.248.27.10 con máscara de red ffffff00. Si nosostros deseamos enviar datos a un host con dirección IP 132.248.27.11, lo que se hace es que a nivel de bits se realiza un AND lógico con la dirección destino IP y la máscara de la red, y el resultado de dicha operación se le compara con la parte de la red de la dirección, y si son iguales, muestra que la máquina remitente y la destino están conectadas a la misma subred -132.248, así, que ellas, pueden intercambiar datos directamente a través de la subred, en caso contrario los datos tienen que ser encaminados a través de los ruteadores.

3.9 DATAGRAMA INTERNET.

A continuación, se muestra el formato del datagrama IP.

Los campos en el header del protocolo son mostrados en la figura 3.4, los cuales tiene el siguiente significado.

3.9.1 Número de Version

Es un campo de 4-bits que contiene el número de versiónde IP, la versiónactual es la 4.

3.9.2 Longitud

Esta especifica la longitud de la cabecera del protocolo IP en palabras de 32 bits.

3.9.3 Tipo de servicio

Son entradas para los protocolos de dispositivos, para procesar el mensaje de acuerdo a un conjunto de criterios.

Bits	Si 0	si 1
0-2	Precedencia	
3	Retardo Normal	Bajo Retardo
4	Salida Normal	Salida Rapida
5	Confiabilidad normal	Confiabilidad Alta
6-7	Reservado	

Los tres bits de "precedencia" indican la urgencia del datagrama, con valores que van de 0 (precedencia normal) a 7 (control de la red), permitiendo que las máquinas que envían indiquen la importancia de cada datagrama. Los bits D, T y R especifican que tipo de transporte que se desea.

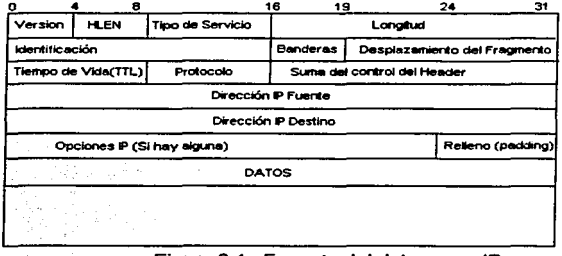


Figura 3.4. Formato del datagrama IP.

3.9.4 Longitud del Paquete

Este campo contiene la longitud del paquete incluyendo el header del protocolo, limitando el datagrama a 65,535 octetos.

3.9.5 Identificación

Este es un identificador único (por ejemplo, un contador) que es creado por el host remitente. Este número, es usado, para reensamblar los fragmentos de todas las piezas de una cadena.

3.9.6 Banderas.

Antes de explicar los demás campos, se deben tener las siguientes consideraciones, en la práctica surgen límites, en cuanto al tamaño del datagrama, a diferencia de los frames de la red física, los datagramas son manejados por software, asi que estos, pueden de ser cualquier longitud de hasta 65,535 octetos. Para hacer eficiente la transportación Internet, se desearía que cada datagrama viajara en un frame físico (la idea de llevar un datagrama en un frame de la red, se le llama encapsulamiento, como se muestra en la figura 3.5).

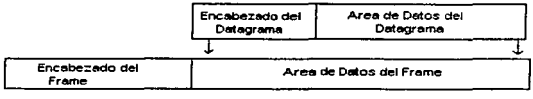


Figura 3.5 Encapsulamiento de Datagramas.

En el mejor de los casos, el datagrama IP entero cabría en un frame físico, haciendo la transmisión a traves de la red eficiente, pero cada tecnología de intercambio de paquetes pone un límite en la cantidad de datos que se pueden transportar en un frame físico, a este límite se le llama MTU (Maximum Transfer Unit) de la red, para Ethernet es de 1500 octetos. El limitar los datagramas, para que quepan en el MTU mas pequeño de la Internet, haria las transferencias ineficientes, cuando esos datagramas pasaran por redes que pudieran llevar frames de mayor tamaño. Sin embargo, permitir

que los datagramas sean mas grandes que el MTU mínimo en una internet significa que un datagrama no siempre cabría en un solo frame de la red.

La elección es obvia: el objetivo en el diseño, de la Internet es ocultar las tecnologías de red subyacentes y hacer la comunicación conveniente para el usuario. Así, en lugar de diseñar datagramas que se adhieran a las restricciones de las redes físicas; el software de IP escoge un tamaño inicial de datagrama de tamaño conveniente y busca la manera de dividir los datagramas mas grandes en piezas mas pequeñas cuando el datagrama necesite atravesar una red que tenga un MTU pequeño. Las pequeñas piezas en las que se divide un datagrama se llaman *fragmentos*, y el proceso de dividir un fragmento se conoce como *fragmentación.*

Dos bits, DF(Dont Fragment) y MF(More Fragments) controlan el manejo de paquetes en el caso de fragmentación. Si el bit DF es colocado, el paquete IP no puede ser fragmentado bajo ninguna circunstancia. Si el bit MF es colocado el paquete IP es seguido por más subpaquetes.

3.9.7 Desplazamiento del Fragmento (offset)

Si MF es colocado, el fragment offset contiene la posición del submensaje contenido en este paquete, relativo al comienzo del mensaje original. Es decir, especifica el corrimiento de los datos, que se estan enviando en el fragmento, en unidades de 8 octetos. Para reensamblar el datagrama, el destino debe obtener todos los fragmentos, comenzando, con el que tiene desplazamiento cero, hasta el que tenga el desplazamiento más alto.

3.9.8 Tiempo de Vida(Time to live)

En este campo se especifica el tiempo de vida del paquete, con el fin de evitar que paquetes viejos sigan circulando por la red, con lo que se podría evitar que existieran al mismo tiempo dos paquetes diferentes con el mismo número de secuencia. El RFC 791 especifica las unidades empleadas en este campo, en segundos, y dictamina que cada nodo debe decrementar este número en 1, aun cuando si el tiempo de procesamiento es menor que 1. Si este paquete contiene 0, el paquete debera ser descartado por el actual procesador. El TTL original por default es de 225 segundos.

3.9.9 Protocolo de Transporte

Este contiene el ID del protocolo de transporte sobre el cual el paquete ha sido enviado. Para TCP, el número es 6, para UDP es 17 y para ICMP es el número 1.

3.9.10 Suma de Control del Header

Este campo contiene, el ckecksum para los campos del header del protocolo. Este control, previene a los nodos o hosts de trabajar con datos erróneos. Por eficiencia los datos del usuario en el paquete IP no son checados; esto se lleva a cabo en la capa de transporte. Este se lleva a cabo mediante el complemento a 1 de la suma de 16 bits de todas las palabras de 16 bits de los datos a ser checados. Este algoritmo es simple y rápido; el cual es un prerequisito para la operación eficiente de los nodos.

3.9.11 Direcciónes Fuente y Destino.

Las direcciónes Internet de 32 bits.

3.9.12 Opciones.

Es utilizado para tareas especiales (administración de red, seguridad).

3.9.13. Relleno(Padding).

Representa bits en cero, que se pueden necesitar para asegurarse que la longitud del datagrama sea un múltiplo de 32 bits, cuando hay banderas activadas en el campo de opciones.

3.10 ARP(Address Resolution Protocol)

Como sabemos la dirección de capa de enlace de datos es la que realmente sirve para enviar información entre una computadora y otra(si bien la ruta es determinada por la capa de red). ¿Cómo se lleva a cabo la transformación entre estas dos direcciónes?

La dirección de capa de enlace de datos está intrínsecamente ligada a la tecnología de red que se esté utilizando. Por ejemplo, una dirección bajo ARCNET es diferente a una de ETHERNET, de esta manera, no podríamos hablar de un algoritmo único que permita llevar a cabo una transformación entre una dirección Internet y cualquier tipo de dirección sobre capa 2.

El Protocolo de Resolución de Direcciónes(ARP) permite que un host encuentre una dirección física para un host destino en la misma red física, si se conoce la dirección IP de este host.

Primero se envía un broadcast a todos los host dentro de la misma red física, este broadcast contiene un código que especifica la pregunta ¿De quién es esta dirección IP? Solo el host con la dirección indicada contestar con un paquete que contendra su dirección física.

Sería muy ineficiente tener que realizar este proceso cada vez que se quiere entablar una comunicación entre dos computadoras, para evitar esto, se tiene un cache, que contiene pares Dirección Internet - Dirección física. Así cada vez que un host quiere enviar un paquete, primero ve en su cache, por si esta dirección ya ha sido resuelta.

Además de lo anterior, cada paquete ARP contiene la correspondencia entre dirección física y de Internet del host que lo envía, con esto se evitan muchos broadcast.

Un paquete ARP viene encapsulado dentro de un frame. El header del frame debe de especificar en su campo de tipo un código que indique que contiene un frame de ARP. En el caso de Ethernet, el código es 0806 Hex.

La figura 3.6 muestra el formato de un mensaje ARP , cuyos campos se describen a continuación:

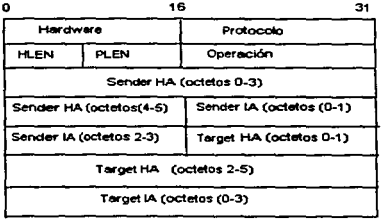


Figura 3.6. Formato ARP

Hardware: Indica el tipo de Hardware, Ethernet, Token Ring.

Protocolo: Especifica el protocolo que utiliza ARP, para IP es 0800.

LHA: Especifica la longitud de la dirección física, dada en octetos,en el caso de Ethernet es 6.

LPA: Especifica la longitud de la dirección IP, dada en octetos, para IP su valor es 4.

Codigo de Operacion: Especifiva si se trata de una solicitud (0001) o de una respuesta (0002).

Dirección Física Fuente (Sender HA): Dada por el solicitante y copiada al mensaje de respuesta.

Dirección IP fuente (Sender IA): Dada por el solicitante y copiada al mensaje de respuesta.

Dirección Física Destino (Target HA): (Petición= broadcast FFFFFFFFFFFF, respuesta= Dirección Física del solicitante).

Dirección IP Destino (Target IA): Dada por el solicitante y copiada al mensaje de respuesta.

3.11 RARP(Reverse Address Resolution Protocol)

Sabemos que las direcciones físicas son dependientes del Hardware, además que cada máquina que utiliza TCP/IP tiene una o más direcciónes de 32 bits, independientes de la dirección física. Los programas de aplicación siempre utilizan la dirección IP cuando especifican un destino.

Las direciones de IP usualmente se mantienen en almacenamiento secundario, entonces la máquina al iniciar puede encontrarlas. Entonces surge la pregunta ¿Cómo determina su dirección IP una máquina sin disco (diskless)?. La computadora sin disco deber enviar una petición a otra máquina, denominada servidor, y esperar hasta que reciba una respuesta. La petición deber identificar a la computadora que desea su dirección de manera única, de tal manera que el servidor pueda regresar la IP indicada. Ya que la computadora que solicita su dirección no sabe como localizar al servidor, simplemente envía un broadcast.

El protocolo que permite todo este proceso se denomina *RARP (Protocolo de Resolución de Direcciones Inversa)*. El valor del campo tipo es 8035 hex. para el caso de un frame de Ethernet que contenga un paquete de RARP. Primero la computadora que necesita la dirección de IP envía un broadcast, entonces un servidor de RARP deberá contestarle.

3.12 ICMP (Internet Control Message Protocol)

El protocolo ICMP (Protocolo Internet de Control de Mensajes) esta definido en el RFC 792 y es una parte requerida por IP.

ICMP proporciona el mecanismo para la transferencia de información de control o error en una red TCP/IP. ICMP permite a los enrutadores enviar mensajes de error o control a otros ruteadores o hosts, es el medio de comunicación entre el software de TCP/IP en una computadora y otra. ICMP no define acciones correctivas a los errores, solo permite la transmisión de información acerca de los mismos. ICMP solo puede reportar errores al emisor de la información, ya que no hay forma de que conozca cuales fueron los puntos intermedios a través de los cuales se envió el datagrama.

Un paquete de ICMP viene dentro de un datagrama IP, ya que tendra que viajar por la interred.

El software de IP no envía mensajes de ICMP a causa de errores en un paquete de ICMP.

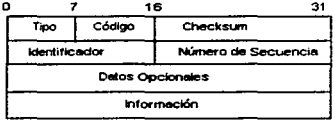


Figura 3.7 Formato ICMP

El formato se muestra en la figura 3.7:

- Tipo(8 bits): Identifica el mensaje de ICMP.
- Codigo (8 bits): Usado para especificar parametros del mensaje.
- Suma de Control (16 bits): Control de Suma del mensaje entero ICMP.
- Parametro (32 bits): Parametros adicionales.
- Información(Variable): Información adicional relacionada con el mensaje.

En los casos en los que el mensaje de ICMP se refiera a un datagrama particular, el campo de información incluye la cabecera de IP, mas los 64 bits de información del

campo de datos del datagrama original, en el cual, se origino el problema. Esto permite a IP a verificar los mensajes provenientes de ICMP con el datagrama original.

Nombre	Tipo	Código	Interpretación
Echo Reply	0	0	
Destino inalcanzable	3	0	Red inalcanzable
		1	Host Inalcanzable
		2	Protocolo Inalcanzable
		3	Puerto Inalcanzable
		4	Fragmentacion Solicitada
		5	Ruta Origen Fallida.
Source Quench	4	0	
Redirección	5	0	Datagrama para red redirigido.
		1	Datagrama para host redirigido
		2	Datagrama para tipo de servicio y red redirigido.
		3	Datagrama para tipo de servicio y host redirigido.
Echo request	8	0	
Timeout	11	0	Tiempo de vida excedido Tiempo de reensamble de datagrama excedido
Problema de parámetro	12	0	El apuntador indica error
Timestamp request	13	0	Petición de la hora y fecha de una máquina remota
Timestamp reply	14	0	Respuesta a la petición anterior. Se usan para la sincronizacion de routers
Solicitud de Información	15	0	Este mensaje es usada para obtener el número de red de un host. Este mecanismo puede ser usado en sistemas de dial-up que usan SLIP.
Respuesta de Información	16	0	
Solicitud de máscara de dirección	17	0	Permite a un nodo descubrir la máscara de la red en la cual esta conectado
Respuesta de Máscara de Dirección	18	0	

Cuando se emplea el comando ping, se envia un paquete de ICMP tipo *Echo Request*, el cual deber ser contestado por el destino con un paquete *echo reply*.

Aun cuando ICMP emplea la capa de IP, se considera al mismo nivel de IP, ya que no necesariamente provee algun servicio a las capas superiores.

3.13 SISTEMA AUTONÓMOS.

Un sistema Autónomo es una colección de LANs y WANs interconectadas por enruteadores, los cuales estan bajo el control de una autoridad administrativa para su configuracion, direcciónamiento, asignación de nombres y ruteo. Las decisiones hechas dentro de una autoridad, no deberan estar visibles al resto de la Internet. Un ruteador, es una computadora de propósito especifico, que trabaja con el software de TCP/IP, para proveer enrutamiento, a los datagramas que estan viajando en la red.

3.14 RUTEO DE DATAGRAMAS

Para entender el funcionamiento de IP en el contexto de enrutamiento, debemos recordar que el objetivo de este protocolo es ofrecer una "red virtual" *connectionless*. Esta red o más bien esta "interred" puede implantarse sobre muchos tipos distintos de redes fisicas. En el momento en que una aplicación intenta comunicarse, el protocolo TCP/IP eventualmente generar uno o más datagramas de IP (con la estructura que ya se analizó). El host deberá de hacer una decisión de enrutamiento, cuando decide hacia donde enviar los datagramas. Aunque las decisiones de enrutamiento se encuentran a cargo de los ruteadores.

Podemos decir que el envio de datagramas en una red se divide en dos tipos: *directo* e *indirecto*. El directo solo puede suceder cuando los hosts se encuentran en la misma red fisica.

La transmisión de un datagrama IP entre dos máquinas no siempre involucra enrutamiento. El emisor encapsula el datagrama en un frame, resuelve la dirección fisica del destino a partir de la dirección de IP (a través de ARP por ejemplo) y manda el frame directamente a su destino.

A través de una comparación entre las partes de red de el mismo y el destino, si son iguales, evidentemente, los puntos de comunicación estarán dentro de la misma red. El envio directo de datagramas es el aspecto final, de la comunicación entre datagramas.

El envio indirecto de datagramas deberá involucrar la decisión de la mejor computadora que actuará como primer enrutador. Recordemos que todo camino comienza con el primer paso. Esta es la primera decisión que debe enfrentar un host cuando envía un datagrama hacia un destino que no se encuentra en su misma red física.

Entonces, éste enrutador, deberá también tomar la decisión de cual será el siguiente paso dentro del camino que llevará al datagrama a su destino.

Los enrutadores, generalmente, basan su decisión en tablas de enrutamiento. Estas tablas no deberán contener direcciónes completas, sino más bien, direcciónes de redes. Las tablas consisten de pares (N,G), en donde N es la dirección de IP de la red destino, y G es la dirección IP completa del siguiente enrutador en el camino hacia la red N. El dominio G son todos los enrutadores que se conectan directamente al enrutador que contiene la tabla, ya que el envío deber de ser directo.

A través de rutas por default se puede disminuir el tamaño de las tablas de enrutamiento, ya que se pueden agrupar un número de destinos en una sola entrada.

Aunque, por razones de seguridad y chequeo de errores, casi todo el software TCP/IP permite especificar rutas por host, en lugar de por red.

Es importante recordar que el enrutamiento por IP no modifica el datagrama original, en particular, no se modifican las direcciónes fuente ni destino. Las direcciónes de cada uno de los brincos deberán resolverse a direcciónes físicas, y que no se especifican dentro del datagrama.

Una vez que un host ha recibido un datagrama, este deberá checar si está dirigido hacia él. Un enrutador puede recibir muchos datagramas que aparentemente estan dirigidos hacia él (debido a que tienen la dirección de capa 2 indicada), pero que marcan un destino IP diferente. Si se recibe un paquete, que no va dirigido hacia él (en el nivel de dirección física), éste deberá descartarlo, ya que si no lo hace, puede causar resultados anómalos, además de causar tráfico innecesario en la red. Si un host, recibe un datagrama de IP no dirigido hacia él, no debera reenviarlo por cuatro razones principales: Cuando el host recibe un datagrama que debió ser para otro equipo, algo estuvo mal con el esquema de direcciónamiento, enrutamiento o envío, ya que es mejor que ocurran fallas para así detectar el problema. Segundo, si reenvía se causaría tráfico innecesario. Tercero, pueden ocurrir errores insospechado, por ejemplo, si un host en una red envío un datagrama hacia una computadora, y erroneamente lo envío en broadcast en capa 2, la computadora destino se vería bombardeada por los renvíos. En cuarto lugar, los enrutadores hacen más funciones que simplemente reenviar los paquetes, también tienen mecanismos de intercambio de información de enrutamiento, reporte de errores, etc.

Los protocolos de ruteo disponibles para los ruteadores TCP/IP y sistemas finales son:

- RIP (Routing Information Protocol).
- OSPF (Open Shortest Path First).
- IS-IS (Integrated Intermediate System to Intermediate System).
- BGP (Border Gateway Protocol).
- IGRP (Interior Gateway Routing Protocol).
- Hello.
- EGP (Exterior Gateway Protocol).
- GGP (Gateway to Gateway Protocol).

3.15 TRASMISSION CONTROL PROTOCOL (TCP)

El Protocolo de Control de Transmisión (TCP) es un protocolo de la capa de transporte y por lo tanto esta encapsulado dentro de IP. Su tarea principal es la transmisión confiable de los datos a travez de la red. TCP está especificado en el RFC 793 y en el MIL-STD 1778. A travez de él se especifican los formatos de los datos y acuses de recibos que las dos computadoras intercambiaran (transmisora y receptora) con el fin de lograr una transferencia confiable, asi como los procedimientos a través de los cuales las computadoras aseguraran que los datos llegan correctamente. Su dirección de transporte en el header del protocolo IP, es 6. Los principales atributos de TCP son:

- Provee un circuito virtual de comunicacion bidirecciónal full duplex.
- Visto por el usuario, los datos, son transmitidos como un flujo de datos(no en bloques).

Transmisión confiable de datos usando:

- Asociacion de puertos con conexiónes.
- Establecimiento de conexiones.
- Segmentación de datos para la transmisión.
- Numeración de los datos.
- acuse de recibo positivo con retrasmisión.
- Manejo de segmentos duplicados.
- Calculo de checksums.

- Regulando el flujo de datos mediante ventanas.
- Cerrando conexiónes de una manera apropiada.
- Abortando conexiónes.
- Interactuando con las aplicaciones de las capas superiores.
- Señalización de datos urgentes.
- Chequeo de errores y reporte de errores.
- Ajustandose al congestionamiento de la red.

El protocolo no incluye la definicion de las interfaces exactas que las aplicaciones utilizaran para solicitar los servicios de TCP, ya que esto depende de las caracteristicas del equipo y SO sobre las que se implante. TCP no define tampo ninguna característica para el medio de transmisión que utilizará, y por lo tanto, éste protocolo podra implementarse de tal manera que utilice lineas telefonicas, enlaces de red de area local, etc.

A continuación se muestra la estructura de TCP en la figura 3.8:

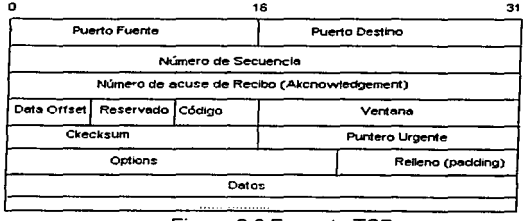


Figura 3.8 Formato TCP

3.15.1 Números de Puerto Fuente y Destino.

Son dos campos de 16 bits los cuales denotan los puntos finales de un circuito virtual. Los números de puerto son empleados para el direcciónamiento en la capa de transporte. Como son campos de 16 bits, una computadora puede establecer teóricamente hasta 65535(2^16-1) conexiónes TCP diferentes concurrentemente.

Los puertos en el rango de 0 a 1023 son denominados *puertos bien conocidos* empleados para accesar servicios estandarizados. Un sistema operativo puede ser configurado con servicios adicionales, que pueden ser accesados en puertos reservados dentro de algun rango superior a 1023.

Para definir el punto final de una comunicacion, TCP utiliza el concepto de puerto. A cada puerto se le asigna un número, que junto con la dirección IP de una computadora, permiten definir el punto de acceso a una determinada aplicación, por ejemplo se podria decir que el servidor de ftp de la DCAA se encuentra en

(132.248.27.10,21). La combinacion de la dirección IP y el puerto usado para la comunicación es llamado *socket address*.

Podemos pensar que TCP va colocando los segmentos en una cola a medida que van llegando, y que hay una cola para cada conexión. Una conexión está definida por dos puntos finales, por ejemplo si la computadora con dirección 132.248.10.1 se quiere comunicar con el servidor ftp antes mencionado, tal vez utilizando el puerto 1038, existiria una conexión definida por lo siguiente: (132.248.10.1,1038)(132.248.27.10,21) con su cola correspondiente. Extendiendo más éste concepto, si otro usuario quiere comunicarse al mismo servidor a través de su computadora (132.248.67.3) y el número de puerto 988, existiría otra conexión definida por (132.248.67.3,988)(132.248.27.10,21), que también tendria esta cola para que el software de TCP fuera colocando los segmentos que llegaran.

Para poder tener el mecanismo de multiplexaje, cada aplicación deberá negociar con el sistema operativo, para obtener un número de puerto. Los datagramas que la aplicación envia a través de ese puerto, contendrán su identificador en el campo puerto fuente. En caso de que se envie un datagrama hacia un puerto destino no existente, el software de TCP/IP en la máquina destino enviar un mensaje de ICMP Port Unreachable.

Para saber en que puerto una aplicación deberá comunicarse con otra, en caso de que la aplicación sea de uso común, por ejemplo whois, bootp, existen puertos bien conocidos, asignados previamente.

Como vemos entonces, la base de la estructura de comunicación de TCP no se basa en los puertos, sino en las conexiones, ya que pueden existir muchas aplicaciones hablando hacia el mismo puerto remoto, sin que los segmentos de datos se mezclen.

Algunos ejemplos de "puertos bien conocidos" son listados a continuación:

Número de Puerto	Aplicación	Descripción
20	Datos-FTP	Puerto de transferencia de datos de la transferencia de archivos.
21	FTP	Puerto de diálogo de la transferencia de archivos.
23	TELNET	Puerto de conexión remota telnet.
25	SMTP	Simple Mail Transfer Protocol.
110	POP	Post Office Protocol:Usado para el servicio de correo para PCs.
80	HTTP	HyperText Transfer Protocol.
513	rlogin	remote login
514	rsh	remote shell.

3.15.2 Números de Secuencia y Acuses de Recibo (Akcnowledgement)

Cada uno es una palabra de 32 bits, los cuales dan la posición de los datos dentro del flujo de datos intercambiado durante una conexión. El *número de secuencia* se refiere al remitente, y el número de *acuse de recibo* (*acknowledgement*) aplica a los números de bytes recibidos del otro lado de la conexión. Debe de quedar claro, que los acuses de recibo(ack) no se refieren a los segmentos, si no a los bytes. *Cada acknowledgement, indica el siguiente octeto que se espera recibir.* Cuando se inicia una conexión cada parte de una conexión TCP genera un número inicial de secuencia cuya mas importante propiedad es que no debe de ser repetido durante el periodo en el cual un paquete pueda permanecer en la red. Sobre la transferencia de datos, el número de secuencia es incrementado por el remitente de acuerdo al número de bytes envíados. El rango numérico de 2^32 provee suficiente protección contra duplicación y traslape, un desbordamiento en la secuencia de números, se empieza otra vez desde cero.

3.15.3 Data Offset

Esta entrada contiene la longitud de la cabecera de TCP en palabras de 32 bits para determinar el comienzo de los datos.

3.15.4 Banderas.

Estas son utilizadas para activar acciones en TCP. Si estas estan colocadas a 1 significa:

URG	Apuntador en el campo URGENT es valido.
ACK	Número de ack valido. El n'mero de acknowledgement es solo invalido en el inicio de una conexión.
PSH	Los datos en el segmento deben de ser pasados directamente a la aplicacion.
RST	Reestablecimento de la conexión, o respuesta a un segmento invalido.
SYN	Petición de iniciar conexión, el segmento debe ser ack.
FIN	Conexión cerrada de un lado debido al fin del flujo de datos desde esa dirección, el segmento debe de ser ack.

3.15.5 Ventana

TCP usa un mecanismo especializado de ventana para resolver dos importantes problemas: transmisión eficiente y control de flujo, el cual se basa en *acuses de recibo positivo con retransmisión*. La técnica requiere de un recipiente para comunicarse con la fuente, mandando un mensaje de regreso llamado acuse de recibo o acknowledge a medida que va recibiendo los datos. El que envía, manda un registro con cada paquete que envía y espera el acuse de recibo. El que envia tambien activa un timer cuando envía un paquete y lo retransmite si el timer expira antes de recibir el acuse de recibo. La figura 3.9 muestra como el protocolo mas simple con acuse de recibo transfiere los datos.

El concepto de "Ventanas Deslizantes", se asegura que la transmisión de flujo(streams) sea eficiente. Tal como se muestra en la figura 3.9, los datos fluyen entre las máquinas en sola dirección a la vez, aun cuando la red sea capaz de realizar comunicaciones en ambos sentidos. La red estará totalmente inactiva durante unos momentos en que las maquinas retarden sus respuestas. Si se imagina una red con largos periodos en la transmisión, el problema es claro: "un protocolo con acuse de recibo positivo desperdicia una parte sustancial del ancho de banda porque debe retrasar el envio de un nuevo paquete hasta no recibir el acuse de recibo del paquete previo".

Los protocolos de *ventana deslizante* usan mejor el ancho de banda de la red porque le permiten al que envía transmitir múltiples paquetes antes de esperar un acuse de recibo. La manera mas facil de conceptualizar la operacion de Ventanas Deslizantes, es imaginandose la secuencia de paquetes a ser transmitidos como lo muestra la figura 3.10 . El protocolo pone una pequeña ventana en la secuencia y transmite todos los paquetes que quepan dentro de ella.

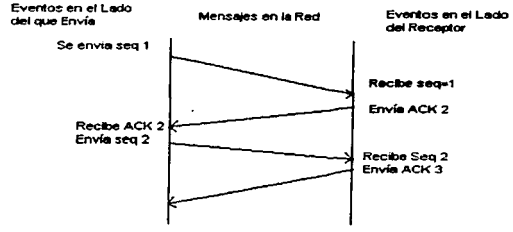


Figura 3.9

Técnicamente, el número de paquetes sin acuse de recibo que puede existir en un momento dado esta limitado por el tamaño de la ventana a un número pequeño y fijo. Por ejemplo, en un protocolo de ventana deslizante con tamaño de ventana 8, el que envía tiene permitido transmitir 8 paquetes antes de recibir un acuse de recibo. Como se muestra, una vez que el que envia recibe un acuse de recibo del primer paquete dentro de la ventana, la mueve longitudinalmente y envía el siguiente. En el lado del receptor, el protocolo mantiene una ventana similar, aceptando y acusando de recibido los paquetes como van llegando.

Figura 3.10 Ventanas Deslizantes

En base a lo anterior, este campo contiene el número de bytes que el recipiente esta capacitado para recibir en su buffer de datos para esa conexión(ventana receptora). El TCP destino emplea esta entrada para controlar el flujo de datos; una ventana de tamaño 0 debera, por ejemplo, efectivamente parar al TCP fuente. El flujo de datos es reinicializado gradualmente incrementando el tamaño de la ventana. Cuando se escoge un tamaño maximo de segmento demasiado pequeño no se aprovecha al máximo la capacidad de transmisión de la red, si por otra parte, éste es demasiado grande, la transmisión se vera afectada por el tiempo que consumira a la capa de red la segmentacion de estos paquetes, para que se ajusten al MTU(Unidad maxima de transferencia para el enlace de red correspondiente). La determinación óptima del tamaño de la ventana durante la transferencia de datos es uno de los algoritmos mas complicados de una implementacion TCP. El tamaño máximo de la ventana es de 204bytes en BSD 4.2 y 4096 bytes en BSD 4.3.

TCP trabaja sobre el principio de la ventana deslizante: cada parte de la conexión puede enviar el número de bytes especificado en la ventana, sin tener que esperar por un ack de la otra parte. Durante el proceso de transmisión, los acknowledgements del receptor de los datos pueden ser simultaneamente recibidos. Esto significa que el emisor no tiene que esperar por el correspondiente acknowledgment después de cada segmento.

3.15.6 Checksum

El checksum es aplicado a la cabecera del protocolo, a los datos y a un pseudo header. El algoritomo del checksum es el mismo que para IP.

En el pseudo header es mostrado en la figura 3.11, un número de campos del header del protocolo IP son liberados al protocolo de transporte junto con el segmento de datos. Estos son incluidos en el cálculo del checksum de TCP para el segmento para prevenir que los paquetes sean incorrectamente enviados por IP.

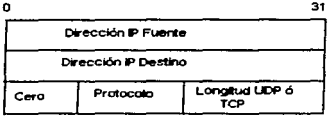


Figura 3.11 Pseudo Header Generado por TCP

Sobre el lado del emisor, un pseudo header es generado en la creación del checksum de TCP y es incluido en el checksum. Los datos en el pseudo header sin embargo, no son transferidos en el segmento.

El campo de checsum es generado por el complemento a 1 a 16 bits del complemento a 1 de la suma de todas las palabars de 16 bits en la cabecera y los datos.

3.15.7 Puntero Urgente.

Junto con el número de secuencia, este es un apuntador a un byte de datos. Este byte de datos es el final de una sección del mensaje, donde, los siguientes datos son considerados importantes. Esta función es denominada "datos urgentes".

3.15.8 Opciones.

TCP tiene además tres opciones: lista final de opción, Número de operación y Maximo Tamaño del Segmento, MSS(Maximun Segment Size), esta última es la más importante de todas, ya que define el tamaño máximo de segmento que las entidades podrán transmitir.

3.15.9 Relleno (Padding).

Si el campo de Options es valido, el padding asegura que los datos comienzan sobre un límite de 32 bits, asi que el desplazamiento de los datos puede apuntarlo correctamente.

3.16 USER DATAGRAM PROTOCOLO(UDP)

UDP es un protocolo connectionless de la capa de transporte.UDP está especificado en el RFC 768 y su dirección de transporte es 17. Sus principales caracteristicas son:

- connectionless.
- Direcciónamiento via el número de puerto.
- checksum de los datos.
- muy simple.
- UDP no garantiza ningún tipo de servicio.
- No ordena los mensajes.
- No cuenta con mecanismos de control de flujo contra la duplicación ni contra la perdida de mensajes.

Simplemente, sirve para que las aplicaciones tengan puntos de entrada a un servicio de transferencia de datos a travez de la red, ya que las capas inferiores no proporcionan el esquema de direcionamiento necesario para llevar esto a cabo.

Como sabemos una arquitectura de red está conformada por un cierto número de capas, algunas más cercanas al hardware y otras más cercanas a las aplicaciones. En TCP/IP son dos los protocolos que dan la cara a las aplicaciones:TCP y UDP.UDP es un protocolo que, como veremos simplemente provee el servicio de multiplexaje entre las aplicaciones y el software de TCP/IP, para que muchas aplicaciones puedan tener las capacidades de comunicación simultanea. TCP además de este servicio, provee capacidades de detección, corrección de errores, ordenamiento de paquetes, y muchas otras que si bien son beneficiosas cuando una aplicación va a mantener una conexión para transmitir gran cantidad de información, son costosas en tiempo y eficiencia. Para aquellas aplicaciones que van a transmitir poca información, y cuyo tiempo de ejecución

es crítico, además de que van a comunicarse en el contexto de una red con pocos errores(por ejemplo una red Ethernet), es mejor utilizar UDP.

UDP permite definir *puertos*, es decir, destinos virtuales dentro de una computadora especifica. Cada puerto es definido a través de un número entero. El sistema operativo de la computadora deber definir un mecanismo especifico para accesar a estos puertos. En Unix este mecanismo es el de *sockets*, en Windows se logra a través de los *winsockets*.

Debido a la poca funcionalidad de UDP, las aplicaciones que lo utilicen deberán tomar la responsabilidad de reparar cualquier error en los datos de transmisión, reordenarlos, etc.

El formato del paquete UDP se ve en la siguiente figura 3.12, esta estructura va encapsulada en la estructura de IP.

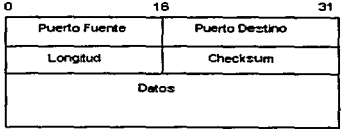


Figura 3.12. Formato UDP

3.16.1 Numeros de Puerto Destino y Fuente.

Como en TCP, los números de puerto son la referencia a los usuarios de la capa de transporte, para los puntos de entrada de las aplicaciones en red.

3.16.2 Longitud.

Esta expresa la longitud completa del datagrama, incluyendo el header del protocolo.

3.16.3 Checksum

El checksum envuelve a los datos, el header del protocolo y el pseudo header. El pseudo Header contiene los mismo campos que el pseudo header para TCP.

3.16.4 Datos.

Contiene los datos de las aplicaciones superiores, que hacen uso de TCP.

Sobre y encima del trabajo ejecutado por IP, UDP solamente provee un número de puerto y un checksum de los datos. A diferencia de TCP, en este caso, no hay acuses de recibo u otras medidas de confiabilidad; sin embargo, la carencia de esas adiciones hace a UDP particularmente eficiente y por lo tanto confiable para aplicaciones de alta velocidad(por ejemplo, sistemas de archivos distribuidos(NFS) o sistemas de directorios, bases de datos y transacciones), las cuales, son solamente apropiadas sobre instalaciones cuyo medio de transmisión es rápido y confiable, como por ejemplo, en Redes de area Local(LANs).

3.17 CONCLUSIONES

IP es el protocolo que le permite a TCP/IP operar eficientemente en ambientes de redes de area local y amplia. A traves de IP y sus protocolos asociados, una interface consistente es provista a los protocolos de las capas superiores, la cual es independiente del medio que se use y de la velocidad de dicho medio.

UDP y TCP proveen los servicios de la capa de transporte de TCP/IP. UDP provee un servicio de datagramas agregando solamente el direcciónamiento de puerto necesario para definir a cual aplicacion deberá ser enviado. TCP, provee además del direcciónamiento, provee un flujo confiable de los datos.

Y actualmente este conjunto de protocolos se ha convertido en un estándar para todos los ambientes de red, dada la popularidad de Internet, la cual hace factible la comunicación entre plataformas diversas, y el acceso a una ilimitada fuente de información y servicios.

IV SISTEMA OPERATIVO UNIX

4.1 INTRODUCCION

UNIX es un sistema operativo desarrollado en los Laboratorios Bell, en New Jersey, Estados Unidos en 1969, los creadores fueron *Ken Thompson y Dennis Ritchie*, cuyo propósito fue el crear un sistema de tiempo compartido, pequeño y de propósito general.

Antes de la creación de UNIX, los Laboratorios Bell participaron junto con la compañía General Electric en un proyecto de desarrollo de sistemas operativos. El objetivo era diseñar un gran sistema multiusuario de nombre *Multics*, el proyecto no culminó porque se trataba de un diseño muy amplio y complejo.

La primera versión de UNIX fue escrita en lenguaje ensamblador de una microcomputadora PDP-7, y al año siguiente Ritchie la instalo en una maquina mas moderna , una PDP-11, y se dedico también a escribir el compilador para el lenguaje de programación C, que acababan de diseñar. En 1973, Thompson y Ritchie escribieron el núcleo de UNIX en el lenguaje C, rompiendo así con la tradición de escribir sistemas operativos en lenguaje ensamblador; con ello se logro que UNIX fuera mas portátil y fácil de modificar.

Un tiempo mas tarde se concedió el permiso para que algunas instituciones no lucrativas tuvieran acceso en la versión de la PDP-11, que ya era muy popular en universidades e institutos de investigación, y eso marco el inicio de una rápida difusión del sistema en todo el mundo. En la actualidad UNIX se considera un estándar virtual para computadoras mutiusuario y ha sido adoptado por una gran cantidad de arquitecturas.

En la Universidad de California en Berkeley un grupo de investigadores entre ellos Bill Joy y Chuck Haley desarrollaron la versión Berkeley de UNIX en 1975, este es el UNIX BSD (Berkeley Software Foundation).

El departamento de defensa de los Estados Unidos DARPA basaron su entorno de computación universal en el sistema UNIX adoptando la versión BSD y fundando posteriormente: 4.1 ESD, 4.2 BSD, 4.3 BSD (1978), etc.

En 1983 AT&T introdujo el UNIX System V Release 1, con este sistema AT&T prometio mantener la compatibilidad ascendente en sus futuras versiones del Sistema UNIX, esto significa que los programas construidos sobre la versión 1 podían continuar operando con versiones futuras del System V.

Esto implica para las diferentes plataformas de hardware que existan se puede adoptar UNIX con la versión BSD o System V. Como el código fuente de UNIX esta escrito en C puede transporte a distintas por lo tanto cada arquitectura adopta su sistema UNIX, entre ellos: SunOs ó Solaris de Sun Microsystems para maquinas SUN, Xenix de Microsoft para microcomputadoras (Intel 80x86), Irix para estaciones de trabajo Silicon Graphics, UNICOS de Cray Research para Supercomputadoras, etc.

UNIX y la mayoría de los sistemas que se ejecutan en él están escritos en lenguaje C, y han servido como demostración de que un sistema operativo interactivo y poderoso no necesariamente es grande y caro, ya sea en equipo o en cantidad de código: puede utilizarse en minicomputadoras de costo reducido y se requirió menos de dos años hombre para el desarrollo inicial del sistema principal. En palabras de sus creadores el objetivo es que, desde el punto de vista del usuario, sea simple y fácil de usar.

4.2 CARACTERÍSTICAS DEL SISTEMA OPERATIVO UNIX

Las características que tiene UNIX hicieron que fuera tan famoso y eficiente, estas son:

- Es un sistema operativo multiusuario, con capacidades de multiprocesamiento y multiprogramación.
- El sistema esta escrito en un lenguaje de alto nivel (Lenguaje C), haciéndolo fácil de leer, entender, y cambiar aun en otras maquinas, a esta característica se le llama portabilidad.
- Dispone de un lenguaje de control programable, llamado shell, csh o ksh.
- Ofrece facilidades para la creación de programas y sistemas, y un ambiente muy propio para las tareas de interconexión de procesos.
- Permite comunicación entre procesos.

- Emplea un sistema jerárquico de archivos, con facilidades de protección de archivos, cuentas y procesos, permitiendo fácil mantenimiento, fácil localización de archivos y eficiente implementación de acceso.
- Todos los archivos poseen el mismo formato de representación (todo en una cadena de bytes) dándole flexibilidad al acceso del contenido.
- Tiene facilidades para redireccionamiento de entradas/salidas.
- La interfaz para accesar los periféricos es consistente para todos, para UNIX todo es archivo, las terminales, el disco, unidades de cinta, impresoras, etc.
- Incluye mas de un centenar de subsistemas, y varios lenguajes de programación.
- Garantiza un alto grado de portabilidad.
- Puede adaptarse a requerimientos de configuración particulares; esto es, puede reconfigurarse en cada instalación.
- Es un buen sistema operativo para programadores y actualmente también para la ejecución de aplicaciones de todo tipo.

UNIX permite que los programas sean independientes de los dispositivos periféricos; la salida de cada programa o utileria del sistema puede ser dirigida a archivos en disco, impresoras o terminales, y existe también la posibilidad de comunicación entre procesos para crear conjuntos arbitrarios y complejos de procesos concurrentes cooperativos.

4.3 ESTRUCTURA DEL SISTEMA OPERATIVO UNIX

El sistema operativo UNIX se basa en un modelo que consta de los siguientes elementos:

- El núcleo o kernel de UNIX.
- El sistema de archivos de UNIX.
- El shell de UNIX (interprete de comandos).
- Programas de usuario y aplicaciones

Esta estructura se muestra en la figura 4.1.

4.4 PROGRAMAS DE USUARIO Y APLICACIONES

UNIX incluye múltiples esquemas para crear, editar y procesar documentos. Existen varios tipos de editores, formadores de texto, macroprocesadores para textos, formadores de tablas, prepocesadores de expresiones matemáticas, y un gran número de utilerias diversas.

Es posible que el programador cree aplicaciones mediante lenguajes como C, C++, Pascal, Algol, etc. Existen un gran numero de bibliotecas (funciones optimas para realizar ciertas tareas) que permiten reforzar las aplicaciones que el programador vaya creando, y que solo necesita incluir en su programa. La gran ventaja es que todos los programas que realice se podrán transportar a otros sistemas y solo bastara compilar de nuevo y ejecutar.

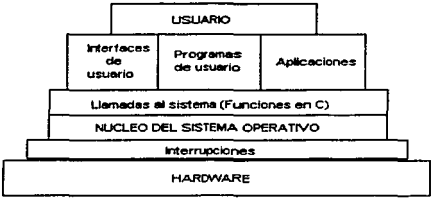


Fig 4.1 Estructura del Sistema Operativo UNIX

En estos tipos de ambientes es fácil de encontrar todo tipo de bases de datos que operan bajo ambiente UNIX y que por su característica, de sistema multiusuario, y la facilidad de conectividad con otras maquinas, las hacen tan eficientes y fáciles de usar.

También , se encuentran un gran numero de aplicaciones para la industria, para la investigación así como también para la iniciativa privada.

4.5 EL SHELL

El shell es el programa que recibe los comandos y los transforma en un conjunto de primitivos del sistema operativo. Es una interfaz entre el kernel del sistema operativo y

el usuario, en el que se le han incluido un gran numero de funciones (comandos), que pudieran considerarse la implementacion de los conceptos mas poderosos y elegantes del sistema operativo UNIX.

Algunas de sus tarea son:

- Es interactivo con el usuario.
- Es un lenguaje de control.
- Es un lenguaje de programación.
- Permite tener acceso a programas mas completos como editores de texto, hojas de calculo y bases de datos.

Visto como lenguaje de programación, incluye estas caracteristicas:

- Sustitución de metacaracteres.
- Control del medio ambiente.
- Procesos sincronos.
- Redireccionamiento de la E/S de datos.
- Procesamiento de archivos de comandos.
- Variables.
- Estructuras de control: secuenciacion, iteración condicional, selección, y otras mas.
- Paso de parámetros.

4.6 INTERFACES DE USUARIO.

Existen diferentes tipos de interfaces con características especiales, cada una de ellas que permiten al usuario comunicarse con el sistema. Algunos de estos son:

Bourne Shell: Localizado generalmente en /bin/sh; es la primera interface creada para UNIX por Steve Bourne de los Labs Bell; esta interface es la que todos los sistemas UNIX tienen por default.

C shell: Localizado generalmente en /bin/csh; esta versión fue desarrollada por Bill Joy en la Universidad de Berkeley. Su característica principal es que ofrece un gran numero de herramientas para mantener una historia de los comandos que se han

ejecutado, mayor control de procesos, facilidad para renombrar comandos y su sintaxis de su lenguaje es muy parecido al lenguaje C.

Korn Shell: Localizado generalmente en /bin/ksh; creado por David Korn de los laboratorios Bell, ofrecen herramientas que aumentan las características del Bourne Shell.

Los tres anteriores son los principales, aunque también tenemos a *tcsh*, *bash*, etc., entre otros, que estan basados en los anteriores pero con mucho más características.

4.7 RESPONSABILIDADES DEL SHELL.

- Ejecución de programas.
- Sustitución de variables y nombres de archivos.
- Redireccionamiento de E/S.
- Comunicación de procesos.
- Control de ambiente.
- Lenguaje de Programación.

4.8 EJECUCIÓN DE PROGRAMAS.

Es el responsable de analizar la línea de comandos para determinar la acción a realizar. Para que el shell pueda ejecutar el comando debe identificarlo del conjunto de caracteres introducidos por el usuario. Generalmente, en la línea de comandos se sigue la siguiente sintaxis:

comando *parámetros*

Una vez que se han identificado todas las partes en la línea de comando el shell busca el comando con ayuda del *PATH* ó ruta que tiene seleccionada.

4.9 SISTEMA DE ENTRADA/SALIDA Y REDIRECCIONAMIENTOS

El sistema de entrada/salida se divide en dos sistemas complementarios: sistema de E/S estructurado por bloques, y sistema de E/S por caracteres. El primero se maneja para el manejo de discos y cintas magnéticas, y emplea bloques de tamaño fijo (512 o 1024 bytes) para leer o escribir. El segundo se emplea para la atención a las terminales, líneas de comunicación e impresoras, y funciona byte por byte.

En general, UNIX emplea programas especiales (escritos en C) conocidos como manejadores (drivers) para atender a cada familia de dispositivos de E/S. Los procesos se comunican con los dispositivos mediante llamadas a su manejador. Además, desde el punto de vista de los procesos, los manejadores aparecen como si fueran archivos en los que se lee o se escribe, logrando con esto gran homogeneidad.

Cada dispositivo se estructura internamente mediante descriptores llamados numero mayor, numero menor y clase (de bloque o de caracteres). Para cada clase hay un conjunto de entradas, en una tabla, que apuntan a los manejadores de dispositivos. El numero mayor se usa para asignar el manejador correspondiente a una familia de dispositivos. El numero menor del dispositivo pasa al manejador como un argumento, y este lo emplea para tener acceso a uno de varios dispositivos físicos semejantes.

Las rutinas que el sistema emplea para ejecutar operaciones de E/S están diseñadas para eliminar las diferencias entre los dispositivos y los tipos de acceso. No existe distinción entre acceso aleatorio y secuencial, ni hay un tamaño de registro lógico impuesto por el sistema. El tamaño de un archivo ordinario esta determinado por el numero de bytes escritos en el; no es necesario predeterminar el tamaño de archivo.

El sistema mantiene una lista de áreas de almacenamiento temporal (buffers), asignados a los dispositivos de bloques. El kernel usa estos buffers con el objeto de reducir el trafico de E/S. Cuando un programa solicita una transferencia, se busca primero en los buffers internos para ver si el bloque que se requiere ya se encuentra en la memoria principal (como resultado de una operación de lectura anterior). Si es así, entonces no será necesario realizar la operación física de E/S.

Debido a que los manejadores de los dispositivos son programas escritos en C, es relativamente fácil reconfigurar el sistema para ampliar o eliminar dispositivos de E/S en la computadora, así como incluir tipos nuevos.

Las llamadas del sistema de mas bajo nivel hacen referencia a un flujo de entrada y salida a través de un numero de identificación llamado "descriptor de archivo" (file descriptor o fd simplemente).

Cada proceso en UNIX tiene asignados, por convención, tres canales de entrada y salida.

entrada estándar fd=0

salida estándar fd=1

salida de error estándar fd=2

Estos tres canales están normalmente asociados en forma directa con el teclado y la pantalla del usuario.

El shell es el responsable de cambiar la entrada y salida estándar para los procesos que genera. Se rastrea la línea de entrada, si se encuentran caracteres como <, <<, >,>>, que indican redireccionamiento de entrada y salida respectivamente, entonces el shell ejecuta el comando, pero cambia su salida o entrada estándar por la especificada en la línea de comandos.

La entrada estándar que se tiene por default es el teclado y la salida estándar es la pantalla.

4.10 PROCESOS

Un proceso es una instancia de un programa en ejecución. Existen dos formas de ejecutar los procesos en UNIX:

- *foreground*. Ejecución sucesiva de procesos.
- *background*. Ejecución asincrona de procesos.

Normalmente todos los comandos los ejecutamos en foreground, esto es, un comando tras otro y el prompt no lo va a liberar el sistema cuando haya terminado de realizar el comando. Si una línea de ordenes termina con el carácter &, el shell no espera a que se termine de ejecutar esa línea antes de pedir la siguiente. El control

regresa al usuario inmediatamente y se le reporta el numero de identificación de proceso (PID Process IDentification). Dicho proceso se ejecutara en un modo que se conoce como background en la terminología de UNIX (es decir, se ejecuta independientemente de la terminal desde la que se invoco).

En general los tipos de procesos que tenemos en UNIX son:

- Interactivos.
- Batch o en lote.
- Daemons

4.10.1. INTERACTIVOS

Son aquellos procesos que están asociados a una terminal ya sea en background o foreground.

4.10.2 BATCH O EN LOTES

Es un conjunto de trabajos que se van encolando. Es una cola de trabajo y que no esta asociada a una terminal.

4.10.3 DAEMONS.

Son procesos en espera de ejecución y van a estar corriendo en background. Funcionan bajo un esquema cliente/servidor.

4.11 COMUNICACIÓN DE PROCESOS.

De la misma forma que el shell busca caracteres de redireccionamiento, también se buscan caracteres de conexión. Estos caracteres permiten la comunicación entre procesos, y es nuevamente el shell el responsable de enlazarlos. El carácter reconocido para comunicación o entubamiento es |.

Cuando el shell encuentra un carácter de entubamiento redirecciona la salida del comando identificado antes del pipe hacia la entrada del comando identificado después del pipe. Una vez que ha hecho eso ejecuta ambos procesos.

4.12 EL SISTEMA DE ARCHIVOS

El sistema de archivos de UNIX esta basado en un modelo arborescente y recursivo, en donde los nodos pueden ser tanto archivos como directorios, y estos últimos pueden contener a su vez directorios y archivos. Debido a esta filosofía, el manejo del sistema es por muy pocas ordenes, que permiten una gran gama de posibilidades. Todo archivo en UNIX esta controlado por múltiples niveles de protección que especifican los permisos de acceso al mismo.

La raíz del sistema de archivos (conocida como root) se denota con el símbolo /, y de ahí se desprende un conjunto de directorios que contienen todos los archivos del sistema. Cada directorio funciona a su vez como la subraíz de un nuevo árbol que depende de el, y que también puede estar formado por directorios o subdirectorios y archivos. Un archivo siempre ocupara el nivel mas bajo dentro del árbol, porque de un archivo no pueden depender otros, es decir, los archivos son hojas del árbol.

Se define de manera unívoca el nombre de todo archivo (o directorio) mediante lo que se conoce como ruta (path name) o trayectoria: el conjunto completo de directorios , iniciando en root (/), por los que hay que pasar para poder llegar al directorio o archivo en cuestión.

Desde el punto de vista del usuario, el sistema de archivos es arborescente con solo tres tipos de componentes:

4.12.1 ARCHIVOS ORDINARIOS

Un archivo ordinario se emplea para almacenar información, puede contener un programa, el texto de un documento, o el registro de cualquier tipo de información que una computadora pueda procesar.

Existen varios tipos de archivos ordinarios: archivos de texto, archivos binarios, etc. El sistema no espera una estructura particular, solo los programas del usuario se encargan de la estructura particular de cada archivo.

4.12.2 DIRECTORIOS.

Cada usuario tiene un directorio para sus propios archivos, y puede crear también subdirectorios formando una estructura jerárquica. El sistema mantiene también varios directorios para su uso propio, como por ejemplo:

- /bin Contiene programa ejecutables y utilerias.
- /etc Contiene programas y archivos de datos para el sistema administrador.
- /dev Contiene archivos especiales.
- /usr Contiene, los directorios de los usuarios, incluyendo mail, bin y news.

Es importante mencionar que, debido a que UNIX es un sistema multiusuario, cada uno de los usuarios inscritos al sistema deberán tener un espacio asignado dentro del árbol. A este directorio se le conoce como directorio HOME.

4.12.3 ARCHIVOS ESPECIALES (Manejadores de dispositivos periféricos)

Cada dispositivo de E/S esta asociado con al menos uno de estos archivos. Los archivos especiales se leen y se escriben como un archivo ordinario de disco, pero las solicitudes de lectura y escritura activan el dispositivo asociado. Los archivos especiales no contienen información; son utilizados para proporcionar un canal conveniente para accesar los mecanismos de E/S. Los nombres de los archivos especiales generalmente indican el tipo de dispositivo al que están asociados.

Hay una entrada para cada archivo especial que reside en el directorio /dev. Los archivos especiales existen para cada línea de comunicación, cada disco, cada unidad de cinta, y para la memoria física. Por supuesto, los discos activos y el archivo de memoria están protegidos contra acceso indiscriminado.

Los nombres de archivos y de dispositivos tienen la misma sintaxis y significado, así que un programa que espera un nombre de archivo como un parámetro puede dársele un nombre de dispositivo; esto permite por ejemplo, mandar una impresión como si se estuviera copiando un archivo.

Finalmente, los archivos especiales están sujetos al mismo mecanismo de protección de los archivos regulares.

4.13 EL KERNEL DE UNIX

El sistema se basa en un núcleo (conocido como kernel) que reside permanentemente en la memoria, y que atiende todas las llamadas del sistema, administra el acceso a los archivos y el inicio o la suspensión de las tareas de los usuarios.

El kernel es un programa escrito casi su totalidad en C con una excepción de una parte para el manejo de interrupciones, que esta escrita en el lenguaje ensamblador del procesador en el que opera. Y tiene las siguientes funciones:

- Creación de procesos, asignación de tiempos de atención y sincronización.
- Asignación de la atención del procesador a los procesos que lo requieren.
- Administración de espacio en el sistema de archivos, que incluye:
  acceso, protección y administración de usuarios;
  comunicación entre usuarios y entre procesos, y
  manipulación de E/S y administración de periféricos.
- Supervisión de la transmisión de datos entre la memoria principal y los dispositivos periféricos.

El kernel reside siempre en la memoria central y tiene el control sobre la computadora, por lo que ningún otro proceso lo puede interrumpir, solo lo puede llamar para que proporcione algún servicio delos ya mencionados. Un proceso llama al kernel mediante módulos especiales conocidos como llamadas al sistema.

El kernel consiste de dos partes principales:

1. Sección de control de procesos.

2. Sección de control de dispositivos.

La primera asigna recursos, programa procesos y atiende sus requerimientos de servicio, y la segunda supervisa la transferencia de datos entre la memoria principal y los dispositivos periféricos.

Cuando se inicia la operación de la computadora se debe cargar en memoria una copia del núcleo, que reside en el disco (esta operación recibe el nombre de bootstrap).

El kernel puede reconfigurarse fácilmente y adaptarse a los requerimientos particulares de hardware que se tenga.

4.14 CONCLUSIONES

En este capitulo se ha mostrado la estructura del sistema operativo UNIX, explicando brevemente cada una de sus partes, más adelante se explicara su importancia en el mundo de las redes, ya que UNIX fue virtualmente el primer sistema operativo en forma, y a sus 25 años de su creación aun sigue vigente, dejando sentir su influencia sobre los sistemas operativos actuales.

V. ADMINISTRACIÓN DE EQUIPOS UNIX.

5.1 INTRODUCCIÓN

A principios de los ochenta, la red DARPA propuso ciertas modificaciones a la universidad de California en Berkeley, lo que ocasionó ciertas modificaciones al ya existente y robusto sistema operativo UNIX. Uno de los cambios fue el soportar los protocolos de la red de la Defensa (TCP/IP). Esto se logró con la versión 4.2 del UNIX de Berkeley (4.2). Posteriormente, la versión 4.3 de BSD incluyo el soporte para comúnicación entre redes con el manejo de routers y dispositivos para comúnicación WAN. Los cambios incluyeron rutinas para mensajes ICMP, algoritmos de retransmisión, parámetros de tiempo de vida en los paquetes y otras cosas. Esto trajo consigo un mayor interés en la comúnidad de los negocios, incrementandose para mediados de los 80s el número de redes locales.

A continuación se presentan los principales archivos y comandos en las que se basa la implantación en UNIX y la visión externa que los usuarios del sistema pueden tener de la implantación de la familia de protocolos TCP/IP. Se trata de archivos de configuración de la red, de los principales comandos de consulta ó administración de TCP/IP.

5.2 LOS ARCHIVOS DE CONFIGURACIÓN

5.2.1 El archivo */etc/hosts*.

UNIX originalmente usaba el archivo */etc/hosts* para llevar el registro de las direcciónes de red de cada host de la red local. Muchos sistemas siguen usando este metodo actualmente. Este archivo contiene información relativa a las diferentes máquinas de red local a la que pertenece el sistema. Tipicamente se administra de manera centralizada y se distribuye a todos los hosts locales. El archivo /etc/hosts de un sistema contiene la tabla de correspondencias entre las direcciónes IP y los nombres simbólicos de estas máquinas. A cada máquina de la red le corresponde una línea del archivo */etc/hosts* de la forma siguiente:

```
132.248.27.10   tzetzal   iimas        # Host de la DCAA, UNAM
```

que indica para cada host:

- La dirección IP [132.248.27.10]
- el nombre del host [tzetzal]
- una lista de alias [iimas]
- uncomentario [# Host de la DCAA, UNAM]

El nombre del host es un nombre del sistema asociado a esta dirección IP. No puede contener espacios, tabuladores, signos #, o caracteres fin de línea. Cada nombre de host debe de ser único.

El alias es otro nombre para el mismo sistema. Típicamente, es un nombre mas corto. Un mismo host puede tener de uno a diez alias.

Cuando se transpasa el límite de la red local, los nombres simbólicos se organizan jerárquicamente como una serie de cadenas que designan conjuntos (dominios) cada vez mas generales. De esta forma una máquina podra llamarse, siguiendo el ejemplo anterior, si la máquina tzetzal se encontrara en el dominio dcaa que a su vez perteneciera al dominio unam.mx (por estar en Mexico y en la UNAM), el nombre completo de esta máquina sería `tzetzal.dcaa.unam.mx` y una línea del archivo */etc/hosts* para ella sería:

```
132.248.27.10      tzetzal.dcaa.unam.mx      iimas      mailhost   www
```

Así, cuando el archivo */etc/hosts* fue insuficiente para los miles (o millones) de máquinas conectadas a una red, este archivo se hizo casi imposible de ser mantenido. Actualmente hay tres sistemas principales para traducir nombres de hosts en direcciónes IP:

- Domain Name Service (DNS). La implantación en Berkeley se llama bind , DNS se expondra con mayor detalle en el siguiente capitulo.
- NIS (Sun Microsystems).
- NetInfo(Next, Inc.).

Aunque estos tres sistemas aparentemente ofrecen la misma funcionalidad, NIS y NetInfo en realidad ofrecen acceso remoto a un archivo */etc/hosts* (o en el caso de NetInfo, a una base de datos de hosts). Tanto NIS como NetInfo deeben usar el servidor de Dominios de Internet para resolver nombres de hosts fuera de la organización.

Estos sistemas son conceptualmente similares a */etc/hosts* en el sentido de que soportan una lista de nombres de hosts y sus direcciónes. Sin embargo , la base de datos de hosts se distribuye sobre la red de tal manera que cada organización solo necesita actualizar sus propias tablas cuando agrega una nueva computadora o cambia la dirección de una existente.

5.2.2 El archivo */etc/networks*

Constituye la base de datos de las redes conocidas que integran la red interna a la que se encuentra conectado el sistema. Existen varias entradas en este archivo, cada una de las cuales contiene:

- el nombre oficial de la red.
- Su dirección IP.
- lista de alias (10 maximo).
- un comentario.

5.2.3 El archivo */etc/protocols*

Este archivo contiene información acerca de los protocolos conocidos usados en la red Internet. Cada línea proporciona información acerca de un protocolo. Una entrada no se puede extender mas alla de una línea y debe tener el siguiente formato:

```
Ip    0     IP    # Protocolo Internet, número de Pseudo Protocolo
Tcp   6     TCP   # Transmission Control Protocol
Udp   17    UDP   # User Datagram Protocol
```

que indica para cada protocolo:

- El nombre del protocolo de Internet (ip) asociado a un número de protocolo.
- El número de protocolo asignado dentro de la familia TCP/IP. El alias es un nombre alternativo del protocolo.

Nota: Todo lo que aparezca entre un signo # y el fin de línea sera interpretado como comentario.

Cuando un datagrama llega a su dirección destino, la capa de IP sabe que el datagrama debe entregarse a alguno de los protocolos de transporte de las capas superiores. Para decidir que protocolo deberá recibir el datagrama, IP busca en el número de protocolo del datagrama. asi, si el número del protocolo del datagrama es 6, IP entregara el datagrama a TCP. Si el número de protocolo es 17, IP entregara el datagrama aUDP. TCP y UDP son los dos servicios de la capa de transporte con los que normalmente se trabaja, pero todos los protocolos mostrados en la tabal IP usan IP como servicio de entrega de datagramas. la figura 5.1 muestra el proceso.

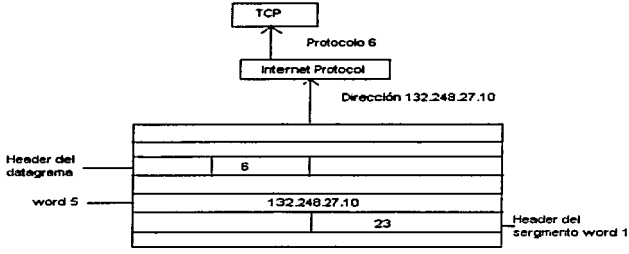


Figura 5.1 Relación del archivo */etc/protocols* con la entrega de datagramas

5.2.4 Números de Puerto.

Después de que IP pasa los datos entrantes al protocolo de transporte, este pasa los datos al proceso correcto de la aplicación. Los procesos de las aplicaciónes (tambien llamados servicios de red)se identifican por los números de puerto, los cuales son valores de 16 bits. Como se recordara, el *número de puerto fuente* que identifica al proceso que envía los datos y el *número de puerto destino*, que identifica al proceso que los va a recibir, están contenidos en la primera palabra del encabezado de cada segmento TCP y paquete UDP.

En los sistemas UNIX, los números de puerto se definen en el archivo */etc/services*. Los números de puerto inferiores a 256 estan reservados para los servicios *bien*

conocidos (como TELNET y FTP) y están definidos en el RFC de Números Asignados. Los puertos numerados entre el 256 y el 1024 se usan para servicios específicos de UNIX, servicios tales como rlogin que originalmente fueron desarrollados para sistemas UNIX. Sin embargo, muchos de ellos ya no son específicos de UNIX.

Los números de puerto no son únicos entre los protocolos de la capa de transporte; los números son solo únicos dentro dentro de un protocolo de transporte específico. En otras palabras, TCP y UDP pueden usar los mismos números de transporte. Es la combinación de protocolo y número de puerto lo que identifica de manera unica al proceso específico al que se le entregaran los datos.

A continuación se muestra un archivo */etc/services* parcial. El formato de este archivo es muy similar al archivo */etc/protocols*. Esta tabla, combinada con */etc/protocols* proporciona toda la información necesaria para entregar los datos a la aplicación correcta. Un datagrama llega a su destino basandose en la dirección destino de la quinta palabra del header del datagrama. IP usa el número de protocolo de la tercera palabra del header del datagrama para entregar los datos al protocolo de transporte correcto. La primera palabra de los datos entregados al protocolo de transporte contiene el número de puerto, el cual le indica al protocolo de transporte la aplicación especifica que recibira los datos.

```
#
# Network services, Internet style
#
tcpmux      1/tcp
echo        7/tcp
echo        7/udp
discard     9/tcp           sink null
discard     9/udp           sink null
systat      11/tcp          users
chargen     19/tcp            ttytst source
chargen     19/udp            ttytst source
ftp-data    20/tcp
ftp         21/tcp
telnet      23/tcp
```

La figura 5.1 muestra el proceso de entrega.

5.2.5 El archivo */etc/inetd.conf.*

Casi todas las aplicaciones de los protocolos TCP/IP están basados en el modelo cliente/servidor. El diseño usual es correr un solo *daemon* que espera por todas las conexiones de red. Cuando una conexión es establecida, este daemon (usualmente llamado inetd) corre el programa apropiado y vuelve a esperar por conexiones.

El archivo */etc/inetd.conf* es empleado para habilitar los servicios de la capa de aplicación. Muestra el nombre común de los servicios y los programas ejecutables de aplicación asociados a ellos, que son activados, para implementar esos servicios. Este archivo debera ser modificado para deshabilitar servicios que no son requeridos. En caso de que este servicio sea requerido en el futuro, es normal deshabilitar el servicio agregando un caracter de comentario (#) en frente de la línea en la cual empieza, en lugar de borrar la línea completamente y perder el formato correcto.

```
#
# Echo, discard, daytime, and chargen are used primarily for
testing.
#
echo      stream   tcp      nowait   root     internal
echo      dgram    udp      wait     root     internal
```

5.2.6 El archivo */etc/services.*

Este archivo contiene información acerca de los servicios conocidos usados en la red IP. Cada entrada da información de un servicio y no puede extenderse mas alla de una línea. El formato es el siguiente:

```
ftp-data          20/tcp
ftp               21/tcp
telnet            23/tcp
smtp              25/tcp              mail
time              37/tcp              timserver
time              37/udp              timserver
```

que indica:

- El nombre del servicio (ftp).

- número de puerto/nombre del protocolo.
- lista de alias.

Los nombres de los servicios generalmente son los que se encuentran en los niveles de aplicación, presentación o sesión, tales como FTP, SMTP y TELNET. El número de puerto corresponde al puerto IP usado por el servicio. El nombre del protocolo con el que esta asociado el servicio. Generalmente se trata de un protocolo de la capa de transporte o de red, tal como TCP o UDP. El alias es un nombre alternativo para el servicio.

5.2.7 El archivo */etc/defaultrouter.*

En este archivo, se le establece la dirección IP del router que usuará la máquina para rutear los datos a través de la red. Este archivo se usa en UNIX System V, para UNIX BSD, este parámetro se especifica en el archivo */etc/hostsconfig*.

5.3 LOS COMANDOS DE ADMINISTRACIÓN

Un cierto número de comandos le permiten a un usuario cualquiera, obtener informacio sobre el estado de la red.

5.3.1 Los comados *hostid* y *hostname.*

Permite obtener las direcciónes Internet y el nombre del sistema del sistema local respectivamente.

5.3.2 El comando *ping.*

Permite probar si una máquina esta activa en este momento (utiliza el protocolo ICMP y los datagramas de ECHO_REQUEST y ECHO_REPLY). Entre las diferentes opciónes, se muestra la opción -s.

5.3.3 El comando *arp*.

Permite visualizar el contenido de la tabla del protocolo *arp* que da la correspondencia entre las direcciónes Ethernet e IP de las máquinas consultadas previamente. A continuación se muestran unos ejemplos.

Ejemplo:

Para desplegar la tabla entera, se usa el comando `arp -a`.

```
$ arp -a
Net to Media Table
Device   IP Address               Mask       Flags   Phys Addr
------ -------------------- --------------- ----- ---------------
hme0    redpro               255.255.255.255       08:00:20:80:1c:c9
hme0    gw-reduno2.reduno.com.mx 255.255.255.255       00:00:0c:0c:ef:43
hme0    redcat               255.255.255.255       08:00:20:81:e8:a9
hme0    151-114.reduno.com.mx 255.255.255.255       08:00:20:7e:53:4f
hme0    reduno               255.255.255.255 SP    08:00:20:7e:35:6a
hme0    BASE-ADDRESS.MCAST.NET 240.0.0.0        SM    01:00:5e:00:00:00
```

La opción *-d* permite eliminar manualmente un host de la tabla de host de la tabla ARP local mientras que la opción -s permite agregar un nuevo host a dicha tabla usando la siguiente sintaxis.

```
arp -d hostname
arp -s hostname
```

Cabe mencionar que todas las entradas recibidas a través del protocolo ARP son temporales. Los valores se mantienen en la tabla un tiempo finito y se borran cuando su tiempo de vida expira (20 min. en Solaris 2.x). Los nuevos valores se obtienen a través del protocolo ARP. Asi, si alguna interface remota cambia, la tabla local se ajusta y continua la comunicación. Por otro lado, los valores que se agregan manualmente, son permanentes.

5.3.4 El comando *netstat.*

Como su nombre lo indica, permite obtener información acerca de la actividad de la red del sistema. Suministra información sobre los diferentes socketes del sistema. Debe recordarse que cada socket utilizado para comunicaciones en la red esta unido a un puerto de uno de los protocolos TCP o UDP. Mediante la opción -i se obtienen las estadísticas sobre las interfaces del sistema local (paquetes emitidos y recibidos, errores y conexiones, red local y nombre del sistema).

Ejemplo

La opción -r aporta información sobre las tablas de enrutamiento.

```
$ netstat -rn
Routing Table:
  Destination           Gateway           Flags  Ref   Use    Interface
-------------------- -------------------- ----- ----- ------ ---------
127.0.0.1            127.0.0.1            UH        0  59716  lo0
192.100.183.176      192.100.183.178      U         3   1710  hme0
224.0.0.0            192.100.183.178      U         3      0  hme0
default              192.100.183.177      UG        0 339852
```

Ejemplo

Finalmente, la opción -a proporciona información sobre los diferentes puertos TCP y UDP (ademas de la información sobre los sockets).

```
$ netstat -a
UDP
   Local Address       State
-------------------- -------
      *.sunrpc         Idle
      *.*              Unbound
      *.32771          Idle
      *.32772          Idle
      *.name           Idle
      *.biff           Idle
      *.lockd          Idle
```

5.3.5 El comando *ifconfig*.

El comando *ifconfig*, es usado para asignar una dirección a una interface de red yu/o para configurar los parámetros de red de la interface. Ifconfig debe de ser usada en tiempo de inicializacion de la máquina para definir la dirección de red de cada interface de red presente sobre la máquina; ademas puede ser usada posteriormente para redefinir la dirección de una interface u otros parámetros operativos. Empleado sin opciones, ifconfig despliega la configuración para una interface de red. Solamente el superusuario puede modificar la configuración de una interface de red.

El parámetro interface es una cadena de la forma nombre fisico-unidad, por ejemplo *le0* o *hme0*.

Ejemplo

```
      $ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet 200.33.150.20 netmask ffffff00 broadcast 200.33.150.255
```

Para la configuración manual de una interface de red, seria:

```
$ifconfig le0 132.248.27.10 netmask 255.255.255.0
```

El sistema operativo Solaris permite múltiples interfaces lógicas asociadas con una interface de red física. Esto permite a una máquina tener multiples direcciónes IP, aun cuando solo tenga una interface de red. Las interfaces de red tienen nombres de la forma driver-nombre fisico-unidad-número, mientras que las interfaces lógicas , tienen nombres de la forma driver-nombre fisico-unidad-número logico-unidad-número. Una interface física es configurada usando el subcomando *plumb*, el cual se encarga de habilitar la interface de red, por ejemplo:

```
$ ifconfig le0 plumb
```

Una vez que una interface física ha sido activada, interfaces locales adicionales pueden ser configuradas simplemente nombrandolas en los subsecuentes usos del comando ifconfig. Por ejemplo, el comando:

```
$ ifconfig le0:1 132.248.27.11 netmask 255.255.255.0
```

debera activar la interface lógica asociada con la interface física le0. Esto puede ser util, cuando se desea que una máquina tenga dos direcciónes IP, como sucedia en los casos de los sitios Web, cuando empresas realizan el hosteo de varios sitios, a cada uno se le asigna una dirección IP diferente, pero en realidad estan haciendo referencia a la misma máquina.

5.3.6 El comando route.

El comando route manipula manualmente las tablas de ruteo mantenidas normalmente por el daemon del sistema *routed*, o a través de rutas por default y mensajes de redirección de los ruteadores. *route* permite al administrador del sistema operar directamente para la red o host indicado por *destination*:

sintaxis:

```
route [ -fn ] add | delete [ host | net ] destination [ gateway [
metric ] ]
```

El argumento gateway, si esta presente, indica el router de la red al cual los paquetes deberan ser direccionados. El argumento *metric* indica el número de "saltos" o el costo para llegar al destino (destination). *Metric* es requerido para la opcion add.

Ejemplo:

```
route add default 192.100.183.254 1
```

Para el ejemplo anterior, se definió el router por *default*, para el host actual.

5.4 CONCLUSIONES.

En este capitulo, se ha mostrado los principales comandos y archivos de configuracion para la configuracion de los servicios de red a traves de TCP/IP.Cabe mencionar, que solo se mencionaraon los archivos y comandos comunes para la administracion de TCP/IP , tanto para BSD, como para System V.

Cabe mencionar, que la sintaxis puede cambiar dependiendo de la variante de UNIX que se emplee(System V o BSD), los archivos y comandos mostrados en este capitulo se hicieron sobre el Sistema Operativo Solaris 2.x, el cual esta basado en la variante de System V, se tomo dicho sistema operativo, ya que actualmente el Sistema Operativo Solaris de Sun, es uno de los lideres del mercado de UNIX.

Tambien se tomo de referencia el sistema operativo NEXT, el cual esta basado en BSD.

No se mostraron otros comandos de administracion de TCP/IP bajo UNIX, debido a que estos dependen de la plataforma en que se trabaje, solamente se presentaron algunos de los archivos y comandos comunes, en las variantes de UNIX.

VI. CONECTIVIDAD DE UNIX CON INTERNET

6.1 INTRODUCCION

En la arquitectura TCP/IP, la capa de aplicación es la responsable de la interface entre las aplicaciones del usuario final y los servicios de la capa de transporte.; el servicio de aplicación no es la aplicación del usuario final, simplemente provee las interfaces a esa aplicación para las comunicaciónes en red. Hay varios servicios de la capa de aplicación para conjuntar los diferentes tipos de aplicaciones. Hay además utilerias de administración. Por ejemplo, *telnet* es un servicio para soportar servicios de terminales, el Protocolo de Transferencia de Archivos(FTP)engloba las aplicaciones de transferencias de archivos y el protocolo SMTP (Simple mail Transfer Protocol) se encarga de las aplicaciones de correo. En este capitulo se mostraran los principales servicios disponibles y una explicación de su operación.

6.2 MODELO CLIENTE SERVIDOR

Los servicios de aplicación de TCP/IP se dice que tienen una relación de cliente/servidor. Estos significa que un servicio de la capa de aplicación consiste de dos partes complementarias, una sobre cada uno de los nodos que estan intercambiando información. El origen de una conexión es llamado el *cliente* y el receptor es llamado el *servidor.*

Un servidor espera por conexiones entrantes o peticiones. Los clientes hacen conexiones o peticiones a los servidores como sean requeridas. Una vez que la conexión es aceptada y abierta, los clientes hacen peticiones y los servidores dan respuestas o replicas a esas peticiones. Los servidores no inician una transacción. Los servidores no puede comúnicarse direcatmente con otro servidor y los clientes no pueden comúnicarse con otros cliente. Así, para establecer conexiones y transferir información, el servidor y el software deben operar en la capa de aplicación.

El servidor es un programa que ofrece un servicio, este servicio puede accesarse a través de la red. Los servidores aceptan peticiones, ejecutan un servicio, y regresan el resultado al que lo solicito. Un cliente envía peticiones al servidor, y espera la respuesta.

Un servidor comienza a ejecutarse antes que comience a interactuar con los clientes y no termina después de que envía las respuestas a un cliente determinado. Un cliente es un programa que hace una petición y espera una respuesta, usualmente termina después de hacer uso del servidor un número finito de veces.

Un servidor espera las peticiones por un puerto bien conocido al cliente, el sistema operativo le asigna al cliente un puerto arbitrario que no se encuentre en uso. En UNIX los puertos para un determinado servicio se especifican en el archivo */etc/services* y los servidores que van a atender a los clientes por un determinado puerto se especifican en el archivo */etc/inetd.conf*, los cuales son levantados por el daemon de *inetd*, el cual es un servidor que recibe todas las peticiones de los clientes (todo esto, con el fin, de que no se malgasten recursos, por el hecho de tener a todos los servidores activos) y se encarga de levantar al servidor apropiado para que atienda al cliente.

En general la estructura de un servidor puede definirse como sigue:

- Abrir un puerto: El maestro abre el puerto bien conocido.
- Esperar a un cliente: El maestro espera a que un cliente le envie una petición.
- Escoger un puerto: Si es necesario, el maestro asigna un nuevo puerto para esta petición y le avisa al cliente.
- Iniciar el esclavo: El maestro inicia un esclavo para manejar la petición, que termina cuando acaba de atender la petición.
- Continuar: El maestro regresa al paso Esperar un cliente.

Debido a que los servidores son software, una computadora con un sistema operativo que soporta multitárea (UNIX o Windows NT) pueden tener diferentes tipos de clientes y servidores corriendo al mismo tiempo. Así, dichas computadoras con multitarea pueden administrar conexiones múltiples dentro de sus servicios y realizar conexiones externas a otras computadoras en los que externamente aparecen como conexiones punto a punto, pero están basadas en la arquitectura cliente servidor para cada conversación.

Así dos piezas diferentes de software son requeridas para habilitar las comunicaciónes. Por ejemplo, si se esta usando una terminal Telnet, tanto la computadora local como la remota deberán requerir Telnet, pero la computadora local requiere el cliente Telnet y el host o computadora remota requiere el servidor Telnet.

Bajo UNIX los servidores son llamados *daemons*, así el servidor Telnet es comúnmente llamado *in.telnetd*, quue significa el daemon Telnet, y así por el estilo. A continuación se muestran algúnos de los daemons mas usuales en la nomenclatura del sistema operativo ***Solaris 2.x***:

Servicio	Servidor	Puerto	Descripcion
telnet	In.telnetd	23	Servicios de acceso a terminal
FTP	In.ftpd	20/21	Servicios de transferencia de archivo
SMTP	Sendmail	25	Servicios de correo electrónico
HTTP	Httpd	80	Protocolo de Transferencia de hypertexto
POP3	Popper	110	Protocolo de Oficina Postal

6.3 PROTOCOLO TELNET

Antes de que las PCs fueran comúnes los usuarios se debían de conectar auna computadora usando un terminal de caracteres. La terminal estaba provista de un teclado y una pantalla para desplegar la salida de programas que corrian sobre la computadora. Las computadoras de redes de datos crecieron como los usuarios que deseaban accesar a muchas computadoras. Telnet fue diseñado para proveer este servicio, permitiendo a los usuarios accesar a todas las computadoras que estubieran conectadas a la red. La máquina local deberia aparecer como si estuviera directamente conectada al servidor remoto. Ellos deberian emplear comandos estándar sobre sus máquinas locales para accesar cualquier tipo de computadora.

Telnet es una interface de la capa de aplicación para un acceso a terminal. Las terminales pueden conectarse en dos formas:

- Terminales reales conectadas directamente a hosts corriendo TCP/IP.
- Computadoras corriendo emulación de terminal y software TCP/IP.

El protocolo Telnet es un protocolo que permite a un usuario en una computadora remota pueda establecer una conexión de TCP hacia otra, y que esta computadora reciba lo que el usuario teclee en la primera. TELNET tambien lleva la salida de la máquina remota a la terminal del usuario

6.3.1 EL ROL DE TELNET.

Telnet ofrece tres servicios básicos, primero define una terminal virtual, que sirve para presentar una interfaz común hacia sistemas remotos. De esta manera los programas que utilizen esta manera de interactuar con sus usuarios, no tienen que conocer los detalles de todos los sistemas remotos posibles. En segundo lugar, incluye un mecanismo a través del cual el cliente y el servidor pueden negociar, opciones de interaccion. Finalmente, Telnet trata a ambos extremos de la comunicación de una manera simétrica, es decir, cualquiera de ellos puede negociar las opciones que definiran la manera de interacción.

La principal funcionalidad que telnet clama es que provee una compatibilidad entre los sistemas de computo así que las aplicaciones del cliente final pueden tener acceso a servidores remotos sin tener conocimiento previo de como soportar algún tipo particular de terminal. El uso específico es permitir a terminales remotas conectarse a una aplicación del servidor anfitrion, quizas a través de una computadora intermediaria, y que aparecen como si estuvieran directamente conectadas a ese host.

El servicio que el usuario ve, es el servicio de terminal del host remoto. Una vez conectado, ellos son validados por identificadores de usuario (UID) y contraseñas, así, puede entonces, tener acceso a aplicaciones autorizadas.

Telnet tiene que hacer parecer a la conexión cliente, como otra terminal de usuario a su sistema operativo, activando todos los programas necesarios de validación y creando un ambiente para cada conexión lógica que es hecha a el. Una vez que un usuario termina la sesión de telnet sobre el servidor, este es responsable de terminar la conexión lógica que fue hecha a través de la red.

En la figura 6.1 todos los usuarios pueden accesar al `host A` con la misma funcionalidad como si ellas estuvieran conectadas directamente a ese host, como por ejemplo, la terminal A. Cada conexión debe aparecer a ese host como si fuera una terminal directamente conectada a ella, así un servicio de transporte, es requerido, así, Telnet normalmente opera sobre TCP.

6.3.2 EL AMBIENTE DE TELNET

Telnet fue diseñado para un ambiente de diferentes tipos de arquitecturas de computadoras, sistemas operativos y terminales, es decir, un ambiente heterógeneo. Muchos tipos de terminales reales deben de ser soportadas. No deberán ser hechas acepciones, acerca del tipo de terminal que se usara para comunicarse con algún tipo de host. Aun cuando muchas terminales operan sobre el mismo conjunto básico de caracteres (ASCII), sus caracteristicas avanzadas son incompatibles.

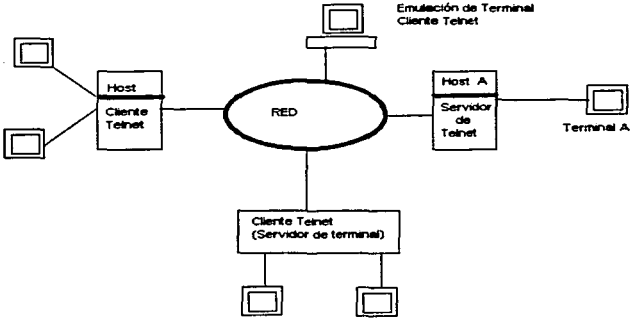


Figura 6.1. Arquitectura de Telnet

Telnet intenta coincidir terminales y hosts. Cuando una terminal hace primero una conexión, el protocolo comienza asumiendo que la terminal es del tipo mas básico posible; Un terminal que tiene un teclado, que produce una salida línea por línea, usa echo local y emplea el conjunto de caracteres ASCII, así como caracteres simples de control, como salto de carro, inicio de línea, e inicio de forma. Esta es una definición de la antigua terminal TTY (Teletype). Esta definición basíca es referida como una Terminal de Red Virtual(*NVT*: Network Virtual Terminal). Sin tomar en cuenta que tan sofisticada sea

la terminal real, en el inicio de la conexión esta no es mayor que una NVT. Como una NVT, todas las terminales pueden comúnicarse con muchos tipos de computadoras; este es el nivel mas básico de compatibilidad entre muchas terminales y computadoras.

Lo que el usuario teclee en la terminal sera transformado al estándar NVT, y una vez que el programa que esta interactuando con el usuario genere una salida, este será transformado de NVT al formato propio de la terminal. El NVT comprende la secuencia de caracteres para retorno de carro, interrumpir proceso, etc.

Comenzando con el nivel mas básico de comunicación, Telnet provee un mecanismo estándar para aceptar la funcionalidad de la Terminal real y como es usada con su aplicación. Este proceso es referido como Negociación telnet.

6.3.3 NEGOCIACIÓN TELNET

Para tomar la ventaja de caracteristicas mas sofisticadas que las terminales reales pueden tener disponibles y cuales el host puede estar habilitado para entender, Telnet ejecuta una negociación, un proceso que establece las capacidades de la terminal partiendo de la base de NVT. Así, Telnet emplea un conjunto de comandos especiales *IAC,DO,IAC DON'T,IAC WILL, IAC WON'T*, como una forma conversaciónal de negociar las facilidades que terminales y hosts pueden soportar. Esta negociación puede ser compleja y ardua y puede tomar muchos segundos. Los comandos especiales frecuentemente usados IAC,DO, TERMINAL TYPE donde algúna de las conexiones sugiere el uso de algún tipo de terminal real, algunos de estos comandos se muestran en la siguiente tabla:

Código	Valor Decimal	Explicación
SE	240	Fin de la subnegociación
NOP	241	No operacion
DM	242	Marca de datos
BRK	243	Break
IP	244	Interrupcion de proceso; envía una interrupcion al proceso remoto

Código	Valor Decimal	Explicación
AO	245	Abortar Salida; limpia todas las salidas pendientes hasta la siguiente petición de comando
AYT	246	Estas alli (Are you There); petición al lado remoto y regresa algúna evidencia de que todavia esta aun alli
EC	247	caracter de boorado; borra caracteres a la izquierda
EL	248	Borra Linea; Borra la línea actual de datos
GA	249	Adelante(Go ahead); un modo de operación half-duplex requerido para las terminales que puede solamente transmitir o recibir en un tiempo dado
SB	250	Comienza subnegociación
WILL	251	deberá negociar comando
WONT	252	Negociación de comando fallida
DO	253	Hacer Negociación de comando
DONT	254	No hacer negociación de comando
IAC	255	Interpretar como comando (el siguiente)

Si la negociación falla, la terminal puede aun interactuar con el host remoto, pero solamente usando las facilidades de NVT, la cual da solamente operación de línea por línea. La base de la negociación es a través de peticiones al lado remoto para usar una caracteristica, por ejemplo:

```
IAC,DO,ECHO
```

Esto es "Deseo un echo"(I want you to echo), para la cual la respuesta deberá ser,

```
IAC,WILL,ECHO
```

Esto es "Yo deber hacer echo" o,

```
IAC,WON'T ECHO
```

"No debere hacer echo". Si la petición es negativa el originador no esta capacitado para cambiar del modo local echo a modo echo remoto. Una vez que una

caracteristica ha sido rechazada, forzando al que envio tomar la responsabilidad, esta no puede ser renegociada.

La otra forma de negociar puede ir esta con "Yo deseo hacer" algo o,

```
IAC,WILL,ECHO
```

Esto es indicando "Yo deso echo". La respuesta deberia de ser

```
IAC,DO,ECHO
```

Esto es, "OK, adelante y echo" o

```
IAC,DON'T, ECHO
```

Esto es, "Echo no realizado"

A través de estos comandos de negociación, que son secuencias de caracteres especiales colocadas en el flujo de datos, Telnet esta capacitado para negociar un nivel de servicio, o quiza solamente controlar cuando realizar un echo o cuando alguna caracteristica es activada o no. Hay otra caracteristicas para la cual Telnet tiene códigos de comandos reservados, los cuales no fueron mostrados en la tabla.

Algúnas opciones requieren niveles rápidos de negociación o subnegociación. Una Subnegociación es requerida para una actividad particular cuando ciertos parámetros deben de ser aceptados, por ejemplo, cuando los tamaños de la pantalla deben de ser aceptados.

Las negociaciones suceden principalmente al comienzo de una conexión, y en muchos casos es muy rápido, a través de IAC,DO, TERMINAL TYPE, aunque subnegociones pueden ocurrir en algún tiempo a través de la vida de una conexión. Aparte de manejar las negociaciones y esos caracteres especiales, Telnet envía los datos del usuario en ambas direcciones entre las estaciones de trabajo y los hosts. Estos datos son pasados al ambiente normal del sistema operativo de la terminal.

El servidor de telnet en UNIX que acepta todas las conexiones entrantes de los clientes de Telnet es levantada a través del daemon de inetd.

6.4 FTP (FILE TRANSFER PROTOCOL)

Existen muchos motivos para utilizar la red como una infreestructura que permita accesar archivos depositados en una máquina remota, desde la disminución de costos, hasta la comparticion de datos en sistemas de negocios. Existen dos maneras de tener acceso a archivos remotos: por acceso en línea y por copia de archivos completos.

Primero veremos el protocolo que bajo TCP/IP se utiliza para la transferencia de archivos:FTP.

El Protocolo de Transferencia de Archivos(FTP) se vale de TCP para el envio y recepcion de datos. Aunque puede parecer trivial la labor de un programa de transferencia de archivos una vez que se utiliza TCP, las dificultades que aparecen debido a las diferencias en la forma de nombrar a los archivos, otorgar permisos, y representar los datos, hacen que FTP sea un protocolo complejo. FTP además otorga las siguientes facilidades:

- Interfaz interactiva.
- Especificacion del formato de transferencia.
- Control de autentificación.

FTP controla y realiza la transferencia de datos en base a pares de procesos, como se ilustra en la figura 6.2. En esta se muestra que el sistema cliente y el sistema servidor tienen cada uno un par de procesos. Un proceso de cada lado se encarga de la "sesión de control", es decir, de la transferencia de información que determinara que archivos y como se van a transmitir. Por otra parte, otro proceso de cada lado se encarga de la transferencia en si.

La conexión utilizada para el intercambio de información de control permanece activa durante toda la "sesión" de FTP, es decir de que el cliente proporciona su login y password al servidor, hasta que ese mismo cliente decida salir y terminar la conexión. Las conexiones de datos se crean dinámicamente a medida que se van necesitando nuevas transferencias.

En las computadoras personales, que generalmente no tienen capacidad real de multitárea, una sola aplicación se encarga de las conexiones de control y de transferencia de datos, sin embargo, estas siguen dependiendo de conexiones TCP independientes.

Cuando un cliente se conecta a un servidor, el cliente utiliza un número de puerto aleatorio, asignado localmente, pero contacta al servidor en un puerto bien conocido (21). Cuando se necesita una asociación de transferencia de datos, el cliente utiliza un nuevo puerto local, que asocia a otro puerto bien conocido que FTP utiliza para la transferencia de datos(20). Para evitar problemas de seguridad, el servidor solo aceptara que se abra

una conexión desde un puerto específico del cliente, el cual se notificara a través de la conexión de control, de cliente a servidor.

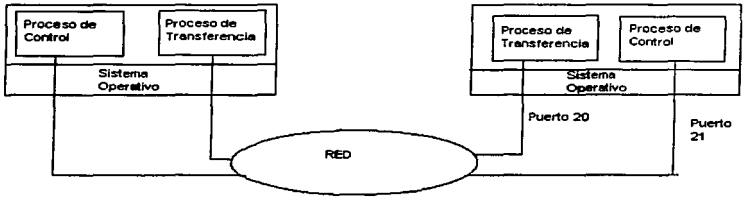


Figura 6.2. Proceso de transferencia de ftp.

El formato que utiliza FTP para la transferencia de mensajes de control a través de la conexión que tiene este fin es Telnet.

Para poder proporcionar el servicio de transferencia pública de archivos, existe una clave especial que generalmente es *anonymous*.

La forma de configurar un sistema de FTP anónimo bajo Solaris 2.x es como sigue:

El usuario anonymous ó ftp deben tener una entrada en el archivo */etc/passwd* y */etc/shadow*, pero no deben tener un password válido(se debe usar en el campo de password cifrado NP, en el archivo */etc/shadow*) ni un shell definido, para que no se realizen conexiones de Telnet hacia esas cuentas.

Para usuarios de ftp anónimo , el daemon de ftp, *in.ftpd*, toma medidas especiales para restringir los privilegios de los usuarios. El servidor ejecuta un comando de cambio de raiz al home directory del usuario ftp. La estructura de directorios de la cuenta de ftp debe de ser realizada en base a las siguientes reglas:

- El propietario del directorio de ftp debe de ser root y no debe de tener permisos de escritura para ninguno.
- Crear dentro del directorio de la cuenta de ftp, el directorio *bin*. Realizar una liga de ese directorio al directorio, dentro de ftp, */usr/bin*. El programa ls (lista archivo o directorios) debe de estar presente en el directorio.

- Crear el directorio usr/lib dentro de la cuenta de ftp cuyo propietario sea el superusuario y que ninguna persona tenga permisos de escritura. Copiar las siguientes librerias compartidas de /usr/sbin dentro de ese directorio:

  *ld.so**
  *libc.so**
  *libdl.so**
  *libintl.so**
  *libw.so**
  *libnsl.so**
  *libsocket.so**
  *nss_nis.so**
  *nss_nisplus.so**
  *nss_dns.so**
  *nss_files.so**
  *straddr.so**

- Crear el directorio *etc* dentro de ftp, cuyo propietario sea root y que no tenga permisos de escritura para nadie. Copias de los archivos *passwd*, *group* y *netconfig* deben de estar presentes para que el comando *ls* trabaje apropiadamente, cabe señalar que dichos archivos deben tener solamente la cuenta de ftp, y el grupo al cual pertenece ftp, las cuentas restantes del sistema no deben de estar definidas.
- Crear el directorio *pub* dentro de ftp cuyo propietario sea ftp y con permisos de escritura para todos. Los usuarios deberán colocar los archivos que deberán ser accesibles vía la cuenta anonymous .
- Crear el directorio *dev* dentro del directorio de ftp, cuyo propietario sea el superusuario y que no tenga permisos de escritura para nadie. Crear los siguientes archivos especiales dentro de ese directorio:

  | */dev/zero*
  /dev/tcp
  /dev/udp
  /dev/ticotsord

Colocar los permisos de escritura y lectura , de tal manera , que no se tengan permisos de acceso.

- Crear el directorio */usr/share/lib/zone/info* con permisos de lectura y ejecución, cuyo propietario sea el superusuario. Copiar dentro de él el contenido del directorio */usr/share/lib/zoneinfo*. Esto permite a *ls* ó *dir* desplegar la fecha y hora correcta de los archivos.

Un ejemplo de una sesión ftp anónimo se muestra a continuación:

```
$ ftp ftp.super.unam.mx
Connected to mezcal.super.unam.mx.
220 mezcal FTP server (Version wu-2.4(1) Sun Feb 26 13:23:05 CST
1995) ready.
Name (ftp.super.unam.mx:mrocha): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
lost+found
.login
bin
usr
dev
etc
pub
incoming
.ncrecent
info
226 Transfer complete.
72 bytes received in 0.016 seconds (4.5 Kbytes/s)
ftp>
```

6.5 SMTP (Simple Mail Transfer Protocol)

El Protocolo de Transferencia Simple de Correo ofrece una maner rápida y comoda de comunicación. El correo electrónico utiliza una técnica denominada *spooling*. Cuando un usuario envía un mensaje , el sistema coloca una copia de este mensaje en su área de spool privada, junto con la identificación del usuario que envía, el que deberá recibir, la computadora destino y la hora en que fue depositado. El sistema entonces comienza la transferencia de esta información hacia la computadora remota como un proceso en *background*.

El proceso que enviara el mensaje del usuario se convierte en un cliente que deberá mapear el nombre de la máquina destino a una dirección IP, y deberá intentar realizar

una conexión TCP hacia el servidor de correo en la computadora destino. Una vez que el cliente y el servidor estan de acuerdo en que el mensaje ha llegado, el cliente remueve la copia local. Si la conexión falla, o inclusive si no pudo realizarse, el proceso que transfiere, registra la hora en que se intento el envio y termina. Este proceso volvera a revisar el área de *spool* periódicamente siempre que encuentre un mensaje por enviar, o cuando un usuario deposite nueva correspondencia, se intentara de nuevo el envio. Si se detecta que después de un tiempo (usualmente tres dias), el mensaje no pudo llegar a su destino, el mensaje regresara al que envía.

El estándar permite que los identificadores de usuario y de la máquina destino puedan ser diferentes a la clave del usuario y a la dirección de la computadora de ese usuario.

Usualmente los sistemas de correo permiten la utilización de *alias*, es decir, nombres que identifican a un usuario ó grupo de usuarios, los alias pueden utilizarse para mapear muchas claves de usuario en el buzon de uno solo.

Al utilizar SMTP se obtienen todas las ventajas que da TCP/IP, es decir, que los mensajes pueden viajar a través de redes físicas heterógeneas. Sin embargo, SMTP no es el unico estándar para el intercambio de correo electrónico. Cuando se desee intercambiar correo electrónico con sistemas diferentes a SMTP, se deberá utilizar una computadora *gateway* que haga la traducción.

La desventaja de utilizar un gateway es que no se asegura el envio de la información, ya que el cliente borrara su copia local del mensaje en cuanto el primer gateway lo reciba, y no se verificara si llego a su destino final. Sin embargo, la gran importancia que tiene el correo electrónico para muchos personas en su trabajo diario, hace que se utilice este sistema a pesar de sus desventajas. Equipos que no esten conectados directamente a Internet pueden intercambiar correo electrónico con equipos que si lo esten, utilizando uno de esos "gateways".

El estándar para el intercambio de correo electrónico comprende dos partes. La definición del formato del mensaje, y la definición del mecanismo para intercambiar estos mensajes. De esta manera, un sistema de correo propietario puede utilizar el formato de mensaje de SMTP, aunque utilice mecanismos de envio distintos, esto facilita la labor de un gateway.

El estándar que define el formato, especifica que cada mensaje tenga un *encabezado* y un *cuerpo*. El encabezado se encuentra en formato legible y se encuentra dividido en líneas, cada una con una llave, seguida de dos puntos, y después un valor. Algunas llaves son obligatorias, otras son opcionales. Por ejemplo, el encabezado deberá contar con una línea que especifique el destino, esto se indica con una línea que comienza con *To:*. El remitente se indica con una línea que comienza con *From:*. Opcionalmente el remitente puede especificar una dirección a la cual se deberán de enviar las respuestas, esto se indica con una línea que comienza con *Reply-to:*.

Un aspecto complicado del sistema de correo electrónico es la definición de direcciones. Si bien el mecanismo es sencillo cuando se habla de transporte de correo entre computadoras que utilizan el estándar:

usuario@máquina

Cuando se deben de utilizar gateways, el problema se complica, pudiendo resultar direcciones como la siguiente:

usuario%máquina@gateway

La interpretación de dirección es que el mensaje deberá enviarse al gateway y este interpretara la parte restante (usuario%máquina). Sin embargo, si el gateway al cual llego el mensaje no tiene el caracter % como separador entre usuario y máquina, puede existir un problema.

En pocas palabras, el formato de dirección para aquellos mensajes que deberán enviarse entre sistemas de correo no esta estándarizado.

Ahora veremos la parte del estándar de SMTP que corresponde a la manera de intercambiar mensajes.

Para poder controlar el intercambio de mensajes, el cliente y el servidor intercambian una serie de comandos, que estan en forma legible. En primer lugar, el cliente estable una conexión confiable con el servidor, y espera que este le envie un mensaje: `220 READY FOR MAIL`. Cuando el cliente recibe este mensaje, envía al servidor el mensaje `HELO`. El fin de la línea marca el fin del comando. El servidor responde con su identificación. Una vez que se ha establecido la comunicación, el transmisor puede enviar uno o más mensajes, terminar la conexión, o intercambiar papeles con el servidor, de tal manera que el correo pueda fluir en sentido inverso. El receptor debe de dar acuse de

recibo de cada mensaje, tambien puede abortar la conexión o abortar la transferencia del mensaje actual.

Las transaciones de correo comienzan con el comando `MAIL FROM` que da la identificación de donde se deberán de reportar los errores. El receptor prepara sus estructuras de datos para recibir el correo y contestar al comando `MAIL` mandando la respuesta `250 Ok`, queriendo esto decir que se atendera al correo que llegue.

Después de esto, el transmisor envía una serie de comandos `RCPT` para identificar que usuarios deberán recibir un mensaje. El receptor deberá de dar acuses de recibo a cada `RCPT` mandando un `250 OK` o mandando un mensaje de `550 No such user here.`

Después de todos los comados `RCPT`, el transmisor envía el comando `DATA`. El receptor responde con el mensaje `354 Start mail input`, y especifica la secuencia de caracteres que se utilizara para terminar el mensaje.

Una vez que un cliente termina el envio de un mensaje, el cliente puede enviar el comando `TURN` para intercambiar papeles. Si lo hace, el receptor responde con el comando `250 OK` y asume el control de la linea. El lado que tiene el control de la sesión puede terminar la asociación a través del comando `QUIT`, el otro lado deberá responder con un comando `221`. Ambos lados deberán terminar la conexión TCP.

Un ejemplo de como funciona SMTP se muestra a continuación:

```
220 fw.reduno.com.mx Generic SMTP handler
>>> HELO reduno
250 fw.reduno.com.mx talking to reduno.reduno.com.mx ([192.100.183.178])
>>> MAIL From:<mrocha@reduno.com.mx>
250 <mrocha@reduno.com.mx>... Sender ok
>>> RCPT To:<becmrp@tzetzal.dcaa.unam.mx>
250 <becmrp@tzetzal.dcaa.unam.mx>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 QAA29216 Message accepted for delivery
>>> QUIT
221 fw.reduno.com.mx closing connection
becmrp@tzetzal.dcaa.unam.mx... Sent (QAA29216 Message accepted for delivery)
```

Cabe señalar que todo el proceso anterior es transparente para el usuario, los programas de aplicación proveen una interfaz intuitiva para el envío de correos a través

de SMTP, ejemplos de clientes tenemos a ***mail, mailx, pine***, bajo UNIX, y para PC tenemos los clientes, como son ***Eudora, MS-Exchange, MailOnnet***, entre otros.

La configuración de un servidor de correo varía dependiendo del sistema operativo UNIX que usemos, ya sea basado en BSD o System V, el archivo de configuración bajo Solaris 2.x se denomina *sendmail.cf* , y se encuentra bajo el directorio */etc/mail*, el daemon es conocido com *sendmail*.

6.6 NFS (Network File System)

El Sistema de Archivos de Red (*NFS*) es un sistema que provee conexiones transparentes entre redes de computadoras, permitiendo a los directorios y archivos de computadoras remotas parecer como si fueran locales. De todas las aplicaciones de TCP/IP, su interface transparente, hace fácil su uso, una vez configurado correctamente.

Al igual que TCP/IP, el término NFS engloba a una familia completa de productos, aunque *Sun Microsystems Inc.*, los desarrolladores de NFS, lo prefieren llamar *Open Network Computing (ONC)*. Sun Microsystems libero las especificaciones de algunos componentes de NFS al dominio público, como RFCs y se han convertido en una parte integral de la suite protocolos TCP/IP.

Un número de desarrolladores de software han producido versiones de NFS, los cuales operan en sus sistemas, asi, que NFS esta ahora disponible sobre muchas arquitecturas de computadoras, desde PCs hasta grandes *mainframes*. En esencia, NFS provee conexiones virtuales de discos entre computadoras. Un sistema de archivos remoto puede aparecer como una extensión al sistema de archivos local. Asi, muchas personas pueden accesar el mismo archivo y compartir datos.

Asi, NFS provee un sistema de red compartido con características similares a las de Novell Netware y Grupos de Trabajo de Microsoft.

6.6.1 Arquitectura NFS

NFS esta usualmente implementado sobre la suite de protocolos TCP/IP, pero no es exclusivo de esta suite. La arquitectura es mostrada en la figura 6.3.

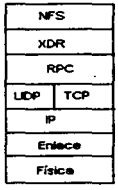


Figura 6.3. Arquitectura de NFS.

Hay tres capas: *RPC (Remote Procedure Calls)*, la cual define el formato de los mensajes usados por las llamadas de procedimientos remotos; *XDR (eXternal Data Representation)*, la Representación de Datos Externa es una representación consistente de datos entre máquinas de diferentes arquitecturas; y NFS, el Sistema de Archivos de Red, es una interface de la capa de aplicación para el acceso, transferencia, y administración de archivos.

La operación de NFS, esta basada en una *relación sin estado(stateless)* entre un cliente y un servidor. Un sistema sin estado (*stateless*) nunca declara que hay relación fija entre las dos entidades de comúnicación; ellas simplemente pasan transacciones entre ellas. Un protocolo de datagramas es apropiado para tal sistema, asi que NFS emplea UDP en lugar de TCP, aunque la versión actual de NFS, que es la 3, soportan también TCP. Otros componentes dentro de la familia de NFS pueden usar TCP o UDP.

Siendo una conexión sin estado, permite a los clientes manejar las fallas de los servidores fácilmente. Solamente el cliente necesita tener conocimiento de que esta tratando de hacer y donde, en la secuencia de eventos que él ha disparado. Porque no hay conexiones que cerrar, el cliente simplemente reintenta las peticiones mientras espera por una respuesta. Una vez que el servidor esta en operación de nuevo, el cliente llevará el servicio hasta donde ellos estaban, antes de caerse el servicio. En un sistema

orientado al estado, si un servidor falla, las conexiones a él fallan, y tienen que ser reestablecidas cuando los servidores se recuperan.

Como NFS, usa frecuentemente UDP y retrasmisiones fijas de tiempos fuera, permite un mejor desmpeño sobre LANs con un bajo retardo y donde el número de datagramas perdidos es bajo, asi NFS, puede no operar satisfactoriamente sobre rutas WAN,aunque bajo TCP, este riesgo se disminuye.

6.6.2 Llamadas de Procedimientos Remotos (RPC).

La tecnología de llamadas de procedimientos remotos(*Remote Procedure Calls; RCP*) es una forma de realizar computo distribuido, ya que permite que el poder de procesamiento de todas las computadoras de una red sea incrementado.

Una buena práctica de programación requiere que un "buen" programa este estructurado con funciones y procemientos propios; cada procedimiento maneja una tarea especifica, tal como "despliega este mensaje"; "obten algún dato" o "calcula esta ecuación". Frecuentemente tales procedimientos son comunes a muchos programas; si están compartidos sobre una red, pueden ser útiles para aplicaciones diferentes. Ademas, la máquina local puede no ser la mas apropiada para ejecutar ciertos procedimientos. Puede haber otras máquinas sobre la red optimizadas para cierta tarea. Los sistemas de computo distribuido permiten al mejor procesador ser identificado y usado para cierta tarea.

Muchos procedimientos son mejor ejecutados por servidores en la red, que ejecutarlos sobre los clientes. El principio es: distribuir las funciones donde puedan ser ejecuatadas mas eficientemente y efectivamente. Usando la tecnología de RPC, la operación de la totalidad del sistema puede ser racionalizada; en lugar de pasar grandes cantidades de datos alrededor de la red, solamente pasando el resultado requerido.

RPC define el formato de la cabecera usada para ejecutar procedimientos sobre un servidor de computo, la cual esta definida en el formato XDR.

6.6.3 Representación de Datos Externa(XDR).

Unos de los problemas presentados para proveer acceso trasparente de red en un ambiente de computo heterógeneo, es el concerniente con la representación de los datos. Los diferentes Sistemas Operativos y Hardware tienen diferentes formas de representar la informacion, como son los valores numéricos, por ejemplo ¿Cómo son representados los números positivos y negativos? , y ¿Cuál bit es el mas significado?, el primero o el último? Un número negativo puede ser representado con un bit de signo o en complemento a dos. Una computadora que usa complemento a dos mal interpretara un valor enviado desde otra computadora que emplea la representación de bit de signo. Cada tipo de valor numérico o de caracter necesita ser definido en una forma común, para asegurarse que cuando lleguen a la máquina remota, puedan ser pasados a la aplicación en una forma que retenga el significado original.

XDR define datos en multiplos de cuatro bytes (32 bits). Si el valor a ser representado no concuerda exactamente dentro de un múltiplo de cuatro bytes, el valor es rellenado con ceros.

XDR es una definición, de como los datos deberán ser formateados antes de ser transportados sobre una red. Un cliente NFS formatea los datos basados en XDR asi que un servidor NFS deberá transladar los datos en una forma que el sistema operativo de la computadora pueda entenderlos.

6.6.4 Los Servicios de NFS.

NFS define la capa superior de la arquitectura ONC, la cual es una interface de la capa de aplicación. NFS mapea los recursos de los sistemas de archivos remotos a través de la red, así, que ellos aparecen como un sistema de archivos local. Las transacciones de red estan completamente ocultas a cualquier aplicación sobre esa computadora.

Los servicios provistos por esta capa de aplicación son acceso y administración a los sistemas de archivos , los cuales son mostrados en la siguiente tabla, como una consecuencia de un sistema sin estado, NFS, no abre archivos; solamente realiza lecturas y escrituras para accesar áreas de un archivo:

Procedimiento NFS	Número de Procedimiento	Descripción
Null	0	No trabaja, solamente regresa una respuesta
Getattr	1	Obtiene atributos de archivos
Setattr	2	establece atributos de archivos
Getroot	3	Obsoleto
Loookup	4	Obtiene el manejador de archivo y atributos
Readlink	5	Lee de liga simbolica
Read	6	Lee un número definido de bytes de un archivo
Cache	7	escribe a cache
Write	8	Escribe un número definido de bytes a un archivo
Create	9	Crea un archivo
Delete	10	Borra un archivo
Rename	11	Renombra un archivo
Mkdir	14	Crea un procedimiento
Rmdir	15	Remueve un directorio
Readdir	16	Obtiene un listado del directorio
Statfsres	17	Obtiene los atributos del sistema de archivos

6.6.5 Portmapper

Cada programa sobre un servidor necesita un número de puerto diferente UDP (o TCP). ONC comprende de un conjunto de programas que requieren varios puertos diferentes.

Los programas ONC emplean un proceso conocido como *portmapper.* Cuando se inicia un proceso de RPC, obtiene un puerto libre, y lo registra con el proceso portmapper. Cuando un programa desea accesar un procedimiento sobre otra computadora, emplea el procedimiento de portmapper sobre esa máquina. Portmapper es un proceso RPC que esta siempre sobre el puerto bien conocido 111. Empleando la llamada al procedimiento portmapper, una aplicación puede encontrar el puerto sobre el cual un número de programa ha sido registrado. El portmapper responde al origen, dando el número de

puerto, así el cliente esta capacitado para enviar directamente al puerto de dicho procedimiento. Esto puede ser comparado con el proceso de ARP, pero lo anterior está relacionado con números de puerto, en lugar de direcciones MAC.

6.6.6 Mountd

mountd es el daemon que permite "montar"o hacer disponibles sistemas de archivos remotos sobre la computadora local en UNIX. mountd es el progrma servidor para peticiones remotas de montaje. Como NFS, está diseñado para proveer un servicio transparente al usuario, el comando *mount* es usado para establecer conexiones con un sistema remoto.

Como NFS es un sistema sin estado, no crea una conexión entre el cliente y el servidor cuando monta un sistema de archivos. Para establecer una relación entre el cliente y una área particular del disco sobre el servidor, mount debe tener una referencia para el directorio apropiado; esta referencia al directorio es usualmente provista por un valor numérico conocido como manejador de archivo (*file handler*). Cuando un sistema remoto es montado a través del comando mount, el cliente emplea una petición RPC para accesar un sistema de archivos particular, al cual el servidor (mountd) retorna un apropiado manejador de archivo.

Aunque NFS emplea un conexión sin estado, mountd registra cuales clientes han hecho peticiones de montaje de sistemas de archivos. Esto ayuda a la administración de la red, como por ejemplo, cuando hay una necesidad de cerrar el sistema o realizar algunos cambios. Pero el registro de los clientes es complicado por la naturaleza sin estado del sistema.

En la figura 6.4, el cliente emplea mount para accesar el sistema de archivos sobre el host remoto. mountd retorna el manejador de archivo (92) para el sistema de archivos apropiado, asi NFS puede emplear ahora esa referencia para accesar ese disco o directorio.

Bajo el sistema operativo Solaris 2.x, para otorgar permisos de acceso a través de NFS a un sistema particular de archivos, es realizado a través del comando *share*, un ejemplo se muestra a continuación, cabe mencionar que solo el superusuario del sistema puede emplear este comando:

```
# share ※F nfs /opt
```

Cabe mencionar, que este comando incluye opciones, para definir las máquinas cliente, los permisos de acceso a ese sistema de archivos (escritura, lectura ,etc), como una medida de seguridad para el servidor de NFS. Sin opciones el comando share, nos muestra , los sistemas de archivos que estamos compartiendo por NFS, ejemplo:

```
#share
-               /Novell/conductor97    rw    ""
-               /Novell/communication_A    rw    ""
-               /Novell/vanguard    rw    ""
```

Si que quieren compartir los sistemas de archivos, cada vez que se inicialize la máquina, se debe de editar el archivo */etc/dfs/dfstab*, en el cual se le agregan las entradas correspondientes, de los sistemas de archivos que se desean compartir por NFS, empleando el comando share. Para descompartir sistemas de archivos por NFS, se emplea, el comando *unshare*.

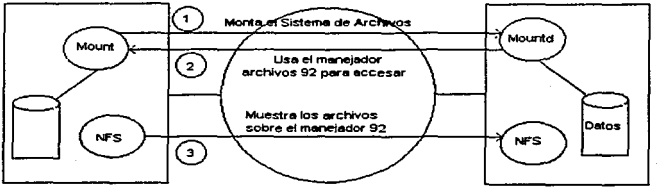


Figura 6.4. El proceso mount.

El comando *showmount*, nos muestra cuales son los clientes que estan usando nuestros sistemas de archivos compartidos por NFS, ejemplo:

```
conbd% /usr/sbin/showmount
106-101.reduno.com.mx
106-219.reduno.com.mx
106-227.reduno.com.mx
152-52.reduno.com.mx
155-146.reduno.com.mx
155-148.reduno.com.mx
```

Los daemons que deben estar corriendo en el servidor UNIX, para permitir el acceso a través de NFS son:

- `lockd`. El daemon *lockd* registra el acceso a los archivos, previniendo de que pudieran a ser modificados por varios usuarios simultáneamente.
- `Statd`. El programa *statd*, es un monitor del estado de la red, e interactúa con lockd para realizar funciones de recuperación de caídas de los servicios de lockd sobre NFS
- `Nfsd`. El programa *nfsd*, es el daemon que maneja las peticiones de sistemas de archivos de los clientes, solamente el superusuario, puede correr este daemon.
- `Mountd`. Es un servidor de RCP que contesta todas las peticiones de montaje de sistemas de archivos.
- `rpc.pcnfsd`. Es el servidor de RCP, que contesta todas las peticiones de NFS, realizadas por usuarios de PCs.

Para accesar un sistema de archivos remoto, por un cliente, se emplea el comando *mount*, el cual permite agregar en la estructura de directorios local, el sistema de archivos remoto, ejemplo:

```
#mount ※F nfs 200.4.139.167:/Novell/publico /opt/Novell/publico
```

La primera parte de la instrucción, nos indica que vamos a montar un sistema de archivos remoto, disponible a través de NFS, indicado por la bandera –F, después sigue el nombre ó dirección IP de la máquina que nos va a compartir el recurso, seguida por el nombre del recurso compartido, y por último, la ruta en el sistema local, donde va a estar disponible dicho recurso para nuestros usuarios locales,. Después de esto, el sistema de archivos remotos es accesado de manera transparente, como cualquier otro directorio o archivo del sistema local, a continuación se muestra una máquina con varios sistemas de archivos montados a través de NFS:

```
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t3d0s0      28669   11819   13990    46%    /
/dev/dsk/c0t3d0s4     288391  188042   71519    73%    /usr
/proc                      0       0       0     0%    /proc
fd                         0       0       0     0%    /dev/fd
/dev/dsk/c0t3d0s3      76894   24682   44532    36%    /var
/dev/dsk/c0t3d0s6      28669   10436   15373    41%    /home
```

```
Filesystem              kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t3d0s5       249937  205447   19500      92%  /opt
swap                    159376      56  159320       1%  /tmp
conbd:/webs             480912  400760   32056      93%  /usr/local
apps:/usr/Users/sanborns/htdocs/WebObjects/media
                        514640  418368   44616      91%  /opt/media
```

Así, si nos quisieras cambiar al directorio */usr/local*, en realidad estaríamos accesando al directorio */webs*, de la máquina denominada *conbd*, y los accesos al recurso /usr/local, los realizaríamos, con los comandos usuales del sistema local.

Para desmontar un recurso, se emplea el comando *umount*, cabe señalar que para desmontar un recurso, éste no tiene que estar siendo usado en el momento de llevar a cabo la operación, ejemplo:

```
#umount /usr/local
```

6.6.7 PC-NFS

Empleando NFS en el ambiente de PCs tiene un problema particular: la validación del usuario. El usuario de una PC es raramente autentificado; cualquiera usando una PC tiene acceso a todos los programas sobre ella. Si los programas en turno dan acceso a servicios sobre hosts en red, entonces pueden comprometer la seguridad de esos hosts.

Los usuarios sobre mini y mainframes son usualmente validados para el acceso a esas computadoras. Se proporcionan paswords y quizas números de cuenta que establecen sus derechos para accesar ciertas areas del sistema. los derechos son manejados a través de un número de identificación de usuario (*UID*) y número de identificación de grupo (*GID*). Cuando los usuarios intentan usar un directorio o un archivo, sus UIDs y GIDs son revisados para verificar que tienen derechos de acceso. Si usuarios intentan accesar un sistema de archivos remoto, sus credenciales sobre el sistema original, esto es el UID y GID, pueden ser vetados para accesar al otro.

Para usuarios de PC usando NFS, el servicio es provisto por un programa llamado *pcnfsd*, el cual es un servicio basado en RPC. Este proceso corre sobre un servidor y permite al sistema administrar y definir derechos de acceso para los usuarios de PC.Este programa pregunta por una identificación de usuario y un password a los usuarios de PCs

para permitir el acceso a los recursos de red provistos por NFS. Cuando una PC realiza una petición de conexión, pcnfsd retorna un UID y GID para ese usuario, que debe usar cuando hace una petición de acceso a sistemas de archivos remotos. Esto permite a un usuario de una PC ser validado por un host como cualquier otro usuario.

Así, el uso de pcnfsd es crítico para la operación de PCs usando NFS, ya que valida al usuario sobre esa PC y controla su acceso a los demas servidores dentro de una red.

Cabe mencionar que hay muchos programas basados en Windows, que nos permiten el acceso a un recurso compartido por NFS, de una manera transparente, a como estamos acostumbrados a trabajar en dicha plataforma.

6.7 DNS

Si bien el esquema de direcciónamiento de TCP/IP permite la identificación única de una computadora dentro de una red compuesta de muchas redes físicas diferentes (como por ejemplo, la Internet), el ser humano maneja mas fácilmente nombres que pueda asociar con algo. Es mas fácil recordar el nombre `reduno.reduno.com.mx` que la dirección `192.100.183.178`.

Ahora bien, si el software de TCP/IP debe de trabajar con direcciones, es evidente que debera haber una transformación entre el nombre y la dirección en algun momento. El *Domain Name System* (Sistema de Nombres de Dominio) es un sistema que analizaremos a continuación:

Al comienzo de la Internet, es esquema de direcciones era plano, existía una administración central que se encargaba de determinar si el nombre era adecuado (es decir que no causará conflictos con otros nombres ya asignados). Pronto esta estructura no pudo mantenerse debido al inmenso crecimiento de la red. Entonces se decidió adoptar un esquema jerárquico de direcciones. Entonces se decidio adoptar un esquema jerárquico de direcciones en el que se delega autoridad para la asignación de nombres. La administración central de la Internet solo es responsable de especificar las partes de "mas alto nivel" del nombre, si por ejemplo, tenemos las siguientes direcciones: `pc47.ibm.com` y `spectrum.ieee.org`. Aqui la administración central fue la encargada de asignar los nombres `ibm.com` y `ieee.org` y dentro de estas

orgaqnizaciones existe una administración encargada de especificar la parte de más "bajo nivel" de la dirección.

Por ejemplo, en la UNAM existe una administración central que determina nombres como:

```
dcaa.unam.mx
politicas.unam.mx
fciencias.unam.mx
```

Si embargo, dentro de cada una de estas entidades, existe una administración de nombres de tal manera que aseguren que son únicos dentro de su "dominio". Por ejemplo:

```
tzetzal.dcaa.unam.mx
mitla.dcaa.unam.mx
```

Corresponden a direcciones de máquinas cuya parte del nombre de "bajo nivel" tuvo que ser escogida de tal manera que fuera unica dentro del dominio "dcaa.unam.mx". Si por ejemplo existiera una máquina llamada:

```
mitla.fciencias.unam.mx
```

no habria ningun problema.

Cabe notar aqui que la estructura que determina la forma del nombre de una computadora no es la estructura física de la red, sino la estructura administrativa dentro de la institución que sea dueña de esta red. Por ejemplo, en el mismo segmento de red fisico coexisten las computadoras de la DCAA y de la DGAE, sin embargo, las computadoras de la DGAE tienen el dominio `dgae.unam.mx`, mientras que las de la dcaa, `dcaa.unam.mx`.

Las partes de "alto nivel" son administrados por el NIC central de la Internet. Existen dos tipos de asignación de nombres, por organización y por area geográfica. Las computadoras pueden dividirse segun el tipo de organización de acuerdo a los siguientes tipos:

Nombre de Dominio	Explicación
COM	Organizaciones comerciales
EDU	Organizaciones educativas
GOV	Organizaciones gubernamentales
MIL	Grupos militares
NET	Centros de acceso a red
ORG	Otro tipo de organizaciones
INT	Organismos internacionales

La administración central podra entonces definir nombres como `ibm.com`, y la administración interna en esas instituciónes sera la encargada de definir los nombres de sus computadoras, o bien, de definir jerarquias internas como en el caso de la UNAM.

Existe otro tipo de jerarquias, que divide los dominios en areas geográficas. En la siguiente tabal se muestra un ejemplo de estructura geográfica de nombres:

Código	País
AU	Australia
BE	Bélgica
ES	España
JP	Japón
MX	México
US	Estados Unidos

Y asi cada país puede definir la estructura interna que tendrá su dominio, por ejemplo, el dominio `us` también se divide en areas geográficas por estado, México ha escogido la division por tipo de institución, asi por ejemplo el dominio `com.mx`, (reduno.reduno.com.mx).

Se puede entender fácilmente la forma en que la resolución de nombres se lleva a cabo, si pensamos que cada subdominio cuenta con un servidor encargado de la traducción de nombres a direcciones dentro de su área de influencia. Sin embargo, es común que los servidores de nombres solo tengan información sobre direcciones IP, sino

que la estructura de DNS se utiliza también para la resolución de por ejemplo direcciones de correo. El cliente, entonces no solo deberá mandar su petición al servidor, sino que también debera indicar de que tipo de información es la que esta proporcionando y que información espera a cambio, por ejemplo, mando nombres espero direcciones.

Como el DNS es una computadora mas dentro de la red la que ejecuta el software de resolución de nombres, obviamente también tiene una dirección que los clientes deberán conocer para poder encontrarla. No existe un estándar para especificar la manera en que los clientes conoceran una computadora que pueda resolver los nombres que requieran, pero generalmente esta información se configura en algun archivo de red, para Solaris 2.x, se define los servidores de nombres en el archivo `/etc/resolv.conf`, un ejemplo de este archivo se muestra a continuación:

```
domain reduno.com.mx
nameserver          192.100.183.178
```

En el cual se el especifica el dominio en el cual se encuentra el cliente, así como la dirección IP del servidor de DNS, al cual se le deberán enviar las peticiones cuando se requiera conocer la dirección IP de una máquina a partir de su nombre.

El cliente tiene dos maneras de especificar al servidor el tipo de resolución. En la primera o *recursiva*, el servidor debera checar si la petición puede ser resuelta por el mismo, pero si no es asi, el servidor debera comunicarse con un servidor de mas alto nivel que pueda conocer la respuesta, esperar la contestacion y enviarla al cliente. En la segunda manera o *iterativa*, en caso de que el servidor no pueda resolver la petición, debera unicamente indicarle al cliente que servidor lo podra hacer.

Como se nota aqui la resolución de nombres no es de "arriba hacia abajo" sino de "abajo hacia arriba". Los servidores usualmente mantienen un cache para mejorar los resultados.

El Sistema de Nombres de Dominio (DNS) es una base de datos distribuida. Esto permite el control de los segmentos de la totalidad de la base de datos, y asi cada segmento esta disponible a través de la red a través de un esquema cliente/servidor.

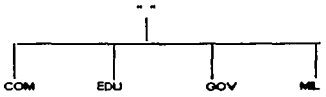


Figura 6.5. Estructura de la base de datos DNS.

Programas llamados servidores de nombres *(name servers)* comprenden la parte del servidor en el mecanismo cliente/servidor del DNS. Los Servidores de Nombres contienen información acerca de algunos segmentos de la base de datos y la hacen disponible a los clientes llamados, *resolvers*. Los resolvers son frecuententemente solo rutinas que crean peticiones y los envian a través de la red a un servidor de nombres.

La estructura de una base de datos DNS, se muestra en la figura 6.5, es muy similar a la estructura de un sistema de archivos UNIX. La totalidad de la base de datos es mostrada como un arbol invertido con la raiz, en la parte superior. El nombre de la raiz del DNS tiene la etiqueta nula (""). Cada nodo en el arbol representa una particion de la totalidad de la base de datos o un *dominio* en el Sistema de Nombres de Dominio. Cada dominio puede ser divido en particiones llamadas *subdominios*, como subdirectorios en los sistemas de archivos. Los subdominios son dibujados como hijos de sus nodos padres, como se muestra en la figura 6.6.

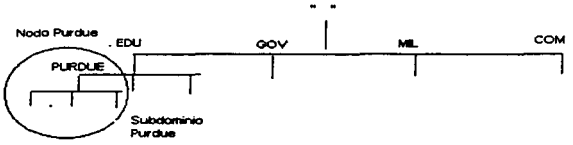


Figura 6.6. Subdominios en la base de datos DNS.

El conjunto de dominios que un servidor de nombres controla es referido como una *zona*. Una zona es un grupo de servidores de nombres conectados dentro de la jerarquía.

Hay cuatro tipos de servidores de nombres: *primario, secundario, cache y esclavo.*

- El servidor primario tiene la base de datos maestra de nombres y direcciones IP para su dominio. El tiene autoridad sobre el mapeo en su base de datos. Este habilita la base de datos cuando se inicia por vez primera.
- Un servidor Secundario tiene la misma base de datos que el servidor primario y actúa como un respaldo para el servidor primario.
- Un servidor de Cache no tiene una base de datos para el mapeo entre direcciones IP y nombres, pero conoce los servidores primarios y secundarios que pueden hacerlo, entonces puede suministrar tal información si es requerida. En grandes organizaciones, este servidor es usado para tomar el sobre procesamiento de los servidores primarios y secundarios.
- Un servidor esclavo opera en una forma similar al servidor de cache.

6.7.1 Configuración de Servidores de Nombres.

Un servidor de nombres de dominio es una aplicación sobre una computadora corriendo TCP/IP. El DNS no tiene que correr sobre una máquina dedicada, por ejemplo el servidor de nombres de la organización `reduno.com.mx`, trabaja sobre una máquina,

que a su vez es también servidor de correo. El procesamiento del DNS no es demasiado alto, solamente en grandes organizaciones, como es el caso de la UNAM, en la cual se requiere de servidores dedicados; un servidor de nombres de dominio puede operar como una tarea en background sobre un sistema operativo multitarea como UNIX. El DNS es suministrado con UNIX, con el daemon de nombre `in.named` (para Solaris 2.x) o `named` sobre sistemas trabajando en una implementación BSD.

Las decisiones hechas cuando es establecido un servicio de este tipo son más organizaciónales que técnicas. Las decisiones mas importantes para este servicio son la estructura organizacional apropiada y los nombres que deberán tener los hosts.

Diferentes implementaciones de DNS emplean diferentes archivos de configuración pero son similares a aquellos usados por el Servidor Internet de Nombres de Dominio de Berkeley, comúnmente conocido como BIND. BIND es distribuido con el UNIX BSD.

Generalmente, hay cinco archivos que deberán ser configurados:

- El archivo de boot.
- El archivo de base de datos de los hosts.
- El archivo de cache.
- El archivo de base de datos invertido.
- El archivo de loopback.

6.7.1.1 El archivo de boot

Este archivo le indica al servidor donde encontrar y que hacer, con los diferentes archivos de bases de datos que necesita para su operacion. Hay normalmente cuatro archivos diferentes archivos de bases de datos que un servidor puede usar. En la figura 6.7, se muestra un ejemplo de un archivo *boot*. Los comentarios se indican con ";", para hacer el archivo mas entendible.

```
; Indica en d'onde est'an los zone file
;
directory       /var/named
;
; En este archivo est'an los servidores dns ra'iz
;
cache    .      named.ca
;
; Los siguientes son los zone files, contienen los hosts que pertenecen a
dicho
; dominio.
;
primary         cablevision.net.mx              named.cable
primary         caliope.com.mx                  named.caliope
primary         condumex.com.mx                 named.condumex
primary         euzkadi.com.mx                  named.euzkadi
primary         generaltire.com.mx              named.generaltire
;
; Los siguientes archivos contienen el dominio inverso
;
primary         177.23.200.in-addr.arpa         named.rev200.23.177
primary         146.33.200.in-addr.arpa         named.rev200.33.146
primary         144.4.200.in-addr.arpa          named.rev200.4.144
primary         152.4.200.in-addr.arpa          named.rev200.4.152
primary         159.4.200.in-addr.arpa          named.rev200.4.159
```

Figura 6.7. Archivo `/etc/named.boot` bajo Solaris 2.x

6.7.1.2 Los archivos de base de datos.

El archivo principal es la base de datos de los hosts en el dominio sobre el cual este servidor tiene autoridad. Este es el mapeo de nombres a direcciones IP de nodos dentro del dominio de este servidor.

Este archivo contiene los siguientes campos:

- *Name*. El nombre de la máquina que esta contenida en este registro. Si este campo es "@", es un abreviación para el dominio actual para este servidor de nombres de dominio.
- *ttl* (time to Live). El tiempo, en segundos, después del cual la información debe de ser actualizada para este registro si el registro fue copiado a otro DNS. No usado por este DNS.
- *address class*. La clase de dirección de esta entrada, *IN*. Un registro para una máquina la cual es compatible con Internet.

- *entry type*. Tipo de entrada, sus posibles valores son: *A, NS, SOA, MX, HINFO, WKS, CNAME*. En este primer registro, la entrada es de tipo *SOA* (Start Of Authority), la cual indica que sigue una descripcion completa del conjunto de registros del DNS en el cual el DNS tiene autoridad.
- *server*. El contenido depende del tipo de entrada (entry type). Esta puede ser una dirección IP o un nombre de dominio.

El registo *Start Of Authority* contiene campos adicionales que describen la validez de la información del DNS. Dentro de los parentésis hay valores describiendo la validez de la información:

- *serial*. Un número de 16 bits, el cual empieza en 1 e indica la version de la información de la base de datos.
- *refresh*. Un número de 32 bits, el cual es el intervalo de tiempo en el cual todos aquellos servidores que copiaron la base de datos de este servidor deberán actualizarla.
- *retry*. Un número de 32 bits, indica el intervalo de tiempo en segundos en la cual deberán reintentar actualizar su bases de datos, en caso de que haya fallado el primer intento.
- *expire* Un número de 32 bits, el cual es el limite superior en tiempo (segundos) en la cual la zona no es la mayor autoridad.
- *minimum*. El tiempo minimo de ttl.

En la siguiente parte del archivo esta la totalidad de la base de datos, mostrando los nombres y las direcciones IP dentro de la autoridad. IN indica que la direcciones es un dirección IP .

Los siguientes son algunos valores válidos para el campo de tipo de entrada:

- *A*. Un registro de dirección IP de un host. Lo que sigue es la dirección IP de ese host nombrado en el campo name.
- *NS*. Una entrada para un servidor de nombres.En un registro SOA es la repetición del nombre nombre de dominio inmediatamente después de SOA.
- *MX* . Un registro de mail Exchanger. Indica que el host actua como un repartidor de correo.

- *HINFO*. Un registro de información de un hosts. Contiene información util acerca del host como tipo de máquina, sistema operativo, y número de serie.
- *WKS*. Registro de servicio bien conocido. Da la dirección, protocolo y servicios provistos por una máquina, por ejemplo:

  132.248.27.10 Telnet FTP
- *CNAME*. Un nombre canónico. Un alias para ese host.

Estos campos extra, proveen información adicional útil para la administración de la base de datos. Todos esos detalles pueden ser obtenidos desde el servidor de nombres de dominio por un host remoto. Información Detallada de esos campos y como configurarlos dependen del programa que estemos usando en particular y de la plataforma en que lo estemos implantando. A continuación se muestra uno de estos archivos de base de datos.

```
    @               in              soa             dns.plazaemall.com.mx.
mrocha.reduno.reduno.com.mx.
                (
                970514001       ; yymm.dd[0-9][0-9]serial
                84000   ;refresh
                3600    ;retry
                604800  ;expire
                172800  ;minimum
                )
                in      ns      dns.plazaemall.com.mx.
    localhost       in      a       127.0.0.1
    apps    in      a       200.4.159.23
    catbert in      a       200.4.152.218
    contenido       cname   webs.plazaemall.com.mx.
    darkside        in      a       200.4.152.214
    ratbert in      a       200.4.152.216
    dino    in      a       200.4.152.211
    dns     in      a       200.4.159.19
```

Figura.6.8. Ejemplo de un archivo de Base de Datos

6.7.1.3 El archivo de cache

El archivo de cache es cargado en memoria del servidor de nombres cuando este es inicializado. Contiene información, la cual debe de estar disponible inmediatamente y normalmente identifica a los servidores los cuales tienen autoridad sobre este servidor de dominio. El, ademas indica las direcciones de los servidores de nombres raíz.

El archivo de cache mostrado en la figura 6.9 muestra los servidores de nombres de dominio para este dominio y la dirección de un servidor de nombres raíz, los cuales deberán ser cargados en la base de datos de cache, asi, el servidor de nombres de dominio puede llamarlos sin tener que buscarlos.

```
;
; Prep the cache (hotwire the addresses).  Order does not matter.
;

A.ROOT-SERVERS.NET.      99999999        IN      A       198.41.0.4
B.ROOT-SERVERS.NET.      99999999        IN      A       128.9.0.107
C.ROOT-SERVERS.NET.      99999999        IN      A       192.33.4.12
D.ROOT-SERVERS.NET.      99999999        IN      A       128.8.10.90
E.ROOT-SERVERS.NET.      99999999        IN      A       192.203.230.10
F.ROOT-SERVERS.NET.      99999999        IN      A       192.5.5.241
G.ROOT-SERVERS.NET.      99999999        IN      A       192.112.36.4
H.ROOT-SERVERS.NET.      99999999        IN      A       128.63.2.53
I.ROOT-SERVERS.NET.      99999999        IN      A       192.36.148.17
```

Figura 6.7. Archivo /var/named/cache.ca, bajo Solaris 2.x.

6.7.1.4 Mapeo Inverso

El segundo archivo de base de datos es el de mapeo inverso. En ciertas circustancias es beneficioso permitir ejecutar mapeos inversos, esto es, encontrar el nombre de dominio de un host a partir de su dirección IP. Este tipo de mecanismo puede ser usado para descubrir las direcciones IP de gateway para una red en particular. Como las direcciones IP no son geográficas en su estructura esto puede ser difícil. Para ayudar a resolver este problema un dominio especial ha sido definido, *IN-ADDR* (INverse ADDress). El dominio es tratado como la raíz de nombres que estan basados en las direcciones IP de los hosts y redes, asi que un nombre puede ser formulado para descubrir hosts sobre una red particular,una base de datos *IN_ADDR.ARPA* se muestra a continuación:

Como la base de datos principal, este archivo contiene nombres y direcciones IP, pero su objetivo es permitir encontrar nombres de dominio a partir de direcciones IP.

```
@                   in                  soa                 dns.plazaemall.com.mx.
mrocha.reduno.reduno.com.mx.
                    (
                    970514001           ;serial
                    84000     ;refresh
                    3600      ;retry
                    604800    ;expire
                    172800    ;minimum
                    )
          in        ns        dns.plazaemall.com.mx.
18        in        ptr       webs.plazaemall.com.mx.
19        in        ptr       dns.plazaemall.com.mx.
;21       in        ptr       www.reduno.com.mx.
22        in        ptr       www.prodigy.com.mx.
23        in        ptr       apps.plazaemall.com.mx.
```

Figura 6.8. Ejemplo de un archivo de mapeo inverso denominado

/var/named/named.rev200.4.159

6.7.1.5 El Archivo de Loopback

El último archivo es un archivo de loopback el cual guarda las direcciones de loopback usadas por el host.

Normalmente para probar el DNS por default , o bien cualquier otro DNS, se emplea el comando *nslookup*, o bien *nstest*, respectivamente, en el cual se le especifica el nombre de una computadora, y el DNS, nos debe de regresar la dirección IP, siempre y cuando este nombre de la computadora haya sido registro en un DNS válido, también un DNS, nos puede devolver información acerca de los servidores de correo de un determinado dominio:

```
$ /usr/sbin/nslookup
Default Server:  uninet.uninet.com.mx
Address:  200.33.137.129

> www.cisco.com
Server:  uninet.uninet.com.mx
Address:  200.33.137.129

Name:    cio-sys.Cisco.COM
Address: 192.31.7.130
Aliases: www.Cisco.COM
```

6.8 WWW

El World Wide Web, ademas conocido WEB o W3, es una arquitectura de información, este término apareció en Marzo de 1989 cuando Tim Berners-Lee del Laboratorio Europeo de Particulas Físicas, conocido como CERN, inicio el proyecto para la transmisión de ideas de investigación de la organización a través de la red.

A finales de 1990 fue introducida la primera pieza de software WEB sobre una máquina NEXT. Este tuvo la capacidad de visualizar *hypertexto* sobre la pantalla, asi como también transmitirlos a través de la Internet.

El Web permite visualizar texto, imagenes en movimiento,imagenes, sonido, con algunas capacidades interactivas, asi como tambien ligas (las cuales son palabras que nos llevan a otro documento), de tal manera que la información es presentada de una manera mas amigable al usuario pudiendo ir a otros documentos a través de ligas.

Las especificaciones son públicas y cualquiera puede seguirlas para construir un cliente o servidor.

La forma en que se comunican los clientes y servidores esta especificada a través del protocolo *HTTP (HyperText Transfer Protocol)*,introducido por primera vez en 1990.

La terminología que se utiliza en el WEB es

- *URL (Uniform Resources Locator)*, identifica al protocolo, servidor y nombre de archivo de un documento web.
- *HTTP (HyperText Transfer Protocol)*, el protocolo que permite a los clientes y servidores comunicarse.
- *HTML (HyperText Markup Language)*, Especifica al formato de la información de los documentos web: texto, imagenes, sonidos, ligas.
- Servidor, es el que hace la información disponible al cliente.
- *BROWSER*. El cliente (Explorer, Netscape, etc) corriendo sobre la máquina local que obtiene la información de los servidores.

6.8.1 Configuracion de un Servidor WEB en UNIX.

Normalmente la configuración de un servidor web, depende del producto, así como también de la plataforma que estemos usando, pero por lo general , todos traen los siguientes archivos de configuración:

- *Httpd.conf.* En este archivo, se configura las características del web referentes al número de puerto, donde va a escuchar el servidor, normalmente es el 80, aunque se le puede configurar cualquier otro siempre y cuando no este siendo usado, también se le configuran el nombre del servidor, número maximo de conexiones simultaneas, archivos de registro de conexiones al servidor, así como otras características operativas del servidor web.
- *Access.conf.* En este archivo se le configuran las derechos de acceso a determinadas páginas y directorios del servidor web.
- *Srm.conf.* En en este archivo, se le especifican la ruta donde puede encontrar los diferentes recursos del servidor web, como pueden ser: directorio de localización de los paginas web en el servidor, directorio de programas CGI (Common Gateway Interface, son programas que permiten la interacción a través de páginas en el Web), el directorio de imagenes, de sonidos, etc.
- *Mime.types.* En este archivo se le configuran los diferentes tipos de archivos que el servidor puede manejar y la aplicación soportada.

Normalmente en UNIX, un servidor web, puede trabajar como daemon, es decir, que siempre este corriendo en espera de conexiones, ó bien puede ser levantado a través del programa *inetd*, cada vez que llegue una petición que tenga que ser atendida por el servidor web.

Se recomienda que el servidor web trabaje como daemon, debido a la gran cantidad de peticiones, que se realizan a través de este servicio, así como también a lo tardado que puede ser el servicio al ser iniciado por inetd.

6.9 CONCLUSIONES.

En esteb capítulo se han mostrado, algunos de los servicios de UNIX, que han trascendido y han sido implantadas en otras plataformas, por su gran utilidad en los servicios de red. Muchas de estas aplicaciones estan revoluciando como en el caso del Worl Wide Web, la forma de acceso y distribución de la información en la Internet, por último cabe mencionar que bajo UNIX, es donde se ha dado el mayor desarrollo de aplicaciones para servicios en red, dado su estrecha relación con la suite de Protocolos TCP/IP, en los cuales está basada la Internet.

VII. PROGRAMACIÓN DE SOCKETS

7.1 INTRODUCCIÓN

Hasta ahora nos hemos concentrado en la discusión de los conceptos y principios que envuelven a los protocolos TCP/IP, sin especificar la interface entre los programas de aplicación y el software del protocolo. En este capitulo se revisaran las interfaces entre los programas de aplicación y el software del protocolo TCP/IP. Primero nosotros debemos distinguir entre la interface y los protocolos TCP/IP, porque los estándares no especifican exactamente como los programas de aplicación interactuan con el software de protocolos. Asi, la arquitectura de la interface no esta estandarizada; su diseño esta fuera del alcance de la suite de protocolos. Segundo, en la práctica, es inapropiado para los protocolos estar ligado a una interface particular, ya que, una simple interface no podra trabajar bien sobre todos los sistemas. En particular, debido a que el software de protocolo reside en los sistemas operativos de las computadoras, los detalles de las interfaces dependen de los sistemas operativos donde se implante,por ejemplo para el UNIX basado en System V emplea la interface conocida como *Transport Layer Interface (TLI)*, mientras que para Windows, se emplean los denominados *Winsocks*, pero todos, están basados en los sockets desarrollados para el UNIX BSD .

7.2 EL PARADIGMA UNIX DE RED DE ENTRADA/SALIDA

Desarrollado a finales de los 60s y a principios de los 70s, UNIX fue originalmente diseñado como un sistema de tiempo compartido para procesos simples. UNIX es un sistema orientado a procesos en los cuales de aplicación se ejecutan como procesos de nivel de usuario. Un programa de aplicación interactua con el sistema operativo haciendo *llamadas al sistema*. Desde el punto de vista de los programadores, las llamadas al sistema se parecen y comportan exactamente como llamadas de procedimientos. Estas toman los argumentos y regresan uno o más resultados. Los argumentos pueden ser valores (por ejemplo, un contador de enteros) o punteros a objetos en el programa de aplicación (por ejemplo, un buffer factible de ser llenado por caracteres).

La primitivas de entrada y salida de UNIX (I/O), son derivadas de Multics y otros sistemas, siguiendo un paradigma algunas veces referido *como apertura-lectura-escritura-cierre* (*open-read-write-close*). Antes de que un usuario pueda ejecutar

operaciones de I/O, llama a *open* para especificar el archivo o dispostivo a ser usado y obtener los permisos correspondientes. La llamada a open regresa un entero pequeño denominado *file descriptor* (El termino "file descriptor" se refiere en UNIX a que todos los dispositivos son mapeados dentro del espacio de nombres del sistema de archivos. En muchos casos, archivos, dispositivos y otras operaciones de I/O son indistinguibles) que el proceso emplea cuando se ejecutan operaciones de I/O sobre la aperura de un archivo o dispositivo. Una vez que un objeto ha sido abierto, el proceso del usuario realiza una o mas llamadas de apertura para leer, escribir ó para transferir datos. *Read* transfiere datos a los procesos del usuario; *write* transfiere datos, de los procesos de los usuarios a un archivo ó dispositivo. Tanto read como write toman tres argumentos que especifican el descriptor de archivo a usar, la dirección de un buffer, y el número de bytes a transferir. Después de que todas las transferencias estan completas, el proceso del usuario llama a *close* para informar al sistema operativo que ha terminado de usar el objeto(El sistema operativo automaticamente cierra todos los objetos abiertos, si un proceso termina sin llamar a close).

7.3 AGREGANDO ENTRADA/SALIDA DE RED A UNIX

Originalmente, los diseñadores de UNIX englobaron todas las operaciones de E/S en el paradigma open-read-write-close descrito anteriormente. El esquema incluía E/S para dispositivos orientados a caracteres como terminales CRT y dispositivos orientados a bloque como discos y archivos de datos. Las primeras implementaciones de TCP/IP bajo UNIX ademas incluyeron el paradigma de open-read-write-close con un nombre de archivo especial, */dev/tcp*.

El grupo que agregó los protocolos al BSD UNIX decidió que debido a que los protocolos de red eran más complejos que los dispositivos convencionales de E/S, la interacción entre los procesos de los usuarios y los protocolos de red deberia de ser mas complejo que las interacciónes entre los procesos de los usuarios y las convencionales facilidades de E/S. En particular, la interface de protocolo debe permitir a los programadores crear servidores que esperan conexiones pasivamente así como también clientes que formen conexiones activamente. Mas complejo, programas de aplicación enviando datagramas que deberían especificar la dirección destino a lo largo de cada

datagrama, en lugar de ligar destinos, en el tiempo que hacen una llamada de open. Para manejar todos esos casos, los diseñadores abandonaron el paradigma tradicional de apertura-lectura-escritura-cierre y agregaron nuevas llamadas al sistema , asi como también nuevas librerias de rutinas. Agregar protocolos de red a UNIX incremento la substancialmente la complejidad de la interfaces de E/S.

7.4 LA ABSTRACCIÓN SOCKET

Las bases de E/S de red del Sistema BSD UNIX 4 , se centra en la abstracción conocida como *socket*. Nosostros pensamos en un socket como una generalización del mecanismo de acceso de archivos UNIX, que provee un punto final para una comunicación. Como con los accesos a archivos, los programas de aplicación, pide al sistema operativo crear un socket cuando se necesita. El sistema regresa un entero que el programa de aplicación emplea para referirse al socket creado. La diferencia principal entre los descriptores de archivos y los descriptores de sockets es que el sistema operativo liga un descriptor de archivo a un dispositivo o archivo especifico, cuando la aplicación llama llama a open, pero puede crear sockets sin ligarlos a una dirección destino especifica. La aplicación puede escoger, ya sea suministar una dirección destino cada vez que usa el socket (por ejemplo, cuando se envian datagramas), o puede escoger ligar la dirección destino al socket y evitar especificar el destino repetidamente (por ejemplo, cuando se realiza una conexión TCP).

Siempre que lo deseen, los socketes se pueden comportar como archivos o dispositivos tradicionales UNIX, asi que ellos pueden ser usados con operaciones tradicionales como read y write. Por ejemplo, una vez que un programa de aplicación crea un socket y crea una conexión TCP del socket a un destino lejano, el programa puede usar write para enviar un flujo de datos a traves de la conexión (el programa de aplicación en la otra parte puede usar read para recibirlos). para hacerlo posible emplea primitivas como read y write con archivos y sockets, el sistema operativo coloca los descriptores de sockets y los descriptores en el mismo conjunto de enteros y verifica que si un entero ha sido colocado como un descriptor de archivo, no deberá ser colocado como un descriptor de socket.

7.5 CREACION DE UN SOCKET.

La llamada al sistema socket crea sockets en base a la demanda. Toma tres enteros de argumentos y regresa un entero como resultado:

```
resultado=socket(af,tipo,protocolo)
```

El argumento *af* especifica la familia del protocolo usado por el socket. Esto es, especifica como interpretar las direcciones cunado son suministradas. Las familias actuales incluyen el protocolo TCP/IP, (AF_INET), PUP de Xerox (AF_PUP), Appletalk Network (AF_APPLETALK), y sistemas de archivos UNIX (AF_UNIX), etc.

El argumento *tipo* especifica el tipo de comunicación deseado. Posibles tipos incluyen servicio de envio confiable de flujo (SOCK_STREAM) y servicio de envio de datagramas no orientado a conexión (SOCK_DGRAM), asi como un tipo crudo (SOCK_RAW) que permita a los programas privilegiados, accesar a los protocolos de bajo nivel o interfaces de red.

Aunque el diseño general de separar los tipos y las familia de protocolos puede parecer suficiente para manejar fácilmente todos los casos, no es asi. Primero, puede ser que una familia de protocolos no soporte uno o más de los posibles tipos de servicios. Por ejemplo, la familia UNIX tiene un mecanismo de comunicación de interprocesos llamado pipe que emplea un servicio de envio confiable, pero no tiene un mecanismo para envio de paquetes secuenciados. Asi, no todas las combinaciones de tipos y familias de protocolos tiene sentido. Segundo, algunas familias de protocolos tienen multiples protocolos que soportan un tipo de servicio. Por ejemplo, puede ser que una simple familia de protocolo tenga dos servicios de envio de datagramas no orientados a conexión. Para acomodar múltiples protocolos dentro de una familia, la llamada al socket tiene un tercer argumento que puede ser usado para seleccionar un protocolo especifico. Para emplear el tercer argumento, el programador debe entender la familia del protocolo lo suficiente para conocer el tipo de servicio que cada protocolo suministra.

7.6 PROCESOS DE SOCKETS

UNIX emplea las llamadas al sistema *exec* y *fork* para comenzar un nuevo programa de aplicación. Es un procedimiento de dos pasos. En el primer paso, fork crea una copia separada del programa de aplicación ejecutandose. En el segundo paso, la

nueva copia reemplaza por si misma por el programa de aplicación deseado . Cuando un programa llama a fork, la copia recien creada hereda el acceso a todos los sockets abiertos asi como también hereda a todos los archivos abiertos. Cuando un programa llama a exec, la nueva aplicación retiene el acceso a todos los sockets abiertos.

Cuando un proceso finaliza, emplea una llamada al socket close. Close tiene la forma:

```
close(socket)
```

Donde el argumento `socket` especifica el descriptor de un socket a cerrar.

A continuacion veremos las instrucciones de programacion de sockets mas comunes:

Inicialmente, un socket es creado sin alguna asociación a alguna dirección local o remota. Una vez que un socket ha sido creado, el servidor emplea la llamada al sistema *bind* para establecer una dirección local para él. Bind tiene la siguiente forma:

```
bind(socket,localaddr,addrlen)
```

El argumento `socket` es el entero descriptor del socket a ser ligado. El argumento `localaddr` es una estructura que especifica la dirección local en la cual el socket deberá ser ligado, el argumento `addrlen` es un entero que especifica la longitud de la dirección medida en bytes.

Inicialmente, un socket es creado en un estado sin conexión, lo cual significa que el socket no esta asociado con alguna dirección remota. El sistema llama a `connect` para ligar un destino a un socket, poniendolo en conexión. Un programa de aplicación debe de llamar a connect para establecer una conexión antes de que pueda transferir datos a traves de un flujo socket confiable. La llamada al sistema connect tiene la forma:

```
connect(socket, destaddr, addrlen)
```

Donde los argumentos tienen el mismo significado que los anteriores.

Una vez que una aplicación ha establecido un socket, puede usar el socket para transmitir datos. Hay varias llamadas al sistema , entre las que tenemos: `send`, `sendto`, `sendmsg`, `write` y `writev`. write toma tres argumentos

```
write(socket, buffer, tama&o)
```

El argumento `socket` contiene el descriptor, `buffer` contiene la dirección de los datos a ser enviados, el agumento tamaño especifica el número de bytes aser enviados.

La llamada al sistema `send` tiene la forma:

```
send(socket, message, lenght, flags)
```

Donde `socket` especifica el socket a usar, `message` especifica la dirección de los datos a 0ser enviados, `lenght` especifica el número de bytes a ser enviados, y el argumento `flags` controla la transmision.

Analogo a las diferentes operaciones de salida, UNIX ofrece llamadas que un proceso puede usar para recibir datos a traves de un socket; `read`, `readv`, `recv`, `recvform`, y `recvmsg`, las diferencias entre ellas son mínimas. Read tiene la forma:

```
read(descriptor, buffer, length)
```

Donde el argumento `descriptor`, da el descriptor de un socket o un descriptor de archivo, desde el cual se puede leer los datos, `buffer` especifica la dirección de memoria en la cual se almcenaran los datos, y `length` especifica el número maximo de bytes a leer.

7.7 ESPECIFICANDO UNA COLA PARA UN SERVIDOR

Una de las opciones que aplica a los sockets, es usada frecuentemente, un llamada al sistema separada ha sido dedicada para eso. Para entender como surge, consideremos un servidor. El servidor crea un socket, lo liga a un puerto de protocolo bien conocido, y espera por peticiones. Si el servidor emplea un envio de flujo confiable, o si una respuesta toma una cantidad no trivial de tiempo, puede ser que una petición llegue antes de que el servidor termine de responder a una vieja peticon. Para evitar que los protocolos rechazen o descarten las peticiones entrantes, un servidor le debe de decir al protocolo de software subyacente, que desea tener tales peticiones encoladas, hasta que tenga el tiempo para procesarlas.

La llamada al sistema *socket*, permite a los sockets preparar un socket para conexiones entrantes. En términos de los protocolos subyacentes, `listen` pone el socket en modo pasivo, listo para aceptar conexiones. Cuando el servidor invoca a listen, informa ademas al sistema operativo, que el software de protocolo deberá encolar peticiones de conexión simultáneas múltiples que llegan al socket. La forma es:

```
listen(socket, qlength)
```

El argumento `socket` da el descriptor de un socket que deberá estar preparado para ser usado por un servidor, el argumento qlength especifica la longitud de la cola para

ese socket. Después de la llamada, el sistema deberá encolar las peticiones hacia `qlength`. Si la cola se llena cuando una petición llega, el sistema deberá rechazar las conexiones descartando la petición. Listen aplica solamente a los sockets que han seleccionado el servicio de envio de flujo confiable.

7.8 COMO UN SERVIDOR ACEPTA CONEXIONES

Como se ha visto, un servidor de procesos emplea las llamadas al sistema socket, bind, y listen paraq crear un socket, ligarlo a un bien conocido puerto, y especificar una cola de longitud para las peticiones de conexión. Hay que notar, que la llamada a bind, asocia al socket con un puerto bien conocido de protocolo, pero el socket no está conectado a un destino remoto especifico. De hecho, el servidor debe de permitir al socket recibir peticiones de conexión desde un cliente arbitrario.

Una vez que el socket ha sido establecido, el servidor necesita esperar por una conexión. Para hacer esto, el emplea la llamada al sistema `accept`. Una llamada a accept bloquea hasta que una conexión llega. Tiene la forma:

```
newsock= accept(socket, addr, addrlen)
```

El argumento `socket` especifica el descriptor de un socket, sobre el cual se va a esperar conexiones. El argumento `addr` es un puntero a una estructura del tipo sockaddr, y `addrlen` es un puntero a un entero. Cuando una petición llega, el sistema llena el argumento addr con la dirección del cliente que ha originado la petición y coloca addrlen a la longitud de la dirección. Finalmente, el sistema crea un nuevo socket que tiene su destino conectado a la dirección del cliente y retorna el nuevo descriptor del socket al que inicio la llamada. El socket original, aún tiene un destino remoto y aún permanece abierto. Así, el servidor maestro puede continuar aceptando conexiones adicionales en el socket original.

Cuando una petición arriba, la llamada a accept regresa, y el servidor puede, manejar las peticiones *iterativamente ó concurrentemente*. En el acceso iterativo, el servidor maneja la petición por si mismo, cierra el nuevo socket, y entonces llama a acept para obtener la siguiente petición de conexión. En un acceso concurrente, después de la llamada para aceptar, el servidor maestro crea un esclavo para manejar la petición (en la terminologia UNIX, el bifurca (forks) un proceso hijo para manejar la conexión). El

proceso esclavo hereda un copia del nuevo socket , asi que puede proceder a atender la petición. Cuando termina, el esclavo cierra la conexión y termina. El servidor de procesos original (maestro) cierra su copia del nuevo socket después de comenzar al esclavo. Entonces llama a accept para obtener la siguiente petición de conexión.

El diseño concurrente para servidores puede parecer confuso porque múltiples procesos deberán estar usando el mismo número de puerto local de protocolo. La clave para entender el mecanismo descansa en la forma en que los protocolos subyacentes tratan a los puertos de protocolo. Recordando que en TCP un par de puntos finales define una conexión. Así, no importa cuantos procesos usan un puerto local de protocolo ni como ellos se conecten a destinos diferentes. En el caso de un servidor concurrente, hay un proceso por cliente y un proceso adicional que acepta conexiones. El socket que el servidor maestro usa, tiene una carta para un destino, permitiendole conectarse con un sitio arbitrario remoto. Cuando un segmento TCP llega, deberá ser enviado al socket conectado al segmento fuente. Si tal socket no existe , el segmento deberá enviar al socket con una carta de destino remoto que no tiene una conexión abierta, y deberá aceptar solamente segmentos TCP que piden una nueva conexión.

7.9 EJEMPLO DE PROGRAMACIÓN DE SOCKETS.

A continucación veremos el ejemplo de un servidor y un cliente TCP, empleando programación de sockets, los programas fueron realizados en el Sistema Operativo Solaris 2.5.1, empleando el compilador de C, *gcc* de GNU, el listado y la explicación se muestran a continuación:

```
#include "sockhelp.h"

/* La sig. Función toma un nombre de servicio, y regresa un número de
puerto. Si
   el nombre del servicio no es encontrado, lo trata como un numero
decimal. El
   numero regresado es ordenado por la red. */

int atoport(service, proto)
char *service;
char *proto;
{
  int port;
  long int lport;
  struct servent *serv;
```

```
  char *errpos;

  /* Primero lo trata de leer de /etc/services */
  serv = getservbyname(service, proto);
  if (serv != NULL)
    port = serv->s_port;
  else { /* No esta en  services, puede ser un numero*/
    lport = strtol(service,&errpos,0);
    if ( (errpos[0] != 0) || (lport < 1) || (lport > 65535) )
      return -1; /* Dirección de puerto invalida  */
    port = htons(lport);
  }
  return port;
}

/* Converts ascii text to in_addr struct.  NULL is returned if the
address
    can not be found. */
struct in_addr *atoaddr(address)
char *address;
{
  struct hostent *host;
  static struct in_addr saddr;

  /* First try it as aaa.bbb.ccc.ddd. */
  saddr.s_addr = inet_addr(address);
  if (saddr.s_addr != -1) {
    return &saddr;
  }
  host = gethostbyname(address);
  if (host != NULL) {
    return (struct in_addr *) *host->h_addr_list;
  }
  return NULL;
}
/* Esta función, escucha sobre un puerto, y regresa conexiones.  Esta
   función deberá crear un nuevo proceso para cada conexion entrante,
   asi, al proceso que esta escuchando, el nunca debera regresar. Esto
   significa que el codigp que lo llama no debera realizar un ciclo.
Los parametros son como sigue:
      Socket_type: SOCK_STREAM o SOCK_DGRAM (sockets TCP o UDP
      Port: El puerto sobre el cual va a escuchar. Hay que recordar que
los
        Puertos < 1024 estan reservados para root. Deben de ser pasados
en el
        Orden de bytes de red.
      Listener: Este es un apuntador a un variable que almacenara el
descriptor
            De archivo (file descriptor)d del socket que esta siendo
      usado para          escuchar. Asi, se puede escribir un manaejador
      de se&al, para cerrar el descriptor, en el evento de terminacion
      del programa. Cabe mencionar que los UNIX modernos pueden cerrar
```

```
      los descriptores de archivos con exit, asi que esto puede no ser
      requerido. */
int get_connection(socket_type, port, listener)
int socket_type;
u_short port;
int *listener;
{
  struct sockaddr_in address;
  int listening_socket;
  int connected_socket = -1;
  int new_process;
  int reuse_addr = 1;

  /* Iniciar la informacion de la direccion internet
     Esto es usado con la llamada bind() */

  memset((char *) &address, 0, sizeof(address));
  address.sin_family = AF_INET;
  address.sin_port = port;
  address.sin_addr.s_addr = htonl(INADDR_ANY);

  listening_socket = socket(AF_INET, socket_type, 0);
  if (listening_socket < 0) {
    perror("socket");
    exit(EXIT_FAILURE);
  }

  if (listener != NULL)
    *listener = listening_socket;

  setsockopt(listening_socket, SOL_SOCKET, SO_REUSEADDR, &reuse_addr,
    sizeof(reuse_addr));

  if (bind(listening_socket, (struct sockaddr *) &address,
    sizeof(address)) < 0) {
    perror("bind");
    close(listening_socket);
    exit(EXIT_FAILURE);
  }

  if (socket_type == SOCK_STREAM) {
    listen(listening_socket, 5); /* Encolando hasta cinco conexiones
antes que
                                    tengar que ser automaticamente negadas
*/
     while(connected_socket < 0) {
      connected_socket = accept(listening_socket, NULL, NULL);
      if (connected_socket < 0) {
        /* Ya sea un error real, o el bloque fue interrumpido por
           alguna razon. Solamente aborta la conexión si un error
ocurrio. */
      if (errno != EINTR) {
```

```
            perror("accept");
            close(listening_socket);
            exit(EXIT_FAILURE);
          } else {
            continue;
        }
        }

        new_process = fork();
        if (new_process < 0) {
          perror("fork");
          close(connected_socket);
          connected_socket = -1;
        }
        else { /* Se tiene un nuevo proceso... */
          if (new_process == 0) {
            /* Este es un nuevo proceso. */
            close(listening_socket); /* Cierra la copia de este socket */
          if (listener != NULL)
                *listener = -1; /*  Cerrado en este proceso. */
          }
          else {
            /* Este el ciclo principal. Cierra la copia del socket
conectado, y
               continua el ciclo. */

           close(connected_socket);
            connected_socket = -1;
          }
        }
      }
      return connected_socket;
    }
    else
      return listening_socket;
}

/*  Esta es una función general para hacer una conexion a un
servidor/puerto
     cualquiera.
     Service es el puerto nombre/numero.
     Type es SOC_STREAM o SOCK_DGRAM, y
      Netaddress es el nombre del host para conectarse.
      Esta función retorna el socket, listo para usarse. */
int make_connection(service, type, netaddress)
char *service;
int type;
char *netaddress;
{
  /* Primero se convierte el servicio de una cadena, a un numero... */
  int port = -1;
  struct in_addr *addr;
```

```
    int sock, connected;
    struct sockaddr_in address;

    if (type == SOCK_STREAM)
      port = atoport(service, "tcp");
    if (type == SOCK_DGRAM)
      port = atoport(service, "udp");
    if (port == -1) {
      fprintf(stderr,"Haciendo conexion:  Tipo de Socket Invalido.\n");
      return -1;
    }
    addr = atoaddr(netaddress);
    if (addr == NULL) {
      fprintf(stderr,"Haciendo conexion: Direccion de red Invalida.\n");
      return -1;
    }

    memset((char *) &address, 0, sizeof(address));
    address.sin_family = AF_INET;
    address.sin_port = (port);
    address.sin_addr.s_addr = addr->s_addr;

    sock = socket(AF_INET, type, 0);

   printf("Conectandose a %s sobre puerto
%d.\n",inet_ntoa(*addr),htons(port));

    if (type == SOCK_STREAM) {
      connected = connect(sock, (struct sockaddr *) &address,
        sizeof(address));
      if (connected < 0) {
        perror("connect");
        return -1;
      }
      return sock;
    }
    /* En otro caso, debe de ser udp, asi que se debe ligar a la
direccion. */
    if (bind(sock, (struct sockaddr *) &address, sizeof(address)) < 0) {
      perror("bind");
      return -1;
    }
    return sock;
}

/* Esto es como la llamada al sistema read(). */
int sock_read(sockfd, buf, count)
int sockfd;
char *buf;
size_t count;
{
   size_t bytes_read = 0;
```

```
  int this_read;

  while (bytes_read < count) {
    do
      this_read = read(sockfd, buf, count - bytes_read);
    while ( (this_read < 0) && (errno == EINTR) );
    if (this_read < 0)
      return this_read;
    else if (this_read == 0)
      return bytes_read;
    bytes_read += this_read;
    buf += this_read;
  }
  return count;
}
/* Esto es como la llamada al sistema write(), pero aplicado a sockets
*/
int sock_write(sockfd, buf, count)
int sockfd;
char *buf;
size_t count;
{
  size_t bytes_sent = 0;
  int this_write;

  while (bytes_sent < count) {
    do
      this_write = write(sockfd, buf, count - bytes_sent);
    while ( (this_write < 0) && (errno == EINTR) );
    if (this_write <= 0)
      return this_write;
    bytes_sent += this_write;
    buf += this_write;
  }
  return count;
}
/* Esta funcion lee de un socket, hasta que recibe una un caracter de
inicio de              linea. El llena el buffer "str" hasta el tama&o
maximo "count".
   Esta funciom debera regresar -1 si el socket es cerrado durante la
operación
   De lectura.
   Cabe se&alar que si una linea excede la longitud de count,los datos
extras
   Deberan ser descartados.*/
int sock_gets(sockfd, str, count)
int sockfd;
char *str;
size_t count;
{
  int bytes_read;
  int total_count = 0;
```

```
  char *current_position;
  char last_read = 0;

  current_position = str;
  while (last_read != 10) {
    bytes_read = read(sockfd, &last_read, 1);
    if (bytes_read <= 0) {
      /* El otro lado puede cerrar la conexion inesperadamente*/
      return -1;
    }
    if ( (total_count < count) && (last_read != 10) && (last_read !=13)
) {
      current_position[0] = last_read;
      current_position++;
      total_count++;
    }
  }
  if (count > 0)
    current_position[0] = 0;
  return total_count;
}
/* Esta función escribe una cadena  a un socket. Debera regresar -1
   si la conexion es cerrada  mientras  esta tratando de escribir.*/
int sock_puts(sockfd, str)
int sockfd;
char *str;
{
  return sock_write(sockfd, str, strlen(str));
}
/* Este funcion ignora la se&al SIGPIPE. SIGPIPE es enviada cuando se
trata
   de escribir a un socket desconectado.*/

void ignore_pipe(void)
{
  struct sigaction sig;

  sig.sa_handler = SIG_IGN;
  sig.sa_flags = 0;
  sigemptyset(&sig.sa_mask);
  sigaction(SIGPIPE,&sig,NULL);
}
```

Cabe mencionar que el ejemplo anterior, es una serie de funciones sencillas para realizar una pequeña aplicación en red, mediante el empleo de sockets, no se trata aqui de explicar en detalle la forma de implementación usando sockets, ya que esto requeriria el un trabajo particular, si no de explicar su importancia en el mundo de las comunicaciones de UNIX basadas en TCP/IP.

El programa desarrollado, fue un pequeño servidor denominado tcpserver, el cual simplemente lo que hace es convertir en mayúsculas, todas las cadenas que el programa cliente, denominado tcpclient, le envie en minúsculas, la forma de compilacación de los programas fue como sigue:

```
serunix%gcc  -c tcpserver.c
serunix%gcc -c tcpclient.c
serunix%gcc -o tcpclient tcpclient.o sockhelp.o -lsocket -lnsl
```

La instrucción anterior, se encarga de generar el programa ejecutable, ligando las librerias de sockets.

```
serunix%gcc -o tcpserver tcpserver.o sockhelp.o -lsocket -lnsl
```

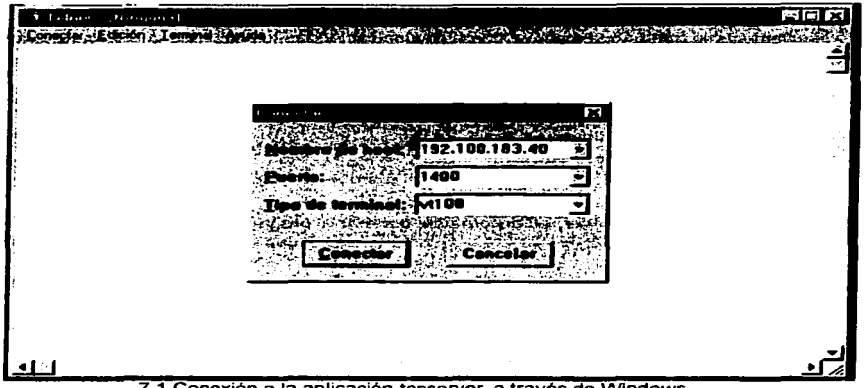
Para iniciar el programa, es como sigue:

```
serunix% tcpserver 1400&
```

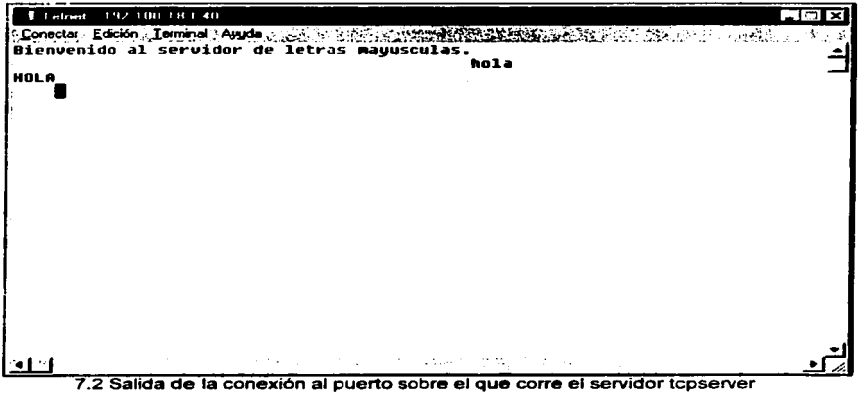
Y para correr al cliente, se teclea:

```
serunix% tcpclient 192.100.183.40 1400
Conectandose a 192.100.183.40 sobre el puerto 1400.
Server: Bienvenido al servidor de letras mayusculas.
Client: hola
Server: HOLA
Client: <CLOSED>
```

Y bajo una sesión Windows, para conectarse al Servidor, seria de la siguiente manera, como se muestra en la figura 7.1, y la salida de la conexión se muestra en la figura 7.2:



7.1 Conexión a la aplicación tcpserver, a través de Windows.



7.2 Salida de la conexión al puerto sobre el que corre el servidor tcpserver

7.10 CONCLUSIONES

Porque el software de protocolo TCP/IP reside dentro del sistema operativo, la interfaz exacta entre algún programa de aplicación y los protocolos TCP/IP depende sobre los detalles del sistema operativo; esta interfaz no esta especificada en el estándar del protocolo TCP/IP. Nosotros examinamos la interface socket usada en UNIX BSD, y mostramos que ha sido adoptada dentro del paradigma de apertura-lectura-escritura-cierre. Para usar TCP, un programa debe de crear un socket, ligarle una dirección, aceptar conexiones entrantes, y entonces comunicarse usando las primitivas read o write. Finalmente, Cuando finalizamos de emplear un socket, el programa debe de cerrarlo. En adición a la abstracción socket y a las llamadas al sistema que operan sobre los sockets, UNIX BSD incluye librerias de rutinas que ayudan a los programadores a crear y manipular direcciones IP.

La interface socket ha llegado a ser muy popular y es ampliamente soportada por muchos fabricantes. Los sistemas que no ofrecen las facilidades en su sistema operativo, frecuentemente proveen una libreria de sockets que permite a los programadores escribir aplicaciones usando llamadas a sockets aun cuando el sistema operativo subyacente use un conjunto de llamadas diferentes de sistema.

CONCLUSIONES

Actualmente podemos ver que muchos de las aplicaciones que corrían originalmente en sistemas UNIX han sido portadas a otras plataformas, así vemos que sistemas basados enWindows NT, contienen programas de DNS, servicios de correo electrónico basados en SMTP, servidores WEB, así como tambien servidores de FTP,etc.

Dichas plataformas son mucho mas económicas que sus contrapartes UNIX, y su administración y operación es mucho fácil que los programas basados en UNIX. Así podemos ver que en los próximos años se espera un gran avance de Sistemas basados en Windows NT, el cual se espera llegue a ocupar el primer lugar en Sistemas Operativos en Red, quedando UNIX en un tercer lugar, atras de Novell.

Actualmente a pesar del gran avance que ha tenido Windows NT, las aplicaciones de misión crítica de las grandes empresas aun continuan trabajndo en Sistemas UNIX, debido a la gran confiabilidad que ha demostrado este Sistema Operativo a lo largo de sus mas de 20 años de existencia.

Aunque también dentro de las apliciones críticas se empezara un gran empuje de Windows NT, que hoy por hoy, representa la plataforma operativa con mas crecimiento en el mercado, debido al gran dominio de las aplicaciones desktop de Microsoft que pueden ser llevadas a NT, sin ningun problema debido a que son totalmente compatibles, y son mucho más fácil de usar debido a que emplean la misma interfaz.

Pero aunque UNIX, sea desplazado del mercado, se espera una agresiva lucha por parte de los fabricantes de sistemas basados en UNIX, en lo que respecta a reducciones de precios, facilidades de uso, y compatibilidad con sistemas PCs. Pero donde aun podrá seguir subsistiendo es dentro del ámbito académico, ya que UNIX es verdaderamente una plataforma de desarrollo para aplicaciones basadas en red, ya que con un simple compilador de C, se pueden realizar aplicaciones de gran complejidad, a diferencia de las PCs, que debido al gran número de herramientas de desarrollo existentes, acaban por confundir a los desarrolladores, y ha sido dentro del ámbito académico donde han surgido las aplicaciones, siempre primeramente basadas en UNIX, que han revolucionado a la aplicaciones en red, y no aquellos estándares que han impuesto las grandes compañias del computo, por lo que podemos decir, que a UNIX, todavia le queda un gran camino por delante.

BIBLIOGRAFIA

1. **Computer Security Basics.**
Deborah Rusell & G.T. Gangemi Sr.. O'Relly & Associates, Second Edition, 1994.

2. **DNS and BIND.**
Paul Albitz & Cricket Liu. O'Relly & Associates,Second Edition 1994.

3. **Esential System Administration.**
AEleen Frisch. O'Relly & Associates, 1st Edition, 1991.

4. **Internetworking with TCP/IP. Volume I. Principles, Protocols and Architecture.**
Douglas E. Comer. Prentice-Hall, Second Edition, 1991.

5. **Mannaging NFS and NIS.**
Hal Stern. O'Relly & Associates, 1st Edition, 1991.

6. **Networking Essentials.**
MicrosoftPress. 1st Edition, 1996.

7. **OSI. A Model For Computer Communications. Standars.**
Uyless Black. Prentice-Hall. Second Edition, 1991.

8. **Practical UNIX Security.**
Simson Garfinkel & Gene Spafford. O'Relly & Associates, 1st Edition, 1991.

9. **System Performance and Tunning.**
Mike Loukides. O'Relly & Associates, 1st Edition, 1990.

10. **TCP/IP. Architecture, Protocols and Implementation.**
Sidnie Feit, et. al. McGraw-Hill, 1993.

11. **TCP/IP Ilustred. Volume 1. The Protocolos.**
W. Richards Stevens. Addison-Wesley, 1st Edition,1991.

12. **TCP/IP Network Administration.**
Craig Hunt, O'Relly & Associates, 1st Edition, 1992.

13. **TCP/IP & Related Protocols.**
Uyless Black. McGraw-Hill, Second Edition, 1994

14. **UNIX Power Tools.**
Jerry Peek, Tim O'Reilly, and Mike Loukides.
O'Reilly & Associates. 1st edition, 1993.