UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

Implantación de un Protocolo de Transporte en una Red Local

T E S I S

QUE PARA OBTENER EL TITULO DE

M A T E M A T I C O

P R E S E N T A

MONICA LEÑERO PADIERNA

Ciudad Universitaria, Octubre de 1992

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

IMPLANTACION DE UN PROTOCOLO
DE TRANSPORTE EN UNA RED LOCAL

INDICE

INTRODUCCION

El presente trabajo describe la implantación de un protocolo de transporte basado en TCP, bajo un sistema operativo UNIX. La necesidad de definir e implantar un protocolo para la capa de transporte surgió al empezar a trabajar en el proyecto "Sistemas Distribuidos en Redes y Sistemas Operativos Heterogéneos". Este proyecto se desarrolla con colaboración de personal de la Facultad de Ciencias y el IIMAS (Instituto de Investigación en Matemáticas Aplicadas y en Sistemas) de la UNAM y de LANIA (Laboratorio Nacional de Informática Avanzada. Xalapa, Veracruz), y cuenta con apoyo de la DGAPA (Dirección General de Apoyo al Personal Académico, UNAM).

El objetivo del proyecto es implantar una red propia para procesamiento distribuido. Debido a que se desea desarrollar la red partiendo de los componentes básicos (los físicos), no pueden tomarse programas ya desarrollados (experimental o comercialmente) para cubrir las diferentes capas; se necesita tener la capacidad de alterar la configuración de la red, sin que eso genere problemas de incompatibilidad. La filosofía de este proyecto es definir la red, sus protocolos e interfaces para poder adaptarlas a las necesidades propias; esto incluye el analizar e implantar las diferentes capas que forman la red.

Este trabajo cubre una de las capas que forman la red: la capa de transporte. Transporte marca la frontera entre las capas que tratan directamente con los aspectos de implantación física de la red y las que usan los servicios que proporcionan las capas inferiores, asumiendo que cuentan con un sistema de comunicación confiable. La capa de transporte es la encargada de ocultar a las capas superiores a ella, los detalles relacionados con la

implantación de la red; además, es responsable de proporcionar un servicio de comunicación **confiable**, independientemente del tipo de red con que se trabaje. El contar con una implantación del protocolo de transporte proporciona la ventaja de poder experimentar con diferentes implantaciones de red (comunicación por puerto serial, *ethernet*, etc) y con diversas interfaces hacia los procesos de aplicación.

El trabajo se organiza de la siguiente manera:

En el capítulo 1 se presentan los conceptos que se manejarán a lo largo del trabajo; se analizan las redes de computadoras, los protocolos, las tendencias en estandarización y el modelo de referencia OSI.

En el capítulo 2 se justifica la implantación del protocolo de transporte y se analiza la capa de transporte partiendo de los dos modelos de referencia más importantes: TP4 (de OSI) y TCP (de DARPA); este último es el modelo en el que se basa el protocolo que se implanta.

En el capítulo 3 se presentan los aspectos más importantes de la implantación: las definiciones, estructuras de datos, algoritmos y el código de las funciones que se usan en el protocolo. Los ejemplos que se incluyen en este capítulo son los más releventes por su complejidad o por el papel que desempeñan en el protocolo.

En el capítulo 4 se describen las pruebas que se realizaron para validar el protocolo de transporte. La validación se llevó a cabo implantando un programa sobre el protocolo de transporte y probando que dicho programa se comporta de manera análoga cuando funciona sobre el TCP de la HP.

Finalmente, se plantean las perspectivas para desarrollos futuros en la implantación del protocolo de transporte y su

relación con las siguientes etapas del proyecto.

La parte de programación del protocolo de transporte no fue desarrollada en su totalidad porque transporte es la capa más grande y compleja en la arquitectura de una red y desarrollarla completamente sería una tarea demasiado larga y pesada para una sola persona. Existe un *software* de dominio público desarrollado por Phil Karn y su grupo, que sigue el modelo ARPA y cubre cada una de las capas de dicho modelo. El *software* original ocupa 1.5 *Mbytes* aproximadamente y tiene 190 archivos; se programó de tal manera que, teóricamente, es posible hacerlo funcionar en diferentes tipos de máquinas y con distintos sistemas operativos. La ventaja de este *software* es que se cuenta con el código fuente, por lo que puede experimentarse con la implantación.

Partiendo del código desarrollado por Karn, fue necesario, después del estudio detallado de los modelos de referencia OSI y ARPA,

1. Entender la filosofía de funcionamiento del código de Karn (incluye análisis de código y pruebas en diferentes sistemas operativos).

2. Separar las funciones de Karn que son superfluas para la red local y la máquina en que se implantó el código (HP 9000/840), y que no corresponden al protocolo de transporte.

3. Modificar y adaptar el código para que funcionara en la HP (el código fuente original considera diferentes sistemas operativos sobre los que puede funcionar, pero el de la HP del Centro de Cómputo de la Facultad de Ciencias no estaba incluido, por lo que fue necesario adaptar funciones y estructuras de datos a las características de la HP).

A pesar de que no se desarrolló en su totalidad el *software* del protocolo de transporte, creo que los conocimientos y la experiencia adquirida al trabajar con OSI, ARPA, la red local y el código de Karn son invaluables y serán un gran apoyo para alcanzar las metas que se plantean en el proyecto. Este es uno de los primeros pasos del proyecto en el área de protocolos y ha resultado un fundamento firme y positivo.

1. CONCEPTOS INTRODUCTORIOS

REDES DE COMPUTADORAS

Al empezar a usarse comercialmente las computadoras (a principios de los años 50), se contaba con máquinas muy grandes y caras. Estas máquinas funcionaban por medio de tubos al vacío (bulbos) activados a través de impulsos electrónicos; estos elementos ocupaban mucho espacio y era necesario tener sistemas de ventilación e instalaciones muy sofisticadas para su buen funcionamiento, a pesar de lo cual, el tiempo de procesamiento era lo más costoso. Las primeras máquinas tenían alrededor de 18 000 bulbos electrónicos, efectuaban 300 multiplicaciones por segundo y pesaban más de 30 toneladas, ocupando una superficie de 180 metros cuadrados.

Al final de los años 50, los tubos al vacío son sustituidos por transistores [1]. Los transistores son mucho más baratos y más rápidos que los bulbos, son más pequeños y la cantidad de calor que generan es mínima, con lo que el tamaño de las máquinas se reduce al poder integrar más componentes en menos espacio; gracias a esto se logra reducir el tamaño y complejidad del equipo. Aparecen entonces los sistemas de tiempo compartido (60's), que permitían compartir recursos y tiempo de procesador a un conjunto de usuarios. Aún cuando estos sistemas fueron adecuados para satisfacer las necesidades de procesamiento que había en ese momento, el número de usuarios aumentó y los sistemas se volvieron sumamente lentos.

[1] El transistor es inventado en 1947 por W. H. Brattain y W. Shockely y es un dispositivo electrónico formado por un cristal de silicio o germanio.

La tecnología siguió desarrollándose y los recursos se abarataron. Con objeto de proporcionar a los usuarios recursos descentralizados y sistemas más flexibles, se introducen las minis y microcomputadoras (70's). Las microcomputadoras son más baratas y permiten a los usuarios administrarlas, pudiendo agregarles recursos según sus propias necesidades. Sin embargo, aún con las ventajas que representa la independencia, se pierden características importantes con las que generalmente se contaba en los sistemas de tiempo compartido; por ejemplo: cuando dos o más programadores trabajan en un proyecto conjunto, generalmente necesitan intercambiar información, o consultar una base de datos común, lo que ya no es posible dado que el esquema de trabajo es descentralizado y en máquinas independientes.

Aún cuando el precio de los recursos se ha reducido, no es costeable tener equipo especializado para cada programador que participa en un proyecto, por lo que se necesita brindar a los usuarios la posibilidad de compartir recursos. Por otro lado, desde que aparece la computación ha existido la necesidad de intercambiar información; con el fin de satisfacer ambas premisas (compartir recursos e información) nacen las redes de computadoras.

Por **redes de computadoras** se entenderá un conjunto de computadoras **autónomas** interconectadas, siguiendo la definición de Tanenbaum [Tanenbaum]. Las redes de computadoras ponen a disposición de cualquier usuario de la red los programas, datos y recursos con que se cuenta, sin importar donde se encuentre físicamente dicho usuario, preservando la independencia de cada máquina. Algunas de las ventajas que se obtienen al trabajar con redes son:

- Posibilidad de intercambio de información.
- Puede agregarse equipo para satisfacer necesidades futuras.
- Las fallas que se presentan en una parte de la red no afectan a las otras máquinas.
- Permiten compartir recursos.

CLASIFICACION.

Dos aspectos importantes que deben considerarse al planear una red son qué tan grande será la red (qué tan alejado está el equipo que se va a conectar) y cómo se llevará a cabo la conexión física de las máquinas; estos aspectos determinan dos de las clasificaciones importantes que se han hecho de las redes de computadoras.

1. ALCANCE GEOGRAFICO

De acuerdo a la distancia a la que pueden estar los equipos que conectan, las redes se clasifican en los siguientes tipos:

o Las **redes extensas** (**WAN** por sus siglas en inglés *Wide Area Network*) se usan para conectar recursos geográficamente dispersos, por lo que se necesita equipo especializado para establecer la conexión. Este tipo de redes son recomendables para conectar equipo separado por grandes distancias y cuando se requiere comunicación de grandes volúmenes de datos. Pueden implantarse a través de líneas telefónicas, microondas o satélite, lo que hace que la velocidad de transmisión de datos varíe entre 1200 *bits* y 50 *Mbits* por segundo. Como desventaja se tiene que la comunicación depende del estado o disponibilidad del equipo especializado para establecer la conexión, los factores ambientales y resultan muy costosas. Otro problema importante es que el medio de comunicación no es propio, lo que puede dar lugar a complicaciones administrativas (la planeación de crecimiento, reparaciones y administración real del medio no corresponde a los administradores de la red).

o **Las redes de área local (LAN** por sus siglas en inglés *Local Area Network*) pueden definirse como sistemas de comunicación de datos que permiten intercambio confiable de información entre dispositivos independientes que se encuentran en una área limitada geográficamente (1m. a 10km). Se usan medios de transmisión y dispositivos generalmente baratos y confiables, entre los que se encuentran el cable coaxial, el par trenzado y la fibra óptica. Una ventaja importante de estas redes es que el medio de transmisión es propio. La velocidad de transmisión de datos puede variar entre 100 Kbps y 100 Mbps.

o **Las redes metropolitanas (MAN** por sus siglas en inglés *Metropolitan Area Network*) son una clasificación intermedia de redes locales y redes extendidas; abarcan una ciudad (por lo que podrían considerarse WAN), pero usan tecnología de redes locales (por lo que podrían clasificarse como LAN).

2. COMUNICACION

Debido a que las redes están formadas por un conjunto de computadoras autónomas se necesita algún medio para permitirles intercambiar información. La comunicación física en la red puede hacerse de dos maneras:

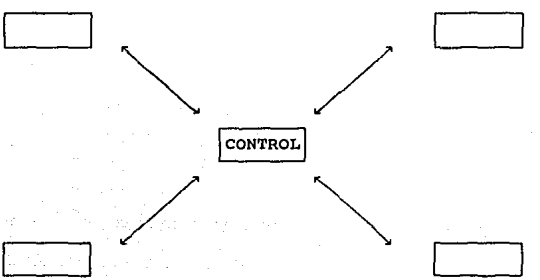
Punto a punto.
En difusión (*broadcast*).

o En la comunicación punto a punto existen líneas de comunicación directa entre las máquinas. Si alguna quiere transmitir datos a otra a la que no está conectada directamente, la comunicación se lleva a cabo de manera indirecta: se envía el mensaje, que va pasando de máquina a máquina hasta que llegue a su destino. La máquina que lo recibe lo almacena completo de manera temporal, decide si es para ella o no y en caso de que no sea la

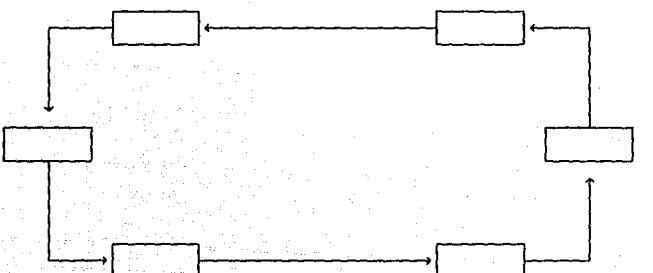
máquina destino, lo retiene hasta que su propia línea de salida se desocupe, momento en el que retransmite el mensaje.

Con este esquema, las maneras en que puede existir una comunicación punto a punto se clasifican de acuerdo a cómo está conectado el equipo que forma la red, por medio de las siguientes **topologías:**

▪ **Estrella** (*star*). Se tiene un control central por medio del que se conecta todo el sistema, lo que permite un fácil acceso a los recursos (la trayectoria que debe seguirse para comunicarse con alguien es siempre la misma: de la máquina fuente al control central y del control central a la máquina destino). Otra ventaja que tiene esta topología es que el agregar o eliminar equipo es muy fácil y no afecta la comunicación de la red, pero se tiene la desventaja de que el sistema debe ser muy confiable, ya que si el control central llega a fallar, toda la red queda incomunicada.

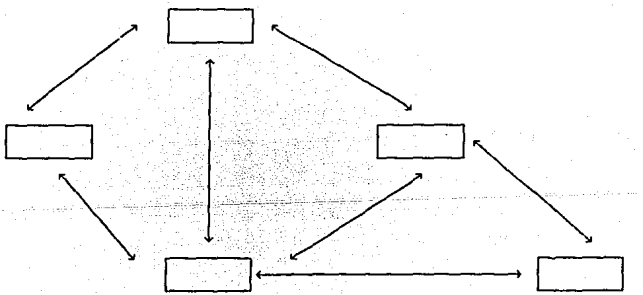


■ **Anillo** (*ring*). Se cuenta con una serie de conexiones punto a punto entre las que se transmiten los mensajes en una dirección; cada máquina está conectada a dos y sólo dos miembros del anillo; los mensajes transmitidos eventualmente regresan al emisor para permitirle efectuar las verificaciones necesarias. Las desventajas que se tienen son que para agregar o eliminar equipo debe interrumpirse la comunicación, lo que también ocurre si alguna máquina falla; por otro lado, la comunicación puede ser muy lenta debido a que los mensajes tienen que dar una vuelta completa a la red.



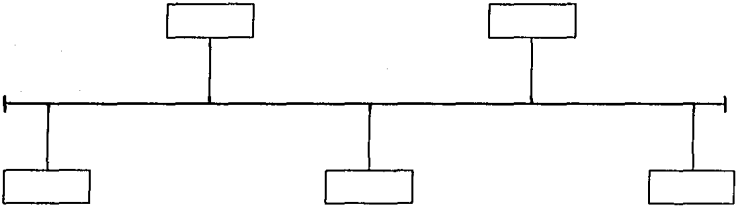
■ **Irregular** (*Mesh*). Una red irregular es aquella en la que para cada par de nodos existen al menos dos trayectorias ajenas [*IEEE*]. La ventaja de esta topología es que no hay problema alguno para añadir o eliminar equipo de la red y no se interrumpe la comunicación si alguna máquina falla. La desventaja es que el manejo de este tipo de redes es muy complejo. Deben mantenerse bases de datos acerca de la ubicación del equipo que está

conectado a la red, para poder establecer la comunicación y elegir la trayectoria que debe seguir un mensaje para llegar a su destino.



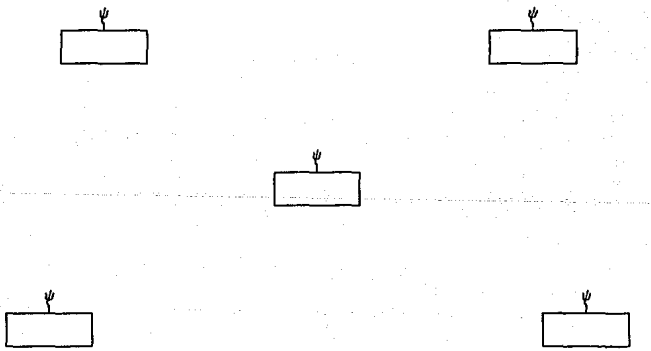
o En la comunicación por **difusión**, existe un medio de comunicación que se comparte, de tal manera que al transmitir un mensaje todas las máquinas conectadas al medio "escuchan" la transmisión. Debido a ésto, no es necesario implantar mecanismos para elegir la ruta que debe seguirse para llegar al destino. Otra ventaja que tiene esta manera de intercambiar información es que, en la mayoría de los casos, puede añadirse o eliminarse equipo sin que se interrumpa la comunicación en la red. Como en este esquema de comunicación todos tienen acceso al medio de transmisión, deben implantarse mecanismos de administración del medio. Como ejemplos de redes que usan un medio de comunicación para la difusión de los mensajes se tienen:

▪ Red multipuntos o **Canal** (*Bus*). Cada máquina está conectada a un medio de comunicación multipuntos (canal de comunicación) en el que se difunden los mensajes en ambas direcciones. Normalmente se utiliza para redes que realizan transmisiones a baja velocidad, aunque, dependiendo del medio de transmisión, la velocidad llega hasta 10 *Mbits* por segundo (en el caso de *ethernet*). Debido a que la señal se envía en ambos sentidos del canal de comunicación, es necesario tener terminadores en los extremos que "absorban" las señales que llegan a ellos, para evitar que los datos vuelvan a ser difundidos a través del canal.



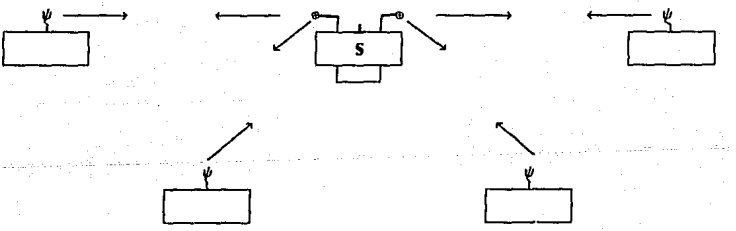
▪ **Radio.** Para la comunicación por medio de ondas de radio se necesitan antenas de difusión. Todas las estaciones comparten la misma frecuencia de radio. Cada estación está dentro del rango de tansmisión de las otras y los mensajes se difunden directamente (de una estación a las otras). Una desventaja importante es que la instalación del equipo especializado que se necesita para la comunicación (antenas) resulta muy costosa o tiene alcance limitado (radio *modems*) y además la comunicación es generalmente

lenta y se ve afectada por factores ambientales (lluvia).



- **Satélite.** El satélite es una estación de relevo de señales usado para comunicar receptores/transmisores de microondas (llamados estaciones terrestres). La comunicación por medio del satélite se realiza de la siguiente manera: el satélite recibe transmisiones en una frecuencia de banda, las amplifica o repite la señal y la transmite en otra frecuencia. Algunas de las ventajas que se tienen en la comunicación por satélite son que el costo de la transmisión es independiente de la distancia siempre y cuando las estaciones se encuentren dentro del alcance del satélite, la calidad de la transmisión es normalmente muy buena, el retraso en la propagación de las señales tierra-satélite-tierra es de un cuarto de segundo y la velocidad de transmisión es alta. Como en el caso de la comunicación por radio, una desventaja importante es el costo de las instalaciones y la comunicación

también se ve afectada por factores ambientales (aunque en menor proporción).



La **administración del medio** de transmisión puede realizarse a través de asignación estática o dinámica del medio. En la **estática**, la asignación se hace a través de intervalos discretos de tiempo dentro de los que cada máquina puede usar el medio. Esto tiene la desventaja de que aún cuando una máquina no tenga datos por transmitir, su tiempo le está asignado y nadie más puede enviar datos. Para la administración **dinámica** del medio se emplean dos técnicas: centralizada y descentralizada.

En la **centralizada** existe un árbitro que determina quién puede transmitir. Un ejemplo de administración centralizada del medio es la que se utiliza en *Token Ring* (IBM). La administración se lleva a cabo por medio de un paquete (*token*) que circula por el anillo; si no hay máquina que desee transmitir, el *token* se marca como libre; si alguna máquina quiere enviar datos, espera a que el *token* llegue a ella, lo marca como ocupado y transmite los paquetes de datos inmediatamente después del *token*. Para terminar

de transmitir, la máquina espera a que el *token* dé la vuelta completa al anillo ya que terminó de enviar sus datos (ambas condiciones son necesarias para liberar el *token*). Marcar el *token* como ocupado garantiza que ninguna máquina podrá transmitir mientras otra lo está haciendo. Sin embargo, pueden presentarse problemas como que el *token* esté siempre ocupado y no se dé oportunidad a otras máquinas de transmitir, o que no haya *token* circulando por el anillo. Para solucionar estos problemas, existe una estación especial que se llama monitor. El monitor es el árbitro del sistema y su objetivo es garantizar la funcionalidad y unicidad del *token* que circula por el anillo.

En el caso de administración descentralizada, se tiene el ejemplo del protocolo usado por *ethernet*: en él, cada máquina avisa de su deseo de usar el medio y si nadie se opone, lo hace. En el caso de que alguien se oponga a que el medio sea usado (esto es, porque desee también transmitir), cada máquina se pone a sí misma en espera por un tiempo aleatorio, después del cual vuelve a manifestar su deseo de usar el medio de transmisión.

ARQUITECTURA DE RED.

Para reducir la complejidad del diseño, las redes se organizan como una serie de capas o niveles; cada nivel se construye sobre el nivel anterior y tiene por objeto ofrecer servicios (conjunto pequeño de funciones relacionadas) a las capas superiores, evitándoles los detalles de su implantación; cada nivel se comunica con las capas adyacentes usando dichos servicios (que pueden implantarse como llamadas a funciones o procedimientos) y por medio de intercambio de parámetros. La ventaja de seguir este diseño es que si cambia la implantación de alguna de las capas (por ejemplo, que el medio de comunicación cambie de cable coaxial a fibra óptica), ninguna otra capa resulta afectada si la nueva implantación proporciona los mismos servicios

que la anterior. Cada capa puede pensarse como un programa o proceso que tiene como objetivo comunicarse con el proceso correspondiente en otra máquina. A este tipo de diseño se le conoce como **modelo jerárquico por capas.**

Para poder establecer la comunicación en la red, deben seguirse ciertas reglas y convenciones que especifican los formatos, tiempos de espera, manejo de canales lógicos y elección de rutas, control de flujo de datos y errores, congestión, etc. Este conjunto de reglas es lo que se conoce como **protocolo de comunicación.** El objetivo principal del protocolo de comunicación es establecer el intercambio ordenado de información entre procesos y manejar de manera eficiente los recursos de la red. Las reglas para el intercambio de información entre las entidades que manejan niveles correspondientes en diferentes máquinas se conocen como protocolos homólogos.

El protocolo de una red generalmente cubre tres aspectos:

Sintaxis.- Estructura de los datos y de los mensajes de control.

Semántica.- Mensajes de control que se intercambiarán, acciones por ejecutarse y respuestas que se regresarán.

Eventos en el tiempo (*timing*).- Especifica el orden en que deben ocurrir los eventos.

y está formado por tres componentes básicos:

Entidades.- Programas de aplicación y/o máquina en la que se ejecuta el programa de aplicación.

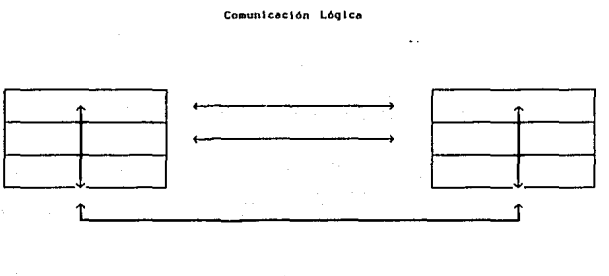
Asociación de entidades (*binding*).- Trayectoria lógica de comunicación entre dos entidades. Puede ser permanente o temporal.

Mecanismo de comunicación.- Para transportar información entre las entidades asociadas.

Dados estos tres componentes básicos, el protocolo de red debe establecer las reglas para usar y manejar los servicios de su capa inferior y ofrecer un servicio transparente a los usuarios de su capa superior.

En el caso de intercambio de información entre procesos correspondientes de máquinas diferentes, cada homólogo supone que se comunica directamente con su contraparte y sólo se apega al protocolo que existe entre ellos; aunque en realidad los mensajes se van pasando hacia la capa inmediata inferior, hasta llegar a la capa que maneja la conexión física, que es la única que verdaderamente se comunica con su homóloga.

El esquema de comunicación queda de la siguiente manera:



A través de protocolos se definen: las rutas que deben seguir los mensajes; cómo dividirlos en paquetes más pequeños en caso de que sea necesario; las rutinas para manejo de errores, control de

flujo y congestión; manejo de prioridades; sincronización de procesos; cómo establecer, mantener y terminar la comunicación entre procesos; transmisión y entrega secuencial de datos; uso eficiente de recursos; transmisión confiable, manejo de niveles de seguridad, etc.

Al conjunto de capas y protocolos que definen la comunicación se le conoce como **Arquitectura de Red.**

ESTANDARIZACION

Debido a las ventajas que representa el trabajar con redes, en muchas organizaciones comenzaron a implantarlas definiendo la arquitectura de red que más convenía a sus necesidades. Sin embargo, el conectar redes independientes para aprovechar al máximo los recursos fue imposible, ya que cada red tenía sus propias capas y protocolos para establecer la comunicación. Por otro lado, para poder usar el equipo que las grandes compañías producen, es necesario que la red a la que se quiere añadir el equipo entienda el protocolo que éste trae de fábrica. Para solucionar los problemas anteriormente descritos, se definen **estándares.** Informalmente hablando, definir un estándar es especificar la manera en que debe llevarse a cabo una tarea.

EL MODELO OSI DE ISO

Entre las organizaciones que se han dado a la tarea de definir estándares para redes de computadoras se encuentra ISO (*International Standards Organization* [Saal]). Uno de sus objetivos es desarrollar y validar estándares para sistemas de información distribuida. ISO define los aspectos básicos de funcionalidad para lograr la comunicación entre procesos de

sistemas heterogéneos, por medio de un modelo de referencia que se llama OSI (*Open Systems Interconnection*); en él se establecen las capas, interfaces (protocolos entre los niveles que integran una máquina) y protocolos que deben formar una red. Lo llaman **modelo abierto para la interconexión de sistemas** porque se pretende tener la capacidad de conectar equipo de diferentes marcas, si se sigue el modelo de referencia. La importancia de este modelo es que permite la comunicación de procesos independientemente del ambiente en que se ejecuten. El modelo abierto de ISO para la interconexión de sistemas define siete capas para una máquina dentro de una red:

Nivel	Nombre de la capa
1	Física (*Physical*)
2	Enlace de Datos (*Data Link*)
3	Red (*Network*)
4	Transporte (*Transport*)
5	Sesión (*Session*)
6	Presentación (*Presentation*)
7	Aplicación (*Application*)

La función de la capa física es transmitir un flujo de *bits* por la red, es decir, controla el flujo de *bits* entre las interfaces de comunicación de los nodos interconectados. A la capa física no le afecta la manera en que los *bits* se agrupan ni se ocupa del significado que puedan tener. La capa física es la única que realmente se comunica con su capa homóloga.

Debido a que la capa física no está capacitada para detectar o corregir errores en la transmisión, ni para identificar si los datos llegan al receptor más rápido de lo que los puede procesar, la capa de enlace de datos proporciona las funciones de control de errores, sincronización y administración de la línea física de comunicación.

La capa de red permite a los procesos enviar unidades de

datos, aún cuando no estén conectados a una línea física dedicada. Incluye funciones para elegir la trayectoria que un mensaje debe seguir para llegar a su destino (ruteo, o en inglés *routing*) y control de flujo y congestión para redes y líneas, por lo que sus protocolos dependen fuertemente de la topología de la red. Por medio de los servicios de la capa de red se intercambian paquetes de información independientes (llamados **datagramas**).

Las tres capas anteriores manejan directamente aspectos de implantación de la red, como son tarjetas, líneas de comunicación, etc. y, en el caso de la capa de red, debe conocerse la topología de la red (o redes) con que se trabaja para poder llevar a cabo la elección de la mejor ruta por la que se transmiten las unidades de datos. La capa de **transporte** oculta a las capas superiores todos los aspectos mencionados, de tal manera que es transparente para ellas el tipo de red sobre la que están trabajando; se encarga de establecer y manejar las conexiones lógicas (líneas lógicas de comunicación) para el intercambio de información entre procesos y garantiza que dicho intercambio sea confiable. Otra de sus funciones es mejorar la calidad y características del servicio de comunicación proporcionado por la capa de red.

La capa de **sesión** proporciona los medios para sincronizar el diálogo y manejar el intercambio de datos de los procesos que residen en la capa de presentación. Establece conexiones (llamadas **sesiones**) entre parejas de procesos, es decir, cuando un proceso desea hablar con otro pide una sesión a esta capa, que usa los servicios de la capa de transporte para establecer la comunicación con la máquina donde se ejecuta el segundo proceso.

La capa de **presentación** se encarga de representar la información de tal manera que se preserve su significado, a pesar de las diferencias de código, formatos o la manera en que puedan comprimirse o dividirse los datos durante la transmisión. Algunas transformaciones útiles de la información que se realizan a este nivel son: compresión de texto, encriptamiento y conversión de y

hacia formatos estándares de red.

La capa de aplicación proporciona servicios a los usuarios de la red.

Para que cada capa pueda llevar a cabo correctamente la función que tiene asignada se definen reglas (protocolos) para el intercambio de datos.

Existen protocolos para el manejo de canales lógicos (conexiones), transferencia confiable de datos (a diferentes niveles), elección de rutas, control de flujo de datos y congestión, para prevenir bloqueo mutuo (*deadlock*), etc.

En resúmen, las redes de computadoras surgen para satisfacer necesidades importantes en el mundo de la computación (compartir recursos e información), lo que ha ocasionado que sean fácilmente aceptadas y rápidamente implantadas. Esto presenta el peligro de implantar redes que sólo sean capaces de funcionar en el caso particular para el que fueron desarrolladas. La implantación de redes que funcionen en casos particulares puede y debe evitarse siguiendo los estándares (modelos de referencia) que se han desarrollado, lo que implica conocer las diferentes capas y los servicios y protocolos que forman el modelo de referencia sobre el que se trabaje.

BIBLIOGRAFIA

REDES

- *Design and analysis of computer communications networks*
  Vijay Ahuja [Ahuja]
  Ed. *Mc Graw Hill Computer Science Series*
  1985

Además de los aspectos que comunmente se estudian de redes, este libro trata con muy buen detalle la parte de protocolos: técnicas para describir protocolos (diagrama de transición de estados, redes de Petri, BNF) y para validarlos.

- *Introduction to Local Area Computer Networks*
  K. C. E. Gee [Gee]
  Ed. MacMillan
  1983

Descripción formal e informal de las redes de área local; trata las diferentes topologías, medios de transmisión, requerimientos de *software* y *hardware*; aplicaciones de redes locales.

- *A lan primer*
  Dick Lefkon [Lefkon]
  Byte
  julio de 1987 pp. 147-158

Artículo descriptivo, aunque no recomendable para empezar debido a que tiene cuestiones técnicas complicadas. Habla de topologías y estándares (802.2 a 802.6) y tiene una breve (pero buena) descripción de las capas de OSI.

- Seminario de comunicaciones
  Ken Roberts [Roberts]
  Digital equipment corporation
  Decus, México
  noviembre de 1987

Panorama general de redes: funciones básicas, redes locales y extensas, topologías, el modelo OSI, las arquitecturas DNA (*Digital Network Architecture*) y SNA (*System Network Architecture*), diseño de aplicaciones en un ambiente distribuido, aspectos de seguridad, crecimiento de las redes, perspectivas.

- *Local area networks*
  Dr. Harry J. Saal [Saal]
  Byte
  octubre de 1981 pp. 92-112

Buen artículo introductorio. Habla un poco de tarjetas y *ethernet*. Describe topologías, estándares y brevemente las capas de OSI. Tiene un ejemplo práctico de lo expuesto en el artículo: *Netstar Cluster/One Model A* (LAN basada en *ethernet*) que funciona con *Apple* II.

- *Data and Computer Communications*
  W. Stallings [Stallings]
  Segunda edición
  Ed. MacMillan

Muy buen libro de redes. Habla de comunicación de datos: transmisión, codificación, multiplexaje; redes de comunicación de datos: técnicas para comunicación en redes, permutación de circuitos y paquetes; redes locales; arquitecturas, protocolos, comunicación entre redes (*internetworking*); a lo largo del libro, se tratan a detalle los protocolos del modelo OSI.

- *IEEE Project 802 :setting standards for local area networks*
  W. Stallings [Stallings-2-84]
  Computerworld
  febrero de 1984 pp. ID/27-ID/33

Estandarización y redes locales. El punto de vista de quienes definen los estándares.

- *Local networks*
  W. Stallings [Stallings-3-84]
  Computing Surveys V16 N1
  marzo de 1984 pp. 3-41

Artículo muy completo (y algo complejo). Trata de topologías, medios de transmisión, protocolos para control de acceso al medio, estándares y los sistemas *baseband* y *broadband*.

- *Computer Networks*
  A. S. Tanenbaum [Tanenbaum]
  Ed Prentice Hall
  1981

Muy buen libro. Estudia detalladamente el modelo OSI (capas, y servicios); analiza las características de cada capa, dedicándole un capítulo completo; complementa el buen nivel teórico del libro con ejemplos interesantes. Excelente fuente de referencias bibliográficas.

PROTOCOLOS

- *Design and Validation of Computer Protocols*
  Gerard J. Holzmann [Holzmann]
  Ed. Prentice Hall
  1991

Libro que trata con muy buen nivel de formalidad los aspectos de definición, especificación y validación de protocolos.

- *A tutorial on protocols*
  Louis Pouzin & Hubert Zimmermann [Pouzin]
  Proceedings pf the IEEE V66 N11
  Nov. de 1978

Introducción a protocolos y sus elementos básicos; concepto de arquitectura de un sistema distribuido; elementos funcionales de las capas típicas; cuestiones de implantación. Artículo bastante completo. Pasa de definiciones informales a discusiones de las técnicas más usadas.

- *Network protocols*
  Andrew S. Tanenbaum [Tanenbaum-81]
  Computing Surveys V13 N4
  dic. de 1981

Artículo que trata a detalle protocolos que forman parte de cada una de las capas del modelo OSI. Incluye ejemplos (X.21, HDLC, X.25) para las tres primeras capas y analiza a detalle protocolos de las capas superiores (manejo de conexiones y control de flujo, transferencia de archivos, compresión de texto, manejo de terminales virtuales).

ISO y OSI

- *OSI: A model for Computer Communications Standards*
  Uyless Black [Black]
  Ed. Prentice Hall
  1991

Muy buen libro que estudia detalladamente cada una de las capas del modelo OSI. Define la terminología del modelo, aplicándola a cada capa; estudia diferentes posibilidades de implantación para las capas.

- *Of local networks, protocols and the OSI reference model*
  F. M. Burg, C. T. Chen & H. C. Folts [Burg]
  Data communications
  noviembre de 1984 pp. 129-150

Define conceptos importantes de redes y comunicación (orientada y no orientada a conexión); relaciona las capas de OSI con los estándares IEEE 802; analiza las capas de la red desde un punto de vista funcional y X.25 en redes locales.

- *A tutorial on the Open Systems Interconnection Reference Model*
  Harold C. Folts [Folts]
  Open Systems Data Transfer, N 1
  junio de 1982 pp. 2-21

Artículo intermedio. Panorama general de OSI, definición de conceptos importantes del modelo (capa, conexión, SAP, etc), funcionamiento de cada capa y servicios mínimos que debe proporcionar; describe la manera en que interactúan las capas cuando se establece comunicación entre procesos de la capa de aplicación.

- *Data communications, computer networks and OSI (2nd. edition)*
  Fred Halsall [Halsall]
  Ed. Addison-Wesley
  1992

Arquitecturas de sistemas distribuidos, protocolos, aspectos de *hardware*, redes públicas y locales. Libro bastante completo.

- *Open systems, today and tomorrow*
  Sigman Schindler [Schindler]
  Computer Networks 5
  1981 pp. 167-176

Plantea las perspectivas de los sistemas abiertos desde

diferentes puntos de vista (económicos, prácticos, etc).

- *The OSI Reference Model*
  John D. Day & Hubert Zimmermann [Day]
  Proceedings of the IEEE Vol. 71 No. 12
  diciembre de 1983

Este es un buen artículo para el estudio introductorio de la nomenclatura y los conceptos esenciales del modelo de referencia OSI (arquitectura, capas del modelo, SAP's, etc).

2. PLANTEAMIENTO DEL PROBLEMA

El diseño de una red de computadoras toma como referencia los modelos estándares para hacerla lo más abierta posible. En la mayoría de los modelos de referencia se describe la arquitectura de red a un nivel tal que los programadores puedan desarrollar *software* para proporcionar los servicios de cada capa, que aseguren el acceso eficiente a los recursos a los procesos de las capas superiores.

El objetivo de este trabajo es implantar el *software* necesario para obtener los servicios equivalentes al nivel cuatro del modelo OSI de ISO (capa de transporte). Se trabaja con la capa de transporte porque ésta tiene la función de proporcionar un servicio confiable de comunicación (lo que garantiza el buen funcionamiento de la red), ocultando a las capas superiores los detalles que tienen que ver con la implantación de la red (por ejemplo, el medio físico por el que se establece la comunicación).

La implantación del protocolo de transporte debe seguir un modelo de referencia. El modelo OSI de ISO es muy completo, pero debido a su complejidad continua perfeccionándose, por lo que no es conveniente implantar los servicios de transporte basándose en él. Para que la implantación pueda apegarse a un modelo real y en funcionamiento, se decidió trabajar con la arquitectura planteada por ARPA (*Advanced Research Projects Agency* del Departamento de Defensa de los E. U. A.). Esta arquitectura se analiza posteriormente.

El llevar a cabo la implantación proporciona la ventaja de poder controlar a detalle todos los aspectos relacionados con el protocolo, además de que permite adaptarlo a necesidades particulares, lo que es fundamental para el proyecto a partir del

que surge este trabajo.

LA RED LOCAL.

La red local sobre la que se implanta el protocolo de transporte está formada por:

dos computadoras *GATEWAY* 2000 con procesador Intel 80386, conectadas a *ethernet* y funcionando, por el momento, con MS-DOS.

dos estaciones de trabajo *SUN IPC*, conectadas a *ethernet* y con sistema operativo propio basado en UNIX.

una HP 9000/840 también conectada a *ethernet*, con sistema operativo HP-UX versión 7.0 (sistema propio basado en UNIX)

La implantación del protocolo de transporte se realizó en la HP 9000/840; las pruebas para validación se llevaron a cabo en la propia HP y usando el resto del equipo que forma la red local. Se asume que se cuenta con los servicios correspondientes a los niveles inferiores (equivalentes a los proporcionados por las capas de red, enlace de datos y física). El protocolo de transporte se implanta en una red local, comunicada a través de *Ethernet*.

ETHERNET

Informalmente hablando, la manera en que funciona la red con *Ethernet* es la siguiente: supóngase que en un cuarto obscuro pequeño se encuentra un grupo de personas platicando. Cada persona escucha lo que cualquier otra esté diciendo (sensibilidad al medio, o en inglés, *carrier sense*). Todos en la habitación tienen el mismo derecho de empezar a hablar (acceso múltiple -*multiple*

access-) y ninguno dice grandes discursos para no importunar a los demás (si alguno lo hace, se le invita amablemente a hablar con mesura). Nadie habla mientras otro lo esté haciendo y si varios comienzan a hablar simultáneamente, se dan cuenta de la situación al escuchar algo diferente a lo que dijeron (detección de colisión -*collision detection*-) y esperan un momento antes de intentar volver a tomar la palabra. Es importante resaltar el hecho de que cada persona tiene un nombre único, que sirve para evitar confusiones (la analogía con la red es que cada máquina tiene una dirección *ethernet* única). Cuando alguien habla, el mensaje es precedido por el nombre de la persona que habla y el de aquella a quien se dirige (direcciones fuente y destino). Si el mensaje se dirige a todo el auditorio, quien habla debe especificar que el destino es "todos" (mensaje en difusión, o en inglés, *broadcast address*).

Formalmente hablando, la comunicación con *Ethernet* (estándar IEEE 802.3) se realiza a través de un canal que se comparte usando el protocolo CSMA/CD (*Carrier Sense Multiple Access / Collision Detection*). Se permiten hasta 1024 nodos distribuidos en a lo más 2.8 Km. *Ethernet* proporciona un servicio de datagramas (es decir, se envían los mensajes pero no se puede asegurar que lleguen al destino -análogo al servicio normal de telegramas -), por lo que el garantizar una comunicación confiable depende de los protocolos superiores. El tamaño de los mensajes que se transmiten por *Ethernet* es de 64 a 1500 *bytes*.

MODELOS DE REFERENCIA PARA EL PROTOCOLO DE TRANSPORTE

Para poder definir el protocolo de transporte que se implantó en la red local, se analiza más detalladamente la capa de transporte en los dos modelos de referencia que han tenido gran influencia en el diseño de redes de computadoras: OSI y ARPA.

CAPA DE TRANSPORTE DE ACUERDO AL MODELO OSI DE ISO.

Como de vió en el capítulo anterior, el modelo OSI de ISO define una estructura por capas, en la que el objetivo de cada capa es proporcionar servicios a la capa que funciona sobre ella; para lograr su objetivo, cuenta con los servicios que le brinda la capa inmediata inferior.

NOMENCLATURA.

Debido a que ISO usa un modelo por capas para definir su estándar, las estructuras de datos y los servicios relacionados con cada capa se distinguen de las demás, anteponiendo al nombre la inicial (en inglés) de la capa correspondiente. Cada capa se comunica con las capas adyacentes por medio de servicios (o interfaces), que se conocen como SAP's (*Service Acces Points*). Cada SAP tiene asociada una dirección que lo identifica de manera única.

Un usuario se comunica con la capa inmediata inferior a través de un SAP usando el conjunto de primitivas que la capa ofrece. Los protocolos intercambian información a través de estructuras llamadas unidades de datos del protocolo (PDU's por sus siglas en inglés *Protocol Data Units*).

MANEJO DE CONEXIONES

El protocolo que se sigue para establecer una conexión usa las primitivas de comunicación del modelo. Cada capa recibe de la inferior las primitivas y a su vez, ofrece estos servicios a las capas superiores; dichas primitivas de comunicación son:

REQUEST (Petición). Una entidad (elemento activo de la capa correspondiente) pide algún servicio.

INDICATION (Indicación). Se informa a una entidad acerca de algún evento.

RESPONSE (Respuesta). Una entidad desea responder a un evento.

CONFIRM (Confirmación). Se informa a una entidad acerca de la petición que hizo.

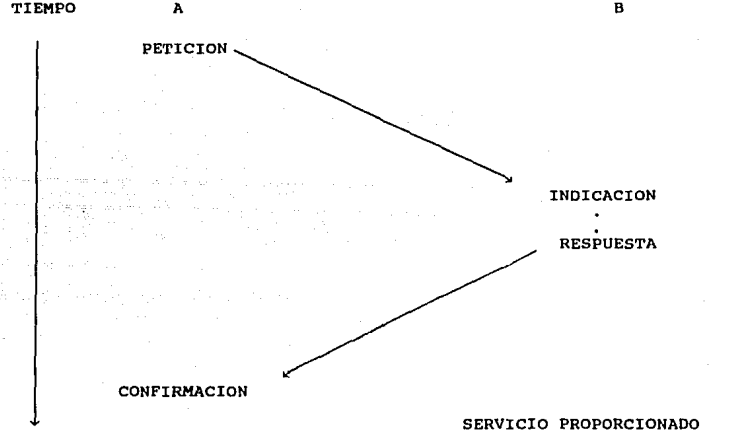
Las entidades usan las primitivas anteriores de la siguiente manera:

1) El primer proceso (A) envía una petición.

2) El segundo proceso (B) recibe una indicación de que se requiere el servicio.

3) Si el segundo proceso (B) puede proporcionar el servicio envía al primero respuesta.

4) El proceso A recibe la confirmación de que se le proporcionará el servicio.



Dentro del protocolo, los servicios pueden proporcionarse o no. En caso de que puedan brindarse, la sucesión de eventos será: petición, indicación, respuesta y confirmación. Si no puede darse el servicio sólo se tendrán los eventos petición e indicación.

Como ejemplo de la manera en que se manejan estas cuatro primitivas, se tienen los siguientes elementos del protocolo para establecer una conexión, intercambiar datos a través de ella y terminarla:

CAPA-CONNECT.INDICATION. Se notifica al usuario que el establecimiento de la comunicación está en progreso.

CAPA-CONNECT.CONFIRMATION. La comunicación ha sido establecida.

CAPA-DATA.INDICATION. Se avisa al usuario que puede leer datos de la comunicación.

CAPA-DISCONNECT.INDICATION. La conexión está cerrada.

CAPA-CONNECT.REQUEST. El usuario indica que desea establecer una conexión.

CAPA-CONNECT.RESPONSE. Se atenderá la petición recibida.

CAPA-DATA.REQUEST. El usuario envía datos.

CAPA-DISCONNECT.REQUEST. El usuario indica que va a terminar la conexión.

Adicionalmente, el protocolo de transporte (capa 4) ofrece los siguientes servicios:

T-EXPEDITED_DATA.INDICATION. Se informa al usuario que puede leer datos expeditos de la conexión.

T-EXPEDITED_DATA.REQUEST. El usuario envía datos expeditos.

Un ejemplo muy claro de la manera en que funcionan estas

primitivas se tiene en el libro de Tanenbaum [Tanenbaum]:

CONNECT.REQUEST. La persona A marca el número telefónico de otra (B).

CONNECT.INDICATION. B escucha sonar el teléfono.

CONNECT.RESPONSE. B descuelga el teléfono.

CONNECT.CONFIRM. A se da cuenta de que dejó de sonar el teléfono.

DATA.REQUEST. A invita a B a comer.

DATA.INDICATION. B oye la invitación que le hace A.

DATA.REQUEST. B acepta la invitación de A y se lo dice.

DATA.INDICATION. A se emociona al escuchar que B acepta.

DISCONNECT.REQUEST. Por la emoción, A cuelga el teléfono sin despedirse (lo importante aquí es que A termina la comunicación).

DISCONNECT.INDICATION. B cuelga el teléfono cuando se da cuenta de que A ya colgó (y a lo mejor decide no acudir a la cita).

La capa de transporte ofrece clases de servicio que dependen de la calidad de servicio que proporcione la red sobre la que se encuentre:

Clase 0. Usada cuando se tiene una red que ofrece una gran calidad de servicio (e.d., es altamente confiable).

Clase 1. Cuenta con recuperación básica de errores.

Clase 2. Permite Multiplexaje.

Clase 3. Con recuperación de errores y multiplexaje.

Clase 4. Detección y recuperación de errores. Usada cuando la red no es confiable.

El protocolo de transporte implantado ofrece un servicio equivalente a la clase 4 de la capa de transporte definida por OSI (es decir, asume que el servicio de red sobre el que funciona no es confiable), por lo que debe llevar a cabo todas las rutinas para detección y correción de errores.

ARPANET

Aún cuando el modelo OSI es ampliamente aceptado, existe un modelo anterior muy importante: la red ARPA (*ARPANET*). La importancia de ARPA radica en que es una red actualmente funcionando, mientras que OSI es un modelo que se espera guíe las futuras implantaciones de redes. Por otro lado, dado que se ha trabajado para que OSI sea un protocolo completo y capaz de adaptarse a diferentes tipos de redes, la especificación ha llegado a ser muy compleja, mientras que los protocolos de ARPA, aunque completos y funcionales, son más sencillos que aquellos que se definen en OSI.

La red ARPA de computadoras surgió en la DARPA (*Defense Advanced Research Projects Agency*) para estudiar y demostrar las ventajas de comunicar y compartir recursos. Su objetivo es permitir el intercambio de información entre máquinas con sistemas y arquitecturas heterogéneas. Los protocolos de DARPA se conocen como TCP/IP, aunque TCP e IP son sólo dos de los protocolos que lo forman. Estos protocolos se desarrollaron para funcionar en redes no confiables y se define además la manera en que pueden comunicarse redes con diferentes características (*Internetworking*); dichas redes se conectan por medio de máquinas que se conocen como *gateways*.

Se plantea tener al menos dos capas: la que se encarga del envío y recepción de los mensajes a través de la red y los *gateways* [1], y la encargada de la conversación a nivel de procesos finales, aunque generalmente existe una capa más a nivel de la capa de aplicación de OSI y la capa que maneja las interfaces con la red. Estas cuatro capas son las que definen el modelo ARPA y la

[1] Computadoras que se usan como puentes de una red a otra. Tienen la capacidad de traducir formatos entre las redes que comunican.

equivalencia con las capas OSI es la siguiente:

7	APLICACION	PROCESO/ APLICACION (FTP / SMTP / TELNET)
6	PRESENTACION	
5	SESION	
4	TRANSPORTE	PROCESO/PROCESO (*TCP* / UDP)
3	RED	IP / ICMP
2	ENLACE DE DATOS	ACCESO A
1	FISICA	LA RED

Las funciones de los principales protocolos que se definen en ARPA son:

IP (*Internet Protocol*): servicio para manejo de datagramas que permite direccionamiento de máquinas, manejo de rutas y en caso necesario, fragmentación y reensamble de datos.

TCP (*Transmission Control Protocol*): transmisión de datos confiable, sin duplicados y con control de flujo de datos. Se maneja una suma de verificación para asegurar la integridad de los datos.

UDP (*User Datagram Protocol*): servicio para manejo de datagramas a nivel de transporte; permite el uso de una suma de verificación (opcional), para validar la integridad de los datos.

ICMP (*Internet Control Message Protocol*): usado para reporte

de errores y para las tareas relacionadas con su manejo; en general, no es accesible al usuario.

Algunos de estos protocolos se discutirán más a detalle posteriormente.

EL PROTOCOLO DE TRANSPORTE

Debido a que el protocolo de transporte que se implanta se define tomando como referencia el protocolo de transporte TCP, es necesario describir los servicios, interfaces y estructuras de datos relacionadas con TCP.

TCP (*Transmission Control Protocol*)

Según Holzmann, para especificar completamente un protocolo debe definirse el servicio que proporciona, las condiciones por omisión del ambiente en el que funciona el protocolo, el vocabulario (tipo) de sus mensajes, el formato de cada mensaje y las reglas que siguen los procedimientos que forman el protocolo para garantizar la consistencia del intercambio de mensajes [Gerard J. Holzmann]. Brevemente se describen algunos de estos puntos; información más detallada se discute a lo largo de este capítulo.

SERVICIO. TCP permite el intercambio confiable de información entre procesos, para lo que se establece un canal de comunicación virtual (conexión). TCP cuenta con funciones para la detección de los errores que ocurran durante la comunicación (datos perdidos, duplicados o que hayan sido alterados); los mecanismos usados para detectar errores son acuses positivos con retransmisión, y asociar una suma de verificación a los datos enviados.

AMBIENTE. Se asume que existen al menos dos procesos (usuarios de TCP) que desean intercambiar información y un medio que permita la comunicación. No puede asegurarse que el medio de comunicación sea confiable, por lo que TCP detecta los errores y lleva a cabo la recuperación.

VOCABULARIO. Se definen 6 tipos de mensajes; los mensajes que se intercambian contienen datos y opcionalmente información de control. Un mensaje es de control si alguna de las siguientes banderas está prendida:

- *SYN (synchronization)* es usado por procedimientos que desean establecer una conexión.
- *ACK (acknowledge)* para acuses de recibo positivos.
- *URG (urgent)* indica que debe efectuarse transferencia urgente.
- *FIN* termina la comunicación en un sentido (de parte del emisor).
- *PSH (push)* para forzar transmisión inmediata.
- *RST (reset)* para terminar una conexión cuando se presentan causas extraordinarias.

FORMATOS. Los mensajes que se intercambian en TCP están formados por encabezado (que contiene información de control) y datos. El formato y los detalles de la información de control se discuten en la sección que trata de la transferencia de datos en TCP.

REGLAS. Uno de los mecanismos formales para describir las reglas que siguen las funciones de TCP, que garantizan la consistencia en el intercambio de mensajes, es el diagrama de transición de estados. En el caso de TCP el diagrama muestra, a través del tiempo, los estados por los que pasa una conexión; los cambios de estado se producen como respuesta a diferentes eventos

(por ejemplo recepción de mensajes con información de control). Toda esta información se analiza detalladamente en la sección que trata del manejo de conexiones.

CARACTERISTICAS GENERALES.

TCP es un protocolo de comunicación por medio del cual se especifican los formatos y acuses de recibo que dos computadoras intercambian para lograr una comunicación confiable; define también las rutinas que deben usarse para asegurar que los datos se transfieran correctamente y para corregir pérdida o duplicación de información y establece la secuencia de pasos que las computadoras deben seguir para poder comunicarse.

En un sistema jerárquico por capas, el protocolo de transporte TCP se ubica de la siguiente manera:

PROTOCOLOS SUPERIORES
T C P
I P
RED

En relación al modelo OSI de ISO, el protocolo TCP se ubica a nivel de los protocolos de transporte (nivel 4), con la característica adicional de ser un protocolo altamente confiable.

Las características básicas de este protocolo son:

Orientación a la conexión (*stream orientation*). La transferencia

de información se ve como flujo de *bytes*. El servicio de transferencia de la máquina destino pasa al receptor exactamente la misma secuencia de octetos que el emisor le pasó en la máquina fuente. La comunicación se lleva a cabo sólo cuando ambas partes están de acuerdo.

Conexión por circuito virtual (*virtual circuit connection*). Para que pueda efectuarse la comunicación debe establecerse una conexión. Durante la transferencia de información, los protocolos de ambas máquinas se comunican para verificar que los datos se reciban correctamente; se dice que es una **conexión de circuito virtual** porque los programas de aplicación ven la conexión como un circuito de *hardware* dedicado, aunque físicamente no es así.

Transferencia por *buffers* (*buffered transfer*). Al transmitir datos, cada aplicación elige el tamaño de paquetes que necesite, aunque el protocolo puede dividirlos según se requiera. Al transmitir, usualmente se juntan varios paquetes en un datagrama con el fin de que la transferencia sea más eficiente y se reduzca el tráfico a través de la red. Normalmente el protocolo de transporte define una área de almacenamiento (*buffer*) para los datos que se transmiten y hasta que dicha área de almacenamiento se llena se procede a la transmisión; sin embargo, se proporcionan servicios para las aplicaciones cuyos datos deben enviarse aún cuando no se haya llenado el *buffer* (existe un servicio para envío normal de datos, pero sin esperar a que el área de almacenamiento se llene, y otro para manejo de datos urgentes).

Transmisión no estructurada (*unstructured stream*). El servicio de transferencia no interpreta la información que se intercambia. Los programas de aplicación deben tener acuerdo previo acerca de cómo interpretar los datos que se transfieren.

Conexión en ambos sentidos (*full-duplex connection*). Las conexiones deben permitir la transferencia concurrente en ambas direcciones. Desde el punto de vista de la aplicación, la conexión *full-duplex* está formada por dos flujos de información independientes que circulan en direcciones opuestas. Se permite que el flujo de información termine en un sentido y continue en el otro.

TCP asume que cuenta con un servicio de comunicación de datagramas (esto implica que el servicio no garantiza que los datos lleguen a su destino), en el que exista la posibilidad de manejar precedencia (transmisión de datos con prioridad) y seguridad (lo que significa enviar los datos eligiendo trayectorias a las que no tienen acceso personas no autorizadas).

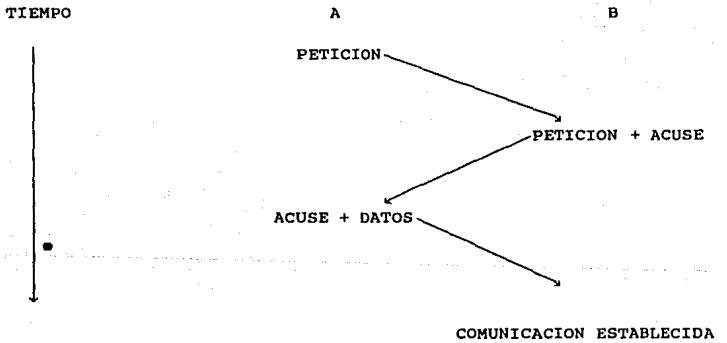
Para que dos procesos puedan establecer comunicación, antes deben ponerse de acuerdo. En TCP se maneja un protocolo de tres pasos:

1) El primer proceso (A) envía una petición de conexión.

2) Si el segundo proceso (B) acepta la comunicación, envía al primero una petición de conexión y el acuse correspondiente al paquete con que recibió la petición

3) El primer proceso envía al segundo el acuse del paquete recibido y opcionalmente, puede empezar a transmitir los datos.

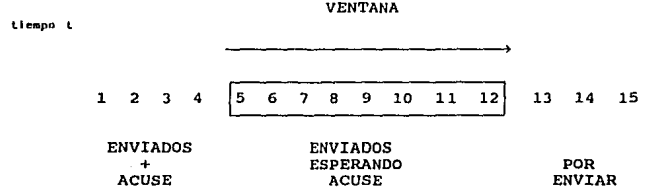
Gráficamente el protocolo se vería así:



Para que se establezca la comunicación se requiere necesariamente de tres pasos, por lo que este método se conoce como **acuerdo en tres pasos** (*three-way handshake*). El procedimiento para establecer la comunicación debe funcionar también en el caso de que las dos partes realizen la petición simultáneamente.

Ya que se establece la comunicación, cada vez que se reciben datos, se regresa un acuse de recibo (*acknowledgement*). El emisor mantiene un registro de cada paquete que envía y espera el acuse correspondiente antes de enviar el siguiente paquete; al momento de enviar se inicia un cronómetro, para retransmitir el paquete en caso de que el tiempo límite (*timeout*) se agote y aún no se reciba el acuse. Esta técnica se conoce como **acuse positivo con retransmisión** (*positive acknowledgement with retransmission*). Para poder manejar los casos en que haya errores o retrasos en los paquetes que circulan por la red, cada petición y cada acuse

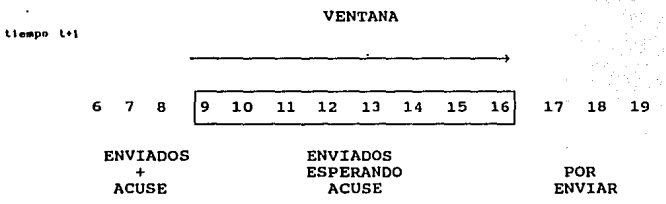
incluye un número de secuencia único. Con el fin de aprovechar al máximo los recursos de la red se envían varios paquetes a la vez, sin esperar los acuses de recibo correspondientes. Al usar esta técnica (ventanas corredizas, o en inglés *sliding windows*), se mantiene un registro (ventana) de los paquetes enviados para los que aún no se ha recibido el acuse correspondiente. Cuando se envía el acuse de recibo, el protocolo sólo transmite un número de secuencia, lo que se interpreta como "se han recibido todos los paquetes con número de secuencia menor al que te envío". El usar esta técnica permite determinar fácilmente el número de secuencia que la otra parte espera recibir la siguiente ocasión (de hecho, es el número que él transmite). El número de paquetes que pueden estar sin acuse en cada momento está especificado por el tamaño de la ventana; esto quiere decir que el emisor está autorizado a enviar, antes de recibir un acuse, tantos paquetes como lo indique el tamaño de la ventana.



Tamaño de la ventana = 8

Una vez que el emisor recibe el acuse del segmento de datos que envió, corre la ventana tantos lugares como lo permita el acuse, y procede a enviar los datos que quedan dentro de la ventana. Por ejemplo, supóngase que se recibe un acuse con número 9, por lo tanto se recorre la ventana hasta tener el paquete con número de secuencia 9 al inicio y se procede a enviar los paquetes de datos que se añadieron a la ventana; con esto, la ventana que

se presentó en el esquema anterior se modificaría de la siguiente forma:



De esta manera, en todo momento, a la izquierda de la ventana se encuentran los paquetes que ya se enviaron y para los que se recibió el acuse, dentro de la ventana están los paquetes enviados pero para los que aún no llega el acuse y a la derecha están los paquetes por enviar.

La efectividad de las ventanas corredizas depende del tamaño de la ventana y la velocidad a la que la red hace la transferencia de información. La ventana corrediza debe mantener registros acerca de los paquetes para los que se recibió el acuse, y cronómetros separados para proceder a la retransmisión en caso necesario.

El aspecto de duplicación y/o pérdida de paquetes se maneja también por medio de los números de secuencia. A cada paquete que se transmite se asocia un número, y cuando el receptor envía el acuse de recibo correspondiente especifica de qué paquete se trata. De esta manera, si se reciben datos duplicados se detectan fácilmente al tener varios paquetes con el mismo número de secuencia; en este caso, se envía el acuse correspondiente (para evitar seguir recibiendo paquetes duplicados) y se descartan los

datos. La detección de datos perdidos ocurre al intentar ordenar los datos por el número de secuencia; en este caso, simplemente no se envía acuse de recibo y al transcurrir el tiempo de espera, la otra parte retransmite los datos.

Los números de secuencia se generan de manera cíclica: a los primeros paquetes se asigna el número 1, 2, ..., max; al llegar el siguiente paquete se le asocia el número 1, por lo que debe asegurarse que al enviar este último paquete, el anterior que tenía el mismo número de secuencia ya no esté circulando por la red. Esto puede lograrse teniendo un ciclo "suficientemente grande". El tamaño del ciclo normalmente se determina por medio de análisis estadísticos, en los que se analiza el tráfico de los paquetes en la red; la capacidad de respuesta de la red se ve afectada por factores como el tamaño de los paquetes, las características de la red (por ejemplo qué tan confiable es) y la cantidad de paquetes que se estén transmitiendo a través de ella. Todo lo anterior debe considerarse para elegir el tamaño del ciclo, de la ventana y de los paquetes.

INTERFAZ USUARIO-TCP.

Como toda capa que forma una red, TCP proporciona servicios a los procesos de las capas superiores. El usuario de TCP tiene acceso a estos servicios utilizando llamadas a funciones. Usar estas funciones es muy sencillo, pues son parecidas a las que se utilizan para manejo de archivos, sólo que en este caso las funciones de TCP se usan sobre conexiones. Existen funciones para abrir y cerrar una conexión, y para enviar y recibir datos a través de una conexión establecida; además hay dos funciones de control: una de ellas reporta el estado de la conexión y la otra aborta la comunicación.

Para poder manejar las conexiones, se utiliza una estructura

en la que se almacena información de control (el estado de la conexión, número inicial de secuencia para envío, número inicial de secuencia para recepción, etc); a dicha estructura de control se le llama **TCB** (por sus siglas en inglés *Transmission Control Block*).

Las rutinas que se usan para manejo de conexiones son:

open_tcp (ID_USUARIO_LOCAL, ID_USUARIO_REMOTO, ACTIVA, VENTANA, *R_UPCALL*, *T_UPCALL*, *S_UPCALL*, *TOS*).

Usada para iniciar (abrir) una conexión; regresa un apuntador a un bloque de control de transmisión (*TCB*). ID_USUARIO_LOCAL es la identificación del usuario local de **TCP** (esta identificación es única); ID_USUARIO_REMOTO es la manera de identificar de manera única al usuario remoto de **TCP**; ACTIVA indica si la conexión se abre para intercambiar datos inmediatamente (comunicación activa), o si sólo se está avisando que el proceso está dispuesto a establecer comunicación con quien se lo pida (comunicación pasiva; este es el tipo de conexión usada por los servidores); VENTANA es el tamaño de los *buffers* de recepción y transmisión; *R_UPCALL* es la función que se llama cuando llegan datos (*receive upcall*); *T_UPCALL* cuando puede efectuarse una transmisión (*transmit upcall*); *S_UPCALL* cuando la conexión cambia de estado (*state upcall*) y *TOS* define el tipo de servicio de internet que se usará (*Type Of Service*).

send_tcp (*TCB*, *BP*).

Envía datos a través de la conexión especificada. *TCB* es el apuntador al bloque de control de transmisión regresado por **open_tcp**; *BP* (*buffer pointer*) apunta a la estructura de datos donde se encuentra la información del usuario que se transmitirá.

recv_tcp (*TCB*, *BP*, CONT).

Recibe datos de la conexión indicada. *BP* apunta a la información recibida, que a lo más tiene tamaño de CONT *bytes*. *TCB* como antes apunta al bloque de control de transmisión.

close_tcp (*TCB*).

Con *close* el usuario local avisa que no tiene más datos que transmitir por la conexión especificada por el *TCB*, aunque puede seguir recibiendo datos.

del_tcp (*TCB*).

Termina la conexión asociada al *TCB* especificado. Previamente debe haberse ejecutado la rutina **close_tcp** en ambos lados y los datos pendientes ya deben haberse leído.

state_tcp (*TCB*).

Reporta el estado de la conexión indicada por el *TCB*.

INTERFAZ TCP-RED.

TCP asume que la capa inmediata inferior le proporciona servicios para enviar y recibir mensajes a través de la red. En el caso de que el protocolo de red sobre el que está construido **TCP** sea IP, se cuenta con argumentos para especificar el tipo de servicio y el tiempo de vida de los paquetes dentro de la red

Dentro de los aspectos que cubre el tipo de servicio se encuentran:

prioridad

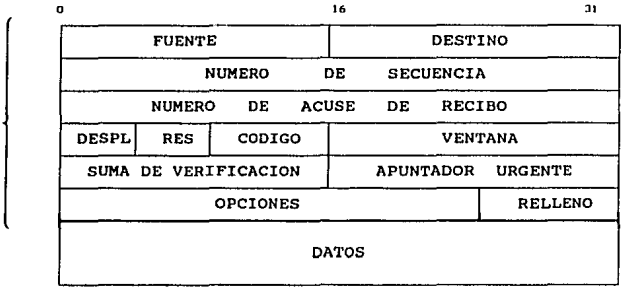
retraso máximo permitido
confiabilidad
capacidad de respuesta esperada

En el capítulo tres se discuten de manera más detallada las funciones y parámetros que definen esta interfaz.

TRANSFERENCIA DE DATOS.

La transferencia de información a nivel de **TCP** se hace por medio de unidades llamadas **segmentos.** Dichos segmentos se usan para manejar las conexiones, transferir datos y enviar acuses de recibo.

El formato de los segmentos de **TCP** es el siguiente:



Al transmitir, cada segmento se divide en dos partes: encabezado y datos.

Los campos FUENTE y DESTINO del encabezado identifican a los programas de aplicación que están comunicándose. El NUMERO DE SECUENCIA indica la posición, en el flujo de datos, de la información enviada en el segmento. El NUMERO DE ACUSE DE RECIBO señala la posición del *byte* de mayor orden que se ha recibido. Esto se maneja así debido a que TCP envía datos en segmentos de longitud variable, por lo que en un segmento la cantidad de *bytes* enviados es variable; con ésto se tiene que los acuses de recibo especifican el número del siguiente *byte* que el receptor espera. DESPL contiene el desplazamiento de la porción de datos del segmento; este valor es necesario, pues el tamaño del campo de opciones es variable. RES es un campo reservado "para usarse en el futuro" (según el RFC 793 que es donde se especifica oficialmente el protocolo de transporte TCP). El campo CODIGO se usa para definir el tipo de segmento de que se trata, por medio de banderas de control. El campo VENTANA sirve para indicar la cantidad de *bytes* que pueden intercambiar los procesos en un momento dado. La SUMA DE VERIFICACION contiene un entero de 16 *bits* usado para asegurar la integridad del encabezado y los datos. Para calcular dicha suma, el TCP emisor usa aritmética de 16 *bits* y el complemento a uno de la suma en complemento a uno de los datos contenidos en el segmento; el TCP receptor realiza el mismo cálculo para verificar que los datos recibidos sean correctos. Con el APUNTADOR URGENTE y una bandera de control en el campo CODIGO, se indica al TCP emisor que los datos deben enviarse tan pronto como sea posible y al TCP receptor que avise al programa destino que entra en modo de recepción urgente. Tan pronto como se termine la comunicación urgente, el programa de aplicación regresa al modo normal de operación. El campo de OPCIONES se usa para comunicaciones específicas entre los TCP's de ambas máquinas; en particular puede servir para definir el tamaño máximo del segmento. El campo RELLENO se usa si el número de *bits* del segmento no es un múltiplo de 16, en cuyo caso este campo se llena con tantos ceros como se necesite.

MANEJO DE CONEXIONES

Una conexión pasa por varios estados durante su tiempo de existencia. Dichos estados son:

Listen. Se espera una petición de conexión de algún anfitrión remoto.

Syn-sent. Espera por la respuesta (confirmación de que la otra parte acepta establecer la comunicación), después de enviar una solicitud de conexión.

Syn-received. Se espera el acuse de recibo, después de que ambas entidades han enviado las solicitudes de conexión y recibido la confirmación de que se acepta la comunicación (de hecho, el acuse es para la última confirmación enviada).

Established. Indica una conexión abierta.

Fin-wait-1. Espera una petición de terminación del anfitrión remoto o un acuse para la señal de terminación previamente enviada.

Fin-wait-2. Se espera una petición de terminar la conexión por parte del anfitrión remoto.

Close-wait. Espera una petición de terminación por parte del usuario local.

Closing. Espera el acuse del anfitrión remoto de la petición

de terminación hecha.

Last-ack. Espera el acuse de la petición de terminación de conexión que se envió al TCP remoto.

Time-wait. Espera a que pase la suficiente cantidad de tiempo para asegurar que el TCP remoto recibió el acuse de su petición de terminación.

Closed. No hay conexión

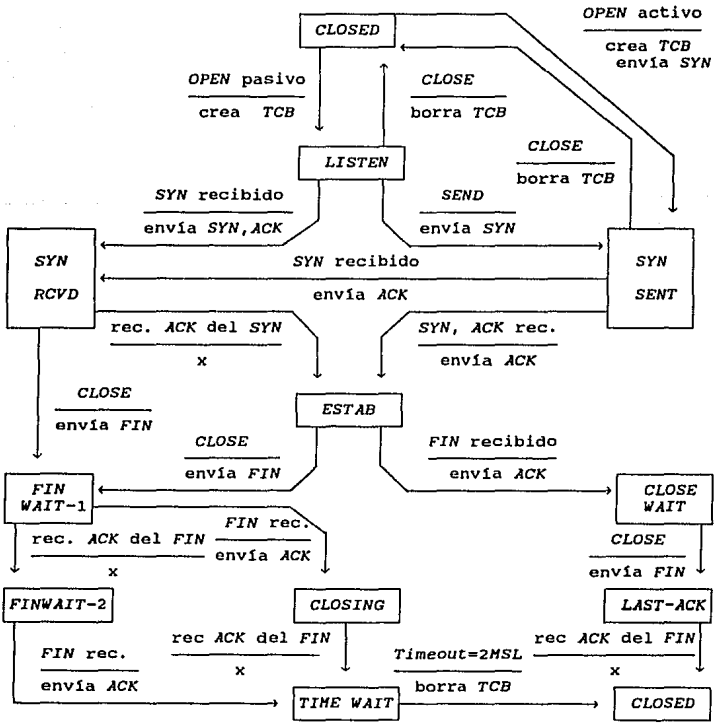
Una conexión TCP cambia de un estado a otro como respuesta a eventos formados por las llamadas de usuario (*OPEN*, *SEND*, *RECEIVE*, *CLOSE*, *ABORT* y *STATUS*), por la llegada de paquetes (en particular los que llevan señales de control) y al transcurrir los tiempos de espera.

Las señales de control se incluyen en el campo CODIGO del encabezado, y se usan para especificar que el segmento es un acuse de recibo (*ACK -acknowledge*) o que debe efectuarse transferencia urgente (*URG*) de datos. La opción de control para sincronización (*SYN -synchronization*) es usada por procedimientos que desean establecer una conexión y la de fin (*FIN*) para indicar que el emisor ha terminado su transferencia de datos. La opción (*PSH -push*) se usa para indicar que los datos deben enviarse al receptor inmediatamente; la opción para reinicio (*RST -reset*) se utiliza cuando debe terminarse una conexión por causas extraordinarias. Cuando el TCP receptor encuentra esta opción, aborta la conexión y avisa al programa de aplicación; al emplear esta opción se termina la conexión en ambos sentidos y los recursos que se estaban usando se liberan.

La manera en que ocurren los cambios de estado, las señales que se reciben y los datos o señales que se envían como respuesta,

se ilustran en el siguiente diagrama (tomado del RFC 793):

DIAGRAMA DE TRANSICION DE ESTADOS PARA TCP



Para establecer una conexión se usa el siguiente protocolo: primero, se envía un mensaje con la opción *SYN*. El mensaje de repuesta debe llevar las opciones *ACK* para el mensaje recibido y *SYN* para indicar que está dispuesto a establecer la conexión. El último mensaje que se envía dentro de este protocolo es el acuse de recibo correspondiente, con lo que se completan los tres pasos del protocolo.

En resumen, la comunicación entre entidades de TCP se lleva a cabo en ambas direcciones (comunicación *full-duplex*); al momento de terminar una conexión, se maneja el concepto de que el proceso que realiza esta petición en realidad termina la comunicación en un sentido, es decir, el proceso que desea cerrar la conexión avisa que no tiene más datos que transmitir pero puede seguir recibiendo datos. Una vez que se ha establecido la conexión, el intercambio de datos se hace a través de segmentos. Para asegurar que la transmisión es confiable, se implanta una técnica para retransmisión de los mensajes en caso de que la suma de verificación no concuerde o de que los mensajes se pierdan al viajar por la red. Antes de proceder a la retransmisión, se espera un tiempo fijo, calculado de tal manera que en el peor de los casos, el mensaje llegue a su destino. Si este tiempo (*timeout*) transcurre y no se ha recibido el acuse correspondiente, se lleva a cabo la retransmisión. Para el manejo de paquetes duplicados, TCP utiliza los números de secuencia de los segmentos. En caso de que lleguen paquetes duplicados, el protocolo de transporte envía el acuse correspondiente al paquete recibido (para que el TCP emisor no vuelva a enviarlo) y deshecha el segmento.

La implantación del protocolo de transporte se apega a la definición de TCP tanto como es posible. En el capítulo siguiente (que describe la implantación del protocolo y las decisiones de diseño) se especifican las diferencias entre el protocolo de transporte implantado y TCP.

BIBLIOGRAFIA

ETHERNET

- *The ethernet: a local area network; data link layer and physical layer specifications* V 1.0
  Digital equipment corp., Intel corp. and Xerox corp. [DIX]
  Comput. Commun. Rev. 11(3)
  sep. 1980 pp. 20-66

Definición oficial de *ethernet*. Es indispensable leerla si se va a trabajar con *ethernet*.

TRANSPORTE

- *Features of the transport and session protocols*
  J. Burrus [Burrus]
  National Bureau of standards
  Report No. ICST/HLNP-80-1
  marzo de 1980 pp. 1-70

Descripción bastante detallada de las características que deben tener los protocolos de las capas correspondientes a transporte y sesión.

- *ISO Transport Protocol Specification ISO DP* 8073
  ISO [ISO]
  RFC 905
  abril de 1984

Especificación detallada de las cinco clases de servicio de transporte de OSI, sus funciones y procedimientos asociados, de la estructura y codificación de las unidades de datos (TPDU's) y funciones para control de flujo y errores y manejo de tiempos de espera.

- *ISO Transport Service on top of the TCP*
  Marshall T. Rose & Dwight E. Cass [Rose]
  RFC 1006
  mayo de 1987

Define el estándar a seguir para implantar la capa de transporte de OSI sobre máquinas que trabajan con TCP.

- *Implementation Guide for the ISO Transport Protocol*
  Wayne McCoy [McCoy]
  RFC 1008
  junio de 1987

Define el protocolo de transporte de OSI y señala los aspectos relevantes que deben cuidarse al implantarlo, pero deja muchas decisiones de diseño sin especificar para permitir flexibilidad a los programadores. Incluye código de la suma de verificación en el lenguaje de programación 'C' y en ensamblador.

- *Service specification and protocol construction for the transport layer*
  Murphy, S. & Shankar, A. [Murphy]
  Comput. Commun. Rev. 18(4)
  agosto de 1988 pp. 88-97

Proporciona una especificación formal del servicio de transporte en un sistema que funciona por medio de eventos.

- *A primer: understanding transport protocols*
  W. Stallings (editor) [Stallings]
  Data communications
  noviembre de 1984 pp. 201-215

Buen artículo introductorio. Describe servicios básicos y mecanismos de control del protocolo de transporte (conexiones, *three way handshake*, control de flujo y errores, etc). Hace un análisis interesante de TP4.

- *Transport protocols for computer networks*
  C. A. Sunshine [Sunshine]
  Protocols and techniques for data communications networks
  Ed F. Kuo
  1981 pp. 35-77

Descripción informal de las características de un protocolo de transporte que proporcione comunicación confiable, de propósito general.

ARPA

- *The DOD Internet Architecture Model*
  V. G. Cerf & E. Cain [Cerf]
  Computer Networks V. 7
  1983 pp. 307-318

Plantea los principios en los que se basa la arquitectura de Internet; analiza algunos de los protocolos por medio de los que se implanta.

- *The design and philosophy of the DARPA internet protocols*
  Clark, D. [Clark]
  Comput. Commun. Rev. 18(4)
  agosto de 1988 pp. 106-114

Analiza el punto de vista de quienes definieron los protocolos de Internet, sus objetivos, arquitectura e implantación.

- *The DARPA internet: interconnecting heterogeneous computer networks with gateways*
  Robert Hinden, Jack Haverty & Alan Sheltzer [Hinden]
  Computer
  sep. de 1983

Describe las experiencias, decisiones de diseño y los problemas del grupo que desarrolló el modelo INTERNET, para comunicar redes heterogéneas de computadoras.

- *Internetworking with TCP/IP* (Vol. 1 y 2)
  principles, protocols and architecture
  Douglas Comer [Comer-1] [Comer-2]
  Ed. Prentice Hall
  1991

Muy buenos libros de TCP/IP. Van de definiciones sencillas a cuestiones técnicas completas y complejas, pero muy bien explicadas. Cubren los protocolos ARP, IP, ICMP, UDP, TCP, RIP y SNMP; explican muy bien la interfaz a nivel de *sockets*.

- *Transmission control protocol*
  DARPA Internet Program Protocol specification
  RFC 793 [RFC 793]
  sep. de 1981

Especificación oficial de TCP. Describe las funciones de TCP, los aspectos más importantes que debe cubrir el programa que lo implante y sus interfaces hacia las capas adyacentes. Todo aquel que desee saber sobre TCP debe leer este documento.

- *TCP on a LAN*
  M. A. Padlipsky
  RFC 872 [RFC 872]
  septiembre de 1982

Describe brevemente lo que es una LAN, sus diferentes topologías y TCP; realza las características de TCP que lo hacen adecuado para las LAN's.

- *A TCP/IP Tutorial*
  T. Socolofsky & C. Kale
  RFC 1180 [RFC 1180]
  enero de 1991

Explica las características relevantes de TCP/IP, enfatizando los aspectos técnicos.

3. IMPLANTACION DEL PROTOCOLO

La implantación del protocolo de transporte se realiza sobre una red local planeada como un modelo jerárquico por capas, que se comunica a través de *ethernet*. Sobre *ethernet* se encuentra el protocolo de *Internet* (*IP- Internet Protocol*), y sobre éste el protocolo de transporte.

Con esto, la relación entre las diferentes capas de la red es la siguiente:

PROGRAMAS DE APLICACION
PROTOCOLO DE TRANSPORTE
INTERNET PROTOCOL
ETHERNET

SERVICIOS REQUERIDOS.

El protocolo de transporte asume que cuenta con los servicios de red básicos para envío y recepción de datagramas (esto es, se envían los mensajes, pero no puede asegurarse que lleguen a su destino, por lo que el servicio de comunicación no es confiable); además debe existir la posibilidad de que el usuario defina la prioridad de sus mensajes y la seguridad en la comunicación (como se dijo anteriormente, el manejo de la seguridad es en el sentido de que la transmisión se lleve a cabo a través de rutas a las que no tengan acceso personas no autorizadas). Todas estas

características se logran en la red local a través de IP.

La transmisión de mensajes en una red formada por capas jerárquicas se lleva a cabo de la siguiente manera: El usuario envía a la capa de transporte un paquete de datos:

DATOS

El protocolo de transporte obtiene una copia de los datos enviados por el usuario y decide si caben en uno solo de los paquetes de datos que él maneja. En caso contrario, dividirá la copia de datos del usuario y enviará los segmentos obtenidos por medio de varios paquetes. El protocolo de transporte que recibe los paquetes los volverá a unir para entregar al usuario destino el mensaje original que se le mandó. Para que el protocolo de transporte pueda conservar el orden original de los datos y asegurar su integridad, a cada paquete que envía le añade un encabezado con información de control, con lo que el segmento enviado por el protocolo de transporte es el siguiente:

ENCAB. TRANS.	DATOS

Este segmento se pasa a IP, quien genera una copia del segmento, y le añade información propia de control; nuevamente, en caso de que el tamaño de los datos recibidos sea mayor que el de los paquetes de datos que maneja IP, el segmento de datos será dividido. El paquete de datos enviado por IP tiene el siguiente aspecto:

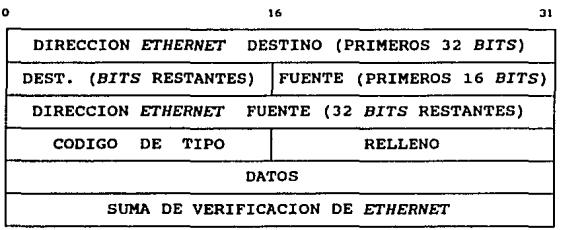
ENCAB. IP	ENCAB. TRANS.	DATOS

Finalmente, los datos son enviados a *ethernet*, quien sigue los mismos pasos que los protocolos anteriores y además acota los

paquetes de datos agregándoles su información de control por medio de un encabezado y una suma de verificación, para asegurar la integridad de la información, que sirve como delimitador final. Con esto, los datos que circulan a través de la red tienen el siguiente formato:

ENCAB. ETHER.	ENCAB. IP	ENCAB. TRANS.	DATOS	SUMA VERIF.

De acuerdo al estándar DIX (*Digital Intel Xerox*), la estructura de los paquetes que circulan por la red es (las direcciones en *ethernet* están formadas por 48 *bits*):



donde la parte de datos para *ethernet* está formada por la siguiente información:

ENCABEZADO PROTOCOLO *INTERNET*
ENCABEZADO PROTOCOLO DE TRANSPORTE
DATOS

El código de tipo sirve para identificar a la familia de

protocolos a que pertenece este paquete (TCP/IP, DECnet, etc).

Uno de los aspectos importantes de esta implantación es que para transmitir los datos de un nivel a otro no se hacen copias del paquete, sino que se manejan apuntadores a las estructuras de datos, con lo que el manejo de recursos es más eficiente.

La discusión a detalle de la manera en que se transmiten los mensajes a través de *Ethernet* (fragmentación, direccionamiento, elección de rutas, etc.) puede encontrarse en [RFC 894] y en [RFC 826].

PROTOCOLO DE INTERNET (*INTERNET PROTOCOL*)

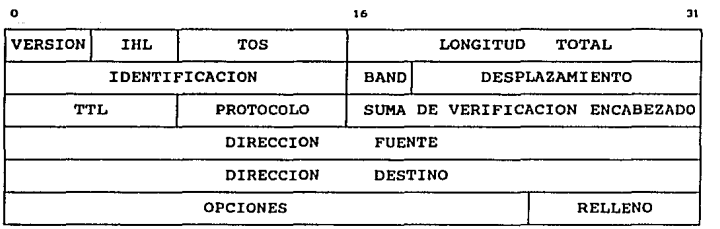
Debido a que el protocolo de transporte interactúa con IP, antes de hablar de la implantación es necesario conocer la manera en que se intercambian mensajes a través de IP.

El objetivo de IP es transmitir paquetes de datos (datagramas) a través de un conjunto interconectado de redes. Esto lo realiza transmitiendo los datagramas de una máquina a otra, siguiendo una ruta determinada, hasta llegar a la máquina destino. IP se desarrolló bajo la filosofía de que debe pedirse sólo lo indispensable a los servicios proporcionados por la red; brinda un servicio simple de datagramas que permite conectar redes completas a través de máquinas que se conocen como *gateways*.

IP permite la transmisión de datagramas de máquinas fuentes a destinos; las máquinas fuente y destino se identifican por medio de direcciones de longitud fija (32 *bits*) que se asignan de manera global; con esta información, IP realiza la elección de la ruta que deben seguir los paquetes para llegar a su destino, es decir, las direcciones indican dónde está la máquina a la que se desea hacer llegar los datos y dónde está la máquina que los envió y la

ruta dice cómo llegar de la máquina fuente a la máquina destino. IP tiene la capacidad de segmentar un datagrama y reensamblarlo para poder funcionar en redes que tengan paquetes de datos "pequeños". No hay funciones que lo hagan confiable (por ejemplo, funciones para controlar el flujo de datos o manejar los datagramas por medio de números de secuencia) ni hay conexiones (cada datagrama enviado es independiente).

El formato del encabezado de los datagramas que se transmiten a través de IP es el siguiente:



El campo VERSION indica el formato del encabezado internet; el campo IHL (*Internet Header Length*) es la longitud del encabezado en palabras de 32 *bits*.

El usuario de IP puede especificar la calidad de servicio que desea por medio de un conjunto de parámetros: tipo de servicio (*Type of service* -TOS-), tiempo de vida (*Time to Live* -TTL-) y OPCIONES (*Options*). El tipo de servicio (TOS) es un conjunto de parámetros que caracterizan el servicio proporcionado en la red y es usado por las máquinas que funcionan como puntos intermedios en

la ruta que siguen los datagramas, para elegir los parámetros de la transmisión actual, la red por la que conviene seguir transmitiendo, etc. Estos parámetros cubren aspectos como:

- Precedencia (tratar la transmisión como normal, prioritaria, inmediata, etc.).
- Retraso aceptado (normal o bajo)
- Capacidad de respuesta esperada (normal o alta)
- Confiabilidad (normal o alta)

La LONGITUD TOTAL indica la longitud del datagrama en octetos, incluyendo el encabezado y los datos; normalmente se transmiten datagramas de a lo más 576 octetos, lo que permite un encabezado de 64 octetos y datos de 512. La IDENTIFICACION es definida por quien envía y se usa para reensamblar los fragmentos del datagrama, en caso de que haya sido necesario dividirlo para poder efectuar la transmisión. El campo BAND (banderas *o* en inglés *flags*) se usa para especificar valores de control como si puede fragmentarse el datagrama y si hay más fragmentos o éste es el último. DESPLAZAMIENTO indica donde empiezan los datos en este datagrama.

Para evitar congestionar la red con paquetes que pudieran llegar a transmisiones en "círculo" sin poder llegar a su destino, se define el tiempo de vida (TTL -*Time To Live*-). El tiempo de vida indica el tiempo máximo que puede circular un datagrama a través de la red. Si algún paquete ha sido transmitido por la red durante su tiempo de vida sin llegar a su destino, será destruido. El tiempo de vida original es decrementado por cada máquina por la que pasa el datagrama, hasta que llega a su destino o se convierte en cero, momento en que la máquina en la que se encuentra lo destruye. La unidad en la que se mide el tiempo de vida es segundos (formalmente hablando, el tiempo de vida indica el número máximo de máquinas por las que puede pasar el datagrama, ya que independientemente del tiempo que tarde el datagrama en llegar a una máquina, si ésta no es la máquina destino, decrementa el

tiempo de vida en una unidad).

El campo PROTOCOLO indica el protocolo del siguiente nivel (por ejemplo ICMP, TCP, GGP *-Gateway to Gateway Protocol-*, etc.) al que va dirigido el datagrama. El objetivo de la SUMA DE VERIFICACION es análogo al que se describió para TCP y el algoritmo que se sigue en IP es el mismo. Las OPCIONES cubren aspectos como seguridad y manejo de rutas especiales. Como en TCP, el RELLENO se usa para asegurar que el encabezado está formado por un múltiplo de 32 *bits*.

INTERFAZ USUARIO-IP

La interfaz usuario-IP se realiza por medio de llamadas a las siguientes funciones:

ip_send (FUENTE, DESTINO, PROTOCOLO, TOS, TTL, APTBUF, LONG, ID, DF).

donde, FUENTE y DESTINO son las direcciones de las máquinas que están intercambiando datos; PROTOCOLO es el protocolo de la capa superior que está usando a IP; TOS es el tipo de servicio solicitado; TTL es el tiempo de vida para los datagramas; APTBUF es un apuntador al *buffer* que contiene los datos que se envían, LONG es el tamaño del *buffer*; ID es la identificación para los paquetes en que se divida del datagrama, en caso de que sea necesario, y sirve para identificar a todos los fragmentos que pertenecen a un mismo datagrama; DF (*Don't Fragment*) indica si el datagrama puede fragmentarse o no.

ip_recv (APTBUF, PROTOCOLO, FUENTE, DESTINO, TOS, LONG, OPCIONES)

APTBUF es el apuntador al lugar donde se guardan los datos que se reciben; PROTOCOLO indica a que protocolo va dirigido el datagrama; FUENTE y DESTINO determinan quiénes se están comunicando; TOS, como antes, representa el tipo de servicio y LONG es la longitud de los datos recibidos. En el caso de la red local con la que se trabaja, se maneja la opción de determinar si el datagrama que llega se transmitió en modo de difusión (es decir, se envía a todas las máquinas conectadas a la red, independientemente de quién se trate), a través de una opción que se llama recep_x_difusión (recepción por difusión). Esto queda definido en la función añadiendo a la parte de OPCIONES recep_x_difusión.

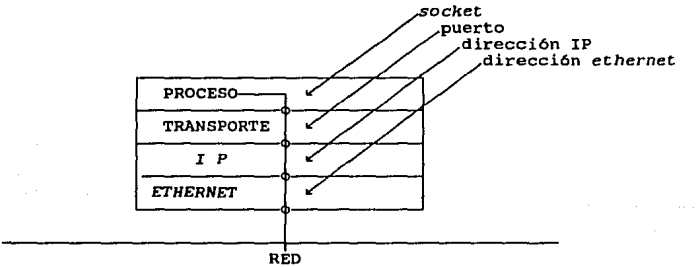
Información adicional de IP puede hallarse en [RFC 791].

EL PROTOCOLO DE TRANSPORTE.

Antes de continuar es importante definir terminología del protocolo de transporte. El protocolo debe permitir la comunicación de cualquier pareja de procesos (o usuarios), por lo que se necesita identificar de manera única a las entidades involucradas en la comunicación. Para lograr ésto, se definen los *sockets* y los puertos (*ports*).

Un *socket* es una entidad direccionable (*internet*) que se encuentra como parte final de ambos lados de la conexión para la comunicación entre procesos. En el sistema operativo UNIX, los *sockets* se identifican por medio de descriptores semejantes a los que se usan en el lenguaje de programación 'C' para manejo de archivos.

Un **puerto** es una dirección dentro de un anfitrión que se usa para diferenciar entre varios *sockets* con la misma dirección *internet*.



El protocolo brinda sus servicios a los procesos por medio de *sockets*. En un anfitrión, existen varios tipos de *sockets*: los definidos por el usuario y los que proporcionan servicios relacionados con el sistema. Debido a que los procesos que desean usar los servicios deben saber a qué *socket* referirse, existen los *sockets* "bien conocidos" (*Well-Known sockets*), que tienen asociada una dirección fija (normalmente entre 0 y 255).

Un *socket* se define por medio de la dirección de la máquina anfitrión (que es un entero de 32 *bits*) y el puerto asociado (16 *bits*). Las conexiones en el protocolo de transporte se identifican de manera única por medio del *socket* local y el *socket* remoto.

Para la definición e implantación del protocolo de transporte, se analizó el RFC 793 (*Transmission Control Protocol*). Se usa notación similar a la del RFC, tomando de él las características relevantes del protocolo de transporte. A partir del RFC, se definen los algoritmos para las funciones que se

proporcionan al usuario para manejo de conexiones (*open, send, recv, close, status* y *abort*) y el algoritmo correspondiente al manejo de eventos (segmentos que llegan, llamadas de usuario y término de los tiempos de espera).

INTERFAZ USUARIO-PROTOCOLO DE TRANSPORTE

De la misma manera que se lleva a cabo en TCP, la interfaz usuario-protocolo de transporte está definida por medio de funciones para manejo de conexiones. Los nombres y parámetros de dichas funciones son los que se especifican en [RFC 793] para TCP.

Como ejemplo se muestra el algoritmo para abrir una conexión. Al ejecutar esta llamada, el usuario debe indicar si la conexión se abre en modo pasivo o activo. Si la conexión se abre de manera activa debe especificarse con quién se desea establecer comunicación.

ALGORITMO PARA LA FUNCION *OPEN*

Datos de ENTRADA: *socket* local, *socket* remoto, modo, ventana, *tos*, funciones: *r-upcall*, *t-upcall*, *s-upcall*, estado de la conexión.
Dato de SALIDA : apuntador al *TCB*

```
si estado = CLOSED
  crea TCB
  llena (TCB, modo)     /* modo = pasivo o modo = activo*/
  si NO correctas (seguridad, precedencia) entonces
    error ("Precedencia o seguridad no permitida")
    termina
  si modo = pasivo entonces
    estado = LISTEN
    termina
  si no,      /* modo = activo */
    si NO especificado (socket remoto) entonces
      error ("Socket remoto no especificado")
      termina
```

```
      si no,    /* socket remoto especificado */
        ISE=elige (ISS)  /* Número Inicial de Secuencia Envío */
        envía_segmento (control= SYN, SEQ=ISE)  /* envía segmento
          con la bandera de control SYN y número de secuencia

          igual al número inicial de secuencia de envío (Initial
          Send Sequence) */
         SND.UNA = ISE    /* Variable para identificar los números
          de secuencia para los que aún no ha llegado el acuse de
          recibo */
         SND.NXT = ISE + 1     /* Número de secuencia del siguiente
          segmento que se enviará */
         estado = SYN-SENT
         termina
si no,        /* el estado no es CLOSED */
  si estado = LISTEN entonces
    si (modo=activo) Y (especificado (socket remoto)) entonces
      ISE = elige (ISS)
      envía_segmento (control= SYN, SEQ=ISE)
      SND.UNA = ISE
      SND.NXT = ISE + 1
      estado = SYN-SENT
      termina
  si no,      /* El estado no es CLOSED ni LISTEN */
    en caso de que estado sea
        SYN-SENT,
        SYN-RECEIVED,
        ESTABLISHED,
        FIN-WAIT-1,
        FIN-WAIT-2,
        CLOSE-WAIT,
        CLOSING,
        LAST-ACK,
        TIME-WAIT : error ("La conexión ya existe")
                    termina
```

Con esta manera de definir las funciones, la interfaz al usuario queda establecida de acuerdo a [RFC 793], brindando servicios por medio de las funciones *open*, *close*, *send*, *recv status* y *abort*.

Una vez definidos los algoritmos para las funciones que se ofrecen al usuario, se procedió a la codificación. La implantación del protocolo de transporte se llevó a cabo en el lenguaje de programación 'C' y bajo el sistema operativo UNIX (versión HP-UX 7.0) en una HP 9000/840. Como ejemplo, se muestra el código generado para la función *OPEN:*

```
/* Función para abrir una conexión.
 * Si la solicitud se hace en modo activo, se espera que ya exista
 * alguien que desee comunicarse y en caso contario, falla. Si se
 * hace en modo pasivo, el proceso se suspende hasta que haya
 * alguien que quiera comunicarse con éste.
 *  Esta función regresa un apuntador al nombre local de la
 * conexión */

struct tcb *open_tcp (lsocket, fsocket, mode, window, r_upcall,
                      t_upcall, s_upcall, tos)
struct socket *lsocket;       /* Socket Local */
struct socket *fsocket;       /* Socket Remoto */
int mode;                     /* Llamada Activa/pasiva */

/* Lo siguiente no forma parte de la especificación */

int16 window;       /* Tamaño de los buffers (recep. y trans) */

/* Funciones por llamar cuando... */
void (*r_upcall) ();      /* llegan datos */
void (*t_upcall) ();      /* pueden enviarse datos */
void (*s_upcall) ();      /* cambia el edo. de la conexión */

char tos;                 /* Tipo de servicio */
{
  struct connection conn;
  register struct tcb *tcb;
  void send_syn ();

 if (lsocket == NULLSOCK){      /* ¡no existe el socket local! */
   net_error = INVALID;
   return NULLTCB;
 }

/* Se crea la estructura para la conexión */

  conn.local.address = lsocket->address;
  conn.local.port = lsocket->port;
  if (fsocket != NULLSOCK){
    conn.remote.address = fsocket->address;
    conn.remote.port = fsocket->port;
  }
  else {                          /* Socket remoto no especificado*/
    conn.remote.address = 0;
    conn.remote.port = 0;
   }

  if ((tcb = lookup_tcb (&conn)) == NULLTCB){    /* no existe.. */
    if ((tcb = create_tcb (&conn)) == NULLTCB){ /* créalo */
      net_error = NO_SPACE;
      return NULLTCB;
    }
  } else if (tcb->state != LISTEN){           /* existía y en uso */
      net_error = CON_EXISTS;
```

```
      return NULLTCB;
    }

  tcb->user = user;
  if (window != 0)     /* window <> 0 => el usuario define valores */
    tcb->window = tcb->rcv.wnd = window;
  else
    tcb->window = tcb->rcv.wnd = tcp_window;

  tcb->r_upcall = r_upcall;
  tcb->t_upcall = t_upcall;
  tcb->s_upcall = s_upcall;
  tcb->tos = tos;

  switch (mode){
       case TCP_PASSIVE:       /* edo = esperando solicitud */
                          setstate (tcb,LISTEN);
                          break;
       case TCP_ACTIVE: /* envía SYN y entra al edo. SYN_SENT */
                          tcb->flags |= ACTIVE;
                          send_syn (tcb);
                          setstate (tcb,SYN_SENT);
                          tcp_output (tcb);
                          tcp_stat.conout++;
                          break;
  }
  return tcb;
}
```

ESTRUCTURAS DE DATOS.

Para formar los segmentos que el protocolo de transporte maneja, se define un área de almacenamiento (*buffer*) de la siguiente manera:

```
struct mbuf {
  struct mbuf *next;   /* Buffers formados por un solo paquete */
  struct mbuf *anext;  /* Paquetes ligados en colas */
  int16 size;          /* Tamaño del área de almacenamiento*/
  char *data;          /* Apuntadores activos de trabajo */
};
```

con las funciones: struct mbuf *alloc_mbuf () y *free_mbuf () para manejo dinámico de memoria.

Por portabilidad, se definen los siguientes tipos de datos:

```
typedef long int32;           /* entero de 32 bits con signo */
typedef unsigned short int16; /* entero de 16 bits sin signo */
```

El tamaño máximo de los segmentos que pueden intercambiarse por medio del protocolo de transporte es inicialmente de 512 *bytes*, pero las partes involucradas en la comunicación pueden modificar este valor. El tamaño por omisión del MSS se define por medio de la declaración:

```
#define DEF_MSS 512      /* Maximum Segment Size por omisión */
```

La parte correspondiente al encabezado se define a través de la estructura:

```
/* Encabezado del segmento -- representación interna
 * Nota : esta estructura no corresponde al encabezado formal del
 * protocolo de transporte (faltan los campos offset y checksum).
 *  El manejo de esa información se lleva a cabo a través de
 * funciones que pertenecen a otro módulo */


/* Banderas del encabezado */

#define URG 0x20          /* "URGent" (Datos urgentes) */
#define ACK 0x10          /* "ACKnowledgment" (Acuse de recibo) */
#define PSH 0x08          /* "PuSH" (Envío de datos) */
#define RST 0x04          /* "ReSeT" (Abortar conexión) */
#define SYN 0x02          /* "SYNchronize" (Iniciar conexión) */
#define FIN 0x01          /* Bandera "FINal" */


struct tcp {
 int16 source;            /* socket fuente */
 int16 dest;              /* socket Destino */
 int32 seq;               /* Número de Sequencia */
 int32 ack;               /* Número de Acuse de recibo */
 char flags;              /*  Banderas + desplazamiento de los
                             datos */
 int16  wnd;              /*  Ventana de control de flujo de
                             Recepción */
 int16 up;                /* Apuntador Urgente */
 int16 mss;               /* maximum segment size opcional */
};
```

La confiabilidad del protocolo de transporte se basa en su capacidad para detectar y corregir datos dañados, duplicados o que llegan en desorden y recuperar datos perdidos. Para poder detectar los casos mencionados y llevar a cabo la implantación de los mecanismos de detección y corrección correspondientes, el protocolo de transporte asocia a cada paquete de datos que envía un número de secuencia y una suma de verificación.

Al empezar la comunicación, los homólogos establecen cuál es el número inicial de secuencia de los paquetes que van a intercambiar, y para cada paquete de datos que llega, envían de regreso un acuse de recibo en el que se indica el número de secuencia del siguiente segmento que se espera que llegue.

La elección del número inicial de secuencia se implanta por medio de una variable de tipo estático a la que se asigna un valor aleatorio para la primera conexión que se establezca. La función regresa dicha variable incrementada en 250 000 como número inicial de secuencia para las siguientes conexiones. Es importante notar que la implantación debe asegurar que los números de secuencia sean positivos y menores o iguales a $2^{32}-1$ (4 294 967 295):

```
/* Determina el número inicial de sequencia */

int32 iss ()
{
  static int32 seq;

  seq += 250000;
  return seq;
}
```

Asociado a cada segmento existe un reloj; al enviar cada segmento, el reloj correspondiente se pone a funcionar y el segmento se inserta en una cola; en caso de que el tiempo fijado transcurra y no se reciba el acuse correspondiente, el segmento se

retransmite. De esta manera, se elimina el problema de los datos perdidos, pero se genera la posibilidad de recibir datos duplicados, si por alguna razón el acuse tarda mucho tiempo en llegar. Este problema se soluciona por medio de los números de secuencia: si llegan varios segmentos con el mismo número de secuencia, se envían los acuses de recibo correspondientes (para que no ocurran más retransmisiones) y se desechan los datos duplicados.

Otra situación que puede presentarse es que los segmentos lleguen en desorden (los paquetes de datos no tienen por qué seguir la misma ruta para llegar a su destino, lo que significa que el tiempo que tardan en recibirse es diferente). Al recibir los segmentos, los datos se van añadiendo a una cola para su ordenamiento, lo que puede hacerse fácilmente usando el número de secuencia que cada segmento tiene asociado.

La estructura por medio de la cual se maneja la cola de reordenamiento es la siguiente:

```
/* Cola de reordenamiento (Resequencing queue) */

struct reseq {
 struct reseq *next;          /* Apuntador al sig. elem de la lista */
 char tos;                    /* TOS (Type of service) */
 struct tcp seg;              /* Encabezado de TCP */
 struct mbuf *bp;             /* datos */
 int16 length;                /* longitud de los datos */
};
```

El tiempo de espera para recibir el acuse correspondiente a un paquete enviado se calcula tomando en cuenta el tiempo máximo que puede tardarse un paquete de datos en llegar a la otra máquina y regresar (tiempo de viaje redondo), y esto depende del tiempo máximo de vida del segmento. En esta implantación el tiempo de vida inicial es de 15 segundos; los valores correspondientes a los tiempos de vida y tiempo de viaje redondo se definen por medio de

las declaraciones:

```
#define DEF_RTT 5000      /* Tiempo inicial para viaje redondo
                             (round trip time): 5 seg. */
#define MSL     15        /* Tiempo máximo de vida */
#define MSL2    30        /* El doble del tiempo máximo de vida de
                             los segmentos (maximum segment life) */
```

El protocolo de transporte usa la técnica de ventanas corredizas (*sliding window*) para control de flujo. En esta implantación el tamaño inicial de la ventana es de 2048 caracteres.

```
#define DEF_WND 2048      /* Ventana de recepción por omisión */
```

La implantación de la ventana corrediza se hace a través de un contador y dos colas: el contador representa el tamaño de la ventana (es decir, indica cuántos datos pueden enviarse sin esperar acuse). Los datos que están a la izquierda de la ventana (enviados y para los que ya se recibió acuse) no se guardan; los datos dentro de la ventana (enviados y sin acuse) se almacenan en una cola para poder retransmitirlos en caso necesario. Los datos a la derecha de la ventana (por enviar) se guardan en la cola de transmisión, definida de la siguiente manera:

```
struct mbuf *sndq;      /* Cola de datos por enviar */
```

El algoritmo que se sigue para calcular la suma de verificación es obtener el complemento a uno en 16 bits de la suma en complemento a uno de las palabras de 16 bits que forman el encabezado del protocolo de transporte y los datos. La información que lleva cada segmento y que debe ser considerada al calcular la suma de verificación es la siguiente: los campos fuente y destino, el número de secuencia, el número de acuse, las banderas de control, el valor de la ventana de recepción, el apuntador

urgente, el valor del tamaño máximo del segmento y, por supuesto, los datos que se transmiten. Para calcular la suma de verificación se asume que el campo asociado a la propia suma de verificación tiene valor de cero. El código para la implantación de la suma de verificación es el siguiente:

```
/* Suma de verificación. */

/* Definiciones previas */

/* Obtiene la parte alta de la palabra */
#define hiword (x) ((int16) ((x) >> 16))

/* Obtiene la parte baja de la palabra */
#define loword (x) ((int16) (x))

/* Pseudo-encabezado para la suma de verificación */
struct pseudo_header {
  int32 source;          /* fuente IP */
  int32 dest;            /* destino IP */
  char protocol;         /* Protocolo */
  int16 length;          /* longitud de los datos */
};

#define TCP_PTCL 6       /* El protocolo es TCP */

/* Calcula la suma de  verificación.
   Opcionalmente  se  tiene  el  pseudo-encabezado  */
int16 cksum (ph, m, len)
struct pseudo_header *ph;          /* Pseudo encabezado */
register struct mbuf *m;           /* Datos */
int16 len;                         /* long. total */
{
  register unsigned int cnt, total;
  register int32 sum, csum;
  register char *up;
  int16 csum1;
  int swap = 0;
  int16 lcsum ();   /* Suma de verificación sobre 16 bits */

  sum = 01;          /* limpia sum */
  /* Obtiene la suma de verificación del pseudo encabezado */
  if (ph != NULLHEADER){
    sum =  hiword (ph->source);     /* En este caso, 0 */
    sum += loword (ph->source);     /* fuente */
    sum += hiword (ph->dest);       /* cero */
    sum += loword (ph->dest);       /* destino */
    sum += uchar (ph->protocol);    /* protocolo
    sum += ph->length;              /* long. de los datos */
  }
```

```
  /* Calcula la suma de verificación para los datos */
  for (total = 0; m != NULLBUF && total < len; m = m->next) {
    cnt = min (m->cnt, len - total);
    up = (char *) m->data;
    csum = 0;
    if (((long) up) & 1){      /* Empieza en una localidad impar? */
      /* Sí: suma el byte del inicio */
      if (swap)
        csum = uchar (*up ++); /* La inf. está en la parte baja */
      else      /* La inf. está en la parte alta */
        csum = (int16) (uchar (*up ++) << 8);
      cnt--;
      swap = !swap;
    }
    if (cnt > 1){    /* Aún hay datos? */
      /* Se calcula la suma de verificación para las palabras de
       * 16 bits que podemos asegurar que están a partir de loca-
       * lidad par. El buffer que se pasa como parámetro contiene
       * bytes y la suma se realiza en 16 bits, por lo que la lon-
       * gitud real de datos es cnt / 2 */
      csum1 = lcsum ((unsigned short *) up, cnt >> 1);
      if (swap)
        csum1 = (csum1 << 8) | (csum1 >> 8);
      csum += csum1;
    }
    /* Si quedan datos en una localidad impar: */
    if (cnt & 1){    /* Suma el byte a los datos */
      if (swap)
        csum += uchar(up[--cnt]);
      else
        csum += (int16)(uchar(up[--cnt]) << 8);
      swap = !swap;
    }
    sum += csum;
    total += m->cnt;
  }
  /* Regresa el complemento a uno de la suma en 16 bits */
  return (~eac(sum) & 0xffff);
}
```

MANEJO DE CONEXIONES.

Internamente, el control de las conexiones se hace por medio de una estructura llamada bloque de control de transmisión (tcb), que contiene la siguiente información: estado en que se encuentra la conexión (*CLOSE*, *OPEN*, *ESTABLISHED*, *CLOSING*, etc.), variables

de transmisión (número de secuencia del siguiente segmento por enviar, número de secuencia del segmento que se envió para el que aún no ha llegado el acuse de recibo, número inicial de secuencia para los paquetes que se están transmitiendo, cola de paquetes por transmitir, etc.), variables de recepción (número de secuencia del segmento que se espera recibir, número inicial de secuencia de los paquetes que se están recibiendo, cola de recepción, etc.), tamaño máximo de los segmentos y cola para reordenar los segmentos que se recibieron. Los tcb's se definen por medio de la siguiente estructura:

```
/* Bloque de control para las conexiones TCP */
/* Estados por los que puede pasar una conexión */

#define CLOSED  0               /* No hay conexión */
#define LISTEN  1               /* Esperando solicitud de conexión */
#define SYN_SENT 2              /* Envió solicitud de conexión */
#define SYN_RECEIVED 3          /* Recibió solicitud */
#define ESTABLISHED 4           /* Comunicación establecida */
#define FINWAIT1 5              /* Envió solicitud de terminar */
#define FINWAIT2 6              /* Espera sol. terminar del usuario
                                   remoto */
#define CLOSE_WAIT 7            /* Espera sol. terminar del usuario
                                   local */
#define CLOSING  8              /*  Espera acuse de la sol. de
                                   terminacion */
#define LAST_ACK 9              /*  Espera acuse para cerrar la
                                   conexion */
#define TIME_WAIT 10            /* Tiempo de espera para cerrar la
                                   conexion */


/* Razones por las que puede terminar una conexión */
#define NORMAL  0       /* Terminación Normal */
#define RESET   1       /* La otra parte envió Reset */
#define TIMEOUT  2      /* # de retransmisiones excesivo */
#define NETWORK  3      /* Problemas de la red (mensaje ICMP) */

struct tcb {
 struct tcb *prev;      /* El manejo de colisiones se hace por */
 struct tcb *next;      /* medio de una lista doblemente ligada */

 struct connection conn; /* Información de la conexión */

 char state;            /* Estado de la Conexión */
 char reason;           /* Razones para cerrar una conexión */
 char type;             /* Si la razón para terminar la conexión
                           es* /
 char code;             /* problemas de la red los valores del */
```

```
                        /* ICMP correspondientes al tipo y có- */
                        /* digo se guardan en estos campos */

/* Variables de secuencia de Envío */
struct {
 int32 una;          /* Primer número de secuencia sin acuse */
 int32 nxt;          /* Sig. # de secuencia */
 int32 ptr;          /* Apuntador transmisión actual */
 int16 wnd;          /* Ventana de recep. ofrecida por el otro */
 int16 up;           /* Apuntador para envío urgente */
 int32 wl1;          /* # sec. última actualización de ventana */
 int32 wl2;          /* # acuse última actualización de ventana */
} snd;

int32 iss;           /* Número inicial de secuencia para envío */
int32 resent;        /* Contador de bytes retransmitidos */

/* Variables de secuencia de recepción */
struct {
 int32 nxt;          /* Sig. # de secuencia que debe llegar */
 int16 wnd;          /* Ventana de recepción que ofrecemos */
 int16 up;           /* Apuntador urgente de recepción */
} rcv;

int32 irs;           /* Número inicial de secuencia de recepción */
int16 mss;           /* Tam. max. seg. (Maximum Segment Size) */
int32 rerecv;        /* Cont recep. bytes duplicados */

int16 window;        /* Limite ventana de recep. y cola de trans. */

void (*r_upcall)(); /* Ejecutada cuando llegan suficientes datos */
void (*t_upcall)(); /* Ejec. cuando pueden enviarse más datos */
void (*s_upcall)(); /* Ejec. si cambia el estado de la conexión */

char flags;          /* Banderas de control */
char tos;            /* tipo de servicio (para IP) */

struct mbuf *rcvq; /* Cola de recepción */
int16 rcvcnt;        /* Cont. recepción */

struct mbuf *sndq; /* Cola de transmisión */
int16 sndcnt;        /* Cont. de #'s de secuencia sin acuse en la
                        cola de trans. (incluye SYN y FIN, aunque
                        no aparecen en la cola de trans.) */

struct reseq *reseq;     /* Cola de segmentos desordenados */
struct timer timer;      /* Reloj de retransmisión */
struct timer rtt_timer; /* Tiempo viaje redondo (Round Trip Timer) *
int32 rttseq;            /* # secuencia para el que corre el reloj */
};
```

Es importante conocer esta información para poder llevar a cabo el manejo de cada conexión; el mecanismo de control que se implanta es una tabla de dispersión. El indice de la tabla se obtiene haciendo un OR exclusivo de las palabras altas y bajas de las direcciones remotas y locales de la conexión módulo el tamaño de la tabla, que en esta implantación se ha definido para poder manejar 20 conexiones.

```
#define NTCB 20          /* # TCB's en la tabla de dispersión */
extern struct tcb *tcbs [];
```

Una función importante que hay para manejo de la tabla es:

```
int16 hash_tcb (conn)
```

que calcula el lugar de conn de acuerdo a la función de dispersión. La implantación de hash_tcb es la siguiente:

```
/* Función de dispersión */
static int16 hash_tcb (conn)
struct connection *conn;
{
  register int16 hval;

  hval =  hiword (conn->remote.address);
  hval ^= loword (conn->remote.address);
  hval ^= hiword (conn->local.address);
  hval ^= loword (conn->local.address);
  hval ^= conn->remote.port;
  hval ^= conn->local.port;
  hval %= NTCB;
  return hval;
}
```

Estas no son todas las funciones implantadas para lograr el funcionamiento correcto del protocolo de transporte; son sólo una muestra de la manera en que se llevó a cabo. El código correspondiente a la implantación del protocolo de transporte ocupa 61071 bytes.

La diferencia básica con TCP es que no se verifican las cuestiones de seguridad, porque en la red local no tiene sentido;

además, garantizar que personas no autorizadas no puedan leer o alterar la información que se transfiere complica la implantación, pues la cantidad de verificaciones que se tienen que hacer es muy grande.

Dado que ésta es la única diferencia, el protocolo de transporte proporciona los servicios necesarios para que cualquier programa de aplicación que funciona sobre TCP funcione sobre el protocolo de transporte sin necesidad de modificarlo. De lo anterior se puede concluir que el protocolo de transporte es completo y funcionalmente equivalente a TCP.

BIBLIOGRAFIA

ETHERNET

- *The ethernet: a local area network; data link layer and physical layer specifications* V 1.0
  Digital equipment corp., Intel corp. and Xerox corp. [DIX]
  Comput. Commun. Rev. 11 (3)
  sep. 1980 pp 20-66

Definición oficial de *ethernet*. Es indispensable leerla si se va a trabajar con *ethernet*.

- *Ethernet: Distributed Packet Switching for Local Computer Networks.*
  Robert M. Metcalfe & David R. Boggs [Metcalfe]
  Communications of the ACM Vol. 19 No. 7
  julio de 1976

Plantea las decisiones de diseño y aspectos de implantación que deben considerarse si se quiere desarrollar una red local sobre *ethernet*.

- *Evolution of the Ethernet local computer network*
  John Shoch et al. [Shoch]
  Computer
  agosto de 1982, pp. 1-27

Análisis histórico del desarrollo de *ethernet* y de las circunstancias que han afectado su evolución; analiza de manera interesante CSMA/CD y CRC; incluye un resúmen técnico bastante completo de la versión 1.0 de *Ethetnet*.

IP

- *Internet Protocol*
  DARPA Internet Program Protocol Specification
  RFC 791 [RFC 791]
  septiembre de 1981

Especificación oficial de IP. Define formatos, direccionamiento, manejo de errores, códigos de opciones, seguridad, precedencia, fragmentación y ensamble, interfaz a los procesos de la capa superior, etc.

- *A standard for the Transmission of IP datagrams over Ethernet Networks*
  Charles Hornig [Hornig]
  RFC 894
  abril de 1984

Define un método estándar para empacar los datagramas de IP en paquetes de *ethernet* (especificamente, *ethernet* a 10 *Mbits*, con direcciones de 48 *bits*).

- *The ARPA Internet Protocol*
  Jonathan B. Postel, Carl A. Sunshine & Danny Cohen [Postel]
  Computer Networks V5 N4
  jul. de 1981.

Análisis bastante completo del protocolo IP.

TCP

- *A TCP/IP Tutorial*
  T. Socolofsky & C. Kale
  RFC 1180 [RFC 1180]
  enero de 1991

Explica las características relevantes de TCP/IP, enfatizando los aspectos técnicos.

- *Internetworking with TCP/IP* (Vol. 1 y 2)
  principles, protocols and architecture
  Douglas Comer [Comer-1] [Comer-2]
  Ed. Prentice Hall
  1991

Muy buenos libros de TCP/IP. Van de definiciones sencillas a cuestiones técnicas completas y complejas, pero muy bien explicadas. Cubren los protocolos ARP, IP, ICMP, UDP, TCP, RIP y SNMP; explican muy bien la interfaz a nivel de *sockets*.

- *Transmission control protocol*
  DARPA Internet Program Protocol specification
  RFC 793 [RFC 793]
  sep. de 1981

Especificación oficial de TCP. Describe las funciones de TCP, los aspectos más importantes que debe cubrir el programa que lo implante y sus interfaces hacia las capas adyacentes. Todo aquel que desee saber sobre TCP debe leer este documento.

- *TCP on a LAN*
  M. A. Padlipsky
  RFC 872 [RFC 872]
  septiembre de 1982

Describe brevemente lo que es una LAN, sus diferentes topologías y TCP; realza las características de TCP que lo hacen adecuado para las LAN's.

4. VERIFICACION Y PRUEBA

CASO DE PRUEBA : TELNET

Las pruebas de verificación se llevaron a cabo implantando un programa de aplicación sobre el protocolo de transporte y comprobando que el programa de aplicación funciona de la manera usual. El programa elegido es **telnet**.

METODOLOGIA DE DISEÑO

Para poder desarrollar *software* que funcione sobre la capa de transporte es necesario considerar varios aspectos; como se dijo anteriormente la comunicación entre las entidades del protocolo se lleva a cabo por medio de conexiones, de manera análoga a las llamadas telefónicas. Para poder intercambiar información lo primero que debe hacerse es establecer la comunicación, indicando con quién se desea hablar (marcar el número telefónico); si se establece la comunicación la parte que la inició se identifica (lo que en el protocolo se hace por medio de la dirección fuente y en el ejemplo de la llamada telefónica es equivalente a decir quien habla); ya que se está intercambiando información en ocasiones hay problemas (como el que las personas hablen al mismo tiempo o que haya interferencia en la línea), lo que hace que sea necesario el comprobar que la información llegó correctamente (en el protocolo esto se hace por medio de los acuses de recibo y las sumas de verificación; en las llamadas telefónicas ésto es más sencillo: simplemente se pregunta a la otra persona si nos escuchó); si en el transcurso de la llamada se tienen dudas acerca de si se ha interrumpido la comunicación o si la interferencia es demasiado grande como para continuar, normalmente se pregunta a la otra persona si puede escucharnos (lo que en el protocolo puede hacerse

preguntando por el estado de la conexión). Una vez que concluye el intercambio de información y en situaciones normales, se avisa a la otra persona que ya va a terminarse la comunicación (en el protocolo se cierra la conexión); en casos críticos, la comunicación puede terminarse sin avisar a la otra parte (abortar la comunicación).

En caso de que el *software* que se desarrolla sea para ofrecer un servicio, debe asegurarse que la conexión se inicia en modo pasivo y que la otra parte sabe la dirección del servidor. Este es el caso de telnet. Lo que se quiere es tener un servidor en el puerto por omisión de telnet (puerto 23 de TCP) que lleve a cabo las mismas funciones.

La siguiente sección da un panorama general del comando telnet. Para distinguir el programa de aplicación del protocolo que éste usa para intercambiar información, se escribirá TELNET para referirse al protocolo y telnet para hacer referencia al comando.

PANORAMA GENERAL DE telnet

El objetivo de telnet es proporcionar servicios de comunicación de propósito general, bidireccional y orientada al manejo de *bytes* (ocho *bits*).

El comando telnet proporciona una interfaz al usuario con un sistema remoto. Telnet se comunica con un sistema remoto usando el protocolo TELNET. La comunicación en TELNET se realiza por medio de conexiones; una conexión TELNET es una conexión a nivel del protocolo de transporte, que se usa para transmitir datos, a los que se añade información de control de TELNET. Si telnet se ejecuta sin argumentos, entra al modo de comandos, lo que se indica por medio del prompt telnet>. El modo de comandos puede usarse en cualquier momento, tecleando la secuencia de escape ^]

Si se ejecuta con argumentos, telnet establece la conexión usando los argumentos que recibe.

Ya que se estableció una conexión, telnet pasa al modo de *input*, en el que el texto que recibe es transmitido a la máquina remota. Dependiendo de las características de la máquina remota, la transmisión se efectuará por caracter o por línea. La transmisión por caracter significa que los caracteres son enviados a la máquina remota tan pronto como se reciben del teclado local. En la transmisión por línea, los caracteres son mostrados en la pantalla local, pero no se transmiten hasta haber completado el tamaño de línea determinado por el sistema.

ASPECTOS TECNICOS

Con TELNET se define un método estándar para tener la interfaz adecuada entre terminales y procesos orientados a terminal. El protocolo definido por TELNET también puede usarse para establecer comunicación terminal-terminal (*linking*) y proceso-proceso (*computación distribuida*).

El protocolo maneja los aspectos relacionados con el control de dos flujos de datos independientes y que circulan entre las máquinas en direcciones contrarias. Las terminales virtuales con que trata el protocolo están formadas por una sola línea de longitud ilimitada, de tal forma que los caracteres que son tecleados en la terminal se transmiten como un flujo de *bytes* a través de la línea. Existe un proceso que convierte el código de cada caracter que sale de la terminal al estándar de la red (código ASCII). En el intercambio de información se tienen, a nivel de protocolos homólogos, quince comandos de control que pueden mezclarse con los datos. La transmisión de los comandos es precedida por el caracter IAC (*Interpret As Command*), que indica al protocolo que lo que sigue es un comando.

En TELNET, existen dos conceptos fundamentales:

▪Terminal Virtual de Red (*Network Virtual Terminal*). Al establecer una conexión TELNET, ambas partes asumen que en cada extremo de la conexión se encuentra una terminal virtual de red (NVT por sus siglas en inglés). Una NVT es un dispositivo imaginario bidireccional, orientado a caracter, que proporciona la representación estándar intermedia de una terminal. Se asume que la NVT cuenta con un teclado y una impresora; por medio del teclado se producen los datos de salida y la impresora maneja los datos de entrada. Todas las máquinas anfitriones mapean las características de sus dispositivos locales de tal forma que parezca que se está trabajando con una NVT. Durante la existencia de una conexión TELNET se asumen las siguientes convenciones:

– Para transmitir, los datos se almacenan en la máquina donde se originan hasta completar una línea de información, o hasta que se reciba alguna señal local para transmitir; esta señal puede ser enviada por el usuario o por un proceso.

– Cuando un proceso que forma parte de una conexión TELNET no puede continuar sin recibir datos de la otra parte, el primer proceso debe transmitir el comando *GO AHEAD* (ga).

▪Opciones negociadas (*Negotiated Options*). Como parte del protocolo se manejan varias opciones que pueden negociarse (por ejemplo cambiar el conjunto de caracteres o el modo de *echo*). Las opciones se usan para modificar de manera dinámica las características de la conexión, y así poder adaptarse a los cambios locales que ocurran en el sistema. Las negociaciones se realizan cuando alguna parte (o ambas) piden que alguna opción se ponga a funcionar. La otra parte acepta o rechaza la petición. Si acepta, la opción se ejecuta inmediatamente; si se rechaza, la opción se sigue manejando según lo especifica el NVT. Se sigue la convención de que siempre puede rechazarse una petición de

habilitar algún servicio, pero no puede rechazarse una petición para deshabilitar. Si ambas partes envían la petición simultáneamente, cada parte interpreta la petición que recibe como el acuse positivo de su petición.

Cada conexión de TELNET comienza negociando opciones. Durante este proceso, los comandos que se manejan son: *WILL, WON'T, DO* y *DON'T*. *WILL* X se usa para indicar que la terminal que envía desea comenzar a ejecutar la opción X (es decir, ofrece el servicio X). La terminal que recibe el aviso, contesta con *DO* X o *DON'T* X, que son la manera de indicar si está de acuerdo o no con que se ejecute la opción. *DO* X y *DON' T* X son interpretados como acuses positivo y negativo, respectivamente, para la petición *WILL* X.

DO X se envía para pedir que la otra parte ejecute la opción X (es decir, requiere el servicio X). En este caso, *WILL* X y *WON'T* X son enviados como acuses positivo y negativo para comunicar a la terminal si se va a ejecutar la opción que solicitó.

Debido a que las características del NVT son respetadas cuando se rechaza un ofrecimiento o una petición, enviar *DON'T* y *WON'T* como respuesta garantiza que la conexión permanecerá con una configuración que pueden manejar ambas partes.

Dado que la sintaxis de las negociaciones es simétrica, existe el peligro de llegar a ciclos infinitos de acuses de recibo, pues al negociar opciones cada parte interpreta los comandos que recibe como nuevas peticiones para las que debe enviar un acuse de recibo. Para evitar ésto, durante las negociaciones se siguen las reglas:

1. Cada parte envía opciones sólo para solicitar un cambio de estado (no se permite enviar opciones para avisar en qué estado se encuentra trabajando actualmente).

2. Si se recibe una petición para cambiar a algún modo

en el que ya se está trabajando, no se envía acuse. Esto significa que los acuses se transmiten sólo cuando se recibió una petición para cambiar de modo, aún cuando el cambio no se haya efectuado. Las opciones que se rechazan no vuelven a solicitarse hasta que ocurra algún cambio en las condiciones locales.

3. Los comandos se añaden a la información en el punto en que se desea que tengan efecto. Como existe un tiempo entre la transmisión de la solicitud y la recepción del acuse (que puede ser negativo), normalmente se almacenan los datos hasta conocer el resultado de la solicitud.

FUNCIONES DE CONTROL

En TELNET se define la representación estándar de cinco funciones para control de las conexiones:

- *Interrupt Process* (ip). Función para suspender, interrumpir o terminar un proceso de usuario

- *Abort Output* (ao). Permite a un proceso que genera datos de salida funcionar hasta terminar, sin enviar los datos de salida a la terminal del usuario.

- *Are You There* (ayt). Proporciona al usuario evidencia de que el sistema aún está funcionando. Esta función se usa principalmente cuando no se recibe respuesta del sistema debido a carga excesiva.

- *Erase Character* (ec). Borra el último caracter tecleado del flujo de datos proporcionado por el usuario.

- *Erase Line* (el). Borra todos los datos de la línea actual de entrada.

IMPLANTACION DE telnet

La implantación de telnet se llevó a cabo bajo el mismo ambiente que el protocolo de transporte. A continuación se describen las partes fundamentales de la implantación.

DEFINICIONES

```
#define LINESIZE 256      /* Longitud del buffer local de edición */

/* Comandos de Telnet */

#define IAC   255         /* Interpret as command */
#define WILL  251
#define WONT  252
#define DO    253
#define DONT  254

/* Bloque de control para Telnet */

struct telnet {
 struct tcb *tcb;
 char state;

 char local [NOPTIONS];       /* Opciones Locales */
 char remote [NOPTIONS];      /* Opciones Remotas */
 struct mbuf *inbuf;     /* Apt. al buffer de caracts. recibidos */
 struct mbuf *outbuf;    /* Apt. al buffer de caracts. por enviar */
 int fd;                 /* Descriptor de archivo de tty */

 struct session *session; /* Apuntador a la estructura de sesión */
};
```

DESCRIPCION DE LAS PRUEBAS

Después de implantar telnet sobre el protocolo de transporte, se decidió cambiar temporalmente los nombres de los servicios proporcionados por el protocolo de transporte para asegurar que telnet funciona sobre él y no sobre la versión de TCP con que

cuenta la HP 9000/840.

Los nuevos nombres son el equivalente en español al que ya tenían (abre, cierra, aborta, envía, recibe y estado). Este cambio no se hizo permanente, pues ello implicaría que los programas de aplicación no podrían funcionar sobre el protocolo de transporte, pues según el RFC 793, ellos asumen que los servicios que pueden utilizar se llaman *open*, *close*, etc.

Una vez hecho el cambio, se establecieron sesiones remotas; primero a dos de las máquinas que integran la red local: la *sun sparc station ipc* y la *sun sparc station* 1. Las sesiones se desarrollaron de la manera usual. No pudieron realizarse pruebas de sesiones remotas con las 386's por limitaciones del *software* instalado en ellas.

Posteriormente se establecieron sesiones remotas a máquinas de la UNAM con las que existe comunicación a través de *ethernet*. Estas máquinas fueron: una IBM 4381 (en DGSCA) y una *sun* (en el IIMAS).

La última prueba que se llevó a cabo fue establecer una sesión remota con una *sun sparc station ipc* que se encuentra en Sn. José California, E.U.A.

No pudieron hacerse más pruebas por no contar con clave de acceso a otras máquinas, pero en el transcurso de las pruebas efectuadas, y tratando de cubrir los casos que eventualmente podrían presentarse, las sesiones se desarrollaron de la misma manera en que se comportan con el TCP y el telnet de la HP.

Como comentario final, se tomó una implantación de dominio público de ftp (*File Transport Protocol*. Con este programa pueden transferirse archivos entre sistemas remotos); se le hicieron las modificaciones necesarias para que funcionara sobre el protocolo de transporte y bajo el sistema operativo de la HP 9000/840. Ya

que se logró esto, se probó la transferencia de archivos entre varias máquinas (las *sun*'s ya mencionadas, las 386's de la red local y máquinas a las que se tiene acceso a través de *ethernet* y que se encuentran en distintos lugares del mundo), funcionando todo correctamente.

BIBLIOGRAFIA

TELNET

- *Telnet Protocol Specification*
  RFC 854 [RFC 854]
  J. Postel & J. Reynolds
  Mayo de 1983

Definición oficial de *telnet* para ARPA, su funcionamiento, ambiente de ejecución, funciones de control y manejo de conexiones.

CONCLUSIONES

La importancia de este trabajo radica en la experiencia adquirida al trabajar con la implantación de un protocolo tan completo como lo es TCP. Por otro lado, el contar con el código para el protocolo de transporte da la ventaja de poder experimentar con las interfaces hacia las capas superiores e inferiores que forman el modelo de red.

Otro aspecto importante de este trabajo es que siendo las redes de computadoras una parte relativamente nueva en computación, aquí pueden encontrarse conceptos introductorios y referencias a todos los niveles para cada uno de los temas tratados. Este es un aspecto relevante pues al empezar este trabajo de tesis tuvo que invertirse mucho tiempo y esfuerzo para encontrar bibliografía.

Sin duda el trabajo de investigación desarrollado para obtener información de redes, OSI, TCP/IP y telnet y la experiencia lograda al trabajar con la implantación del protocolo de transporte será una valiosa ayuda para colaborar de manera importante en el proyecto del que surgió este tema de tesis.

PERSPECTIVAS

Debido a la complejidad del proyecto a partir del que se desarrolla este trabajo, el tener el protocolo de transporte funcionando en la HP no significa que el objetivo del proyecto, en relación a la capa de transporte, se haya alcanzado: aún no se ha terminado con la programación de transporte, pues se espera proceder a la implantación del protocolo en la *sun sparc station*

ipc y en las 386's bajo el sistema operativo MS-DOS y MINIX (un sistema operativo basado en UNIX para PC's); dado que se implantará el protocolo en sistemas operativos diferentes (tres basados en UNIX, además de MS-DOS), se plantean interesantes perspectivas de conexión entre computadoras con sistemas operativos heterogéneos, lo que se traduce en avances significativos en los objetivos globales del proyecto.

Por otro lado, sería muy interesante implantar sobre el protocolo de transporte los servicios que normalmente trabajan sobre TCP (*ftp* y *smtp*, además de *telnet*, que ya se tiene), para darle versatilidad y continuar realizando pruebas de funcionalidad y eficiencia.

RECONOCIMIENTO

El trabajo aquí presentado se basó fuertemente en el *software* de red desarrollado por Phil Karn en 1986.