UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

DISEÑO DE MODULOS PARA LA EXPERIMENTACION CON REDES DE COMPUTADORAS

T E S I S

Que para Obtener el Título de

M A T E M A T I C O

P r e s e n t a

José de Jesús Galavíz Casas

Ciudad Universitaria, Marzo de 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

PROLOGO

El presente trabajo se ha hecho como respuesta a la necesidad de que el curso de Introducción a las Redes de Computadoras, que se imparte en la Facultad de Ciencias de la UNAM, contara con herramientas que permitieran a los alumnos experimentar aquello que se les plantea en la parte teórica del curso.

Dado que la teoría del curso se basa en el modelo de referencia OSI (*Open Systems Interconnection*), un primer esfuerzo se encaminó a resolver la necesidad de experimentar en lo referente al diseño e implantación de *software* de red de bajo nivel, concretamente a nivel de las capas 1 y 2 del modelo de referencia (capa física y de enlace de datos, respectivamente), para poder hacer esto se necesitaba seleccionar el *hardware* de comunicación sobre el que se pretendía trabajar. La elección debía poseer las siguientes características:

- Que el *hardware* estuviera disponible en cantidades suficientes para los alumnos.
- Que existiera suficiente información al respecto.
- Que no fuera difícil de programar.

Bajo estos criterios se eligió programar sobre el puerto de comunicaciones seriales de las PC's de IBM. Se desarrolló entonces una interfaz sobre puerto serial que implantó en un módulo los servicios correspondientes a nivel de capa física y en otro los correspondientes al nivel de enlace. La construcción de esta interfaz en la clase con los alumnos, les plantea a estos una serie de problemas que si bien no son muy complicados, si dan una idea de lo que significa diseñar y construir las capas más bajas de una red.

Pero entonces surge la necesidad de que los alumnos del curso interactúen con una red de verdad y no sólo construyan una "red de juguete". Es decir, a pesar de que las capas de red que se hacen sobre puerto serial plantean problemas interesantes, es escasa su utilidad en el ámbito de las redes de computadoras reales.

Para resolver este problema se eligió trabajar sobre *Ethernet*, la red de área local más popular del mundo y una de las que tenemos disponibles en la Facultad de Ciencias. Se pretendía construir una interfaz de red que proporcionara servicios básicos para el envío y recepción de datos a través de la red al nivel al que se había quedado la interfaz sobre puerto serial. Esto es, continuar utilizando la interfaz anterior para que los alumnos experimenten en lo referente a las dos primeras capas del modelo OSI y proporcionarles (ya no construir con ellos), una interfaz de red que ofrezca servicios de enlace de datos para que sobre ella se puedan construir aplicaciones más serias que verdaderamente corran en un ambiente de red.

Lo que aquí se presenta es precisamente el diseño e implantación de la segunda de las interfaces de red mencionadas, la que se hizo sobre *Ethernet*. El trabajo se inscribe en el marco de un proyecto de innovación docente titulado: **Sistemas Distribuidos en Redes y Sistemas Operativos Heterogéneos**, el cual se lleva a cabo con participación de personal de la Facultad de Ciencias, el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas y el Instituto de Matemáticas, todos ellos de la UNAM y el Laboratorio Nacional de Informática Avanzada (LANIA). Se ha hecho como parte del trabajo que se realiza en lo que a redes de computadoras y sistemas de comunicación se refiere y, más allá de sus fines didácticos, se ha integrado perfectamente al trabajo que ya se había realizado en este rubro y ha posibilitado la construcción de aplicaciones útiles más generales.

El trabajo está organizado de la siguiente manera: en el capítulo I se exponen los conceptos fundamentales respecto a redes

de computadoras (qué son, topologías, arquitecturas de red); en el capítulo II se describen las características más sobresalientes de la red sobre la que se realizó el trabajo (*Ethernet*); en el capítulo III se describe, a grandes rasgos, el diseño y la implantación de la interfaz de red; finalmente, en el capítulo IV se describen las aplicaciones más importantes que se han construido sobre la interfaz.

I REDES DE COMPUTADORAS

1.1. INTRODUCCION.

Hasta antes de la década de los 60's los sistemas de cómputo utilizados eran enormes máquinas constituidas por decenas de miles de tubos al vacío (bulbos). Dadas las características de estos dispositivos dichas computadoras disipaban gran cantidad de energía en forma de calor, lo que traía consigo la necesidad de proveerlas de sistemas de enfriamiento para que pudieran operar sin sufrir daños. Debido a estas condiciones: gran cantidad de componentes, gasto de energía y sistemas de enfriamiento, estos equipos resultaban sumamente costosos y sólo unas cuantas empresas y universidades podían darse el lujo de contar con uno de ellos[1]. En estos sistemas sólo podía trabajar un usuario (de hecho un programador experimentado) a la vez, éste suministraba su programa a la computadora por medio de una lectora de tarjetas y esperaba a que sus resultados fueran enviados a la impresora, se poseía una muy limitada capacidad de entrada/salida. A este tipo de sistemas se les denomina de "procesamiento por lotes".

Con el advenimiento de los transistores y su utilización en la elaboración de los equipos de cómputo en la década de los 60's, el esquema de trabajo cambió notablemente, entonces se poseía una máquina a la que se encontraban conectadas varias terminales, desde éstas el usuario enviaba información a la computadora y recibía los resultados que ésta producía, podía haber varios usuarios en sesión simultáneamente en la misma computadora. Estas máquinas resultaban mucho menos costosas y voluminosas que sus ancestros y ya no requerían de sistemas de enfriamiento tan

[1] Aun hoy en día podemos encontrar esta misma situación en equipos de cómputo modernos. Las supercomputadoras actuales requieren de sistemas de enfriamiento, son enormes y muy costosas.

eficientes. A estos sistemas se les denomina de "tiempo compartido".

Tanto en los sistemas de procesamiento por lotes como en los de tiempo compartido los recursos de cómputo están centralizados, es decir se posee una sola computadora con su *software*. Pero en el caso de los segundos ocurre que estos recursos son compartidos por varios usuarios, mientras que en los primeros la computadora y su *software* forman una unidad integral dispuesta a atender a un sólo usuario a la vez.

Tras la utilización de los circuitos integrados a gran escala en la industria de los equipos de cómputo, sobrevino el auge de los sistemas pequeños (microcomputadoras) a partir de la década de los 70's. Estos sistemas son mucho más baratos y pequeños que todos los anteriores y pueden llegar a tener un poder de cómputo superior al que tenían aquellos. El arribo de las microcomputadoras trajo consigo una descentralización de los recursos y la ventaja de ponerlos en completa disponibilidad del usuario, pero también la desventaja de no poder compartirlos, es decir, se regresa a un caso análogo al de los sistemas de procesamiento por lotes lo que representa un retroceso en lo que a compartir recursos se refiere.

Es posible ejemplificar esta situación de la siguiente manera: supongamos que una compañía posee una base de datos necesaria en todos sus departamentos y que esta base es actualizada por las transacciones que se efectuán sobre ella en más de un departamento. Si esta compañía posee sólo microcomputadoras aisladas en cada departamento, será menester que haya alguien que corra por toda la compañía con los *diskettes* de la base de datos actualizada cada vez que se efectúa una transacción sobre una de las copias de ésta. Es decir, para disponer de poder de cómputo descentralizado pero a la vez tener la ventaja de compartir recursos es necesario que, de alguna forma, se pueda intercambiar información entre las computadoras. Si bien esta necesidad ya se había planteado con anterioridad, es

la llegada de las microcomputadoras el acontecimiento que hace esta necesidad más urgente.

Las redes de computadoras surgieron justamente para resolver el problema de compartir recursos. Una red de computadoras es un conjunto de máquinas independientes capaces de intercambiar información. Al decir que cada computadora de la red es independiente de las otras se está tratando de indicar que entre ellas no existe una relación de subordinación, ninguna es capaz de obligar a otra a hacer algo o de controlarla de alguna forma [Tanenbaum 91].

Además de resolver el problema de compartir recursos, tanto de *hardware* como de *software*, las redes de computadoras ofrecen algunas otras ventajas. Si se tiene información importante almacenada en una computadora aislada y se desea acceder a ella, este acceso dependerá de si funciona o no correctamente la computadora. En cambio si se posee una red de computadoras y la información importante se duplica en dos o tres máquinas, se puede tener acceso a ella sin importar que fallen una o dos de estas computadoras, dado que, por ser independientes entre sí las máquinas de la red, la falla de una de ellas no afecta de modo importante el funcionamiento de las demás; es decir, la red vista como un sistema integral, es mucho más confiable y tolerante a fallas que una computadora aislada. Por otra parte, en un sistema de tiempo compartido es una sola computadora la que procesa todo lo que sus multiples usuarios desean, mientras que en una red de computadoras la carga de tabajo se puede distribuir.

Físicamente una red de computadoras está constituida por tres elementos fundamentales: las computadoras conectadas a la red, también llamadas anfitriones (*hosts*); el medio de comunicación entre ellas (cable coaxial, par trenzado, fibra óptica etc.) y uno o varios dispositivos que los interconectan, es decir la interfaz física entre el anfitrión y el medio de comunicación. A los anfitriones se les suele llamar "Equipo terminal de datos" (DTE, *Data Terminal Equipment*) y al dispositivo físico directamente

conectado al medio de comunicación "Equipo de comunicación de datos" (DCE, *Data Communication Equipment*).

Además de los componentes físicos, para hacer posible la comunicación de datos entre equipos de cómputo es menester definir conjuntos de reglas que gobiernen la interacción de las computadoras involucradas en dicho intercambio, a esto se le denomina *protocolo de comunicación* [Holzman 91].

1.2. CLASIFICACION DE LAS REDES.

Las redes de computadoras se pueden clasificar de múltiples formas dependiendo de las características que se consideren relevantes. Se presentan aquí las clasificaciones hechas con base en:

- El tipo de arquitectura del medio de comunicación.
- El tamaño de la red.
- La topología de la red.

1.2.1. POR EL TIPO DE ARQUITECTURA DEL MEDIO DE COMUNICACION.

Si se considera como característica relevante el tipo de arquitectura para el medio de comunicación de la red se tienen dos posibilidades:

- o Comunicación punto a punto.
- o Comunicación por difusión.

En la comunicación punto a punto (fig. 1.1) cada segmento del medio de comunicación conecta dos anfitriones o nodos de la red. Si dos anfitriones desean comunicarse, pero no comparten una línea de comunicación, tendrán que hacerlo indirectamente a través de otros anfitriones u otros nodos de la red.

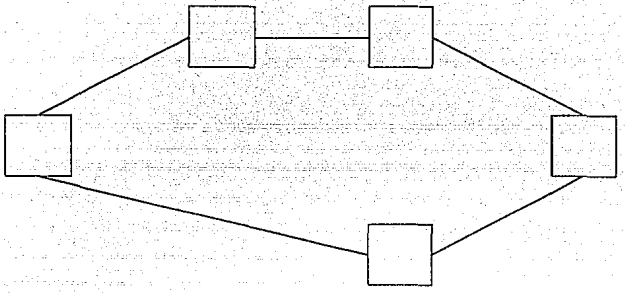


Fig. 1.1: Red con arquitectura de comunicación punto a punto.

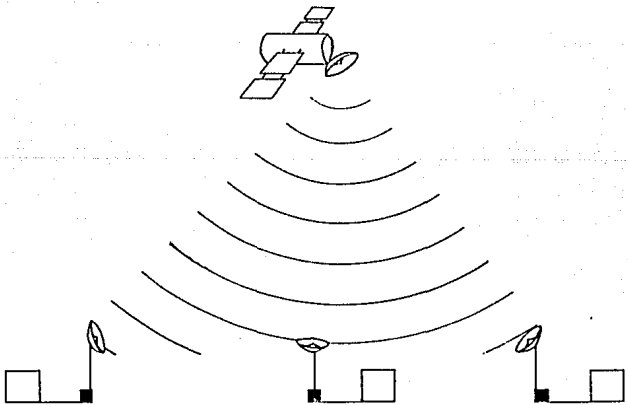


Fig. 1.2: Red con arquitectura de comunicación por difusión.

En la comunicación por difusión (fig. 1.2) existe un sólo canal de comunicación, el cual es compartido por todas las máquinas de la red. Lo que es transmitido por una de las computadoras que la integran es recibido por todas las demás. El destinatario de la información transmitida será informado de ello y procesará lo que recibe. Las máquinas a quienes no va dirigida dicha información la ignorarán.

Dentro del esquema de comunicación punto a punto, las redes pueden catalogarse de acuerdo al tipo de tecnología utilizada para el transporte de mensajes. De esta manera las redes con comunicación punto a punto pueden subdividirse en tres categorías, a saber:

- Redes de conmutación de circuitos.
- Redes de conmutación de mensajes.
- Redes de conmutación de paquetes.

Las primeras redes de computadoras eran de conmutación de circuitos, en ellas se utilizaban dispositivos llamados nodos de conmutación de circuitos; estos poseían múltiples líneas de entrada/salida, dos de ellas conectadas al medio de comunicación de la red y el resto a algunos anfitriones, como se muestra en la figura 1.3. Cuando se transmite un mensaje en una red de este tipo, los nodos de conmutación de circuitos proveen un camino completo de ligas físicas de transmisión entre el emisor y el receptor, de esta forma la transmisión se efectua sobre una línea directa dedicada. La función de los nodos de conmutación consistía en elegir por cual de sus salidas enviar la señal de entrada dependiendo del destinatario del mensaje. La desventaja de tener líneas dedicadas, aunque sólo sea temporalmente, es que no se pueden establecer diálogos cuyos mensajes pasen por nodos ocupados momentáneamente en la transmisión de un mensaje previo.

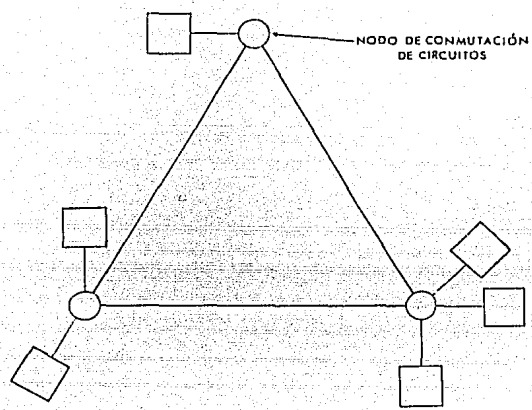


Fig. 1.3: Red de conmutación de circuitos.

En las redes de conmutación de mensajes la transmisión de información del anfitrión fuente al anfitrión destino se efectúa dividiendo esta en un conjunto de mensajes. Cada mensaje se mueve a través de ciertas ligas de transmisión pasando por los anfitriones que se encuentran en la ruta que debe seguir para llegar a su destino. Cuando un anfitrión recibe un mensaje que no es para él, lo almacena en un *buffer* aguardando a que el siguiente anfitrión en la ruta del mensaje esté listo para recibirlo; en cuanto esto ocurre extrae el mensaje del *buffer* y lo retransmite. Debido a esto, a las redes de este tipo se les denomina de almacenamiento y reenvío.

Las redes de conmutación de paquetes funcionan del mismo modo que las de conmutación de mensajes, salvo que en éstas el mensaje es partido en trozos de igual tamaño, llamados paquetes y son estos los que se envían, almacenan y retransmiten entre los anfitriones de la red.

Existe un tipo específico de paquete de información denominado *datagrama*, un datagrama se envía asumiendo, pero no asegurando, que llegará a su destino correctamente.

1.2.2. POR EL TAMAÑO DE LA RED.

Las redes de computadoras también pueden clasificarse de acuerdo a su tamaño, es decir, de acuerdo a la distancia máxima entre los nodos o anfitriones que la componen. Usando este criterio las redes se han clasificado en tres grupos:

- Redes de cobertura local (*Local Area Networks*, LAN's).
- Redes de cobertura extensa (*Wide Area Networks*, WAN's).
- Redes de cobertura metropolitana (*Metropolitan Area Networks*, MAN's).

La distancia a la que se conectan los anfitriones de la red es una factor importante en aspectos de confiabilidad y velocidad en la transmisión de la información.

En general no existe concenso al fijar las distancias que sirven de frontera entre las distintas clases de redes. La opinión más común es que si el máximo de las distancias entre cualesquiera dos nodos de la red es menor de 10 km., la red en cuestión es una **LAN** y si excede esta distancia se trata de una WAN [Tanenbaum 91]. Pero este criterio no considera a las MAN's; otras opiniones señalan un máximo de 1 km. para una **LAN** y más de 10 km. para una **WAN**, de modo que se puede acomodar fácilmente a las MAN's entre ambas [Halsall 92]. Es menos preciso, pero más entendible, si se dice que en una LAN los anfitriones se encuentran en el mismo edificio o grupo de edificios, que en una MAN estos se encuentran en la misma ciudad (ciudad de tamaño normal, no como la monstruosidad de la ciudad de México) y que en una WAN se encuentran en una o varias ciudades o paises.

Resulta evidente que tender un cableado a lo largo de un país entero para implantar una WAN es muy caro, así que las redes de ámbito extenso y hasta las de ámbito metropolitano utilizan medios de comunicación ya existentes para enlazar sus anfitriones: líneas telefónicas, comunicación vía satélite, microondas u ondas de radio. Estos medios están expuestos a muchos y muy diversos tipos de inteferencias, además de que algunos de ellos no admiten velocidades de transmisión muy altas, así que la transmisión a través de ellos es menos confiable y más lenta que en el caso de un cableado especial y de cortas distancias como en las LAN. La cobertura de la red determina en buena medida que tan rápida y confiable es la transmisión de información. En las LAN la velocidad varía entre 4 Mb/s (1 *Mega bit*/segundo = 1 millón de *bits* por segundo) y 2 Gb/s (1 *Giga bit*/segundo = 1×10^9 *bits* por segundo) [Comer 91], aunque lo más común es que se encuentre en el intervalo de 10 a 20 Mb/s. La tasa de errores en una LAN es de 1 *bit* erróneo por cada 10^n *bits* transmitidos. En las WAN la velocidad de transmisión va de 9.6 Kb/s (1 *Kilo bit*/segundo = mil *bits* por segundo) hasta 45 Mb/s con tasas de errores de alrededor de 1 *bit* erróneo por cada 103 o 105 *bits* transmitidos salvo en el caso de los enlaces por fibra óptica [Black 90].

En la actualidad, el abaratamiento de la fibra óptica y lo óptimo de sus características está revolucionando drásticamente lo que se ha mencionado. La fibra óptica permite la transmisión de señales luminosas a través de grandes distancias prácticamente sin pérdida, no le afectan los campos magnéticos ni eléctricos dado que no viajan por ella señales eléctricas, tampoco las condiciones atmosféricas ni la temperatura; su capacidad para transportar información es mucho más alta que la de cualquier otro medio. Esto ha ocasionado que se le utilice cada vez más como medio de comunicación en redes de computadoras ya que es posible construir redes bastante grandes y con muy alta velocidad y confiabilidad.

1.2.3. POR LA TOPOLOGIA DE LA RED.

Este criterio de clasificación se aplica más bien al caso particular de las LAN; dado que son redes más pequeñas se puede hablar de la forma que tienen, mientras que en el caso de las WAN su configuración física es, en la mayoría de los casos, irregular. Esto se debe a que las LAN normalmente se diseñan desde el principio, teniendo en mente una idea clara de la totalidad de la red, mientras que en el caso de las WAN su diseño obedece más bien a las necesidades que van surgiendo y a los medios de que se dispone para satisfacerlas.

o BUS O CANAL.

En esta configuración los anfitriones se conectan a un sólo canal común, es decir, el tipo de canal es de difusión (fig 1.4). Debido a esto, cada nodo de la red puede "escuchar" lo que es transmitido por el canal, es la interfaz de cada anfitrión con el medio la que tiene que discernir si lo que escucha está dirigido a él o no, en cuyo caso hay que ignorar lo que se recibe. Dependiendo de qué reglas se utilicen en la red para que un nodo pueda transmitir (acceder al medio) se presentarán ciertos problemas. En principio, la estructura física de la red nos sugiere que todos los nodos tiene igualdad de derecho a transmitir y que no hay uno de ellos más importante que los demás, sin embargo, se pueden implantar ciertas jerarquías entre los anfitriones, o restringir sus derechos de transmisión.

La topología en bus o canal es hoy por hoy la más popular entre las redes de área local. Su exponente más usual es *Ethernet*, red cuyo diseño original pertenece a las compañías DIGITAL EQUIPMENT CORPORATION (DEC), INTEL y XEROX.

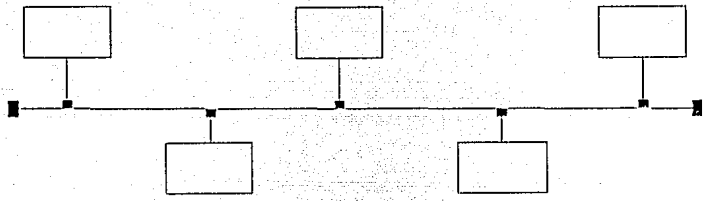


Fig. 1.4: Topología de canal.

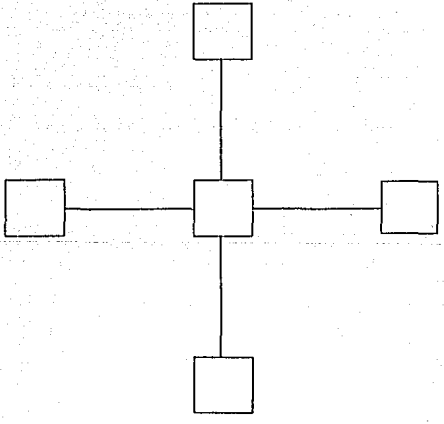


Fig. 1.5: Topología de estrella.

VENTAJAS:

- Es fácil conectar nuevos equipos a la red.
- Al conectar o desconectar anfitriones de la red no se interrumpe el tráfico entre los anfitriones restantes[2].
- No existe, en principio, ningun anfitrión que controle la red y que, en caso de dañarse, ocasione que se trastorne la comunicación (i.e. no existe control centralizado).

DESVENTAJAS:

- Es necesaria una interfaz inteligente para acceder al medio de comunicación.
- Daño físico al medio, o una instalación incorrecta de ciertos dispositivos conectados a él, ocasionan reflexiones indeseables de la señal.
- No existe control centralizado, por lo tanto es difícil detectar fallas en el sistema.
- Dado que los anfitriones comparten un medio común y tienen, en principio, el mismo derecho de transmitir puede ocurrir interferencia si varios de ellos tratan de transmitir, al mismo tiempo (colisiones).

o ESTRELLA.

En este tipo de configuración todos los anfitriones de la red se conectan a un anfitrión o nodo central (fig. 1.5), de tal forma que todos los mensajes deben pasar por él antes de llegar a su destino. Actualmente está siendo muy usada esta topología debido a que muchas compañías utilizan sus líneas telefónicas internas para conectar sus equipos de cómputo, de esta forma enlazan diferentes oficinas del mismo edificio o conjunto de edificios con líneas no dedicadas. En este caso, las pequeñas centrales telefónicas, PABX (*Private Automatic Branch eXchange*), hacen las veces del nodo central de una red en estrella.

[2] Esto no es cierto siempre: existen versiones de la ya mencionada ETHERNET en las cuales el conectar equipo a la red implica "romper" momentáneamente el canal.

VENTAJAS:

- Cada nodo es independiente; si uno de ellos falla y no es el nodo central, la comunicación entre los demás no se ve afectada.
- Fácil mecanismo de direccionamiento (i.e. decidir a quien va dirigida la información).
- Se pueden utilizar medios de transmisión mixtos.
- Se posee un control central, lo que facilita la detección de fallas.

DESVENTAJAS:

- Es muy vulnerable a fallas en el nodo central.
- Se requiere de tecnología compleja en el nodo central de la red (múltiples puertos de entrada/salida).
- La cantidad de cable que se utiliza es considerable.

o CICLO.

La manera de conectar a los anfitriones en una topología de ciclo es similar a la utilizada en la de canal, (fig 1.6) al igual que ésta, utiliza comunicación por difusión. De hecho el ciclo es como un canal cuyos extremos se unen. Generalmente las redes en ciclo están provistas de un nodo o anfitrión especial que se encarga de controlar la red. Sin embargo esto no tiene por qué ser así siempre, se pueden utilizar mecanismos de comunicación como los usados en las redes en canal sin jerarquías.

VENTAJAS:

- Es fácil conectar nuevos equipos a la red.
- Es posible que el conectar o desconectar anfitriones de la red no interrumpa el tráfico entre los anfitriones restantes.

DESVENTAJAS:

- Por lo general la red depende de un controlador para su correcta operación; si éste falla se trastorna la funcionalidad del sistema.

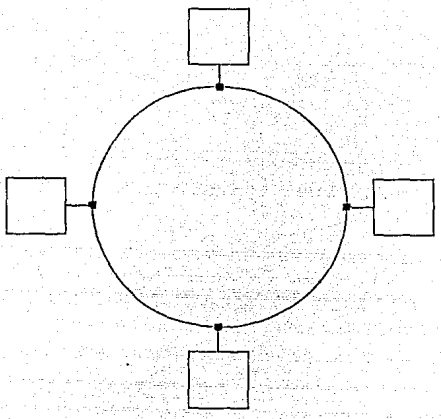


Fig. 1.6: Topología de ciclo.

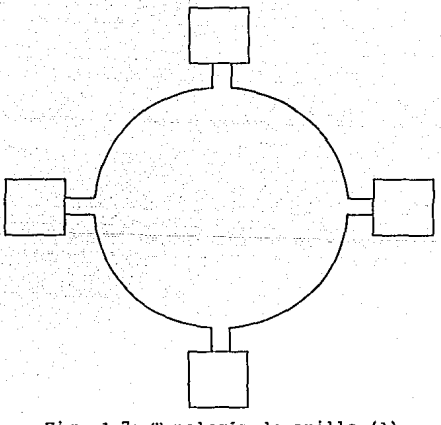


Fig. 1.7: Topología de anillo (1).

o ANILLO (1).

Este tipo de configuración se muestra en la fig 1.7. Como puede apreciarse, el tipo de comunicación usada en estos casos es punto a punto. En la mayoría de los casos los datos en el anillo fluyen en un sólo sentido. La organización lógica de una red de este tipo es sencilla, debido a que son muy elementales las tareas que los anfitriones tienen que realizar. Solo tienen que aceptar la información que llegue por el canal y retransmitirla en caso necesario al siguiente anfitrión en la red.

VENTAJAS:

- No se depende de un dispositivo central para el control de la red.
- Los problemas de ruteo de mensajes son triviales.
- Se pueden tener medios de transmisión mixtos.

DESVENTAJAS:

- La confiabilidad de las transmisiones depende de la totalidad del anillo y de los anfitriones conectados.
- Usualmente es necesario un anfitrión monitor de la red.
- La inclusión de un nuevo anfitrión ocasiona interrupción temporal en la operación de la red.

o ANILLO (2).

Para evitar algunos de los problemas típicos de la configuración en anillo ya mencionada, se ha creado una configuración que, en esencia, es igual a la anterior salvo que es posible redefinir la ruta de los datos en la red cuando algún anfitrión repentinamente deje de operar (fig. 1.8). Para ello las líneas que enlazan a cada anfitrión con sus vecinos pasan por un lugar central común, en caso de que un nodo deje de funcionar se conmutan los circuitos dentro del control central para excluir a

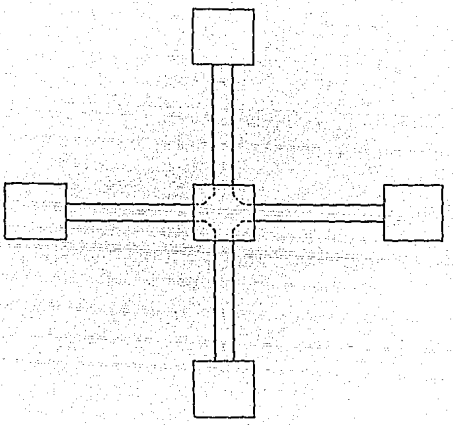


Fig. 1.8: Topología de anillo (2).

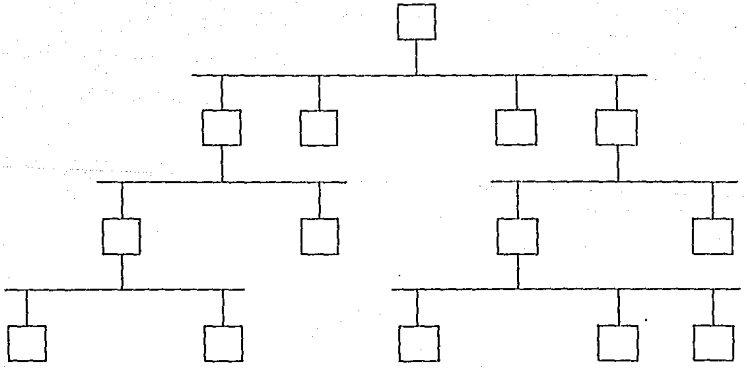


Fig. 1.9: Topología de árbol.

dicho nodo de la ruta de la información. La más popular de las redes de área local con topología de anillo es de éste tipo; su diseño pertenece a IBM y se le denomina *Token Ring* (paso de estafeta en anillo), debido a su mecanismo de acceso al medio.

VENTAJAS:

- Los problemas de ruteo de mensajes son trivialmente simples.
- Se pueden tener medios de transmisión mixtos.
- No se afecta la transmisión de información en caso de que un anfitrión falle.

DESVENTAJAS:

- Usualmente es necesario un anfitrión monitor de la red.
- La longitud de cable utilizado es considerable.
- Se depende del conmutador central para asegurar el correcto funcionamiento de la red.

o ARBOL.

Esta topología es llamada también jerárquica (fig. 1.9) y esencialmente consiste de una serie de canales interconectados. Usualmente el anfitrión que se encuentra en la raíz del árbol es el que se encarga de controlar la totalidad de la red. Esta topología ofrece las mismas ventajas y desventajas que la de canal.

o IRREGULAR.

Esta es la configuración más usual en redes de cobertura extensa (WAN) debido a que en dichas redes un anfitrión puede conectarse directamente con muchos otros, dado que las líneas de comunicación generalmente no son permanentes (fig. 1.10). Cuando cada nodo de la red está conectado con todos los demás se dice que la red es completamente conexa.

VENTAJAS:

- En caso de existir más de un camino entre dos nodos de la red, ésta es relativamente inmune a problemas de congestionamiento o descompostura de un nodo, dado que existen múltiples rutas de acceso por donde se puede orientar el tráfico.

DESVENTAJAS:

- Es difícil que exista control centralizado y por lo tanto detectar fallas en el sistema.

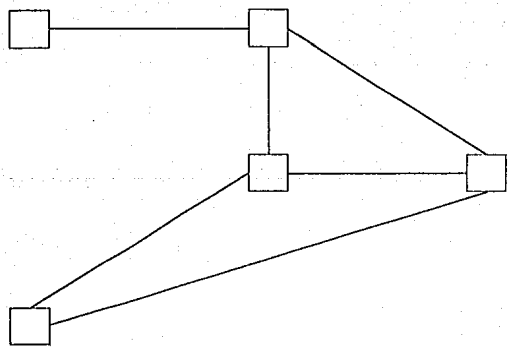


Fig. 1.10: Topología irregular.

1.3. ARQUITECTURAS DE RED.

Diseñar una red de computadoras es mucho más que planear la configuración física de sus componentes. La red ha de proveer a sus usuarios de un conjunto mínimo de servicios: intercambio confiable de datos (i.e. libre de errores) entre las máquinas que la integran; capacidad para comunicar cualesquiera dos anfitriones de la red sin importar que no estén directamante conectados, haciendo transparente para el usuario la elección de la ruta que han de tomar los datos y ocultando las diferencias que puedan existir en cuanto a la forma de representarlos (i.e. debe poderse intercambiar información entre una máquina que utiliza código ASCII y otra que utilice EBCDIC sin que el significado de la información se altere).

No es sencilla la tarea de diseñar una red de computadoras que proporcione los servicios que se han mencionado. Para facilitar esto lo que se hace es abstraer un problema o un conjunto de ellos y resolverlos independientemente de los demás. El tratar de solucionarlos plantea, con frecuencia, otros problemas que habrá que resolver también. De esta forma, gradualmente se van resolviendo problemas cada vez más sencillos hasta llegar a aquellos que tienen que ver con la transmisión física de la información entre dos computadoras.

El resultado de todo este proceso de diseño es una estructura organizada por capas o niveles. Cada una de éstas se construye sobre su predecesora utilizando los servicios que ésta le proporciona pero sin enterarse de como son implantados dichos servicios. El número de capas y sus nombres varían de una red a otra. La n-ésima capa de una máquina asume que se comunica directamente con la capa análoga de otra máquina; las reglas y convenciones utilizadas en esta comunicación se denominan *protocolo de la capa n*. Sin embargo esta comunicación es sólo virtual, en realidad la única capa que se comunica con su análoga

en la otra máquina es la más primitiva, la que se encarga de la transmisión física de la información a través del medio. Todas las demás se comunican con la que se encuentra inmediatamente debajo de ellas mediante llamadas a subrutinas con paso de parámetros; esto constituye la interfaz de las capas. En resumen, la capa n (n>1) de la máquina A sigue ciertas reglas en las que está de acuerdo con la capa n de la máquina B para intercambiar información, pero en realidad ésta es pasada a la capa n-1 de A, la cual a su vez la pasa a la capa inferior hasta que la información llega a la capa 1 de A, que la transmite físicamente al medio de comunicación de la red, de donde eventualmente es tomada por la capa 1 de B y pasada después a los estratos superiores hasta llegar a la capa n de B.

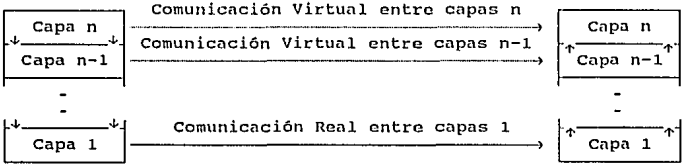


Fig. 1.11: Estructura de capas de una red.

Al conjunto de capas y protocolos que constituyen una red se le denomina *arquitectura de red*. En la actualidad existen múltiples arquitecturas en operación: SNA (*Systems Network Architecture*, IBM), DEC-Net (*DEC-Network, Digital Equipment Corporation*), X.25 (CCITT, *Consultative Commitee on International Telephony and Telegraphy*), ARPANET (DARPA, *Defense Advanced Research Project Agency*), etc.

CAPA	FUNCION
APLICACION (7)	Provee a los usuarios de acceso al ambiente de red.
PRESENTACION (6)	Provee a la capa de aplicación de medios para asegurar la integridad del significado de la información.
SESION (5)	Provee de organización y sincronización en el intercambio de la información.
TRANSPORTE (4)	Brinda servicios que hacen confiable la transferencia de información entre interlocutores o puntos terminales.
RED (3)	Hace confiable la transferencia de información punto a punto. Provee de funciones de ruteo.
ENLACE DE DATOS (2)	Provee de mecanismos para la transferencia de información con detección y corrección de errores. Agrupa la información en paquetes.
FISICA (1)	Provee de una transmisión transparente de cadenas de bits a través de la interconexión física entre los sistemas.

Tabla 1.1: Modelo de referencia OSI de ISO.

1.3.1. EL MODELO OSI.

Para evitar la anarquía en la definición de las arquitecturas de red y fijar parámetros que permitan homogeneizarlas lo suficiente como para interconectar redes de distintas arquitecturas, la ISO (*International Organization for Standardization*) creó el llamado modelo de referencia de interconexión de sistemas abiertos[3] (OSI, *Open Systems Interconnection*) éste se muestra esquemáticamente en la tabla 1.1.

[3] Un "sistema abierto" es aquel cuyo estado en un momento dado no depende sólo de él, sino que depende también de otros con los que interactúa.

El modelo de referencia de OSI es comunmente utilizado como un parámetro por el resto de las arquitecturas, una pauta a la que es recomendable apegarse para asegurar que una red determinada podrá interactuar con otras de arquitecturas diferentes. Sin embargo, es un modelo que no se ha terminado de definir en todos sus detalles: aún existe trabajo por hacer en la definición de los protocolos de algunas de sus capas.

1.3.2. LA ARQUITECTURA DE ARPANET.

Una de las arquitecturas de red de amplia difusión es la de ARPANET, ésta no sólo ya ha sido terminada, sino que además ha demostrado ser funcional y es, de hecho, la más utilizada en ambientes de red académicos y de investigación.

La arquitectura de ARPANET está constituida por cuatro capas, la primera denominada de *acceso a la red* se encarga de ofrecer servicios primitivos de bajo nivel, los que corresponderían a las capas 1,2 y parte de la 3 del modelo OSI. El resto de las capas de la arquitectura de ARPANET lo constituyen una serie de protocolos (de comunicación virtual entre capas equivalentes); a estos se les conoce genéricamente como protocolos TCP/IP. Inmediatamente sobre la capa de acceso a la red se coloca IP (*Internet Protocol*), el equivalente a la capa de red del modelo OSI. Sobre IP se encuentra alguno de los protocolos de nivel de transporte de ARPANET, estos son: un protocolo orientado a conexión, TCP (*Transmission Control Protocol*) y uno orientado a datagramas, UDP (*User Datagram Protocol*). Finalmente sobre estos se colocan los protocolos de nivel superior FTP (*File Transfer Protocol*), Telnet, SMTP (*Simple Mail Transfer Protocol*), los cuales conforman por sí solos aplicaciones, aunque no se encuentren propiamente (mejor dicho exclusivamente) a nivel de aplicación del modelo OSI.

La arquitectura de ARPANET es anterior al modelo de referencia OSI, por lo que no existe una analogía directa entre ellos, sin embargo, dado que el objetivo de ambas partes es el mismo, se puede establecer una equivalencia aproximada entre dichas arquitecturas; esta equivalencia se muestra esquemáticamente en la figura 1.12 [Stallings 90].

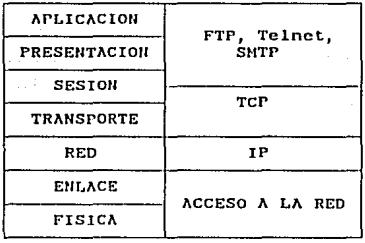


Fig. 1.12: Equivalencia aproximada entre el modelo OSI y la arquitectura de ARPANET.

Como dice Tanenbaum [Tanenbaum 91]: "lo bueno de tener estándares es que hay muchos de donde escoger"; aunque eso es justamente lo que se procura evitar al hacerlos. Se han presentado aquí dos estándares para arquitecturas de red, OSI y ARPANET, éste último es de los que se denominan *de facto*. Ambas arquitecturas han sido importantes en la realización de este trabajo de tesis. En el proyecto dentro del que se elaboró se ha tendido más a utilizar la arquitectura de ARPANET, sin embargo se hace referencia constante a OSI por ser la arquitectura mejor definida. Por otra parte el curso de Introducción a las Redes de Computadoras que se imparte en la Facultad de Ciencias y para el

que se han elaborado los módulos, está basado en el análisis detallado del modelo OSI. En el capítulo IV se volverá a mencionar la arquitectura de ARPANET y su relación con el trabajo desarrollado.

R E F E R E N C I A S

[Black 90]
Black, Uyless. *Redes de Computadoras*. Macrobit RA-MA. 1990. 421 pp.

[Comer 91]
Comer, Douglas E. *Internetworking with TCP/IP*. 2ª ed. Vol. 1 Prentice Hall. 1991. 547 pp.. 547 pp.

[Halsall 92]
Halsall, Fred. *Data Communications, Computer Networks and OSI*. 2ª ed. Addison Wesley. 1992.

[Holzman 91]
Holzman, Gerard. *Design and Validation of Computer Protocols*. Prentice Hall. 1991. 500 pp.

[Stallings 90]
Stallings, W. *Data and Computer Communications*. 2ª ed. MacMillan Press. 1990.

[Tanenbaum 91]
Tanenbaum,Andrew S. *Computer Networks*. Prentice Hall. 1991.

II ETHERNET

El trabajo de tesis se realizó en una red *Ethernet*, hoy por hoy la más popular de las redes de cobertura local. Este capítulo se avoca a la tarea de describir este tipo de redes para tener una idea clara del entorno en el que se realizó el trabajo.

2.1. CONSTITUCION FISICA.

2.1.1. GENERALIDADES.

Ethernet posee topología de *bus* o canal, por lo que presenta las características señaladas en el capítulo anterior para este tipo de configuraciones. En *Ethernet* no existen jerarquías entre los anfitriones de la red, todos tienen los mismos derechos y no existe ninguno que cumpla funciones especiales de control de la red. Una configuración típica de *Ethernet* se muestra en la figura 2.1. La definición de *Ethernet* establece que un segmento de red puede medir hasta 500 mts; la velocidad intrínseca de transmisión en *Ethernet* es de 10 Mbits/seg, degradable dependiendo de las condiciones de tráfico en la red y se pueden conectar hasta 5 segmentos consecutivos si se hace a través de dispositivos llamados *repetidores*. Estos sirven para interconectar 2 (y a veces más) segmentos, lo único que hacen es retransmitir lo que reciben por un canal hacia los demás amplificando la señal para contrarrestar la atenuación del medio.

En *Ethernet*, un anfitrión se conecta al medio común de comunicación mediante un cable que va de él hasta un pequeño dispositivo llamado *transceiver*, el cual se conecta directamente al canal. El término *transceiver* (transceptor) es una contracción de *transmitter-receiver*, (transmisor-receptor) y también se le conoce con el término AUI (*Attachment Unit Interface*).

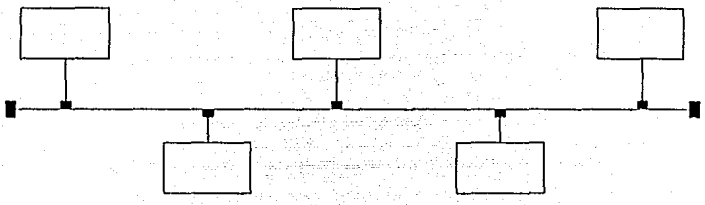


Fig. 2.1: Estructura típica de una red *Ethernet*.

Existen otras versiones de *Ethernet*, a la configuración estándar también se le llama 10BASE5 por ser a 10 Mbits/seg. *Base band* y $5x10^2$ Mts. de longitud máxima de segmento. El trabajo de tesis se hizo en una red mixta donde se mezclan la configuración 10BASE5 y la 10BASE2 (185 Mts. de longitud máxima de segmento) que utiliza cable más delgado y en ocasiones no requiere *transceivers*.

2.1.2. CABLE.

La versión original de *Ethernet*, diseñada por las compañías: *Digital Equipment Corporation* (DEC), Intel y Xerox (por lo que se le llama DIX Ethernet), utiliza cable coaxial como medio de comunicación. Este cable tiene un diámetro total de 1.02 cm, el conductor central es de cobre de 2.17 mm de diámetro, sobre él hay una cubierta de polietileno que lo rodea completamente hasta un diámetro conjunto que varía de 6.14 mm a 8.28 mm, sobre él se coloca una malla de conductor para proteger de campos externos el interior y rodeando todo ésto se encuentra finalmente el forro externo de teflón o cloruro de polivinilo (PVC) [Shoch 82], [Kauffels 89]. Este cable posee una impedancia característica de 50 Ohms ± 2 Ohms y la atenuación de una señal de 10 MHz en un

segmento de 500 mts no debe exceder de 8.5 dB. Esta atenuación es muy baja, por lo cual se hizo la analogía entre el cable coaxial y el éter luminífero (*ether* en inglés) utilizado en el pasado para explicar la propagación de la luz en el vacío. De esta analogía proviene el nombre de la red, una "red de éter" (*ETHER NETwork*).

Como ya se mencionó existen dos configuraciones de *Ethernet* que utilizan cable coaxial, en el párrafo anterior se ha hablado de la llamada 10BASE5 o *thick Ethernet* (*Ethernet* grueso). En la configuración 10BASE2 el cable es mucho más delgado (aproximadamente tiene un diámetro de 0.5 cm.), por ello también se le ha denominado a esta configuración *thin Ethernet* (*Ethernet* delgado). Mientras más delgado es un cable más resistencia ofrece, por ello la longitud máxima del segmento en 10BASE2 se ve reducida a 185 mts. La impedancia característica del cable delgado debe ser de 50 Ohms aproximadamente, es decir, la misma que la del cable grueso.

2.1.3. TRANSCEIVERS Y TERMINADORES.

En un segmento de cable grueso de 500 mts se pueden conectar más de 100 *transceivers*, siempre y cuando la distancia entre ellos sea al menos de 2.5 mts. lo cual reduce la probabilidad de que ocurran condiciones indeseables en la señal que viaja por el canal.

Los *transceivers* se conectan al anfitrión mediante un cable que puede llegar a medir hasta 50 mts. y contiene cinco cables de par trenzado: dos para datos de entrada y salida, dos para control de entrada/salida y uno para alimentación del transceiver desde la computadora [Tanenbaum 91].

Siempre que se transmite una señal por un cable coaxial ésta se propaga a lo largo del cable y al llegar al extremo parte de la señal se pierde, pero otra parte se refleja en el borde y viaja en sentido contrario de la señal original superponiéndose a ésta y ocasionando interferencia. Para evitar este problema se debe

colocar en los extremos del cable una resistencia de valor igual o mayor al de su impedancia característica. Es por eso que en los extremos de un segmento de *Ethernet* se colocan resistencias de 50 Ohms que reciben el nombre de terminadores (*taps*).

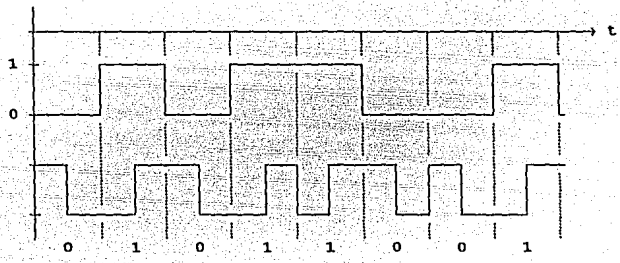


Fig. 2.2: Representación de la cadena de *bits* 01011001 en señalización binaria (arriba) y en señalización Manchester (abajo).

2.1.4. CODIFICACION MANCHESTER.

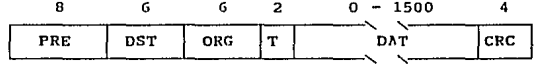
La codificación[1] Manchester es utilizada para señalizar los *bits* en el cable coaxial. El intervalo de tiempo utilizado en *Ethernet* para representar un *bit* es de 100 nanosegundos, a esto se le llamará un periodo de *bit*. La señalización Manchester se basa en el hecho de que es más fácil detectar una variación brusca en el nivel de voltaje que un nivel de voltaje por sí mismo. Lo que se pretende entonces es asegurar que en cada periodo de *bit* ocurra una transición de voltaje que indique qué *bit* es el contenido en

[1]Se ha utilizado aquí el término codificación por ser el más comunmente utilizado, sin embargo, como se verá, el mecanismo descrito corresponde a una señalización y no a una codificación donde se representa un conjunto de símbolos mediante otro.

el periodo. De esta manera, en *Ethernet* un paso de voltaje bajo (-0.85 V) a alto (+0.85 V) representa un 1 mientras que la transición inversa representa un 0 (Fig 2.2.).

2.2. FORMATO DE PAQUETE.

En *Ethernet* la información viaja empaquetada por el canal. A los datos que originalmente se desean transmitir se les añade información útil para que puedan llegar a su destino y para que el destinatario pueda verificar si llegaron correctos o sufrieron alteraciones en el camino, todo esto se transmite formando un paquete. El formato de DIX *Ethernet* de los paquetes que viajan en la red es el siguiente [Metcalfe 76]:



PRE : Preámbulo.
DST : Dirección *Ethernet* del destino.
ORG : Dirección *Ethernet* del origen.
T : Tipo de paquete.
DAT : Datos.
CRC : Código cíclico de redundancia.

El preámbulo del paquete es una secuencia de sincronización que consiste de 64 *bits* alternándose 1's y 0's y finalizando con dos 1's consecutivos, esta es una señal que por su uniformidad permite a la tarjeta de red prepararse para recibir información útil.

Las direcciones *Ethernet* son números de 48 *bits*, cada tarjeta

de red *Ethernet* tiene una dirección intrínseca asociada la cual es única en todo el mundo, ésta no puede ser cambiada; generalmente se encuentra grabada en un chip de ROM. Cuando una máquina recibe un paquete examina la dirección de la máquina destino, en caso de que coincida con la suya acepta el paquete, en caso contrario lo ignora. Existen algunas convenciones en cuanto a las direcciones, por ejemplo, si un paquete tiene como dirección destino solo 1's el paquete está destinado a todas las máquinas (*broadcast*); si esta dirección solo comienza con un 1 entonces el paquete está destinado a un grupo de máquinas (*multicast*).

El campo de tipo de paquete contiene un código que indica cual de todos los protocolos que se construyen sobre *Ethernet* es el que envía el paquete. Es decir, basta con observar el campo de tipo de un paquete de *Ethernet* para saber qué protocolo lo envió. Cada protocolo tiene un número único asociado.

El campo de CRC contiene un código que permite la detección de errores en el paquete, concretamente en los campos DST, ORG, T y DAT. Este código se calcula de la siguiente forma: considérese la cadena de *bits* de los campos DST, ORG, T y DAT concatenados. Se pueden considerar los *bits* de esta cadena como los coeficientes de un polinomio de grado a lo más 12111 (1500*8 + 2*8 + 2*6*8 - 1); sea D(x) este polinomio. Si ahora agregamos 32 *bits* a cero a la derecha de la cadena que asociada con D(x) esto representará el polinomio: $P(x) = x^{32} D(x)$. Ahora dividamos P(x) entre G(x) donde:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{0} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Al hacer esto se obtiene un residuo R(x) que se coloca en el campo de CRC (el término de más alto orden corresponde al primer *bit* del campo, el primero que se transmite). El colocar R(x) en el campo CRC genera un nuevo polinomio, resultado de concatenar D(x) y R(x), esto es equivalente a hacer la resta (módulo 2, es decir, sin acarreos) de P(x) y R(x); evidentemente dado que:

$$P(x) = Q(x)\ G(x) + R(x)$$

para algún polinomio Q(x), entonces:

$$P(x) - R(x) \text{ es divisible por } G(x).$$

Cuando una tarjeta de red *Ethernet* recibe un paquete hace operaciones que equivalen a dividir el polinomio asociado con la concatenación de los campos DST, ORG, T, DAT y CRC (es decir P(x) - R(x)) entre G(x) . Si esta división genera algun residuo entonces el paquete sufrió alteraciones durante el trayecto y debe ser descartado.

En *Ethernet* es necesario que exista un breve intervalo de tiempo entre la transmisión de un paquete y otro, este periodo es de 9.6 μsegs, el tiempo mínimo (estándar) que se necesita para que un controlador de *Ethernet* pueda volver a escuchar el canal después de haber recibido un paquete. El tiempo que tarda la señal en hacer un viaje redondo en una red de longitud máxima, (2.5 Km con 4 repetidores), es de 51.2 μsegs (512 periodos de *bit*).

2.3. C S M A / C D .

Dado que en *Ethernet* todos los anfitriones de la red tienen los mismos derechos y se posee un canal de comunicación común, puede ocurrir que más de un anfitrión intente transmitir en un instante determinado, en este caso las señales en el canal se interfieren mutuamente y se dice entonces que ha ocurrido una colisión.

Para evitar en la medida de lo posible que ocurran colisiones y para detectarlas y poderlas manejar, *Ethernet* posee un mecanismo de acceso al medio conocido como CSMA/CD (*Carrier Sense, Multiple*

Access / Collision Detection o bien sensibilidad al medio[2] con acceso múltiple / detección de colisión). Este mecanismo ha sido consagrado por IEEE en su estándar IEEE-802.3.

La manera de operar de CSMA/CD se puede ilustrar bien con el siguiente ejemplo [Scolofsky 91]: Imaginemos a un grupo de gente hablando en un cuarto pequeño y completamente obscuro. En esta analogía lo que se transmite son ondas sonoras en el aire en vez de señales eléctricas en un cable. Cada persona puede oir lo que las demás dicen (sensibilidad al medio). Todas en el cuarto tienen la misma capacidad y derecho de hablar (acceso multiple), pero por educación ninguna habla demasiado y ninguna habla mientras otra está hablando. Pero puede ocurrir que dos personas accidentalmente empiecen a hablar al mismo tiempo, éstas se darán cuenta de ello porque siempre escuchan todo lo que se dice (detección de colisión). Cuando se dan cuenta de lo que pasa ambas se callan y esperan un momento, hasta que alguna de ellas se decida y empiece a hablar, las otras escuchan lo que dice y esperan a que termine para empezar a hablar.

CSMA/CD opera de la misma manera. En *Ethernet* todas las máquinas comparten un canal de comunicación común; cuando una máquina pretende transmitir algo "escucha" el canal, si alguna otra máquina lo está ocupando, espera antes de intentar transmitir, si no, lo hace en ese momento. Pero si alguna otra máquina también pretendía transmitir y estaba esperando a que se desocupara el canal transmitirá al mismo tiempo, así que ocurrirá una colisión. Cuando esto pasa las computadoras involucradas en la colisión se dan cuenta de ello después de a lo más 51.2 μseg. (se detecta una colisión cuando lo que se envía es distinto de lo que se recibe). Inmediatamente transmiten de 4 a 6 octetos más, violando la señalización Manchester, para que sea del todo evidente que ha ocurrido una colisión, a esto se la llama CCEP

[2] Aunque "carrier" se traduce generalmente como "portadora", en realidad no se puede hablar de una portadora como tal dado que la transmisión es en banda base; es más correcto traducir "sensibilidad al medio".

(*Collision Consensus Enforcement Procedure*). Después de esto todas las estaciones conectadas se callan y esperan un tiempo aleatorio para volver a intentar transmitir, el tiempo de espera se calcula mediante un algoritmo conocido como regresión exponencial binaria (*binary exponential backoff*). En general después de *i* colisiones se selecciona un número aletorio entre 0 y $2^i - 1$ y la computadora se esperará ese mismo número de veces 51.2 μseg. antes de volver a escuchar el canal para intentar transmitir. En la práctica la cota superior del intervalo no crece indefinidamente sino que al llegar a 8 o 10 colisiones sucesivas el intervalo queda fijo. Después de 15 intentos se deja de intentar tratar de transmitir y se reporta un error a la capa de nivel superior.

El algoritmo de CSMA/CD es el siguiente:

```
0.....  s = 51.2 μsegs.
1.....  col = 0
2.....  SI hay paquete para transmitir ENTONCES
3.....     MIENTRAS  esté ocupado el canal
4.....        espera
5.....     transmisión durante tiempo  s
6.....     SI  (NO colisión)  ENTONCES
7.....        continua transmisión
8.....     SI NO
9.....        col = col + 1
10.....       SI col = 15 ENTONCES   VE A  16
11.....       transmisión de ráfaga de ruido
12.....       r = 2^min (col, 8) - 1
13.....       elige aleatoriamente  k ∈ {0,1,...,r}
14.....       espera tiempo  t = s * k
15.....       VE A 3
16.....  FIN.
```

2.4. CONFIGURACION DE LA RED LOCAL.

El trabajo fue elaborado, como se ha dicho antes, en una red *Ethernet*. El segmento local de la Facultad de Ciencias de la UNAM está constituido de 8 máquinas, 4 de las cuales pertenecen al proyecto de Sistemas Distribuidos en Redes y Sistemas Operativos Heterogéneos, estas máquinas son:

> Dos computadoras *Gateway 2000*, con procesador 80386 a 25 MHz, 4 Mb de RAM, 100 Mb en disco duro y tarjetas de red Western Digital Ethercard Plus 16 *bits* (WD8013), con sistema operativo MS-DOS Ver 4.1.
>
> Una SUN Sparc Station IPC y una ELC con tarjetas *Ethernet* propias y sistema operativo propio basado en UNIX, (Solaris).

Las computadoras utilizadas en este trabajo fueron las *Gateway* 386.

Anteriormente se mencionó que existen diferentes versiones de *Ethernet*, en particular se habló de una configuración que utiliza cable delgado, conocida como 10BASE2. Es pertinente señalar que en la Facultad de Ciencias se posee una red mixta: algunas computadoras se conectaron por medio del cable grueso y *transceivers*, mientras que otras se enlazaron mediante cable delgado y sin *transceivers*; en particular las *Gateway* se conectaron con cable delgado.

Un esquema de la configuración de la red en la Facultad se muestra en la fig. 2.3.

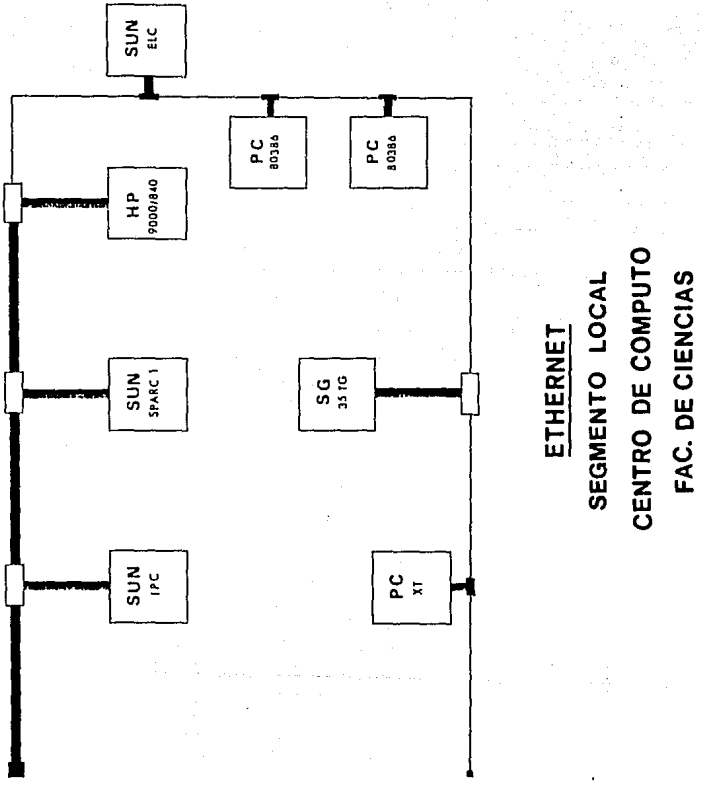


Fig. 2.3: Esquema de la configuración del segmento local de la red *Ethernet* en la que se llevó a cabo el trabajo de tesis.

R E F E R E N C I A S

[Kauffels 89]

Kauffels, F.J. *Practical LAN's Analysed.* Ellis Horwood Limited. 1989. 334 pp.

[Metcalfe 76]

Metcalfe, Robert M., David R Boggs. *Ethernet: Distributed Packet Switching for Local Computers Networks.* Communications of the ACM. ACM. julio 1976, Vol 19, No 7. pags 395-404.

[Scolofsky 91]

Scolofsky, T., C. Kale. *A TCP/IP Tutorial.* RFC 1180. Network Working Group. 1991. 28 pp.

[Shoch 82]

Shoch, John F., Yogen K. Dalal, David D. Redell, Ronald C. Crane. *Evolution of the Ethernet Local Computer Network.* Computer. IEEE. agosto 1982, Vol 15, No 8. pags 10-27.

[Tanenbaum 91]

Tanenbaum, Andrew S. *Computer Networks.* 2ª ed. Prentice Hall. 1991.

III LA INTERFAZ DE RED

3.1. LA TARJETA DE RED Y SU MANEJADOR.

3.1.1. LA TARJETA.

Las máquinas en las que se hizo el trabajo de tesis poseen tarjetas de red *Western Digital Ethercard Plus Elite 16* (*WD8013*), herederas de las *WD8003* y de hecho casi idénticas a éstas, salvo que poseen un *bus* de 16 *bits* en vez de uno de 8 como las *WD8003* y un área de memoria de 16 K en lugar de una de 8 K.

Estas tarjetas tienen posibilidad de conectarse a un segmento de *Ethernet* a través de un *transceiver*, como en la versión original de la red; o directamente, en caso de que el segmento esté constituido de cable delgado. Es decir se poseen dos puertos de salida, uno de 15 patas (*pins*) para un cable de *transceiver* (AUI) y otro para un conector (BNC) "T" de cable delgado. El puerto que se utilizó es el de cable delgado; en esta configuración un segmento puede medir hasta 185 metros (en vez de los 500 mts. del estándar). Con estas tarjetas se pueden conectar a *Ethernet* PC's de IBM o compatibles [Western 87].

La tarjeta de red puede generar interrupciones al procesador, para lo que se le asigna (en su configuración) un nivel de interrupción, esto es, una prioridad para interrumpir al procesador central. A este nivel se le conoce como IRQ (*Interrupt ReQuest*). El controlador de interrupciones programable de las PC's (8259) posee líneas de entrada por donde llegan al controlador todas las peticiones de interrupción que generan los periféricos, estas líneas son los IRQ's, indexados en orden inverso al de su prioridad. El controlador se encarga de discernir cual de todas

las peticiones que llegan en un momento dado tiene prioridad más alta y entonces pone en alto la señal de petición de interrupción del procesador 80X86. Si el procesador tiene habilitadas las interrupciones del exterior entonces responde al PIC (*Programmable Interrupt Controler*) que acepta la interrupción. El PIC coloca en el *bus* de datos del procesador un número *n* (el índice de IRQ más 8) que es el índice de la tabla de vectores de interrupción asociado al dispositivo en cuestión. El procesador saca 4 *bytes* a partir de la dirección *n**4 y ejecuta la rutina apuntada por esos 4 *bytes*, a la que se le conoce generalmente como rutina de servicio.

Además del nivel de interrupción, la configuración de la tarjeta incluye algunas otras características, la dirección de entrada/salida (*jumpers*) y la dirección base de RAM de memoria compartida (*software*) [Western 87].

3.1.2. MANEJADORES DE CLARKSON.

Para poder hacer programas que utilicen la comunicación a través de la red, es necesario contar con pequeños programas que sirvan de interfaz con la tarjeta. Es decir, es necesario contar con un programa que maneje la tarjeta de red y provea de servicios elementales. A estos programas se les conoce como manejadores de tarjeta o *drivers*.

Por lo general, las aplicaciones de red poseen sus propios manejadores de tarjeta interconstruidos, los cuales se adueñan de la tarjeta. Es decir no dejan que otro programa la utilice. Esto ocasiona que para cambiar de aplicación se tenga que reconfigurar la máquina y recargar el sistema (dar *boot*).

Una propuesta de solución a este problema la implantan los manejadores de tarjeta de la Universidad de Clarkson. Estos manejadores proveen una interfaz estándar para todos los paquetes de *software* que utilicen servicios de red. Esta interfaz es

independiente del *hardware* de comunicación que se posea, es decir todos los manejadores de Clarkson ofrecen la misma interfaz sin importar para qué tarjeta se hayan elaborado. De este modo, el manejador puede adueñarse de la tarjeta y permitir que más de una aplicación la use, siempre y cuando la aplicación conozca la interfaz ofrecida por el manejador. Además de esto, el uso de los manejadores de Clarkson permite que se construyan aplicaciones de red independientes de la tarjeta: si la aplicación se atiene a los servicios proporcionados por los manejadores de Clarkson, correrá igualmente bien sobre cualquier tarjeta contemplada en el paquete de manejadores [Doupnik 90].

La manera de operar de los manejadores es la siguiente: el manejador se instala a sí mismo como una rutina de servicio de interrupción, esto hace que su operación se parezca a la de la interrupción 21H del sistema operativo MS-DOS. El manejador queda residente y coloca su dirección de inicio en alguna entrada (*n*) de la tabla de vectores de interrupción. Cuando se desea invocar alguna de las funciones del manejador se debe solicitar la ejecución de la rutina de servicio asociada con la interrupción *n*. Cada una de las funciones provistas por el manejador tiene asociado un número, mismo que se debe colocar en un registro del procesador (AH), en caso de que esta función requiera para su ejecución de algunos parámetros, estos deberán ser colocados en otros registros.

Un usuario del manejador, antes que nada, debe establecer una relación con él. Debe informar al manejador que tipo de paquetes de datos desea que éste le pase una vez recibidos. De esta manera es posible que distintas aplicaciones de red soliciten el tipo de paquetes que necesitan sin que se enteren de aquellos que no les interesan. Cuando se hace esto, el manejador le entrega a la aplicación una identificación, una ficha que ésta tiene que mostrar cada vez que requiera algún servicio. Cuando se recibe un paquete de datos de la red, el manejador busca en una tabla los identificadores de los programas que requieren ese tipo de paquetes y procede a almacenarlos donde se le indique [FTP 89].

Se recordará que la tarjeta de red tiene una rutina de servicio de interrupción intrínsecamente asociada. El manejador de Clarkson solicita al programa usuario la dirección de una rutina que se ejecutará cada vez que arribe un paquete que le interese y que estará asociada con la interrupción intrínseca de la tarjeta, pero no será única, dado que cada aplicación que utilice el manejador poseerá una. Esta rutina debe entregar al manejador la dirección del *buffer* donde desea que sea guardado el paquete que se acaba de recibir [Nelson 89].

Las principales funciones de los manejadores de Clarkson son:

- Especificar el tipo de paquetes que se recibirán, al tiempo que se obtiene una identificación asignada por el manejador y se establece una rutina que se ejecutará cada vez que sea recibido un paquete del tipo especificado.
- Enviar paquetes de *Ethernet* por la red.
- Extraer la dirección física de la tarjeta de red.
- Establecer el modo de recepción:
    1. No se recibe nada.
    2. Se reciben solo los paquetes destinados a la tarjeta de red propia.
    3. Se reciben los paquetes del modo 2 más los que sean enviados en difusión.
    4. Se reciben todos los paquetes (modo promiscuo).
- Extraer información acerca del tipo de interfaz de red que se posee.

3.2. DISEÑO E IMPLANTACION DE LA INTERFAZ DE RED.

3.2.1. CARACTERISTICAS DE DISEÑO.

Sobre la interfaz proporcionada por los manejadores de Clarkson se construyó una interfaz de red que provee de los servicios elementales de comunicación. El objetivo de esta es proporcionar herramientas que permitan al alumno la experimentación e implantación de los conceptos que se le presentan en la parte teórica del curso de introducción a las redes de computadoras. Debido a que la interfazfue elaborada con fines didácticos debe poseer ciertas características, a saber:

- Que utlice un medio físico de transmisión que esté disponible.

  Para que los alumnos tengan fácil acceso a la red y puedan experimentar con ella.

- Que ofrezca un conjunto completo y suficiente de servicios para acceso a la red.

  Para que, una vez terminada, la interfaz pueda servir como una herramienta útil en la construcción de aplicaciones en ambiente de red.

- Que sea fácil de usar.

  Para que, al construir aplicaciones sobre ella, el programador no tenga que averiguar como funcionan las cosas y pueda dedicarse a pensar en la aplicación que pretende construir.

- Que posea una estructura organizada por capas.

  Para ejemplificar lo que se ve en la parte teórica del curso (modelo de referencia OSI) y para que su construcción sea más estructurada y por tanto fácil de mantener y depurar.

- Que sea lo suficientemente versátil como para soportar diversos tipos de aplicaciones, con distintos requerimientos, construidas sobre ella.

  Para que el alumno tenga libertad suficiente al elegir qué aplicaciones construir sobre la interfaz.

- Que su elaboración presente problemas representativos y verosímiles durante la etapa de diseño.

  Para relacionar al alumno con problemas reales del diseño e implantación de redes de computadoras.

3.2.2. IMPLANTACION.

La implantación de la interfaz para *Ethernet* se llevó a cabo en lenguaje C, por ser éste un lenguaje que facilita la programación a bajo nivel y porque posee la eficiencia necesaria en el código generado.

En primera instancia, se implantó una estructura que posee los mismos campos que un paquete de *Ethernet*. Mediante ella es más fácil manipular los paquetes que se reciben y envían por la red. Cabe señalar, respecto a esta estructura, que los *unsigned char* de C miden 8 *bits* y que LDETHER (la longitud máxima del campo de datos de un paquete de *Ethernet*) vale 1500. A la estructura se le ha llamado PAQUETHE por ser un PAQUete de ETHErnet.

```
/*
   Formato de PAQUete de ETHErnet
*/
struct PAQUETHE
  {
   unsigned char  dst[6];          /* dirección de destino */
   unsigned char  org[6];          /* dirección de origen */
   unsigned int   tipo;            /* tipo de paquete */
   unsigned char  datos[LDETHER]; /* campo de datos del paquete */
};
```

La interfaz elaborada proporciona los siguientes servicios a través de llamadas a funciones y paso de parámetros:

- Inicio de una interacción con el manejador de Clarkson, es decir, inicio de los servicios de red.
- Envio de paquetes de *Ethernet* hacia la red.
- Recepción de paquetes de *Ethernet* provenientes de la red.
- Establecimiento del tipo de paquetes que se desea recibir.
- Extracción de la dirección *Ethernet* de la tarjeta.
- Cierre de la relación entre el manejador de tarjeta y la aplicación que utiliza la interfaz, es decir, terminación de los servicios de red.

Los encabezados de estas funciones y algunas definiciones relacionadas se encuentran listados abajo, con una breve explicación acerca del significado de los parámetros de cada función.

```
/*
     Tipos posibles de recepción
*/
#define R_NADA      1   /* se ignoran todos los paquetes */
#define R_PROPIOS   2    /* se reciben solo los dirigidos a esta
                            tarjeta */
#define R_DIFUSION 3    /* 2 + los paquetes "en difusión" */
#define R_MULTI_L  4    /*3+los paquetes en "multicast" limitada */
#define R_MULTI_T  5    /* 3 + todos los paquetes en "multicast" */
#define R_TODOS    6    /* todos los paquetes */
```

```
int inicia (int num_int, int *num_man);

/*  Establece la interfaz de la aplicación y el manejador
de Clarkson.
     num_int   =   número de interrupción asociada al manejador
                   de Clarkson.
     num_man   =   número de identificación que proporcionará el
                   manejador a la aplicación que invoca "inicia"

   Regresa EXITO_FUNC en caso de que haya sido exitosa la
inicialización de la interfaz, en caso contrario regresa un
código de error:  MAL_MAN, NO_I_CLASE, NO_I_TIPO, NO_I_NUM,
MAL_TIPO, NO_ESPACIO, TIPO_EN_USO, MAL_MODO.
*/


int recibep (struct PAQUETHE *ap);

/* Deja en "ap" el último paquete recibido de la red.
     ap  =  apuntador al PAQUETHE a recibir

   Regresa EXITO_FUNC en caso de que haya sido exitosa la
operación, en caso contrario regresa un ERROR_FUNC.
*/


int enviap (int num_int, struct PAQUETHE *p, int  nocts);

/* Envía el paquete que es apuntado por "p".
num_int = número de interrupción asociada al manejador de
          Clarkson.
     p    = apuntador al PAQUETHE a enviar
  nocts = número de octetos en el campo de datos

   Regresa EXITO_FUNC en caso de que haya sido exitosa la
operación, en caso contrario regresa un ERROR_FUNC.
*/


int tipo_r (int num_int, int num_man, int tipor);

/* Establece el tipo de paquetes que de aplicación desea recibir

     num_int  = número de interrupción asociada al manejador de
                Clarkson.
     num_man   =  identificación entregada por "inicia" a la
                aplicación que invoca "tipo_r"
     tipor     = tipo de recepción (se puede pasar una constante de
                las definidas arriba)

   Regresa EXITO_FUNC en caso de que haya sido exitosa la
operación, en caso contrario regresa un código de error: MAL_MAN,
MAL_MODO.
*/
```

```
void xdirl (unsigned char far d[6]);

/* La función "xdirl" extrae la dirección Ethernet de la tarjeta
de la máquina donde se ejecute.
        d  =  arreglo de 6 caracteres donde será entregada la
              dirección ethernet solicitada.
*/


int termina (int num_int, int num_man);

/* Termina la relación entre la aplicación y el manejador de
Clarkson.
    num_int  = número de interrupción asociada al manejador.
    num_man  =  identificación entregada por "inicia" a la
                aplicación que invoca "termina"

   Regresa EXITO_FUNC en caso de que haya sido exitosa la
operación, en caso contrario regresa un código de error:
MAL_MAN.
*/
```

Todo esto se encuentra en un archivo denominado ETHER.H, el cual contiene todas las declaraciones "públicas" de la interfaz de red, es decir, todo aquello que es visible desde el exterior y que puede ser utilizado por los programas que así lo requieran. Además existe otro archivo: ETHER.C, que contiene la implantación de las funciones ya mencionadas, así como otra que no es visible desde el exterior y que no puede, ni debe, ser utilizada por los programas de aplicación. Esta función se llama cada vez que se recibe un paquete de interés para la aplicación y se encarga de especificarle al manejador de tarjeta la dirección del buffer donde debe dejarlo, así como de modificar una variable (bandera) para indicar que se ha recibido un paquete.

La función mencionada se llama recepción, recibe como parámetros los registros del procesador, el orden en que aparecen estos en el encabezado de la función es inalterable dado que la función ha sido especificada como tipo interrupt. El compilador de C genera un código específico para este tipo de funciones asumiendo que se han declarado los parámetros en el orden debido. La función recepcion fue declarada del tipo que se ha dicho debido

a que el manejador de Clarkson asume que la rutina, cuya dirección le es entregada, termina con un IRET (*Interrupt RETurn*) o con un RETF (*RETurn Far*) [Nelson 89]. El tipo de función interrupt termina siempre con un IRET, por lo que se ajusta a las especificaciones del manejador. El código de la función recepcion se muestra a continuación.

```
/* La funcion que aparece abajo se ejecuta cada vez que es
recibido un paquete de Ethernet. La dirección de esta rutina se
pasa al manejador de paquetes en "inicia". */

void interrupt recepcion (unsigned bp, unsigned di, unsigned si,
                          unsigned ds, unsigned es, unsigned dx,
                          unsigned cx, unsigned bx, unsigned ax,
                          unsigned ip, unsigned cs,
                          unsigned flags)
{
     if  (ax)
          bandera++; /* si AX no es cero, se recibió un paquete */
     else {          /* si AX = 0 */
          if (!bandera) {      /* y bandera = 0 */
               es = FP_SEG (buffrec);
               di = FP_OFF (buffrec);

                    /*ES:DI = dirección del buffer de recepción */
          }
          else {   /* si no */
               di = 0;
               es = ax;
          }
     }
}
```

Dado que el manejador se instala a sí mismo como una rutina de servicio de interrupción, la manera natural de llamar a sus funciones es mandando ejecutar la rutina de servicio de interrupción asociada con el manejador y colocando los parámetros necesarios en los registros del procesador. Esto se hizo utilizando las funciones **int86** e **int86x** del compilador de C de Borland®. También se puede lograr esto leyendo de la tabla de vectores de interrupción la dirección en donde está cargado el manejador y llamando por referencia a la rutina apuntada por esta

dirección. Se prefirió la primera alternativa, dado que la segunda presentaba algunos problemas en lo que al manejo de la pila se refería, específicamente en lo que tocaba al espacio reservado para variables locales dentro de algunas funciones.

Se incluye a continuación un pequeño programa que ejemplifica el uso de la interfaz:

```
/*
El presente programa envía paquetes a la red hasta que sea pulsada
una tecla. Los paquetes enviados poseen direcciones de orígen y
destino fijas.
*/
#include "ether.h"     /* Encabezados de la interfaz */
#include <conio.h>
#include <stdio.h>

main ()
{
   struct PAQUETHE p;  /* "p" es un paquete de Ethernet */
   int ninterr,nman,i; /* interrupción del manejador,
                          identificación y un índice */
   char estado,c;

   ninterr = 0X60;    /* Número de interrupción del manejador */

   p.dst[0] = 0X02;   /* Dirección Ethernet del destino */
   p.dst[1] = 0X60;
   p.dst[2] = 0X8C;
   p.dst[3] = 0X3B;
   p.dst[4] = 0X4D;
   p.dst[5] = 0X30;

   p.org[0] = 0X00;   /* Dirección Ethernet del orígen */
   p.org[1] = 0X00;
   p.org[2] = 0XC0;
   p.org[3] = 0X3B;
   p.org[4] = 0XAD;
   p.org[5] = 0X08;

   p.tipo = 0X0050;   /* Tipo del paquete a enviar */

   for (i=0; i<LDETHER; p.datos[i++]='A'); /* campo de datos = A*/

/* "inicia" relación con el manejador, éste nos entrega una
   identificación en "nman"*/
   inicia (ninterr,CUALQUIERA,&nman);

/* Mientras no se pulse una tecla se envían paquetes */
   do {
      estado = enviap (ninterr,&p,LDETHER);
```

```
   } while (!kbhit());

/* Se termina la relación con el manejador */
   termina (ninterr,nman);
}
```

En el ejemplo mostrado se puede ver la facilidad con la que se pueden enviar paquetes a través de la red. Es igualmente fácil recibir paquetes de ella, como se puede observar en el siguiente programa:

```
/*
 El presente programa recibe TODOS los paquetes que pasen por el
segmento local de ETHERNET y despliega en pantalla el contenido de
sus campos de direcciones y de tipo.
*/

#include "ether.h"     /* Encabezados de la interfaz */
#include <stdio.h>
#include <conio.h>

void escribep (struct PAQUETHE paq);

struct PAQUETHE p;
int ninterr,nman;
int  sale;

main ()
{
   ninterr = 0X60;  /* Interrupción asociada al manejador */
   inicia (ninterr,CUALQUIERA,&nman);

   do {
      sale = recibep (&p);      /* Se recibe un paquete en "p" */
      if (sale == EXITO_FUNC)   /* Si se pudo recibir algo */
         escribep (p);             /* se escribe */
   } while (!kbhit());      /* Mientras no se pulse  una  tecla */

   termina (ninterr,nman);
}

/* La función "escribep" escribe en una línea de pantalla la
información del encabezado de un paquete de Ethernet */
void escribep (struct PAQUETHE paq)
{
   int i;

   printf ("  Destino: ");
   for (i=0; i<6; i++)      /* Dirección Ethernet destino */
   {
      printf ("%02X",paq.dst[i]);
      if (i!=5)  putchar (':');
```

```
    }

    printf ("      Fuente: ");
    for (i=0; i<6; i++)      /* Dirección Ethernet orígen */
    {
       printf ("%02X",paq.org[i]);
       if (i!=5)  putchar (':');
    }

    printf (" Tipo: %04X",paq.tipo);  /* Campo de tipo */
}
```

Como puede verse se logró lo que se pretendía en cuanto a la sencillez con la que se puede hacer uso de la interfaz construida. Con el uso de las funciones que se implantaron, el programador puede dedicarse por entero a pensar en la aplicación que desea hacer y no ocuparse de como implantar los servicios de red para esa aplicación. Es más, podría ni siquiera saber para qué se usan algunos de los parámetros que le pasa a las funciones (por ejemplo "nman"), ni preocuparse por conocer su valor (como es el caso de los programas mostrados). Durante el diseño de la interfaz se procuró ocultar al usuario la mayor cantidad de información irrelevante, de tal forma que tuviera que saber lo mínimo posible para hacer uso de ella.

Después de elaborar la interfaz que se ha descrito se hizo otra que pretendía cumplir exactamente las mismas funciones que la que nos ha ocupado, solo que se quería obtener más modularidad y ocultamiento de información innecesaria para el usuario. Con esto en mente se pensó en elaborarla dentro del paradigma de la programación orientada a objetos. La intención era definir una serie de clases instanciables en C++ que proporcionaran los mismos servicios que ya se tenían implantados en C y que ofrecieran una interfaz aún mucho más agradable al usuario. Desgraciadamente se tuvieron problemas serios con el compilador de C++ utilizado, dado que no fue posible que admitiera del todo bien a la función recepcion como un miembro de clase debido a sus características especiales mencionadas.

R E F E R E N C I A S

[Doupnik 90]

Doupnik, Joe R. *Packet Drivers Made Simple*. Utah State University. 1990. 3 pp.

[FTP 89]

FTP Software Inc. *PC/TCP Packet Driver Specificatión*. Ver 1.09. 1989. 14 pp.

[Nelson 89]

Nelson, Rusell. Bob Clements. *Código Fuente Genérico y Específico del Manejador para WD8003E*. 1988-1989. 4420 líneas en ensamblador para 8086.

[Western 87]

Western Digital. *EtherCard PLUS Elite 16, High performance 16-bit Ethernet local area network adapter for all coaxial wiring (User Installation Guide)*. 1987.

IV APLICACIONES

Se construyeron varias aplicaciones con el fin de probar que la interfaz de red funciona y observar su desempeño. En este capítulo se presentarán dos de las aplicaciones construidas sobre la interfaz. Estas son las más interesantes, pero no las únicas, además de éstas se elaboraron algunos pequeños programas como los mostrados en el capítulo anterior.

La primera aplicación que se presenta es un analizador de tráfico para redes *Ethernet* con protocolos de ARPANET. La segunda es la implantación de un protocolo de transporte con el correspondiente *software* de capa de red sobre la interfaz.

4.1. INTRODUCCION: LOS PROTOCOLOS DE ARPANET.

A lo largo de este capítulo se hará referencia constante a los protocolos de ARPANET, (los cuales ya se habían mencionado en el capítulo 1), comunmente conocidos como protocolos de la familia TCP/IP. Por esta razón se decidió incluir un esquema jerárquico que muestra la organización de estos protocolos (fig 4.1).

ARP es un protocolo encargado de traducir direcciones lógicas (direcciones IP) en direcciones físicas (p.ej. direcciones *Ethernet*). IP es el ya mencionado protocolo correspondiente a la capa de red, sobre él se montan una serie de protocolos: ICMP para mensajes de control (como reporte de errores en la red); IGP, EGP y GGP, protocolos para *gateways*[1]; TCP, protocolo de transporte orientado a conexión y UDP no orientado a conexión (para intercambio de datagramas). Sobre estos dos últimos protocolos se construyen algunos otros, sobre TCP: Telnet para efectuar sesiones

[1] Un GATEWAY es una máquina encargada de interconectar dos redes de computadoras que de arquitectura diferente.

remótas; FTP para transferencia de archivos; SMTP para correo electrónico y NFS para compartir sistemas de archivos. Sobre UDP estan: SNMP para administración de redes y SUN-RPC para efectuar llamadas a procedimientos remotos, entre otros.

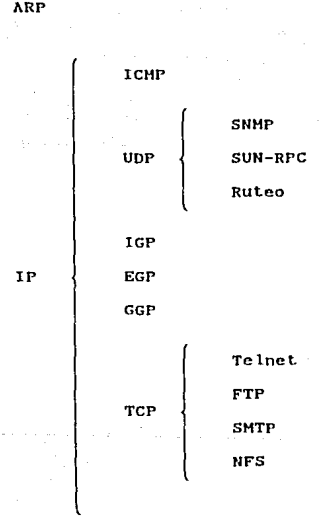


Fig. 4.1: Esquema jerárquico de la organización de los protocolos de ARPANET.

4.2. UN ANALIZADOR DE TRAFICO PARA REDES ETHERNET Y PROTOCOLOS DE ARPANET.

Cuando se elaboran o instalan programas que utilizan un ambiente de red, las cosas generalmente no salen bien a la primera y eventualmente después de que funcionan pueden estropearse. Siempre es bueno saber qué es lo que no funciona. Si se hace o instala *software* que constituye varias de las capas del modelo OSI ¿Cómo saber cual o cuales de las capas no funcionan? ¿Cómo saber si el problema es de *hardware* o de *software*?.

Por ejemplo, supóngase que se instala *software* de TCP/IP y que cuando se pretende efectuar una sesión remota mediante Telnet, desde la máquina A a la máquina B, A se "muere" (ya no responde a los comandos del usuario ni se realiza la sesión remota), ¿dónde está el problema? ¿B no contesta o A no trata de comunicarse con B? ¿se ha dañado la red físicamente o es problema del *software*?. Para facilitar la respuesta de preguntas como estas se elaboran herramientas tales como monitoreadores de redes y analizadores de tráfico. Si se posee un analizador de tráfico, esto es, un programa que capture todos los paquetes que viajan por la red y permita visualizar partes importantes de su contenido, es posible observar si A envía paquetes hacia B y si B hace lo propio, o bien si ninguno de los dos lo hace y entonces podría pensarse en un daño físico a la red. Un analizador de tráfico hace más fácil la tarea de depurar y analizar el desempeño de las distintas aplicaciones de red y encontrar problemas en la comunicación.

Por todo lo anterior, la primera aplicación que se construyó sobre la interfaz desarrollada fue un analizador de tráfico. Este programa captura todos los paquetes que viajan por el segmento local de la máquina en donde se ejecute. Una vez capturado, el paquete se clasifica de acuerdo con los protocolos de la familia de TCP/IP que lo han enviado, esta información se despliega en la pantalla junto con las direcciones IP de origen y destino del paquete en cuestión utilizando una línea para cada paquete.

Por ejemplo, supóngase que se ha recibido un paquete de IP, esto es fácil de detectar verificando el contenido del campo tipo (T, véase el capítulo 2) del paquete de *Ethernet* que lo contiene. Observando el contenido del encabezado de IP se puede saber que protocolo de transporte lo envía (TCP o UDP), una vez hecho esto se procede a analizar el encabezado del paquete de transporte y se determina qué protocolo superior lo ha enviado (p.ej. FTP, Telnet o SUN-RPC). En el monitor aparecerá una línea que dice de que dirección IP enviaron el paquete y cuál es su destino, que el paquete es de IP, TCP y FTP por ejemplo (fig 4.2).

Cuando se detiene la operación del analizador se muestran en la pantalla estadísticas obtenidas durante su periodo de ejecución. Cantidad total de paquetes recibidos, cantidad de paquetes de cada protocolo y su proporción respecto al total (fig 4.4). Posteriormente se muestra una gráfica de las condiciones de tráfico durante la ejecución del analizador.

Este programa divide el tiempo en intervalos regulares de *n* segundos. Durante cada uno de estos intervalos cuenta el número de paquetes que recibe para después hacer una gráfica de número de paquetes recibidos por intervalo, ésto proporciona una visión global de las condiciones de tráfico en el segmento durante el tiempo total de ejecución del analizador (fig 4.3). El valor de *n* es especificado por el usuario en un archivo de configuración, en el que también se da el número de interrupción asociada con el manejador de Clarkson.

A pesar de haber sido hecho solo con propósitos didácticos y de prueba, el analizador de tráfico ha resultado ser una herramienta útil para verificar el correcto funcionamiento de *software* de red. Su funcionamiento puede ser comparado dignamente con el de programas comerciales hechos para el mismo fin y actualmente se le utiliza con éxito en la Facultad de Ciencias y el Instituto de Fisiología Celular (ambos de la U.N.A.M.). Dado que ha sido construido sobre la interfaz de red que a su vez ha

sido construida sobre los manejadores de Clarkson, el analizador puede correr sobre cualquier tarjeta contemplada en el paquete de manejadores. De hecho ha sido probado en máquinas con tarjetas 3Com, NE1000, NE2000 y WD8003, además de las WD8013 ya mencionadas y en todos los casos ha funcionado bien. En máquinas con un procesador lento (8086) el analizador de tráfico pierde algunos paquetes, es decir no logra capturarlos por encontrarse procesando otro, esto también les ocurre en igual medida a otros programas analizadores de tráfico comerciales y no comerciales; en máquinas con mejor procesador (80386) este problema se reduce en un 90% [Galavíz 92].

```
132.248.28.3   -----> 132.248.28.1    IP   TCP   Telnet
132.248.28.1   -----> 132.248.28.3    IP   TCP
132.248.204.1  -----> 132.248.28.3    IP   UDP   SNMP
132.248.28.6   -----> 132.248.51.2    IP   TCP   FTP
132.248.28.3   -----> 132.248.28.1    IP   TCP   Telnet
132.248.28.3   -----> 132.248.28.1    IP   TCP   Telnet
132.248.28.3   -----> 132.248.28.1    IP   TCP   Telnet
132.248.28.3   -----> 132.248.28.1    IP   TCP   Telnet
132.248.51.2   -----> 132.248.28.6    IP   TCP   FTP
132.248.51.2   -----> 132.248.28.6    IP   TCP   FTP
132.248.51.2   -----> 132.248.28.6    IP   TCP   FTP
132.248.51.2   -----> 132.248.28.6    IP   TCP   FTP
132.248.51.2   -----> 132.248.28.6    IP   TCP   FTP
132.248.204.1  -----> 132.248.28.3    IP   UDP   SNMP
132.248.51.2   -----> 132.248.28.6    IP   TCP   FTP
132.248.51.2   -----> 132.248.28.6    IP   TCP   FTP
132.248.51.2   -----> 132.248.28.6    IP   TCP   FTP
```

Fig 4.2: Ejemplo de una pantalla mostrada por el analizador de tráfico (ANTRAF) durante su ejecución.

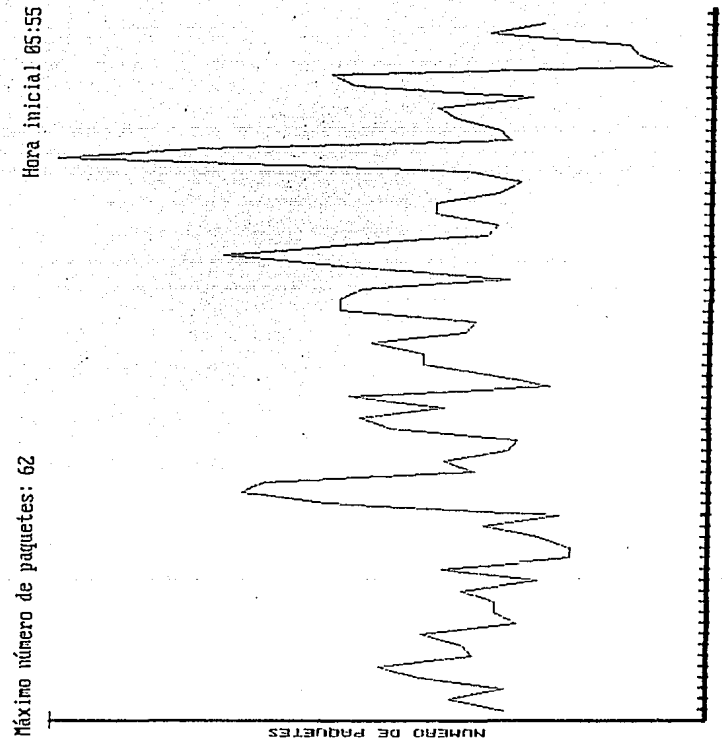


Fig. 4.3: Gráfica de intensidad de tráfico mostrada por ANTRAF.

```
Paquetes recibidos:  7472

IEEE 802.3:              204  = 3.5330 %
Loopback:                405  = 5.4200 %
ARP:                     590  = 7.8960 %
IP:                      6206 = 83.057 %
     ICMP:               385  = 5.1530 %
     TCP:                874  = 11.697 %
          FTP:           346  = 4.6310 %
          Telnet:        528  = 7.0663 %
          SMTP:          0    = 0.0000 %
     UDP:                4881 = 65.534 %
          NFS:           0    = 0.0000 %
          SNMP:          4701 = 62.915 %
          Ruteo:         139  = 1.8600 %
     SUN-RPC:            0    = 0.0000 %
```

Fig 4.4: Pantalla mostrada por el analizador de tráfico al detenerse.

4.3. UN PROTOCOLO DE TRANSPORTE.

Como parte de lo que se ha desarrollado en el proyecto dentro del cual se enmarca este trabajo, existe un protocolo de transporte. La evolución del proyecto hizo patente la necesidad de contar con *software* de red propio que permitiera adecuarlo a las necesidades que fueran surgiendo. La capa de transporte es la primera de las del modelo OSI que es independiente de las características de la red, es de fundamental importancia para crear aplicaciones transportables, es decir capaces de comunicar máquinas en redes heterogéneas. Por estas razones se decidió llevar a cabo la elaboración de un protocolo de transporte propio, este protocolo es un clon del TCP de ARPANET y su implantación

constituyó la tesis de licenciatura de otro integrante del proyecto [Leñero 92]. Sin embargo, a pesar de que se había concluido con la elaboración del *software* correspondiente al protocolo, éste no podía ser probado debido a que no existía *software* que lo sustentara, es decir, no se poseía ninguna de las capas inferiores a la de transporte.

Para solucionar este problema se intentó primero implantar el protocolo en una HP 9000/840 con sistema operativo HP-UX (UNIX), utilizando *sockets*, pero no fue posible programar a nivel de red, solo se podía acceder a servicios de transporte ya instalados en la HP. Se intentó despues utilizar una SUN Sparc Station IPC con sistema operativo SUN-OS (UNIX), pero se presentaron problemas similares. Así que se decidió implantar el *software* del protocolo en las *Gateway 2000* con MS-DOS 4.1 [Leñero 92].

La interfaz desarrollada en el trabajo de tesis se encuentra a nivel de enlace de datos, el protocolo que se deseaba implantar está a nivel de transporte, así que hacía falta aún montar, sobre la interfaz, *software* correspondiente al nivel de red, concretamente IP. Afortunadamente se poseía una versión vieja del *software* de dominio público desarrollado por Phil Karn, éste no es más que una implantación de TCP/IP, así que se procedió a hacer lo siguiente: se tomó la parte correspondiente a IP del *software* de Karn, se modificó su interfaz hacia abajo, es decir se cambiaron las estructuras de datos y los parámetros que esperaba recibir de las capas inferiores, de modo que se adaptara a los servicios y estructuras de datos utilizadas por la interfaz que se desarrolló. Para hacer esto se elaboraron una serie de funciones que tienen por objeto "traducir" las estructuras de datos utilizadas por Karn a las estructuras usadas por la interfaz (Ether). La más utlizada por Karn se denomina mbuf y la más utlizada por Ether es el PAQUETHE, a continuación se muestran las funciones elaboradas para pasar de un mbuf a un PAQUETHE y viceversa:

```
/* Funciones para traducir estructuras de datos. Karn utiliza
```

```
"mbuf's"  yo utilizo PAQUETHEs. */

/* Traducción de PAQUETHE a mbuf
nb = número de octetos de datos en "p"
p  = apuntador al PAQUETHE     */
struct mbuf *paq2mbuf (int  nb, struct PAQUETHE  *p)
{
   int  i;
   struct mbuf  *mb;
   unsigned char  uc;

   mb = alloc_mbuf (nb+14);  /* reserva memoria para mbuf */
   mb->cnt = nb+14;          /* número de octetos en mbuf:
               octetos de datos + 2 de tipo + 12 de direcciones */
   mb->data = p;
/* invertimos octetos 13 y 14 (indices 12 y 13) para obtener
                                                  "Big-Endian" */
   uc = *((mb->data)+13);
   *((mb->data)+13) = *((mb->data)+12);
   *((mb->data)+12) = uc;
   return (mb);
}

/*   Traducción de mbuf a PAQUETHE  */
void  mbuf2paq (struct PAQUETHE  *p, struct mbuf *mb, int  *nb)
{
   int i;
   unsigned char  *ac;

   for (i=0; i<LDETHER; p->datos[i++] = '\x0');

   *nb = (mb->cnt)-14;      /* Número de octetos de datos */
   ac = mb->data;
   for (i=0; i<6; i++) {    /* Copiamos direcciones */
      p->dst[i] = *(ac+i);
      p->org[i] = *(ac+i+6);
   }
/*  "Little-Endian"  /*
   p->tipo = (( (int) (*(ac+12)) )<<8) | *(ac+13);
   ac += 14;

   for (i=0; i<*nb; i++)    /* Copiamos datos */
      p->datos[i] = *(ac+i);
}
```

Además de esta "traducción" de estructuras de datos se eliminaron algunas cosas del IP de Karn. Por ejemplo: se dejaron fijos algunos parámetros que la interfaz establecía y ya no eran manejables al nivel de IP; se quitó el control de la cola de recepción de paquetes de red, esto también lo controla la interfaz y la capa de red ya no tenía que ocuparse de ello.

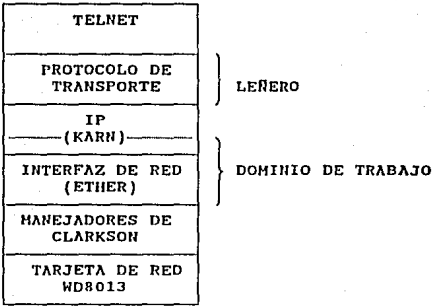


Fig 4.5: Esquema de la arquitectura obtenida en MS-DOS mediante la interfaz de red.

Una vez hecha la adaptación se procedió a montar sobre la interfaz, el *software* de Karn modificado y sobre éste el protocolo de transporte. Para probar el funcionamiento total se implantó Telnet sobre todo ello y el resultado fue muy satisfactorio, obteniéndose tiempos de respuesta comparables y hasta superiores a los del *software* comercial. El esquema de la arquitectura obtenida se muestra en la figura 4.5.

La implantación del protocolo de transporte trae consigo la posibilidad de crear aplicaciones más complejas que asuman que se les provee de mecanismos confiables de comunicación. Así pues, la interfaz desarrollada durante el trabajo de tesis ha ampliado, insospechadamente, el horizonte de posibilidades futuras para las aplicaciones que se elaboren dentro del proyecto de sistemas distribuidos.

Por otra parte, el hecho de que la interfaz haya sido capaz de soportar el *software* de IP denota que posee suficiente versatilidad y robustez; es posible construir confiablemente otras aplicaciones sobre ella.

R E F E R E N C I A S

[Galavíz 92]

Galavíz, Casas José y Salvador López M. *Diseño de Módulos para la Experimentación con Redes de Computadoras.* Memorias de la Octava Conferencia Internacional "Las Computadoras en las Instituciones de Educación e Investigación", DGSCA-UNISYS, noviembre de 1992.

[Leñero 92]

Leñero, Padierna Mónica. *Implantación de un Protocolo de Transporte en una Red Local.* Tésis de Licenciatura para obtener el título de matemático, Facultad de Ciencias, UNAM. octubre de 1992.

CONCLUSIONES

Cuando este trabajo fue desarrollado no se pensó en que los alcances del mismo pudieran ser tan significativos. En principio los objetivos eran puramente didácticos, conforme se fue avanzando se pensó en otras posibilidades no menos importantes que las iniciales. A fin de cuentas se obtuvo una interfaz de red que ha permitido:

- Integrar otros desarrollos elaborados dentro del pròyecto de Sistemas Distribuidos en Redes y Sistemas Operativos Heterogéneos.
- Elaborar herramientas útiles y competitivas para el control y monitoreo de redes reales.

Esto no hace que se aparte la vista de la intención inicial: la interfaz posee las características que se señalaron en el capítulo II y se espera probar su eficacia didáctica en futuros cursos de introducción a las redes de computadoras. Sin duda habrá que depurarla y hacerle modificaciones, esto estará en función de la respuesta de los alumnos ante ella y no en una búsqueda de eficiencia operacional. Es posible que más tarde se decida tener dos interfaces distintas, una para propósitos de enseñanza y otra como base para el desarrollo de aplicaciones más complejas.

En cuanto a la construcción de aplicaciones las perspectivas son las siguientes:

- Desarrollo de *software* sobre la interfaz. Por ejemplo: un programa de monitoreo de redes ya no solo de análisis del tráfico.
- Construcción de aplicaciones sobre la capa de transporte instalada sobre la interfaz y el IP de Karn. Por ejemplo: una implantación de SUN-RPC, lo que permitiría comunicar las SUN con las *Gateway* (interconexión de computadoras con sistemas operativos heterogéneos).
- Elaboración de una interfaz idéntica para SUN y sistema

operativo SUN-OS o Solaris, de tal forma que cualquier aplicación construida sobre la interfaz de MS-DOS se pueda ejecutar casi sin cambios en la otra.

Por todo lo anterior considero que los resultados obtenidos con este trabajo son satisfactorios ya que cumple y supera los objetivos que se plantearon inicialmente y ha favorecido el trabajo que se realiza dentro del proyecto del que surgió.