BORLAND DELPHI ▼ MULTITAREA EN C++

MAYO 1995 ▼ NÚM. 7 ▼ 950 PTAS.(CANARIAS 975 PTAS.)

RPP

REVISTA PROFESIONAL PARA PROGRAMADORES

SECCIONES:

- PROGRÁMATE TU OCIO:
  La Paleta y las Pantallas Virtuales
- UNIX:
  Tratamiento de Señales
- OS/2:
  Interoperabilidad y Multiprotocolo
- CLIPPER:
  Recursividad
- RINCÓN DEL PRINCIPIANTE:
  Redes de Área Local

MULTITAREA EN C++

ANALIZAMOS

BORLAND DELPHI

PATRONES DE DISEÑO

EXCEPCIONES EN WINDOWS

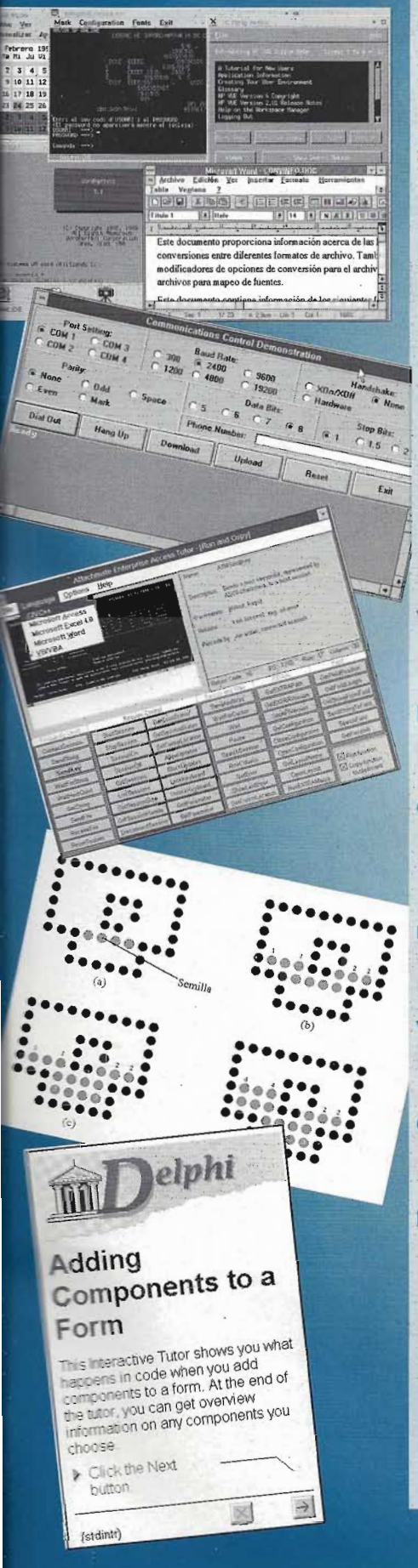
WINDOWS NT Y SUS CAPACIDADES GRÁFICAS

Disco en el interior con:
El Editor de Pantallas
E-SCREEN V.3.01a

ANAYA MULTIMEDIA

EN ESTE NUMERO ...

Mayo 1995 Número 7



Interoperabilidad de redes y multiprotocolo en OS/2 Warp 3.0

STE artículo es una introducción a la interoperabilidad de redes y el multiprotocolo en OS/2 Warp 3.0, y recoge algunas de las experiencias desarrolladas por el autor en diversas secciones de la Universidad de Barcelona (Servicios Informáticos de la UB, Servicio de Lengua Catalana y Escuela de Idiomas Modernos) durante los últimos cuatro años.

El problema de la interoperabilidad

Cada día más instalaciones informáticas se encuentran con la necesidad de explotar simultáneamente entornos de red distintos, y hacerlos interoperables entre sí. Una pequeña empresa puede tener sus bases de datos en un AS/400 y utilizar a la vez una red Novell o LAN Server; una gran empresa contará probablemente con un *mainframe* para las aplicaciones corporativas, redes locales en las distintas sucursales, y probablemente algunos *minis*. Las aplicaciones son caras y están optimizadas para el tipo de plataforma para el que fueron diseñadas, lo que en muchos casos hace inviable económica o técnicamente su migración a una única arquitectura. Por otra parte, la revolución microinformática de los últimos años ha puesto un microordenador en cada puesto de trabajo, y para esos entornos lo ideal es una red local. El mismo usuario deberá, en muchos casos, acceder a todos esos entornos, como mínimo secuencialmente, y a ser posible, simultáneamente. Es lo que llamamos interoperabilidad.

Además, los protocolos de comunicación utilizados diferirán en cada caso (IPX para Novell Netware, NetBIOS para LAN Server, LAN Manager y Windows for Workgroups, IP para el acceso a *hosts* Unix, SNA para el acceso a *mainframes* IBM,...). Una estación de trabajo ideal deberá ser capaz de manejar simultáneamente y de un modo transparente para el usuario todos esos protocolos de comunicación, de modo que el usuario pueda acceder a los diversos recursos que la red a la que tiene acceso le ofrece, sin necesidad de saber cuál es la tecnología subyacente, concentrándose así en *qué* desea hacer en vez de en *dónde* están los datos, o, mucho peor, en *cómo* acceder a ellos.

Una preocupación de este estilo es de importancia cada vez mayor a medida que crece la complejidad de un parque informático. Una pequeña empresa que disponga de una red Novell estará utilizando multiprotocolo si instala Windows for Workgroups 3.1 (IPX para Novell y NetBIOS para WfW), pero si se usa WfW 3.11 podrá operar tan solo con IPX. Acceder simultáneamente a redes Novell, *minis* Unix y unos cuantos AS/400 será algo más complicado, y la complejidad

José María Blasco

José María Blasco, nacido en 1960, es Licenciado en Matemáticas por la Universidad de Barcelona. Ha sido profesor de Programación en la Facultad de Informática de Barcelona, y Coordinador Nacional de la red EARN en Alemania. Ha trabajado como analista de sistemas para los Servicios Informáticos de la Univesidad de Barcelona y la empresa alemana GMD. En la actualidad trabaja como integrador de sistemas y consultor informático independiente. Sus temas de interés más recientes son las redes, entornos objetuales de trabajo y desarrollo, las plataformas de integración como OS/2, y las aplicaciones cliente/servidor. Su E-MAIL es: JMBLASCO@EIM.UB.ES

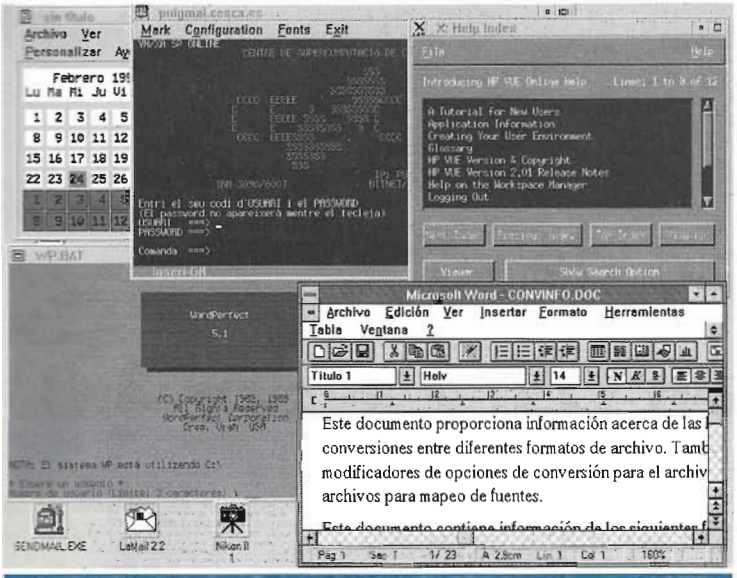


Figura A Acceso simultáneo a DOS, Windows, OS/2, un mainframe IBM 3090 con emulación de terminal 3270, y un mini Unix con X-Windows. La captura de pantalla es de 1992

crecerá si nos trasladamos a una gran empresa, en la que contaremos casi con seguridad con algún tipo de *mainframe*. El caso más complejo posible lo podremos encontrar en algunos entornos universitarios, donde a la diversidad propia de toda institución grande habremos de añadir la derivada de la variedad de equipos, producto de la autonomía de compra de los diversos departamentos.

Cuando nos enfrentamos a entornos de red complejos, la estabilidad de la plataforma se convierte en una cuestión crucial. Si la mayoría del trabajo lo realizamos en nuestro ordenador, y sólo de vez en cuando nos conectamos a una Novell para leer o grabar algún fichero, podemos tolerar que el ordenador se "clave" cada tanto por una *General Protection Fault*: rebotamos el ordenador, y ya está. Pero si una *clavada* significa restablecer comunicaciones con seis o siete servidores distintos (que además utilizan diversas tecnologías), varias bases de datos, y unas cuantas sesiones con *mainframes* y *hosts* Unix, la inestabilidad de la plataforma se convierte en un asunto mucho más grave.

> En DOS/Windows la interoperabilidad es un drama. Cada protocolo requiere de una serie de drivers específicos, que ocupan parte de la siempre escasa memoria que más tarde deberán utilizar los programas.

Necesitamos además poder realizar cualquier acción con nuestra estación de trabajo sin comprometer la estabilidad de las comunicaciones, y ser capaces de cancelar las comunicaciones *colgadas* sin afectar a las demás. Queremos decir que hemos de poder ver ese video que nos envió el jefe mientras estamos formateando un disco óptico o un diskette, y mientras se está realizando un *download* de FTP en una ventana y una reorganización masiva de una base de datos remota en otra, y todo eso con la asiduidad necesaria, y sin temer por los resultados de las demás sesiones si una de ellas nos da algún problema. Y, naturalmente, sin perder la posibilidad de usar las aplicaciones OS/2, DOS y Windows a las que estamos acostumbrados.

En DOS/Windows la interoperabilidad es un drama. Cada protocolo requiere de una serie de drivers específicos, que ocupan parte de la siempre escasa memoria que más tarde deberán utilizar los programas. Al instalar un nuevo *driver*, muchas veces dejan de funcionar los antiguos, o se pierde alguna configuración cuidadosa del CONFIG.SYS o del AUTOEXEC.BAT, o, aún peor, del WIN.INI. Una vez se han solucionado los problemas (si es que es posible hacerlo), nos encontramos con que el espacio máximo para ejecutar un programa ha decrecido hasta tal punto que se hace imposible cargar aplicaciones que antes funcionaban correctamente. En OS/2 nada de esto sucede, ya que es posible tener tantos *drivers* cargados como se quiera y disponer aún de 620K en cada sesión DOS para poder ejecutar con comodidad los programas que deseemos; esto se debe a que los drivers OS/2 se cargan en el espacio de memoria de OS/2, y los servicios de red se virtualizan en cada sesión DOS, sin ocupar apenas memoria.

Estas son algunas de las reflexiones que me impulsaron a considerar OS/2 como la plataforma idónea para la mayoría de los usuarios universitarios, tanto científicos como administrativos, desde la aparición de la versión 2.0. OS/2 2.0 apareció a la vez que Windows 3.1, y en aquel momento la interoperabilidad en Windows era simplemente imposible (la cosa no ha mejorado mucho desde entonces), y Windows como plataforma era sumamente inestable (cosa que no ha cambiado). Por otra parte, OS/2 ha madurado mucho desde entonces: hemos tenido ocasión de trabajar con OS/2 2.0 LA, OS/2 2.0 GA, OS/2 2.1, OS/2 2.11, y OS/2 3.0 Warp (versiones *for Windows* y *Full Pack*), además de las distintas versiones betas,

y sin mencionar los distintos *Service Packs*, allí donde Windows no ha producido ninguna versión nueva, si descontamos las versiones *for Workgroups* y Windows NT, que es una plataforma distinta, cara e incompatible. Y lo mismo ha sucedido con los productos de red y su capacidad para trabajar juntos. Desde 1992 hasta ahora se ha simplificado enormemente la instalación de los diversos productos de red disponibles para OS/2, además de haberse ampliado notablemente el soporte para distintos tipos de red, adaptadores, etc.

El multiprotocolo

Cuando dos programas residentes en ordenadores distintos necesitan comunicarse, deben estar de acuerdo en cómo van a hacerlo, es decir, en qué protocolo de comunicaciones van a utilizar. Para simplificar, cada protocolo es una especie de "lenguaje de red" distinto. Hay arquitecturas de red, como SNA de IBM, que tienden a una administración centralizada y priman la fiabilidad y la detección de errores: si un ordenador de comunicaciones de un banco deja de funcionar, una zona entera de oficinas puede quedar sin servicio, y por lo tanto es esencial asegurar la fiabilidad de las comunicaciones y saber cuáles son los equipos que hay que reemplazar si surge algún problema. Otras arquitecturas, como TCP/IP, que fue diseñado para el departamento de defensa de los Estados Unidos de América para operar fiablemente en tiempo de guerra, están pensadas para la administración descentralizada, y no importa tanto la fiabilidad como el intentar asegurarse de que las comunicaciones siguen funcionando aunque parte de la red deje de ser operativa: TCP/IP está pensado para seguir funcionando aunque algunos de los nodos caigan bajo fuego enemigo.

Igualmente, hay protocolos pensados para grandes redes (como el propio TCP/IP o SNA), y protocolos que no pueden ir más allá de una red local (como NetBIOS). Para cada uno de esos protocolos hay aplicaciones y servicios de red que los hacen útiles; en la Universidad de Barcelona, se utilizan todos ellos.

La necesidad de multiprotocolo aparece cuando intentamos acceder simultáneamente a dos o más servicios de red que utilizan protocolos distintos. Nótese que esto no sucede siempre que queremos acceder a dos servicios distintos: por ejemplo, LAN Server y DB/2 pueden *hablar* los dos NetBIOS o los dos IP (utilizando NetBIOS sobre IP), con lo que si éstas son las únicas aplicaciones que utilizamos no necesitamos del multiprotocolo, ya que sólo existe uno.

Cuando dos programas residentes en ordenadores distintos necesitan comunicarse, deben estar de acuerdo en cómo van a hacerlo, es decir, en qué protocolo de comunicaciones van a utilizar.

Cuando la misma placa de comunicaciones necesita manejar más de un protocolo, se hace necesaria la presencia de un *discriminador de protocolo*, que examine los paquetes que provienen de la red y los reencamine al driver y al programa adecuado. Con eso se consigue que cada protocolo pueda operar como si fuese el único que funcionase en la placa, implementando efectivamente lo que se conoce como una *pila de protocolos* o *protocol stack*.

Un ordenador con DOS y Windows conectado a una red Novell normalmente utiliza un *driver* (IPX) que se encarga de manejar los paquetes IPX utilizados para comunicarse con el servidor Netware. IPX controla en exclusiva la tarjeta de red, y, si llega un paquete perteneciente a otro protocolo, lo descarta o da un error. Por el contrario, cuando contamos con un protocol stack, el control de la tarjeta de red lo realiza un programa (en el caso de NDIS para OS/2, PROTMAN) que examina cada paquete, decide de qué protocolo se trata, y lo entrega al driver específico de ese protocolo. El *driver* no controla la tarjeta de red, sino que se comunica de forma ordenada con PROTMAN para permitir la operación simultánea de varios *drivers*, cada uno encargado de un protocolo distinto. Por ejemplo, Novell sobre NDIS en OS/2 cuenta con un *driver* (ODI2NDIS.OS2) que recoge los paquetes IPX de PROTMAN y los entrega a la pila Novell (LSL.SYS, IPX.SYS, SPX.SYS, etc), y viceversa.

TCP/IP está pensado para seguir funcionando aunque algunos de los nodos caigan bajo fuego enemigo.

En OS/2, el *protocol stack* más conocido es NDIS, aunque Novell proporciona una versión de ODI para OS/2, con una funcionalidad equivalente. El problema con ODI de Novell es que si se desea utilizar productos no Novell, que son casi todos NDIS, habremos de contar con un emulador de NDIS sobre ODI (la opción ODINSUP del cliente Novell para OS/2), con lo que terminamos por tener

protocolos sobre la pila NDIS, que a su vez se asienta sobre la pila ODI. Implementar Novell sobre NDIS requiere del mismo truco, pero al revés, ya que Novell sólo proporciona un cliente ODI para OS/2. En este caso, tenemos el cliente Novell corriendo sobre la pila ODI, que corre sobre la pila NDIS (se trata del driver *ODI2NDI.SYS* de IBM).

Nos decidimos por la segunda aproximación (es decir, NDIS y ODI sobre NDIS) por dos razones: la primera es que, excepto el cliente Novell, todos los productos que utilizamos (acceso a LAN Server, DB/2, emulación 3270, TCP/IP, etc.) están escritos para NDIS; la segunda es la deficiente calidad de soporte que en general hemos sido capaces de obtener de los revendedores de Novell.

NDIS, LAPS, MPTS

NDIS es una arquitectura multiprotocolo desarrollada por Microsoft, 3Com e IBM; la versión 2 es la propia de OS/2. Los primeros productos de IBM requerían la construcción a mano, o al menos la modificación, de los diversos ficheros que definen el acceso a la red, pero últimamente se está produciendo una convergencia de los programas instaladores en lo que primero se bautizó como LAPS (*LAN Adapter and Protocol Support*) y ahora se conoce (en la versión que viene con LAN Server 4.0) como *Anynet* o MPTS (*Multiple Protocol Transport Services*). Las noticias que tenemos de las *betas* de la versión cliente de OS/2 Warp 3.0 indican que probablemente esas capacidades estarán integradas en la versión base de ese producto.

Instalar soporte para multiprotocolo con el MPTS de LAN Server 4.0 es muy sencillo: basta con utilizar las diversas posibilidades que ofrece la pantalla de configuración del programa.

El programa nos permite elegir la(s) placa(s) de comunicaciones de que dispone nuestro ordenador (arriba a la izquierda) , y para cada una de las placas, los protocolos de comunicaciones que se utilizan sobre esas placas (arriba a la derecha). El resultado aparece en otro recuadro (abajo a la izquierda), desde donde podemos cambiar los parámetros propios de la placa o de cada uno de los protocolos de comunicaciones asociados con esa placa.

La interface gráfica es bastante sencilla, teniendo en cuenta la complejidad subyacente

Listado 1.- *Fichero PROTOCOL.INI*

```
[PROT_MAN]

   DRIVERNAME = PROTMAN$

[IBMLXCFG]

   ODI2NDI_nif = ODI2NDI.NIF
   TCPBEUI_nif = TCPBEUI.NIF
   TCPIP_nif = TCPIP.NIF
   EL3IBMO2_nif = EL3IBMO2.nif

[NETBIOS]

   DriverName = netbios$
   ADAPTER0 = tcpbeui$,0

[ODI2NDI_nif]

   DriverName = odi2ndi$
   Bindings = EL3IBMO2_nif
   NETADDRESS = "0020AF43EC77"
   TOKEN-RING = "no"
   TOKEN-RING_SNAP = "no"
   ETHERNET_802.3 = "yes"
   ETHERNET_802.2 = "no"
   ETHERNET_II = "no"
   ETHERNET_SNAP = "no"
   TRACE = 0x0

[TCPBEUI_nif]

   DriverName = tcpbeui$
   Bindings = EL3IBMO2_nif
   OS2TRACEMASK = 0x0
   SESSIONS = 130
   NCBS = 225
   NAMES = 21
   SELECTORS = 15
   USEMAXDATAGRAM = "NO"
   NETBIOSTIMEOUT = 500
   NETBIOSRETRIES = 2
   NAMECACHE = 0
   PRELOADCACHE = "NO"
   NAMESFILE = 0
   DATAGRAMPACKETS = 20
   PACKETS = 50

[TCPIP_nif]

   DriverName = TCPIP$
   Bindings = EL3IBMO2_nif

[EL3IBMO2_nif]

   DriverName = ELNK3$
   MaxTransmits = 20
```

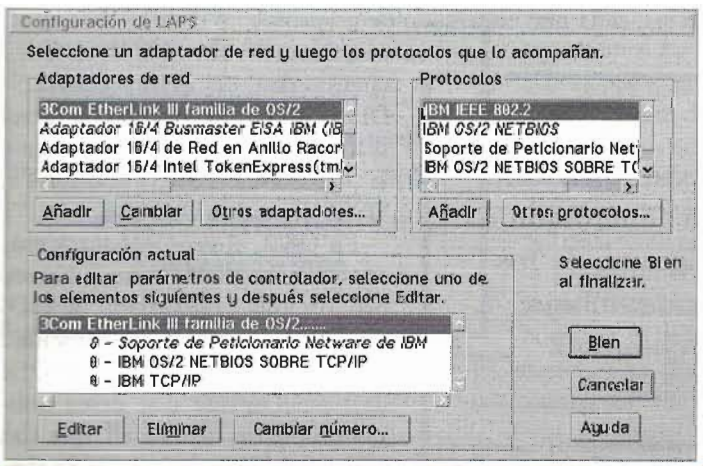


Figura B Pantalla de configuración de MPTS (LAPS)

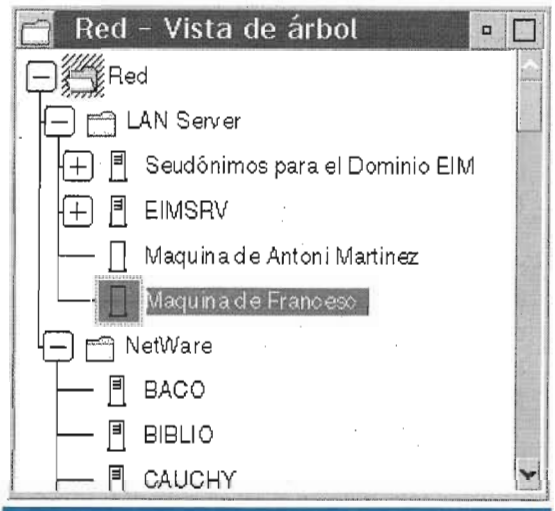


Figura C Novell Netware e IBM LAN Server se integran de un modo homogéneo en OS/2

del problema. La implementación también es bastante sencilla: los resultados de las elecciones del usuario se guardan en un fichero *ASCII* llamado PROTOCOL.INI, que se encuentra en el directorio C:\IBMCOM (suponiendo, como lo haremos en todo el artículo, que C: es la unidad de *boot* de OS/2), y en CONFIG.SYS; de sincronizar esos dos ficheros se encarga MPTS. Vamos a examinar el fichero PROTOCOL.INI correspondiente a la figura, comentando a la vez algunos de los efectos que tienen las distintas opciones en el CONFIG.SYS.

Hay que resaltar que este fichero ha sido generado automáticamente por MPTS; nosotros sólo tuvimos que especificar en el programa MPTS la dirección de placa y el tipo de frame para el cliente Novell, ya que éste lo requiere.

La sección [PROT_MAN] establece cuál es el administrador de protocolos o (*PROTocol MANager*). Este es, por defecto, el proporcionado por el sistema, PROTMAN.OS2:

```
DEVICE=C:\IBMCOM\PROTOCOL\LANPDD.OS2
DEVICE=C:\IBMCOM\PROTOCOL\LANVDD.OS2
DEVICE=C:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICE=C:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM
```

Las primeras líneas virtualizan la red para las sesiones DOS; la tercera añade soporte para mensajes y errores de red, y la cuarta es la referencia al mencionado PROTMAN.

La sección [IBMLXCFG] establece cuáles son las placas y los protocolos que se utilizarán en conjunto. Aquí encontramos referencias a ODI2NDI (ODI sobre NDIS, para el cliente Novell), TCPBEUI (para NetBIOS sobre TCP/IP), TCPIP (TCP/IP), y EL3IBMOS2, para el driver NDIS de la placa 3Com Etherlink III. Los ficheros NIF a los que se hace referencia guardan los valores por defecto para esas placas y esos protocolos, y se encuentran en C:\IBMCOM\MACS y C:\IBMCOM\PROTOCOL. En el CONFIG.SYS, estas lineas se traducen por inclusiones de los correspondientes *drivers*:

```
DEVICE=C:\MPTN\PROTOCOL\MPTN.SYS
DEVICE=C:\MPTN\PROTOCOL\LIPC.SYS
DEVICE=C:\MPTN\PROTOCOL\INET.SYS
DEVICE=C:\MPTN\PROTOCOL\IFNDIS.SYS
RUN=C:\MPTN\BIN\CNTRL.EXE  /P  mptn_os$
mptn_in$
CALL=C:\OS2\CMD.EXE /Q /C C:\MPTN\BIN\MPTS-
TART.CMD
RUN=C:\IBMCOM\PROTOCOL\NBTCP.EXE
DEVICE=C:\IBMCOM\PROTOCOL\TCPBEUI.OS2
DEVICE=C:\IBMCOM\PROTOCOL\ODI2NDI.OS2
DEVICE=C:\IBMCOM\PROTOCOL\NETBIOS.OS2
DEVICE=C:\IBMCOM\MACS\ELNK3.OS2
```

Los demás apartados de PROTOCOL.INI especifican los parámetros correspondientes a los protocolos y placas utilizados; si el usuario cambia algún parámetro, el cambio queda reflejado en PROTOCOL.INI, mientras que los ficheros .NIF originales no se cambian; MPTS realiza una fusión de los distintos .NIFs con PROTOCOL.INI, dando primacía a este último. Por ejemplo, en la sección para soporte Novell se observa la presencia de una dirección hardware de placa (requerida por Novell para el cliente OS/2). Se observará también la presencia de enlaces (*bindings*) de cada protocolo a una o más de las placas (son las líneas "Bindings =" de las secciones correspondientes a los protocolos).

NDIS es una arquitectura multiprotocolo desarrollada por Microsoft, 3Com e IBM

Una vez están cargados todos los *drivers*, se ejecutan las siguientes dos utilidades:

```
CALL=C:\IBMCOM\PROTOCOL\NETBIND.EXE
RUN=C:\IBMCOM\LANMSGEX.EXE
```

NETBIND activa las placas y los protocolos de comunicación, y avisa si hay algún error, a

Figura D

través de LANMSGEX. Si todo funciona bien, estamos preparados para trabajar, y disponemos de acceso a las placas y a los protocolos definidos. (No hemos entrado en el detalle para la instalación del cliente Novell, ni en la configuración de TCP/IP, que ocuparían demasiado espacio, aunque son igualmente sencillas).

Las interfaces de usuario

Obviamente, lo que hemos contado hasta aquí no tiene que ser manejado, en general, por el usuario, sino por el administrador de la red (El procedimiento puede incluso automatizarse, de modo que pueda instalarse una estación OS/2 con clientes de diversos tipos de modo totalmente desatendido, utilizando el sistema CID de IBM). Al usuario lo que le interesa es disponer de una interface clara y lo más homogénea posible. Por ejemplo, si está conectado a servidores Novell y LAN Server a la vez, le interesará que el número de diferencias en el momento del uso de los recursos que publican sea mínimo. En OS/2 el tema está bastante solucionado.

La **Figura C** muestra una vista de árbol de la red; cada tecnología de red tiene una carpeta, de la cual cuelgan los servidores accesibles; los servidores, a su vez, pueden abrirse (si se dispone de los privilegios adecuados para ello), y aumentan la profundidad del árbol correspondientemente. Como por ese método podemos ir abriendo sucesivamente carpetas hasta llegar al último subdirectorio, disponemos de una visión integrada de la red, una especie de Administrador de Ficheros de Windows, pero en serio, objetual, de red, y con la ventaja de que no es necesario asignar letras de unidad para operar con los ficheros (si disponemos de programas que soporten UNCs).

Si lo que deseamos es trabajar con letras de unidad, no hay nada más sencillo (**Figura D**).

Basta con señalar la carpeta de red deseada y asignarle una letra de unidad. Nótese que el procedimiento es el mismo independientemente de que estemos trabajando con Netware o con LAN Server.

El uso simultáneo de TCP/IP 2.0 y el IAK (por ejemplo para disponer de X-Windows y las utilidades como Gopher, WebExplorer o "Retrieve Software Updates" a la vez) requiere de un orden de instalación específico.

Conclusión y perspectivas

MPTS incluye NetBIOS sobre TCP/IP, para poder correr aplicaciones NetBIOS sin utilizar NetBIOS nativo (lo cual permite tener clientes de aplicaciones NetBIOS que estén fuera de la LAN o MAN de origen), y, curiosamente, un nivel de TCP/IP sobre NetBIOS, para correr aplicaciones TCP/IP en una LAN que ya está utilizando NetBIOS nativo sin incurrir en el esfuerzo extra de utilizar el multiprotocolo. La estrategia AnyNet de IBM permite correr programas APPC (una especificación API originalmente pensada para SNA) sobre TCP/IP. Algunas aplicaciones (como el Person-to-Person [P2P] incluido en el BonusPack de OS/2 Warp) pueden comunicarse indistintamente mediante IP, IPX o NetBIOS, o un módem, dependiendo del software instalado; la funcionalidad es la misma. Éstas parecen ser las tendencias generales del software: por una parte, soporte y coexistencia de todos los protocolos; por otra, definir niveles de abstracción en las comunicaciones, de modo que dos programas puedan comunicarse sin que importe el protocolo utilizado; finalmente, como hemos visto, se empiezan a encapsular unos protocolos en otros para los casos en los que no sea posible utilizar los nativos. Se tiende a pensar los distintos protocolos como niveles de transporte, y definir APIs independientes de transporte. A la larga, estas tendencias no pueden sino redundar en programas más flexibles, redes más sencillas, y, en general, en más facilidad de uso de las redes. LAN Distance, un producto de IBM, permite que un portátil o en general cualquier ordenador conectado vía módem actúe como

un miembro de pleno derecho de la red a la que se conecta: desde mi casa me conecto con LAN Distance a mi servidor, y si allí tengo LAN Server sobre NetBIOS, TCP/IP, Novell IPX y SNA, eso mismo tengo en casa.

Los programas de instalación y administración se están haciendo más sencillos, más fiables y más potentes. Cuando apareció en el mercado OS/2 2.0, nos llevó meses averiguar cómo conseguir la interoperabilidad completa de los protocolos usados en la Universidad; hoy día todo se reduce a instalar MPTS y los demás productos en orden. Sigue habiendo una serie de pequeños problemas: por ejemplo, DB/2 1.2 debe instalarse después de MPTS y LAN Requester 4.0 si se desean evitar algunos problemas con UPM; y el Internet Access Kit del BonusPack de Warp, aunque de hecho funciona con MPTS, avisa de posibles incompatibilidades (que no existen) y da un mensaje para conectarse por modem (al que paradójicamente hay que contestar "no te conectes") cada vez que se arranca una aplicación. El uso simultáneo de TCP/IP 2.0 y el IAK (por ejemplo para disponer de X-Windows y las utilidades como Gopher, WebExplorer o "Retrieve Software Updates" a la vez) requiere de un orden de instalación específico. Queda por ver de qué modo habrá integrado IBM toda esa funcionalidad en la versión cliente de OS/2 Warp 3.0, que por lo que parece se va a llamar *Warp Connect* y está por salir en cualquier momento. Probablemente el ofrecer todos los productos juntos en un sólo paquete va a permitir ir eliminando los pequeños problemas actuales de instalación e integración, haciendo del acceso en conjunto a las distintas redes y servicios remotos una actividad más sencilla y transparente. Con OS/2 Warp y todo el conjunto de software de redes disponible para Warp, IBM está poniendo el listón muy alto a sus futuros competidores; en estos momentos, para la gestión real, estable, sólida y productiva de entornos de red en estaciones de trabajo normales de tipo PC, esa competencia es simplemente inexistente.O

> **OS/2 2.0 apareció a la vez que Windows 3.1, y en aquel momento la interoperabilidad en Windows era simplemente imposible.**

Solución al problema planteado el mes anterior.

Aprende de tus errores

El mes anterior vimos que el código escrito en C++ producía un grave error al no liberar la memoria reservada, con lo cual el sistema podría quedarse rápidamente sin su recurso más preciado. Esto era fácil de ver si ejecutábamos paso a paso el programa. El mensaje "Libero los bytes reservados para el hijo" no figuraba en la salida estándar. La solución es tan sencilla como declarar el destructor de la clase base del tipo *virtual* (ver código). De esta forma se ejecutarán ambos destructores, el de la clase base y el de la clase derivada. El orden de ejecución de ambos es inverso al de llamada, es decir, primero se ejecutará el de la clase derivada y luego el de la clase base. Señalar que el destructor de una clase derivada de una clase base con un destructor virtual siempre es virtual.

```
class padre {
        char cadena_padre[30];
        public:
        padre() { strcpy(cadena_padre, "Este es el padre."); }
        virtual ~padre() { cout << "Adios." << endl; }
        virtual void ImprimeCadena() { cout << cadena_padre << endl; }
        };
class hijo : public padre
        {
        char cadena_padre[30];
        int *DatosHijo;
        public:
        hijo() {
                strcpy (cadena_padre, "Soy el hijo.");
                DatosHijo = new int [1000]; }        // Reservo espacio para 1000 int.
        ~hijo() {
                cout << "Libero los bytes reservados para el hijo." << endl;
                delete DatosHijo; }                  // Libero memoria.
        void ImprimeCadena () { cout << cadena_padre << endl;    }
        };
```

Abel Bartolomé Rodríguez