3TECH

The 3Com Technical Journal

VOLUME 1, NUMBER 3
Winter 1991

VOLUME 1, NUMBER 3
Winter 1991

3TECH

The 3Com Technical Journal

FEATURES

DEPARTMENTS

3TECH

The 3Com Technical Journal

ISSN 1051-9637

Photo on page one courtesy of Joan Tabb

3TECH, 3Com's Technical Journal, is published quarterly by 3Com Corporation, Santa Clara, CA 95052.
Officers: L. William Krause, Chairman; Eric A. Benhamou, President and Chief Executive Officer; Bob Finnochio, Executive Vice President, Field Operations; Christopher B. Paisley, Vice President and Chief Financial Officer; Debra Engel, Vice President, Corporate Services; Andy Verhalen, Vice President and General Manager, Network Adapter Division; John Hart, Vice President and Chief Technical Officer.

Subscriptions
Subscription rate: $35 per calendar year. To order subscriptions for 3TECH, or to report a change of address, write to 3TECH Journal, 3Com, P.O. Box 58145, Santa Clara CA 95052-9953, ATTN: Dept. CSI. All orders must be prepaid and reference 3C2869. Subscriptions may be entered or cancelled by 3Com employees by sending e-mail requests to: MPS USER:FO:3Com

Submissions
Manuscript submissions, inquiries, and all other correspondence should be addressed to 3TECH's Editor: Marianne Cohn, 3Com Corporation, 5400 Bayfront Plaza, Santa Clara CA 95052-8145. Articles in 3TECH are primarily authored by 3Com employees, however, articles from non-3Com authors dealing with 3Com-related research or solutions to technical problems are encouraged for publication.

When Standards are Standard

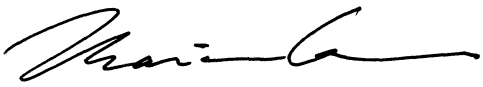
These days, it's difficult to know what is meant by the term *networking standard.* Is it a protocol that has been jointly developed by a team of engineers from different networking vendors through an industry consortium? Is it an application interface that has been developed by a single company and has since become widely implemented throughout the industry? Or is a "true" standard one that consists of a set of specifications developed by workgroups under the auspices of the International Standards Organization? The answer is (as you might expect) all of the above.

A networking "standard" is simply a way of making interoperability between network elements actually happen. The way in which the standard was developed is not as important as whether it is widely implemented and used.

This issue of *3TECH* looks at many such standards—*de facto* industry standards such as NetBIOS and NDIS, and standards such as X.400 and FDDI that have been developed in international standards committees. The 3FORUM column in this issue has been written by Kevin Mills, one of the chief architects of the U.S. government OSI profile (GOSIP). There are also articles on emerging standards such as the Point-to-Point Protocol (PPP), and Heterogenous LAN Management (HLM)—a joint proposal by 3Com and IBM. This issue of *3TECH* also includes a tutorial that describes a basic element of OSI and TCP/IP networking standards: network addressing.

The Winter 1991 *3TECH* also features a host of new tech tips gleaned from the Ask3Com electronic bulletin board. The 3WIZDOM section includes a special "Wizbiz" update from the 3Wizard council President, Michael Chacon, as well as our usual "3WIZDOM" Q & A column. You'll also find expanded coverage in the 3INFO section.

As usual, I welcome your comments and suggestions for making *3TECH* even better. Please drop me a note with your ideas!



Marianne Cohn

Open Systems: How, When, and Why?

By Kevin L. Mills

3Forum presents opinion, analysis, and other commentary on computer networking. The ideas and opinions expressed in 3Forum are the author's and do not necessarily represent those of 3Com Corporation or, in this case, the United States government.

Over the past five years, the trade press has been filled to overflowing with announcements and chronicles of multivendor workshops, consortia, demonstrations, and other cooperative ventures, or adventures. This deluge raises questions in the minds of many users. Why are information technology (IT) companies joining together? How are they joining together? What will result from these efforts in the near future?

Customer demands, changing economic conditions, and rapid technological change are driving IT companies to work together. Customers are demanding multivendor interoperability, portability of software, and plug-compatibility of peripherals—so-called "Open Systems." This demand has been responsible for the creation of the U.S. Government Open Systems Interconnection Profile (GOSIP), the comparable MAP/TOP specifications, and the POSIX operating system interface standard. Similar requirements are emerging even more strongly in Europe, backed by the European Standards Commission.

Vendors working together can meet the user demands, thus stimulating a larger market for IT products and services. If vendors don't play, they may lose market share, or be forced to compete in a market defined by competitors. And economic competition in the IT field has never been more intense.

The 1980s run of mergers and acquisitions combined with the recent economic downturn in the United States IT industry are creating interesting problems for vendors. Some large computer companies have combined; for example the merger of Burroughs and Sperry formed Unisys, and Hewlett-Packard swallowed Apollo Computers. More consolidation is likely. Consolidating companies are faced with the technical challenges of combining computer architectures and product lines into a coherent strategy that can be sold to customers. The vendors' strategy of choice is the open system architectures already in demand. The recent economic downturn only serves to reinforce vendors' sensitivity to customer requirements.

Of course, even without these market-driven incentives, the velocity of technological change in computing and communications challenges every computer company, even IBM. The diverse IBM product line includes five operating systems: MVS, VM, AS/400, OS/2, and AIX. How will IBM integrate these offerings to provide customers with a coherent range of products? While most computer companies are settling on some version of UNIX, the challenges of data communications, database management, graphics, document interchange, security, and network management can only be addressed in industry-wide forums.

The OSI Network Management Forum (OSI/NMF), the Corporation for Open Systems (COS), the Open Software Foundation (OSF), and X/OPEN are the big-budget, high-visibility forums where IT industry players meet to produce agreements on various aspects of open systems. Whether these forums themselves are truly open is, of course, debatable. But the goals, objectives, and products of these groups are all aimed at meeting the perceived customer demand for vendor independence.

The credibility of multivendor efforts comes from more open processes: international standards organizations (e.g., ISO/IEC, CCITT, IEEE) and related open workshops (e.g., the OSI Implementors Workshop, the European Workshop on Open Systems, and the Asian Oceania Workshop). These open standards and standards-related processes create the basic specifications upon which the industry consortia build. The consortia add value in several ways. OSI/NMF provides interim network management specifications to jump-start the standards process; COS develops testing requirements and sponsors the development of tests and test systems; OSF solicits UNIX-based software from industry and integrates the software into a working system; and X/OPEN specifies application programming interfaces in anticipation of POSIX extensions. There is no master plan. The various processes are driven, in true capitalistic fashion, by market forces.

So what can we expect of this simmering stew as our portions are served to us? In computer networking, I predict that OSI will gain widespread acceptance in the market. The serious efforts by the European Community, the U.S. government, and, to a lesser extent, industry, through initiatives by such companies as GM, Boeing, and DuPont should ensure the success of OSI.

I expect that TCP/IP will continue to be available but, as more advanced OSI products enter the market and as the Internet faces the limitations of a 32-bit address space, I believe that the Internet will give way to the global deployment of OSI technology.

SNA will, I am sure, remain a dominant market factor. The investment of IBM in maintaining, evolving, and supporting SNA provides a credibility that managers accept; the public standards process will find it difficult to convince business executives that the worldwide investment in standards, distributed as it is, can ever be managed as efficiently as IBM manages its own corporate investment.

Other proprietary protocols, except for PC LAN protocols, should diminish in importance. The fight for dominance in the PC LAN market will continue between Banyan, Novell, 3Com, and others. Whether one of the PC LAN protocols will dominate is unclear, but during the strife, OSI networking will gain an important presence on PC LANs.

Tumult will continue as industry slowly tames the many beasts living among dense foliage of the network management jungle. The growing availability of proprietary solutions to manage computers, modems, switches, multiplexors, links, LANs, and routers is creating a diversity akin to the computer networking protocols of the 1980s. The network management beasts will be tamed, and OSI is a key foundation upon which to build. While the OSI/NMF has been slow to achieve results, the groundwork established there and in the international standards groups will begin to pay off over the next five years. A standards-based solution will appear that integrates vendor proprietary management schemes for both computers and telecommunications.

The story for operating systems is similar to networking. The push for POSIX will be provided by the European Community, by the U.S. government and, to a lesser but still significant extent, by industry. The range of services supported in the POSIX standard will grow beyond basic operating system functions to include networking, database, and graphics. The UNIX systems available from OSF and from its rival, UNIX International, will comply with POSIX. Proprietary offerings such as DEC's VMS and IBM's VM, AIX, and OS/2 will also comply with POSIX. IBM's SAA will remain a dominant market force.

The computing industry provides a rich marketplace for a range of advanced technologies. Such diversity demands management by all the IT industry players—users and vendors alike. Exploding technology drives us further into chaos, anxiety, and indecision. Thus, technology is both a strength and weakness of the IT industry. Industry players are struggling together to harness the pace and direction of change to create benefits for both vendors and users. ■

Kevin Mills directs the U.S. government's OSI program. In eight years at the National Institute for Standards and Technology (NIST), Mr. Mills has contributed to specification of the class 4 transport protocol, to transport measurement methodologies, and to performance improvements for the transport protocol. He is one of the principal architects of the U.S. Government OSI Profile (GOSIP). Prior to entering public service, Kevin spent five years with System Development Corporation, now a part of Unisys; one year at Tesdata Systems Corporation; and three years in R&D in the U.S. Marine Corps.

NDIS Concepts

By Rex Allers

The Network Driver Interface Specification (NDIS) is a standardized interface for OS/2 or DOS network platforms. NDIS provides access to network services at the Data Link layer and is especially useful if the access must be shared. Software developers who need to employ their own network protocol implementations can program to the NDIS interface and utilize NDIS-compliant drivers provided by network hardware vendors. This frees the protocol developer from programming directly to various network interface cards and solves compatibility problems on machines with multiple protocols.

This article explains why the Network Driver Interface Specification was developed, and describes its organization and operation. Some of the information is general and will be of interest to a broad group of people involved with networks. A good deal of the information is quite detailed and technical. It is intended primarily to give network programmers an overview of NDIS and what is involved in binding a protocol to a vendor-supplied MAC driver for a network adapter board.

The Old Way

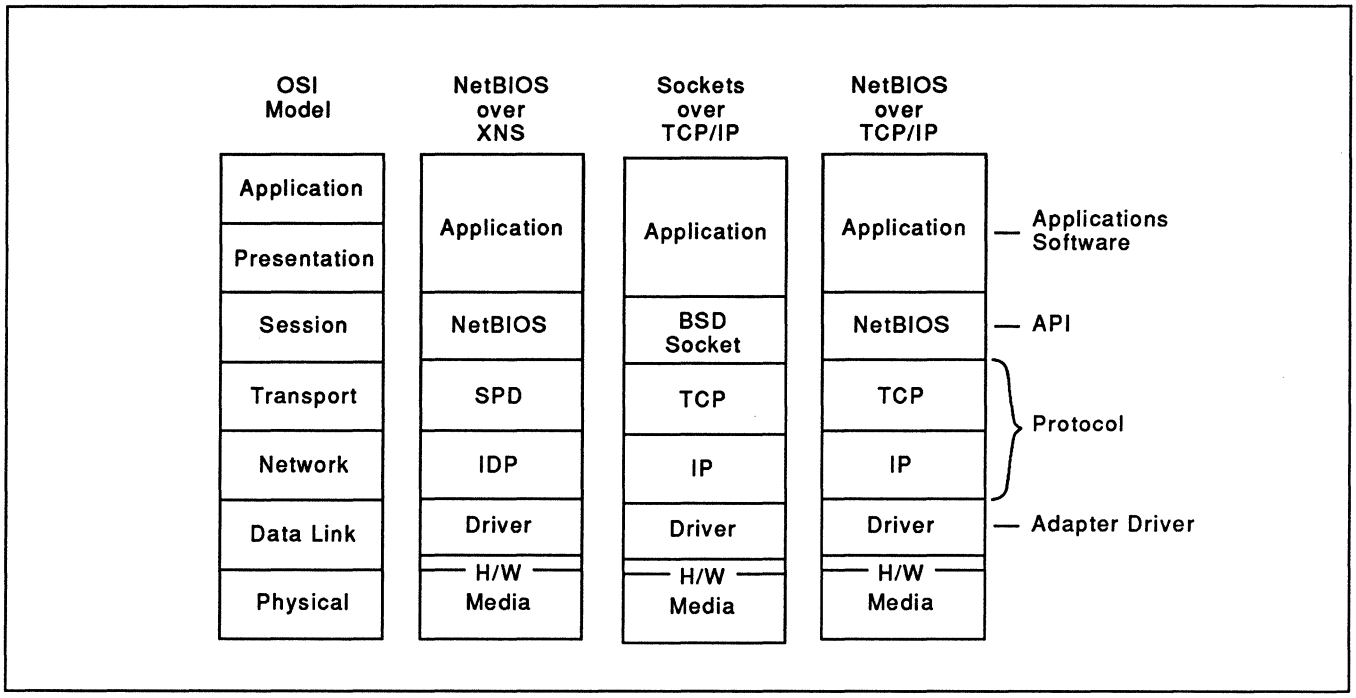
Traditionally, network software vendors for the MS-DOS environment have used ad hoc methods to implement the protocols and drivers that link applications to their resident network hardware. The entity that performs these network functions and provides communication between applications is usually referred to as a protocol stack. In the OSI Reference Model, the stack would correspond to the Data Link, Network, Transport, and Session layers, with some stacks possibly including higher layers. At the stack's top end is a user interface or some type of applications programming interface (API) and, at the bottom end, are the interfacing routines that control the network adapter hardware. In implementation, a stack might consist of one system driver, multiple drivers, a program, or a combination of drivers and programs.

Figure 1 shows three alternative stacks that could be used to perform equivalent network functions. In a typical implementation (for example, NetBIOS over XNS in Figure 1), the Data Link, Network, and Transport layers might be implemented as three separate system drivers, and the Session layer implemented as a TSR program. The interface between the layers would usually be accomplished through a proprietary interface developed by the vendor, and the application would communicate to NetBIOS via software interrupts. This works well in a homogeneous network environment but, as networks grow more complex, it is becoming desirable to have the flexibility to utilize mixtures of different protocols, application interfaces, and network media.

The Problem to Be Solved

Compatibility issues between various networking implementations can make it difficult or impossible to accomplish some seemingly simple tasks. For example, assume that we have an Ethernet network that has two types of file servers attached. One server runs an XNS protocol with NetBIOS at the Session layer, the other server runs a TCP/IP protocol with a Berkeley Socket Session interface.

Figure 1. Typical Protocol Implementations Without NDIS



We would like to write a program to run on a workstation that can copy a file from the first server to the second. In this example, let's say we have two sets of software from two vendors that are designed to communicate with each of the servers, and that the vendors have defined a programmer's interface that should allow us to write a program that talks to the two stacks. Assuming that we have enough memory to load both stacks at one time, we will probably find that our biggest configuration problem occurs at the bottom of the stacks.

At the Data Link layer, each of the vendors has supplied us with a driver for use with their protocol stack that can control the EtherLink II adapter board that we have in our station. Most likely, we will find that each of these drivers expects to have exclusive ownership and control of the EtherLink II. As one of the drivers tries to control the board, it interrupts or corrupts the functions being attempted by the other driver. What is needed is one driver that can control the adapter and be shared by the two protocols.

In May 1988, 3Com and Microsoft released NDIS, which was jointly developed in conjunction with LAN Manager. The NDIS specification is a standard designed to alleviate compatibility issues for both OS/2 and DOS network platforms. The NDIS specification should be beneficial to both the protocol-level network software developer, who now has a standard interface available, and the user, who gains from the flexibility and interoperability advantages of protocols using NDIS.

NDIS Organization

All network software components compliant with NDIS definitions are drivers. These drivers can be classified into two types: protocol drivers, and Media Access Control (MAC) drivers. NDIS allows protocol drivers to be device drivers, TSRs, or DOS applications; however, in this discussion, it is assumed that all NDIS drivers are device drivers—the simplest and most common implementation.

The MAC driver forms the bottom layer of the stack and is the driver that directly controls the network hardware. The remaining higher layers of the protocol stack are implemented in one or more protocol drivers.

The MAC layer is a sublayer within the OSI Data Link layer that is defined by IEEE 802 specifications. This

layer is the appropriate point for a driver that manages the network hardware and implements the transmission and reception of network data packets. NDIS MAC drivers are provided by 3Com and many other network hardware vendors (see Table 1 for a partial list) and can be used with any vendor's NDIS-compliant protocol drivers.

NDIS Stacks

All NDIS drivers, both MAC and protocol, share a common modular structure. Each driver has an upper and lower boundary. The drivers are linked to form a stack by connecting, or binding, the upper boundary of one driver to the lower boundary of another driver during the binding portion of driver initialization. This binding process can be repeated multiple times, linking several drivers, daisychain fashion, to form the stack. The MAC driver at the bottom of the stack always has its lower boundary connected to the physical layer—the network hardware.

The simplest configuration of drivers is one MAC driver supporting one network adapter card bound to a single protocol driver spanning from the MAC layer to the Session layer (see Figure 2). This forms a single protocol stack of two drivers. Optionally, the protocol part of this stack might be made using two or more protocol drivers to form a single stack of three or more drivers. NDIS also allows us to have two completely parallel stacks in one machine, each with its own adapter card and MAC driver, to implement two different protocols.

Table 1. Companies Supporting NDIS

Companies with NDIS MAC Drivers

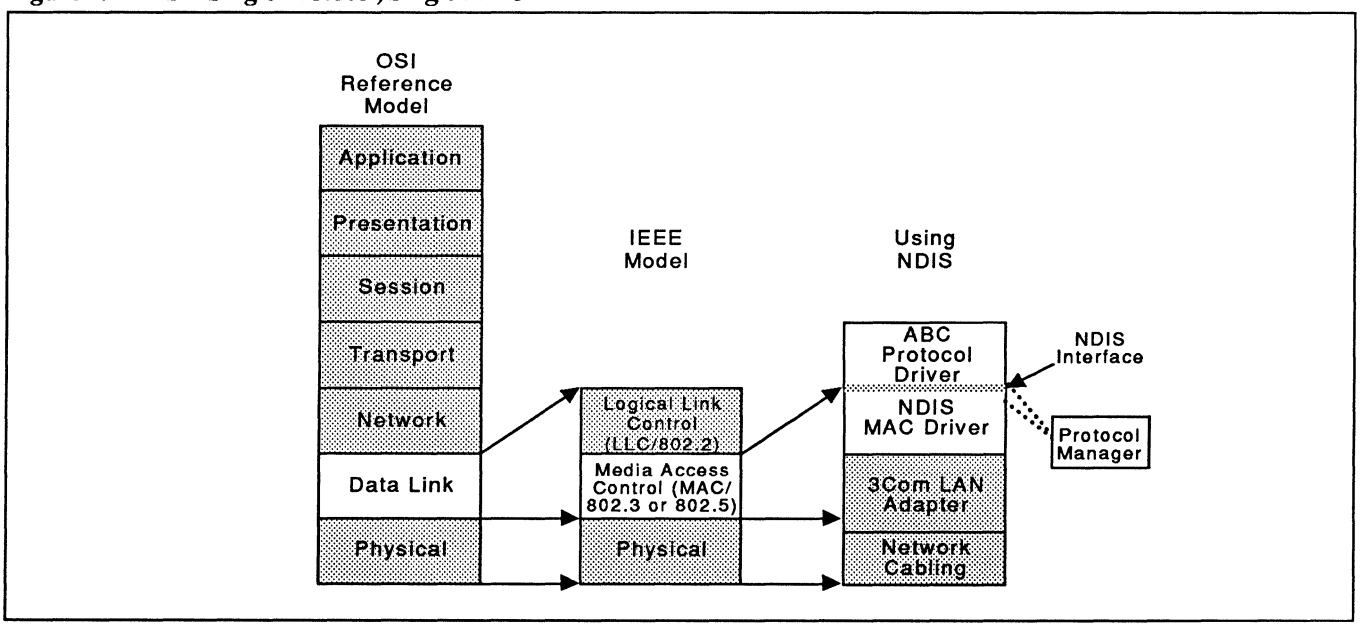
Note: The entire list of companies shipping NDIS MAC drivers is too long for this article, but includes:

3Com	Interlan
AST Research	Proteon
AT&T	Tiara
Compaq	Ungermann-Bass
Exelan	Western Digital
IBM	

Companies with OS or Applications Supporting NDIS

3Com	— LAN Manager
AT&T	— LAN Manager
Banyan	— Vines (workstation)
DEC	— LAN Works
FTP	— PC/TCP
IBM	— LANServer, OS/2 Extended
Microsoft	— LAN Manager
Pacer	— Pacerlink
Sun	— PC-NFS

Figure 2. NDIS—Single Protocol, Single MAC



More importantly, NDIS lets you have just one adapter card and a single MAC driver with the MAC driver bound to two separate protocol drivers (see Figure 3). Therefore, two protocols (for instance, XNS and TCP/IP) can share the same MAC driver and adapter card. This configuration solves the problem discussed earlier in which files need to be copied from two servers running different protocols.

To complete the picture, we can also have two adapter cards and MAC drivers, with both the MAC drivers bound to one protocol driver (Figure 4). This configuration could be used to create a network bridge with one protocol connected to two networks.

Figure 3. NDIS—Multiple Protocols, Single MAC

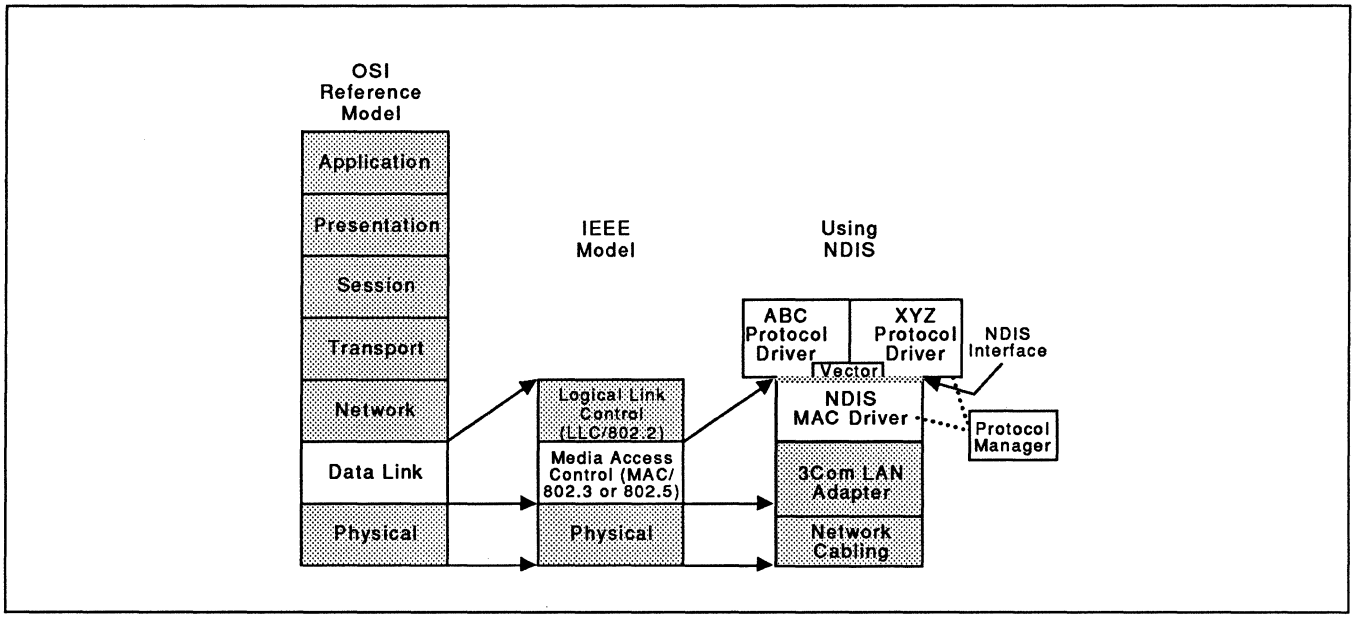
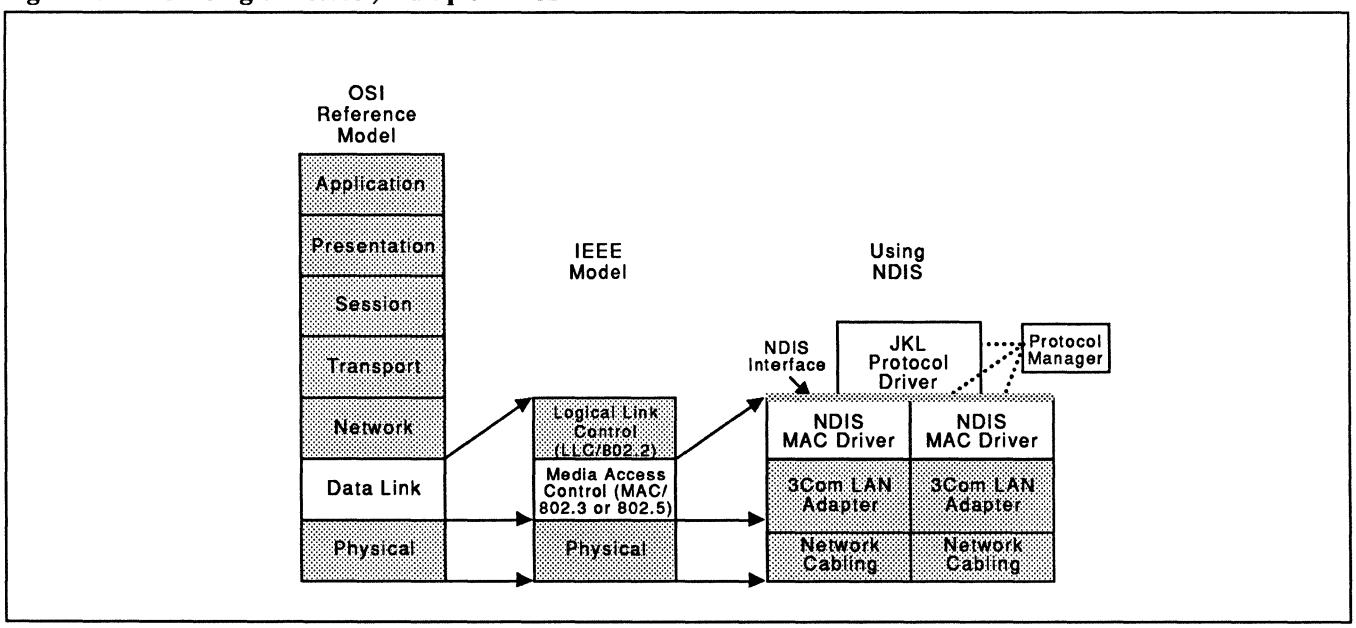


Figure 4. NDIS—Single Protocol, Multiple MACs



Driver Structures

The drivers communicate with each other by a defined set of primitives. The NDIS document has clearly specified a set of primitives for the interface between the MAC driver and protocol driver and for managing the NDIS driver binding process. Although it is possible to implement a protocol stack with multiple protocol drivers, currently no primitives are defined by NDIS for these upper layers. This is not a serious limitation, because a stack with multiple protocol drivers would generally have all of the protocol drivers common to one vendor. There is less need for a standardized interface between protocol drivers than there is at the MAC layer where sharing resources and multiple vendors are more likely.

Each driver contains a series of module-characteristic data structures that provide information about the purpose and capabilities of the driver, and that manage the linkage and operation of the driver during and after initialization.

The main structure is called the Common Characteristics table and contains the name of the driver and version information. This is the highest-level table for a driver; other types of characteristics tables are located from pointers in this table. The Common Characteristics table also contains basic information about what type of binding is supported at the upper and lower boundaries of the driver. The binding information is in the form of a byte identifying the OSI layer that is supported for the boundary. This byte can be examined by other drivers to determine if it is appropriate to bind to the driver.

The Common Characteristics table contains pointers to the other module characteristics tables—the Service-Specific Characteristics table, Service Specific Status table, and Upper and Lower Dispatch tables. These tables give specific information related to the service that the driver performs, manage its operation, and record linkage points to other drivers after the driver is bound.

Managing Binding and Initialization

To form the protocol stacks from the individual drivers we need to get the right drivers connected in the desired sequence. This is accomplished in the initialization and binding process. Three components are used to manage and control the process—PROTOCOL.INI (an ASCII configuration parameter file), PROTMAN.DOS or PROTMAN.OS2 (the protocol manager—a special driver), and NETBIND.EXE (a program that initiates the final driver binding process.)

The initialization and binding process is essentially the same whether the operating system is DOS or OS/2. Some minor adjustments need to be made (for instance, selecting either PROTMAN.DOS or PROTMAN.OS2) and some different parameters may be required, but the discussion that follows applies to either environment.

The ASCII file PROTOCOL.INI contains the instructions for assembling the protocol stack or stacks from the NDIS network drivers. It also contains parameters that are needed to configure the individual drivers. At CONFIG.SYS initialization time, the Protocol Manager Driver reads this file. The file is created—much as CONFIG.SYS is created—either directly, by the administrator typing the information with an editor, or by some type of installation program. The PROTOCOL.INI information is grouped into a number of logical sections of the form:

```
[module name]
  parameter=value
```

The module name is the name of the NDIS driver as contained in the Common Characteristics table for the driver. There will be one module section for each of the NDIS drivers that describes the driver's configuration. Each section can have multiple parameters, but must have at least one, the DRIVERNAME.

Figure 5 illustrates the contents of a simple PROTOCOL.INI file that has entries for three drivers. The first is Protocol Manager, the special driver that controls the binding process—more about its purpose later. In PROTOCOL.INI the Protocol Manager entry is currently optional, but it may be required in the future,

so it's a good idea to include it. The second module section is for the EtherLink II adapter's MAC driver, and the last section is for an arbitrary protocol driver.

Notice that in each section, the first parameter is DRIVERNAME=. This parameter must be included and must specify a name that uniquely defines the NDIS module. In most cases, it will be the driver name that the driver registers to the operating system during initialization. The driver determines the name that must be used, because it uses the DRIVERNAME entry as a key when searching PROTOCOL.INI data for its relevant module section.

Figure 5. PROTOCOL.INI File Contents

```
;***************************
; Example PROTOCOL.INI file
;***************************


[PROTMGR]
      DRIVERNAME=PROTMAN$

[ETHERLINKII]
      DRIVERNAME=ELNKII$
      INTERRUPT=3
      TRANSCEIVER=EXTERNAL

[PROTOTST]
      DRIVERNAME=PROTO$
      BUFFSIZE=2048
      BINDINGS=ETHERLINKII
```

Any number of additional, optional parameter entries can be included in a module section. One purpose of these parameters is to allow control of the driver configuration. A set of valid configuration options will be defined for any particular driver. In the case of the ETHERLINK II section in Figure 5, we have selected two of the possible options for this driver. INTERRUPT=3 tells the driver to use hardware interrupt channel 3, and TRANSCEIVER=EXTERNAL tells the driver to configure the adapter for its external transceiver. In the PROTOTST protocol driver, the BUFFSIZE parameter might direct a protocol to use a particular size for its internal buffers.

The BINDINGS= parameter is a special parameter that is valid only for protocol drivers and specifies the module name of the driver with which the protocol should attempt to bind on its lower boundary. This parameter determines which drivers will be bound together to form the stack or stacks. In Figure 5, PROTOTST is bound to the ETHERLINKII driver.

As mentioned earlier, the component of the NDIS environment that manages the binding process is the Protocol Manager, which has the file name PROTMAN.DOS or PROTMAN.OS2. Protocol Manager has two main functions: it keeps and manages common data for the NDIS drivers, and it controls the binding sequence. Functions of the Protocol Manager are needed by the NDIS drivers during their system level initialization, so the Protocol Manager driver must be loaded in CONFIG.SYS before any of the other NDIS drivers.

The Protocol Manager driver was written by 3Com and is available from both 3Com and Microsoft. Protocol Manager or NETBIND.EXE is available for any vendor for use in the initialization of their network software products. They are also a standard part of LAN Manager as shipped by 3Com and Microsoft.

Driver Initialization

The Protocol Manager gathers NDIS-related information during the system CONFIG.SYS initialization of the drivers. During initialization, the Protocol Manager driver reads the PROTOCOL.INI file and parses the information into a set of structures, called the Configuration Memory Image, which is accessible by the other NDIS drivers. Because other NDIS drivers use this information, the Protocol Manager must be the first to initialize.

As CONFIG.SYS processing continues, the operating system directs the other drivers to initialize. During initialization, they must open the Protocol Manager device (PROTMAN$) and then issue a GetProtocolManagerInfo primitive to obtain a pointer to the Configuration Memory Image (the PROTOCOL.INI data). The drivers find the section of this data that pertains to them and use any parameters found there to adjust their initialization

process. One result of this is that drivers may modify their loaded size, based on parameter requirements, to optimize host memory consumption. If the driver is a protocol and it finds a BINDINGS= parameter, it will assess whether or not this binding is valid. Finally, the driver must issue a RegisterModule primitive to the Protocol Manager to register itself. During this register, the driver passes a pointer to its Common Characteristics table and, for a protocol driver, a list of modules to which it wants to bind, based on the BINDINGS= parameter.

After CONFIG.SYS processing completes, the Protocol Manager has a list of the active NDIS drivers, their characteristics (including entry points), and the desired bindings.

Driver Binding

The actual binding of NDIS drivers starts when some program issues the BindAndStart primitive call to the PROTMAN$ device. For all current Microsoft OS implementations, this call will come from the execution of NETBIND.EXE within a BAT or CMD file.

After receiving the BindAndStart directive, Protocol Manager will take the binding information from the module registrations and build a binding hierarchy tree. Starting at the bottom of this tree (the MAC end), the Protocol Manager works up the tree and issues an InitiateBind primitive to each protocol module that needs a driver bound on its lower boundary. As part of the InitiateBind call, the driver is passed a pointer to the Common Characteristics table of the module to be bound. The protocol driver that was instructed to initiate the bind will then issue a Bind primitive directly to the driver that it wishes to bind. When the bind completes, each driver will have a pointer to the Common Characteristics of the other, and therefore to its entry points.

After the Protocol Manager has processed all of the binding tree, all the appropriate network drivers will be bound to each other. The protocol stack is then fully operational, and the drivers can access each other by calling the dispatch entry points for communication. Figure 6 summarizes the binding process.

Figure 6. Initialization and Binding Process

A. `CONFIG.SYS` Initialization begins.

1. Protocol Manager driver does its initialization
   a. Protocol Manager reads the `PROTOCOL.INI` file and builds the Configuration Memory Image.
2. Other NDIS drivers do their initialization.
   a. Open the `PROTMAN$` device.
   b. Issue `GetProtocolManagerInfo` to gain access to ProtMan Configuration Image.
   c. Read config parameters from the Image and use them to complete initialization.
   d. Issue `RegisterModule` to register characteristics info with Protocol Manager

B. `CONFIG.SYS` processing ends.

C. Binding process starts when the `NETBIND.EXE` program opens the `PROTMAN$` device and issues a `BindAndStart` to Protocol Manager.

1. Protocol Manager builds a binding tree from `RegisterModule` info.
2. Protocol Manager starts at the bottom and calls drivers with `InitiateBind`
   a. Each called driver issues `Bind` to the specified module to complete binding.

D. When all modules are bound, Protocol Manager returns from `BindAndStart`.

One more factor is involved in the binding process if more than one protocol is to be bound to a MAC driver. MAC drivers can only have one binding at their upper boundary. To link one MAC to multiple protocols, the Protocol Manager inserts a component, called Vector, between the MAC and the protocols (see Figure 3). Vector is part of the Protocol Manager and will be bound between the MAC and each of the protocols. To do this, the Protocol Manager first binds the MAC driver to Vector by issuing a Bind call to the MAC driver, then it issues an InitiateBind call to each of the protocols directing them to bind to a Vector entry rather than to the MAC entry.

The basic function of Vector is to route incoming packets between the protocols. When a packet is received by a MAC driver, it will issue a notification of the event to its upper boundary. When vector is involved, it will pass this notification, first to one, then to the other protocol, until one protocol accepts the packet or all have rejected it. Other functions can be passed, essentially directly, between protocols and the MAC, but this vectoring of incoming packets is key to implementing multiple protocols on one MAC.

MAC-to-Protocol Interface and Operation

The main purpose of the NDIS interface is to let the bound drivers communicate with each other. To that end, NDIS specification is largely concerned with defining a set of functions that dictate how the MAC driver will communicate with the protocol bound on its upper layer. Table 2, NDIS Primitives, lists the primitives that are defined for this MAC-to-Protocol communication. All communication between the MAC and its bound protocol will be accomplished using these primitives.

In Table 2, individual primitives are grouped into the main functional categories that they perform. In the first group are functions for the transmission of network packets from the protocol through the MAC and onto the network, and for the reception of packets in the reverse direction. In the Control group are all the functions that the protocol uses to control or modify the operation of the adapter and MAC driver. The Asynchronous Status group contains functions that the MAC uses to report events to the protocol. Finally, the Binding group has the functions used to accomplish the driver binding process. The most important of these have already been described. The remainder are extensions to allow binding and unbinding of dynamically loadable protocols. More on this subject later.

The center column of the primitive table has a symbol that indicates the direction in which the primitive calls are passed. To submit a primitive, the caller driver pushes a series of parameters on the system stack and calls an entry point in the called driver. The entry points are known to the calling driver as a result of the binding process. The Common Characteristics table, whose address was passed during binding, has Dispatch tables chained off of it. The defined entry points for the driver are in a Dispatch table.

The MAC driver's Upper Dispatch table and the addresses it contains are shown below:

MAC Upper Dispatch Table
- GeneralRequest
- TransmitChain
- TransferData
- ReceiveRelease
- IndicationOn
- IndicationOff

These are entry points that the protocol driver will call to request primitive execution by the MAC. All except GeneralRequest are called direct primitives because they serve only one primitive function. The GeneralRequest entry serves the remainder of the primitive calls, other than Binding, that are passed to the MAC. In Table 2, the GeneralRequest primitives are all the primitives in the group labeled CONTROL, except IndicateOn and IndicateOff, which have their own direct entries.

The Direct Primitives have their own entry points because they are performance-critical functions. The primitives employing the GeneralRequest entry, which are less critical, can share a common entry. The GeneralRequests identify themselves by passing an opcode as one of their parameters.

On the protocol driver side, the Protocol Lower Dispatch table defines the entry points from the MAC to the protocol. The table and its contents are as follows:

Protocol Lower Dispatch Table
- GeneralRequestConfirm
- TransmitConfirm
- ReceiveLookahead*
- IndicationComplete
- ReceiveChain*
- Status*

* denotes Indications

Table 2. NDIS Primitives

TRANSMIT AND RECEIVE		
TransmitChain	v	Initiate transmission of a frame
TransmitConfirm	^	Imply completion of frame transmit
ReceiveLookahead	^	Indicate arrival of received frame and offer lookahead data
TransferData	v	Request transfer of received frame from MAC to protocol
IndicationComplete	^	Allow protocol to do post-processing on indication
ReceiveChain	^	Indicate reception of a frame in MAC managed buffers
ReceiveRelease	v	Return frame buffer to the MAC that owns it
CONTROL		
IndicationOff	v	Disable indications from the MAC
IndicationOn	v	Enable indications from the MAC
InitiateDiagnostics	v	Start MAC runtime diagnostics
ReadErrorLog	v	Get error log info from MAC
SetStationAddress	v	Set network address of the station
OpenAdapter	v	Issue open request to network adapter
CloseAdapter	v	Issue close request to network adapter
ResetMAC	v	Reset MAC software and adapter hardware
SetPacketFilter	v	Specify filtering params for received packets
AddMulticastAddress	v	Specify multicast address for adapter
DeleteMulticastAddress	v	Remove previously added multicast address
UpdateStatistics	v	Cause MAC to update statistics counters
ClearStatistics	v	Cause MAC to clear statistics counters
InterruptRequest	v	Protocol requests later async indication from MAC
SetFunctionalAddress	v	Cause adapter to change its functional address
SetLookahead	v	Set length of visible data for ReceiveLookahead
GeneralRequestConfirmation	^	Confirm completion of previous General Request (see text for more explanation)
ASYNCHRONOUS STATUS		
RingStatus	^	Indicate a change in ring status
AdapterCheck	^	Indicate error from adapter
StartReset	^	Indicate adapter has started a reset
EndReset	^	Indicate adapter has completed reset
InterruptIndication	^	MAC response due to InterruptRequest
BINDING		
InitiateBind	p>m	Instruct a module to bind to another module
Bind	m>m	Exchange Characteristic Table info with another module
InitiatePrebind	p>m	In OS/2 dynamic bind mode, instruct a module to restart its prebind initialization
InitiateUnbind	p>m	Instruct a module to unbind from another module
Unbind	m>m	Delete linkage info with another module
GetProtocolManagerInfo	m>p	Retrieve pointer to Configuration Image
RegisterModule	m>p	Register Characteristics and Bindlist with Protocol Manager
BindAndStart	e>p	Initiate the binding process
GetProtocolManagerLinkage	m>p	Get entry point for Protocol Manager
GetProtocolIniPath	d>p	Get file path for the PROTOCOL.INI file
RegisterProtocolManagerInfo	d>p	Dynamic mode, register new Configuration Image
InitAndRegister	d>p	Dynamic mode OS/2, restart prebind initialization
UnbindAndStop	d>p	Dynamic mode, Unbind and terminate a module
BindStatus	e>p	Retrieve info on current bindings
RegisterStatus	e>p	Query if a specific module is registered
KEY		
	^	- MAC to protocol
	v	- protocol to MAC
	p>m	- Protocol Manager to driver module
	m>p	- driver module to Protocol Manager
	e>p	- ? to Protocol Manager; ? is normally an executable program
	d>p	- dynamic protocol or control program to Protocol Manager

All of these entries except Status are direct. Status is the entry for the Asynchronous Status group in the primitive table, and these primitive calls also have an opcode that identifies the type so they can share one entry. In a sense, GeneralRequestConfirm can also be viewed as a shared entry. There is only one primitive for this entry, but it is generated by the MAC at the end of any of the GeneralRequests to the MAC, and contains the request handle of the original primitive request to the MAC. Therefore, it is shared functionally in response to all of the GeneralRequest primitives.

Some of the entries in this list are marked with an asterisk to show that they are Indications. Indications are a special class of requests that imply some special handling. In implementation, they are usually associated with notifications to the protocol that are made from the MAC while it is in interrupt context. Because of this, the protocol is required to handle Indications as efficiently as possible. For example, it might put the received frame on a queue. After the protocol processes the indication, it returns control to the caller, the MAC. The MAC will enable as much interrupt processing as possible and then call IndicationComplete to give the protocol an opportunity to perform more processing on the Indication in a less critical mode (e.g., to decode the previously queued receive frame).

Transmit and Receive

The information in Table 2 identifies the basic purpose of each NDIS Primitive. Of these, transmit and receive are the key functions at the MAC-to-protocol interface level, so let's examine the primitives serving these functions in a bit more detail.

Network packets, or frames, are the data units that are transferred between the MAC and the protocol by the transmit and receive primitives. These packets include all the information, other than hardware-related functions such as preamble and checksum, that will be sent out on the network medium. As a result, the protocol, on transmit, must build the entire packet, including Data Link fields such as source and destination addresses. Likewise, on receive, the protocol must process the packet down to these levels.

The passing of packet data across the interface between the protocol and MAC is accomplished, whenever possible, by exchanging pointers to buffers or to a descriptor that in turn, points to several data buffers. The objective is to avoid unnecessary, time-consuming copying of data between buffers.

In transmit, the packet data buffers are owned by the host system and managed by the protocol driver. NDIS defines a structure, the Transmit Data Buffer Descriptor, that allows the packet data to be contained either in one buffer or in a series of chained buffers. To initiate a transmit, the protocol assembles the packet data into buffers, puts the buffer addresses in the buffer descriptor structure, and calls the MAC with the TransmitChain primitive. The primitive contains a pointer to the buffer descriptor.

The MAC has two options for processing the transmit. It will choose one or the other at its own discretion; the protocol must be capable of handling either. In the first, called synchronous transmission, the MAC copies all the packet data and returns to the protocol with a code signifying that the data buffers are free and the transmit is complete. Optionally, the MAC can return with a code signifying that the transmit is queued (this is called asynchronous transmission). It implies that the buffers are not free and the transmit has not yet completed. Later, after the MAC has copied all the transmit data, it will call the protocol with a TransmitConfirm primitive (the asynchronous response) to inform the protocol that the buffers are now free and the transmit is complete.

An additional feature available in transmit is immediate data. The protocol has the option of beginning the transmit buffers with up to 64 bytes of immediate data. This immediate data, if present, is always the first data to be transmitted. The MAC must fully process or copy this data before returning from the TransmitChain call, even if it will asynchronously process any remaining data buffers for the call. This feature allows the protocol to have a small, locally managed buffer that only needs to be valid during the TransmitChain primitive call. A protocol might use this for building packet header information, or for entire small protocol-generated packets, such as acknowledgements.

For receiving packets, the process can work in one of two ways: using ReceiveLookahead and TransferData primitives or using the ReceiveChain primitive. The method that is used is determined by the MAC, depending on how the MAC and adapter can handle data buffering. It is generally a function of whether the adapter has on-board receive buffers that use I/O or DMA to transfer the data, or whether the adapter buffers are memory-mapped and accessible directly by the host.

For buffers on the adapter that use programmed I/O or DMA to transfer the data, the reception process will use a ReceiveLookahead and TransferData pair of primitives. When the MAC has received a packet that it wants to present to the protocol, it indicates this by calling the ReceiveLookahead primitive of the protocol. The ReceiveLookahead call contains a pointer to a short portion of the data at the beginning of the packet. This usually means that the MAC must have first DMA'ed this lookahead data to a buffer in the host.

At this point, the protocol driver can examine the lookahead data to determine if it wants the packet. In some cases the packet may not be of interest to the protocol. If the packet is not needed, the protocol can return to the MAC indicating a reject and stating that the receive is complete. If the packet is needed, the protocol calls the TransferData primitive of the MAC, which results in the MAC transferring the remainder of the data to a protocol buffer.

The purpose of the Receive Lookahead implementation is to avoid unnecessary data transfers between the MAC and the protocol. This technique improves the efficiency of the network stack.

If the adapter has receive buffers that are accessible as host memory, receive will be implemented with the ReceiveChain primitive. For this type of buffering, the MAC will own and manage the receive buffers. For flexibility, this mode has a ReceiveChain buffer descriptor structure, similar to the transmit structure, that lets multiple separate buffers be joined for one packet transfer. When the MAC has a received packet to present to the protocol, it builds a buffer descriptor for the packet and calls the protocol with ReceiveChain.

When the protocol gets the ReceiveChain Indication, it has two options. In the simplest case, the protocol can copy all of the packet data and return to the MAC specifying that the receive is complete and the buffers are free. In the other case, the protocol can defer copying the buffers and return to the MAC specifying that the buffers are still in use. The protocol will later complete the copying of the buffers and then call the MAC with a ReceiveRelease primitive to indicate that the buffers are now free and the receive is done.

In all of these receive scenarios, the primitive calls issued to the protocol are indications. This means that the protocol drivers need to observe certain rules and that the MAC must issue an IndicationComplete call to the protocol as part of the process. For more information about these indication issues, refer to the NDIS specification document.

Dynamic Binding

As mentioned earlier, some primitives are provided to support Dynamic Binding. Dynamic Binding is a new concept that has been added in Version 2.0.1 of the NDIS document. It allows a protocol to be added to or removed from an existing network configuration after the initialization process has completed. The dynamic protocol driver must be written for this purpose and normally will be implemented as a TSR or transient program module.

The main advantage of Dynamic Binding is freeing system memory until it is actually needed for a particular protocol. This is most useful in the DOS environment where there is a 640K memory limit and it is difficult to have multiple protocol stacks loaded simultaneously.

DOS Versus OS/2

NDIS has all the features needed to allow writing network drivers that will run in either the DOS or the OS/2 environment. A driver can only be used with one of these operating systems (the same driver can't be used for both), but the structure of the driver can be identical for both environments. We have found that one set of

source code can be used to make versions for both DOS and OS/2. Where different techniques are required, a small piece of code can be selected via conditional compile or assembly statements for the two versions. An example of such a difference is that a certain call might need to use Interrupt 21h in a DOS-based driver, versus an IOCTL call for OS/2. A conditional selection of a few lines of code can implement one or the other. The make process for the driver will select the correct environment.

Network device drivers are not very different in structure from other types of device drivers. The device driver must be written to conform to the architecture in which it will run—DOS or OS/2. All of the normal issues apply in writing NDIS drivers for both of these environments. Several books and articles are available that explain general driver development issues. See the list of references at the end of this article for suggested reading.

Summary

One of the main goals of the Network Driver Interface Specification is to save network software developers from "reinventing the wheel" for each new version of network adapter hardware. A protocol that is written with an NDIS interface at its base should be able to function unchanged with many different types of adapter hardware. In addition, the manufacturers of the network hardware should be in the best position to write efficient and bug-free MAC drivers for their own boards. 3Com and many other vendors have NDIS MAC drivers available to support their network hardware.

Using the standardized NDIS interface allows a new level of sharing of network resources in a machine. Multiple protocols and multiple hardware adapters can coexist—even those from different vendors. Input was sought from many leaders in the network industry to guarantee that the specification is flexible enough to meet most networking needs. Further, 3Com carefully defined the functions so that performance was not sacrificed for this flexibility.

If you are a software developer who would like to evaluate NDIS for your specific needs, the next step is to obtain a copy of the Network Driver Interface Specification. The NDIS document can be obtained from 3Com by writing to the following address:

3Com Corporation
Network Adapter Division
Software Product Marketing
5400 Bayfront Plaza
P.O. Box 58145
Santa Clara, CA, 95052-8145 ■

Rex Allers is a Systems Engineer in the Technical Services Organization at 3Com, specializing in support for developers. Rex has worked in engineering and support in the computer industry for longer than he cares to admit, and has been at 3Com since 1986.

Suggested Reading:

1. *Microsoft/3Com LAN Manager Network Driver Interface Specification,* 3Com/Microsoft, 1990

2. *The Open Book,* Marshall Rose, Prentice Hall, 1990

3. *Advanced MS-DOS,* Ray Duncan, Microsoft Press, 1986

4. *Writing MS-DOS Device Drivers,* Robert S. Lai, Addison-Wesley, 1987

5. *OS/2 Programmer's Guide,* Ed Iacobucci, McGraw-Hill, 1988

6. *Writing OS/2 Device Drivers,* Raymond Westwater, Addison-Wesley, 1989

Planning and Maintaining FDDI Networks

by Dono van-Mierop

FDDI is gaining momentum as an important industry standard. Currently, the most implemented application of FDDI is a campus-wide communication backbone. In the future, FDDI will find its way to the desktop, coexisting with Ethernet and Token-Ring. FDDI has several significant advantages over other types of LAN technology: its bit rate is an order of magnitude higher, it is specifically designed for fiber optics, it has built-in fault detection and recovery mechanisms, it extends to distances two orders of magnitude further, and it has the most advanced built-in network management functions.

Unlike papers describing the FDDI technology [1] or specific products [2], this article discusses the planning and maintenance issues of FDDI. It presents the relevant technical specifics for network planners and managers to understand FDDI's Station Management functions and how they relate to its operation. It discusses the planning of the static physical topology, the configuration of the physical network to create a logical token ring topology, the tuning of the performance of the dynamic token passing mechanism, and the built-in tools available to manage the FDDI network as a whole.

Overview of the FDDI Standard

Fiber Distributed Data Interface (FDDI) is a standard for data networking that covers the physical and most of the data link layers of the OSI model. It operates at 100 Mb/s, over a fiber-optic cable plant that is structured as a ring of trees using a token ring media access mechanism. An FDDI network is very extensible—it can span distances of a few meters to hundreds of kilometers with minimal effect on performance. In addition, the standard includes a set of sophisticated station management tools.

The FDDI standard is the result of work that has been going on for about eight years in the Accredited Standards Committee (ASC) X3T9.5 technical committee. To date, this work has yielded five documents [3,4,5,6,7] that constitute the FDDI standard. The following is a list of current FFDI documents:

- The Media Access Control (MAC) standard defines the services to the higher-level Link Layer Control (LLC), token passing, frame formation, addressing, error detection and recovery, and the bandwidth allocation among nodes.
- The Physical (PHY) standard defines the clocking scheme and the data encoding/decoding.
- The Physical Media Dependent (PMD) standard defines the multimode fiber optic medium, the connectors that attach the nodes to the medium, and the fiber-optic transceiver that converts light into electrical signals.
- The Single-mode PMD (PMD-SM) standard defines alternative physical media-dependent properties for operation with single-mode fiber and laser technology.
- The Station Management (SMT) standard includes a comprehensive Management Information Base (MIB)—a collection of data on the attributes of all the stations on the network. In addition, SMT specifies three major functions: Connection Management (CMT), Ring Management (RMT), and Frame-Based Management (FBM).

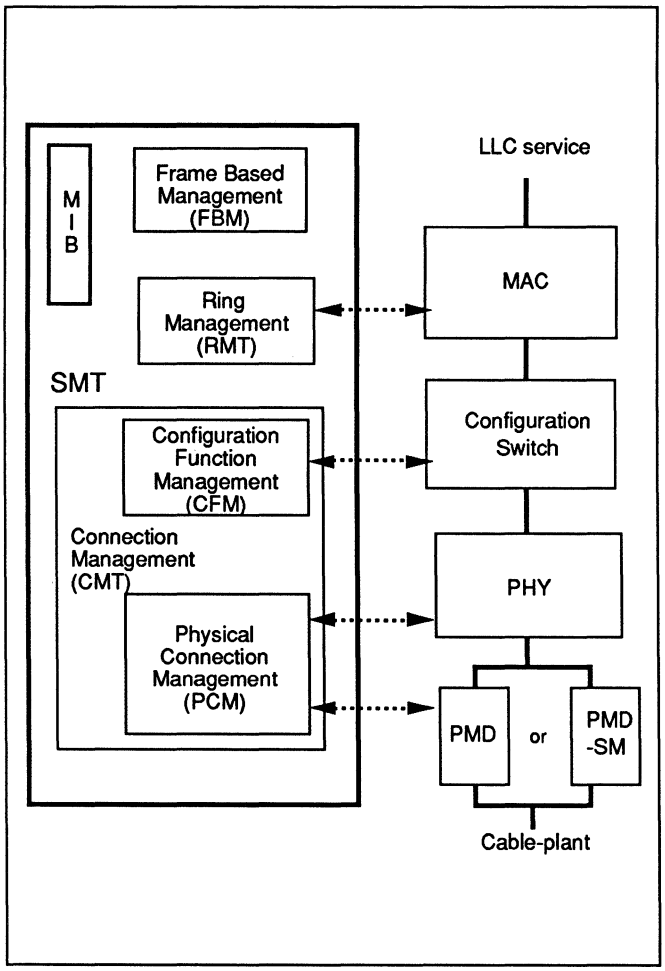
CMT controls the establishment of reliable node-to-node links and the connection of links to the media access units.

RMT monitors the MACs to assure correct ring operation.

FBM provides mechanisms that are based on SMT frames; e.g., status reporting, parameter management, neighborhood information, status information, and duplicate address detection.

Figure 1 shows the relationship between the FDDI standard documents: MAC, PHY, PMD, PMD-SM, and SMT. The Configuration Switch (CS) forms the internal connections between MACs and PHYs. It is not an explicit FDDI standard because it is implementation-dependent and has no effect on interoperability.

Figure 1. Functional Relationships between the FDDI Standard Documents



The FDDI standard's current status, as of October 1990 (date of the last X3T9.5 meeting), is that MAC, PHY, and PMD are published ASC and ISO international standards; PMD-SM is in the publication process; and SMT has passed the first letter ballot (with numerous comments). Although SMT is not yet a standard, two backward-compatible versions of it serve as industry standards for interoperability tests.

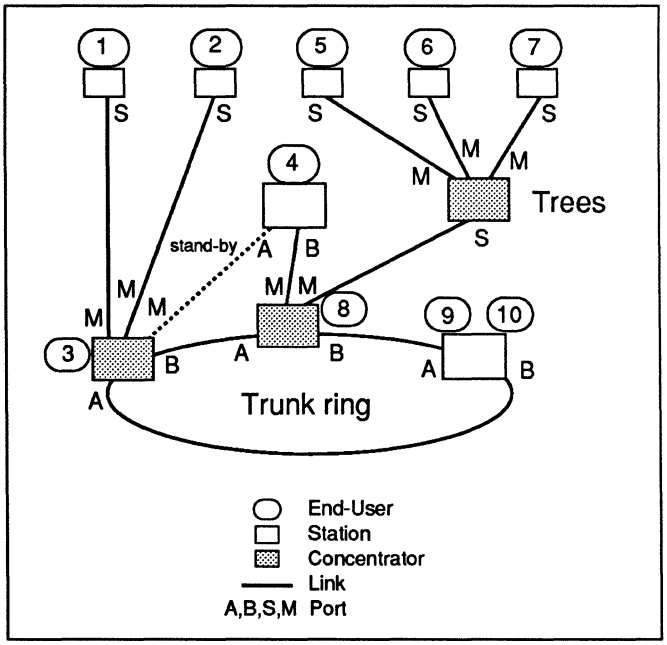
Planning the Physical Topology

Network topology is determined by the specific application; namely, by the desired location of the end stations and by the most cost-effective cable plant. FDDI provides a flexible way to accommodate a variety of useful topologies.

The physical topology of FDDI is based on three types of elements: a *station* is a node in the topology that includes an FDDI MAC (i.e., it provides end-user access to the network); a *concentrator* is a node that serves as the root of a topological tree; and a *link* is a duplex cable that interconnects nodes (both stations and concentrators).

A "legal" FDDI physical topology is a *ring of trees* (or a subset thereof). An example is shown in Figure 2.

Figure 2. Example of an FDDI Physical Topology



Physical Rings and Trees

The *tree* part of the topology maps well onto typical building wiring installations, where each office is directly connected to the floor wiring closet (called horizontal wiring), and each wiring closet is directly connected to a communication closet in the basement (called vertical wiring). This wiring structure is typical of utilities such as telephone and electricity. This structure was recently standardized by the IEA 41.8.1 subcommittee for local data networking [8], and it has already received wide industry acceptance. The major advantage of the tree topology is the flexibility of installation and modification. Further, the concentrators simplify control over the network by decoupling the manageable network from the activities of the end users. A recent paper[9] shows that in large FDDI networks, the quality of service perceived by the users will often increase when a tree structure is used to attach end users to the FDDI ring.

The *trunk ring* part of the FDDI physical topology is useful in the backbone part of the network; i.e., where the configuration is local (a computer room, a communication riser) or is stable (interconnecting buildings). The trunk ring is typically mapped on existing cable plants and is used where continuous fault tolerance is important. The trunk ring has a built-in automatic fault recovery mechanism that is independent of the end stations. Whenever a link fails, the trunk ring "wraps," and the connectivity between all the stations and concentrators on the ring is retained.

Typical installations are combinations of rings and trees where the trunk ring provides a robust backbone and the trees provide flexible end-user connections. Of course, every network is planned according to specific needs, taking into account the planned location of end stations, existing wiring, standardization, link size, required flexibility, desired control, and cost.

Physical Links

Links in FDDI provide point-to-point connections between stations and concentrators. The point-to-point physical structure (as opposed to multidrop buses) provides the advantage that each link is independent of the others. This provides the capability to tailor each link to specific requirements (length or bit error rate.).

FDDI was originally designed for fiber-optic links, though accommodation of twisted pair copper wire and mapping to Sonet links are in the standardization process too.

Fiber-optic cables have a number of very important advantages over copper. The primary advantage is that more data can be transferred longer distances. The light in fiber is not affected by EMI or RFI interference. This makes fiber a more reliable medium—especially in environments where "noise" from large motors or lightning can interfere with electronic transmission and corrupt data. As fiber does not generate any electrical emissions, it creates a more secure network. The cost of fiber is coming down, and its installation is becoming simpler.

FDDI specifies the use of fiber with an outer diameter of 125 microns and operation at a light wavelength of 1300 nm. The network planner can tailor the type of fiber (i.e., inner diameter) and the type of transceiver (i.e., optical power) separately for each link. There are several alternatives for selecting the type of fiber; the most common in the U.S. is 62.5/125 multimode. The most common in Europe and Japan is 50/125. When using multimode fiber with the standard's category I (LED) transmitter over multimode fiber, the distance is limited to 2 kilometers because of the effect of chromatic dispersion. Longer distances can be reached by using single-mode fiber and category II (laser) transmitters that allow distances of up to 50 kilometers or more.

The link length limitations can be calculated from the optical power budget. The output power of the category I transmitter is in the range of -14 dBm to -20 dBm. The input power of the receiver is in the range of -14 dBm and -31 dBm. This assures that the receiver will never be saturated by the transmitter, and it provides an optical loss budget of 11 dB. The category II transmitter is specified with an output power range of 0 dBm to -4 dBm. This provides a loss budget of 27 dB. Saturation of the receiver is possible, and an attenuator may have to be planned in order to prevent that. Losses depend on the length of the fiber (about 1 dB per kilometer, depending on fiber quality). Additional sources of optical loss that the planner needs to take into account are connectors, splices, bypass switches, transition between different fiber types, and patch panels.

Types of Ports and Links

A *port* is the PHY/PMD entity in each of the nodes adjacent to a link. The ports implement SMT's Physical Connection Management (PCM) protocol that establishes the node-to-node connections and maintains the quality of the link.

The standard specifies four types of of ports—A, B, S, and M—combinations of which yield 16 types of links. PCM uses the information about types of ports to enforce the physical ring of trees topology. The following links are possible:

- A-B and B-A links are peer-to-peer trunk ring connections.
- M-S and S-M links are master-slave tree connections. Concentrators have M ports to connect end stations or other concentrators on the tree.
- S-S links create the smallest possible topology of a two-station ring.
- M-A, M-B, A-M, and B-M links provide possible dual-homing in the tree: the A port is not active as long as the B port is. The A-M link is a redundant standby link, as shown with dotted lines in Figure-2. In case the link connected to port B fails, the link connected to port A becomes active (and connectivity is restored).
- M-M links are never allowed because they would create loops in the tree.

Figure 3 shows the types of links allowed in FDDI topologies. Several link types are specified as "undesired," though the network manager could configure certain policies to allow these links to become active. For example, stations may have mechanisms to recover from a twisted ring created by an A-A link.

Whenever the link quality falls below a certain alarm threshold (set by the manager) or PCM detects an incorrect connection, the link is automatically deactivated and the manager notified. The manager would then need to take corrective action like replacing a cable or tightening a connector.

These connection rules guarantee the ring of trees topology. The only topology violation that is not detected by PCM is the master-slave loop of concentrators. This undesirable configuration is detected (and the manager notified) by a special SMT protocol that is discussed later in this article.

Figure 3. FDDI Link Connection Rules

		Other Port			
		A	B	S	M
This Port	A	V,U	V	V,U	V,P
	B	V	V,U	V,U	V,P
	S	V,U	V,U	V	V
	M	V	V	V	X

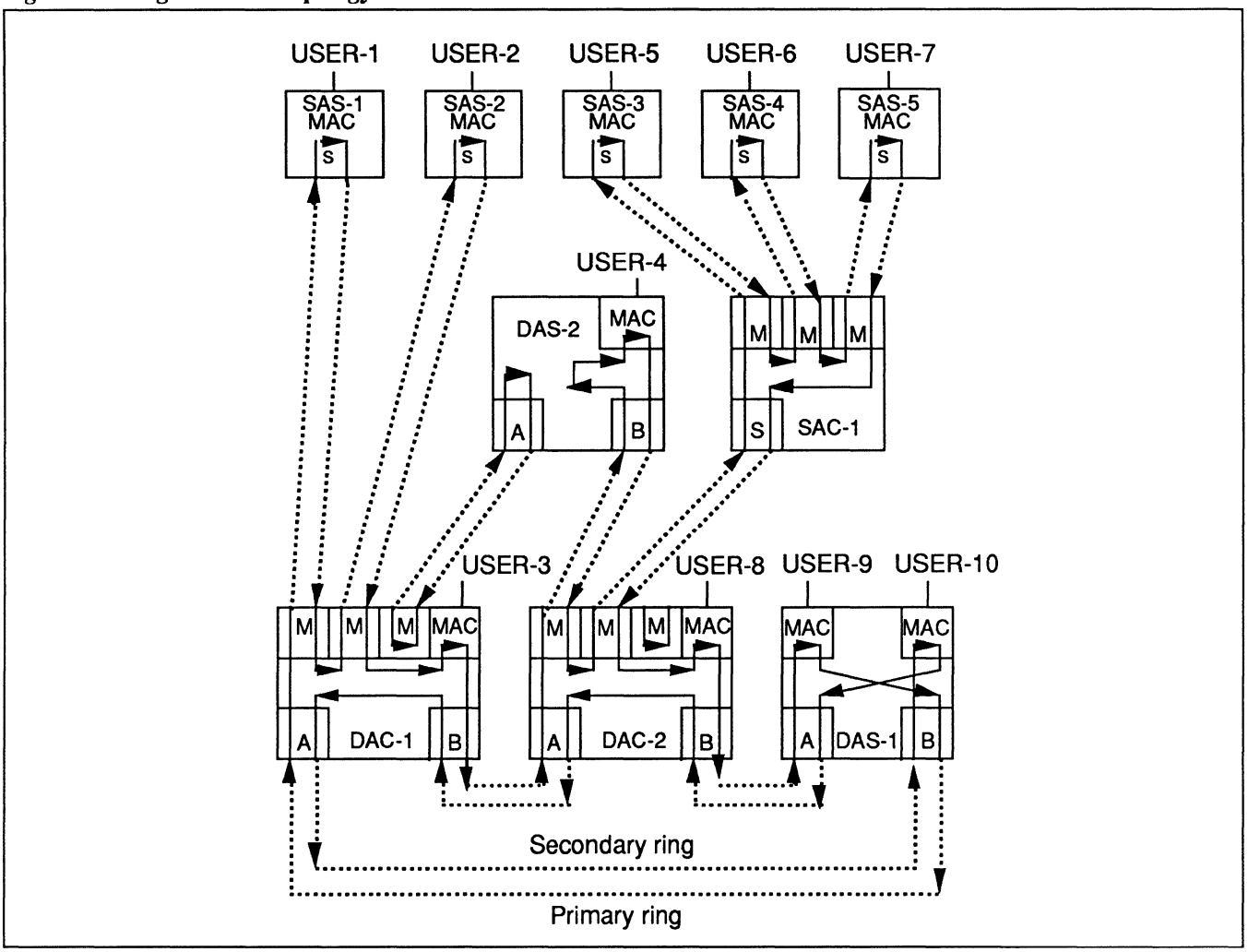
Where: V indicates a valid connection
X indicates an illegal connection with SMT notification required
U indicates an undesireable connection with SMT notification required
P indicates, if Active, prevent THRU, port B takes precedence

Configuration of the Logical Topology

Any user wishing to communicate through the FDDI network needs to be physically connected to a MAC in a station. The logical topology of the FDDI network is a *dual-counter rotating token ring* of MACs or, in special cases, a single token ring. The MACs are interconnected to one another with token paths that traverse stations and links mapped onto the physical topology. The manager configures each MAC in the network to be either on the Primary or the Secondary token ring. Figure 4 shows a logical topology mapped on the physical topology (as shown in Figure 2) where all but one of the MACs (user 10) are on the Primary ring.

Each physical link in the trunk ring provides a token path to both the Primary and Secondary rings (in opposite directions). Each link in the physical tree provides an up and down token path in either the Primary or the Secondary token rings.

Figure 4. A Logical FDDI Topology



FDDI Stations

Any FDDI station is one of five basic types:

- A Dual-Attached Station (DAS) is typically part of the trunk ring, includes an A and a B port and one or two MAC entities. It may also be used for dual homing.
- A Single-Attached Station (SAS) is the "leaf" of a tree and includes one S port, and one MAC entity.
- A Dual-Attached Concentrator (DAC) connects a tree to the trunk ring. It includes an A, a B, and a number of M ports, and one or more MAC entities. Dual homing can also be used here.
- A Single-Attached Concentrator (SAC) connects a tree to a subtree. It includes an S port, a number of M ports, and zero or more MAC entities.
- An Unrooted Concentrator includes only M ports and possibly MACs.

The internal token path of each station is established by the station's *configuration switch*, depending on its state. The basic states are THRU and WRAP-p (where p stands for A, B, or S).

The *THRU* state applies only to dual attached stations and concentrators when they participate in both the primary and secondary rings. Port B is the input and port A the output of the Primary ring; port A is the input

and port B the output of the Secondary. A MAC can be included in one or both of the paths, at the output port.

The *WRAP*-p state applies to any station when it participates only in one ring; a particular port, p, is both the input and output of the token path on the same ring. A MAC can be included in the path.

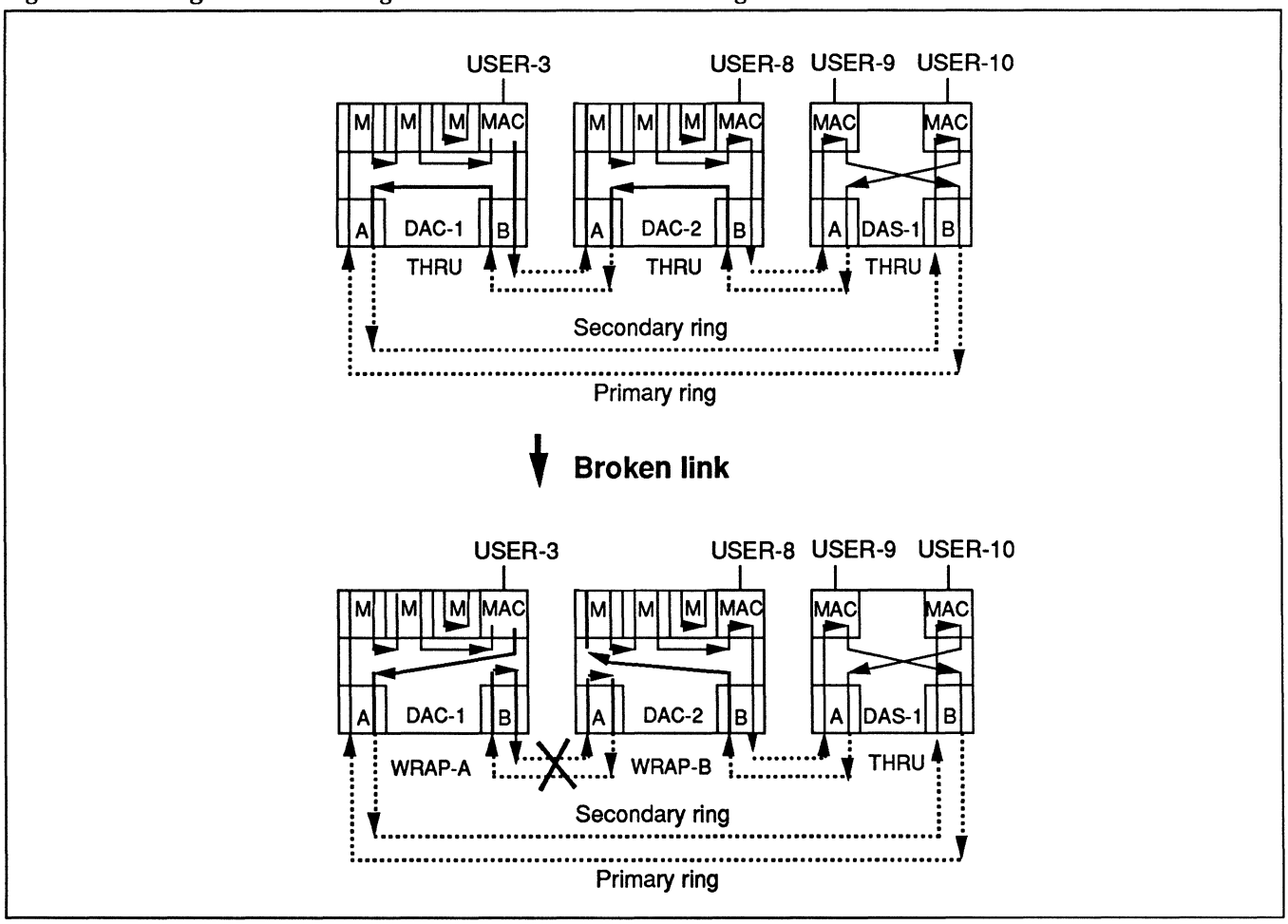
The configuration switch is controlled by the Configuration Management (CFM) machine specified in SMT. The collection of CFMs in all the stations establishes the logical topology of the ring under the manager's control. After the stations are physically connected with links to form the physical topology, the CFM in each station attempts to use its active links in order to establish the logical topology that includes the maximum number of MACs. The logical topology of a particular physical network can be changed automatically by the CFMs because of link failure or station failure or by a specific manager configuration command.

Fault Recovery

Two examples illustrate fault recovery when there are broken links in the trunk ring or in the tree.

The first example (see Figure 5) shows the recovery from a broken link in the trunk ring. Initially, all stations are configured to THRU, User-3, User-8, and User-9 are on the Primary ring, and User-10 is on the Secondary ring. As soon as the link between DAC-1 and DAC-2 is broken, the PCMs on both sides isolate the link, and the CFMs in the two stations adjacent to the break reconfigure the stations to WRAP-A and WRAP-B, respectively.

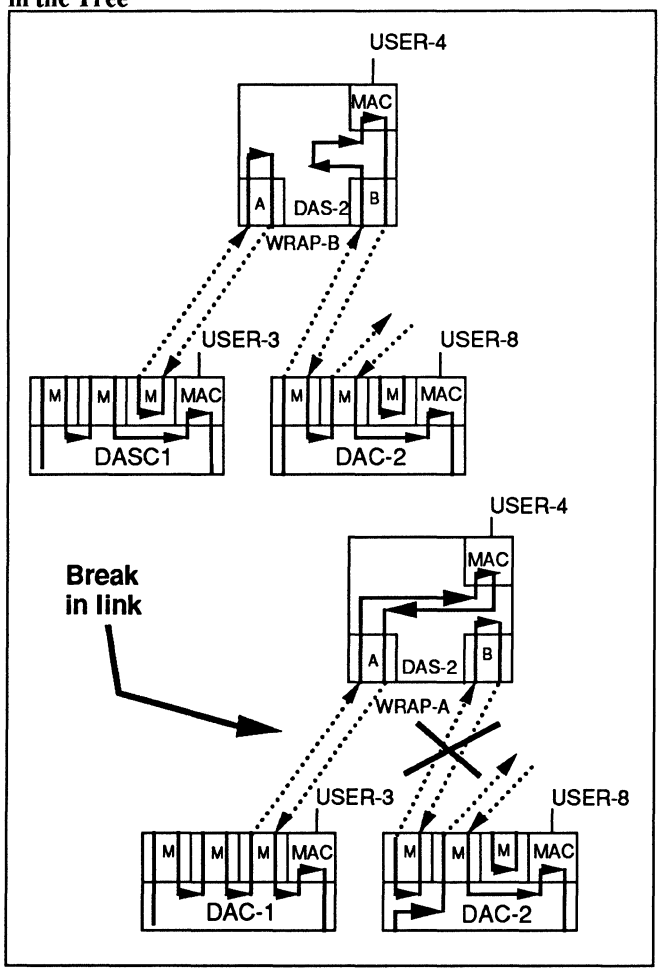
Figure 5. Reconfiguration Following a Broken Link in the Trunk Ring



The net result is that all users end up on the Primary ring. The important point is that all users continue to communicate in spite of a break in the trunk ring.

The second example (see Figure 6) shows the recovery from a broken link in the tree in the presence of dual homing. Concentrators DAC-1 and DAC-2 are on the trunk ring. Station DAS-2 is connected to DAC-2 and has a redundant link to DAC-1. Normally DAS-2 is in WRAP-B. Note that dual homing in a DAS or DAC (detected by a B-M or an A-M link) is not allowed to be configured to THRU. When the link between DAC-2 and DAS-2 breaks, it goes inactive, DAS-2 reconfigures to WRAP-A, and the configuration of the concentrators changes. The important point here, in addition to the recovery, is the convenient configuration control that is exercised by the concentrators.

Figure 6. Reconfiguration Following a Break in a Link in the Tree



Media Access Services

Next, we will look at the dynamic behavior and unique characteristics of FDDI logical rings and their impact on performance and reliability.

Ring Initialization

The logical ring is initialized automatically, after construction or reconfiguration by a distributed Claim process that is triggered by a timeout of inactivity. The Claim process yields a single *Token* and a *T[op]* parameter, agreed upon by all stations, that is used for fair ring access. If the Claim process is successful, then normal transmission and reception over the ring starts.

If a break occurs in the logical ring, the Claim process will time out and the MAC will begin sending out Beacon frames to request intervention by the network manager. Each MAC yields to Beacon frames from its upstream neighbor so that, if the break in the logical ring persists, the station downstream from the break will be the only one Beaconing. This allows the network manager to identify the location of the break and take corrective action. If the MAC receives its own Beacon it knows that the ring was restored and it goes back to the Claim process.

The Claim process does not work properly if two or more MACs have the same address. RMT specifies a protocol that detects duplicate addresses and, if any are present, notifies the network manager. Stations with duplicate addresses should be physically removed from the ring. Duplicate addresses should be rare, however, because of the allocation of manufacturer-specific address blocks by the IEEE.

RMT specifies another protocol, called *Trace*, that kicks in when Beaconing persists. The Trace protocol traces the ring upstream from the fault (the Beaconing MAC) and performs self tests in the stations until it discovers a functioning station. The bad segment of the logical ring is automatically isolated and the network manager is notified and can take corrective action.

MAC Services and Frame Format

The FDDI frame format (see Figure 7) is generally similar to the IEEE 802 frame format. The frame starts with a Preamble and a unique Starting Delimiter octet, followed by the Frame Control (FC) octet.

The FC determines whether the frame is one of the following: a Token, an SMT frame, a MAC frame (e.g., claim or beacon), a synchronous LLC frame, or an asynchronous LLC frame.

Two types of LLC frames provide the data transport services to the higher layer. The *synchronous* service provides a preallocated guaranteed bandwidth that the network manager specifies to the station on a long-term basis. This service is designed for applications that require a fixed guaranteed bandwidth, like voice, video, or other real-time traffic.

The *asynchronous* service provides fair media access where each station is allowed to transmit frames from the moment it receives the Token until the expiration of the Token Rotation Timer (TRT), which is measured from the previous reception of the Token.

This mechanism balances the bandwidth allocated to the specific MAC with the real-time load on the whole network. The parameter that controls this mechanism is T[op], which is established during the claim process described above. T[op] determines when TRT expires. T[op] also serves as a timeout; after two expirations of TRT, the ring becomes inoperable and the claim process starts.

The next four fields are the 48-bit Destination Address, the 48-bit Source Address, the Information field, and the 32-bit Frame Check Sequence. The maximum size of the Information field is close to 4500 octets.

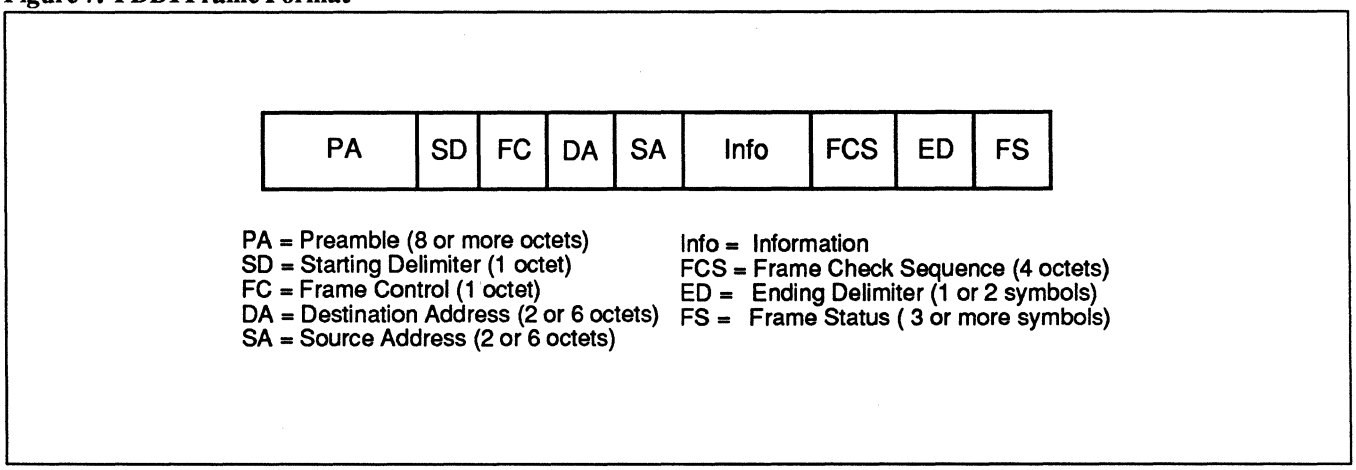
The frame is terminated by a unique Ending Delimiter and three status symbols: an Error indicator, an Address Recognized indicator, and a frame Copied indicator.

Error Detection

On rare occasions, frames that travel over the ring are subject to errors. The least reliable part of the system is the links that are monitored for errors and deactivated below a certain bit error rate, as discussed earlier. FDDI's MAC frames include three powerful mechanisms to detect errors whenever they occur: the unique starting and ending delimiters, the 32-bit Frame Check Sequence, and the Violation code symbols. The probability of errors going undetected is extremely low; about 10E-25.

The MAC provides a number of mandatory counters that are very valuable to the network manager in predicting and locating faults: a Frame Ct (count) that serves as a reference for the traffic on the ring, an Error Ct that includes the number of error frames detected by this MAC (and not by others), and a Lost Ct that includes the number of occurrences of an error during the reception of a frame or token. The network manager can reconfigure any MAC out of the network when the MAC is suspected to be faulty.

Figure 7. FDDI Frame Format



Performance Factors

FDDI has two fixed performance parameters: bit rate and ring latency. The bit rate of FDDI at the MAC level is 100 Mb/s. The ring latency is the time it takes a bit to go around the ring and depends on the number of stations and the length of the links on the logical ring. The ring latency in microseconds is approximately the number of stations plus five times the ring size in kilometers.

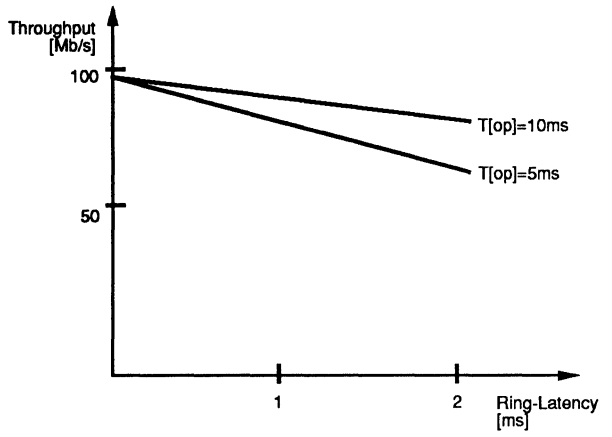
The two most important performance parameters from the user's point of view are *throughput* and *access latency* (not to be confused with ring latency). Various factors affect these two parameters and allow the network manager to tune their performance.

The maximum aggregate throughput depends on T[op] and the ring latency. The more idle rotations the token performs, the lower the maximum throughput. The maximum throughput is 100 Mb/s times (1 minus the ratio between the ring latency and T[op]). From this formula, the larger the T[op], the higher the maximum throughput.

The maximum access latency is determined directly by T[op]. If the Token does not reach the MAC after two expirations of TRT (i.e., twice T[op]), then the MAC assumes that the Token got lost and starts a new Claim process. This means that the access latency is bound by twice T[op]. The actual latency depends on the load on the ring. A smaller T[op] yields a shorter (better) access latency; however, if T[op] is lower than the ring latency, then the ring will never work because the token will not make it around the ring.

Figure 8 plots the maximum throughput versus the ring latency of a number of T[op]s. These plots should guide the tuning of T[op] according to the number of stations, the ring size, and maximum allowed token latency. The plots show that even for very large networks, the maximum throughput is still very close to 100 Mb/s. *Analysis and Tuning of the FDDI Media Access Control Protocol*[10] presents a detailed analysis of these tradeoffs.

Figure 8. Maximum Throughput versus Ring Latency



Note that 1 ms could be 100 stations and 180 Km ring
2 ms could be 200 stations and 360 Km ring

Managing the Operation of the FDDI Network

The FDDI standard includes a set of management tools that are based on SMT frames that use the MAC access service. These mechanisms facilitate ring-map construction by polling the stations for status information, alerts from the stations about abnormal events and conditions, and access of the MIB.

Neighbor Information

The network manager can periodically extract the ring map that reflects the current logical configuration of the network. This configuration is represented as the ordered (cyclic) list of "neighbor" MACs. Any changes in the ring map may indicate a problem that the network manager should investigate further.

The ring map can be constructed using the information gathered from the results of the periodic (approximately every thirty seconds) Neighbor Information Frame (NIF)

exchanges whereby every station finds out the address of its immediate upstream neighbor. Putting all the pairs of neighbors together yields the whole ring map. The same protocol verifies the operation of the local MAC transmit and receive paths in the absence of other traffic. It also verifies the absence of duplicate addresses that would prevent the ring from operating properly.

Station Information

SMT includes a mandatory request/response mechanism whereby every station has to respond to a request by another station within thirty seconds. The network manager can use this mechanism to poll other stations and gather status and operational information. This mechanism uses Station Information Frames (SIFs) that include the following information:

- A time stamp for frame identification.
- A station descriptor specifying whether it is a station or concentrator, the number of MACs, the number of M ports, and the number of A, B, or S ports.
- The station state: THRU or WRAP configuration, a connection to a twisted ring, a duplicate address, an Unrooted Concentrator (i.e., without an active A, B, or S port).
- The station policies for rejecting particular link types.
- The individual addresses of both MAC neighbors.
- The description of the internal token path.
- Parameters relating to T[op] and a number of counters (Frame-Ct, Error-Ct, and Loss-Ct), for every MAC in the station.
- Parameters relating to the status of the link error monitor for every port in the station.

Status Reports

The SMT includes a protocol that is used by stations to announce events or conditions that may be of interest to the network manager. The protocol uses Status Report Frames (SRFs) that are transmitted five times for each instantaneous event and transmitted while a continuing condition persists. The minimum intervals between SRFs is 2 seconds. A new event or condition is reported again after four seconds, then eight seconds, then sixteen seconds, and from then on every thirty-two seconds. These intervals provide a good balance between responsive notification and keeping the steady SRF traffic low.

The SRFs report on the following conditions:

- MAC frame error, which is active while the ratio between the sum of the error Ct and loss Ct over the sum of the frame-count and loss Ct is above an implementation-specific threshold.
- PHY LER, which is active while the link error rate estimate exceeds the alarm threshold.
- Duplicate address, which is active while a duplicate address is detected.

In addition, any configuration change to the station, any attempts to set up an undesired or illegal connection, and any changes in the MAC's neighbor addresses are also reported in SRF frames.

Parameter Management

The SMT also defines a simple mandatory echo request/response protocol that allows interrogation of each station to find out whether it is alive.

All the attributes and parameters are organized in a Management Information Base that is defined in the SMT document. The FDDI MIB is very comprehensive and complements a number of other MIBs defined for other networks and other networking layers (e.g., Token Ring, Bridging, TCP/IP). The SMT defines an optional protocol to access the MIB using Parameter Management Frames (PMFs). This protocol allows getting or setting any parameter in the MIB, subject to a well-defined authentication mechanism.

Summary

The FDDI data networking technology has been described from the point of view of the network planner and manager. In spite of its sophistication, FDDI is relatively simple to plan and maintain.

The logical topology is simple; a collection of stations, concentrators, and links connected according to the geographical requirements of the installations. The FDDI "ring of trees" topology provides a good balance between flexibility and structure. Various types of links allow the construction of cost-effective cable plants. Many types of installation errors are automatically detected and reported to the network manager for corrective action.

The logical topology is a simple dual-counter rotating ring. The network planner decides on the distribution of the end-user stations on the Primary and Secondary rings. The network itself reports on any changes in configuration and, in many cases, provides fault recovery that allows the network manager some time to fix the situation without disrupting service.

Once the network's physical and logical topology is constructed, FDDI constantly attempts to configure itself to provide maximum connectivity. At the same time, the network does not tolerate any errors, and its various error detection mechanisms deactivate parts of the network where quality is not acceptable. Faults in any part of the network are immediately reported to the network manager. The performance that the network provides is very high and degrades very little with the growth of the network. The network manager has the tools to balance desired throughput and access latency.

Finally, no other kind of network provides such an elaborate and straightforward set of station management tools. These tools can be used in various network management architectures and open the way for the implementation of very sophisticated automatic management functions. ■

Dono van-Mierop joined 3Com in November 1989 and has been directing engineering activities in the areas of FDDI and hubs. Prior to joining 3Com, he was the director of engineering at Fibronics and, prior to Fibronics, he worked at Intel, where among other activities, he designed Intel's Ethernet VLSI controller.

Dr. van-Mierop earned B.S.E.E. and M.S.E.E. degrees from the Technion in Israel, and a Ph.D. in Computer Science from UCLA. He has been a member of the ANSI X3T9.5 (FDDI) committee since 1984.

References

1. F. Ross, "Rings are 'round for good," *IEEE Network Magazine*, January 1987.

2. D. van-Mierop, "System Finex: The first implementation of a high speed networking standard," *High Speed LANs*, North Holland, 1987.

3. ISO, "Fibre Distributed Data Interface (FDDI) - Media Access Control," ISO 9314-2:1989.

4. ISO, "Fibre Distributed Data Interface (FDDI) - Physical Layer Protocol," ISO 9314-1:1989.

5. ISO, "Fibre Distributed Data Interface (FDDI) - Physical Layer Medium Dependent," ISO 9314-3:1990.

6. ANSI, "SMD-PMD," X3.184-199X, Rev. 4.2, 1990.

7. ANSI, "FDDI Station Management," ANSI X3T9/90-078, Rev. 6.2, 1990.

8. EIA TR-41.8.1 - EIA Building Wiring Subcommittee

9. J. Hutchison, "The role of concentrators in FDDI rings," *IEEE 14th Conference of Local Computer Networks*, October 1989.

10. D. Dykman and W. Bux, "Analysis and Tuning of the FDDI Media Access Control Protocol," *IEEE Journal on Selected Areas in Communications*, July 1988.

Suggested Reading:

F. Ross, "An Overview of FDDI: the Fiber Distributed Data Interface," *IEEE Journal on Selected Areas in Communications*, September 1990.

D. van-Mierop, "FDDI—a sign of things to come in data communications," *International Conference on Computer Communications*, November 1988.

Acronyms Used in This Article:

ASC: Accreditted Standards Committee
CFM: Configuration Management
CMT: Connection Management
CS: Configuration Switch
DAC: Dual-Attached Concentrator
DAS: Dual-Attached Station
FBM: Frame-based Management
FC: Frame Control
FDDI: Fiber Distributed Data Interface
ISO: International Standards Organization
LLC: Link Layer Control
MAC: Media Access Control
MIB: Management Information Base
NIF: Neighbor Information Frame
PCM: Physical Connection Management
PHY: Physical
PMD: Physical Media-dependent
PMD-SMD: Physical Media-dependent Single Mode
PMF: Parameter Management Frame
RMT: Ring Management
SAC: Single-attached Concentrator
SAS: Single-attached Station
SIF: Station Information Frame
SMT: Station Management
SRF: Status Report Frame
TRT: Token Rotation Timer

OSI Now and into the Future: An Investigation into X.400

By Dr. Andrew Bartholomew

The X.400 series of CCITT recommendations has become the most widespread OSI application in use today with an international network of public carriers now able to exchange electronic mail between five continents using X.400.

This article is a tutorial on X.400-1988 and examines in detail its architecture and principal components. It traces the development of X.400 from the 1984 to the 1988 recommendations, discusses the major differences between them, and considers future directions.

This article aims to provide a thorough technical background to the inner workings of the X.400 standard. It is intended for those who are contemplating implementing an X.400 network and want to learn more about its architecture. It is also intended for those already familiar with the 1984 standard who want to find out more about X.400-1988 from a functional viewpoint. It should also be of interest to software developers involved with OSI applications and to people looking for an introduction to the CCITT recommendations.

Introduction

Much has been written about X.400 since its introduction in 1984, and many X.400 systems have been implemented throughout the world. Most of the implementations currently available are based on the 1984 version of X.400 and not on the more recent and advanced 1988 version. It will still be some time before most international carriers are implementing X.400-1988 exclusively, but the transition has already begun. For these reasons, this article concentrates on X.400-1988, and on the capabilities of an X.400 that will carry us towards the next century.

An Overview of X.400

The CCITT X.400 series of recommendations describe a model for a distributed vendor-independent messaging environment known as the Message-Handling System (MHS). As part of the model, the X.400 recommmendations specify the communications protocols used by each participating system and the logical structure of the principal components and how they interact with external services such as the X.500 Directory Services.

The first series of X.400 recommendations appeared in 1984 and were updated in 1988. The 1988 version consolidated the work of 1984 and turned what was a functional messaging system—although it was lacking in security, robustness in certain environments, and a pure OSI architecture—into a complete and integrated OSI messaging application. This was made possible because of advances made with OSI in general between 1984 and 1988 and because of practical experience of implementing 1984-compliant mail systems.

The 1988 MHS forms an application in accordance with the principles of the OSI reference model (in particular the structure of the application layer, ISO 9545) and uses the presentation layer services and services offered by other, more general, application service elements. A 1988 MHS can therefore be constructed by using any OSI-compliant network.

The Message-Handling System

The MHS enables its users to send and receive messages by storing and forwarding them. In the X.400 model, a user can be either a person or a computer process, and a message can be human-readable text and graphics or machine-coded information.

A user accesses the services of the MHS via a user agent (UA). This is an application process that interacts with the Message Transfer Service (MTS) or a Message Store (MS) to submit and receive messages on behalf of its unique user. The MTS relays messages submitted to it and delivers them to one or more UAs, MSs or access units (AUs). The MTS comprises a number of Message Transfer Agents (MTAs) which, operating together, provide the store-and-forward relaying and delivery capability of the MTS.

An MS is a functional object associated with a single UA. A particular UA need not subscribe to an MS; rather, an MS is intended to complement the functionality of a UA.

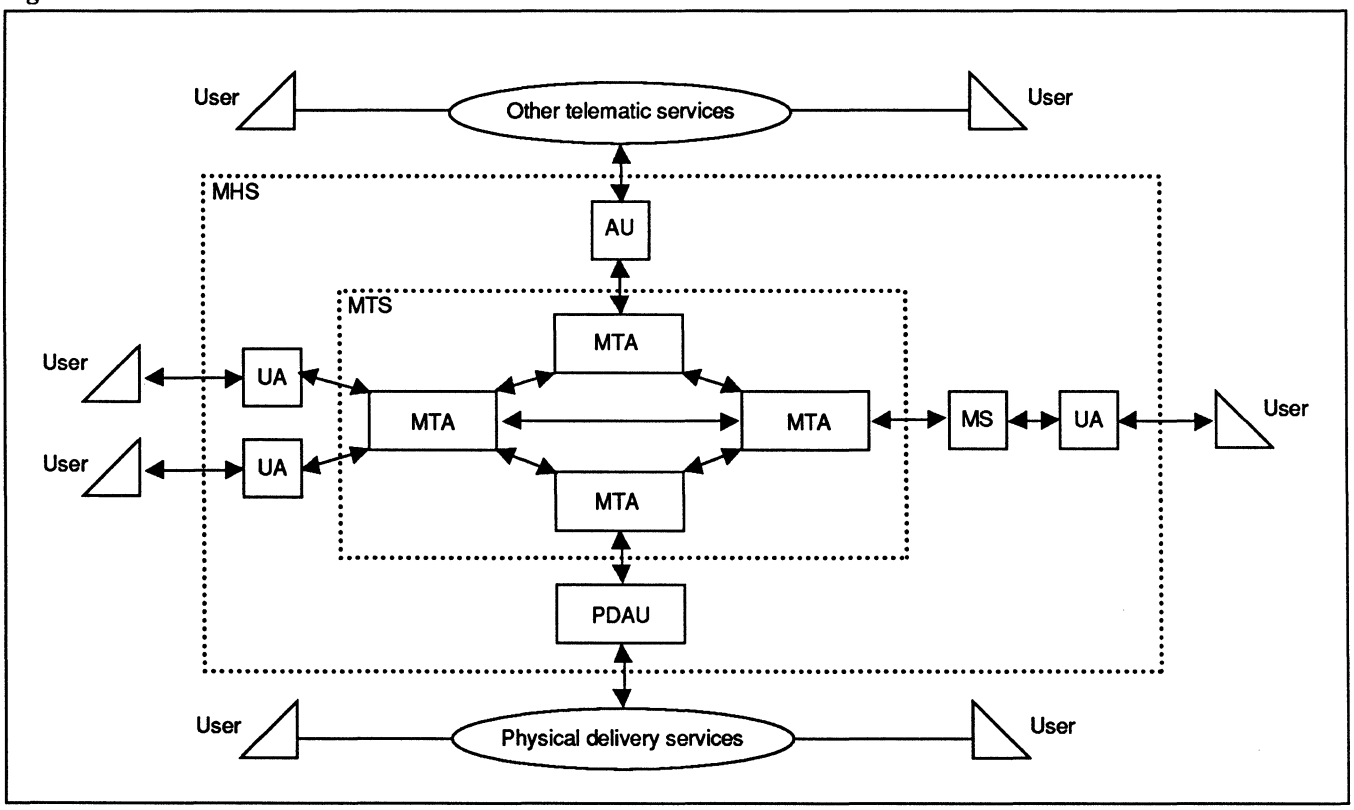
An AU provides intercommunication between the MHS and another system or service, such as telematic services or physical delivery services (for example, the postal system).

The UAs, MSs, AUs, and MTAs that form the MHS are shown in Figure 1.

User Agents

In addition to their MHS functionality, UAs provide local functions for such actions as message preparation, local message database management, and message arrival alerting. UAs are grouped into classes depending on the type of message content they can handle. A UA may be able to handle more than one type of message content and thus be a member of more than one class. Members of the same class of UAs are called cooperating UAs; their common functionality enables them, with local functions, to enhance communication between their users.

Figure 1. Functional Model of the MHS



Message Store

The MS is an optional component that was introduced in 1988. Its purpose is to act as a continuously available storage mechanism to take delivery of messages on behalf of a user. An MS is necessary so that, for example, UAs can be implemented on personal computers that can be removed from the MHS environment. Without the MS, messages destined for such a UA could not be delivered, and the sender would be advised accordingly. The absence of a message store specification in X.400-1984 is one of the major shortfalls of the earlier version.

A UA subscribing to an MS interacts only with that MS and not with an MTA. The UA submits a message to the MS which, in turn, submits it to the MTA before confirming its success to the UA. However, the MS may expand a submitted message if the UA has requested that the existing message be forwarded. The MTA delivers an incoming message to the MS and returns a delivery notification to the originator (if requested). The MTS considers the message to be delivered even though it has not yet reached the UA. The MS then provides the UA with basic message retrieval mechanisms that are independent of the message type. One MS acts on behalf of a single UA; there is no provision for common or shared MS capability.

The Message Transfer Service

The MTS functionality is independent of the type of message being relayed, so it is used by application-specific services to provide various MHS services; for example, the Interpersonal Message Service (IPMS), used to provide normal electronic mail functions between users.

The MTAs within the MTS can be linked by local area networks leased lines, or switched network connections.

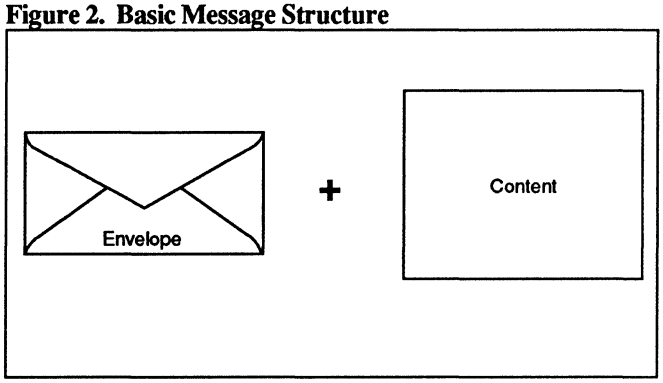
The basic functionality of the MTS has not been changed between X.400-1984 and X.400-1988—its architecture has been modified and its capability to encompass security mechanisms and interact with message stores has been enhanced.

Message Structure

The basic structure of a message is shown in Figure 2. The envelope carries information used within the MHS when it transfers the message from the originator to the intended recipients. The content of the envelope (i.e., the message) is the piece of information that the originator wishes to have delivered to one or more recipients

There are three basic message types: a *message*, which contains information to be conveyed from an MHS user to one or more recipients; a *report*, which contains information about the delivery or nondelivery of messages; and a *probe*, which carries no information but determines whether a message can be delivered to the indicated recipients.

Figure 2. Basic Message Structure



The structure of an IP message (IPM) is shown in Figure 3. IP messages are created by UAs in the IPM service class (IPM-UAs) and are divided into headings and bodies. A heading contains information provided by the user (for example: to, cc, subject) and a body consists of the information that the sender wishes to communicate. The body is divided into one or more parts, each consisting of a single type of encoded information; for example, IA5 text or G3 facsimile. The IPM-UA provides the entire IP message with an envelope for submission into the MTS.

The X.400-1988 recommendations give eight specific content types for messages: IA5 Text, telex, teletex, G3 facsimile, G4 class 1, videotex, voice, and mixed mode (CCITT T.501, T.561). Of these, voice and mixed mode require further study before they can be implemented. In addition, the IPMS accepts encrypted content types, as well as bilaterally defined, nationally defined, and externally defined (unidentified) types. The 1988 recommendations have dropped the TIF (T.73) and SFD (X.420-1984) document types from those appearing in X.400-1984, and added G4 class 1 and mixed mode.

Physical Configurations

The functional objects of the MHS can be implemented in many different ways; a UA and an MTA can be located in the same processing system, or a UA can reside in a stand-alone system. Typically, a single system implements either a stand-alone UA, an MTA and a number of MSs, or an MTA and a number of MSs and UAs. These configurations are shown in Figure 4. Notice that, externally, a colocated UA and MS are indistinguishable from a stand-alone UA.

Figure 3. IP Message Structure

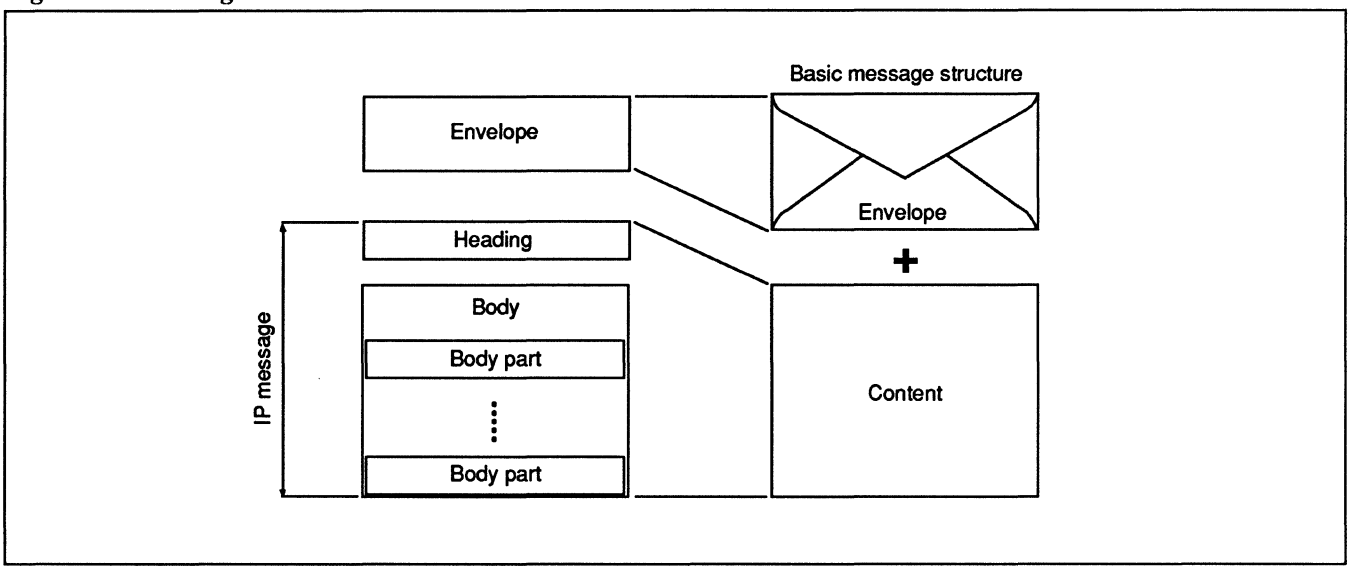
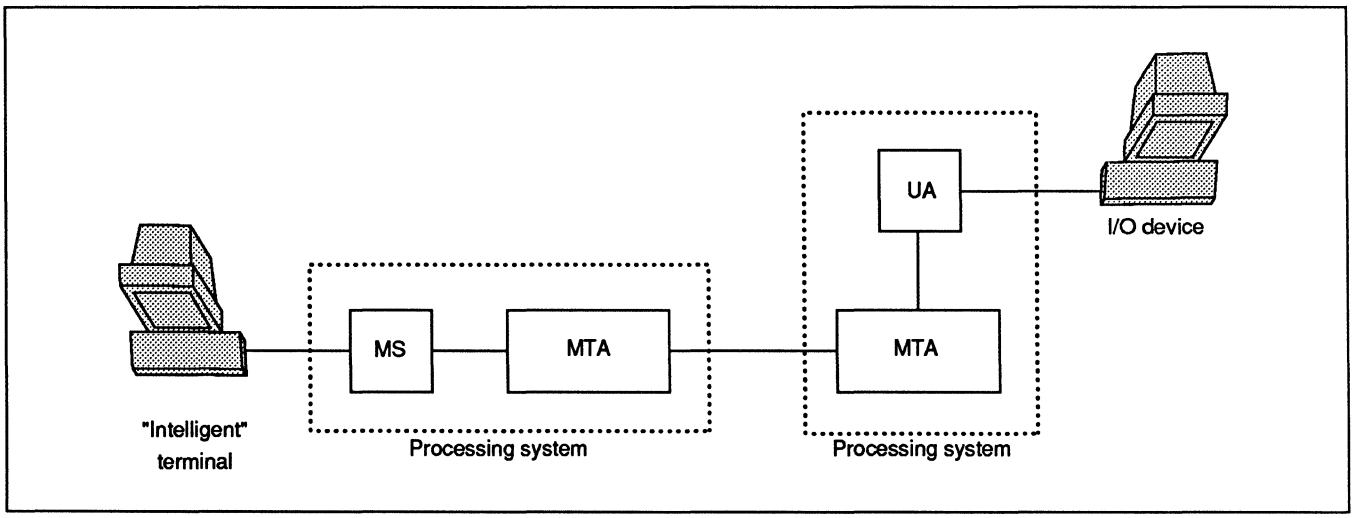


Figure 4. Physical Configuration



Organization of the MHS

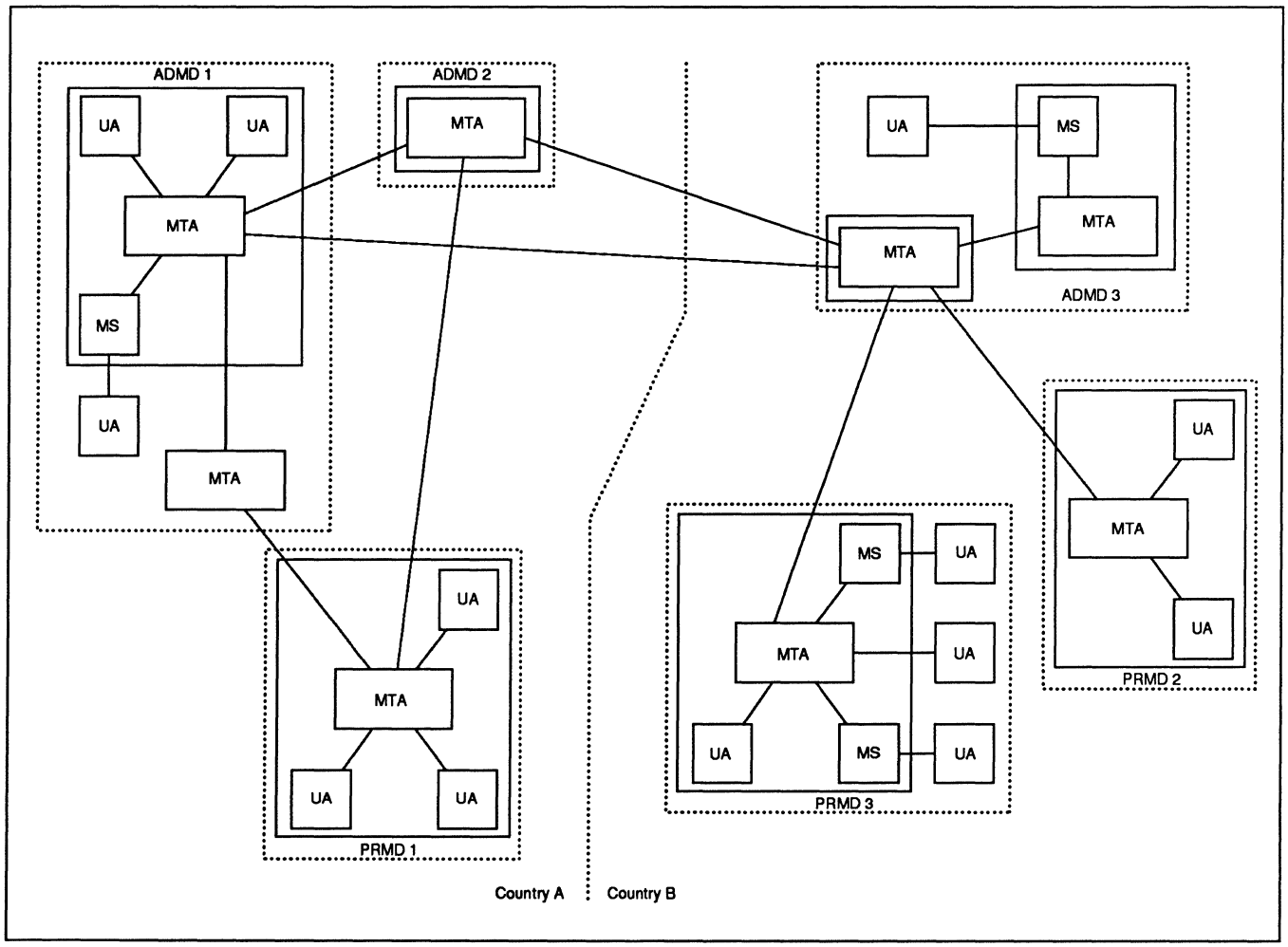
To facilitate addressing within the MHS, and to provide a structure within which public message-handling services can be made available by administrations, the MHS is divided into management domains. At least one MTA, zero or more UAs, zero or more MSs, and zero or more AUs operated by a given organization form a management domain. A management domain operated by an administration such as a PTT or public value-added network supplier is called an Administrative Management Domain (ADMD). A management domain operated by an organization other than an administration is called a Private Management Domain (PRMD). The relationships between management domains is shown in Figure 5.

An ADMD is characterized by its ability to provide relaying functions between management domains. The provision of these services is, in general, governed by national regulations. Within the definition of the MHS, however, they may provide access for their users in one or more of the following ways:

- Users to administration supplied UA
- Private UA to administration MTA
- Private UA to administration MS
- Private MTA to administration MTA

A PRMD is considered to exist entirely within one country and may not act as a relay between ADMDs. In the interaction between a PRMD and an ADMD, the ADMD must take responsibility for the actions of the

Figure 5. Management Domains



PRMD that are related to the interaction. Further, the ADMD is responsible for ensuring that the accounting, logging, quality of service, uniqueness of names, and related operations of the PRMD are correctly performed. The interaction between two PRMDs is not defined in the X.400 recommendations.

Names and Addresses

In the MHS, the principal entity that requires naming is the user. In addition, the X.400-1988 recommendations formally introduced the concept of a distribution list (DL), which also requires a name.

A DL enables a message to be forwarded to all the members of a DL simply by naming the DL. A DL consists of *DL members*, which can be either users or further DLs; a *DL submit permission*, which is a list of users and other DLs that are permitted to use the DL; a *DL expansion point*, a unique MTA within the MTS where the DL members are added to the recipient list of the message containing the DL (messages are transported to the DL expansion point before expansion occurs); and a *DL owner*, a user responsible for the management of the DL.

User and DLs are identified by Originator/Recipient (O/R) names, which comprise directory names and/or O/R addresses.

An O/R address of a user or DL is an attribute list that contains sufficient information for the MTS to uniquely identify the MTA responsible for the user or the DL, and for the MTA to identify the intended user or DL. In the case of a user, the MTA will manage the interface between the user's UA or MS and in the case of a DL, the MTA will be its DL expansion point. The standard attributes defined in X.400-1988 (excluding the physical delivery attributes) are:

- ADMD name
- Common name
- Country name
- Network address
- Numeric user identifier
- Organizational name
- Organizational unit name
- Personal name
- PRMD name
- Terminal identifier

The common name attribute identifies a user or DL by another attribute—for example, organizational name. Thus in the O/R address:

Country = GB
ADMD = TESTMAIL
PRMD= TCOMUK
Organization = 3COM
User Identifie r= AC135
Personal Name = Paul
Common Name = Marketing Manager

The common name identifies Paul, user AC135, as the Marketing Manager. This common name is most useful when the O/R address is stored in a directory. In the above example, users can look for the Marketing Manager without knowing he is called Paul and has a user ID of AC135.

The X.400-1988 recommendations define four standard formats for an O/R address: mnemonic, terminal, numeric, and postal. The postal version is an addition to X.400-1988 which allows a message to be passed to a physical delivery system through an AU, so the O/R address must contain the physical delivery address in addition to the MHS address of the AU.

The most common O/R address is the mnemonic address, which contains the following:

- One country name and one ADMD name, thereby identifying an ADMD.
- One PRMD name, one organizational name, one organizational unit name, one personal name or common name, or a combination of these and, optionally, one or more domain-defined attributes.

A directory name is a construct that unambiguously identifies an object (for instance, an O/R address) within a Directory Information Base. Currently, the syntactic structure of a directory name is defined by the X.500 recommendations to be an ordered sequence of relative distinguished names, although the possibility of widening the scope of names is under study.

A directory name must be looked up in an X.500 directory to find out the corresponding O/R address. This lookup may be carried out either by the user, in

which case the MHS is supplied with an O/R address, or the user can submit a directory name to the MHS and have the name looked up automatically. The particular solution employed is a local decision and is dependent on the implementation being used. However, the specification of how a user accesses the directory is beyond the scope of the X.400-1988 recommendations.

Within the MHS, both UAs and MTAs canaccess the directory; that is, both UAs and MTAs can implement a Directory User Agent (DUA). A UA can present the directory with a directory name and then pass both the name and the resulting O/R address to the MTA. Alternately, the UA can pass the MTA the directory name, and the MTA will access the directory to determine the O/R address of the intended recipient. A functional model of the allowable interaction with the directory from within X.400 is shown in Figure 6.

The use of the directory by X.400 demonstrates the interaction between two different OSI applications to provide a distributed, automatic, user-friendly naming system. In addition, based on information it stores, the directory provides a method of authentication for two MHS functional objects before they establish communication.

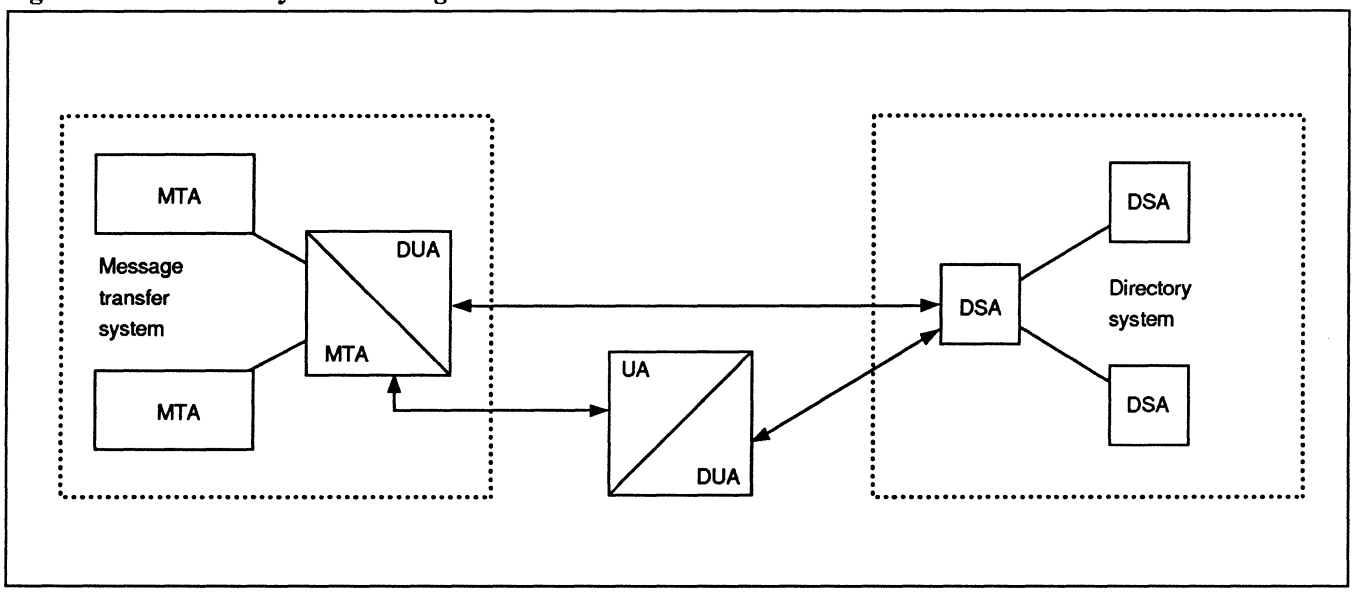
By contrast, in 1984 the X.500 Directory Service had not been completely specified, which made the naming of users in X.400-1984 more difficult. With X.400-1984, the user either had to access a local database or supply the whole O/R address to the UA; it was not possible to supply the MHS with a name.

In addition, X.400-1984 specified that the entity named was the UA, not the user, since the user resided outside the MHS. In 1988, this changed; the entity to be named really *is* the user because a name specifies *for whom* the message is intended. Further, a user is associated with a unique UA, which is the entity to be addressed, because an address specifies *where* the message should be sent.

Security

The 1984 recommendations make no provision for security within the MHS. Messages are transported without any encryption being possible, and there are no mechanisms available to verify the authenticity of an MHS functional object beyond a simple UA and MTA name and password scheme.

Figure 6. MHS Directory Interworking



In X.400-1988 three security risk areas have been identified:

- Intermessage threats, where an intruder could masquerade as a valid user to gain information, a message could be tampered with in transit, or an intruder could record messages and eavesdrop on conversations.
- Intramessage threats, where one of the participants in an exchange of messages could deny involvement in that exchange (of particular importance in financial transactions), or security levels within different domains within the MHS could be breached.
- Data store threats, where an intruder could modify routing information or examine the content of a user's message database.

In order to address these risks, fourteen new optional elements of service have been introduced. These elements of service can be applied to any pair of communicating MHS functional objects (for example, UA/MTA, MTA/MTA, MS/MTA, and so forth) and cover the establishment of an authenticated interaction between objects. Table 1 lists these new security service elements.

Table 1. Security Service Elements

Service Element Name	Description
• Message origin authentication • Report origin authentication • Probe origin authentication	These allow the recipient of the message type (or, in the case of messages and probes, any MTA through which they pass) to authenticate the origin of the message.
• Proof of delivery	Allows the originator to verify a message, its content, and the identity of the recipient.
• Proof of submission	Allows the originator to authenticate that the message was submitted to the MTS with the originally specified recipients.
• Secure access management	Provides for authentication between functional components and the establishment of a security context between them.
• Content integrity	Enables the recipient to verify that the content of a message has not been modified.
• Content confidentiality	Prevents the contents of a message being disclosed to an unauthorized person.
• Message flow confidentiality	Allows the originator to conceal the message flow through the MHS.
• Message sequence integrity	Allows the originator to provide the recipient with proof that the sequence of messages has been preserved.
• Nonrepudiation of origin	Provides the recipient(s) with proof of origin, preventing the originator from falsely denying sending the message.
• Nonrepudiation of delivery	Provides the originator with proof of delivery, preventing the recipient(s) from falsely denying receiving the message.
• Nonrepudiation of submission	Provides the originator with proof of submission, preventing the MTS from falsely denying that a message was submitted for delivery to the originally specified recipients.

The service elements are intended to be implemented within a defined security policy and are designed to protect messages submitted to the MTS by a message store, user agent, or access unit. However, the specification of how they are applied to AUs is not yet available. Moreover, they do not protect user-to-user communication or the communication between the user and the MHS.

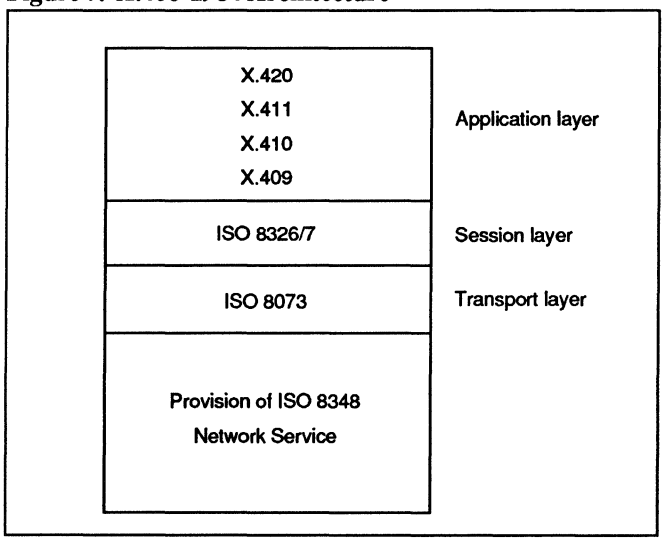
The management of these security features is provided by the directory's authentication framework (X.509). The directory may store certified copies of keys for MHS users that can then be used to provide authentication and key exchange for use in data confidentiality and data integrity mechanisms.

The Architecture of X.400

X.400-1984 Architecture

The 1984 X.400 recommendations were published some five years before the ISO standard Application Layer Structure (ISO 9545) and three years before the Presentation Layer Service Definition and Protocol (ISO 8822/3). It was therefore necessary for X.400-1984 to specify its own application architecture. It did this by using the existing ISO Session Layer Service Definition and Protocol (ISO 8326/7) and having the application define its own presentation layer (X.409-1984), as shown in Figure 7.

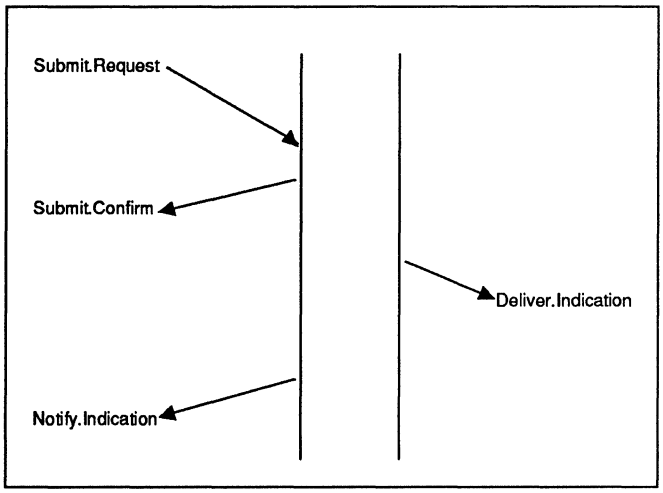
Figure 7. X.400-1984 Architecture



There are two main components in the 1984 MHS—the MTA and the UA—with a third, rarely implemented, component called the Submission and Delivery Entity (SDE) used in conjunction with stand-alone UAs.

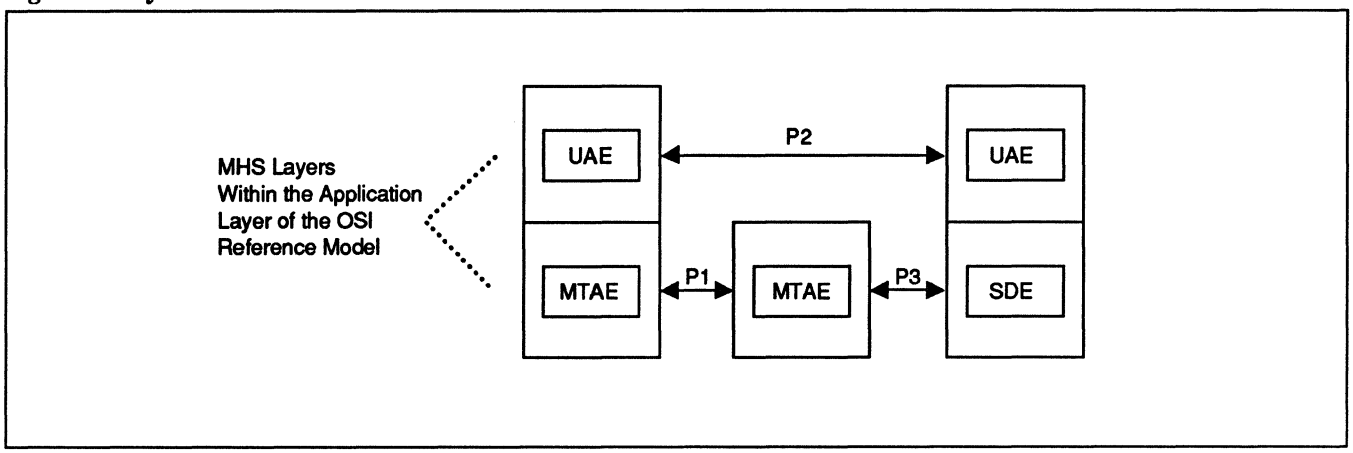
The principle of a layered architecture is carried into the application, with the MTAs and SDEs being considered as the Message Transfer Layer (MTL) and the UAs as the User Layer. In accordance with the rest of the OSI model, the MTL has a boundary service definition, cast in terms of service primitives, that allows a user of the service to submit messages and probes; take delivery of messages, probes, and notifications (reports); and administer the interaction with the MTL. The purpose of the SDE is to present this service boundary to a stand-alone UA. An example of an X.400-1984 primitive sequence is shown in Figure 8.

Figure 8. Example Primitive Sequence, X.440-1984



There are three different protocols used in X.400-1984: the message transfer protocol, called P1, which is used between MTAs for relaying message protocol data units (MPDUs) through the MTS; the SDE-MTA protocol, called P3, which is the protocol used by the SDE to bring the MTL boundary service to the User Layer; and a suite of protocols, called PC, which are used between cooperating UAs. Since X.400-1984 only specifies the IPM service as a user of the MTL, the only example of PC that is defined is used between IPM user agents and is called P2. The relationship between MTAs, UAs, SDEs, and the various protocols is shown in Figure 9.

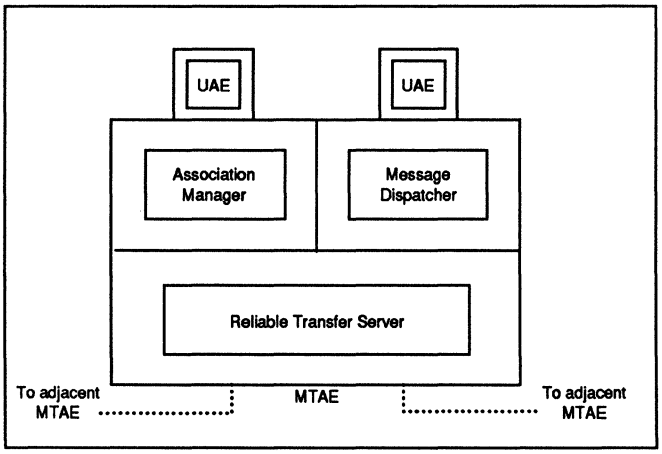
Figure 9. Layered Model of the IPMS - 1984



1984 MTA Architecture

In X.400-1984, the MTA is modeled as having three components: the Message Dispatcher, Association Manager, and the Reliable Transfer Server (RTS) as shown in Figure 10.

Figure 10. Functional Model of an MTA (1984)



Here again, the layered concept is maintained; the RTS being responsible for the interaction with the session layer below it and providing a set of primitives for its interaction with the RTS user. The RTS is specified (X.410-1984) as an intercept on the development of what is now known as the Remote Operations Service Element (ROSE) and the Reliable Transfer Service Element (RTSE). The RTS is responsible for the complete and reliable transfer of MPDUs across a Session Layer connection.

The Message Dispatcher performs the actions of the P1 protocol, indicated by messages submitted by a UA or received from an adjacent MTA for transfer or delivery. It carries out a routing decision based on the type of message, the recipient(s) O/R names, and the record of the message's route to date through the MTS, as recorded in its envelope. If the message is addressed to more than one recipient, the Message Dispatcher may make copies of the message and forward them to different MHS components.

The routing process is not specified in X.400-1984; the recommendations merely state that for a given message, the MTA must carry out a routing algorithm on each recipient O/R address and for each one determine which of the following applies:

- This MTA services the recipient's UA.
- The MPDU must be relayed to another MTA.
- There is an error in the O/R-address.

In the second possiblility, above the routing algorithm must provide sufficient information for the other MTA to be unambiguously identified by the Message Dispatcher.

To send MPDUs to MTAs located in different processing systems, the MTA initiates an *application association* between itself and its peer MTA. The management and control of all such associations is the responsibility of the Association Manager. In the hybrid X.400-1984 architecture, the Association Manager's effect is to

manage and control the session layer connection by using the OPEN.Request and CLOSE.Request service primitives of the RTS. The Association Manager will govern the allowable associations in accordance with bilateral agreements between MTAs covering the following:

- The maximum number of associations that can exist simultaneously.
- Whether one-way or two-way alternate associations are used.
- Which MTA has responsibility for establishing the associations.
- Whether associations are permanently established or established and released as required.

1984 User Agent Architecture

In X.400-1984, the UA is modeled as a single entity, utilizing the services of the MTL via the service primitives it supplies. The recommendations specify only one class of UA, the IPM UA for interpersonal messaging. This class of cooperating UA is specified in X.400-1984, along with the syntax and semantics of the P2 protocol. The P2 protocol specifies how the UA constructs User Agent Protocol Data Units (UAPDUs), which comprise the content of the messages exchanged between IPM UAs; the operations relating to the exchange of UAPDUs that the IPM UA must perform; and the rules that an IPM UA must follow when using the MTL service to convey UAPDUs.

1984 SDE Architecture

The SDE makes the MTL service available to a stand-alone UA, as can be seen in Figure 9. The SDE does not participate in the MTS relaying function, but merely carries information between the UA and its associated MTA.

The SDE-MTA protocol, P3 (specified in X.411-1984), uses the remote operation service and RTS defined in X.400-1984. This enables the RTS to carry operations such as submit message, submit probe, deliver message, and deliver notification (report), in addition to the various management and control operations between two MHS components

X.400-1988 Architecture

By the time the 1988 version of X.400 was being prepared, the specification of the upper layers of the OSI reference model had stabilized, enabling the X.400 application to be described as a true OSI application interacting with generic application service elements (ASEs) and other OSI applications, such as the X.500 Directory Service.

Use of the Presentation Layer

Within the final OSI reference model, the X.400 application uses the services of the Presentation Layer to convey application data structures, defined in terms of an abstract syntax, across a session connection by converting them into an encoded form (the transfer syntax) mutually agreed to by the peer presentation entities.

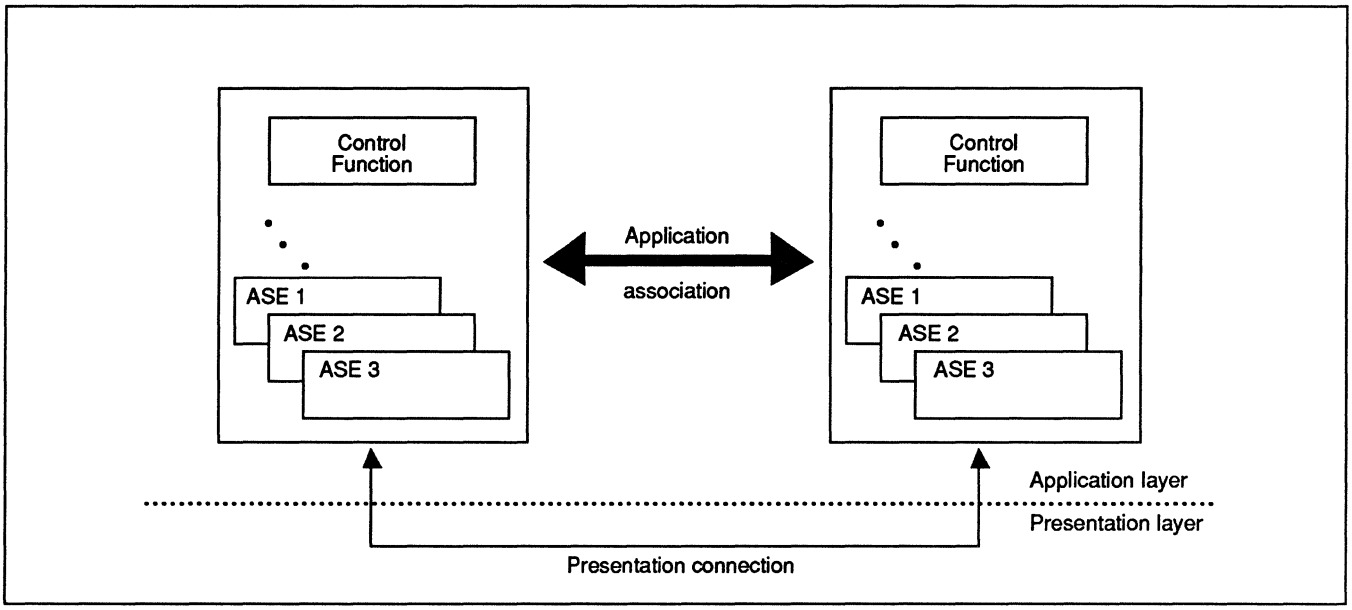
Standardized presentation service primitives are then used to initiate presentation connections and to control the flow of abstract data across the connection once it has been established. The important point is that the user of the presentation service does not need to worry about the encoding rules for the abstract data structures used; rather, it need only deal with those structures themselves.

Application Layer Structure

Within the OSI application model (as specified in ISO 9545), an application is made up of a number of Single Association Objects (SAOs) each of which can be engaged in a single application association with another SAO. An SAO is made up of a control function, which determines its functionality; an Association Control Service Element (ACSE), which initiates and controls the association with the peer SAO; and a number of "user" ASEs whose role is to support the functional requirements of the control function. The structure of an SAO is shown in Figure 11.

Two SAOs can form an application association between themselves, provided they are both able to support the same *application context.* This is a detailed specification, laid down in an international standard, for the use of an association between open systems in a particular environment. For example, X.400-1988 defines several application contexts to allow the provision of MHS

Figure 11. Structure of a Single Association Object



capabilities by processing systems of varying capabilities. The application context specifies, among other things, which ASEs must be present in an SAO to support that application context.

The particular application context to be used across an association is negotiated by the ACSE, which must be present in every SAO.

The 1988 MHS Model

The MHS is described in X.400-1988 in terms of an *abstract model.* This ensures that functionality is defined independently of a concrete realization, since each aspect of the task can be realized in several different ways.

An abstract model is a description of a distributed information processing task and of the environment in which it is carried out. It is based upon the concepts of *abstract objects, ports, services,* and *refinements.*

An *abstract object* is a functional entity that can interact with another abstract object in the abstract model. For example, one abstract object might be a system, and a collection of other abstract objects might be its users. Abstract objects interact by means of *abstract ports* (or ports); for example, one port might represent the way an abstract object accesses a directory service that is represented by another abstract object with one or more ports representing its access points.

Two objects interact by means of a port in one and a port in the other only while those ports are associated with each other, or *bound.* Two ports can only be bound if they *match.*

Abstract ports are of particular importance because they not only mark the boundary between objects in the abstract model but, in concrete realizations of the model, they mark the boundary between physical systems realizing those objects.

An *abstract service* is the set of capabilities offered by one object to another by means of one or more of its ports. The object offering the abstract service is called an *abstract service provider* (or provider), and an object to which the capability is offered is called an *abstract service user* (or user). A particular abstract service can have any number of providers and users. An abstract service is described in terms of *abstract procedures* (procedures), tasks that one object carries out at another's request. There are four types of abstract procedures: *abstract bind, abstract unbind, abstract operation,* and *abstract error.*

An *abstract bind* is the procedure used to bind two ports. An *abstract unbind* unbinds two bound ports. An *abstract operation* is an operation defined within the context of two bound ports; it is *invoked* by one port and *performed* by the other. The definition of the ports involved determines which is which. An *abstract error* is an exceptional condition that may arise during the performance of an abstract operation. It is an abstract procedure requested by the performer of the invoker of the erroneous operation.

An abstract service, therefore, serves much the same purpose within the Application Layer as a boundary service definition does in the lower layers of the OSI reference model.

An *abstract refinement* (or refinement) is a functional decomposition of an abstract object into a number of component objects together with any associated ports and abstract service definitions.

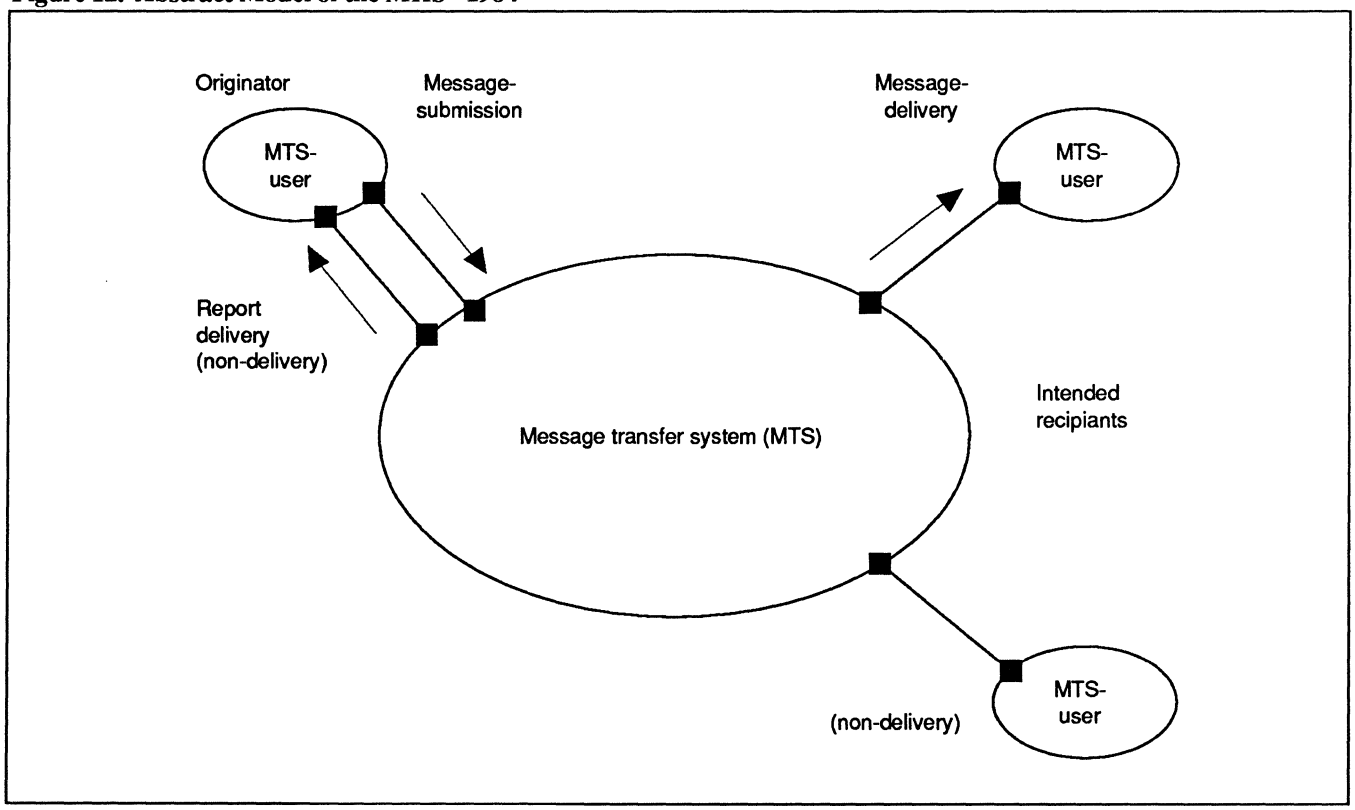
The MTS Abstract Model

The abstract model of the MTS comprises a single object representing the MTS, with the services of the MTS being made available at abstract ports. A user of the MTS is also modeled by a single object that obtains services from the MTS through a port when it is bound to a matching port of the MTS. The model is shown in Figure 12.

The MTS object supports ports of three different types: a *submission port*, a *delivery port*, and an *administration port*. MTS users are also modeled as objects having the same three ports but as consumers of the service supplied by the MTS ports.

The abstract bind and unbind operations that are specified between an MTS user and the MTS bind and unbind all three ports of the MTS user to three matching ports of the MTS. In addition, the bind operation establishes the security context within which any subsequent abstract operations will be performed.

Figure 12. Abstract Model of the MHS - 1984



The following abstract operations are defined for the three port types:

- Submission port
  - Message submission
  - Probe submission
  - Cancel deferred delivery
  - Submission control

- Delivery port
  - Message delivery
  - Report delivery
  - Delivery control

- Administration port
  - Register
  - Change credentials

The abstract operation "cancel deferred delivery" follows the submission of a message to the MTS (via a message submission operation) with the optional request that its delivery be deferred for a specified length of time. The control operations limit the abstract operations that the performing port may invoke. The change credentials operation enables either the MTS or MTS user to change the credentials held by the other; credentials are used to authenticate MTS and MTS user during the bind process.

MTS Refinement

Rather than being a single entity, the MTS comprises a collection of MTAs. This is modeled by a refinement of the MTS object just described into a collection of component MTA objects. The MTA objects also have ports, those visible at the boundary of the MTS, as described above, and an additional port, internal to the MTS, called a *transfer port.*

A transfer port enables an MTA to transfer messages to another MTA object within the MTS. The transfer port has another pair of bind and unbind operations associated with it for binding the port to a transfer port on another MTA.

The following abstract operations are defined for a transfer port of an MTA object:

- message transfer
- probe transfer
- report transfer

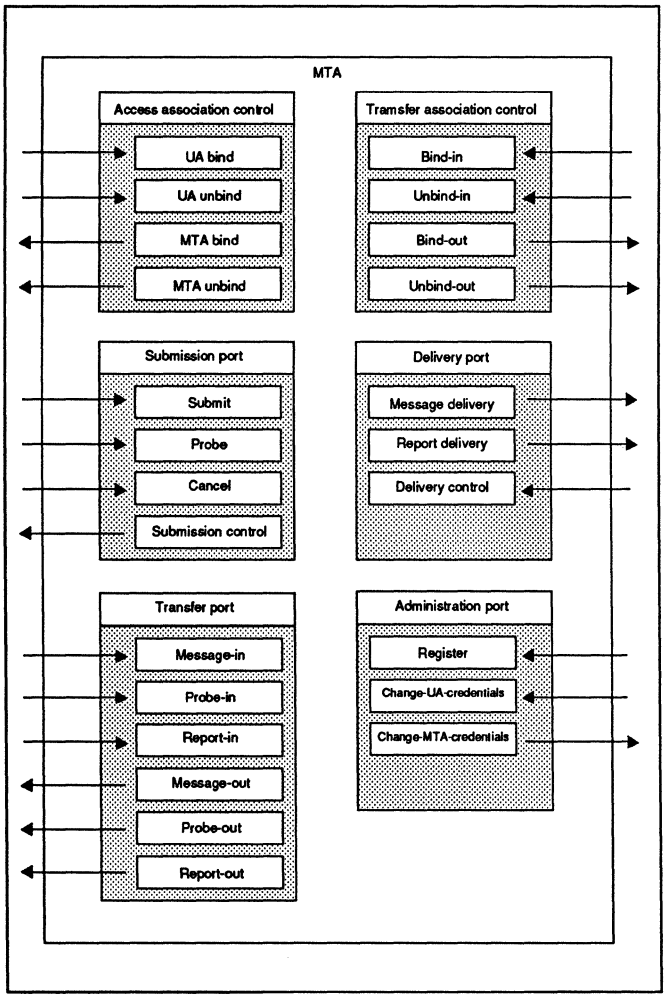
1988 MTA Architecture

The 1988 architecture of the MTA is divided into two parts, the external and internal features. The model of the external features is the same as the MTA object described above and is shown in Figure 13.

The internal features are those modules of functionality required to support the external behavior of the MTA and comprise the main module, the report module, and the deferred delivery module.

The main module is responsible for trace processing, loop detection, routing, recipient redirection, content conversion, distribution list expansion, message replication, origin authentication, and name resolution. The

Figure 13. Functional Model of an MTA



report module generates and routes all report messages, and the deferred delivery module acts as a buffer for those messages that have been submitted to the MTA with a request that they are not forwarded until a specified time.

The relationship between the modules of functionality within the MTA is shown in Figure 14.

1988 MS Architecture

With the terminology introduced above, the description of the MS becomes straightforward. It is modeled as a single abstract object with six ports; three are to match those presented by an MTA—the submission, delivery, and (MS-MTA) administration port; and three are to provide services to a UA—the indirect submission, retrieval, and (UA-MS) administration ports. The MS user is also modeled as an object with an indirect submission, a retrieval, and an administration port, but in addition as a consumer of the service supplied by the MS object. This functional model is shown in Figure 15.

Figure 14. Functional Modules Within an MTA (1988)

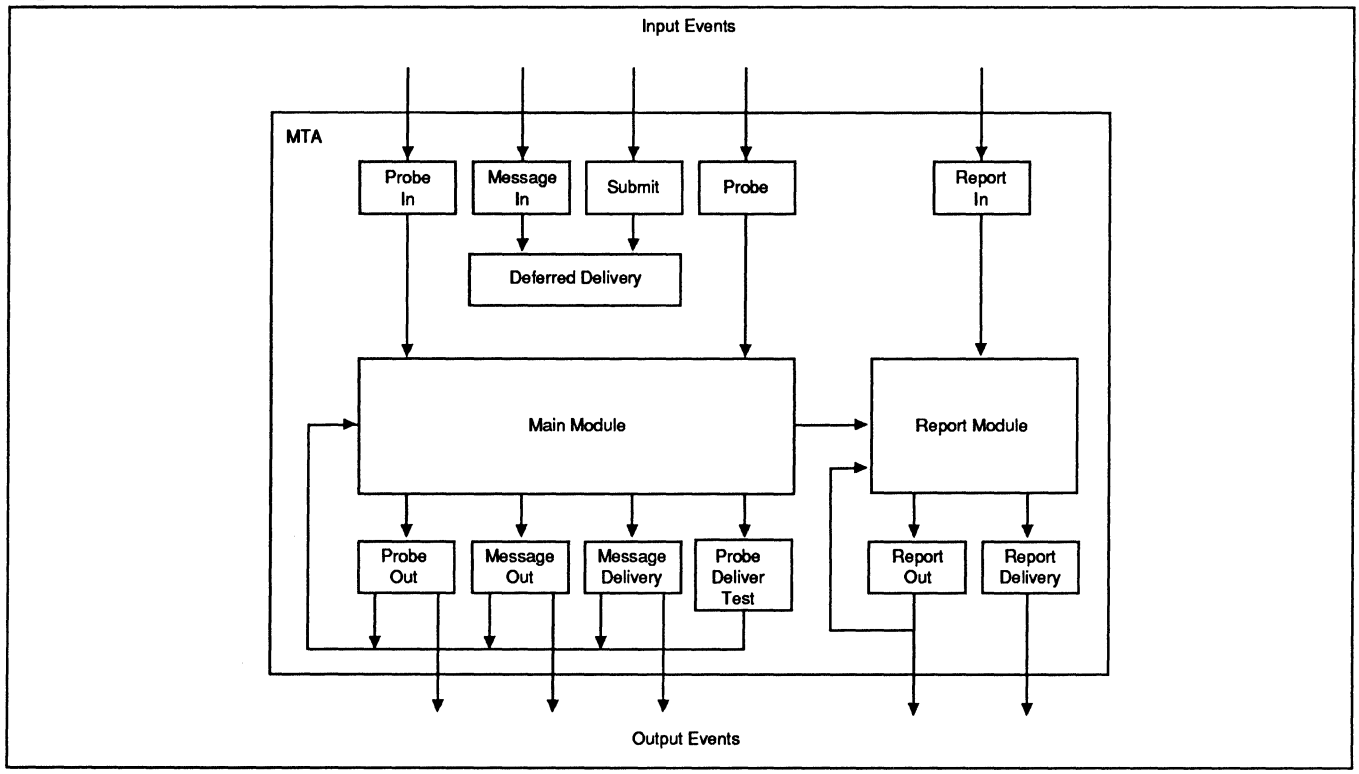
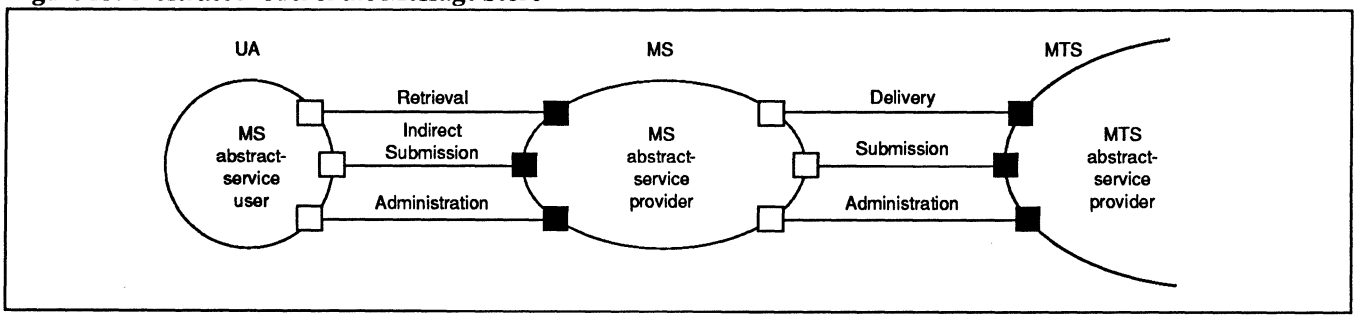


Figure 15. Abstract Model of the Message Store



The MS object is an MTS user and, as such, the MS submission, delivery, and (MS-MTS) administration ports are consumers of the MTS-supplied service.

The abstract operation that binds the MS user object to the MS object binds all three ports of the MS user object to the three matching ports of the MS object. In addition, it authenticates the MS user to the MS object (and vice versa) and establishes the security context within which the interaction will take place.

The indirect submission port and the administration port are defined in an identical manner to the submission and administration port defined for the MTS and MTS user objects.

The following abstract operations are defined for the retrieval port:

- Summarize
- List
- Fetch
- Delete
- Register-MS
- Alert

The Summarize operation returns summary counts of selected entries in the MS's database. In addition, it returns a count of the entries selected and their lowest and highest sequence numbers. The Alert operation allows the MS to indicate to the MS user that a message has arrived in the MS from the MTS.

1988 UA Architecture

The UA architecture model is one of two types, depending on whether or not the UA accesses an MS. If the UA does not subscribe to an MS, it is modeled as an MTS user object with submission, delivery, and administration ports. If the UA does subscribe to a message store, it is modeled as an MS user object with indirect submission, retrieval, and administration ports.

1988 Protocols

When implementing the abstract model for X.400 in an OSI environment, the functionality is provided by a number of application service elements (ASEs) that support the services of each type of port in the model.

These ASEs are, in turn, supported by standard ASEs to form single association objects (SAOs). Depending on the implementation, the ASEs may be combined to provide a number of application contexts within which the protocols of X.400-1988 are operated.

The ASEs defined by X.400-1988 are as follows:

- MSSE - the Message Submission Service Element
- MDSE - the Message Delivery Service Element
- MRSE - the Message Retrieval Service Element
- MASE - the Message Administration Service Element
- MTSE - the Message Transfer Service Element

There are three protocols used in X.400-1988: the MTS access protocol, called P3; the MS access protocol, called P7; and the MTS transfer protocol, P1. The protocols have names similar to those in X.400-1984, but are significantly different. The Application Layer structures for these protocols are shown in Figures 16, 17, and 18, respectively.

Each application layer structure in Figures 16, 17, and 18 may also contain the Reliable Transfer Service Element (RTSE), an optional addition. The addition of this service element enables the corresponding ACSE to negotiate further, reliable application contexts with peer SAOs.

The X.400-1988 recommendations do not restrict the implementation of more elaborate MHS application architectures—for example, employing multiple SAOs in one application entity.

The specification of the 1988 protocols is achieved by setting down the formal definitions for the allowable application contexts and assigning numeric identifiers to the remote operations required by the protocol. These correspond exactly to the abstract operations defined for the ports (realized through ASEs) involved in the protocol. In addition, the abstract errors associated with the operations are also assigned numeric identifiers.

For example, in the P3 protocol, for the MTS user to submit a message to the MTS, the Message Submission operation is carried out across a bound submission port. This abstract operation has been assigned the numeric identifier "3", so within the prevailing application

Figure 16. MTS Access Protocol, P3

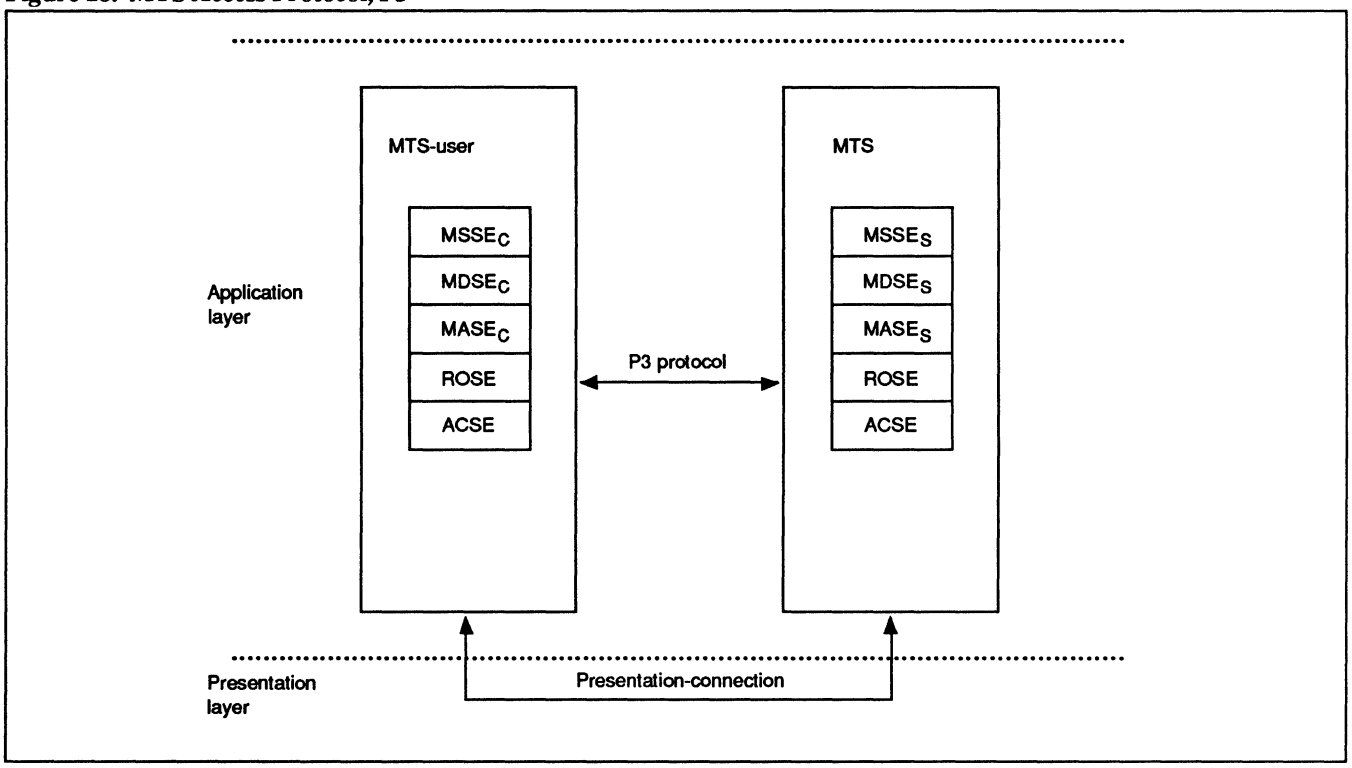


Figure 17. MS Access Protocol, P7

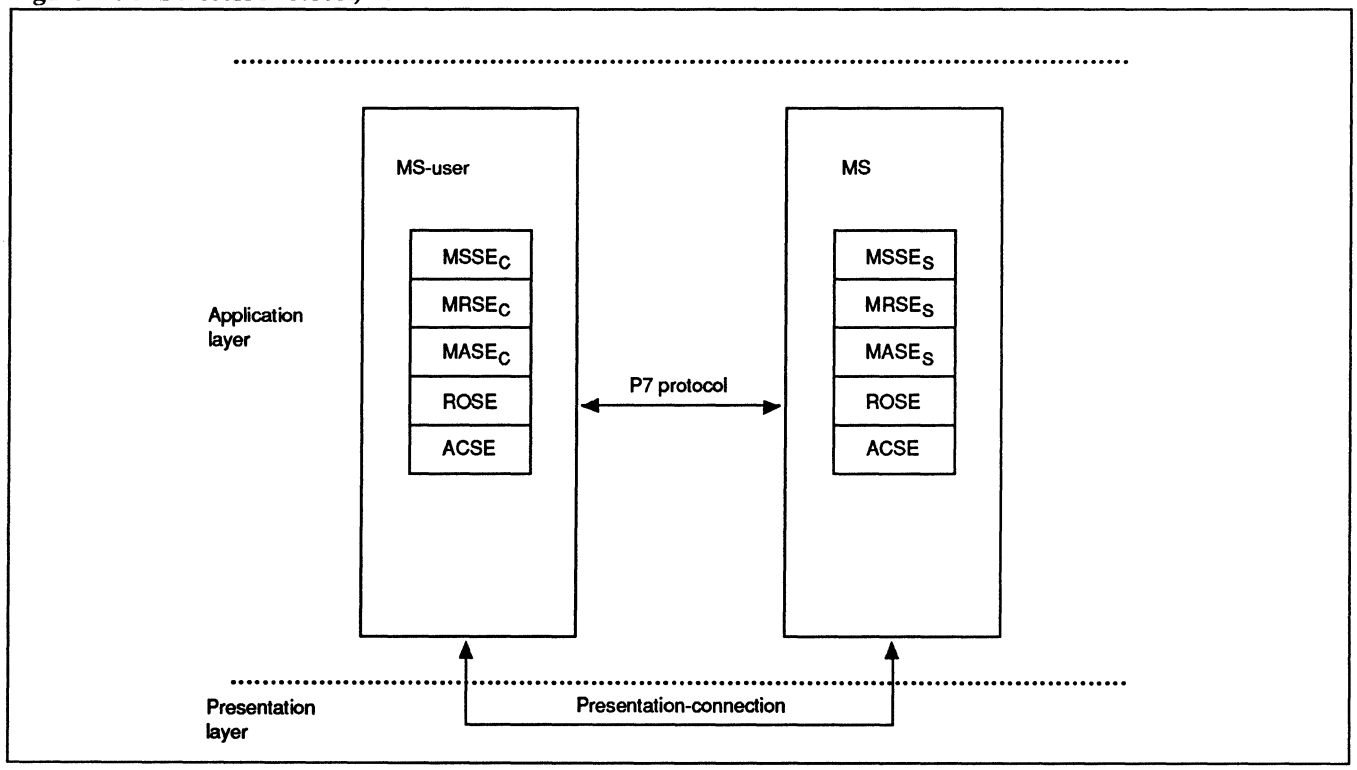
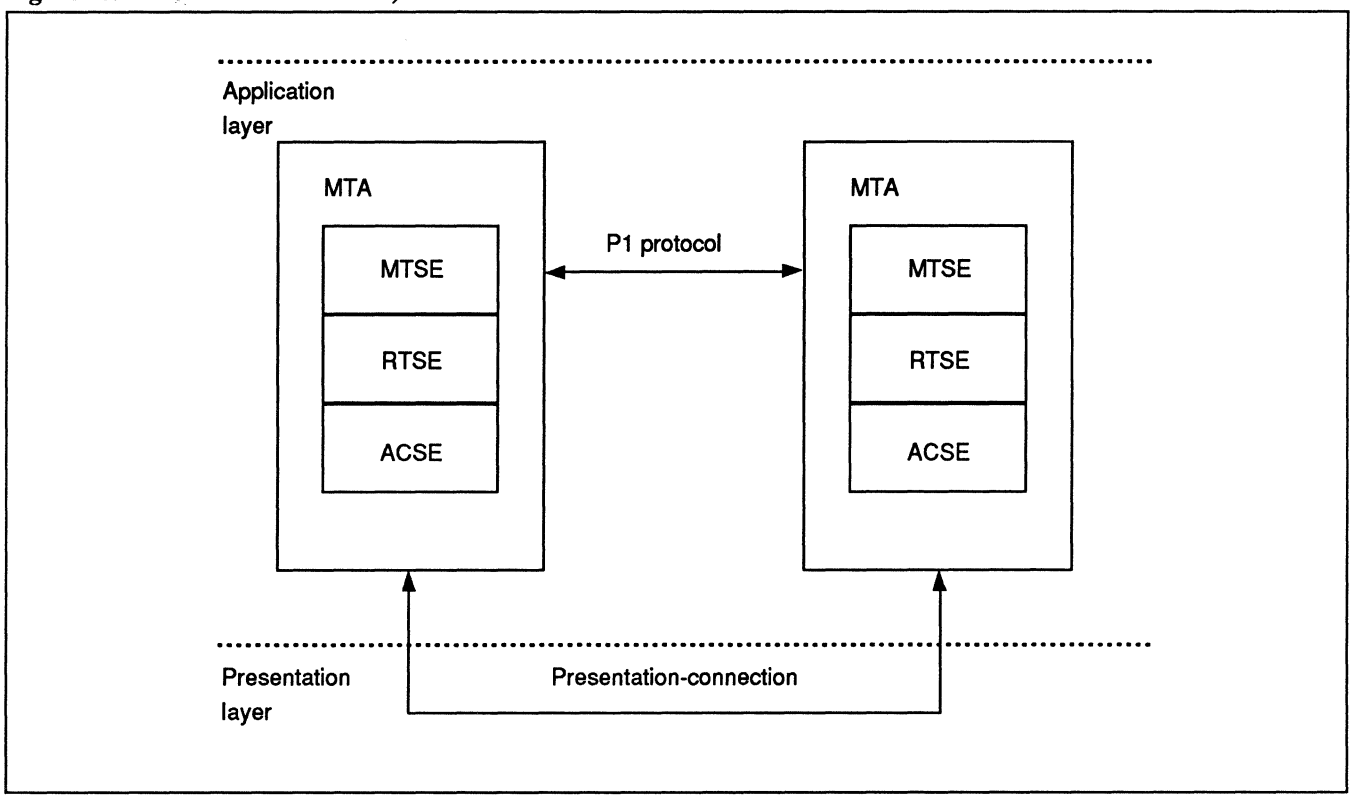


Figure 18. MTS Transfer Protocol, P1



context, the remote operation "3" is understood by the recipient to be a message submission. The rest of the abstract syntax for the data unit that is passed by the MTS user to the presentation layer is specified in the abstract service definition of the MTS.

Comparison of 1988 and 1984 Protocols

The P1-1988 protocol is functionally the same as P1-1984, with amendments to support the enhanced services of X.400-1988.

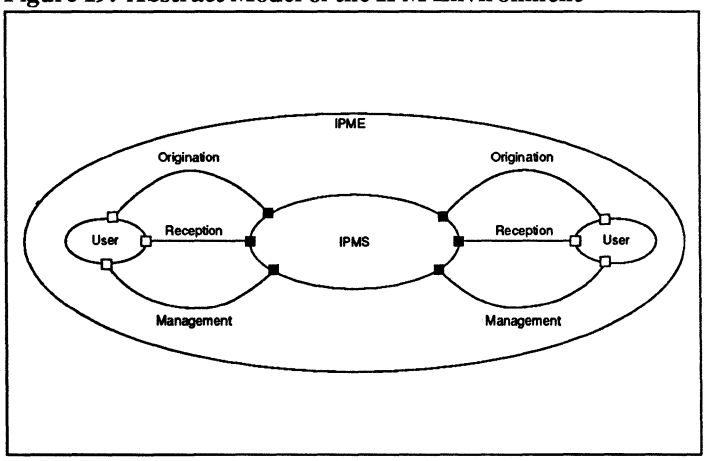
The P3-1988 protocol was developed from the P3-1984 protocol, and is considerably different. It was never particularly favored by commercial implementors because the supporting architecture was not stable. The 1988 version is quite stable but is still unlikely to be implemented very widely, because it is anticipated that the majority of commercial systems will implement colocated MTAs and MSs.

The P7-1988 protocol has no counterpart in the 1984 version.

Interpersonal Messaging in X.400-1988

The provision of the IPMS is modeled in X.400-1988 as an IPMS object with origination, reception, and management ports (as suppliers of the IPMS) and IPMS user objects with that port corresponding to the IPMS object but as consumers of the service supplied by the IPMS. The model is shown in Figure 19.

Figure 19. Abstract Model of the IPM Environment



The method of specification can be used for any application utilizing the services of the MHS. Abstract operations that provide the fundamental elements of service for the MHS are defined for the ports as follows:

- Origination port
  - OriginateProbe
  - OriginateIPM.....(Interpersonal Message)
  - OriginateRN.....(Receipt Notification)

- Reception port
  - ReceiveReport
  - ReceiveIPM
  - ReceiveRN
  - ReceiveNRN.....(Nonreceipt Notification)

- Management port
  - ChangeAutoDiscard
  - ChangeAutoAcknowledgement
  - ChangeAutoForwarding

The ChangeAutoDiscard operation enables or disables the automatic discard by the IPMS of expired or obsoleted IPMs delivered to but not yet received by the user. The ChangeAutoAcknowledgement operation enables or disables the automatic origination of receipt notifications by the IPMS on the user's behalf. The ChangeAutoForwarding operation enables or disables the automatic forwarding of IPMs to prespecified users or DLs.

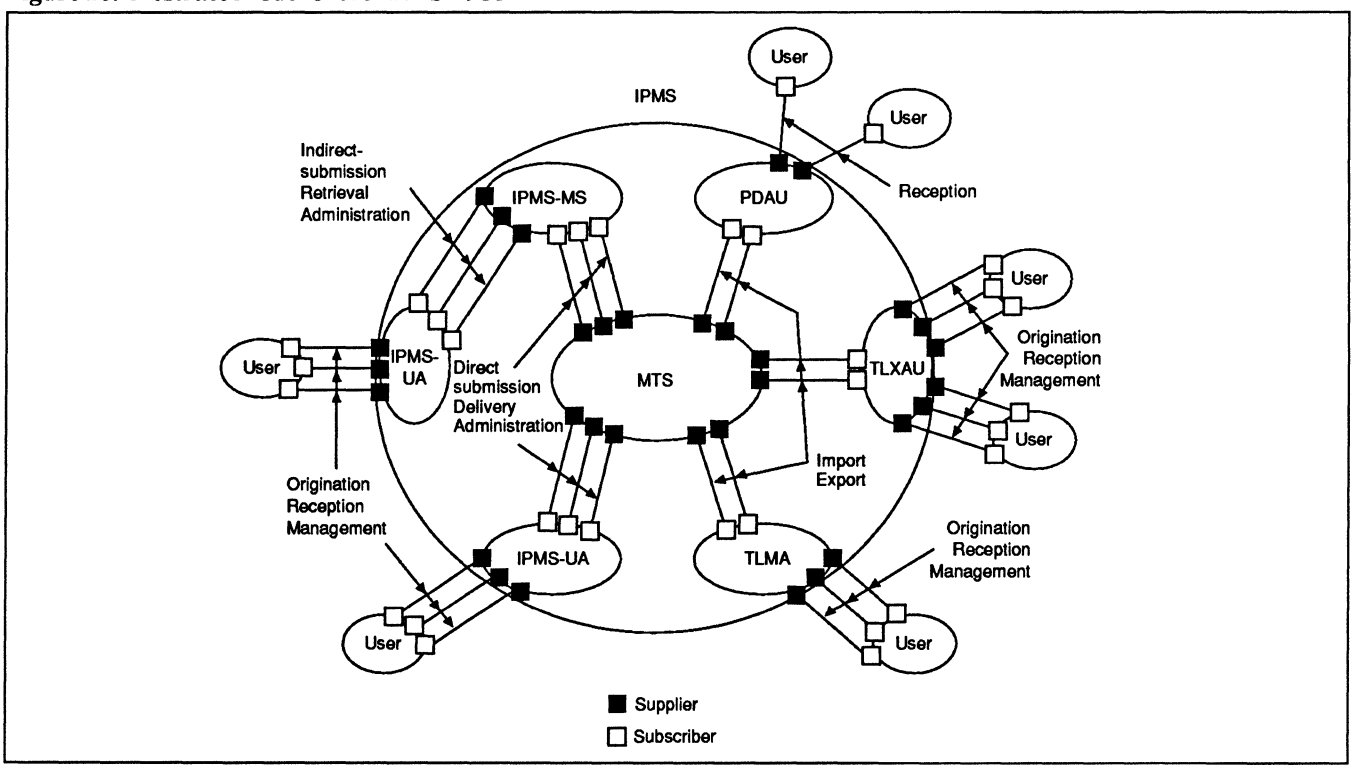
IPM User Agents, Message Stores, and Access Units

The IPM user agent has additional functionality to support the IPMS. In addition to the submission, delivery, retrieval, and administration ports, it has origination, reception, and management ports defined as suppliers of the corresponding elements of service to IPMS users.

The IPM MS has the same structure as defined above for the MHS.

In addition, the IPMS defines the AUs required to access telex and teletext as MTS user objects with origination, reception, and management ports, and the AU to the physical delivery service as an MTS user object with a reception port. The IPMS model is shown in Figure 20.

Figure 20. Abstract Model of the IPMS-1988



Conclusions

With the arrival of X.400-1988, the message-handling system is ready to be used to carry forms of messages other than simply interpersonal messages, such as Electronic Data Interchange (EDI) and Office Document Architecture (ODA) messages.

The widespread installation of public X.400 services already means that IPMS users can exchange messages internationally and, as the public service providers migrate their systems toward the 1988 version, we will see more and more opportunity for more advanced messaging applications.

There will be no loss of service to the user of the 1984-based IPMS leasing a connection into a public MTA, because one of the mandatory conditions of X.400-1988 is that implementations support an application context designed to work with an MTA using the 1984 version. Thus, two 1984 User Agents will still be able to exchange messages over a public 1988 MTS via 1984-compliant MTAs.

There have even been suggestions that X.400 could remove the need for FTAM in many environments. Certainly, while the majority of FTAM implementations remain no more than file transfer mechanisms, the idea seems to have some benefit in many cases, and X.400 has the capability to do this in both the 1984 and 1988 versions.

Finally, in the future, when facilities such as ISDN become readily available for home use and other isolated public data capability requirements, the X.400-1988 MHS will be there to provide the basic message handling across what will, by then, be a network as extensive as the phone network itself, maintaining its position as the most widely used OSI application in the world. ■

Dr. Andrew Bartholomew works for 3Com (UK) Limited based in Marlow, England, as a systems consultant specializing in OSI. He has been involved with X.400 for the past two years, working on beta test programs as well as giving presentations at the 3Wizard conferences in 1989 and 1990.

Dr. Bartholomew graduated from Sussex University, England, with a D.Phil. in Mathematics. Before joining 3Com, he worked for an Ethernet bridge manufacturer implementing, among other things, one of the early versions of the Spanning Tree Algorithm.

References

CCITT Red Book Volume VIII, Fasicle VIII.7, Recommendations X.400-X.430, 1984

CCITT Blue Book Volume VIII, Fasicle VIII.7 Recommendations X.400-X.420, 1988

CCITT Blue Book Volume VIII, Fasicle VIII.8 Recommendations X.500-X.521, 1988

Acronyms used in this article:

ACSE: Association Control Service Elements
ADMD: Administrative Management Domain
ASE: Application Service Elements
AU: Access Unit
CCITT: Consultative Committee on International Telephone and Telegraph
DL: Distribution List
EDI: Electronic Data Interchange
FTAM: File Transfer Access and Management
IPMS: Interpersonal Messaging System
IPM: Interpersonal Message
IPM-UA: Interpersonal Messaging User Agent
ISO: International Standards Organization
MASE: Message Administration Service Element
MDSE: Message Delivery Service Element
MHS: Message-Handling System
MPDU: Message Protocol Data Unit
MRSE: Message Recovery Service Element
MS: Message Store
MSSE: Message Submission Service Element
MTA: Message Transfer Agent
MTL: Message Transfer Layer
MTSE: Message Transfer Service Element
ODA: Office Document Architecture
O/R: Originator/Recipient
OSI: Open Systems Interconnect
PDU: Protocol Data Unit
PRMD: Private Management Domain
PTT: Public Telephone and Telegraph
ROSE: Remote Operations Service Element
RTS: Reliable Transfer Service
SAO: Single Association Objects
SDE: Submission and Delivery Entity
UA: User-Agent

Understanding PPP: The Point-to-Point Protocol

by Dino Farinacci

The Point-to-Point Protocol, commonly known as PPP, is a newly developed protocol that uses a serial point-to-point communication link to connect equipment manufactured by different vendors . Although PPP is new on the scene, many internetworking vendors have jumped on the bandwagon. 3Com was one of several vendors demonstrating PPP interoperability at the Interop conference held in October 1990.

This article explains how PPP enables multivendor interoperabilitiy and describes the the protocol's features. It is intended for network managers and administrators responsible for configuring internetworks. PPP offers administrators who deal with many organizations using multivendor computer equipment a nonproprietary way to connect them into an internet.

What Is PPP?

The Point-to-Point Protocol was designed to provide interoperable serial line networking. It was developed by a working group of the Internet Engineering Task Force (IETF), a standards body funded by the U.S. government to perform computer networking research and standardization. The first set of PPP specifications (known as "Request for Comment" or RFC documents) was released in July 1990[1,2].

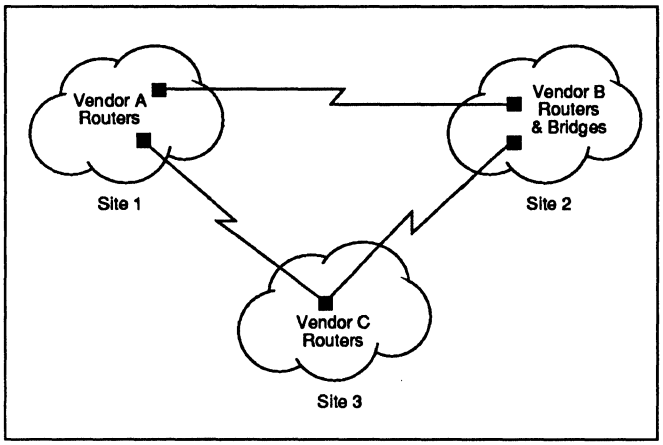
PPP operates at the Data Link Layer of the OSI Reference Model Architecture. It defines an encapsulation format used to transmit data packets across the communications link. PPP has two major components: a Link Control Protocol (LCP) designed to configure and manage the data link connection and a series of Network Control Protocols (NCPs) used to negotiate the operation of different network layer protocols and their protocol-specific options. PPP also provides a protocol multiplexing capability to enable packets generated by different protocols to use the same point-to-point line in environments where more than one protocol is used. This multiplexing capability also is available with other network media such as Ethernet and X.25.

Traditionally, interoperability among different vendor equipment was (and still is) achieved in LAN environments via Ethernet or Token Ring networks, as well as in WAN environments via X.25 Public Data Networks. Some protocol implementors used the X.25 protocol over point-to-point links to solve the serial line interoperability problem. However, this created additional overhead that was undesirable. Specifically, the X.25 Packet Level provides a connection-oriented service that requires connection setup and flow control properties not necessary for a datagram delivery service. Careful tuning of X.25 parameters is also required to handle data burst situations so performance degradation and packet loss can be reduced.

In terminal-to-host connectivity via serial lines, PPP can be used as the Data Link Protocol. The negotiation facilities in PPP allow flexibility to assign addresses and perform compression/authentication functions.

PPP can operate over synchronous or asynchronous communications lines. Internetworking vendors typically implement the synchronous line discipline to connect routers and bridges via high-speed lines (see Figure 1). PPP is very useful in an environment where

Figure 1. PPP Connects Sites Using Different Vendor Equipment



several sites that use equipement from different vendors are connected using point-to-point connections. PPP allows this environment to become an internet that is vendor-independent.

Terminal server vendors implement the asynchronous line discipline to support remote access communications from dial-up terminals or PCs that employ start/stop, character-oriented framing. PCs implementing the TCP/IP protocol suite can have access to all the applications that run over the protocol if a PPP data link is used. This includes, among others, the File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), TELNET, and the Network File System Protocol (NFS). Figure 2 illustrates remote PCs running such applications over a PPP link.

The PPP protocol suite is capable of operating over various physical interfaces and puts no requirement or restriction on the physical media used. Modem signals can be used to aid PPP to determine if the link is ready for data transmission; however, PPP does not require the availability of these signals to operate.

PPP's Encapsulation Method

PPP defines a portion of the Data Link Layer operation and the encapsulation method. The framing of the data is specified in the ISO 3309-1979 HDLC standard for synchronous transmission and ISO 3309-1984 for asynchronous transmission. HDLC requires that all data link frames be delimited with a flag sequence (see Figure 3).

Figure 2. PPP connects Remote PCs to a Host System for a File Transfer Application

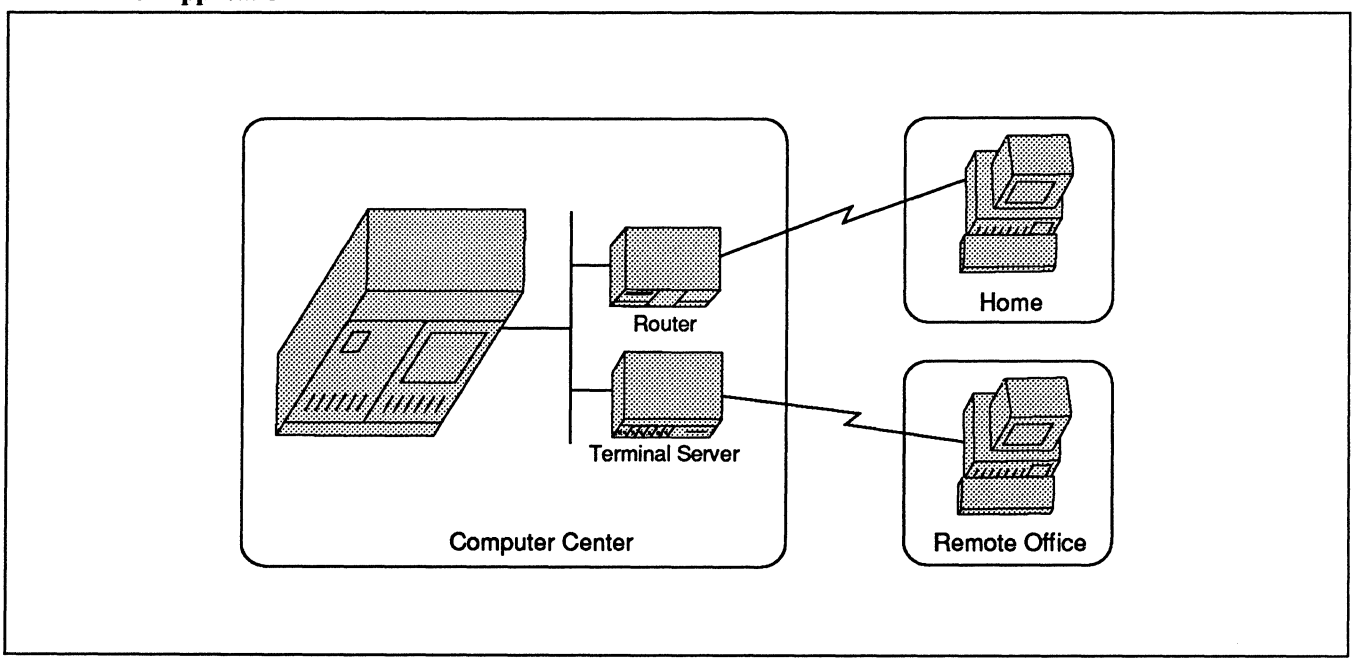
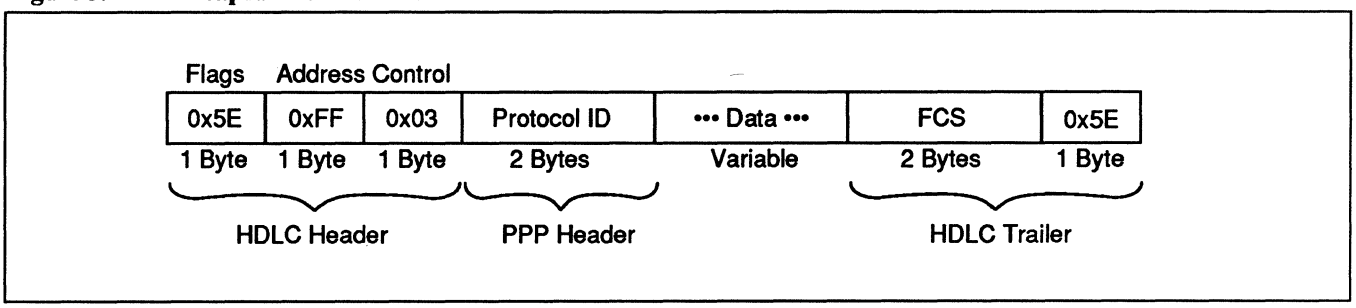


Figure 3. PPP Encapsulation Format



The flag sequence is the 8-bit value 01111110 (or 0x7E). The address and control fields defined in an HDLC frame will be set to 0xFF and 0x03, respectively. These settings mean that the HDLC address is not used and the frame type indicates an Unnumbered Information frame.

After the HDLC header, PPP defines a two-octet protocol ID value used for demultiplexing packets. This allows different protocols to run over serial PPP links. Currently, the PPP standard specifies values for TCP/IP, OSI, DECNET Phase IV, XNS, Appletalk, Novell IPX, VJ Compression TCP/IP, and bridged packets.

Following the PPP header is an information field that may be variable in length. After the information field is a required Frame Check Sequence (FCS) for protecting the data and headers. The PPP protocol has provisions to negotiate use of 16- or 32-bit FCS algorithms.

Operation and Functions of PPP

When a serial link is initialized, PPP operates different phases of its protocol before the link is considered to be fully operational for data transmission. The PPP Link Control Protocol is the initial phase. The Link Quality Determination phase is performed next if negotiated successfully from the LCP phase. The third phase is the network layer protocol configuration, which is executed for each network layer that will run over the link. This is done via the NCPs. The following describes each of these phases in detail.

Link Control Protocol

PPP uses the LCP protocol to establish the link. When establishing the link, it is critical to verify that there is full-duplex operation of the communications line. LCP Configure-Request packets and corresponding Configure-Ack packets are exchanged by each end of the serial line. Each Configure-Request requests negotiation of specific options.

The following lists the LCP options that are available to negotiate the link is being initialized:

Maximum Receive Unit

Indicates to the other end of the connection the largest frame it may transmit.

Asynchronous Control Character Mapping

Indicates to the other end of the connection which control characters should map into 2-octet sequences. This option is used when PPP operates over an asynchronous communications line.

Authentication Type Negotiation

Communicates to the other end of the connection which authentication scheme will be used. Successful negotiation of this option adds an additional phase to the LCP when a link is being initialized.

Magic Number Negotiation

Determines whether the communications link is in loopback mode. Each end of the communications link selects a unique 32-bit value. When packets are received with the local magic number, this indicates that the communications line is either in local or remote loopback mode.

Link Quality Monitoring
Enables Link Quality Monitoring (LQM) on the communications link. This option is used to determine the error rate of the communication link. If it is successfully negotiated, link quality report packets will be sent at the negotiated rate (see Link Quality Monitoring below for a detailed discussion).

Protocol Field Compression
Determines whether the PPP protocol ID field should be compressed from two octets to one. This is useful for very slow-speed links where overhead from the protocol can be reduced.

HDLC Address/Control Field Compression
Used to determine whether the HDLC address and control field should be eliminated in subsequent frames. Again, this is useful for slow-speed links where overhead can be reduced.

Link Quality Monitoring

The LQM feature is enabled only if both sides of the communication link negotiate the option successfully. LQM tests the link for the quality of transmission it is providing. If the link is not providing the desired quality, rerouting on alternative paths can occur. PPP specifies the mechanism used to determine data loss on the communications link. The policy that is used to determine whether the link is usable is left to the implementation, so vendors can implement different policies using a consistent mechanism. For example, in networks where faulty links exist, a network administrator may want to consider the link usable with 50 percent data loss. This policy can be configured by the network administrator so that PPP will disable the link when the threshold is reached.

The sentences that follow describe the LQM procedure. Each side of the communications link transmits Link Quality Report packets indicating the number of packets/octets transmitted and received successfully over the link. The receiver of these packets compares them with the values received from previous Link Quality Report packets. Using this mechanism, the amount of data loss over the link can be determined. Since PPP does not specify any policies regarding when a link is considered usable, it does recommend a "K out-of N" hysteresis approach, so the link does not oscillate to or from UP/DOWN status. If K successes occur in the last N periods, the link is considered to have good quality.

Network Control Protocol

The NCP protocol provides a network layer-specific capability to negotiate options. All protocols that wish to operate over a PPP communications line are required to identify themselves by exchanging Configure-Request/Configure-Ack packets (whether or not there are options to negotiate). For example, a multiprotocol router exists that supports TCP/IP and XNS on one side of the link and another single-protocol TCP/IP router on the other side. The multiprotocol router will send two Configure-Request packets over the PPP link, one for TCP/IP and one for XNS. The receiving side of the link will return a TCP/IP Configure-Ack packet and an XNS Configure-Nak packet. The result of this protocol operation is that only TCP/IP traffic will be sent over the PPP link.

The NCP-specific options have been defined, by the PPP standard, for the TCP/IP network layer. Other network layer-specific options are being defined but have not yet been standardized. In addition, an NCP is being defined to allow bridging over PPP communication lines.

3Com's Implementation of PPP

Synchronous PPP support is provided in 3Com's BR/3000 multiprotocol brouter product line. All five network layer protocols (TCP/IP, OSI, DECNET, XNS, and IPX) run over the PPP communications link. In addition, bridged data traffic may be transmitted concurrently with the network layer protocols being routed.

The 3Com PPP implementation has been successfully interoperating with several other vendors' products and was demonstrated at Interop 90. Support for PPP is forthcoming in 3Com's NETBuilder system of local and wide area routers.

Conclusion

PPP is a data link protocol that can connect multivendor equipment in a nonproprietary fashion. It was designed to be flexible for routing and bridging functions and is extensible for future applications. It will be used primarily to connect internetworking equipment and to connect PCs via dial-up phone lines to computer centers.

PPP is an evolving standard. Many of its features continue to be defined in the IETF workgroup to meet additional communications requirements. Although PPP is new, many vendors have begun implementation of the latest standard in serial line networking. PPP enhances TCP/IP communications today and will enhance OSI communications in the future. ■

Dino Farinacci is a software engineer in 3Com's Network Systems Division. He has been designing and developing communications software for the past five years, specializing in transport and network layer protocols. He is the principal designer of 3Com's multiprotocol brouter, on which he continues development work. Mr. Farinacci is a member of the Internet Engineering Task Force (IETF), where he is active in several working groups related to routing and internet services.

References

1. *Point-to-Point Protocol for the Transmission of MultiProtocol Datagrams Over Point-to-Point Links*, Request for Comments 1171, July 1990.

2. *Point-to-Point Protocol (PPP) Initial Configuration Options*, Request for Comments 1172, July 1990.

The NetBIOS Applications Interface

By Mike Kouri and Bill Nolde

NetBIOS is a standardized software interface between application programs and a local area network (LAN). An application written to the NetBIOS interface will run unmodified on any network operating system that uses NetBIOS, regardless of the hardware or transport protocols used in the LAN.

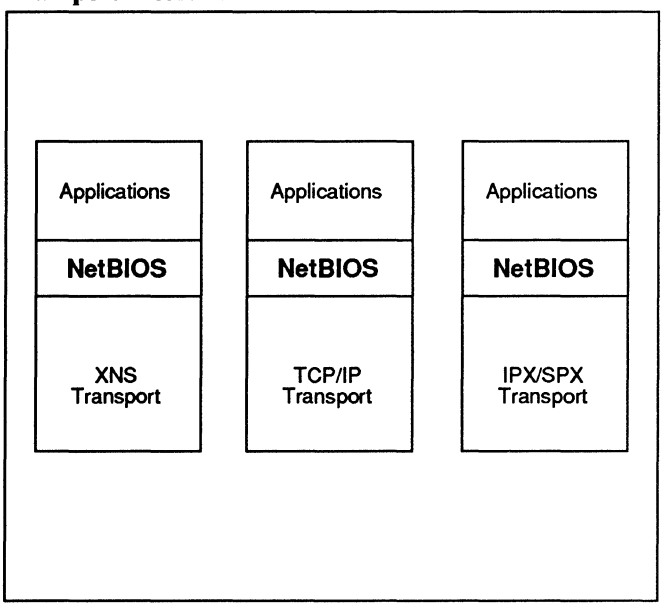
Many network operating systems must have a NetBIOS interface to communicate with their own underlying transport protocols. For example, 3Com's 3+ and 3+Open, Microsoft's MS-Net and LAN Manager, and IBM's PC LAN and LAN Server all require a NetBIOS interface. Other network operating systems, such as Novell's NetWare and Banyan's VINES, provide a NetBIOS interface but do not require it.

Because NetBIOS is used so extensively in modern LANs, both for applications and for internal communication within the network operating system, it is vital for network administrators and technical support personnel to understand how NetBIOS works when tuning or troubleshooting a network. This article describes NetBIOS commands in detail and includes examples of the data structures used by applications to talk to NetBIOS, shown in both the C programming language and Intel 8086 assembly language. A sample NetBIOS program called NBCHAT, used to allow NetBIOS workstations to communicate interactively with each other, is available on request.

NetBIOS Basics

NetBIOS (the network basic input/output system) is a programmatic interface between applications programs and the underlying network software and hardware. It was originally developed by Sytek Corporation as an interface to its intelligent broadband network interface. It gained popularity as a supported applications programming interface when IBM adopted it for its PC Network and PC LAN programs. Other vendors have implemented the NetBIOS interface for use with their own hardware and software to the point that it is a *de facto* standard used in PC local area networks. Using the NetBIOS interface isolates the application programs from the actual type of hardware used in the LAN and the transport protocols used to move data across that LAN, giving applications vendors a wider potential marketplace for their wares (see Figure 1).

Figure 1. NetBIOS Isolates Applications From Underlying Transport Protocols



NetBIOS Names

NetBIOS applications communicate with each other by using names. A name is an identifier for a logical entity, such as an application program or a process within an application. An application or a process may create one or more names to which it will respond. For each workstation, NetBIOS maintains a table of the names used by that computer. This table of local names has historically been fixed at sixteen entries long plus a special name called the permanent name. Names can be up to sixteen characters long. They should not contain an asterisk as the first (or only) character of the name.

The permanent name, also called the node number, is usually in ROM on the LAN adapter card. On some network adapters (but not 3Com's), the node number may be set by DIP switches on the LAN adapter card. The node number consists of ten characters of binary zeros followed by six more characters which must be unique on the network. Since all Ethernet adapters have unique addresses, the permanent name is usually the Ethernet address of the adapter, in byte-reversed order. This name is usually read by the transport protocol(s) and presented to the NetBIOS interface.

With Microsoft LAN Manager, the permanent name (burned into ROM) may be overridden by editing the PROTOCOL.INI file at that machine and adding a new statement to the appropriate adapter section that reads NETADDRESS= followed by twelve hexadecimal digits enclosed in double quotes (for example, NETADDRESS="02608C123456").

NetBIOS names are created locally by applications and processes within applications. NetBIOS names are used across the network to identify these logical entities. A table of locally defined names is maintained by the NetBIOS interface.

Most implementations of NetBIOS allow applications to add up to sixteen local names to the NetBIOS table. However, with the growing number of machines added to individual LANs, the limitation of sixteen names is becoming inadequate. Some recent implementations, such as 3Com's 3+Open XNS, NetBEUI, and NBP transport protocols, allow up to 254 local names to be defined.

Local NetBIOS names may be unique names or group names. A unique name is guaranteed by the NetBIOS to be unique across the LAN. A group name added at one computer may also be added, as a group name, at other computers.

Unique names are the more common of the two. Applications can be hard-coded for particular names. A common mistake when first beginning to program to the NetBIOS interface is to attempt to use group names to establish sessions to several stations. Only the first member of the group to respond will actually establish a session. Group names are useful both for sending datagrams, which will be described later, and when several machines provide the same service and the client doesn't care which of them provides the desired service.

NetBIOS Services

The NetBIOS interface consists of five basic services:

- General Control
- Name Support
- Datagram Support
- Session Control
- Session Data Transfer

Some vendors may provide proprietary extensions to the basic services. Table 1 provides a summary of the commands available in each service.

General Control Services: The NetBIOS General Control Services include resetting the NetBIOS interface, cancelling commands, and finding out the status of the network hardware adapter.

Name Support Services: The NetBIOS Name Support Services include adding unique and group names to the local name table and deleting local names from the table when they are no longer needed.

Datagram Support Services: The NetBIOS Datagram Support Services provide unreliable data transmission; meaning, there is no positive acknowledgment of data reception by the intended recipient's NetBIOS interface. Receipt of datagrams is not guaranteed. Datagram Support Services are also called "connectionless services." They are used to perform functions not possible

Table 1. Net BIOS Command Summary

COMMAND NAME	WAIT CODE	NO-WAIT CODE	COMMAND DESCRIPTION
General Control Services			
RESET	32	—	Reset NetBIOS
CANCEL	35	—	Cancel a pending command
ADAPTER STATUS	33	B3	Get status of a NetBIOS interface
UNLINK	70	—	Cancel boot redirection
Name Support Services			
ADD NAME	30	B0	Add unique name to name table
ADD GROUP NAME	36	B6	Add non-unique name to table
DELETE NAME	31	B1	Delete name from name table
Session Control Services			
CALL	10	90	Establish session with another
LISTEN	1	91	Wait for a CALL from another
HANG UP	12	92	Close session
SESSION STATUS	34	B4	Get status of sessions under name
Session Data Transfer Services			
SEND	14	94	Send session data
CHAIN SEND	17	97	Concatenate and send two buffers
RECEIVE	15	95	Receive session data
RECEIVE ANY	16	96	Receive data from any session under specified name
Datagram Control Services			
SEND DATAGRAM	20	A0	Send data, addressed by name
RECEIVE DATAGRAM	21	A1	Receive datagram to name
SEND BROADCAST	22	A2	Send data to all stations
RECEIVE BROADCAST	23	A3	Enable receive of next broadcast

Microsoft Redirector: The Most Popular NetBIOS Application?

The Microsoft Redirector is very possibly the most popular NetBIOS application in use today. It is used in network operating systems such as MS-Net and its derivatives (3Com's 3Plus and IBM's PC LAN) and Microsoft LAN Manager and its derivatives (like 3Com's 3+Open and IBM's LAN Server). The following describes some particularities of the Microsoft Redirector implementation.

SMB Names

The NetBIOS names used by the Microsoft Redirector are a subset of the allowable NetBIOS names.

When the Microsoft Redirector creates NetBIOS names, it uses the Server Message Block (SMB) naming convention that originated in MS-Net and the IBM PC-LAN programs. SMB names are up to 15 characters long, padded with spaces (hex 20) if necessary, and terminated with a single-character suffix that identifies the purpose of this name.

Even though the NetBIOS interface allows names to be up to 16 characters long, the SMB naming rules allow only 15 characters in a name. The 16th character is reserved as the suffix. Valid characters that can be used in a SMB name are the uppercase alphabetic characters, the numbers 0 through 9, and the following special characters: $%-_@{}~'!#(). Spaces are not allowed in SMB names except as pad characters.

The valid suffix characters for SMB names, in hexadecimal format, are:

00 Redirector Name. Also called a Machine Name. There are two major types; Unique and Group. A Unique Redirector Name is required for every LAN Manager machine.

03 Main or Additional Name. Also called User Names or Aliases. These names are used to send and receive messages. They are created by the Messenger service in LAN Manager.

05 Forwarded Name. A User Name or Alias forwarded from another machine, whose suffix has been changed to 05 hex. When users wish messages addressed to them to be received at another computer temporarily, they forward the User Name or Alias to another machine. The workstation software requests that the target computer add that name as a Forwarded name. The algorithm for sending messages across the network under LAN Manager is to first attempt to deliver this message to a Forwarded Name. If this fails because the target name has not been forwarded anywhere, then the sender directs the message to the User Name (ending with 03 hex).

20 Server Name. This is the name to which the Server service responds. ■

with the Session Control Services such as sending and receiving broadcasts addressed to all machines. Datagram Support Services also include sending and receiving short messages addressed to specific NetBIOS unique names or group names. Datagrams are limited to 512 bytes or less of application data.

Session Control Services: The NetBIOS Session Control Services allow applications to establish sessions, to check their status, and to tear them down. Sessions, also called "virtual circuits," are logical connections between two NetBIOS unique names established on a peer-to-peer basis. Data can only be transferred reliably between two nodes (in other words, guaranteed to arrive at one node in the order it was sent from the other) after a session has been established.

Once a session is established on a node, it is assigned a unique number to differentiate it from all other sessions on that node. The two NetBIOS names connected by a session are usually on separate nodes, but they can be on the same node, or they can even be the same name on the same node. Also, the same two names can be used to establish multiple concurrent sessions.

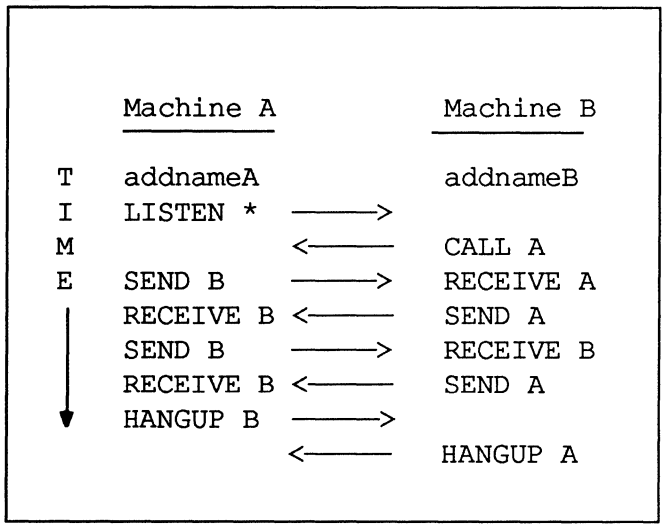
The NetBIOS interface maintains error statistics and current parameter values on a session-by-session basis.

Session Data Transfer Services: The Session Data Transfer Services include sending and receiving data on already established sessions. The maximum application message size that can be sent in one command is 64 KB. If an error occurs while sending data, the session will be closed to maintain data integrity. The NetBIOS interface handles the details of network error detection and breaking larger messages into smaller messages to compensate for any physical limitations of the network.

A session-level data transfer sequence can be grouped into four steps. Each step on a system (application, process, or computer in the network) must have a complementary step performed on another system as shown in Figure 2.

1. Add a unique name to the NetBIOS local name table using the ADD NAME command (performed at both systems.)
2. Initiate a session by using the CALL or LISTEN command. If using a CALL, then the other system should have already performed a LISTEN or the CALL will fail.
3. Transfer messages using the SEND and RECEIVE commands.
4. Terminate the session with the HANGUP command.

Figure 2. Data Transfer Sequence



Calling NetBIOS

NetBIOS commands are passed to the NetBIOS interface in the form of network control blocks (NCBs). The application program is responsible for allocating space for and creating these NCBs. An NCB contains several fields, including a command code field and a command result field. Some fields are used to pass input values to NetBIOS; other fields are used by NetBIOS to return results from the command execution.

After creating an NCB, DOS machines pass it to the NetBIOS interface by loading the address (in the form of segment:offset) of the NCB into the ES:BX registers and then executing a software interrupt 5C hex. The NetBIOS interface will perform the operation specified by the command code field in the NCB.

NetBIOS Command Modes

Three command modes are available to DOS machines when executing a NetBIOS command:

- Wait mode
- No-wait mode with completion post routine
- No-wait mode with polling

In wait mode, the NetBIOS interface will perform the command before returning to the calling program. The AL register will have the result of the operation.

The no-wait modes may queue the command for later execution and return program control to the application program immediately. The application can determine when the command has been completed by testing a flag in the NCB (i.e., polling) or by specifying the address of a POST subroutine to be executed when the command completes.

Under DOS, commands are issued to the NetBIOS interface by the following process:

1. Allocate and initialize an NCB.
2. Load the ES:BX registers with the address of the NCB.
3. Execute an INT 5C (hex) instruction.

The high-order bit of the COMMAND and POST fields of the NCB determine the resulting mode of operation. The COMMAND field is an 8-bit code that specifies the desired action. The POST field is either a double-word pointer to a user-supplied routine to be called when the command is completed, or it is set to 0:0, indicating that there is no user-supplied routine. The valid combinations of high-order bit in the COMMAND field and contents of the POST field are shown in Table 2.

The resulting modes of operation are as follows:

WAIT Mode: When a wait mode command is issued, the NetBIOS interface will not return control to the user program until the command has been completed. Upon return, the AL register will contain 0 if no error occurred, or it will contain an error code value that matches the one contained in the NCB RETCODE field. The contents of the AH register are destroyed but other registers are not changed.

There is a risk when posting a NetBIOS LISTEN command in wait mode; it could hang the process or the machine until another process or machine completes a corresponding NetBIOS CALL command. Wait-mode NetBIOS commands are sometimes referred to as synchronous NetBIOS commands.

NO-WAIT Polling Mode: When a no-wait polling mode command is issued, the NetBIOS interface will, after minimal processing, return immediately to the user program—even before that command may not have been completed. On return, the AL register will contain either an immediate error code or 00 hex. If the AL register is 00 hex, the command has been successfully queued for processing.

Table 2. COMMAND and POST Fields

High-order bit of the COMMAND field	Contents of the POST field	Resulting mode of operation
0	(Any)	Wait
1	0:0	No-wait polling
1	Address of POST routine	No-wait post

When the command is queued successfully, the user program periodically polls the CMD_DONE field of the NCB. This field will contain 0FFh until the command completes, at which point it will contain the return value. Only when the command has completed will all other NCB fields be valid. The AH register contents are destroyed but other registers are not changed.

NO-WAIT Post Mode: Like the no-wait polling mode commands, the NetBIOS interface returns immediately after a no-wait post mode command, even though the command may not have been completed. On return, the AL register will contain either an immediate error code or 0.

If the command could not be queued, the POST routine will not be called. The AH register contents are destroyed; other registers are not changed.

If the command was successfully queued, then, when the command completes, the NetBIOS interface will call the user's POST routine with interrupts disabled, the stack will be set for an IRET, and ES:BX will contain the address of the completed NCB.

If another NetBIOS command is issued from the POST routine, it must also be a no-wait mode command.

The user POST routine should be as short as possible and no registers should be changed. Interrupts may be enabled in the POST routine.

In some cases, the user's POST routine will be called before control is returned from the INT 5Ch that started the command. This is most likely to occur when there is a great difference in speed between the two communicating machines. The faster machine may respond more quickly than the slower machine expects it to.

Both no-wait polling mode and no-wait post mode NetBIOS commands are sometimes referred to as asynchronous NetBIOS commands.

Network Control Block Format

NetBIOS commands must conform to an NCB format. Tables 3a and 3b show the NCB diagram and NCB fields. NCBs are 64 bytes long and consist of 14 different fields, from 1 to 16 bytes each. Each NetBIOS command uses a certain subset of the NCB fields.

Table 3a. Network Control Block Diagram

00 - COMMAND (1byte)	01 - RETCODE (1 byte)
02 - LSN (1 byte)	03 - NUM (1 byte)
04 - BUFFER@ (4 bytes)	
08 - LENGTH (2 bytes)	
0A - CALLNAME (16 bytes)	
1A - NAME (16 bytes)	
2A - RTO (1 byte)	2B - STO (1 byte)
2C - POST@ (4 bytes)	
30 - LANA_NUM (1 byte)	31 - CMD_CPLT (1 byte)
32 - RESERVE (14 bytes)	

Table 3b. Network Control Block Fields

COMMANDS / NCB FIELDS	WAIT: / NO WAIT:	CALL	LISTEN	HANG UP	SEND	RECEIVE	RECEIVE ANY	CHAIN SEND	SEND DATAGRAM	RECEIVE DATAGRAM	SEND BROADCAST DATAGRAM	RECEIVE BROADCAST DATAGRAM	ADD NAME	DELETE NAME	RESET	ADAPTER STATUS	SESSION STATUS	CANCEL	ADD GROUP NAME
	WAIT:	10H	11H	12H	14H	15H	16H	17H	20H	21H	22H	23H	30H	31H	32H	33H	34H	35H	36H
	NO WAIT:	90H	91H	92H	94H	95H	96H	97H	A0H	A1H	A2H	A3H	B0H	B1H		B3H	B4H		B6H
NCB_COMMAND		>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>
NCB_RETCODE		<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<
NCB_LSN		<<	<<	>>	>>	<<	>>	>>							1>>				
NCB_NUM							<>		>>	>>	>>	>>	<<		2>>			<<	
NCB_BUFFER@					>>	>>	>>	>>	>>	>>	>>	>>				>>	>>	3>>	
NCB_LENGTH					>>	<>	<>	>>	>>	<>	>>	<>				<>	<>		
NCB_CALLNAME		>>	<>					4>>	>>	<<		<<				>>			
NCB_NAME		>>	>>										>>	>>			>>		>>
NCB_RTO		>>	>>																
NCB_STO		>>	>>																
NCB_POST@ (Note 5)		>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>	>>		>>	>>		>>
NCB_LANA_NUM (Note 6)																			
NCB_CMD-CPLT (Note 7)		<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<	<<		<<	<<		<<

>> sent
<< received
<> both

Notes:

1. Number of sessions to be supported.
2. Number of commands to be supported.
3. Address of NCB to be cancelled.
4. Contains length and address of the second buffer in a 2-buffer chain.
5. Valid only for no-wait comands.
6. NCB_LANA_NUM is always set to 00H.
7. Contains status/return code for no-wait commands when NCB_POST@ is set to 00H.

All fields in an NCB should be initialized to zero. The fields used in a command should then be set. In DOS, ES:BX is set to point to the NCB and an INT 5C can then be used to pass this NCB to the NetBIOS application's interface. The results of the command are communicated back through the NCB.

The five most important fields for diagnosing network illnesses are, in order, the COMMAND, RETCODE, CALLNAME, NAME, and NUM fields. The following is a complete listing of NetBIOS fields:

COMMAND:
A 1-byte field used for the NetBIOS command code. Set the high-order bit of this field to 1 to indicate no-wait mode or set it to 0 to indicate wait mode. The remaining 7 bits are used for the command code. All NetBIOS commands except RESET and CANCEL have both wait and no-wait options. In wait mode, the NetBIOS interface waits for the command to complete before returning to the calling program. In no-wait mode, the NetBIOS interface queues the command for later execution and then immediately returns to the calling program, which may perform other processing or issue additional NetBIOS commands. In no-wait mode, the POST field must also be set.

RETCODE:
A 1-byte field in which NetBIOS command results are returned. This field will contain an error code if an error was encountered or 0 if there was no error. *(Refer to Table 4a for a full list of Return Codes. In addition, Table 4b shows Return Codes versus Commands.)*

CALLNAME:
A 16-byte field used to indicate the name of the computer a user wants to communicate with for CALL, LISTEN, and datagrams. For a CHAIN SEND command, the CALLNAME field is used to specify the length and address of a second buffer. The first word specifies the length of the second buffer, and the next two words specify the address of that buffer. When used in the LISTEN command, usually only CALL commands issued by the name specified in this field will be answered. An asterisk (*) in this field indicates that any CALL made to the name in the next field (the NAME field) will be answered. The name making the CALL will be returned in place of the asterisk.

NAME:
A 16-byte field used to specify a local name to add to the local name table for ADD NAME or ADD GROUP NAME commands. This field is also used in CALL commands to identify the caller. To establish a session, the name being called (i.e, the name in the CALLNAME field) must have issued a LISTEN command, either for this specific caller or for any caller (*).

NUM:
A 1-byte field used for the name number associated with the name in the local name table. This number is returned by the ADD NAME command and must be supplied for RECEIVE ANY and datagram commands. The value of this number ranges from 1 to 255. The value of 1 is always assigned to the permanent node name. The value 255 is used with RECEIVE ANY and RECEIVE DATAGRAM functions instead of a specific name.

LSN:
A 1-byte field containing the local session number (with a value between 1 and 254) that is assigned by the NetBIOS interface when a session is established. This field is returned by CALL and LISTEN and must be supplied for SEND and RECEIVE commands.

BUFFER@:
A 4-byte field used for the address of data to be sent or received. When this field is used, it contains a pointer (segment:offset) to a buffer to be used by the issuing command. Only commands that send or receive data not contained in the NCB data structure use buffers. The exact contents of the buffer depend on the command and the application.

LENGTH:
A 2-byte field containing the length in bytes of BUFFER@. For receive commands, this field is set to the maximum buffer size, and the actual length of the data received is returned in this field upon completion of the NetBIOS command.

RTO:
A 1-byte field used for RECEIVE time-outs in half-second increments. This field must be set for CALL and LISTEN commands. It indicates the maximum time that will be allowed to pass before an error condition will result on a RECEIVE command. Once a session is established, this value remains constant throughout the session.

STO:
A 1-byte field used for SEND time-outs in half-second increments. This field must be set for CALL and LISTEN commands. If the time expires before the send has completed, a time-out error will result and the session will be terminated.

POST@:
A 4-byte field containing the address of a user interrupt routine to be called when a no-wait mode command completes. If all four bytes are set to 0, then no interrupt will occur and the calling program must assume responsibility for control by polling the CMD_CPLT field to determine the status of the command's execution.

LANA_NUM:
A 1-byte field that contains the number of the adapter card to be addressed by the command. If two adapter cards are installed on the same computer, a 0 in this field signifies that the command is addressing the first card and a 1 indicates that the command is addressing the second card. If there is only one adapter card in the computer, this field should contain a 0.

CMD_CPLT:
A 1-byte field that is set by NetBIOS when a command is completed. A value of 0FFH indicates that the command has not completed. When the command has completed, this field is set to the same value as RET-CODE.

RESERVED:
A 14-byte field, used internally by NetBIOS, which should not be used for any other purpose. Different implementations of NetBIOS use this field differently, so applications should not make any assumptions about its contents.

Figure 3. Assembly Language NCB Structure

```
NCB struct
    ncb_command          db ?    ;command code
    ncb_retcode          db ?    ;err ret code
    ncb_lsn              db ?    ;session number
    ncb_num              db ?    ;name number
    ncb_buffer@          dd ?    ;ptr to send/recv data
    ncb_length           dw ?    ;length of data buffer
    ncb_callname db 16 dup (?)   ;remote name
    ncb_name     db 16 dup (?)   ;local name
    ncb_rto              db ?    ;recv timeout
    ncb_sto              db ?    ;send timeout
    ncb_post             dd ?    ;async cmd complete post addr
    ncb_lana_num         db ?    ;adapter number
    ncb_cmd_cplt         db ?    ;0xFF until command completed
    ncb_reserve  db 14 dup (?)   ;reserved for use by NetBIOS
NCB ends
```

Figure 4. C Language NCB Structure

```
struct ncb {
  byte command;         /* hex code for command to be executed  */
  byte retcode;         /* return code                  */
  byte lsn;             /* local session number      */
  byte num;             /* number returned by the locator       */
  char far *buffer;          /* address of data buffer          */
  int length;           /* number of bytes to be sent or received */
  char callname[16];    /* name of the node to communicate with */
  char name[16];        /* name of your node          */
  byte rto;             /* receive timeout                  */
  byte sto;             /* send timeout               */
  char far *post_routine;    /* pointer to post routine        */
  byte lana_num;        /* local adapter number      */
  byte cmd_cplt;        /* command complete           */
  byte reserve[14];     /* NetBIOS temporary variable storage   */
```

Table 4a. List of NetBIOS Return Codes

Code	Description
00	No error.
01	Illegal buffer length. A SEND BROADCAST or SEND DATAGRAM command specified a length greater than 512 bytes, or a status command specified a buffer length smaller than the minimum allowed.
03	Invalid command.
05	Time-out. For SEND, RECEIVE, and HANG UP commands, the timeout specified when the session was established has elapsed. On a CALL or ADAPTER STATUS command, an internal timer expired.
06	Message incomplete. The buffer size specified in the NCB was not large enough to hold the RECEIVE data. For RECEIVE or RECEIVE ANY commands, the next command will receive the rest of the data. For other commands, the remaining data will be lost.
07	NO-ACK command failed. One or more SEND NO-ACK and/or CHAINSEND NO-ACK commands failed. The session is still active. Resynchronize the data flow (if possible) and continue, or terminate the session and start over.
08	Invalid local session number (LSN).
09	Out of resources. The NetBIOS interface is out of some internal resource, such as buffers. Wait and reissue the command later.
0A	Session closed. For a SEND, RECEIVE, RECEIVE ANY, or HANG UP command, this indicates that the session was terminated by the remote computer.
0B	Command cancelled. Command execution of the NCB was aborted by the CANCEL command.
0D	Duplicate local name. An ADD NAME command specified an existing name.
0E	Name table full.
0F	DELETE NAME completed, but the name has active sessions. The name will be deleted when all sessions have been closed. No new sessions will be allowed with the name.
11	Local session table full.
12	Remote computer not listening. On a CALL, the remote computer was found, but had no outstanding LISTEN for the CALL.
13	Invalid name number.
14	Name not found.
15	Either name not found or an asterisk (*) or 00H was in the first byte of the remote name field on a CALL.
16	Name already exists on network. The specified name is already in ues as a unique name on another adapter.
17	Name was deleted.
18	Session terminated abnormally. Connection with the remote computer was lost.
19	Name conflict. Two computers were detected using the same name.
1A	Incompatible remote device.
21	Interface busy. The NetBIOS cannot execute because it was called from an interrupt handler, or because it is out of local resources.
22	Too many commands issued. The number of commands outstanding equals the maximum number allowed.
23	Invalid LAN adapter (LANA) number.
24	Command completed before it was cancelled. This code is returned in CANCEL NCB when the target command completed normally.
25	Reserved name specified. An ADD NAME or ADD GROUP NAME command specified a reserved name. Use a different name.
26	Invalid cancel command. The target NCB could not be found.
30	Name defined by another process (specific to IBM OS/2 EE). The command referred to a locally defined NetBIOS name. Resources reserved for a given process within a workstation can only be used by that process and the specified local NetBIOS name is already reserved for another process. Use another name or remove the process that is using the required name.
34	NetBIOS environment not defined (specific to IBM OS/2 EE). The RESET command must be the first command issued by a process. Issue the RESET command.

(Cont'd...)

Table 4a. List of NetBIOS Return Codes *(cont'd.)*

35	Required operating system resources exhausted (specific to IBM OS/2 EE). The NetBIOS interface cannot initiate the requested command because OS/2 EE resources are not available to support the command. Retry the command later.
36	Maximum applications exceeded (specific to IBM OS/2 EE). This return code only applies to the RESET command. NetBIOS services requested in the RESET command are not available to this requesting process because the number of processes that the NetBIOS interface is currently serving is the maximum allowed by the NetBIOS load time parameters. Stop a process that is using NetBIOS services, or increase the value of the NetBIOS Application (APP) load time parameter and reboot.
37	No SAPs available for NetBIOS (specific to IBM OS/2 EE). All allocated SAPs are already in use and none are left for NetBIOS. NetBIOS requires only one SAP to support all of its processes.
38	Requested resources not available (specific to IBM OS/2 EE). The requests for NetBIOS resources (names, commands, sessions, or the use of name number 01) exceed the number specified at NetBIOS load time. NetBIOS is still available for the application, but with fewer resources than requested. The use of name number 01 can only be claimed by one process.
40	System error.
41	Hot carrier from remote adapter detected (PC network). Remove the offending adapter from the network and cycle power on your own machine before attempting to use the network again.
42	Hot carrier from this adapter detected (PC network). You have a hardware malfunction. Replace your network adapter.
43	No carrier detected (PC network).
4E	Token ring status bits 12, 14, or 15 on longer than one minute.
4F	Token ring status bits 8 through 11 set to "on."
50-F6	Adapter malfunction or unknown error code.
F7	Error initializing adapter.
F8	Error opening adapter.
F9	IBM LAN Support Program internal error.
FA	Adapter malfunction.
FB	IBM LAN Support Program not loaded.
FC	Error opening adapter or DLC.Open.SAP failed.
FD	Adapter closed unexpectedly.
F7-FE	Adapter malfunction or unknown error code.
FF	Indicates the command has not completed.

Table 4b. Return Codes versus Commands

RET CODE	NetBIOS COMMANDS / RETURN CODE MEANING	WAIT: / NO WAIT:	CALL 10H / 90H	LISTEN 11H / 91H	HANG UP 12H / 92H	SEND 14H / 94H	RECEIVE 15H / 95H	RECEIVE ANY 16H / 96H	CHAIN SEND 17H / 97H	SEND DATAGRAM 20H / A0H	RECEIVE DATAGRAM 21H / A1H	SEND BROADCAST DATAGRAM 22H / A2H	RECEIVE BROADCAST DATAGRAM 23H / A3H	ADD NAME 30H / B0H	DELETE NAME 31H / B1H	RESET 32H	ADAPTER STATUS 33H / B3H	SESSION STATUS 34H / B4	CANCEL 35H	ADD GROUP NAME 36H / B6H
00H	Command accepted/completed		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
01H	Illegal buffer length									■		■					■	■		
03H	Invalid command code		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
05H	Command timed-out		■		■	■	■		■								■			
06H	Message incomplete						■	■			■		■				■	■		
08H	Illegal local session number				■	■	■		■											
09H	No resourch available		■	■																
0AH	Session closed				■	■	■	■	■											
0BH	Command cancelled		■	■	■	■	■	■	■		■		■				■			
0DH	Duplicate in local name table													■						■
0EH	Name table full													■						■
0FH	Name de-registered, but active														■					
11H	Local session table full		■	■																
12H	Session open rejected		■																	
13H	Illegal name number							■		■	■	■	■							
14H	Cannot find name/no answer		■																	
15H	Name not found, no* or 00h		■	■										■	■			■		■
16H	Name in use on remote adapter													■						■
17H	Name deleted			■				■			■		■							
18H	Session ended abnormally		■	■	■	■	■	■	■											
19H	Name conflict detected		■	■				■		■	■	■	■	■			■	■		
1AH	Incompatible remote device		■	■																
21H	Interface busy		■	■	■	■	■	■	■	■	■	■	■	■	■		■	■		■
22H	Too many commands outstanding		■	■	■	■	■	■	■	■	■	■	■	■	■		■	■		■
23H	Invalid number in NCB_LANA_NUM		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
24H	Command was not cancelled																		■	
25H	Reserved name specified																			■
26H	Command not valid to concel																		■	
40H	Locator not responding		■							■				■	■		■			
4XH	Unusual network condition		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
50H-FEH	Adapter malfunction		■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
FFH	Command pending status*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*		*

* Value found in NCB_CMD_CPLT field - valid only for no-wait options

NetBIOS Under OS/2

So far, we have focused exclusively on NetBIOS from the perspective of DOS. Now let's look at the differences in the NetBIOS interface between DOS and OS/2, and the differences between IBM's implementation of NetBIOS with IBM LAN Server and LAN Requestor in IBM OS/2 Extended Edition, and Microsoft's implementation of NetBIOS with OS/2 LAN Manager. 3Com's 3+Open OS/2 LAN Manager follows the rules for the Microsoft implementation.

In all instances, NetBIOS functions are available via OS/2 dynamic linking. Processes present requests to the NetBIOS interface using FAR CALL instructions rather than INT 5C or 2A interrupts. Also, instead of pointing ES:BX to an NCB and executing an INT 5C, OS/2 assembly language programs use the following sequence:

```
push NCB_Selector  ; "segment" to NCB
push NCB_Offset    ; "offset" to NCB
call NetBIOS_Submit; NetBIOS dynamic link
```

The NCB structure for OS/2 is no different than for real-mode NetBIOS (NetBIOS for DOS). OS/2 uses the NetBIOS interface through application programming interfaces (APIs). OS/2 NetBIOS is different from DOS NetBIOS in that, with asynchronous NCBs, the POST field contains a semaphore handle, not the address of a POST routine.

IBM OS/2 EE and LAN Server

The RESET Command: In IBM OS/2 EE, each process operates independently. A process obtains NetBIOS resources by a RESET command, which must be the first NetBIOS command that process issues. Processes cannot share names, including the permanent node name. The right to use the permanent node name is obtained by issuing the RESET command.

Wait and No-Wait Modes: When a command that specifies the wait mode is initiated, the requesting process thread is immediately blocked. When the command completes, execution returns to the requesting process's code. The effect is similar to the DOS commands that specify no-wait operation.

OS/2 EE handles no-wait commands by spawning child threads that are immediately blocked. Execution of the requesting process's thread continues without pause. If a POST routine was specified in the NCB, the NetBIOS interface will invoke it when the command completes. POST routines return by executing a FAR RETURN instruction rather than an IRET instruction.

Because commands specifying no-wait mode operation require more OS/2 EE resources than commands specifying wait operation, no-wait commands may fail (with a return code of 35H) where wait commands would succeed.

New Return Codes:
30H: Name defined by another process
34H: NetBIOS environment not defined
35H: Required operating system resources exhausted
36H: Maximum applications exceeded
37H: No SAPs available for NetBIOS
38H: Requested resources not available

Microsoft OS/2 LAN Manager

Microsoft OS/2 LAN Manager, unlike IBM LAN Server and DOS implementations of NetBIOS, allows multiple NetBIOS interfaces to be installed.

OS/2 uses the NetBIOS interface through APIs. OS/2 NetBIOS is different from DOS NetBIOS in that, with asynchronous NCBs, the POST field contains a semaphore handle, not the address of a POST routine.

The NCB structure itself has not changed from real-mode NetBIOS. Under real mode (in the DOS compatibility box), all NCBs (except no-wait mode POST commands) are supported. The application must instead poll for command completion. NCBs in the real mode are submitted using the same INT 5C as DOS, not by using the LAN Manager APIs.

Reserved Handle 0: An application should use the NetBiosOpen and NetBiosClose calls to obtain and release handles to the NetBIOS interface(s). These handles are intended to be used with NetBiosSubmit. An application can use the reserved handle 0 when submitting NCBs. Such NCBs will be submitted to the

first NetBIOS interface, and implicit calls to NetBiosOpen and NetBiosClose will be performed. Since real mode applications are restricted to using INT 5C to communicate with NetBIOS, they are restricted to the first NetBIOS interface.

Because more than one process may open the same NetBIOS interface, OS/2 LAN Manager allows the opener to specify how it is willing to share access to the NetBIOS interface with other processes. Three access modes are defined to accomplish this, as shown in Table 5.

Table 5. NetBIOS Access Modes

Because more than one process may open the same NetBIOS interface,OS/2 LAN Manager allows the opener to specify how it is willing toshare access to the NetBIOS interface with other processes. Three access modes are defined:

Access Mode	Permissions	Restrictions
NB_REGULAR	Any number of processes may open a NetBIOS interface in regular mode.	Does not allow RESET, RECEIVE BROADCAST DATAGRAM, RECEIVE ANY-TO-ANY NCBs, or the use of permanent names in any NCB.
NB_PRIVILEGED:	One process may open the interface in privileged mode. This mode is compatible with NB_REGULAR; other processes may open the interface in regular mode, even while it is opened by a single process in privileged mode. A privileged open will fail if any other process has a current privileged or exclusive open handle to that interface.	Does not allow RESET or RECEIVE ANY-TO-ANY NCBs.
NB_EXCLUSIVE:	One and only one process may open the interface, if the process opens it in exclusive mode. The open attempt will fail if any other process has an open handle to that interface.	All NCB operations are allowed.

NetBIOS APIs for OS/2 LAN Manager

Five NetBIOS APIs are documented for OS/2 LAN Manager to allow direct access to the NetBIOS interface(s): NetBiosEnum, NetBiosGetInfo, NetBiosOpen, NetBiosClose, and NetBiosSubmit. As mentioned earlier, use can be restricted to just NetBiosSubmit using handle 0. Figures 5 through 9 show these five NetBIOS APIs.

Figure 5. NetBiosEnum (Admin Only). Purpose: enumerates NetBIOS interfaces

```
unsigned far pascal
NetBiosEnum(srvrname, level, buf, bufLen, entRead, totalEnts)
char far *  srvrname;          /* name of target PC (null if local) */
short       level;             /* level of info requested           */
char far *  buf;               /* pointer to info buffer            */
unsigned short bufLen;         /* length of info buffer in bytes    */
unsigned short far * entRead;  /* # of entries returned             */
unsigned short far * totalEnts;/* total # of entries available      */
```

If level is set to 0 when this call is made, the information buffer returned will be filled with "struct netbios_info_0" containing only the names of the NetBIOS interfaces loaded. These names are defined in the [NETWORKS] section of the LANMAN.INI file, which is read by the portion of LAN Manager installed at boot time.

If level is set to 1 when this call is made, the following structures are returned in the information buffer:

```
struct netbios_info_1 {
      char           net_name[NETBIOS_NAME_LEN+1];
      char           driver_name[DEVLEN+1]; /* OS/2 device drive name    */
      unsigned char  lana_num;              /* LAN adapter of this net   */
      char           pad_1;
      unsigned short driver_type;
      unsigned short net_status;
      unsigned long  net_bandwidth;         /* Network bandwidth, bits/s */
      unsigned short max_sess;              /* Max. # of sessions        */
      unsigned short max_ncbs;              /* Max. # of outstanding NCBs*/
      unsigned short max_names;             /* Max. # of names           */
 }
```

Net_Status is a bitmapped field and is defined below:

```
  Bit     Definition

   0      Net managed by LAN Manager
   1      Driver is a loopback driver
  2-13    <Reserved>
 14-15    Open status.  The values of these two bits mean:
             0   net not opened
             1   opened in regular mode
             2   opened in privileged mode
             3   opened in exclusive mode
```

The open status bits of the net_status field will be set to 2 if any process has that NetBIOS interface opened in privileged mode, even though other processes may simultaneously have it open in regular mode.

Figure 6. NetBiosGetInfo (Admin Only). Purpose: gets information about a given NetBIOS Interface

```
unsigned far pascal
NetBiosGetInfo(srvrname, netBiosName, level, buf, bufLen)
char far *       srvrname;    /* name of target PC (null=local)    */
char far *       netBiosName; /* NetBIOS network name              */
short      level;             /* level of info requested           */
char far *       buf;         /* pointer to info buffer            */
unsigned short bufLen;        /* length of info buffer in bytes    */
```

Figure 7. NetBiosOpen. Purpose: gets a handle to a NetBIOS Interface

```
unsigned far pascal
NetBiosOpen(netBiosName, netReserved, netOpenOpt, netHandle)
char far *           netBiosName;    /* name of network            */
char far *           netReserved;    /* MUST BE 0                  */
unsigned short       netOpenOpt;     /* open options               */
unsigned short far * netHandle;      /* word for returned handle   */
```

The NetOpenOpt field is bitmapped as follows:

Bit	Definition
0-1	Access Mode
	0 <Reserved>
	1 NB_REGULAR
	2 NB_PRIVILEGED
	3 NB_EXCLUSIVE
2-15	<Reserved>

Note: Handles returned by NetBiosOpen are process-to-interface associations. Only the process that opened the handle may use it.

Figure 8. NetBiosClose. Purpose: closes a NetBIOS interface handle

```
unsigned far pascal
NetBiosClose (netHandle, netReserved)
unsigned short   netHandle;          /* handle to close            */
unsigned short   netReserved;        /* MUST BE 0                  */
```

Figure 9. NetBiosSubmit. Purpose: passes one or more NCBs to the NetBIOS interface

```
unsigned far pascal
NetBiosSubmit(netHandle, NetNCBOpt, netNCB)
unsigned short   netHandle;          /* handle to issue NCB against */
unsigned short   netNCBOpt;          /* option flags                */
struct ncb far * netNCB;             /* address of NCB              */
```

NetHandle is either a handle returned from a previous call to NetBiosOpen, or 0. A handle of 0 always refers to the first installed NetBIOS interface. This interface will automatically be opened by NetBiosOpen (in regular access mode) the first time a NetBIOS call refers to it using the 0 handle.

NetNCB points to the NCB to be executed (unchained NCB) or to the link word preceding the NCB (chained NCB; see below). The caller need not fill in the ncb_lana_num field; this will be done based on the netHandle used.

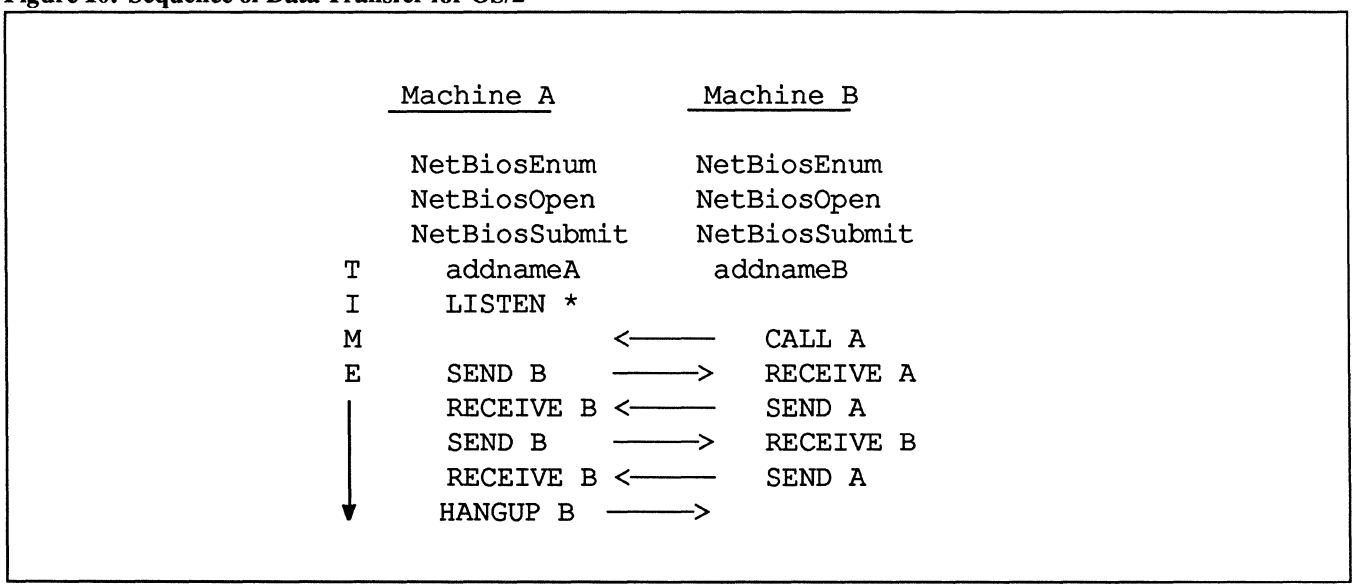
If the NCB is a no-wait NCB, the ncb_post field of the NCB should either be 0, or a system semaphore handle. If a handle is given, the semaphore will be cleared on completion of the NCB. The system semaphore must be public; that is, created without the "exclusive" option.

netNCBOpt specifies NCB processing options. Allowable values are as follows:

0 single NCB being passed
1 single NCB with error retry
2 NCB chain with proceed-on-error
3 NCB chain with stop-on-error

Sequence of Data Transfer: Unlike real-mode NetBIOS (DOS NetBIOS), OS/2 LAN Manager allows multiple NetBIOS interfaces. Therefore, a session layer data transfer sequence using the OS/2 NetBIOS interface would still have the same four steps as DOS, but those steps would be enveloped by OS/2-specific steps to determine the correct NetBIOS interface and bind to it. Each step on a system (application, process, or computer in the network) must have a complementary step performed on another system (see Figure 10).

Figure 10. Sequence of Data Transfer for OS/2



1. List NetBIOS interfaces by using NetBiosEnum (performed at both systems).

2. Get a handle to the chosen NetBIOS interface with NetBiosOpen (performed at both systems).

3. Using the handle acquired in step 2, issue a sequence of NetBIOS commands using NetBiosSubmit (performed at both systems):

- Add a unique name to the NetBIOS local name table using the ADD NAME command (performed at both systems.)
- Initiate a session by using the CALL or LISTEN command. If using a CALL, then the other system should have already performed a LISTEN or the CALL will fail.
- Transfer messages using the SEND and RECEIVE commands.
- Terminate the session with the HANGUP command.

4. Close the NetBIOS interface handle with NetBiosClose.

Conclusion

Microsoft has been quick to recommend that any other means of LAN Manager interprocess communication, such as Named Pipes, be used in place of NetBIOS. However NetBIOS is available on all levels of LAN Manager workstations (OS/2, DOS Enhanced, and DOS Basic) where Named Pipes are only available at a penalty—support for them consumes valuable applications workspace. Applications written to the NetBIOS interface are transportable across a variety of network operating systems platforms including 3Com's 3+ and 3+Open, Microsoft's LAN Manager, Novell's NetWare, and Banyan's VINES. Named pipes applications are restricted to those network operating systems that support named pipes. ■

Mike Kouri has been working in 3Com's Technical Services Operation for about three years ago, specializing in Ethernet transmission system products. When 3Com began testing the initial release of 3+Open LAN Manager, he was transferred to that team and has been supporting 3+Open LAN Manager ever since. His main focus has been on the core LAN Manager product and 3Com's value-added transport protocols. Mr. Kouri denies any memory of 3+, although he says he can still spell EtherSeries on a good day.

Bill Nolde is a member of the engineering group within 3Com's Technical Services Operation. He is in charge of third-party software development support and all 3Com APSs. Mr. Nolde has an MA in mathematics and is a member of both the IEEE and ACM. He has been in the computing industry for 13 years and is the author of several PC software products.

Mr. Nolde and Mr. Kouri would like to acknowledge the invaluable assistance provided by all of 3Com technical support in reviewing this article, especially Cathy Anderson and Frank Burke.

Suggested Reading:

3Com Network Architectures, Services, and Protocols, 3Com Corporation, part number 7310-00, revision A.

NetBIOS Programmer's Reference, 3Com Corporation, part number 3260-00.

Microsoft LAN Manager 1.0 API Summary, Microsoft Corporation

The Microsoft LAN Manager, a Programmer's Guide, Ryan, Ralph, Microsoft Press ISBN 1-55615-166-7.

Sample NetBIOS Program Available:

A listing for a sample NetBIOS program called NBCHAT, used to allow NetBIOS workstations to communicate interactively with each other, is available on request. Write to 3TECH Journal, 3Com, P.O. Box 58145, Santa Clara, CA 95052-8145. Please mark envelope: "ATTN: NBCHAT".

TCP and OSI Network Addressing

By Michael A. Smith

When you mail a letter, it must have the correct address in order to reach its intended destination. The postal service doesn't generally concern itself with the contents of the letter, which can be in any language—its concern is that the envelope is properly addressed to enable it to route the letter to any location around the world. The same is true for network addressing—the network address enables the correct routing of data across a network, regardless of the kind of data being transferred.

Network addressing identifies nodes and facilitates routing—the end-to-end service of data between two nodes. This article describes the principles of TCP and OSI network addressing. It is aimed at network managers, LAN administrators, and datacom people who are interested in building and supporting multivendor enterprise networks.

Overview of Network Addressing

Networks are transmission systems designed to deliver information. Because most of today's networks are heterogeneous in topology and behavior, the routing of data over the network is becoming more complex. It is now more critical than ever that networks efficiently provide routing for delivery of data and audit trail service for internetwork traffic. Ideally, networks provide these services in spite of differences in network interfaces, routing techniques, access control, addressing schemes, maximum packet size, time-outs, quality of service, types of error recovery schemes, status reporting, and connection services (e.g., datagram versus virtual circuit).

These requirements apply to the four basic networking situations:

- Point-to-point links between homogeneous topologies.
- Point-to-point links between heterogeneous topologies.
- Network links (e.g. PDN) between topologies.
- Network to non-network (e.g. host) links.

To facilitate routing, network addresses must have certain characteristics—global uniqueness, a uniform scheme for dynamic handling, and the ability to allow optional uses of network addressing.

Global Uniqueness. The first premise of addressing is uniqueness. Every point on the network must be uniquely identified. This means that the address space must be at least as large as the number of nodes on the network. Also, the address of a node should not require changing if it physically moves or the topology changes. However, this may conflict with another rule of addressing which says that the lower-layer service point (e.g., the MAC address) should be derived easily from the network address. This association can expedite end-routing—for example, extracting the MAC address from inside the network address and insert it into the MAC header.

Uniform scheme, dynamic handling. Addresses must be consistent enough in format to allow dynamic handling. This means that, even though the address is not fixed in size for other characteristics (e.g., ASCII, binary), it must be of a uniform design to allow the network to route. In theory, this rule could allow an infinite number

of schema. In practice, there are two limitations: the address scheme cannot be too simple and the address scheme cannot be too complex.

If the address is too simple (e.g., a random number), it may not facilitate intelligent routing. This would mean that the source must supply the routing information, the intermediate nodes must have unique information about every needed destination on the network, or that the data goes to many places other than its destination.

If the addressing scheme is too complex, the routing service may not be able to figure out how to forward it quickly enough (this is one reason that MAC-level bridges outperform routers).

Optional uses of network addressing. Network addressing allows for more than routing of data. By providing a unique identity to every point on the network, the entire system can be controlled directly. For example, network addressing can facilitate administration (accounting, service, and support) and provide security.

Network Addressing Schemes

TCP/IP Network Addresses

The IP address is a single, fixed-length value that represents a host within a network. Networks are logically delineated by network numbers and physically partitioned by network routers, known as internet gateways[1]. The network number and the host address share the address space. See the "TCP/IP Address Syntax" section for the syntax of IP addressing.

IP networks tie heterogeneous topologies together into a catenet[2]. IP addressing facilitates the routing of data over this catenet, masking the differences of the underlying topologies. The two-part address facilitates this routing in a simple way. But, its simplicity is also its weakness. By defining only one profile, there is no provision for different topologies. Treating all networks (Ethernet, token ring, X.25, ISDN, etc.) in the same way ignores the characteristic advantages of each, making other network functions, such as administration, more difficult.

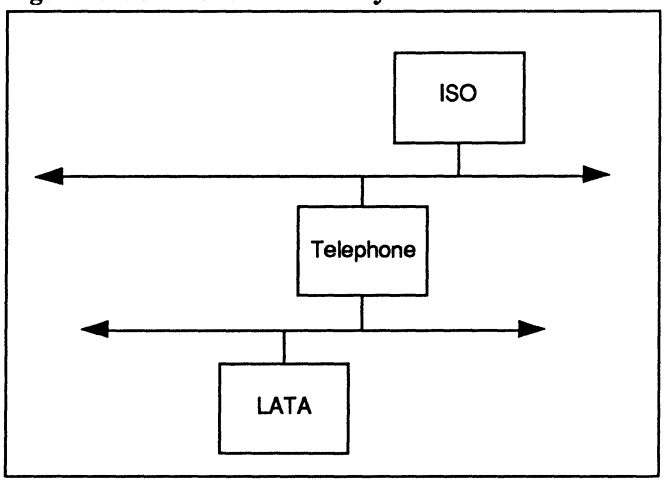
OSI Network Addresses

OSI network addressing, as defined in ISO 8348/Add2, is complex compared to IP addressing. Instead of hosts, as in IP, the basic unit of OSI addressing is a network service access point (NSAP). An NSAP[3] is a logical communication element.

The OSI network address can be of varying length and has many parts. Unlike IP, which was initially designed for wide area networking between large hosts, ISO network addressing was designed for almost every kind of data networking and therefore has to deal with many options.

OSI addressing was designed to be truly global, which means that it encompasses existing networking schemes (including addressing schemes). The OSI approach is to create a hierarchy of "authorities" and make each of these many networks sub*networks*[4]. A mapping scheme allows every subnetwork address to be a part of an OSI network address. However, the hierarchy is not merely two levels. The highest authority is the International Standards Organization (ISO)[5] which can create up to ninety authorities, such as the phone system or X.25. Authorities at this level can create thousands of subauthorities (e.g., phone companies). These authorities, in turn, can create their own subauthorities. An example of a sub-subauthority is "Local Access and Transport Area" (LATA) which identifies the 161 geographical subdivisions used to define local telephone service within the U.S. This concept of global OSI hierarchy is shown in Figure 1.

Figure 1. OSI Network Hierarchy



Under the OSI scheme, authorities assign ranges within a portion of the network address and can also assign lower portions of those addresses. This syntax is defined in the "OSI Addressing Syntax" section. The important concept is the hierarchy—the further down the hierarchy, the fewer bits have to be authorized (used). Currently, there seven authorities are recognized by ISO:

Local. Local implies an OSI network unattached to other "public" OSI networks—for example, a test lab.

X.121. This authority is for the addressing scheme used by the international X.25 community.

ISO DCC. Known as "ISO geographic," this consists of subnetworks corresponding to countries or "an appropriately sponsored organization" in a "non-ISO" country.

F.69. This authority is for telex (CCITT Recommendation F.69).

E.163. Another common authority is public switched telephone networks (CCITT Recommendation E.163).

E.164. This authority is for ISDN.

ISO 6523-ICD. ICD stands for "international code designator," which is a four-digit code (ISO 6523).

OSI Address Format

The OSI network address begins with an **authority and format identifier** (AFI). The AFI describes the network address authority responsible for allocating the values of the initial domain and describing the syntax of the rest of the address.

Within an authority such as X.25, a particular address domain must be specified (e.g., for X.25 and X.121 addresses). This is done through the **initial domain identifier** (IDI). With enough bits almost any "network address" can be inserted into the OSI IDI. This includes X.121 addresses, telephone numbers (including city, area, and country codes), telex numbers, even IP addresses (though presently there is no authority assigned to do IP). And it doesn't stop when the subnetwork "address" (e.g., phone number) is inserted into the OSI address.

Once the AFI is identified and the subnetwork number is inserted into the IDI, there is still room for the **domain specific part** (DSP). The DSP identifies the NSAP to the final subnetwork point of attachment (SNPA)—the point at which the network forwards the data to the underlying network for delivery to the network node. OSI addresses are often referred to as NSAP addresses and represent the intermachine address (as opposed to the address of a process running in the machine).

Some NSAPs have null DSPs—X.121 is an example of this. The X.121 address in the IDI provides enough addressing information to identify the node on the network. Other authorities have different address formats and may require additional address data in the DSP—for example, the subnetwork address field specified by NIST (the U.S. government's National Institute for Standards and Technology) under the ISO 6523-ICD authority. This field turns out to be just right for an 802.3 MAC address (i.e., an Ethernet address).

Domains versus Subnets

In TCP/IP, subnets are a convention used to better administer large IP network numbers. In OSI, domains are the abstract convention used to create a hierarchy of authorities for the purpose of administering the huge OSI address space.

In OSI, subnetworks are considered the real underlying network technologies. X.25, for example, is considered a subnet, as is Ethernet (802.3). Subnet is a term used in ISO 8348/Add.2 to identify a "real" end system.

OSI addressing domains are areas of a network associated by the administration of an addressing authority hierarchy. One example might be a departmental LAN, delineated more by "department" than by "LAN." The addressing authority ensures that all addresses within the domain are unique and conforming to whatever address range is allowed. This domain could also cover multiple subnetworks (e.g., an Ethernet and a token ring).

OSI domains (like IP subnets) are abstract; subnetworks are real (like cable plants).

Network Address Syntax

Now let's look at the syntax for both TCP/IP and OSI network addresses. The formats for these two types of network address are distinctly different. Table 1 compares differences in their address length, range, and format.

Table 1. A Comparison of TCP/IP and OSI Network Addresses

	Address Length	Address Range	Address Format
TCP/IP	32 bits	~2^{30} nodes	network.host
OSI	16-160 bits	~2^{152} nodes	AFI/IDI/DSP

TCP/IP Address Syntax

The IP internet address is 32 bits, divided into four groups. It is typically represented by twelve numbers separated into triads by decimal points—for example, 221.008.100.012, known as "dotted decimal" notation. Each triad represents one octet and, therefore, has a range of values between 0 and 255. 0 and 255 are exception cases[6]. An IP address such as 345.987.555.256 would not be valid because each triad's value is greater than 28-1.

The value of the first triad defines the class of the internet address. This class determines which triads are treated as the network portion and which triads are treated as the host portion of the address (see Table 2).

As shown in Table 2, 001.001.001.001 through 127.254.254.254 are Class A addresses, 128.001.001.001 through 191.254.254.254 are Class B addresses, and 192.001.001.001 through 223.254.254.254 are Class C addresses. For example, 009.001.001.001 through 009.254.254.254 mean that all the nodes are on a Class A network with a network number of 009; addresses 203.100.231.001 through 203.100.231.254 mean that all the nodes are on a Class C network with a network number of 203.100.231. In addition, IP multicasts use the 254.000.000.000 through 255.255.255.255 range. IP multicast traffic is typically contained within each subnet.

The first three bits in the IP address indicate a large network—Class A with up to 16,516,350 (255^2*254) "host" nodes, a medium network—Class B with up to 64,770 (255*254) "host" nodes; or a small network—Class C with 254 "host" nodes. It also means there are 127 possible Class A networks, 16,002 (63*254) possible Class B networks, and 1,999,996 (31*254^2) possible Class C networks. In total, that is about two billion unique addresses, or slightly less than half the 2^{32} possible with four octets.

Two billion might sound like a lot of unique addresses, until the Network Information Center[7] assigns you a Class C address and you realize you have 10,000 nodes on your network. This means you have 254 host addresses for 10,000 addressable nodes to cover.

An obvious limitation to IP addressing is the potentially small number of addresses available. The problem is compounded by the fact that every addressable network port on every node must have a separate IP address. In

Table 2. IP Address Classes

Address Class	Decimal Range	HEX Range	Binary Representation	IP Address Format (n=network, h=host)
A	0 to 127	00-7F	00000000-011111111	nnn.hhh.hhh.hhh
B	128 to 191	80-8B	10000000-101111111	nnn.nnn.hhh.hhh
C	192 to 223	C0-DF	11000000-110111111	nnn.nnn.nnn.hhh

many cases, Class C addresses are assigned to those who have several "small" segments and have internet gateways (such as 3Com BR/3000s) linking them together. The reason for this is the limited number of Class A and Class B addresses. The Network Information Center asks "why give out a Class B network address when a few Class C addresses might do?" As mentioned earlier, subnets are used to better administer large IP network numbers.

Subnet masking is a convention for partitioning the host portion of the IP address into subnetwork and host parts. Subnet masking does not extend the IP address. Conceptually, subnet masking converts a Class B address into several Class C addresses; or a Class A address into several Class B addresses. Instead of being assigned a few Class C addresses, it's possible to get a Class B address and do your own subnet masking. This gives users the latitude to have more than 254 host addresses on a segment and still have their internet gateways do their job.

To specify a subnet mask, the IP address is either nnn.hhh.hhh.hhh, nnn.nnn.hhh.hhh, or nnn.nnn.nnn.hhh. When an internet gateway gets a packet with one of these addresses to forward, it masks out the Internet (nnn) portion of the address. Depending on the address class, the mask varies from 8 to 24 bits. With subnet masking, the first part of the host (hhh) portion of the address is, in effect, made into a part of the Internet (subnetwork) address.

This works for a **Class A internet address** in the following way:

In this example, a large company has been assigned one Class A address. There are sixteen large LANs spread over seven cities, tied together by internet gateways on leased (e.g., T1, 56 Kbps) phone lines. Each LAN has thousands of nodes. Table 3 shows the assigned mask, subnet mask, and possible subnet addresses.

A plain IP address is represented by nnn.hhh.hhh.hhh; a subnet IP is represented by nnn.ssh.hhh.hhh. In this case, "ss" represents the subnet .

Why are the first and last addresses crossed out? Subnet values 000 and 240 have the same functional significance as network and host addresses of 000 and 255. This means that a subnet value of 240 would indicate "all subnets" and 000 would indicate "this subnet." Admittedly, using decimal notation for the octets obscures the meaning. This is shown translated to binary form in Table 4.

Table 3.

The following mask is assigned: 255.240.000.000
This creates a subnet mask of: (11111111) (11110000) (00000000) (00000000)
And the following possible subnetwork addresses:

~~078.000.000.000~~	078.016.000.000	078.032.000.000	078.048.000.000
078.064.000.000	078.080.000.000	078.096.000.000	078.112.000.000
078.128.000.000	078.144.000.000	078.160.000.000	078.176.000.000
078.192.000.000	078.208.000.000	078.224.000.000	~~078.240.000.000~~

Table 4.

078.000.000.000	=	01001110.00000000.00000000.00000000
(11111111) (11110000)	=	01001110.**11110**000.00000000.00000000
nnn.ssh.hhh.hhh	=	nnnnnnnn.**ssss**hhhh.hhhhhhhh.hhhhhhhh

Any IP address having a value whose binary interpretation overlays the masked bits will be interpreted as being on that subnet. Four masked bits means 2^4 combinations. When this is set up on the networks, the high four bits of the host address will be masked off and that will designate which subnetwork the user is on. Address 078.071.012.134 would be on subnet 078.064.000.000, as shown in Table 5.

The above is done by ANDing the IP address to the mask and digging out the subnet number. Remember, nothing happens to the IP address, the identity of the source or destination, or connection service. It simply gives the routers more of a network number to work with for more specific routing—this facilitates more efficient service by reducing the amount of superfluous traffic.

An example of a **Class B internet address** follows:

This example is a college campus that has been assigned one Class B address. The campus has eight LANs in three buildings tied together by routers on leased (T1) phone and PBX lines. Each LAN has hundreds of nodes. Table 6 shows the assigned mask, subnet mask, and possible subnet addresses.

The router would only know that this is a Class B address and that the subnet mask was (11100000). Once again, the first and last addresses are crossed out because they are special cases. To the rest of the world, 128.042.097.012 specifies a network node on internet address 128.042. Subnet masking has no effect on internet routing tables.

Table 5.

078.071.012.134	=	01001110.01001011.00001100.10000110
(11111111)(11110000)	=	11111111.11110000.00000000.00000000
078.064.000.000	=	01001110.01000000.00000000.00000000

Table 6.

The following mask is assigned: 255.255.224.000
This creates a subnet mask of: (11111111)(11111111)(11100000)(00000000)
And the following possible subnetwork addresses:

~~128.042.000.000~~	128.042.032.000	128.042.064.000	128.042.096.000
128.042.128.000	128.042.160.000	128.042.192.000	~~128.042.224.000~~

OSI Address Syntax

As mentioned before, the OSI network addressing scheme is complicated in comparison with TCP/IP—the OSI network address can be of varying length and has many parts. It has a hierarchical structure, with various authorities involved in assigning parts of the address. The OSI network address format is shown in Figure 2.

The OSI network address can be up to 20 octets[8] long. The initial domain part (IDP) can be up to 20 octets or the DSP can be up to 19 octets, but together they cannot total more than 20 octets. The IDP is required, as is the AFI. The IDI or the DSP may be null. This is not unlike the IP network/host scheme—the larger the IDP, the smaller the DSP and vice versa. The main difference in OSI is 128 bits of extra address space.

The format for the AFI is two digits (effectively one octet), an integer between the values of 0 to 99. Possible values for the AFI are:

0-9	Reserved, will not be allocated
10-35	Reserved for future allocation by joint agreement of ISO and CCITT
36-59	Allocated for assignment to the IDI formats (see below)
60-69	Allocated for assignment to new IDI formats by ISO
70-79	Allocated for assignment to new IDI formats by CCITT
80-99	Reserved for future allocations by joint agreement of ISO and CCITT

For current AFI assignments, the IDI can be up to 15 octets. The AFI determines the format of the IDI. For example, if the AFI called out 36 (X.121), the IDI would be an X.121 address.

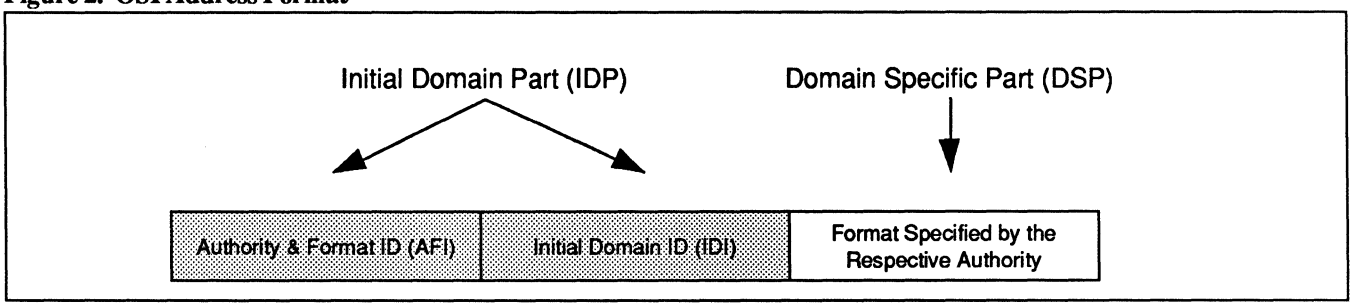
As the name implies, the DSP is specific to the domain. For example, X.121 authority makes the DSP null (the X.25 address is in the IDI). It would not be uncommon to find an Ethernet or token ring MAC address inside this area, or maybe even a phone number. One domain defined by ISO (instead of adapted from a non-ISO network) is international code designator (AFI 47). Under their AFI, the U.S. government received an assignment with an IDI = 0005, used in the Govenrment OSI Profile (GOSIP).

The GOSIP DSP has a four-level hierarchy, as authorized by the National Institute for Standards and Technology:

1. Organization identifier (OI)
2. Subnetwork identifier (SI)
3. Subnetwork address (SA)
4. NSAP selector (NS)

For example:
1. 3Com - 0035
2. Marketing - 0908
3. Michael Smith's PC (on Ethernet) - 026086721997
4. Specified by the authority - 1

Figure 2. OSI Address Format



Authority Syntax

The following is a description of the maximum IDI and DSP lengths for each of the seven current OSI address authorities.

Local: Local provides a huge IDI/DSP to play with that even allows ISO 646 characters (e.g., NetBIOS names).

X.121: The IDI for this is up to 14 digits. The DSP is up to 24 digits. This addressing is a widely used data communication standard in Europe.

ISO DCC: This IDI is 3 digits with a DSP of up to 35 digits.

F.69: The IDI is up to 8 digits, beginning with a 2-3 digit destination code. The DSP is up to 30 digits.

E.163: The IDI can be up to 12 digits, and the DSP can be up to 26 digits.

E.164: This has an IDI of up to 15 digits, starting with the ISDN country code. The DSP is up to 23 digits.

ISO 6523-ICD: The International Code Designator is a 4 digit IDI code; the maximum DSP is up to 34 digits.

Conclusion

TCP/IP and OSI have distinctly different approaches to network addressing—in fact, they can be viewed as two extremes. Despite their extremes, however, they both fulfill the basic requirements of network addressing.

IP will continue to be the prevalent standard, mostly because of its wide acceptance and enormous installed base. However, OSI addressing offers distinct advantages that should one day enable it to dominate the installed base. ■

Michael Smith is product marketing manager for workgroup protocols in 3Com's Network Systems Division. Previously, he served as a OEM services manager for Bridge Communications. Mr. Smith has over ten years of experience in computing, including systems analyst positions at both NEC and Shell Oil Company.

Mr. Smith has an associate degree in Computer Science from De Anza College, and a Bachelor's degree in Business/Journalism from Indiana University.

References

1. In the TCP/IP world, devices that interconnect two networks and route data between them are known as internet gateways.

2. Catenet: a collection of packet switching networks that are connected by gateways.

3. An NSAP is one of many service access points (SAP). An SAP can be at any OSI layer and represents a relationship between a layer and the underlying layers as an addressable point of service for a higher layer. This information is carried in the protocol header.

4. Subnetworks in OSI are more of a physical concept than an addressing scheme such as in IP. Subnetworks are referenced in ISO 8648 and 8348 as "point of attachment" sets of addresses. For example, an 802.3 segment of a network might be considered a subnetwork because all ISO addresses there could map to 802.3 MAC addresses. In fact, these MAC addresses might even be encoded in the domain specific part of the ISO network address (an example of this follows later).

5. The International Standards Organization, ISO, is an international organization (formed by the United Nations) formed for the development of a wide variety of standards. The ISO subcommittee that determines networking standards is known as Technical Committee 97 (TC97).

6. In certain contexts, certain fixed addresses have specific functional significance. For example, the address 0 is to be interpreted as meaning "this", as in "this network". The address of all 1s means "all", as in "all hosts". For example, the address "144.016.255.255" can be interpreted as all the hosts on network 144.016. Or, the address "000.000.000.037" can be interpreted as meaning host 37 on "this network".

7. The Network Information Center (NIC) is an entity of the Stanford Research Institute (SRI) International, Menlo Park, California, and is the authority responsible for assigning network numbers.

8. 8348/AD2 calls out "20 binary octets or 40 digits." But since a digit can be four bits, either format will fit into 160 bits.

Suggested Reading:

Internetwork Routing With TCP/IP, Michael Smith, *CONNECTIONS—Bridge Technical Newsletter*, Vol. 13, 1987.

Internetworking With TCP/IP, Douglas Commer, Prentice Hall, 1988.

The Open Book—A Practical Perspective on OSI, Marshall Rose, Prentice Hall, 1990.

"The OSI Network Layer Addressing Scheme, Its Implications, and Considerations for the Future", Christine Hemrick, U.S. Department. of Commerce, National Telecommunications and Information Administration Report, 85-186, 1985.

Information Processing Systems—Data Communications, Network Service Definition, ISO 8348/Add.2, International Standards Organization, 1987.

Telecommunications and Information Exchange Between Systems, ISO/TC 97/SC 6, DP 9542, International Standards Organization, 1986.

Information Processing Systems—Data Communications, Protocol for Providing Connectionless-mode Network Service, ISO 8743, International Standards Organization/Joint Technical Committee, 1988.

Heterogeneous LAN Management: A Joint 3Com/IBM Architectural Proposal

By Richard Watson and Paul Sherer

The heterogeneous LAN management (HLM) architecture is a set of architectural documents that have resulted from a joint study carried out by 3Com and IBM to define a management platform that will satisfy the management needs of mixed Ethernet and token ring environments regardless of the operating system supporting each LAN node. Addressing today's diverse network management requirements is made more complex by the fact that most major LAN installations have a mix of media types and operating system environments. Heterogeneous configurations pose special challenges to providing a uniform management architecture, but it is vital that these challenges be addressed.

This article should help ISV developers, NDIS developers, and LAN system administrators understand current management issues in heterogeneous installations and provide an overview of HLM.

HLM Overview: The need for Heterogeneous LAN Management

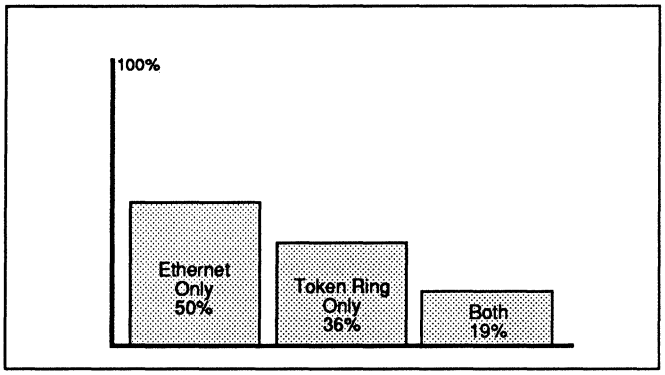
Networks are increasingly essential to the day-to-day activity of corporate enterprises. A network manager is typically responsible for the workgroup (including network cabling, personal computer workstations, and basic network services), internetwork devices (including bridges, brouters, and gateways) and all other network-connected data processing equipment. Network management is the array of tools and techniques that enable organizations to ensure the continued proper operation of a network.

Many of these networks are either Ethernet or token ring, but a growing number consist of a combination of the two. The common factor between them is the need to manage the whole network and reduce LAN failures and downtime.

Heterogeneous LAN installations are not a consideration for the future but exist in today's Fortune 1000 corporations. According to a recent industry report [1], the number of concurrent installations of both Ethernet and token ring technologies is closely related to the type of vertical market industry and company size.

Of the more than 12,000 sites responding to the survey, 19 percent reported having both Ethernet and token ring installed (50 percent reported having only Ethernet and 36 percent reported having only token ring, see Figure 1). No one LAN media has a clear domination in the market. It appears that, for the foreseeable future, mixed (or heterogeneous) LANs will be a fact for larger businesses.

Figure 1. Installed Base of Ethernet and Token Ring LANs



In addressing the broad spectrum of management issues, the international standards bodies have focused on the definition of communications protocols. These protocols provide standard ways to send management information between the nodes on a network. Each of the major communication standards has a provision for network management. Management on TCP/IP is supported by the simple network management Protocol (SNMP), define by the Internet Engineering Task Force. Management on the Open Systems Interconnect's (OSI) common management information protocol (CMIP). While adherence to one of these protocols means that a vendor/customer uses a standard way to communicate management information, it does not ensure multivendor interoperability. In addition, these standard management protocols are not appropriate for management of such memory-constrained devices as networked DOS-based personal computers which are prevalent in today's homogeneous and mixed networks.

What Does HLM Address?

Network management is one of the most important issues and most misunderstood concepts in the local area network business. There is a great deal of confusion about exactly what network management services are required and who will supply those services. Most major vendors (IBM, DEC, 3Com, Hewlett-Packard, AT&T, etc.) are creating offerings in their respective domains of business, but these solutions tend to be vendor-specific and moreover, LAN media-specific. Solutions offered by one vendor's product do not meet today's diverse LAN management needs.

While many management products are available today, none provide the total solution to the network administrator who has to manage a network consisting of mixed physical media and mixed operating systems (e.g., DOS, OS/2, UNIX). Multiple products are often needed to administer each component of management issues in such configurations.

In order to provide the proper management solution for mixed media LAN installations, a unified solution must be available. A unified solution is one in which the network administrator has to interface with only one management system to achieve the necessary control, regardless of the media or operating system environment. To achieve such a solution requires industry-wide cooperation.

3Com and IBM's Approach to a Solution

3Com and IBM are the leading vendors for their respective LAN media products (Ethernet and token ring). Both companies share a common problem: customers require network manageability across mixed media. This common problem brought the two companies together to propose an architectural solution. On July 9, 1990, 3Com and IBM announced a combined effort to create the HLM architecture.

The HLM architecture is designed to be independent of network operating systems and to provide management of devices on mixed-media LANs, particularly those in which system memory is constrained. HLM consists of three basic components: a network management protocol common to Ethernet, token ring, and other network configurations; application programming interfaces (APIs) through which developers can create compatible network management software; and specifications of what management data is accessible.

HLM Architecture

Since both 3Com and IBM are manufacturers of popular network communication products, an important aspect of the HLM architecture is management support of the lower level set of network services. This "bottom's up" view of LANs provides a very practical set of management services that is applicable to any media type and any operating system environment.

Understanding the four major design goals of HLM is key to understanding the basic architecture.

HLM Design Goals:

To manage all types of workstations (DOS, OS/2, or UNIX) in addition to concentrators, hubs, bridges, and servers.

One of the initial design goals was to define an architecture that had the flexibility to provide a set of minimum management services to permit simple extensions for

management of more complex environments. Specifically, providing a set of meaningful management services to memory constrained DOS systems was one of the basic considerations. Any successful network management product must provide management access to DOS workstations while imposing the minimum impact on the memory available for application execution. This means developing a clear definition of what constitutes the minimum set of managed services. Since the physical and logical link layers are vital to any LAN-attached device, HLM management support for OSI layers 1 and 2 (physical and data link) are defined as common for all HLM-compliant systems.

Provide the architecture to manage Ethernet and token ring today and others tomorrow.

Defining a management architecture for Ethernet and token ring is the current requirement, but LAN technology continues to evolve and HLM must be flexible enough to accommodate new media types (e.g., FDDI). As expressed in the HLM design, the interoperability between Ethernet and token ring is dependent on adherence to established industry standards. Such a design will enable support for other types of media in the future.

Provide minimal management services that can be supported by any network node with extensions to accommodate vendor and user extended management entities.

While DOS memory constraints may place a practical limit on the minimum management services defined, other operating systems (e.g., OS/2, UNIX) do not share the same constraints. In order to extend HLM's effectiveness, a mechanism was necessary to permit user and/or vendor extensions that would interoperate with the base HLM products. In this way, management of system or application level entities could be provided.

Be a standards-based architecture.

Perhaps the most important goal in designing HLM was to follow current and proposed management standards. Such a design more clearly assures market acceptance and future system interoperabiliity than creating another proprietary management architecture.

HLM Internal Design

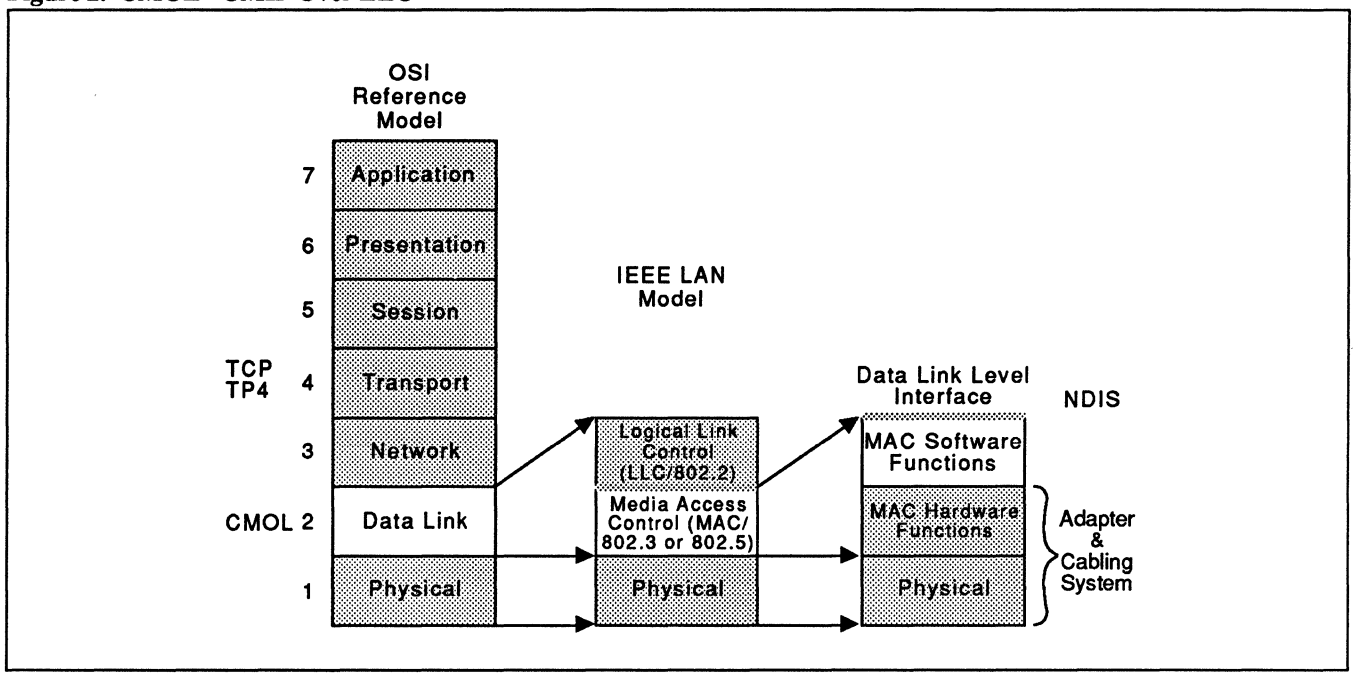
The result of the joint agreement between 3Com and IBM is not a product but, rather, a set of three architectural documents. The first document, *HLM Architecture Specification* [2] provides the architectural framework for the system in describing the protocol data units, protocol elements of procedure, and management information base (MIB) definitions. The second document, *HLM API Technical Reference* [3], describes the necessary functional subcomponents that support the HLM environment. The third document, *HLM LLC Specification* [4], details the logical link control (802.2) functions necessary to fit in the HLM-NDIS (Network Driver Interface Specification) framework. All these documents are available to the public from either 3Com or IBM and are not proprietary to either company.

Base Management Protocol

Several major HLM architectural components were developed and described in the documents. Key to meeting one of the initial goals of the architectural design was the concept of providing DOS workstation management in a standards framework. This was accomplished by specification of the OSI CMIP standard utilizing a lower level network service LLC (802.2). Figure 2 illustrates how this works when the full TCP and OSI stacks occupy considerable memory space and provide networking functions through the transport layer (TCP at layer 4) and the application layer (OSI at all seven layers) of the OSI model. Providing the OSI-CMIP services over a "short" stack significantly reduces the system overhead and provides a platform that is implementable in DOS memory space. The term used to describe this architecture is CMIP over LLC (CMOL).

The LLC and media access control (MAC) sublayers of the IEEE model correspond to the data link layer (layer 2) of the OSI model. The LLC sublayer is an IEEE standard, heavily implemented and enhanced by IBM and other vendors, which defines how command packets are generated and interpreted for support of the logical link functions of one or more logical links.

Figure 2. CMOL - CMIP Over LLC



While CMOL is a logical choice for the memory constrained system environment, it was not the only option available. The two major management protocols SNMP and CMIP may be implemented in other environments than the original protocol. Serious efforts have been made to describe and implement these management protocols using various transport services. Figure 3 describes the position of CMOL in relation to some other popular options. The CMIP protocol was selected because of its design flexibility and its expected interoperability with future systems. Within the CMIP architecture, the basic services are identified as common management information services (CMIS). A CMOL architecture will ensure a mechanism for providing CMIS services across a very large OSI management domain.

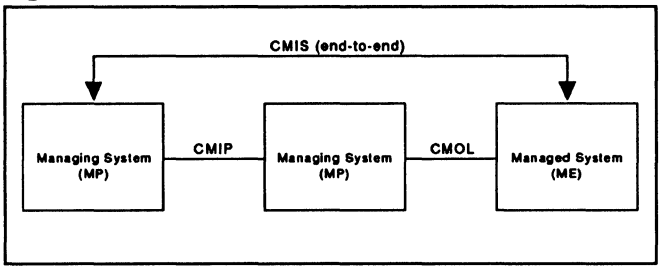
Figure 3. CMOL Compared to CMIP

	SNMP	CMIP
TCP	Standard	CMOT
OSI	RFC 1161	Standard
LLC	Proposed	CMOL

CMOL incorporates CMIP on top of the widely implemented LLC protocol to ensure compatibility with existing mixed-media network installations. Since a large portion of the CMIS interface is implemented at layer 2 rather than layer 4, CMOL will not require a full OSI stack. This leaves additional memory for running other applications on such devices as DOS-based personal computers and internetworking equipment.

Specification of CMOL also fits with the basic OSI approach of having full CMIP services between hierarchical management stations and allowing workstations to be managed by some subset of CMIP (see Figure 4). In this architecture, consistent CMIS services are provided across the entire network while meeting the specific implementation requirements imposed by memory-constrained systems.

Figure 4. End-to-End CMIS Services Under HLM



It is anticipated that HLM's network management protocol, CMOL, will be implemented in a variety of operating system environments. For DOS and OS/2 environments, portions of the initial implementation will be based on proposed extensions to the 3Com-Microsoft NDIS. (see "NDIS Concepts" by Rex Allers in this issue of *3TECH*).

HLM Functions

LAN Station Manager

The basic HLM services are provided by the LAN station manager (LANSM). The functions provided by LANSM will be common to all HLM stations, whether they are managing or being managed. It is this module that will support such base functions as:

1. Discovery/registration
2. CMOL processing
3. Base MIB services
4. Extended API support

In discovery/registration, each participating network station may determine the managing environment of the LAN and proceed to register itself with the appropriate manager. Mechanisms have been described to accommodate the asynchronous nature of network devices and there is no order of initiation required for either managing or managed stations.

All the basic CMIP/LLC and remote operations services element (ROSE) processing will be performed by the LANSM and are responsible for interfacing with the HLM transport system to communicate with other HLM entities. No subelements of the HLM architecture—e.g., layer management entities (LME)—will be required to process and handle the CMOL transactions directly.

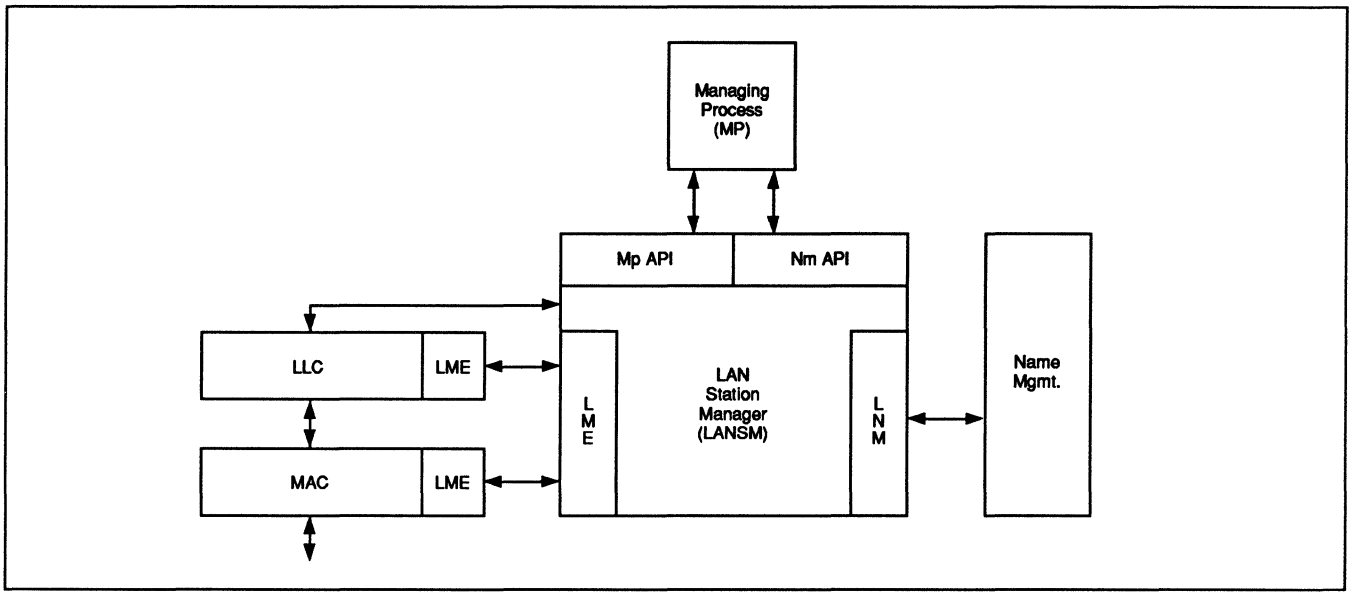
The LANSM also provides the minimum set of management services by providing management access to the MAC, LLC, and "environment" MIB. An MIB is a defined set of manageable variables that relate to the associated managed entity. For example, each system sublayer (MAC and LLC) has its own MIB definition that is accessible through its associated LME. The LANSM will interface with the appropriate LME to respond to HLM management commands directed at the specific sublayer. The MIB information for the MAC and LLC is supplied by the appropriate software component while the "environment" MIB is supported directly by the LANSM. The environment MIB consists of information describing the station and its current system configuration (e.g., DOS I.D., amount of memory, location I.D.).

Support for the other HLM functions is provided by the LANSM in exposing a set of API services. These services provide a programmatic link into the HLM management system that supports generation of managing process (MP) and vendor-extended managed entities (ME).

Managing Process

An MP provides the user interface control for managing services as permitted in the CMIP framework. While HLM does not describe the user interface functionality, it does describe the base control functionality supported through the API services provided by the LANSM. Using these services, *as described in the HLM API Technical Reference,* an MP may identify and control all HLM stations that may be present on the LAN. The scope of the MP function may be set by the developer of the management application, in that a manager may be focused on one layer or may encompass control over all supported management layers. In addition, the architecture does allow for multiple managers to coexist. Figure 5 shows the basic relationship between the MP and the LANSM.

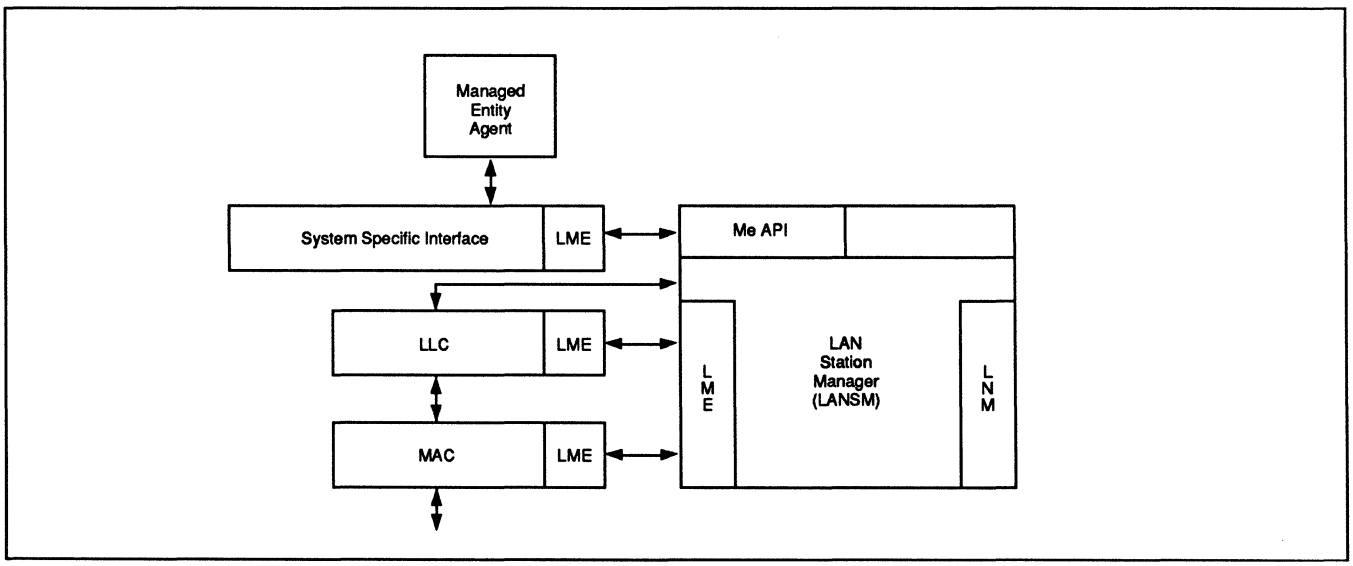
Figure 5. Relationship Between the Managing Process and LAN Station Manager



Managed Entity

An ME is any managed MIB or functional domain that is registered with an HLM manager. Several internal or inferred MEs are architected into HLM: station's "environment" LME, LLC LME, and MAC LME. These MEs provide the minimum management service base as described in the HLM documents. Extended management services may be generated by using the ME APIs provided through the LANSM. These API services allow the creation of extended managed objects (MIBs) that may be vendor or application specific. In systems that permit the added memory requirements, other system-level MEs or even applications MEs may be developed that execute simultaneously on a single workstation. Figure 6 shows the basic relationship between the ME and the LANSM.

Figure 6. Relationship Between the Managed Entity and LAN Station Manager



LLC Name Management

In order for the HLM entities on the network to identify each other, HLM specifies a name management architecture. This name management service provides a low-level (LLC) name-mapping service and is not intended to provide generalized naming services. The *naming* services are available as one of the defined API interfaces and are typically used by a managing process for identifying and tracking "named" entities within the scope of HLM management.

Ethernet and Token Ring MIB Specification

To properly manage the lower level entities, the MIB must be defined. This information base consists of definitions of all the management elements for each of the layers. HLM contains definitions of elements of Ethernet MAC, LLC, token ring layers 1 and 2, and elements of the "system environment" objects. These are expressed in the syntax defined by the standards bodies.

NDIS/HLM Interface Proposal

While HLM is architected to be implemented with almost any type of MAC-level services, the NDIS example is described in detail. The NDIS MAC-level services are very important to the products now offered by 3Com and IBM since 3Com's 3+Open LAN Manager and IBM's LAN Server are architected around the NDIS services for both Ethernet and token ring. The HLM architecture has been designed to integrate smoothly into existing NDIS environments with minimal installation or software upgrade problems. All current NDIS drivers will provide the necessary minimal support required by HLM to operate. HLM modules will be provided by 3Com that bind to NDIS through the standard protocol manager mechanisms. The only functional disparity possible is the loss of management of the MAC-level MIB if the specific NDIS driver does not support statistics gathering (see Appendixes C and D of the NDIS 2.01 specification document). Future releases of 3Com NDIS drivers will all fully support the NDIS statistics services.

HLM's Progress in the Standards Arena

In keeping with the original joint study goal, the HLM specifications have been made available for industry review and comment. Acceptance at the broadest level will be key to the architecture's success and encouragment of industry participation will only strengthen HLM's chances of acceptance as the management architecture of choice. In addition, both 3Com and IBM have announced support of HLM for their respective product lines.

In November 1990, a very significant development in the acceptance of HLM occurred. The IEEE 802.1B committee reviewed the specification and elected to adopt portions of the HLM architecture as the new 802.1B LAN/MAN management standard proposal. The HLM approach to solving the desktop management problem was eagerly received and it is anticipated that the IEEE 802.1B will become an accepted standard by the end of 1991. 3Com and IBM will also approach other standards bodies with the intent of gaining acceptance of all or part of the HLM specification in other standards.

Conclusions

The need to provide a unified management service platform for heterogeneous LAN installations exists today. Without such an architecture, the problem of management in mixed LAN environments results in a severe limitation to the growth of LAN-based applications and their integration into businesses. HLM provides a sound architecture targeted to supply a minimum set of management services that will support the memory-constrained systems and that will also expand to accommodate the more sophisticated management requirements of larger systems.

Acceptance of a common management service must be based on services common to the broadest array of vendor products, therefore cooperation among vendors is imperative. Since the creation of HLM is a cooperative effort subject to review by the industry—and has the ultimate goal of incorporating all or part of HLM into accepted national and international communications standards— its acceptance will be more assured as a standard for desktop management. ■

Richard Watson is manager of enabling software at 3Com's Network Adapter Division, and has over fifteen years of experience in managing development efforts in LANs and telecommuncation.

Paul Sherer is director of architecture in 3Com's Network Adapter Division.

References

1. *International Data Corporation Bulletin*, May 1990

2. *Heterogeneous LAN Mangement Architecture Reference.* IBM, Document Number SC30-3539-0 (Nov. 90).

3. *Heterogeneous LAN Management Application Program Interface Specification.* IBM, Document Number SC30-3540-0.

4. *Heterogeneous LAN Management LLC Specification.* IBM, (SC number unavailable).

Questions On HLM?

IBM and 3Com's joint release of the heterogeneous LAN management (HLM) specifications has prompted many questions from potential developers and others who are now reviewing the document. The following is a list of most-asked questions about HLM.

What is the next step for HLM, following the industry review?
Following the release of the HLM documents, 3Com and IBM will coordinate the comments and suggestions from the reviewers. Pending the applicability of the suggestions, they will be integrated into the specification before the final publication. The industry review should take about two months to complete. One of the underlying goals of the 3Com/IBM work is to create an architecture that would have the broadest industry support, so the input from the reviewers is important.

What about acceptance by standards bodies?
In keeping with the original joint study goal, the HLM specifications have been made available for industry review and comment. Acceptance at the broadest level will be key to any management architecture's success and encouraging industry participation will only strengthen HLM's chances of acceptance as the management architecture of choice. In addition, both 3Com and IBM have announced support of HLM for their respective product lines.

Besides soliciting the support of the industry, the IEEE 802.1B committee reviewed the specification (IEEE Meeting, Nov. '90) and elected to adopted portions of the HLM architecture as the new 802.1B LAN/MAN management standard proposal. The HLM approach to solving the desktop management problem with an OSI position was eagerly received and it is anticipated that the IEEE 802.1B will become an accepted standard by the end of 1991. 3Com and IBM will also approach other standards bodies with the intent of gaining acceptance of all or part of the HLM specification in other standards.

What plans exist for development of HLM-compliant products?
Both 3Com and IBM have made public announcements about their intention to produce products that are HLM compliant. However, no timeframes have been disclosed.

The first HLM-compliant product from 3Com will come as a future release of 3Com's network management station product. HLM will become the sole desktop management protocol, replacing its predecessor, AMP (adapter management protocol).

Both 3Com and IBM will develop and offer the base LAN Station Manager (LANSM) components. It is hoped that other vendors will see product opportunities and develop their management products and provide additional managed entities (ME) modules. Interoperability will be assured by adherence to the released HLM specification.

Will NDIS developers be required to modify their drivers?
There are no mandated changes to existing NDIS-compliant drivers in order to support the HLM environment. For NDIS drivers generated to specifications prior to V2.01, certain HLM services will not be supported. Specifically, older drivers did not keep MAC statistics and will be able to support a limited set of managed MAC elements through "generic" LME modules to be supplied by 3COM. All developers are strongly encouraged to upgrade their products to the most recent release of the specification.

How far along is industry acceptance of HLM?
Prior to the release of the HLM specification, several major "early" partners were included in the review of the HLM architecture. These reviewers included Microsoft, Novell, The Santa Cruz Operation, and Banyan. These organizations represent the major vendors in the network operating system industry. Other strategic product vendors will also be targeted for review. ■

From The Ask 3Com Bulletin Board...

The following articles have been reprinted from ***Ask3Com****, 3Com's electronic information service (available on CompuServe®.) Ask3Com features technical articles, product and service information, patches, fixes, utilities, and a lively message section. Each quarter, a tally is taken of these articles, indicating which are the most popular. Here's our latest "Greatest Hits" list...*

Error in 1.6 Upgrade Documentation

Reference Number: 02880034
Date posted: 7/13/90

There are some key errors that may mislead users in the 3+Backup Version 1.5.2 User Release Notes which shipped with the 1.6 Upgrade (3C2003) product:

1. Throughout the documentation there are numerous references to System Software #1 (version 1.5.2) and 3+Backup #1 (version 1.5.2). The diskettes are actually all labeled version 1.6.

2. Page 15, subsection POOL, incorrectly states "Assign different numbered buffer pools to partitions of equal sector size and sectors per cluster."

   It should read "Assign different numbered buffer pools to partitions of different sector size and sectors per cluster."

3. Page 28, subsection "Is the correct tape driver installed on your server?" incorrectly states that the System Soft ware #1 diskette you use to establish a console connection requires the correct tape driver in the \drivers directory.

   It should indicate that the correct tape driver must be in the root directory. There is no \drivers directory on the System Software #1 diskette.

4. Page 28, subsection "Were you using 120 MB tape cartridge?" lists an example of the use of the /120 parameter with the TDRIVE.SYS driver. The example incorrectly implies that it can be used with the TAPE.SYS driver. It should read as follows:

   device = tdrive.sys /120

Installing 3+Open TCP/IP on a DOS Boot Floppy

Reference number: 03050007
Date posted: 7/2/90

3+Open TCP/IP client software can fit on one high-density diskette of any size, but the TCPSETUP program checks to see if the target drive is a diskette and forbids the installation if it is. Following is a procedure for installing 3+Open TCP/IP on a high-density diskette.

Because TCPSETUP will not install onto a removable drive, you must first install the TCP/IP software onto a fixed disk drive. Use the DOS SUBST command to avoid conflicts between the TCP/IP software and files that already exist on your hard drive.

Using SUBST

SUBST allows you to substitute a drive letter for a given directory path. For example, to create temporary drive letter X: equivalent to subdirectory \TMP on partition C:, you would type:

```
SUBST X: C:\TMP
```

The subdirectory C:\TMP must exist before executing SUBST. Note: the SUBST command cannot be used with a redirected drive.

Installing 3+Open TCP/IP Client Software on Disk

1. Create a subdirectory to contain the TCP/IP software (if one does not already exist), and use SUBST to assign it a drive letter.

```
MD C:\TMP
SUBST X: C:\TMP
```

2. Insert the 3+Open TCP/IP Installation disk into your diskette drive, make that drive the current drive (for example, B:), and run TCPSETUP.

```
B:TCPSETUP
```

3. When prompted by TCPSETUP for the source and destination drives, specify your diskette drive (B:) as the source drive for TCP/IP installation, and specify whatever drive letter you've created using SUBST (for example, X:) as the destination drive to install TCP/IP upon.

```
Source drive...  B:
Destination...   X:
```

4. When prompted by TCPSETUP for the "hosts file" pathname, enter a pathname that refers to the proper diskette drive (for example, A:\ETC) where the diskette being created will be booted up.

```
Hosts file...  "A:\ETC"
```

5. Format a high-density diskette with DOS using the FORMAT /S option, specifying the diskette drive you are currently using (B:).

```
FORMAT B: /S
```

6. Copy all the directories and files, using the DOS XCOPY command, from the drive you have SUBST'ed (X:) to the diskette drive (B:) which contains the diskette you are creating.

```
XCOPY X:\ B:\ /S /E
```

7. If you intend to use a "hosts" file, create the directory specified in step 4 on your diskette, and place your hosts file in it.

```
MD B:\ETC
COPY H:HOSTS B:\ETC (Make sure you have a hosts file to copy.)
```

8. Delete the directories and files on the SUBST'ed drive (X:).

```
DEL X:\3OPEN\DOSWKSTA\LANMAN\DRIVERS
RD  X:\3OPEN\DOSWKSTA\LANMAN\DRIVERS
DEL X:\3OPEN\DOSWKSTA\LANMAN\NETPROG
RD  X:\3OPEN\DOSWKSTA\LANMAN\NETPROG
RD  X:\3OPEN\DOSWKSTA\LANMAN
RD  X:\3OPEN\DOSWKSTA
RD  X:\3OPEN
DEL X:\
```

9. Remove the substitution made to drive X: as shown below:

```
SUBST X: /D
```

10. Boot the 3+Open TCP/IP diskette in the boot drive you specified in step 4.

Configuring a 3+Open Server for a Serial HP LaserJet Printer

Reference number: 06430015
Date posted: 7/13/90

This article provides a step-by-step guide for configuring an HP LaserJet serial printer on a 3+Open platform. For instructions on configuring other printers, refer to the 3+Open LAN Manager Administration Guide or your printer documentation.

For a 3Com dedicated server (3S/40x, 3S/5xx):

1. From a console connection (3C Start) run the Netsetup program on the server.

2. Choose "Configure Network Printer."

3. Type the path to the installed LANMAN files.

4. Chose the comm. port that will connect to the printer.

5. Choose configuration settings as follows:

```
9600 N 8 1 to=off xon=on dtr=on *ALL OTHER VALUES OFF*
```

6. Reboot the server.

7. Log on to an OS/2 workstation with Admin capabilities and execute the command:

```
NET ADMIN \\servername
```

8. Choose "View This Server."

9. Choose "Add Share."

10. Choose "Spooled Printer."

11. Assign a sharename for the printer, and choose "OK" at the following message:

```
The specified print queue does not exist.  Choose <OK> to
create the queue SHARENAME.
```

12. Configure the queue as follows:

```
Priority = 5
Printer device(s) = the com port configured in NETSETUP
Separator file = laserjet.sep
```

ALL OTHER VALUES MAY BE LEFT AT DEFAULT.

(Instructions for creating a separator file called "laserjet.sep" with the appropriate Laserjet reset sequence are given at the end of this article.)

13. You will then be prompted for permissions for the print queue. Set permissions as needed.

14. Ensure the LaserJet is set for "robust xon" and "serial" printing.

15. Ensure that you are using a true HP LaserJet serial cable between the server and LaserJet.

For a PC server, follow the steps above with these modifications:

1. A PC server does not use the 3Com console connection. Skip step 1, and type "NETSETUP" at the \3OPEN\SERVER\LANMAN\NETPROG subdirectory of the server.

2. On a PC server the NET ADMIN command does not need to be executed from another OS/2 workstation, although it may be if you wish. You may type "NET ADMIN" from the keyboard of the PC server itself.

In configuring serial LaserJets, the most common source of failure is bad comm. port parameter settings. If the LaserJet does not work after you have followed the steps above, check the comm. port parameter:

For a 3Com dedicated server:

1. From an OS/2 or DOS Enhanced netstation, establish an administrative connection to the server using command line mode:

```
NET ADMIN \\servername /C
```

2. From the [\\servername] prompt, type NET STOP commands for the SPOOLER and (if applicable) 3INTERNET services:

```
NET STOP SPOOLER
NET STOP 3INTERNET
```

3. Verify the mode settings for the port as follows:

```
C:\OS2\MODE COMx
```

   where COMx is the port to which the printer is attached. The settings returned should be

```
9600 N 8 1 to=off xon=on ..... dtr=on...
```
(all other parameters are off)

4. If the com port settings are incorrect, edit the file SETMODE.CMD to insert the correct values. (SETMODE.CMD is created by NETSETUP when configuring printers and is called by STARTUP.CMD during the boot process.) SETMODE.CMD executes the OS/2 MODE command for each port configured in NET SETUP and will look something like this:

```
C:\OS2\MODE COM1 9600 N 8 1 to=off, xon=on......
C:\OS2\MODE COM3 9600 N 8 1 to=off, xon=on......
```

The values in SETMODE.CMD may be changed with a text editor. You must exit the NET ADMIN connection, link to the C$ sharename of your server, and edit the file C:\SETMODE.CMD.

If the settings returned from step 3 are "1200 E 7 1....." the port is still set at the OS/2 default, and the SETMODE.CMD file did not execute properly when the server was rebooted. The file should be tested manually as explained in step 7.

5. Once the new values for the SETMODE.CMD file are in place, establish a NET ADMIN \\servername /C connection.

6. Ensure that the SPOOLER and 3INTERNET services are not started. If either of those services are started, any attempt to use the OS/2 MODE command to access a serial port (either by typing SETMODE commands or executing the SETMODE.CMD file) will result in this error:

```
Com port specified has not been installed
```

7. Run the SETMODE.CMD file by typing

```
C:\SETMODE
```

If the file executes successfully, the message "Asynchronous communication mode has been set" will be returned.

8. Restart the SPOOLER and (if applicable) 3INTERNET services.

9. Exit the NET ADMIN connection.

10. Test the new comm. port setting by printing a file.

The new SETMODE.CMD file will now be executed whenever the server is rebooted.

For a PC server:

Follow the above steps keeping in mind that a PC server does not require a NET ADMIN \\servername /C connection to be established from a workstation in order to configure the ports, though it may be done this way. You may use the MODE command locally at the server after stopping the SPOOLER and 3INTERNET services.

Creating a separator file for an HP LaserJet

1. Run the NETSETUP program.

2. Choose "Configure network printers."

3. Enter the appropriate paths for the server configuration files.

4. Choose "Manage separator files."

5. Choose "Create new separator page file."

6. Name the file "LASERJET.SEP". If you wish to name the file something else, you may do so as long as you specify the chosen name in the "separator file" for the print queue configuration in the NET ADMIN interface.

7. Under the "Reset Sequence Filename:" option, specify the file "LASERJET.RES." The LASERJET.RES reset sequence is provided by default with your 3+Open.

A separator file will be created from the 3OPEN.SEP model with the appropriate LaserJet reset sequence.

Configuring 3+Open Internet for 3+Open Mail

Reference number: 06430016
Date posted: 7/13/90

This article provides a step-by-step guide to configuring 3+Open Internet for use with 3+Open Mail or 3+Mail. It assumes that the 3+Open Internet service (both server and client) as well as 3+ or 3+Open Name and Mail have already been installed and started. It is written specifically for a Hayes 2400 baud modem. For information on using other modems with this configuration, refer to the 3+Open Internet Administrator's Guide and your modem documentation.

1. From an OS/2 netstation, run the 3ICM ADMIN command.

2. From the View menu, choose "ICM Servers" and select the server you wish to configure.

3. From the Config menu, choose "Comm Ports."

4. From the Config menu, choose "Add."

5. From the Comm Port Attributes menu, make the following entries:

```
Comm Port =           COMx (where x is the port the modem will be
                           connected to)
Modem Type =          Hayes24
Speed =               2400
Inactivity Timeout =  default of 5, may be changed to a new value
Direction =           both
```

Choose "OK" to accept the entries. The "Hayes24" profile comes default with 3+Open Internet and may be viewed under the "Modems" option of the Config menu.

6. From the Config menu, choose "Routes."

7. Under the Routes List window, choose "Add."

8. Make the following entries in the "Route Attributes" window:

```
Network ID =   XXX
```

Where XXX = the network number of the remote network with which you are to exchange mail. You do not need to include preceding zeros.

```
Comm Port =   XXX
```

Where XXX = the port configured in step 5 above.

```
Phone Number =  XXX
```

Where XXX = the phone number for the modem to be used at the remote network listed above.

Do NOT choose "OK" to accept the entries.

9. Under the Schedules box in the attributes window, choose "Add," then choose "OK" to accept the entries.

10. Select the dial-out schedule you want between the networks.

11. Choose "OK" at both the Dialout Schedule window and the Route Attributes window to accept your entries.

12. Exit 3ICM Admin.

13. From your OS/2 netstation, link to the OS2APPS sharename.

14. For each domain that you want to send mail to, use the 3N2 ADD DOM command to add the remote domain and network number.

15. Repeat steps 1-14 at each remote site.

You are now ready to exchange mail with the remote network via 3+Open Internet.

Obtaining 3+Open 1.1 NDIS Drivers

Reference number: 08260001
Date posted: 12/5/89

The Protocol Manager and Network Driver Interface Specification (NDIS) adapter drivers from 3+Open 1.1 were recently uploaded to Ask3Com. The drivers allow a protocol (XNS, NBP, etc.) to access the network in a consistent and well-defined manner, regardless of the adapter or media type (Etherlink II, Token Ring Plus, etc.).

Because NDIS is becoming an industry standard interface to network adapters, the drivers are being used by protocols created by third party developers as well as by 3+Open.

While most of the drivers were shipped in 3+Open 1.0, the 3+Open 1.1 release includes the following upgrades:

- All drivers support NBP (NetBIOS Protocol)
- Etherlink Plus DOS driver. Both OS/2 and DOS versions now support DLC.
- Tokenlink Plus DOS and OS/2 drivers
- Protocol Manager and NETBIND bug fixes
- Protocol Manager text files
- Support for Demand Protocol Architecture (DPA) drivers

See the 3+Open LAN Manager Version 1.1 Release Notes for supported adapter and protocol combinations.

Developers who have the Developer's Guide to Network Adapters (3C576) should also download these files. Included are PRO.MSG and PROH.MSG which were inadvertently omitted from the 3C576 diskettes. Also there is a sample PROTOCOL.INI which may assist developers building this file.

All of the files have been compressed into a single file, NDIS.ZIP, which is in the Ask3Com Forum, Drivers/ Diagnostics library. Use PKUNZIP 1.02 to unpack it into individual driver and text files.

Implementing Reset Sequences

Reference number: 11020017
Date posted: 6/22/90

Reset sequences are files that contain codes for resetting a printer to its default operating state before each print job. They are used for such tasks as printing a landscape document and then resetting the printer to portrait mode for the next document.

Since there is no standard programming language for printers to use, it is the administrator's task to set up a reset sequence for a given printer on the network.

A reset sequence file must have the extension .RES and be placed in the \Spool subdirectory. Here is a sample reset sequence for the HP LaserJet:

```
1B 45 1B 26 6C 31 34 63 31 65 37 2E 36 34 63 36 36 46
```

Follow the steps below to implement the reset file:

1. Type Netsetup.
2. Select "Configure network printers."
3. Select OK at the next screen "Path to 3open directory" if the path is correct. Otherwise, correct the path and then select OK.
4. Select OK at the next screen "System Initialization File" if the path is correct. Otherwise, correct the path and select OK.
5. Select OK at the next screen "MS OS/2 Mode Command File" if the path is correct. Otherwise, correct the path and select OK.
6. At the next screen, "Printer Configuration Menu," select "Manage Reset sequence files."
7. At the "Manage Printer Reset Sequence Files" screen, either select a reset sequence file or select "Create printer reset seqence file."

   If creating a new file, enter the hexadecimal number for each character you want to send, using the format in the example above.
8. Once the file is created or selected, go back to the previous screen "Printer Configuration Menu" and select "Manage Separator Files."
9. Select your .SEP file.
10. Enter the filename of the reset sequence.
11. Return to the main menu and quit.

Your printer will now use the codes in the reset sequence between print jobs.

3SIS Pitfall Using 3+Open 1.1E Maintenance Update Diskette

Reference number: 16530016
Date posted: 7/13/90

During installation of the 3+Open 1.1E Maintenance Update Diskettes (MUD), one possible pitfall is the removal of the apps tree on the C: partition.

Make sure the directories C:\APPS\DOSAPPS, C:\APPS\OS2APPS, and C:\APPS\FAMAPPS still exist. Many times these directories will be moved to another partition and then removed from their location on C:. However, 3SIS often needs all three of these directories to exist on C: to copy files.

After running the MUD installation program, the newly installed files can be copied to the DOSAPPS, OS2APPS, and FAMAPPS sharenames, then deleted from the C: drive if the locations of these sharenames are not the default locations. Make sure you leave the directory tree intact.

Cannot Boot DOS Workstation After 3+Open 1.1 M.U.D.

Reference number: 16530017
Date posted: 7/2/90

The following error may occur during update of existing 3+Open 1.1 files to the 1.1 Maintenance Update Diskettes (MUD):

```
Error spawning c:\3open\doswksta\lanman\.exe
```

It is caused by the NET.EXE program looking for an end of line (EOL) character when parsing the program.

To avoid the error, insert a carriage return on the last line of the LANMAN.INI file on the DOS workstation (to provide the necessary EOL character), or remove the installed files while in NETSETUP, then reinstall them.

Deadlocks Using Paradox Database LAN Versions 2 and 3

Reference number: 17310010
Date posted: 3/7/90

Borland has identified certain circumstances which cause users to be locked out of a database file: multiuser database access can result in a resource deadlock, caused by the record-locking mechanism used by Paradox. Once the locks are released, normal operation can resume. This can occur in any 3Com 3+ or 3+Open network environment.

Every Paradox data file has another file associated with it that is used to lock the records within the file. The lock file is named FILENAME.LCK, where FILENAME is the name of the database file. The FILENAME.LCK file allows one Paradox user to have exclusive access to write to a record in the database file. (Unlocked records are available for use by other users, who are notified which records are locked and who has locked them.)

If a user attempts to lock a record, the lock file is opened, the lock is noted and then the file is closed. However, if more than one user is editing a data base at one time, the lock file may become inaccessible, due to a resource deadlock. One user will not release a resource until another unavailable resource is released. Evenutally, all users contending for the deadlocked resource receive no response from the application and their workstations are frozen.

A Paradox network user cannot release database resources allocated by the lock files as a result of a catastrophic interruption of network operations. The lock files in question are located in the shared data directory or in the user's private directory. Once these are deleted and the server has been shut down and rebooted, the database can be accessed successfully by users. Refer to Borland for further information regarding the implementation of Paradox database operation in a multiuser environment.

Using Dial-Back Modems with Remote PC

Reference number: 18960009
Date posted: 06/22/90

3Com's 3+ and 3+Open network operating systems transmit XNS packets encapsulated within NMP packets across phone lines, so they are less susceptible to unauthorized access than many other hosts and networks because the intruder would have to be running Remote PC, 3+Route, or 3+Open Internet service to successfully generate and decode these packets.

Dial-back modems are also an effective and commonly used method of providing security for computer networks. Because Remote PC was not designed to take dial-back modems into account, they are not supported by 3Com. However, if you require dial-back modem access to a 3Com network, it can be made to work as follows:

1. Set up the Remote PC disk as documented in the Installation/Configuration Guide.

2. Edit the AUTOEXEC.BAT file to pause before dialing.

3. Edit the PROFILE.SYS file to add the line "LINE1=;ATD".

The command tells Remote PC to place the modem in command mode (;) and then pick up the phone line in originate mode.

4. Boot the PC, and wait for the pause statement within the AUTOEXEC.BAT file to be executed.

5. Manually dial the phone number, enter any codes necessary, then hang up. When the phone rings, press Enter on the PC to continue the AUTOEXEC.BAT file and issue the ATD.

These procedures were tested with a LeeMah Datacom Traqnet 2001 dial-back device. ■

For more information on Ask3Com, see 3INFO, page 105 of this issue of 3TECH.

Q & A: Ask The 3Wizards

"Ask the 3Wizards" is a forum for 3TECH readers to pose questions about 3Com products and technology. These questions are answered by members of the 3Wizard council. The 3Wizard council consists of both network integrators and end users of 3Com products. The statements made by the 3Wizard council are their own and do not necessarily reflect an official 3Com position.

Q: "Where can I get a workshop for 3Server3 and 3Server386 tuning? I'd like this workshop to include hands-on training, cover optimization issues, and discuss error codes/resolutions for common situations."

A: Courses are offered by 3Com that deal with tuning 3Com's servers. Several of 3Com's courses cover various aspects of tuning servers. Basic tuning for the 3Server/400 and 3Server/500 is covered in "3+ Network Installation & Administration" (for 3+ NOS) and "3+Open Network Installation & Administration" (for 3+Open LAN Manager).

More advanced tuning is covered in "3+ Advanced System Support" (a class for 3Wizard certification) and "3+Open Advanced Network Administration." All of these courses are about 40-50 percent hands-on training. In addition, 3Com offers a self-paced hardware video for the 3Server/500 and 3Server/600 product line that covers the use of 3Com diagnostics.

Other sources for workshops and training on 3Com networking products are 3Com-certified training centers, located throughout the U.S. and Canada.

Useful reference documents include the *3Server/500 Technical Reference Manual* (3C1093) and the *3+Open LAN Manager Technical Reference* (3C2633). For more information on 3Com training classes and a listing of certified training centers, see the *3INFO* section of this issue of *3TECH*.

To Submit Questions for 3Wizdom:

If you have a technical question for the 3Wizdom Q & A column, send your questions via e-mail to: 3TECH:HQ:3Com (please use "3Tech Q & A" on your subject line). You can also submit your questions by writing to *3TECH*'s Editor, Marianne Cohn, 3Com Corporation, 5400 Bayfront Plaza, P.O. Box 58145, Santa Clara CA 95052-8145.

Looking for Missing 3Wizards...

Are you a 3Wizard who has moved or changed jobs since you were certified? If so, please contact 3Com to update your address and telephone number in the 3Wizard database. This information is needed so you can receive quarterly 3Wizard mailings, and learn about upcoming 3Wizard conferences and special events.

Mailing address/phone information should be sent to:

3Com
Attn: Char Andrews, Training Services Coordinator
2400 Condensa Street
Santa Clara, CA 95052-8145

You can also call with updated address/phone information:

(408) 492-1790, Option 5, Ext. 710

Wizbiz...News from The 3Wizard Council

By Michael Chacon

The 3Wizard council and 3Com's 3Wizard conference management team is hard at work planning for the next 3Wizard conference to be held May 19-24, 1991 in Santa Clara, California. The next conference promises to be the best ever—conference sessions will be led by the engineers who helped to develop the products. Technical session topics will include network management, hubs, and internetworking. Guest speakers from 3Com's executive team will be present to discuss 3Com's product direction *and* to get your feedback on 3Com strategy.

Plans under discussion for the next conference include a live network running at the conference complete with 3Com bridges, routers, and hubs, hands-on network training, and a demo of 3Com's new directory and OMA network management services. Sounds good? If so, start your planning now to attend this important 3Com event! 3Wizards will be receiving more information about the conference in the mail soon—make sure your address in the 3Wizard database is up-to-date!

Other news...The 3Wizard council now represents the 3Wizard community worldwide. We would like to welcome the following new members of the 3Wizard council: Martin Van Schooten, Repko Networking BV, The Netherlands and Leif Gustavsson, Esselte Systems, Sweden. Other members of the 1991 3Wizard Council are Michael Chacon (President), America's Computer Company, La Mirada, California; James Carpenter, U.S. Department of Labor, Washington, D.C.; Gary Cunningham, GTE Business Systems, Lexington, Kentucky; Brian Dluhy, Shared Medical Systems, Malvern, Pennsylvania; Dave Gero, Consultant, Manchester, New Hampshire; Ian Stewart Glass, Cadence Computer Design, Vancouver, B.C., Canada; Claude King, University of Florida, Gainsville, Florida; and Richard Morgan, Base One, Inc., N. Miami Beach, Florida. ■

The 3Wizard council is a group chartered to represent the interests of 3Wizards. Its members serve as technical advisors to 3Com. If you are a 3Wizard and would like to provide feedback to your council members, write to "Wizbiz", c/o 3TECH Journal, Attention: Marianne Cohn, 3Com Corporation, P.O. Box 58145, Santa Clara, CA 95052-8145.

Information On Resources For 3Com Users

Ask3Com Bulletin Board

Ask3Com is 3Com's on-line information service which is available on CompuServe® You'll find technical articles, product and service information, patches, fixes, utilities, and a lively message section.

You can access Ask3Com on CompuServe's HARDWARE and SOFTWARE menus, or by entering "GO THREECOM" at any ! prompt. If you don't have a CompuServe account, 3Com has arranged for you to receive an introductory membership at no charge. Just call 1-800-848-8199, and ask for representative 44. You'll receive a kit containing a $15 usage credit which is good anywhere on CompuServe.

3Com authorized resellers and holders of 3Com service contracts in the U.S. are eligible for a direct access Ask3Com account. If you are a reseller, or hold a contract, and don't have your account, call 1-800-876-3COM (1-800-876-3266), press option 3, then option 2 and ask to be signed up.

3Com User Groups

3Com user groups are a great way for networking professionals to come together to share information and expertise. Members include network administrators, managers, consultants, resellers, developers, technical specialists, and others who work closely with 3Com networks (or would like to learn more about them).

User groups add the human dimension to networking by providing a forum for discussion and knowledge building. They are organized by and for their members, which makes them uniquely responsive to user needs. Since the user group program was established in 1989, the 3Com user group network has grown to more than 80 worldwide with over 5,000 members.

How does 3Com Support User Groups?

The 3Com user group program offers a variety of services to help you build, run, and grow a 3Com user group. These services include:

- A start-up kit with a user group registration form
- The *User Group Handbook*, full of information on how to start and run a user group
- Monthly mailings with product, program, and technical information
- A speaker's bureau, to connect your group with corporate and local 3Com speakers
- The User Group Council—a forum for communication between 3Com and users
- Audio and video tapes featuring presentations from 3Com's technical experts

In addition, 3Com helps individual user groups with membership promotions, regional conferences, trade show tie-ins, and other special programs.

How Do You Become a Member?

Check the list of 3Com user groups (see page 114) and call the one nearest you for more information. If there isn't a group nearby and you're thinking of starting one, give us a call at one of the numbers listed below.

To find out more about 3Com user groups or get a start-up kit:

From the U.S. or Canada, call 1-800-NET-3COM (1-800-638-3266). From outside the U.S. or Canada, call 408-988-1161 or Fax 408-764-5001, Attention: Joan Tabb.

User Groups News: Pan-European User Group Meeting To Be Held

The 3Com user group program is sponsoring its first ever pan-European user group meeting on March 16, 1991 in Hanover, Germany. The meeting will be held in conjunction with Hanover Fair-CeBIT, the world's largest computer and communications industry trade show. The one-day event will afford user group members the chance to exchange information, learn about 3Com's products and strategies, and discuss their computer networking needs with 3Com representatives. For more information, contact Joan Tabb at 408-764-5715 or by fax at 408-764-5001.

3Wizard Program

The 3Com 3Wizard program was conceived in 1986 to create a technical "special-forces" unit—a group of experts in 3Com's network operating systems. Today there are nearly 2000 3Wizards worldwide.

The 3Wizard program includes 3Com-sponsored 3Wizard conferences, quarterly mailings, and eligibility for membership in the 3Wizard Council. 3Wizard conferences provide ongoing training for 3Wizards and a forum to meet and discuss ideas. The 3Wizard mailings contain inside information on 3Com products and technology—exclusively for the 3Wizard community. The 3Wizard Council acts both as a technical advisory group to 3Com and as an independent agent for organizing the 3Wizard community.

Three specialized technical training curricula were developed to certify participants as 3Wizards. The options include becoming a 3+, 3+Open, or an Enterprise Systems 3Wizard. Acknowledging that today's systems engineer needs in-depth exposure to LAN and enterprise technology, 3Com also provides a comprehensive curriculum path known as the 3Wizards Sage program. The title of Sage is reserved for those distinguished individuals who have completed either the 3+ or 3+Open 3Wizard Path, plus the Enterprise 3Wizard Path. To register for these classes, see "3Com Education Services/Course Registration", next.

3Com Education Services/Course Registration:

3Com provides training programs to help you further your computer networking education. The current Education Services Course Catalog and U.S. course schedules may be obtained by calling 1-800-NET-3COM. To register for classes scheduled in the U.S., dial 1-800-876-3Com, press option 7, and select 1. In Canada and Europe, regularly scheduled classes are available through the following 3Com subsidiary offices.

3Com Canada (Toronto)
Phone: (416) 477-3003
Fax: (416) 477-7868

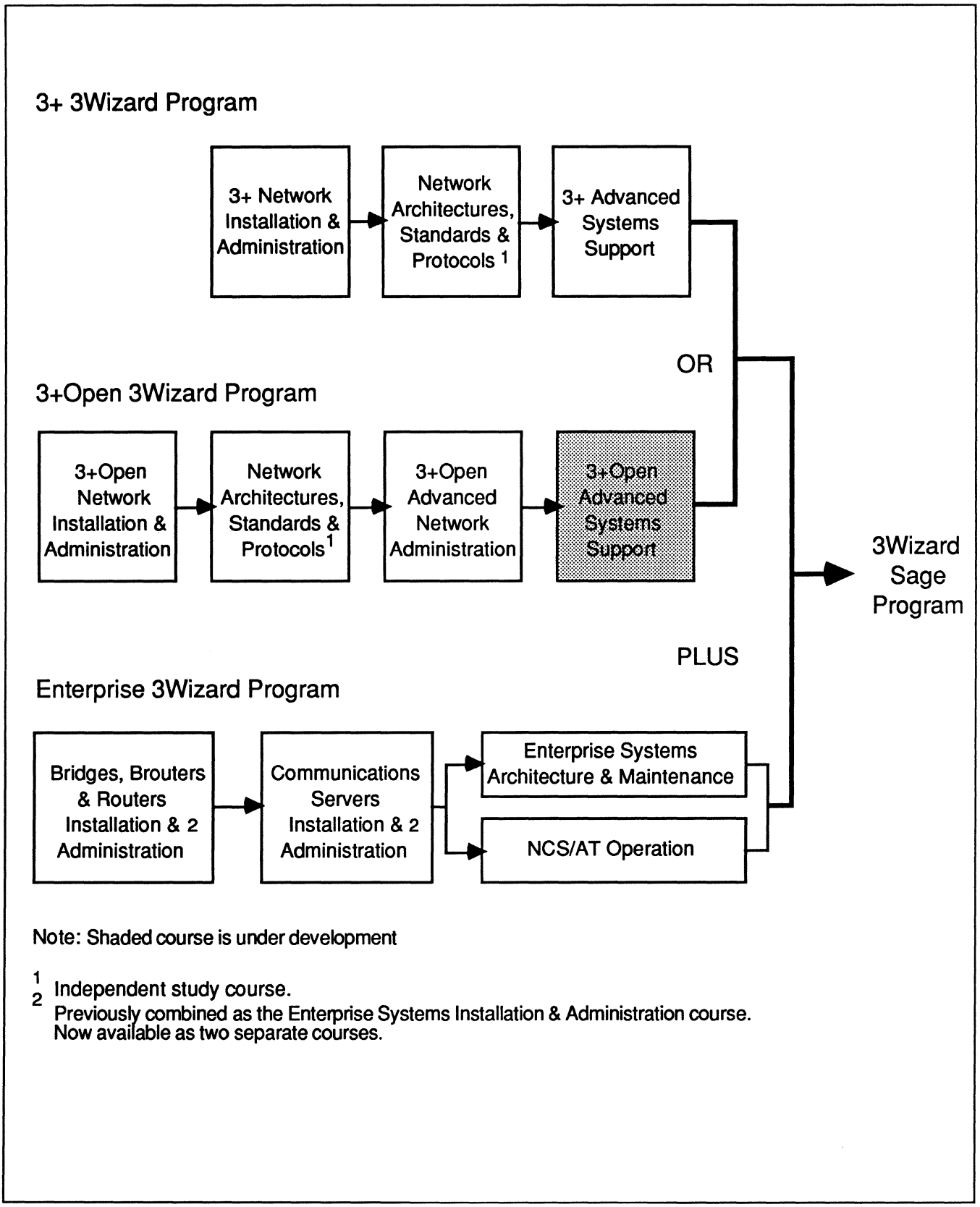
3Com France (Paris)
Phone: 33 1 69 86 68 00
Fax: 33 1 69 07 11 54

3Com Germany (Munich)
Phone: 49 8967 821100
Fax: 49 8967 821233

3Com Italy (Milan)
Phone: 39 2 254 9741
Fax: 39 2 253 8027

3Com U.K. (Buckinghamshire)
Phone: 44 628 890670
Fax: 44 628 898026

3Com Nordic AB (Stockholm)
Phone: 46 8 703 4870
Fax: 46 8 703 4875



Note: Shaded course is under development

1 Independent study course.
2 Previously combined as the Enterprise Systems Installation & Administration course. Now available as two separate courses.

To arrange for a training class in Asia and the Pacific region, resellers should contact:

Tim McCullough
International Training Programs Manager
3Com Corporation
P.O. Box 58145
Santa Clara, CA 95051-8145
Phone: (408) 764-7203
Fax: (408) 764-7290

Regardless of how basic or technically advanced your needs are, 3Com's courses will help you sell, install, use, administer, manage, service, and expand your computer networking systems.

The curriculum recommendations shown on the following page should assist you in determining what courses are most appropriate for you. Descriptions of these courses can be found in the Education Services Course Catalog.

3Com-Certified Training Centers

3Com-certified training centers offer training and workshops on 3Com products and networking technology to augment those offered by 3Com Education Services. The following is a listing of 3Com-certified training centers.

Base One, Inc.
North Miami, FL
305-652-4077

Benchmark Computer Systems
Minneapolis, MN
612-831-2300

Cadence Computer Design, Inc.
Vancouver, B.C. (Canada)
800-ONE-SOFT

Compunet Inc.
Fair Oaks, CA
800-346-9363

Data Systems Network Corp.
Farmington Hills, MI
313-474-4415

Financial Systems & Equipment Corp.
Richardson, TX
214-235-2138

GA Computer Products
Buffalo, NY
716-854-0004

Ingram Micro D, Inc.
Santa Ana, CA
800-456-8000

Fremont, CA
800-456-8000

Buffalo Grove, IL
800-456-8000

Carrollton, TX
800-456-8000

Columbia, MD
800-456-8000

Clarkston, GA
800-456-8000

JWP Information Systems/ Memory Systems
Washington, D.C.
202-429-1922

LAN Solutions
Marietta, GA
404-980-0033

LansPlus
Montreal, Ottawa, Toronto (Canada)
514-875-9023

Network Compatibility Group
Columbus, OH
614-436-2962

NMI/LAN Services
North Brunswick, NJ
201-846-1234

PTXI
San Francisco, CA
800-783-PTXI

Curriculum Recommendations

New Users

3+Open Network User, Computer-Based Training[1] or 3+Open Network User, Lecture Format

3+ Technical Support and Network Administrators

3+ Network Installation & Administration → Network Architectures, Standards & Protocols[1] → 3+ Advanced Systems Support

3+Open 1.1 Technical Support and Network Administrators

3+Open Network Installation & Administration → Network Architectures, Standards & Protocols[1] → 3+Open Advanced Network Administration → 3+Open Advanced Systems Support

LAN Manager 2.0 Technical Support and Network Administrators

LAN Manager 2.0 System Administration → LAN Manager 2.0 Integration & Operation

Enterprise Systems Technical Support and Network Administrators

Bridges, Brouters & Routers Installation & Administration or Communications Servers Installation & Administration → Enterprise Systems Architecture & Maintenance

Bridges, Brouters & Routers Installation & Administration or Communications Servers Installation & Administration → NCS/AT Operations → Enterprise Systems Architecture & Maintenance

Service Technician

3Server Network Server Service & Repair[1]

Enterprise Systems Architecture & Maintenance[2]

LinkBuilder Installation & Maintenance

Los Angeles, CA
800-783-PTXI

Denver, CO
800-783-PTXI

Chicago, IL
800-783-PTXI

Washington, D.C.
800-783-PTXI

New York City, NY
800-527-0177

Dallas, TX
800-527-0177

Austin, TX
800-527-0177

San Antonio, TX
800-527-0177

Houston, TX
800-527-0177

SpectraCom Inc.
Simi Valley, CA
805-527-2326

Valinor
Westford, MA
508-692-9404

Manchester, NH
603-668-1776

Seattle, WA
206-722-4221

Wordlink, Inc.
Champaign, IL
217-359-9378

Schaumburg, IL
708-397-4000

St. Louis, MO
314-878-1422

San Francisco, CA
415-392-5465

3Com Education Services Featured Courses

Enterprise Systems Architecture and Maintenance
3CS-303
Two-day course
$775 list price

This hands-on course provides advanced network installers and support engineers with the skills needed to install, configure, troubleshoot, and replace subassemblies in communications servers, bridges, routers, brouters, and network control servers. You will learn how to properly install an Ethernet network, twisted-pair network, and a fiber network and how to attach devices. You will also learn how to test and replace the media and the major components within individual communications servers. You will participate in interactive sessions exploring common problems associated with communications servers, attached devices, and Ethernet media. Finally, you will work as a team with other participants during various troubleshooting exercises to help define and correct potential network problems. Troubleshooting techniques and tools include the Network General Sniffer®, Protocol Analyzers, NetProbe™, and Multimeters. Successful completion of either Bridges, Routers and Brouters Installation and Administration or Communications Servers Installation and Administration (or their predecessor, Enterprise Systems Installation and Administration) is required before enrolling in this course.

3+Open Advanced Network Administration
3CS-534
Four-day course
$1,200 list price

This hands-on course is essential for any advanced support engineer or experienced network administrator supporting a 3+Open network or installing a multiserver, single-site 3+Open network. Learn to adjust workstation and server parameters to optimize memory and performance. Study the way data flows between workstations and servers. Learn to use the command line interface to administer a 3+Open network. Organize your server's hard disk efficiently. Focus on protocol considerations and the reasons why they make an impact. Master advanced techniques for 3Com's value-added services,

such as moving the NS_DOM files from one working server to another, moving mailboxes from one server to another, what to do with undeliverable mail, and improving Start perfomance. In general, this course helps simplify the complexity of network administration. 3+Open Installation and Administration is a prerequisite.

3Com Education Services Survey

3Com is committed to simplifying the complexity of data networking for its customers. Training can play an important role in achieving that goal. Please take a few minutes to complete the questionnaire at the back of this issue of *3TECH* and fax it back to us today. Your feedback will help us determine what training courses we should offer to meet your training needs.

3Com Service and Support

3Com offers a full range of service offerings including network integration services, network management consulting, installation services, on-site support, telephone support, a variety of service contracts, training, and special service programs for our resellers.

To receive information on 3Com services, end users should contact their local 3Com account manager or 3Com reseller. To find out the location nearest you, call 1-800-NET-3Com. This will connect you to our customer resource center, where a representative can send you a packet of literature on 3Com services, including the Education Services Course Catalog and current course schedule. 3Com resellers can find out details of the service and support programs that 3Com offers its resellers by calling their 3Com account manager.

3Com Laser Library

3Com has just introduced the 3Com Laser Libray, a CD ROM-based information product which will greatly reduce the time required by customers and resellers to locate information about 3Com products. Laser Library, which is available on an annual subscription basis, contains a vast quantity of technical and performance information about 3Com products. The library includes product manuals, release notes, bug fixes and patches, data sheets, brochures, article reprints, and articles from 3Com periodicals such as *CONNECT* and *3TECH*.

The first release includes documentation and release notes for the 3+Open network operating system, as well as bridges and brouters. In addition, it contains over 1200 technical articles on all 3Com product lines. These articles are written by technical services engineers in response to reseller and customer technical questions and are normally posted on Ask3Com, 3Com's technical information database. Many 3Com product data sheets are included as full-color VGA images. 3Com's Laser Library will be updated and added to each quarter—eventually all of 3Com's technical documentation will be available in this new format.

Laser Library users will require a CD ROM drive to access the information. This drive is attached to personal computers running MS DOS, very much like the way an external disk drive is added. Third-party software is available for networking CD ROM drives; however, 3Com has not yet certified such products for compatibility with 3+ or 3+Open networks. Pricing will be $600 for an annual subscription (3CS-007). The Laser Library includes the first CD plus three quarterly updates. Each Laser Library update contains the information from previous issues in addition to new information.

For additional information, call 1-800-NET-3Com and ask for department 1465.

Suggested Reading...

Usually, a programmer's guide is just that—a reference for someone who is writing code for a particular processor, language, or application interface. It's expected to contain function definitions and parametric details such as what procedure calls to make and how to format them.

LAN Manager, a Programmer's Guide, by Ralph Ryan (Microsoft Press, 1990) is, surprisingly, more than a good example of a programmer's guide. It has the definitions and examples a novice LAN Manager 2.0 programmer would hope for, and more. Because of this, the book becomes useful even for people who will never write programs but will need to administer or support LAN Manager on networks.

Ryan has written an introduction of more than forty pages. It contains a brief history of PC LAN, DOS networking support and LAN Manager development. In particular, it provides a very good description of NetBIOS and its relationship to DOS functions and networking protocols. The mysterious naming of NetBIOS nodes is well explained, as are concepts of *peer-to-peer* and *client-server* networking. This is very useful for someone who is new to PC networking, or still confused about its architecture (e.g., named pipes, net run, etc.). The only area that receives too little coverage is the *server message block* (SMB) protocol, which has supported the client-server model in network systems like PC LAN, 3+, and LAN Manager.

For programmers, the introduction also nicely covers concepts important to network programming such as file sharing, file attributes, and security. It also covers points important to program architecture and generation such as the mem*ory models* used in 80x86-based systems, mixing languages in source code and using OS/2 features like *Dynamic-Link Libraries*. There are a few significant typographical errors and the treatment of memory models is sometimes cloudy, but the material, especially on DLLs, is very helpful to anyone wanting to understand the LAN Manager runtime environment.

This information is useful for network administrators too, especially the sections devoted to explaining NetBIOS and the concepts of *workstations* and *servers*.

The remainder of the book is devoted to LAN Manager's *application programminginterfaces* (APIs), or families of procedure calls. The categorization, by chapter, is very well thought out:

- An introduction to calling format and the structured naming scheme for procedures and variables also gives an example of how to enumerate lists of stored information, such as users known by a server.

- A chapter on programming for the workstation includes good examples ranging from doing n*et uses* to running programs at a remote location (server). It includes discussion of services like Messenger and security issues.

- A chapter on LAN Manager server services explains what they are and shows how to write and configure one. It also nicely explains the role of LANMAN.INI in setting up services when they start.

- Three chapters are devoted to server administration, audit information, and security/protection APIs. These are well organized, thorough, and contain good examples. The Information-API section is especially useful for administrators who simply want to know what things are accessible and should be expected to be visible when administering a LAN Manager server.

The next five chapters are aimed at writing network applications.

- The first chapter covers *Named Pipes*, *Alerts* and *Mailslots* and contains a good example that describes the alerting service and how to customize one.

- The next chapter deals with the ever-problematic area of network *print spooling* and the API for managing print queues and jobs. This chapter explains the concept of a print processor and how spooling is now a Presentation Manager function under OS/2 1.2.

- The third chapter on network applications briefly describes multiple network protocol support of NetBIOS and the concepts underlying NDIS drivers for adapters. This coverage is too brief, but the remainder of the chapter nicely explains the *NetBIOS control block* (NCB) calling convention for accessing NetBIOS functions.

- The next two chapters do a good job of introducing the concepts important to successful writing and debugging of applications that are distributed and executed across networked machines. Both data and execution distribution are considered. With plenty of examples, the author helps the programmer decide when to use pipes, mailslots, NetBIOS, and remote procedure calls, and how semaphores can be used to coordinate distributed activities.

- The fifteenth and final chapter is a nice touch—it discusses current variations in LAN Managers provided by Microsoft OEMs. It includes LM/X, LAN Server, and the LM 2.0 Peer Server. It also provides a good summary of what's different in progamming DOS LAN Manager applications. This chapter could actually be much larger and perhaps will grow as subsequent editions are published.

In summary, this book is a good reference for anyone concerned with the technicalities of LAN Manager, whether or not programming is contemplated. Since LAN Manager 2.0 APIs are apparently backward compatible with 1.1, the contents are useful for those not yet moving to 2.0. Administrators can use the introductions and examples as a way of understanding the system better and, in particular, use the API definitions to determine what buttons and values they can expect to be able to push, set, and view in LAN Manager administration interfaces. This is certainly appropriate 3Wizard bedtime reading. ■

3Com User Group Directory

Updated December 1990

ALABAMA

3Com User Group
Raymond Worsham 205/ 733-0100
100 Concourse Parkway Suite 100
Birmingham, Alabama 35244

3Com User Group
Bill Hicks 205/ 730-2677
One Madison Industrial Park
M/S HQ1003
Huntsville, Alabama 35894

ARIZONA

Motorola 3Com User Group (in-house)
Curtis Balvanz 602/ 493-5543
3718 East Gelding Drive
Phoenix, Arizona 85032

State of Arizona 3Com User Group
John Wilcox 602/ 274-7253
5717 N. Seventh St.
Phoenix, Arizona 85014

CALIFORNIA

3Com User Group
Matt Scholz 619/ 297-3218
3211 Fifth Ave.
San Diego, California 92103

Sacramento 3Com User Group
Noel Morgan 916/ 965-3112
8080 Madison Ave. Suite 202
Fair Oaks, California 95628

3Com User Group
Andrew Crawford 714/ 737-7033
2910 Colventry Circle
Corona, California 91719

3Com Bay Area User Group
Greg Bulette 707/ 769-1553
925 Lakeside St. Suite 300
Petaluma, California 94952

Simi Valley 3Com User Group
Bill Nolan 805/ 527-2326
80 West Cochran Street
Simi Valley, California 93065

Stanford 3Com User Group (in-house)
Sandra Senti 415/ 723-1683
Networking and Communication Systems
4122 Pine Hall, Room 115-A
Stanford, California 94305-4122

3Com User Group
Khalid Ashraq 213/ 736-4484
2039 Artesia Suite 154
Torrance, California 90504

COLORADO

3Net User Group
Larry Hines 303/ 236-2768
MSHA/ISC
P.O. Box 25367
Denver, Colorado 80225-0367

3Com User Group
Ron Wallace 719/ 528-4425
5550 Tech Center Drive
Colorado Springs, Colorado 80919

FLORIDA

3Com User Group (in-house)
Jerry Murray 813/ 978-7981
1 E.Telecom Pkwy.
P.O. Box 290152, Code: E1-N
Temple Terrace, Florida 33687-0512

3Com South Florida User Group
Victor Delgadillo 305/ 624-4200 ext. 159
15705 N.W. 13th Avenue
Miami, Florida 33169

3Com User Group
Paul Dinsdale 813/ 893-8522
490 First Ave. South
St. Petersburg, Florida 33701

3Com User Group
Darryl Howard 407/ 889-6694
P.O. Box 5000
Mail Code 5388
Altamonte Springs, Florida 32715-5000

GEORGIA

3Com User Group
Bo Reahard 404/ 237-5400
2964 Peachtree Road NE Suite 580
Atlanta, Georgia 30305

HAWAII

3Hug
Burt Lum 808/ 546-4919
P.O. Box 2200, A-15
Honolulu, Hawaii 96841

ILLINOIS

3Com Chicago User Group
Ron Guzicki 708/ 391-1061
230 South Dearborn, 9th Floor
Chicago, Illinois 60604

Central Ill. 3Com User Group
Scott Novak 217/ 333-0563
506 South Wright Street
Administration Building, Rm. 258
Urbana, Illinois 61801

INDIANA

Indiana 3Com User Group
Daniel Schultz 317/ 571-5834
9455 Delegates Row
Indianapolis, Indiana 46240

KENTUCKY

GTE 3Wizards (in-house)
Gary Cunningham 606/253-6692
318 E. Main St.
Lexington, Kentucky 40507

Bluegrass 3Com User Group
Mary Atcher 606/ 253-4381
318 E. Main
Lexington, Kentucky 40507

LOUISIANA

3Com User Group
Don Barry 504/ 838-3264
Room BH 400
1516 Jefferson Highway
New Orleans, Louisiana 70121

MARYLAND

3Com User Group
Joe Breslin 301/ 955-3735
1830 East Monument St., #533
Baltimore, Maryland 21205

3Com CURE (in-house)
Judy Fabrikant 301/ 496-9814
NIH/DCRT Bldg. 12A, Room 3039
Bethesda, Maryland 20892

NASA Goddard User Group (in-house)
Debbie Sharpe 301/ 286-8519
Goddard Space Flight Center Code 251
Greenbelt, Maryland 20771

MASSACHUSETS

Boston Area 3Com User Group
Joe Grande 617/494-8200, ext. 162
1 Main Street
Cambridge, Massachusets 02142

Don Trimble 617/466-9710
1000 Winter Street Suite 4900
Waltham, Massachusetts 02154

MICHIGAN

West Michigan 3Com User Group (in-house)
Carl VanderZee 616/942-9800
6095 28th Street, SE
Grand Rapids, Michigan 49506

3Com User Group
Jon Lytle 315/354-9018
29425 Northwestern Hwy. Suite 300
Detroit, Michigan 48034

MISSOURI

Kansas City 3Com User Group
Tim Gerdts 816/221-6400
1500 Grand Ave.
Kansas City, Missouri 64108

3Com MALAUG
Al Parato 314/878-5003
12655 Olive Blvd. Suite 325
St. Louis, Missouri 63141

NEBRASKA

1st National Bank of Omaha 3Com User's Group (in-house)
Michael Sibilia 402/341-0500
1 First National Center
Omaha, Nebraska 68102

NEW MEXICO

3Com User Group
Michael Sanchez 505/293-9228
P.O. Box 40687
Albuquerque, New Mexico 87196

NEW YORK

Tri-state User Group
Sherard Murphy 212/564-3000 ext. 271
1250 Broadway
New York, New York 10001

3Com User Group
Fred Skrotzki 716/385-6740
300 Main Street
East Rochester, New York 14445

NORTH CAROLINA

3Com User Group
Helen Pikay 919/396-3131
Chief of Information Center
P.O. Box 70616
Fort Bragg, North Carolina 28307

3Com User Group
Bill Birckhead 919/549-0611 ext. 136
2 Triangle Drive
Research Triangle Park, North Carolina 27709

OHIO

3Ohio User Group
Bruce Adamczak 614/261-2738
650 Ackerman Road
Columbus, Ohio 43202

PENNSYLVANIA

3S/LUG
Linda Carroll 215/956-8593
600 Dresher Rd.
Horsham, Pennsylvania 19044

3Com User Group
Barry Pierce 215/775-2600 X2242
P.O. Box 1498
Reading, Pennsylvania 19603

RHODE ISLAND

3Com Rhode Island User Group
Rick Selleck 401/831-6873
One Davol Square Suite 404
The Penthouse—McGuire Building
Providence, Rhode Island 02903

SOUTH CAROLINA

3Com User Group
Larry Russell 803/ 772-3985
579 St. Andrew Road
Columbia, South Carolina 29210

TEXAS

Motorola 3Com User Group (in-house)
Bud Hesch 512/ 891-3091
6501 William Cannon Drive West
Mail Drop OE 315
Austin, Texas 78735

3NTUG 3Com N. Texas User Group
Dick Tribble 214/ 820-2989
3801 Main Street
Dallas, Texas 75226

3Com Houston User Group
Willy Pan 713/ 233-2356
621 Lockhaven Dr.
suite 100
Houston, Texas 77073

3Com User Group
Richard Westerman 713/ 623-6506
4615 SW Freeway
suite 700
Houston, Texas 77027

VIRGINIA

3Com Mid-Atlantic User Group 3MUG
Rolf Lang 703/ 503-9430
10287 Latney Road
Fairfax, Virginia 22032-3247

3Com User Group
Thomas Mills 804/ 929-6701
Old Dominion Box Co.
P.O. Box 680
Lunchburg, Virginia 24505

WASHINGTON

3Com User Group
Jim O'Farrell 206-391-5099
45 Front Street
Issaquah, Washington 98027

Washington State 3Com User Group
Art Brown 206/ 753-2525
Washington State Parks
7150 Clearwater Lane, KY-11
Olympia, Washington 98504-5711

WASHINGTON D.C.

NAUG National Science Foundation Administrator's Users Group (in-house)
David M. Smith 202/ 357-5904
1800 G St. NW
Washington D.C. 20550

3MUG
Richard Langguth 202/ 357-7886
National Science Foundation
OIS Room 401
Washington, D.C. 20550

WISCONSIN

Southern Wisconsin 3Com User Group
Russ Miller 608/755-4166
P.O. Box 1248
2602 Harvard Dr.
Janesville, Wisconsin 53545

International User Groups

Australian 3Com User Group
Graeme K. LeRoux 6 12 968-1595
3 St. Johns Wood
9-11 Mosman Street
MOSMAN, NSW 2088 AUSTRALIA

3Com User Group
Adam Dunstan 61 3 644-6222
47 Brady St.
South Melbourne, Victoria 3205 AUSTRALIA

3Com User Group
Mike Tiernay 604/ 682-3266
1570-701 W. Georgia St.
Vancouver, B.C. V7Y 1A1 CANADA

3Com User Group
Kelly Campbell 604/ 682-3266
Fax 604/ 688-0099
701 West Georgia Street Suite 1500
Vancouver B.C. V7Y1A1 CANADA

3Com User Group
Arthur Berger 613/ 951-5592
Jean Talon Bldg, 9th Floor, Section A5
Tunney's Pasture
Ottawa Ontario, K1A 0T6 CANADA

3CUG
Mark Pytlik 613/ 951-2420
Room 2306, Main Building
Runney's Pasture
Ottawa Ontario KIAOT6 CANADA

3 V.U.G.
Ian Glass 604/ 733-7638
1681 Chestnut St.
Vancouver B.C. V6J 4M6 CANADA

3Com User Group
George Balouxis
1 Holiday St.
East Tower, Suite 557
Pointe Claire QUEBEC H9R 5N3 CANADA

3Com User Group
Gemal Hummad 306/ 352-9664
1016 Winnepeg St.
Regina Saskatchewan S4R 8P8 CANADA

3Com UK User Group
Greg Micallef 44 01 860 5200
164 Queen Victoria Street
London ENGLAND

3Com Forum
Kari Ervasti 358-0-52721
Sinimaentie 14 Box 83
SF - 02631 Espoo FINLAND

3Com France User Group
Elizabeth Jantzen 33 1 69 86 68 00
27 Ave de la Baltique
Local Postal 609
91945 Les Ulis FRANCE

3Com User Group
Kenneth Leung 852 5 807 1223
Imagineering
405 Citicorp Center
18 Whitefield Road
Causeway Bay HONG KONG

I3U - Italian 3Com Users
Primo Bonacina 39 2 254-9741
Via Michelangelo 1
20093 Cologno Monzese
Milano ITALY

3Com User Group
Hayao Washizaki 81 03 356-6351
2-4-3 Shinjuku
Shinjuku-KU
Tokyo 160 JAPAN

Korean 3Com User Group
Peter S. H. Park 82 2 332-0141
351-27, Seokyo-Dong Mapo-Ku
Seoul KOREA

3Com User Group
Lim Foong Lin 60 03 238-3894
22 Floor, Suite 22-01
Plaza See Hoy Chan
Jalan Raja Chulan
50200 Kuala Lumpur MALAYSIA

3Com User Group
M.J. Van Schooten 31 03473-70650
P.O. Box 245
4130 EE Vianen
THE NETHERLANDS

3NZUG
Yvonne Langridge 66-767899
38 Tumene Dr.
Rotorua NEW ZEALAND

3Com User Group
Anthony Lim
Henderson Industrial Park
211 Henderson Rd. #14-01
SINGAPORE 0315

3Com User Group
Annika Kvarnstrom 46 08 703-4870
Box 1110
S-164 22
Kista SWEDEN

3Com User Group
Eckhard Klockhaus 49 02191 51741
Birke 3
5630 Remscheidt WEST GERMANY

3Com User Group
Jean-Pierre Lucaire 49 089/4708196
Prinzregentenstr. 122
8000 München 80 WEST GERMANY ■

3Com Wants To Fulfill Your Training Needs. . .
But We Need Your Input

3Com is committed to simplifying the complexity of data networking for our customers. Training can play an important role in achieving that goal. Please take a few minutes to complete the following questionnaire and fax it to us today. Your feedback will help us determine what training courses we should offer to meet your needs. Thank you for your feedback.

What skills would you like to develop to make your job easier and your network run more efficiently? _____

_____

From the following list of training topics, **prioritize** those that would be of most interest to you, and assist you in your job.

___ SNMP—Management of Multivendor Networks
___ Protocols (OSI, XNS, TCP)
___ Multivendor Connectivity
___ Data Communications Fundamentals
___ Telecommunications
___ Network Architecture
___ Bridging and Routing
___ Network Troubleshooting
___ Network Design
___ Network Economics
___ Network Support, Strategies/Planning
___ Software Implementation (Applications on the Network)
___ Other _____
_____

How would you prefer to be trained in network management?

___ Attend a class on a specific topic
___ Attend a conference where several topics are presented
___ Purchase self-paced training

Have you attended a network management class in the last year? ___ Yes ___ No

If you answered 'Yes' who provided the training? _____

How would you rate the training? (Please circle one) 1 (poor) 2 3 4 5 (excellent)

Would you attend training on network management if provided by 3Com? ___ Yes ___ No

What is your overall perception of the training provided by 3Com? 1 (poor) 2 3 4 5 (excellent)

How much would you consider a reasonable daily tuition for network management training? _____

How many users do you support on your network? _____

Is it a local area network or a wide area network with many locations? _____

List the major difficulties you encounter in managing your network. _____

_____

_____

On what activities do you spend most of your time? (Please check as many as applicable)

___ Network planning and design
___ Installation
___ Maintenance and support
___ Network expansion
___ User training
___ Software programming
___ Other _____
_____

What is your title? _____

Do you influence network management decisions in your organization? ___ Yes ___ No

If you answered 'No', who does? _____

Thank you again for your valuable input. Please photocopy and fax to (408) 764-7290.

Notes

720002-001
ISSN 1051-9637

3Com Corporation
5400 Bayfront Plaza • Santa Clara • CA 95052-8145

ADDRESS CORRECTION REQUESTED

Bulk Rate
U.S. Postage
PAID
Santa Clara, CA
Permit No. 955