SEPTEMBER 1992

NOVELL® RESEARCH

NetWare and LAN Server Client Interoperability via ODINSUP: Part 1

Bryan R. Clark
Systems Engineer
Systems Engineering Division

Ken Nefi
Principal Technical Writer
Systems Engineering Division

This AppNote is the first in a two-part series about integrating Novell NetWare and IBM LAN Server environments. It will examine the Open Data-Link Interface (ODI) and Network Device Interface Specification (NDIS) architectures for DOS clients, and detail various configurations for supporting multiple redirectors and protocols using the same network adapter. It highlights Novell's most recent ODINSUP protocol stack to demonstrate NetWare's interoperability with NDIS protocol stacks.

Contents

Introduction

Many organizations have heterogeneous networks that include both Novell NetWare servers and IBM LAN Servers. As the first in a two-part series on interoperability issues, this AppNote explores the details of DOS clients requiring dual requesters to access both servers. (Part 2 will cover interoperability solutions for OS/2 clients.) To illuminate the benefits of each solution, we'll describe the architectures, drivers, and requesters involved, highlighting Novell's latest protocol stack--ODINSUP--to show how Novell's Open Data-Link Interface (ODI) and Network Driver Interface Specification (NDIS) drivers interoperate. We'll also provide examples of DOS configurations, including AUTOEXEC.BAT and CONFIG.SYS file changes.

This information will benefit systems integrators, consultants, technical support personnel, and system supervisors working in a multivendor environment. It will also be of some help to software engineers who are developing applications that will integrate NetWare and LAN Server (or LAN Manager, 3+Open, Banyan, DEC, and other NDIS-based environments). It serves as a guide to demonstrate solutions that allow multiple redirectors and multiple protocols to share the same network adapter.

The discussion assumes a working knowledge of DOS, the NetWare shell, and Internetwork Packet eXchange (IPX) communications. It will also be helpful to have an understanding of the OSI model and underlying architecture.

Background on DOS Client Integration

Currently, Novell's support for DOS client interoperability revolves around the coexistence of Novell's NetWare shell and IBM's DOS LAN Requester in the workstation. To gain a better perspective on how this dual requester environment is possible, it is helpful to review the history of Novell-IBM network support. We'll first look at Novell's initial approach to integrating the NetWare shell with the IBM Token-Ring LAN environment. We'll then introduce the technology that IBM developed to enable more advanced DOS client interoperability.

Novell's Initial Approach

From the v2.0x versions of NetWare to the present, Novell has been committed to supporting IBM LAN platforms, including Token-Ring and PC Network. The Novell software that has made this integration possible is the NetWare shell.

NetWare Shell Basics. The NetWare shell is a memory-resident program that is loaded in a DOS machine to enable it to communicate with network servers. The shell is responsible for determining if an application's file or print requests are for DOS or for the network. If a request can be handled locally, the shell passes it on to DOS. If a request is for a network file or print service, the shell intercepts it and prepares it for transmission to the appropriate server. In this way, the shell acts as a redirector.

The shell works with IPX, NetWare's Network-layer communication protocol, and a LAN driver specially written for a particular type of network interface card, to send the request out over the wire to the server. (Prior to NetWare v2.1, these components were all linked as a single executable program called ANETx.COM that was generated via the SHGEN program.)

The shell and the file server OS communicate using a common language called the NetWare Core Protocol (NCP). This protocol defines a request-response relationship between the workstation (client) and the server. When the server receives an NCP request, it returns a reply to the client as determined by the protocol.

The shell accomplishes its redirection tasks by hooking several DOS interrupts. The two we'll be concerned with are Interrupt 21 (INT 21) for file services and Interrupt 17 (INT 17) for print services. Once loaded, the shell monitors any calls that applications make to these two interrupts and redirects those that require network services (see Figure 1).

Figure 1: The NetWare shell hooks INT 21 and INT 17 to redirect file and print service calls to a network server.

As an example of how the shell works, consider the case of an application issuing an Open File request for a file located on a network drive. The application first sets the proper subfunction in the CPU's registers and calls DOS Interrupt 21. The NetWare shell intercepts the call, examines the request, and finds that the file to be opened is on a network drive. (If the file were located on a local drive, the shell would simply chain the request to DOS.)

The shell then creates an IPX packet and fills in the NCP information. IPX hands the packet to the LAN driver. The LAN driver knows how to get packets on and off the network interface card, which transforms the information into the actual electronic signals transmitted over the cable.

The response from the server follows the same path, only in reverse order. The LAN driver receives the packet and hands it up to IPX. IPX removes the IPX header and passes it to the shell. The shell interprets the NCP response, loads the proper registers (just as DOS would), and returns from the INT 21 request.

To determine if a particular file is on a local drive or a network drive, the shell maintains a set of tables that keep track of network drives and their mappings. DOS has its own internal drive table that identifies drives as being local, remote, or undefined. The LASTDRIVE command in the CONFIG.SYS file sets the last valid drive letter that DOS will accept. For example, the LASTDRIVE=E command indicates to DOS that it can access drives (either local or logical) up to drive E. NetWare uses the LASTDRIVE statement to determine where to begin the shell's drive table. For example, in the case of LASTDRIVE=E, the shell uses drives F through Z for network drive mappings.

The TOKREUI Configuration. In NetWare v2.0x, Novell supported IBM Token Ring networks through an IPX-driver combination designed to work with IBM's TOKREUI (TOKen Ring Extended User Interface) software. Installers created this driver by running Novell's GENSH or SHGEN program to link IPX with the TOKEN.OBJ file (v2.50 and earlier).

Figure 2 illustrates the relationship between the various network components in the NetWare v2.0x TOKREUI configuration.

For an application to access more than one file server in this TOKREUI configuration, the application would issue INT 21 requests for DOS and NetWare, or INT 5C requests for NetBIOS.

Figure 2: In NetWare v2.0x, the shell and IPX were linked with a LAN driver that worked with IBM's TOKREUI program.

With TOKREUI, the only interoperability or dual requester configuration had to be built in to the application. Developers whose applications needed to access files from DOS or NetWare would use the INT 21 services. For NetWare file services, the shell redirected INT 21 calls and made far calls to IPX/shell functions. IBM's NETBEUI (NetBIOS Extended User Interface), or INT 5C interface, allowed developers to use peer-to-peer communications to access or share files with another PC.

DOS 3.1's Interoperability Technology

In 1987, IBM introduced several new pieces of LAN technology to support client-server computing:

- DOS 3.1
- IBM DOS LAN Requester (DLR)
- IBM LAN Support Program (LSP)

DOS 3.1. The introduction of the DOS LAN Requester and LAN Support Program necessitated some changes to DOS. The most significant new feature was the Multiplexer (INT 2F) interrupt and interface. The design of this interface called for all requesters to line up in a chain. When DOS received a request that was not local, it would pass the request down the chain until it was claimed.

With DOS 3.1, IBM introduced its LAN Server network operating system based on OS/2 v1.0. DOS workstations could load IBM's DOS LAN Requester (DLR) to communicate with LAN Server.

The DOS LAN Requester. At a conceptual level, IBM's DOS LAN Requester or Redirector is similar to the NetWare shell. The DLR is a memory-resident program that is loaded at the DOS workstation. Like the shell, the DLR hooks several DOS interrupts to do its job. It uses the Server Message Block (SMB) protocol, which is like the NCP in that it employs a client/server, request-response dialogue for communication. SMBs are sent to servers via the NetBIOS protocol.

Figure 3 illustrates the components of the DOS LAN Requester.

Figure 3: The DOS LAN Requester loads at a DOS workstation to provide access to an IBM LAN Server.

The DLR is initialized and loaded into memory via the NET START command. This command gets its parameters from the DOSLAN.INI file, an ASCII text file that is set up when the DLR is installed. (For more information about the DOS LAN Requester and its parameters, see Logging In to IBM LAN Server and NetWare from a DOS Workstation in the November 1991 NetWare Application Notes.)

The DLR also supports the Shared Resource Program Interface (SRPI), which evolved from the PC LAN Program (PCLP). The SRPI interface is used by IBM's 3270 emulator and other host applications.

The LAN Support Program (LSP). Along with the DOS LAN Requester, IBM also released its LAN Support Program. This program consisted of a set of device drivers that provided the programmatic interface to support network applications on Token Ring and IBM PC Network hardware. (Ethernet support was not added until the release of LSP version 1.2 in 1991, as we'll discuss later.)

Figure 4 shows the components that make up the LSP.

Figure 4: The components of the IBM LAN Support Program.

The LSP introduced IEEE 802.2 Logical Link Control (LLC) services. The LLC sublayer within the OSI Data Link layer allows the use of other Media Access Control (MAC) drivers to support other topologies beyond Token Ring (IEEE 802.5).

The LSP also defines a standardized NetBIOS interface between network applications and the network hardware support drivers, and an INT 5C Arbitrator to register and allocate network resources. (INT 5C was the original NetBIOS interface/interrupt. The LSP allows 802.2 or LLC packets to be submitted using the same interface.)

Token Ring adapters have built-in microcode for what IBM calls connections, link stations, and Service Access Points, all of which are used to maintain a connection with remote devices on a network.

- A connection is equivalent to a logical path between devices.
- A link station is equivalent to a physical path between devices.
- A Service Access Point (SAP) is a table entry which identifies protocol stacks to the INT 5C Arbitrator.

Each application or protocol stack must identify itself with the INT 5C Arbitrator. The NetBIOS Define_SAP API tells the Arbitrator who to call back when a packet or response is received.

As an example, suppose NetBIOS wants to open SAP F0. It calls INT 5C and issue a Define SAP F0 request. If SAP F0 is not already defined, the Arbitrator (DXMA0MOD) registers F0 to NetBIOS and calls the Token Ring driver (DXMC0MOD). DXMC0MOD.SYS puts the address of the call-back routine in a table.

SAPs 00 through 04 are permanently allocated; SAPs F0 through FF are reserved. Applications use predefined SAPs; for example, IBM's PC/3270 program uses SAP ##, AS/400 PC Support uses SAP ##, and the NetWare Token Ring IPX driver uses SAP E0.

To properly configure the Token Ring card, the developer must know how many SAPs, link stations, and connections to open when the card is initialized. The LAN Support Program comes with a configuration utility called DXMAID that determines the required configuration and adds these parameters to the CONFIG.SYS with the proper drivers.

Extra SAPs and link stations must be defined for programs that use other protocol stacks besides NetBIOS, such as the IBM PC/3270 program and the NetWare shell (see Figure 5).

Figure 5: The relationship between applications, protocol stacks, the IEEE 802.2 layer, and hardware.

The first LSP release included the following device drivers, which are loaded in the CONFIG.SYS file:

DXM T0 MOD.SYS	NetBIOS interface driver
DXM A0 MOD.SYS	INT 5C Arbitrator
DXM C0 MOD.SYS	Token Ring-specific driver
DXM G0 MOD.SYS	PC Network-specific driver
DXM ?1 MOD.SYS	Adds 3270 (HLLAPI) support
DXM ?2 MOD.SYS	Adds 3278 printing support

Figure 6 illustrates both the DOS LAN Requester and the LAN Support Program loaded on a DOS machine with a Token Ring adapter.

In this configuration, the header of each incoming packet is examined. If a SAP is found in the Token Ring header, the packet is given to the associated call-back routine. If a SNAP address is found, then that routine is used. Otherwise, the packet is given to the Direct interface or bit bucket. This call-back order will become important when we discuss LANSUP.COM, because it uses the Direct interface.

Figure 6: Together, the DOS LAN Requester and the LAN Support Program provide network access for a DOS station.

The LAN Support Program provides the underlying foundation for the dual requester client.

DOS Client Interoperability Solutions

There are currently three LAN Server interoperability solutions for NetWare DOS clients:

- Dedicated IPX (SLANSUP.OBJ)
- Open Data-Link Interface on LAN Support (LANSUP.COM)
- LAN Support on Open Data-Link Interface (ODINSUP.COM plus MLID)

Dedicated IPX (SLANSUP.OBJ)

With this setup, the LAN driver is statically linked with the WSGEN (or older SHGEN) program. There are two drivers which use the LAN Support Program driver:

- Native Token Ring driver (TOKEN.OBJ up to v2.50)
- LAN Support driver (SLANSUP.OBJ after v2.50)

The old NetWare Token Ring dedicated IPX driver uses SAP E0. This driver does not allow dual requesters to coexist. By contrast, the LSP driver talks to the INT 5C Arbitrator (DXMA0MOD) and uses the Direct interface (no extra SAPs or link stations). The current version of SLANSUP.OBJ is v2.63, dated 10-18-91, size 6080 bytes. This version is now available on NetWire.

Figure 7 illustrates the dual requester setup with SLANSUP.OBJ on a Token Ring network.

Figure 7: Configuration using SLANSUP.OBJ on Token Ring.

Let's look at some examples of how an Open File request is handled in this dual requester environment.

First, consider a request for a file on a LAN Server drive. The application issues an Open File request (sets subfunction in registers). The NetWare shell examines the path or drive and compares it to its drive table. Since the file is not on a NetWare drive, it passes the request on to DOS. DOS looks in its internal drive table for remote drives and finds the drive, so it calls INT 2F (Redirector). The DLR creates an SMB packet and calls NetBIOS. NetBIOS calls DXMA0MOD, which hands the packet to DXMC0MOD, which sends the packet out onto the wire.

Now let's look at how the NetWare shell handles an Open File request for a file on a NetWare drive. The application does an INT 21 subfunction. The shell intercepts it and finds the drive in its NetWare drive table. It creates an NCP packet and calls IPX. IPX envelops the NCP buffer and calls INT 5C. The INT 5C Arbitrator (DXMA0MOD) takes the packet and hands it to the Token Ring driver (DXMC0MOD.SYS), which sends the packet onto the wire.

When the NCP response comes back from the server, the Token Ring driver receives it and looks first for a SAP, then for a SNAP, but finds neither. So (by default) it sends the response to Direct, which points to IPX. IPX gets the reply and hands it up to the shell. The shell puts the answer in the registers/stacks and returns.

CONFIG.SYS File Changes. The changes that need to be made to the default CONFIG.SYS file created when you install the DOS LAN Requester are underlined in the listing below.

```
files=40
        buffers=40
```

```
FCBS=16,8
SHELL=C:\COMMAND.COM /E:2000 /P
device = \dxma0mod.sys 001
device = \dxmc0mod.sys
device = \dxmt0mod.sys S=16 C=12 ST=16 O=Y
LASTDRIVE=R
```

The NetBIOS driver options are:

S= The maximum number of NetBIOS sessions that can be defined (default is 16 for LSP v1.25)

C= The maximum number of command NCBs that can be outstanding (default is 12 for LSP v1.25)

ST= The maximum number of link stations, or physical paths between devices, that can be defined (default is 16 for LSP v1.25)

O= The Y setting means NetBIOS will open the adapter and allocate resources

The LASTDRIVE=R command sets the boundary between LAN Server and NetWare drives. By default, LAN Server drives begin after local drives. NetWare always starts with the next drive after the one specified in the LASTDRIVE setting.

AUTOEXEC.BAT File Changes. A sample AUTOEXEC.BAT file that will load both requesters is given below.

```
NET START
        IF ERRORLEVEL 1 GOTO NODLR
        CALL INITFSI.BAT
        :NODLR
        YNPROMPT Y N 19 LOAD IPX and NETX?
        IF ERRORLEVEL 1 GOTO NONW
        IPX
        IF ERRORLEVEL 1 GOTO NONW
        NETX
        :NONW
```

The DOS LAN Requester is loaded first via the NET START command. INITFSI.BAT is a batch file that loads the CUA full screen interface and sets up environment variables, paths, and names. Next the Novell IPX driver, generated by selecting SLANSUP.OBJ in the WSGEN program, is loaded, followed by the NetWare shell.

One potential problem in this configuration is that IBM's LAN Network Manager (a product equivalent to Novell's LANalyzer protocol analyzer) also uses SAP E0. This conflicts with the Token Ring IPX driver. Another disadvantage is that this configuration supports only Token Ring hardware, not PC Network II or Ethernet.

LANSUP.COM on Token Ring

This configuration takes advantage of Novell's Open Data-Link Interface (ODI) architecture. Beginning in 1987 with the introduction of LAN Server, IBM, Microsoft, 3Com, and several other industry leaders got together to create the Network Driver Interface Specification (NDIS). Because this specification was closely tied to OS/2 LAN Manager/LAN Server and OEM products, Novell was not invited to participate.

At this same time, Novell was developing support for OS/2 and Macintosh clients, working closely with Apple on the latter. As a result of these efforts, Novell and Apple published the first Open Data-Link Interface (ODI) specification in 1988. That same year, Novell shipped the first NetWare Requester for OS/2 and the Macintosh Service Protocol Gateway for NetWare v2.1.

Although many of the design goals were the same for NDIS and ODI, the NDIS specification lacked true protocol independence from the MAC layer. Most notable are the addressing differences between the Most Significant Bit (MSB) used for Token Ring and the Least Significant Bit (LSB) used for Ethernet. NDIS also lacked support for multiple simultaneous frame types, source routing, and dynamic loading and unloading of components.

The ODI workstation software includes the following components:

- Link Support Layer (LSL)
- Multiple Link Interface Driver (MLID)
- Multiple Network-layer protocol stacks (IPXODI, TCPIP, ATALKII, and so on)

ODI supports multiple protocols and frame types simultaneously on the same network card. The protocol stacks are media-independent, and the MLIDs are network independent. Because the LSL, the MLID, and the protocol stack are dynamically linked, they can be loaded and unloaded from memory.

The Link Support Layer (LSL) is responsible for identifying the type of packet being received by a MLID and passing it up to the appropriate protocol stack (IPX, TCP/IP, AppleTalk, and so on). On a send, the LSL routes the packet from a protocol stack down to the appropriate MLID for transmission.

The LANSUP.COM driver still uses the Direct interface to talk to the INT 5C Arbitrator (DXMA0MOD). On initialization, LANSUP.COM does a DIR_STATUS call to see which hardware driver is loaded. Everything works fine when you have only one frame type supported on the network. However, multiple frame types on the same MLID causes a problem, because the LAN Support Program owns the adapter and cannot handle multiple frame types.

Let's look at the case of LANSUP.COM on Token Ring first for simplicity. This configuration is illustrated in Figure 8. Note that from a LAN Server perspective, nothing has changed. The only thing new is the IPX driver has been divided into the LSL piece and the IPXODI piece to allow multiple protocols (IBM LSP can't do multiple protocols).

CONFIG.SYS File Changes. The CONFIG.SYS file for this configuration is the same as for SLANSUP.OBJ. The NetBIOS open parameters remain the same (LSP v1.25 defaults).

```
files=40
        buffers=40
        SHELL=C:\COMMAND.COM /E:2000 /P
        device = \dxma0mod.sys 001
        device = \dxmc0mod.sys
        device = \dxmt0mod.sys S=16 C=12 ST=16 O=Y
        LASTDRIVE=R
```

Figure 8: Configuration using LANSUP.COM on Token Ring.

AUTOEXEC.BAT File Changes. In this configuration, the load order is still IBM first and Novell second. Note that in place of IPX, you now load LSL.COM, LANSUP.COM, and IPXODI.COM, in that order.

```
NET START
        IF ERRORLEVEL 1 GOTO NODLR
        CALL INITFSI.BAT
        :NODLR
        YNPROMPT Y N 19 LOAD IPX and NETX?
        IF ERRORLEVEL 1 GOTO NONW
        LSL
```

```
LANSUP
IPXODI
NETX
:NONW
```

The advantages of this configuration are that it allows multiple ODI protocols and applications, and the drivers are fully loadable due to the LSL's dynamic load and link capabilities. Also, the conflict with IBM's LAN Network Manager and SAP E0 is eliminated because LANSUP uses the Direct interface.

The disadvantages are that you are still limited to the drivers that come with IBM's LAN Support Program (Token Ring and PCN II for v1.0). Also, LAN Support doesn't support multiple frame types simultaneously.

LANSUP.COM on Etherand

IBM's LAN Support Program v1.23 introduced Ethernet support via the Etherand driver (DXME0MOD.SYS), which is an NDIS version 1.01 driver. The name Etherand is derived from the idea that this driver supports both Ethernet II and 802.2 frame types. The name is really a misnomerþit should be Etheror because you can't have both 802.2 and 802.3 at the same time.

Etherand is the first of the NDIS drivers for IBM LAN Support. It comes with new NDIS utilities, including:

- PROTMAN.EXE, the NDIS Protocol Manager.
- NETBIND.EXE, the NDIS binding utility.

Protocol Manager is the NDIS glue layer. It is the vehicle through which multiple protocol stacks can use a single adapter. Protocol Manager initializes based on information in the PROTOCOL.INI file, a text file similar to NetWare's NET.CFG.

NETBIND is the NDIS utility that actually binds MAC drivers to protocol stacks and sets up the tables through which protocols and MAC drivers can exchange information.

The Protocol Manager Vector Table serves the same purpose as the SAP Table in LAN Support. When incoming packets are received, the MAC driver uses the Vector Table to find the intended recipient. If there are multiple protocol stacks, each stack is called in a chained fashion until the packet is claimed. If the packet is not claimed by the time it reaches the end of the chain, it is discarded.

To send a packet, the protocol stack is given the proper address of the MAC driver from NETBIND. Therefore, the packet bypasses Protocol Manager on the way down and is passed directly to the MAC driver.

Figure 9 illustrates the LANSUP.COM on Etherand configuration.

Figure 9: Configuration using LANSUP.COM on Etherand.

CONFIG.SYS File Changes. In this configuration, the load order is the same: IBM first, Novell second. Protocol Manager (PROTMAN.EXE) and the NDIS MAC driver (ELINKII.DOS in this example) must be loaded before the LAN Support drivers.

```
files=40
        buffers=40
        FCBS=16,8
        SHELL=C:\COMMAND.COM /E:2000 /P
        device=protman.exe
        device=elnkii.dos
        device = \dxma0mod.sys 001
        device = \dxme0mod.sys
        device = \dxmt0mod.sys S=12 C=14 ST=12 O=N
        LASTDRIVE=R
```

Note that the Etherand driver is loaded in place of the Token Ring driver. Note also the change in the NetBIOS Open parameter to O=N (Open=No). With this setting, NetBIOS will not open the card. Instead, the adapter will be initialized during the bind procedure.

AUTOEXEC.BAT File Changes. The following AUTOEXEC.BAT file loads the IBM modules first, then the NetWare modules. NETBIND is run first to bind DXME0MOD to the NDIS MAC driver (MACWD, WDPLUS, ELNKII, and so on).

```
NETBIND
        NET START
        IF ERRORLEVEL 1 GOTO NODLR
        CALL INITFSI.BAT
        :NODLR
        YNPROMPT Y N 19 LOAD IPX and NETX?
        IF ERRORLEVEL 1 GOTO NONW
        LSL
        LANSUP
        IPXODI
        NETX
        :NONW
```

Note: With Etherand, there is an address nibble swap problem when used with an IBM 8209 MAC Bridge. This is due to the LSB and MSB problem mentioned earlier.

When you start adding connections to IBM mini and mainframe hosts through applications such as IBM's PC Support/400 and PC/3270 emulator, the LLC (802.2) resources get used up rather quickly.

Figure 10 illustrates a full LAN Support configuration, in which multiple protocols are loaded to demonstrate all the possible combinations. The NetWare shell uses IPX, LAN Workplace uses TCP/IP, PC Support/400 and PC/3270 use 802.2, and the DOS LAN Requester uses NetBIOS.

Note that the PC Support/400 and PC/3270 programs use extra SAPs and link stations from the LLC layer. With an average NetBIOS application loaded, this represents a maxed out configuration requiring the following parameters:

```
   C=32  S=32  ST=32
```

In this configuration, all of the 16KB of RAM on most Ethernet cards will be used up.

Figure 10: In a full LAN Support configuration, LLC resources can get used up quickly.

This is where ODINSUP can simplify the configuration and offer the needed performance for all the protocol stacks. Further, if RAM usage becomes an issue, ODINSUP can reduce the memory requirements significantly.

ODINSUP.COM and Ethernet

IBM's strategic direction is to move all LAN Support (DOS) and Communications Manager (OS/2) drivers to NDIS. Novell developed ODINSUP.COM to allow the LAN Support Program and other NDIS protocol stacks to sit on ODI. ODINSUP acts like an NDIS MAC driver to pass packets between the LSL and PROTMAN and NDIS protocol stacks.

As we explained previously, ODI can support multiple protocol stacks and multiple frame types simultaneously. Dynamic loading and linking is easier, and the components can be dynamically unloaded too. ODI also offers true media independence for the protocol stacks and hardware independence at the MAC (LSL) layer. It also supports all NetWare MLIDs. Novell also provides a Media Support Module that parses configuration files, does source routing, initializes the card, and registers with the LSL. To help the driver developer, the ODI specification even details when to enable interrupts.

With ODINSUP, the MLID owns the network adapter. Besides supporting all NetWare-certified MLIDs, the main benefit is raw speed for the ODI protocol stacks. On most topologies/adapters, the throughput doubles. The NDIS protocol stacks may experience a slight degradation, but for most adapters and topologies the decrease is negligible (less than 2 percent).

The cost in terms of memory used is 4900 bytes + (size þ number of LSL buffers). When multiple frame types are used, the driver is loaded re-entrantly and only requires another 2080 bytes.

Figure 11 illustrates a full configuration with ODINSUP.COM and Ethernet hardware.

Figure 11: Configuration using ODINSUP.COM and Ethernet.

In this configuration, all Ethernet frame types except SNAP are being used:

- LAN Workplace is running on TCP/IP using Ethernet II frames.
- NETX is running on IPXODI using Ethernet 802.3 frames.
- NDIS is running on ODINSUP using Ethernet 802.2 frames.

Note that it is not necessary to bind a single protocol to a single frame type. We could have had all protocols using a single frame type. However, to keep from having to load new software on the host side, it is better to use the native protocol of each host at the workstation.

We can trace our example Open File request through various protocol stacks. For instance, LAN Workplace does the call through the socket library and creates a packet for TCP/IP. The TCPIP protocol module envelops the packet and gives it to the LSL. The LSL gives the packet to the MLID, where the MAC header is stamped on and the packet is put on the wire.

The NetWare shell does basically the same thing, only it uses the IPXODI protocol stack.

On the NDIS side, when PC Support/400 or PC/3270 does an Open File, the Router modules creates the packet and calls INT 5C. DXMA0MOD gets the packet and hands it to DXME0MOD. DXME0MOD adds the MAC header before it passes the packet to ODINSUP. Since the MAC header is already provided by the protocol stack, ODINSUP and the LSL do nothing; the packet is ready to give to the MLID. The MLID then sends the packet over the wire.

The SMB/Redirector stack works the same way, only it adds the NetBIOS layers.

ODINSUP requires LSL v1.10 or above (the current version is 1.25). The current version of ODINSUP is 1.1, dated 4-27-92, size 33,376 bytes. This version fixes problems with 3Com/Banyan "Failed to obtain PROTMAN dispatch point" errors.

CONFIG.SYS File Changes. The CONFIG.SYS file for this configuration is the same as for the LANSUP.COM example, except there is no need to load an NDIS MAC driver (such as ELINKII.DOS) between the PROTMAN.EXE driver and the DXMA0MOD.SYS driver. This is replaced by the ODI MLID driver loaded in the AUTOEXEC.BAT file. Note that PROTMAN and NETBIND are still used, even though there is no NDIS MAC driver.

```
files=40
          buffers=40
          FCBS=16,8
```

```
        SHELL=C:\COMMAND.COM /E:2000 /P
        device = \protman.exe
        device = \dxma0mod.sys 001
        device = \dxme0mod.sys
        device = \dxmt0mod.sys S=16 C=12 ST=16 O=N
        LASTDRIVE=R
```

AUTOEXEC.BAT File Changes. The main change from the LANSUP configuration is the load order: the Novell ODI LSL, MLID, and ODINSUP modules must be loaded before the NETBIND command.

```
LSL
        NE2000          ;or other MLID
        ODINSUP
        NETBIND
        NET START
        CALL INITFSI.BAT
        IPXODI
        NETX
        TCPIP
        LANWP
```

The example file above shows NE2000 as the MLID being loaded after the LSL. The MLID you would load (NE2000, 3C523, and so on) depends on the particular network interface card in the workstation. ODINSUP is loaded next.

NETBIND binds the DXME0MOD driver to ODINSUP according to the binding statement in the PROTOCOL.INI file, which must also be modified slightly (as will be explained). NET START loads the DOS LAN Requester, and INITFSI.BAT initializes the full screen interface.

At this point, you finish loading the ODI stacks: IPXODI and NETX, then any others you'll need (TCPIP and LANWP, for example).

In the past, both IBM and Novell recommended that users log on to LAN Server first, then log in to NetWare. This is a good rule of thumb to avoid having NetWare drives excluded from the PATH environment variable.

PROTOCOL.INI File Changes. In the PROTOCOL.INI file, you must remove the NDIS MAC driver completely and change the binding statement to:

```
[DXME0$]
        BINDINGS=NE2000
```

Of course, you would replace NE2000 with whatever MLID you will be using.

The DXME0MOD driver is logically bound to ODINSUP. However, due to the nature of NDIS and NETBIND, the MLID name is used instead of ODINSUP in the PROTOCOL.INI file. This is because ODINSUP has no knowledge of the MAC layer below, which is needed to complete the binding process.

It is necessary to replace all occurrences of the old NDIS MAC driver with the new MLID. In this case, DXME0MOD will probably be the only one, but that doesn't mean that other NDIS protocol stacks can't be loaded.

NET.CFG File Changes. In the Ethernet environment, ODINSUP requires all of the following frame types to be loaded:

- ETHERNET_II
- ETHERNET_SNAP

- ETHERNET_802.2
- ETHERNET_802.3 (optional for backwards compatibility with NetWare v2.x)

Also, the LSL Buffer Size for Ethernet must be at least 1514 bytes (the maximum packet size).

Token Ring hardware requires both the TOKEN-RING and TOKEN-RING_SNAP frame types. The LSL Buffer Size must be at least 4210 bytes to accommodate a 4KB packet.

DOS Client Interoperability Issues

Heterogeneous network computing is not without its caveats. Once you have set up and installed any of the dual requester environments described in this AppNote, there are a few user tips to be aware of. Because the NetWare shell assumes it is the only redirector loaded, resource contention problems can occur when the same drives or printers are redirected for both NetWare and LAN Server. Two such problems are explained below.

Mapping Over a LAN Server Drive

As explained earlier, the LASTDRIVE statement in CONFIG.SYS serves as the logical boundary between LAN Server drives and NetWare drive mappings. However, NetWare's MAP command will let you overwrite a local drive mapping. NetWare considers all drives that come before the drive letter indicated in the LASTDRIVE statement as local drives, even if they are being used for LAN Server drives.

For example, suppose a dual requester client has the LASTDRIVE=R statement in its CONFIG.SYS file, giving NetWare drives S through Z. The user then issues the following commands:

```
NET USE G:=LAN_SERVER/VOL1:HOME <Enter>
        MAP G:=NW311/SYS:RESEARCH <Enter>
```

The NetWare MAP command will indicate that drive G is mapped to a local drive and ask the user to confirm the overwrite. If the user answers Yes, NetWare will map over the LAN Server drive and drive G now points to a NetWare drive. Because the NetWare shell hooks INT 21, it sees any read or write requests first, before DOS or the DOS LAN Requester. The LAN Server drive that was mapped to drive G will not be seen again unless the user enters a NetWare "MAP DEL G:" command.

The NetWare MAP command doesn't show LAN Server drives in its on-screen display. However, the NET USE command will show both the LAN Server and NetWare drives.

Ending CAPTURE with NET USE

The other problem is with the CAPTURE command. You can have LPT1 redirected to a NetWare queue and also have LPT1 redirected to a LAN Server queue. However, whichever command is entered last wins.

For example, suppose a dual requester client types the following commands to redirect LPT1 output:

```
  CAPTURE LPT1 /S=NetWare_Server /Q=Laser_Tag <Enter>
          NET USE LPT1 A=LAN-SERVER/VOL1:HOME/LASER-JET <Enter>
```

In this case, LAN Server gets the LPT1 redirection because both the shell and the DLR hook INT 17 for print requests, and the NET USE command was the most recent command issued.

The problem occurs when the user wants to end the LPT1 redirection and types the NET USE LPT1 DELETE command. Because the NET USE command issues an INT 21 service, the NetWare shell assumes the command is meant for it and kills the NetWare capture (which in this case has nothing captured). A second NET USE LPT1 DELETE command actually gets to DOS and is redirected to LAN Server via INT 2F, which ends the LAN Server redirection of LPT1.

<u>Note:</u> The NetWare v4.0 shell will use the DOS Resource and Drive Tables to insure interoperability. This will resolve both of these resource sharing problems. To accommodate these changes, the new NetWare shell will require LASTDRIVE=Z in the CONFIG.SYS file.

Summary

This AppNote has described three dual requester configurations for DOS workstations that require access to both LAN Server and NetWare server resources. The ODINSUP solution provides the greatest flexibility and brings the full benefits of ODI to the dual requester client, while also interoperating with NDIS protocol stacks.

Part 2 of this series will cover interoperability issues for OS/2-based workstations.