NOVEMBER 1992

NOVELL® RESEARCH

NetWare and LAN Server Client Interoperability via ODINSUP: Part 2

Bryan R. Clark
Systems Engineer
Systems Engineering Division

Ken Nefl
Principal Technical Writer
Systems Engineering Division

This AppNote covers various OS/2 dual requester configurations and gives examples of the corresponding CONFIG.SYS, NET.CFG, and PROTOCOL.INI files. It gives a historical perspective to show the evolution of OS/2 client interoperability, and details various configurations beginning with OS/2 v1.0 Standard Edition and ending with dual requester configurations in OS/2 v1.3 with EE and with ES and LS components. This is the second in a series on NetWare client interoperability.

Contents

Introduction

This AppNote, the second in a series on NetWare client interoperability, covers OS/2 client integration. As in the previous AppNote (see NetWare and LAN Server Client Interoperability via ODINSUP: Part 1 in the September 1992 NetWare Application Notes), this AppNote covers various OS/2 configurations and gives examples of the corresponding CONFIG.SYS, NET.CFG, and PROTOCOL.INI files.

This installment gives a historical perspective to show the evolution of OS/2 client interoperability, and details various configurations beginning with OS/2 v1.0 Standard Edition and ending with dual requester configurations in OS/2 v1.3 with EE and with ES and LS components. The third AppNote in the series will cover OS/2 v2.0 dual requester configurations.

This information will benefit developers, systems integrators, consultants, technical support personnel, and network supervisors working in multivendor OS/2 environments. Although the discussion and examples are geared toward interoperability between NetWare and IBM's LAN Server, all NDIS v1.02-based drivers and protocol stacks can be loaded in dual requester configurations with the NetWare Requester for OS/2.

Background on OS/2 Client Dual Requesters

In 1987, IBM and Microsoft introduced Operating System/2 (OS/2) Standard Edition, a 16-bit operating system designed to overcome many of the limitations of DOS. Due to its improved memory management and multitasking capabilities, this new platform was immediately targeted for mission-critical applications.

Starting with the first release of OS/2, a NetWare Requester for OS/2 has always been included in the box. This demonstrates Novell's commitment to support all client platforms from Apple to UNIX and everything between. Although OS/2 Standard Edition didn't support dual requesters, we will use this as a reference point for later configurations.

The release of OS/2 v1.3 and the Extended Edition (EE) added the Communications Manager and Database Manager. An OS/2 LAN Requester was also included to communicate with LAN Server. This was the first complete configuration able to support dual requesters on the same LAN adapter. The examples in this AppNote will detail configurations based on OS/2 v1.3.

The new OS/2 v2.0 includes many network-aware features built in to the desktop. These changes will be outlined in a later AppNote. However, we will concentrate on new Extended Edition (EE) components. EE has been rebundled to include new drivers for NetBEUI and TCP/IP. These new products, Extended Services (ES) and LAN Services (LS), include support for DOS and OS/2 clients, along with the server applications.

OS/2 Architecture

To provide a foundation for discussing interoperability, let's first examine the technology platform and architecture of OS/2. OS/2 was designed to be a multitasking, dynamically extensible, virtual machine operating system. It was built specifically for the Intel 80286 platform and uses the hardware features of a 16-bit data path and 16 megabytes of addressable memory. Memory segmentation and memory protection were also inherent to the 286 CPU. The basic OS/2 kernel is illustrated in Figure 1.

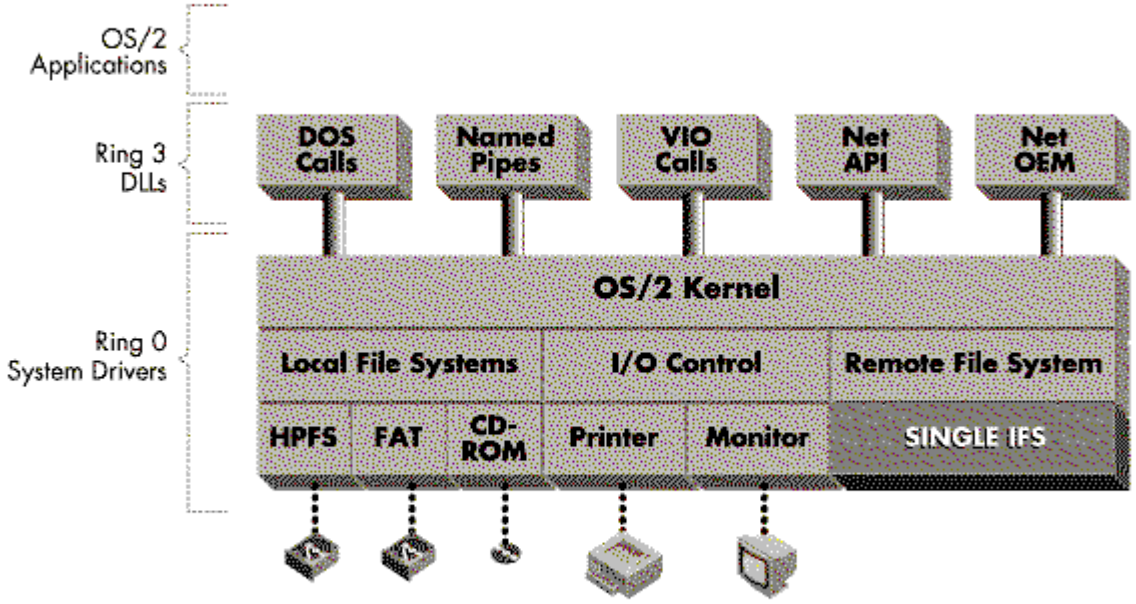

Figure 1: Architecture of the OS/2 kernel.

Intel's ring protection is basically an onion skin layered model, with applications on the outside (ring 3) and the operating system and drivers in the middle (ring 0). The physical hardware would be in the center of this model. The outer rings are very restrictive and prevent the direct access to the hardware except through specified APIs. Instead of issuing an interrupt as in DOS, OS/2 applications make a far call to the API, which inspects the command for authorization and integrity before relaying it to the operating system or proper device driver.

All hardware devices must load the appropriate drivers at boot time from the CONFIG.SYS file. Device drivers are named with the .SYS file extension. Device drivers call a special set of IOCTL (I/O ConTroL) APIs to register interrupts, shared memory segments, and DMA channels with the OS/2 kernel. This allows OS/2 to share these devices with all tasks.

Dynamic Link Libraries (DLLs) are loaded on demand when called from an application or by the OS/2 kernel itself. A DLL is a special type of .EXE file that also acts as the API interface between applications and the OS or drivers. This abstraction allows for applications to be independent from hardware and also to multiplex many requests from multiple tasks.

OS/2 uses universal naming convention (UNC) to extend the file system beyond the local desktop. A UNC is a fully qualified path denoted by the beginning double backslash. For example, to create a UNC, you would first create a string with "\\" followed by the file server name. Next, add a single backslash and the volume name followed by a colon. Then add the rest of the directory path just as you would in DOS:

\\file_server\volume:dir1\dir2\...\filename

Networking with OS/2 v1.0 Standard Edition

The NETAPI and NETOEM DLLs were the first extensions to OS/2 specifically for networking. These were available only from IBM as part of Extended Edition. To support the NetWare Requester on OS/2 Standard Edition, Novell provided their own version of these DLLs. Thus users did not have to buy EE just to use the NetWare Requester for OS/2.

Figure 2 shows the basic configuration for running the NetWare Requester with OS/2 v1.0. Note that the NetWare Requester for OS/2 was designed to use Novell's Open Data-Link Interface (ODI) driver specification, which includes IPX.SYS, the Link Support Layer (LSL.SYS), and the LAN driver (TOKEN.SYS).

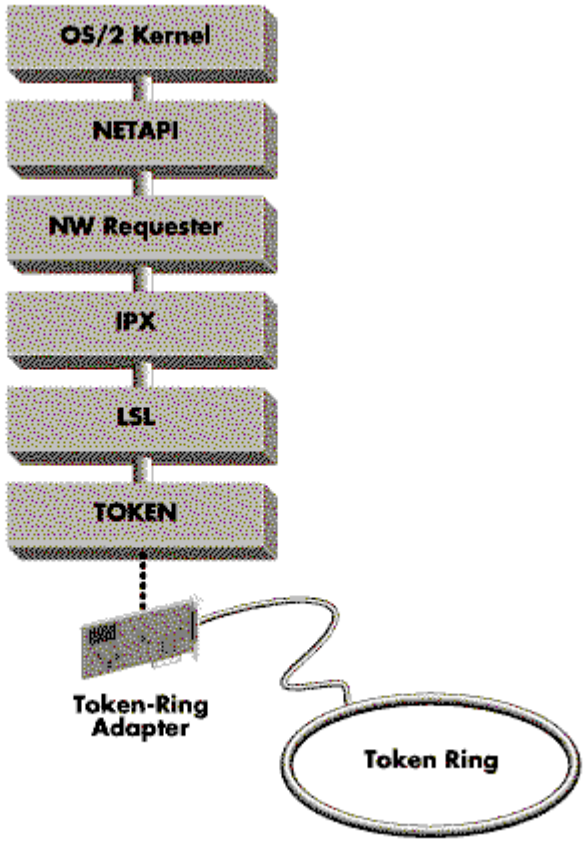

Figure 2: Networking with OS/2 v1.0 Standard Edition.

OS/2 v1.3 and Extended Edition

The three elements necessary for dual requesters to run in an OS/2 client include:

- OS/2 v1.3 with CSD 5015 applied
- EE Communications Manager
- NetWare Requester for OS/2 v1.3F

With the release of OS/2 v1.3, the file system was enhanced to allowed for a remote or installable file system (IFS). The file system was also modified to use Extended Attributes (EA), including long names up to 254 characters. Through the IFS, the OS/2 file system can now support High Performance File System (HPFS), CD-ROM drives, and remote file systems such as NetWare. This enables each requester to register its own IFS and look at the UNC requests as they are sent through the chain.

From a communications perspective, OS/2 v1.3 had a problem in that only one remote file system could be loaded at a time. IBM assembled a fix for this and several other problems into Corrective Service Diskette (CSD) 5015. Subsequent CSD releases (5016 and 5050) also include this fix.

Now that multiple IFSs can use the remote file system, OS/2 v1.3 is capable of loading dual requesters. Figure 3 shows the changes incorporated into the OS/2 kernel.

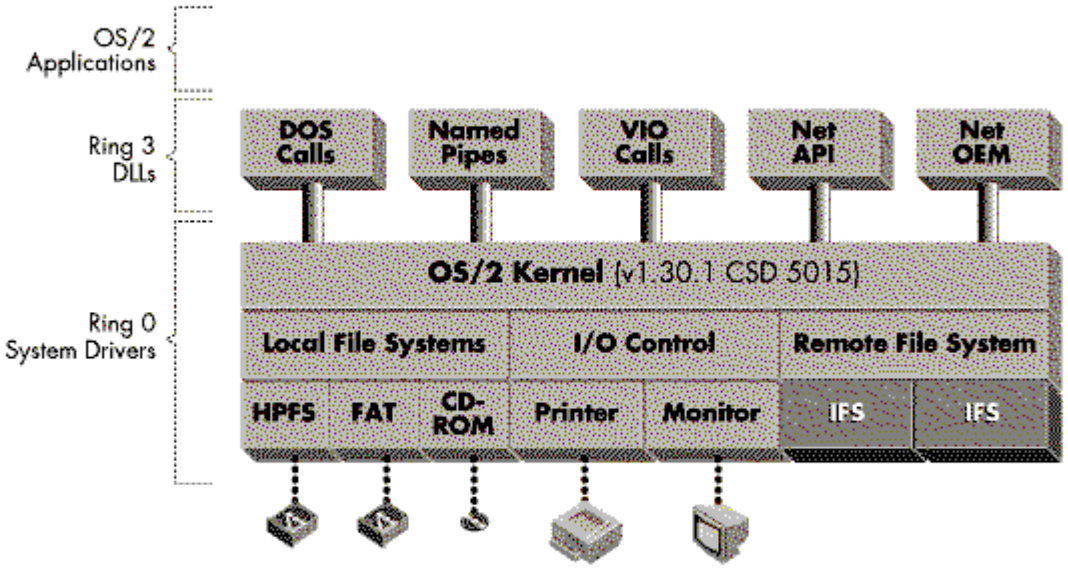

Figure 3: OS/2 v1.3 kernel with the CSD 5015 update.

Communication Manager Components

IBM's Communications Manager (CM) is modeled very similar to the IBM DOS LAN Support Program (discussed in Part 1). The components of CM are diagrammed in Figure 4.

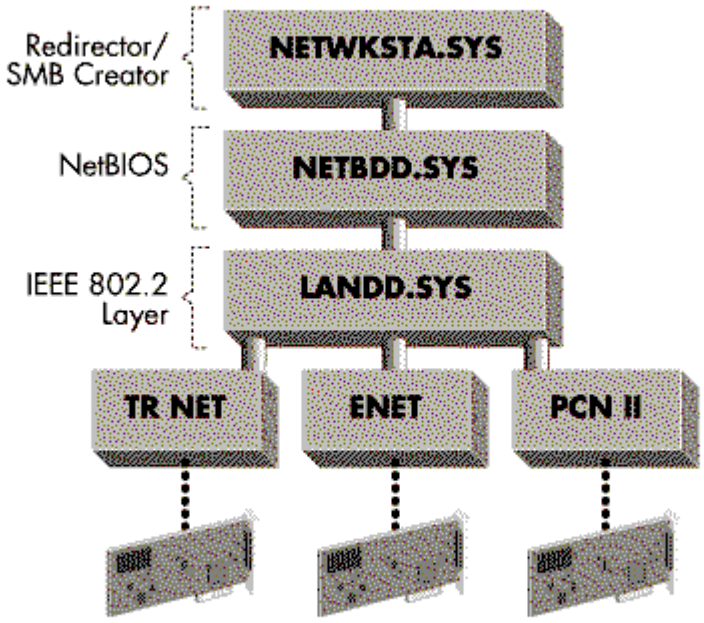

Figure 4: IBM Communications Manager (CM) components.

The CM driver, called LANDD.SYS, is the middle module between the protocol stacks and the actual LAN driver. (The DD at the end of each name stands for Device Driver). The NetBIOS protocol stack, called NETBDD.SYS, sits on top of the CM, linking it to the LAN Server Requester (NETWKSTA.SYS). The NETWKSTA (NET WorKSTAtion) driver is actually IBM's LAN Server IFS and requester combined. The CM includes device drivers for Token Ring, PC Network II, and Ethernet adapters from IBM.

NetWare Requester and NSD 004

The NetWare Requester uses the Open Data-Link Interface (ODI) architecture. The familiar LSL and IPX protocol stacks are divided from the LAN drivers. The supported LAN topologies are Token Ring, PC Network II, Ethernet, and RX-Net. The NetWare requester supports the IPX/SPX, NetBIOS, and Named Pipes (NP) protocol stacks.

The NetWare Requester itself is actually two modules: NWIFS and NWREQ, as shown in Figure 5. The NWIFS claims requests from OS/2 and the NWREQ creates the NCP packets necessary to send these requests to the server. The design of OS/2 allows for a traditional UNIX approach of using daemons, or monitors for background tasks. The NetWare requester comes with four daemons, one for each supported protocol:

- NWdaemon for IPX and the NetWare Requester

- SPdaemon for SPX
- NBdaemon for NetBIOS
- NPdaemon for Named Pipes

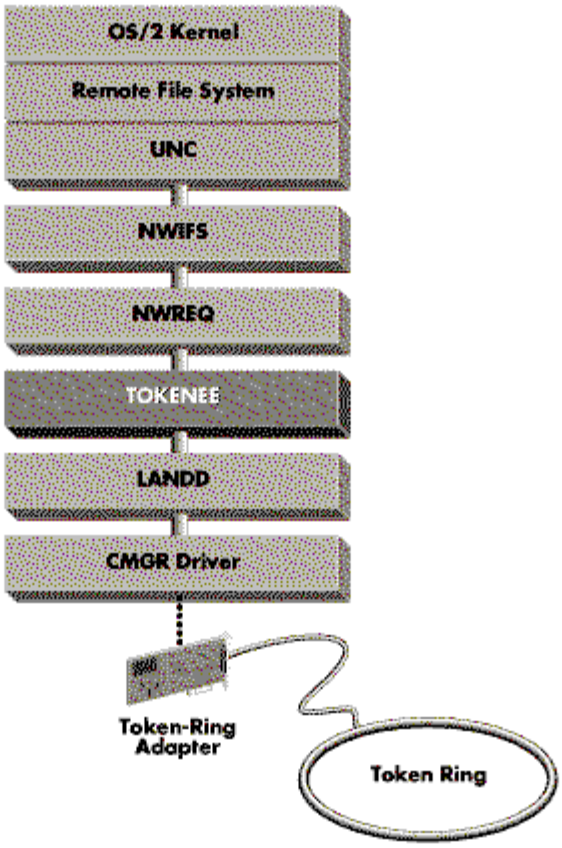

Figure 5: Components of the NetWare Requester for OS/2.

The NetWare Requester for OS/2 includes a Presentation Manager-based Install program that automatically copies the files to the workstation's hard drive and updates the CONFIG.SYS file. Several utilities are bundled into the NWTOOLS, which help the user attach to servers, map network drives, and set up printers from the desktop.

To configure dual requesters, the NetWare requester needs to have Novell Service Diskette (NSD) 004 applied. This diskette contains a new NWIFS and updated drivers, including the ODINSUP module. The NSD Install utility will copy the new drivers and update the CONFIG.SYS file.

TOKENEE Configuration

The TOKENEE driver shown in Figure 5 was first released with the OS/2 Requester v1.3. It is similar to the DOS LANSUP driver discussed in Part 1. TOKENEE uses the Direct interface to the LANDD or Communications Manager. It is limited to Token Ring (802.5) hardware and it does not support dual requesters.

With NSD004, Novell created a new driver called CMGRLAN to support all CM topologies. This driver also uses the Direct interface, but it does support dual requesters.

Dual Requesters with OS/2 v1.3

With OS/2 v1.3, there are two basic ways to configure the protocol stacks for dual requesters:

- **CMGRLAN.SYS.** In this configuration, the NetWare Requester sits on top of the CMGRLAN driver; the CM LANDD.SYS driver owns the LAN adapter.
- **ODINSUP.SYS.** In this configuration, NDIS drivers sit on top of ODINSUP; an MLID driver owns the adapter.

Although both of these drivers use ODI to support the NetWare Requester, only ODINSUP uses MLID drivers to own the LAN adapter. This configuration allows the user all of the benefits of CM/EE applications along with ODI's performance and flexibility. In configurations where there is no ODI MLID driver available and several NDIS-based applications are being loaded, the CMGRLAN driver should be used.

CMGRLAN.SYS and Token Ring

The CMGRLAN driver uses the Direct interface to LANDD and supports all Communications Manager device drivers (Token Ring, PC Network II, and Ethernet). For simplicity, we'll start with the Token Ring driver, which is not an NDIS driver. This configuration is diagrammed in Figure 6.

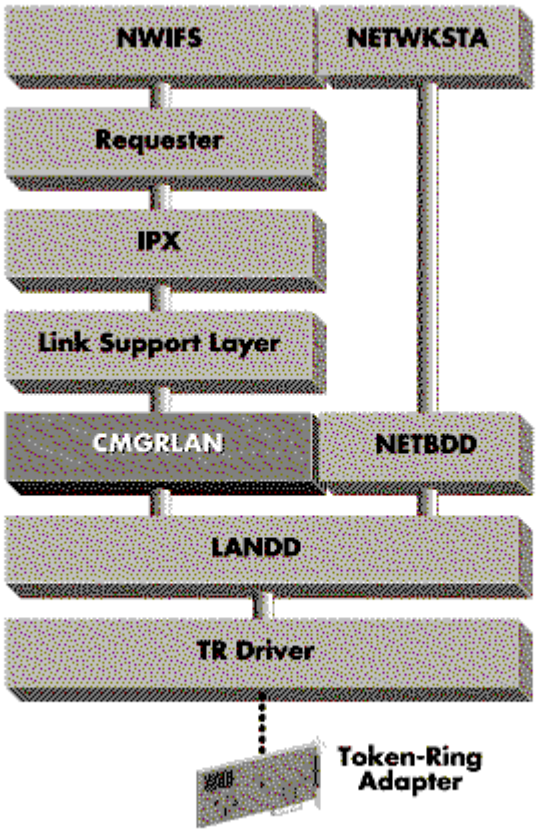

Figure 6: The CMGRLAN.SYS and Token Ring configuration.

The steps for installing this dual requester configuration with CMGRLAN are:

1. Install OS/2 v1.3 with the EE Communications Manager.

2. Run the NetWare Requester for OS/2 INSTALL program. Specify the directory c:\netware to copy the files. Configure the requester by selecting the CMGRLAN driver, which will be added to the CONFIG.SYS. (In the example below, the SPX and Named Pipes options were also selected.)

Note: If the Install program does not run, remark out the NetBIOS driver line:

REM device= c:\cmlib\netbdd.sys ...

Then reboot the machine and rerun the Install program. Remember to undo the remark after the installation is completed.

3. Run the NetWare Requester NDSINST utility to update the drivers.

4. Edit the CONFIG.SYS and add L:\OS2 to the DPATH.

CONFIG.SYS File Changes. The NetWare Requester INSTALL program automatically modifies the CONFIG.SYS file. The CM Install program should also update the LIBPATH statement in the CONFIG.SYS file to include c:\muglib\dll and place it in the string before c:\netware. To eliminate any further chance of

NETAPI and NETOEM DLL conflicts, delete these DLLs from the c:\netware directory.

A sample CONFIG.SYS listing for a Token Ring setup is given below. (Note that some lines, such as the OS/2 preamble section and the NetWare deamons, have been omitted from this listing.)

```
device=c:\cmlib\landd.sys
        device=c:\cmlib\token.sys
        device=c:\cmlib\netbdd.sys
        device=c:\netware\lsl.sys
        device=c:\netware\cmgrlan.sys
        device=c:\netware\ipx.sys
        run=c:\netware\ddeamon.exe
        device=c:\netware\spx.sys
        device=c:\netware\nwreq.sys
        device=c:\netware\nmpipe.sys
        device=c:\netware\npserver.sys
        run=c:\netware\npdeamon.exe np_serv_name
        ifs=c:\netware\nwifs.ifs
        ifs=c:\cmlib\netwksta.sys
```

Note that the Novell NWIFS is loaded before the IBM NETWKSTA IFS. There are two reasons for doing this:

- The IBM IFS issues a broadcast to resolve unknown file server names, and thus does not chain the request until the NetBIOS timeout has expired.
- If there are duplicate names with NetWare servers and LAN Servers, the NWIFS will claim the request and the NWREQ will establish the connection quicker.

If you need source routing support for Token Ring or PCN2 networks, the NetWare ROUTE.SYS driver must be loaded.

NDIS and Etherand

The Network Driver Interface Specification (NDIS) and Open Data-Link Interface (ODI) are the two most prominent driver standards in the LAN industry. ETHERAND is the first NDIS-based driver that IBM has released for both the DOS LAN Support Program and the EE Communications Manager.

The basic concepts behind NDIS and the binding of protocol stacks in OS/2 are the same as in the DOS environment. Briefly, PROTMAN (the PROTocol MANager) acts as the glue layer between the MAC drivers and the protocol stacks. The NETBIND utility reads the PROTOCOL.INI file, exchanges information between the drivers and protocols, and binds them together. (For more details, see NetWare and LAN Server Client Interoperability via ODINSUP: Part 1 in the September 1992 NetWare Application Notes. Other reference material is listed at the end of this AppNote.)

If both an ODI and an NDIS driver are available for the adapter, the next factors to consider are the workload characteristics. This should include the bandwidth of the LAN adapter, performance of applications, and data access, along with the amount of time spent using each applications. For example, a user who does mostly word processing on a NetWare server could take advantage of the increased performance of ODI. However, a user who spends 80 percent of the time in a 3270 emulator running on NDIS might notice the delay while doing file transfers.

CMGRLAN.SYS and Etherand

The only NDIS driver for OS/2 v1.3, namely ETHERAND, can also be configured with the CMGRLAN driver. However, a few functions are limited or not supported.

The first limitation is that the LANDD driver does not correctly pass Ethernet 802.3 or Ethernet DIX v2 frame types. Only the Ethernet 802.2 frame type is supported. Second, since CMGRLAN uses the Direct interface, performance is limited to the speed of LANDD.

To install CMGRLAN with ETHERAND, follow the same instructions outlined above for Token Ring, only substitute ETHERAND for Token Ring. Notice that the EE install program will also create the necessary PROTOCOL.INI file. The ETHERAND driver, along with the PROTMAN and NETBIND statements, will be added automatically to your CONFIG.SYS. Figure 7 illustrates this configuration.

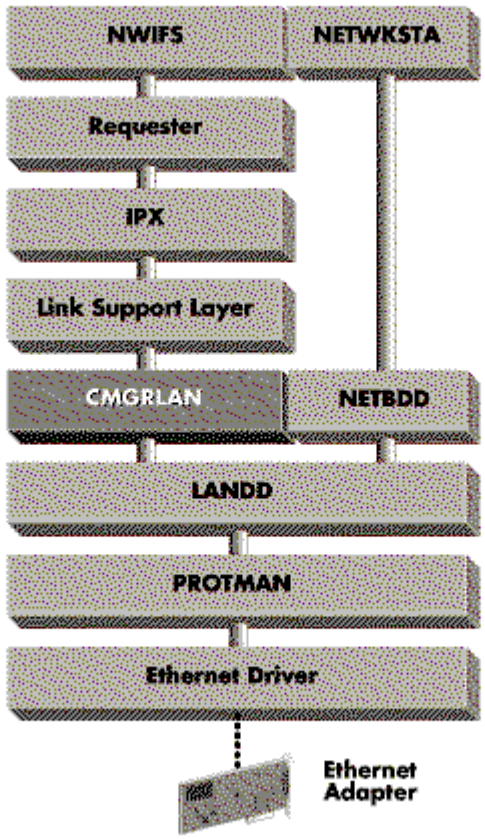

Figure 7: The CMGRLAN and ETHERAND configuration on OS/2 v1.3.

ODINSUP for OS/2 v1.3

Previously, IBM's drivers used a proprietary interface to communicate with the hardware. Novell supports this interface with the CMGRLAN and the LANSUP drivers. However, these drivers are limited to 802.2 frame types and performance is restricted by the corresponding CM hardware drivers.

To increase performance and interoperability, Novell introduced the ODINSUP (ODI under and NDIS UP) driver or shim to allow ODI and NDIS protocols to share the same LAN adapter. Since there is not a one-to-one correspondence between NDIS MAC functions and the ODINSUP interface, a few of the functions are emulated or not supported. Protocol stacks using asynchronous interrupts via the MAC Interrupt Request call will be limited to one every 32 milliseconds due to software emulation. Also, card address overrides cannot be done on the fly, but they can be done in the NET.CFG file.

One of the biggest benefits of ODINSUP is increased performance, because the ODI MLID owns the adapter. In Novell Systems Engineering tests, ODINSUP has shown an increase in throughput of between 200% to 300%, depending on topology and LAN adapter. Considering that the NDIS stack performance decreases only 5% to 8% when placed on top of ODINSUP, this is a great improvement.

With ODINSUP, both the ODI protocols and the NDIS stacks can use multiple frame types simultaneously. The Ethernet 802.3, DIX v2, and SNAP frame types need to be loaded for ODINSUP. Ethernet 802.3 frame type support is optional.

ODINSUP requires the use of an ODI MLID to own the LAN adapter. Also, the LSL v1.1 (or newer) should be used. If the MLID from the LAN adapter manufacturer does not work, request an updated driver. The new driver should include Media Support Module (MSM) v1.1, which Novell provided to developers in September of 1991.

ODINSUP and Ethernet

Figure 8 illustrates the use of ODINSUP.SYS with Ethernet hardware.

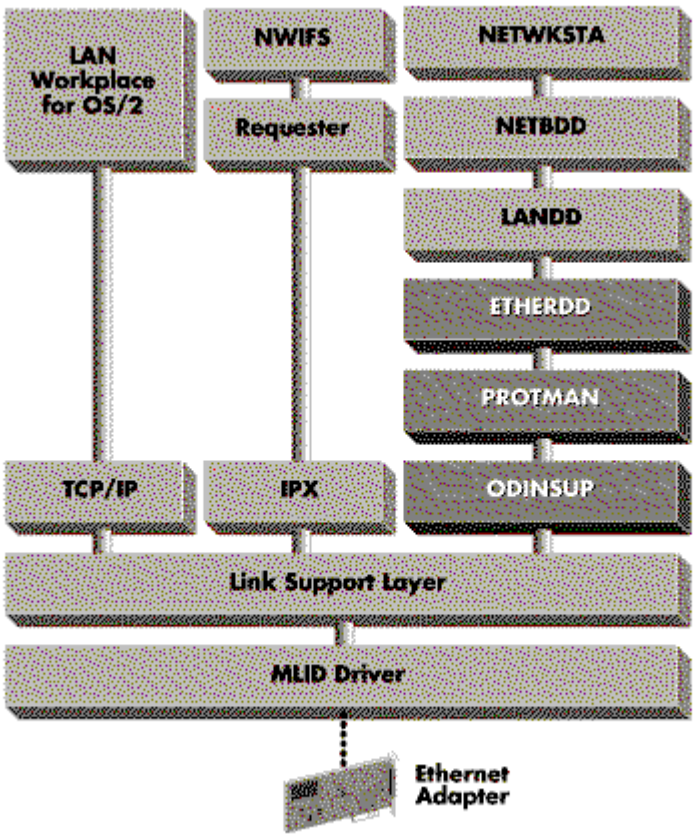

Figure 8: The ODINSUP.SYS and Ethernet configuration.

To configure ODINSUP and Etherand for dual requesters, follow the same instructions as for CMGRLAN and Etherand above, substituting ODINSUP as the driver in the NDSINST program. Note that NDIS's PROTMAN and NETBIND are still required. The following steps will finish the installation:

1. Run the NDSINST program, selecting the ODINSUP driver.

2. Edit the CONFIG.SYS to remark out the NDIS MAC driver.

3. Edit the NET.CFG to include all the Ethernet frame types. (Ethernet 802.3 is optional, unless the workstation will attach to a NetWare v2.2 file server.) Also specify the maximum LSL buffer size for Ethernet (1514 bytes).

4. Edit the PROTOCOL.INI file to change all protocol bindings that reference the NDIS MAC driver to the ODI MLID.

CONFIG.SYS File Changes. The most significant change to the CONFIG.SYS file is to remark out the NDIS MAC driver (the ELNKII driver in the example below). Next load the LSL, the ODI MLID, and ODINSUP. Note that the PROTMAN and NETBIND statements are still necessary and that PROTMAN and ODINSUP must be loaded (in that order) before NETBIND can be executed. Then load the other NetWare Requester modules and finish with the IBM LAN requester.

```
device=c:\cmlib\landd.sys
        device=c:\cmlib\protman.os2
        rem device=c:\cmlib\elnkii.os2
        device=c:\cmlib\etherdd.sys

device=c:\netware\lsl.sys
        device=c:\netware\3c503.sys
        device=c:\netware\odinsup.sys

run=c:\cmlib\acsepsys.exe
        run=c:\cmlib\netbind.exe
        device=c:\cmlib\netbdd.sys cfg=...

device=c:\netware\ipx.sys
        device=c:\netware\nwreq.sys
        ifs=c:\netware\nwifs.ifs

device=c:\ibmlan\netprog\rdrhelp.sys
        ifs=c:\ibmlan\netprog\netwksta.sys
```

NET.CFG File Changes. The sample NET.CFG listing below shows the modifications necessary to support all Ethernet frame types (including Ethernet 802.3), and the LSL buffer size set for 1514 bytes (the maximum size for Ethernet).

```
Protocol ODINSUP
        bind 3C503      ;bind to first 3C503

Link Support
        buffers 10 1514     ;maximum ethernet frame size

Link Driver 3C503
        frame Ethernet_802.2;NetBIOS, IPXODI and SNA
        frame Ethernet_II   ;TCP/IP, OSI
        frame Ethernet_SNAP ;AppleTalk, DECnet
        frame Ethernet_802.3;optional NetWare v2.2 IPX
```

PROTOCOL.INI File Changes. The changes to the PROTOCOL.INI file are the most involved to actually complete. With an ASCII text editor, find the [ETHERAND] section and change the reference to the NDIS MAC driver to the ODI MLID. In the example below, the ELNKII binding is remarked out by placing a semicolon (;) in front of that line. Note in the example that the new Bindings = x3C503 statement has an x to

denote all strings that begin with a number.

At the end of the PROTOCOL.INI file, add a section header for the ODI MLID driver. A section header is denoted by square brackets "[ ]" around the protocol or driver name. In the example, the header "[x3C503]" was added. Note that even though no information is given under the header, the header must be created for NETBIND to run successfully. The MLID configuration and binding information is obtained either in the NET.CFG file or from the driver defaults.

```
[PROT_MAN]
        DriverName = PROTMAN$

[ETHERAND]
        DriverName = OS2EE$
        Bindings = x3C503
        ;Bindings = ELNKII

[ELNKMC]
        DriverName = ELNKII$
        Interrupt = 3
        IOAddress = 0x300

[x3C503]
```

An obvious question remains: "If I'm using ODINSUP to talk to the PROTMAN driver, why do I put the MLID name in the bindings statement?" The answer involves the nature of the NDIS NETBIND utility and the exchange of MAC information with the protocol stacks. Since ODINSUP is a protocol stack itself, it doesn't know any MAC or hardware specifics. However, it does know how to query that information from the NET.CFG file and the MLID by making the proper ODI calls.

Extended Services and LAN Services

An updated OS/2 Extended Edition is now available that includes the new NDIS drivers. The new EE has been divided into Extended Services (ES) and LAN Services (LS).

ES consists of the Communications Manager (CM), Database Manager (DM), and the new LAN Transport components. The ES Communications Manager (CM) has improved EHLLAPI support, dynamic allocation from a LU pool, and LAN over coax.

The obvious changes in LAN Transports are the new NDIS protocol stacks and the NDIS MAC drivers, including some new layers. The 802.2 Logical Link Control (LLC) information has been taken out of LANDD and is now called LANDLL. The NetBIOS driver was also divided into several pieces and rewritten to run at Ring 0 to increase performance. The new NDIS protocol stack, NetBEUI, is nicknamed JetBEUI.

LS consists of LAN Server and the new DOS and OS/2 LAN requesters. The LAN requesters use the new NDIS LAN Transport drivers.

The installation program for CM now offers manufacturers the ability to create Network Information Files (NIF) to provide LAN adapter information used to create the PROTOCOL.INI file. This will reduce the editing that a user must do to complete the configuration and installation of LAN adapters.

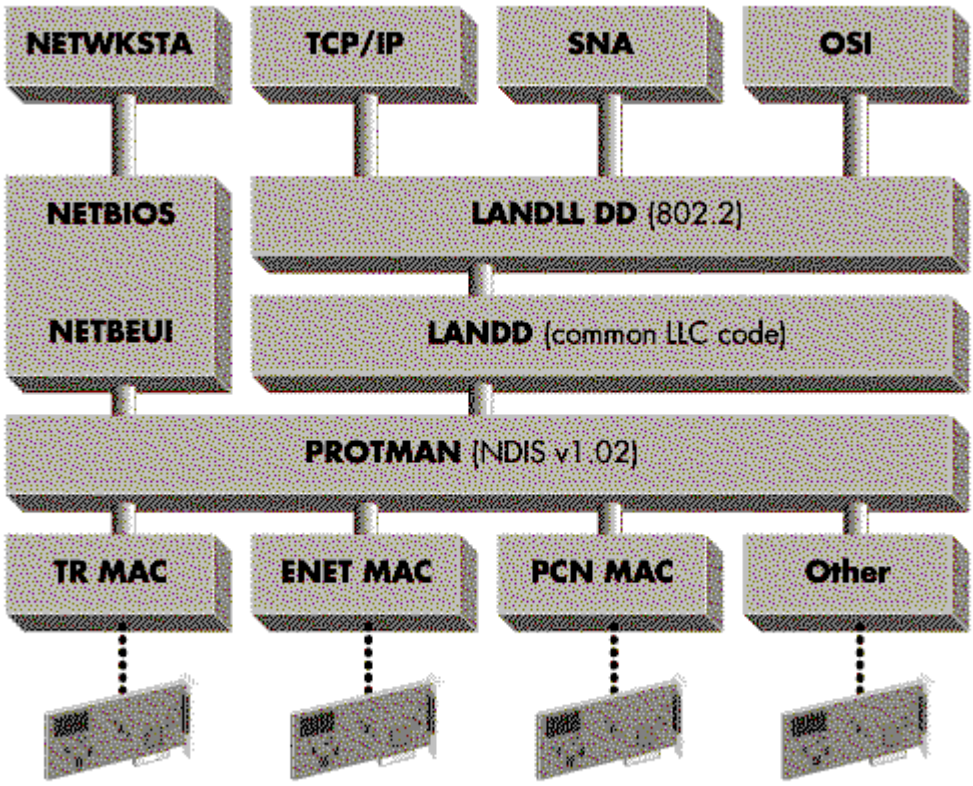

Figure 9: OS/2's ES and LS components.

To run the new ES and LS components on OS/2 v1.3, CSD 5050 must be applied to create OS/2 v1.30.2. (These components and identical configurations will also work with OS/2 v2.0.)

LANSUP and ES/LS

The following configuration illustrates how LANSUP can be configured when both a Token Ring and an Ethernet adapter are installed with the Communications Manager. This configuration could be used to run the CM SNA gateway and the NetWare Requester with a Named Pipes Server in the same workstation. Obviously, it is important to know on which LAN the NetWare server resides, as well as any other servers or hosts.

In this example, we will use a configuration where the NetWare server, LAN server, and Named Pipes clients exist on the Ethernet segment, while the IBM host is on the Token Ring segment (see Figure 10).

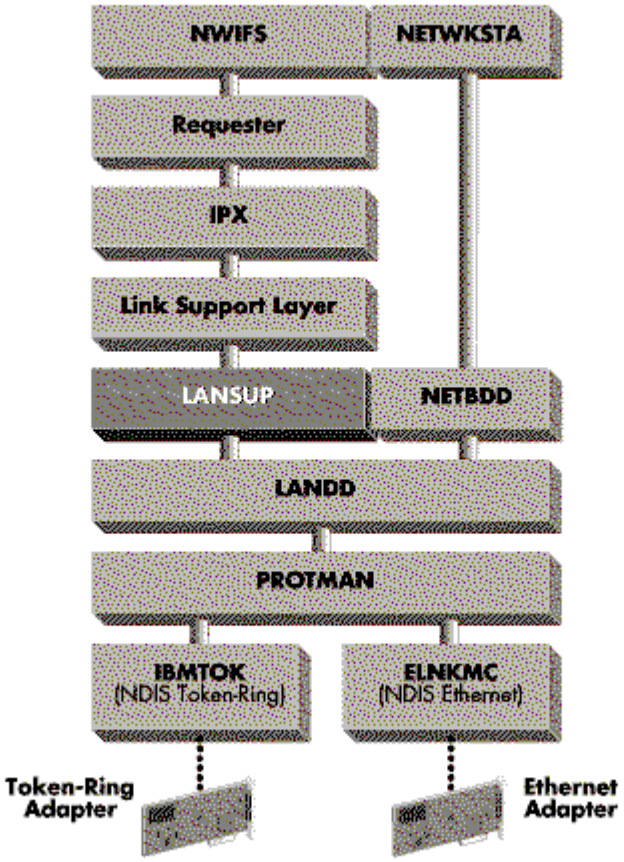

Figure 10: LANSUP configuration with both Token Ring and Ethernet.

In this diagram, note that the NDIS drivers own the LAN adapters and the Protocol Manager will the bind or link the proper stacks to the MAC drivers. These relationships will be clarified when we look at the sample PROTOCOL.INI file.

The steps for installing dual requesters with dual LAN adapters using the LANSUP driver are:

1. Install OS/2 v1.3 with the EE Communications Manager.

2. Install the CSD 5050 update.

3. Install the CM SNA gateway. (Since this is not a trivial process and the scope of this article is limited to interoperability, we will not detail the SNA gateway installation. Refer to the IBM documentation to get the SNA gateway and the IBM LAN Requester properly configured and running. Once that is done, you can proceed to install the NetWare requester.)

4. Edit the PROTOCOL.INI file and match the LANDD stack to both the Token Ring driver and ELNKII drivers. This will allow SNA traffic to flow from the Ethernet segment to the Token Ring segment. NetBEUI will then be linked to the ELNKII driver.

5. At this point, logon to the LAN Server and make sure that the SNA gateway is working properly.

6. Run the NetWare Requester for OS/2 INSTALL program and select the CMGRLAN?? driver. Also

select SPX and Named Pipes Server support and enter the NP server name.

7. Run the NetWare Requester NDSINST utility to update the drivers.

8. Edit the CONFIG.SYS file and add L:\OS2 to the DPATH.

CONFIG.SYS File Changes. The CONFIG.SYS file is not much different from the previous example of CMGRLAN, except that there are several new driver layers and the name for the NetWare driver changed to LANSUP. The CM drivers are loaded before the NetWare drivers. In the listing below, all of the lines for the CM and IBM LAN Requester and NetWare Requester are given. Only the OS/2 preamble is omitted.

```
device=c:\ibmcom\lanmsgdd.os2 ...
        device=c:\ibmcom\protman.os2 ...
        device=c:\ibmcom\protocol\landd.os2
        device=c:\ibmcom\protocol\lanlldd.os2
        device=c:\ibmcom\protocol\netbeui.os2
        device=c:\ibmcom\protocol\netbios.os2
        run=c:\ibmcom\protocol\landll.exe
        run=c:\ibmcom\protocol\netbind.exe
        device=c:\ibmcom\protocol\netbeui.os2 ...
        device=c:\ibmcom\protocol\netbios.os2 ...
        device=c:\ibmlan\netprog\rdrhelp.sys
        device=c:\ibmcom\protocol\elnkii.os2
        device=c:\ibmcom\protocol\ibmtok.os2

device=c:\netware\lsl.sys
        run=c:\netware\ddaemon.exe
        device=c:\netware\lansup.sys
        device=c:\netware\ipx.sys
        device=c:\netware\spx.sys
        run=c:\netware\spdaemon.exe
        device=c:\netware\nmpipe.sys
        device=c:\netware\npserver.sys
        run=c:\netware\npdaemon.exe  np-serv-name
        device=c:\netware\nwreq.sys
        ifs=c:\netware\nwifs.ifs
        run=c:\netware\nwdaemon.exe
        device=c:\netware\netbios.sys
        run=c:\netware\nbdaemon.exe
        run=c:\netware\nwspool.exe

ifs=c:\ibmlan\netprog\netwksta.sys
```

Note that the IBM IFS is still loaded last. However, this is not as critical with the CSD 5050 and the newest IBM IFS. If desired, you can move the IBM IFS up with the rest of the CM drivers.

PROTOCOL.INI File Changes. The PROTOCOL.INI file has changed significantly--note especially the use of the Network Information File (NIF) and the new NetBEUI drivers. In this example, the LANDD driver is bound to both LAN segments and NetBEUI is bound to the Ethernet LAN segment. For purposes of illustration, we have omitted many of the lines within each of the individual sections. The complete PROTOCOL.INI will be built by the CM and LAN Requester installation programs.

```
[PROT_MAN]
        DriverName = PROTMAN$

[IBMLXCFG]
        LANDD_nif = LANDD.nif
```

```
        NETBEUI_nif = NETBEUI.nif

;********** PROTOCOL SECTION **********

[LANDD_nif]
        DriverName = LANDD$
        Bindings = TOKEN
        Bindings = 3C503

[NEBEUI_nif]
        DriverName = netbeui$
        Bindings = TOKEN
        Bindings = 3C503

;********* MAC DRIVER SECTION *********

[TOKEN]

[3C503]
```

NET.CFG File Changes. The NET.CFG file does not have to exist, because LANSUP can only bind to LANDD and defaults to Ethernet (LSB) addressing and a maximum packet size of 1514. LANSUP does not support 802.3 or Ethernet DIX II frame types. With Token Ring, LANSUP supports both Token Ring 802.5 and Token Ring SNAP. Due to differences in the ODI and NDIS specifications, the maximum packet size count will not be exactly the same. When using packet sizes other than 1514 or 4096, the ODI LSL buffers should be 6 bytes less than the NDIS driver.

In this sample NET.CFG file, the LSL buffer size is set to 1514 and the Ethernet 802.2 frame type is the only one loaded.

```
Protocol IPX
        bind LANSUP        ;default bind IPX to LANSUP

Link Support
        buffers 10 1514    ;maximum ethernet frame size

Link Driver LANSUP
        frame Ethernet_802.2;make sure 802.2 is loaded
```

LANSUP can easily be configured for Token Ring by changing the maximum packet size and Node Addressing (MSB) in the NET.CFG file.

ODINSUP and ES/LS

To continue our example of dual LAN adapters, we will show the ODINSUP configuration for the scenario pictured in Figure 10.

In this configuration, it is imperative that the Communications Manager SNA gateway and the IBM LAN Requester be installed first. ODINSUP requires changes to the PROTOCOL.INI file and the NET.CFG file. Once the CM SNA gateway and the IBM LAN Requester are installed, the ODINSUP installation will be very straightforward.

Figure 11 illustrates the use of ODINSUP with dual requesters and dual LAN adapters. Note that ODINSUP can be loaded reentrantly to support the use of Ethernet and Token Ring adapters simultaneously. One benefit in this scenario is that the NDIS protocol stacks can take advantage of the Ethernet DIX II and 802.3 frame types. Also note that the largest packet size (4096 bytes) is used for both adapters.

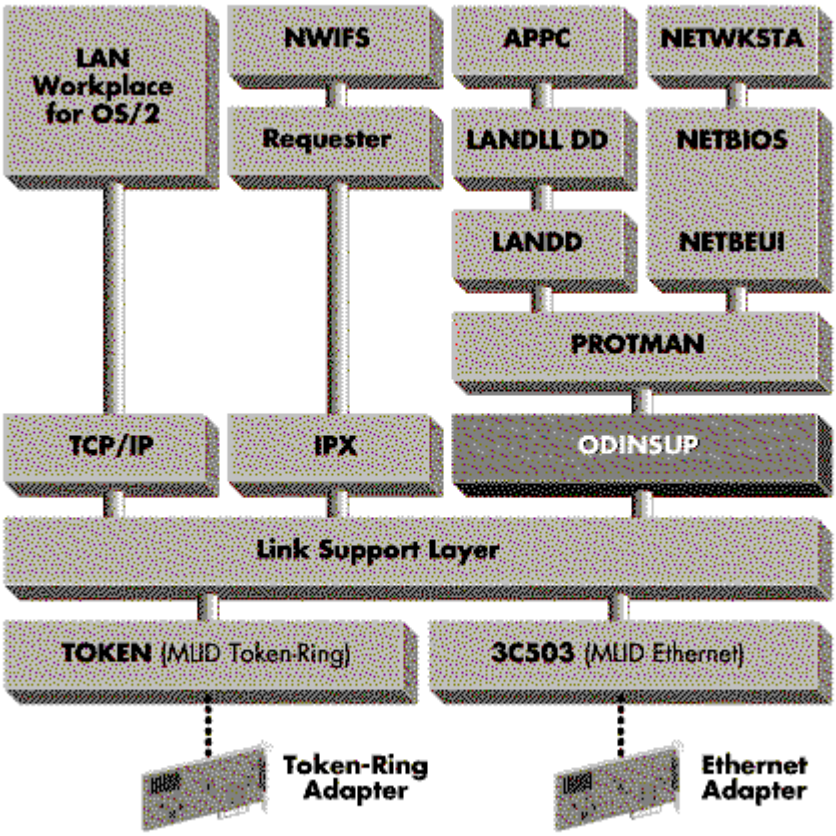

Figure 11: ODINSUP with Token Ring and Ethernet

The steps for installing ODINSUP with dual requesters and dual LAN adapters are:

1. Install OS/2 v1.3 with the EE Communications Manager.
2. Install the CSD 5050 update.
3. Install the CM SNA gateway.
4. At this point, logon to the LAN Server and make sure that the SNA gateway is working properly.
5. Run the NetWare Requester for OS/2 INSTALL program and select ODINSUP. Also select the SPX and Named Pipes Server support and enter the NP server name.
6. Run the NetWare Requester NDSINST to update the drivers. ODINSUP requires LSL v1.2 and above. ODI MLIDs should be dated after October 1991 and include the corresponding MSM v1.2 provided by Novell.
7. Edit the CONFIG.SYS and add L:\OS2 to the DPATH. Next, find the NDIS MAC drivers and remark out the device load statements. In our example, IBMTOK and "ELNKII" should be deleted or remarked. Check that the ODINSUP driver is being loaded after PROTMAN and before NETBIND.
8. Edit the PROTOCOL.INI file to change all occurrences of the NDIS MAC "Bindings" statements to the

ODI MLID driver. Also create a section heading for each ODI MLID; for instance, "[Token]" and "[3C503]."

CONFIG.SYS File Changes. The CONFIG.SYS file is not significantly different from the previous example of ODINSUP, except that there are several new NDIS protocol driver names. Note that the NDIS PROTMAN and LANDD drivers are loaded first, then the ODINSUP shim, followed by the NETBIND. Once these drivers are loaded, the ODI and NDIS stacks can be completed in any order. For simplicity, the only order change shown in the sample listing below is ODINSUP.

```
device=c:\ibmcom\lanmsgdd.os2 ...
        device=c:\ibmcom\protman.os2 ...
        device=c:\ibmcom\protocol\landd.os2
        device=c:\ibmcom\protocol\lanlldd.os2
        device=c:\ibmcom\protocol\netbeui.os2
        device=c:\ibmcom\protocol\netbios.os2
        rem device=c:\ibmcom\protocol\elnkii.os2
        rem device=c:\ibmcom\protocol\ibmtok.os2
        run=c:\ibmcom\protocol\landll.exe
        device=c:\ibmlan\netprog\rdrhelp.sys

device=c:\netware\lsl.sys
        run=c:\netware\ddaemon.exe
        device=c:\netware\odinsup.sys

run=c:\ibmcom\protocol\netbind.exe

device=c:\netware\ipx.sys
        device=c:\netware\spx.sys
        run=c:\netware\spdaemon.exe
        device=c:\netware\nmpipe.sys
        device=c:\netware\npserver.sys
        run=c:\netware\npdaemon.exe  np_serv_name
        device=c:\netware\nwreq.sys
        ifs=c:\netware\nwifs.ifs
        run=c:\netware\nwdaemon.exe
        device=c:\netware\netbios.sys
        run=c:\netware\nbdaemon.exe
        run=c:\netware\nwspool.exe

ifs=c:\ibmlan\netprog\netwksta.sys
```

Note that the IBM IFS is still loaded last. This is not as critical with the CSD 5050 and the newest IBM IFS, and it could be moved up with the rest of the IBM CM drivers.

PROTOCOL.INI File Changes. The required editing of the PROTOCOL.INI file is very simple. Delete or remark out the NDIS MAC driver sections and replace the protocol Bindings statements to bind to the ODI MLIDs. In this example, TOKEN replaces IBMTOK and "3C503" replaces "ELNKII."

```
[PROT_MAN]
        DriverName = PROTMAN$

[IBMLXCFG]
        LANDD_nif = LANDD.nif
        NETBEUI_nif = NETBEUI.nif

;********** PROTOCOL SECTION **********

[LANDD_nif]
```

```
        DriverName = LANDD$
        Bindings = Token
        Bindings = x3C503

[NEBEUI_nif]
        DriverName = netbeui$
        Bindings = Token
        Bindings = x3C503

;********** MAC DRIVER SECTION *********

[x3C503]

[Token]
```

NET.CFG File Changes. The NET.CFG file for this configuration is very straightforward. ODINSUP has two bind statements (loaded reentrantly), and each of the MLIDs has a Link Driver section that loads the distinct frame types.

Note that the Token Ring driver is loaded first so that the LSL 4096 byte maximum packet size will apply to all stacks. Also note that ODINSUP for Ethernet requires the Ethernet 802.2, DIX II, and SNAP frame types (all the possible types except Ethernet 802.3). For Token Ring, ODINSUP requires both the Token-Ring (802.5) and the SNAP frame types.

```
Protocol ODINSUP
        bind to Token      ;bind ODINSUP to TR
        bind to 3C503    ;and also to Ethernet

Link Support Layer
        buffers 10 4096     ;maximum Token Ring frame size

Link Driver Token
        frame Token-Ring ;default 802.2 inside 802.5
        frame Token-Ring_SNAP;also supported SNAP

Link Driver 3C503
        frame Ethernet_802.2;NetBIOS, IPXODI, and SNA
        frame Ethernet_II   ;TCP/IP, OSI
        frame Ethernet_SNAP ;AppleTalk, DECnet
        frame Ethernet_802.3;optional NetWare v2.2 IPX
```

The flexibility of ODI enables other ODI and NDIS applications to use the multiple frame types simultaneously.

Summary

ODINSUP offers the dual requester configuration increased performance.

A follow-up AppNote will cover OS/2 v2.0 configuration details for OS/2, WIN-OS/2, and DOS sessions for NetBIOS, Named Pipes and IPX. A discussion of installation over the network, Remote Initial Program Load (RIPL), and Multiple Virtual DOS Machines (MVDMs) will also be included.

Additional References

ODINSUP Interface for NDIS Protocols, 13 November 1991, Novell Inc. (included with developer SDK for NetWare 3.2 (now 4.0))

ODINSUP Readme file included on NetWire BBS

NetWare Requester for OS/2 v2.0 Reference (April 1992) 100-001157-002, Novell Inc.

IBM and Novell LAN Software Coexistence, Doug Spelce and Steve French, IBM Corp., Austin Texas. Published in IBM Personal Systems Technical Solutions, April, 1992 G325-5015

IBM OS/2 LAN Server and NetWare from IBM Coexistence Guide (November 1991) S04G-1139-96F8311

What's New in OS/2 Communications Manager (session L.1.4) Sam McAfee, IBM Education Network Systems, October 1991, IBM Communications Systems Technical Conference 1991.

IBM Extended Services for OS/2: Start Here S04G-1000-00

IBM Extended Services for OS/2: Communications Manager User's Guide S04G-1015-00

IBM Extended Services for OS/2: Workstation Installation Guide S04G-1008-00

IBM Extended Services for OS/2 Hardware and Software Reference S04G-1014-00