Chapter 3
NIOS DOS/MS Windows Client NLM

Client NLM Introduction

A driving concept behind the design of the NIOS Client was creating modular, reusable, and platform-independent software. To that end, all NIOS Client modules are in an NLM executable format and can be loaded and unloaded as necessary.

NIOS Client system modules:

- Are dynamically loadable and unloadable.
- Use NLM executable format.
- Run exclusively in a 32-bit flat memory model.
- Allocate memory that is guaranteed not to move or be discarded.
- Are fully language enabled.
- Are configured by utilities that use a configuration file.
- Require little or no static configuration information from the user. For example, it is no longer necessary to supply the number of open IPX sockets. As more sockets are opened, IPX dynamically allocates more memory to handle the additional sockets.
- Are written in C and Assembly.
- Run on single-processor Intel 386/486/Pentium systems.

DOS/MS Windows NIOS Client NLMs

Each functional component of the NIOS Client is briefly described here. Refer to each component's individual functional specification for more information.

The system level components are:

- NetWare I/O Subsystem (NIOS)
- LAN drivers
- Topology Support Modules
- Link Support Layer
- Internetwork/Sequenced Packet Exchange Protocol
- 16-bit ODI Protocol Compatibility Module
- Transport Service Interface
- Requestor/Shell
- System Debugger
- NDIS 3.0 Shim Module
- Source Routing Module

The user interface components are:

- DOS Installation / Upgrade / Configuration Utilities
- Windows Installation / Upgrade / Configuration Utilities

NetWare I/O Subsystem

The NetWare I/O Subsystem (NIOS) is the isolating layer between NetWare system-level components (core modules and various API mappers) and the host OS. As such, NIOS contains a full set of platform-independent APIs. These APIs provide a base from which powerful system-level functionality can be built. (See Figure 3.1.)

Additionally, NIOS contains the Client NLM loader/unloader functions. These APIs are responsible for loading and unloading NLMs and LAN drivers.

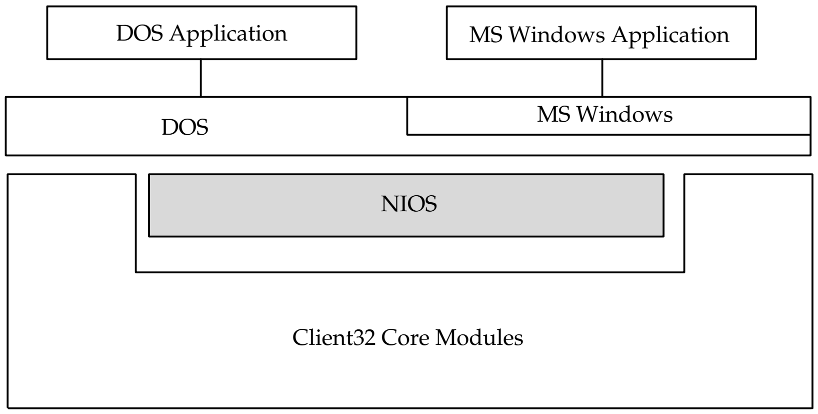

Figure 3.1: NIOS and NIOS Client Core NLMs

NIOS implements a number of NetWare OS APIs (called the NetWare OS Emulation module) that are necessary in order to use NetWare OS-compatible LAN drivers (LAN files) in the DOS/MS Windows environment.

Because DOS does not provide a protected-mode system-level environment, NIOS must. Under DOS, NIOS offers a number of DOS-specific APIs that allow NLMs running in protected-mode to manage the real-mode environment. This includes APIs that allow execution of real-mode code, real-to-protected callbacks, hooking real mode interrupts, and the like. An MS Windows transition architecture is also provided that allows environment- dependent modules the ability to correctly handle the DOS-to-Windows, Windows-to-DOS transition.

NIOS allows DOS real-mode and MS-Windows applications to access the 32-bit NLM environment from their respective 16-bit environments. NIOS provides a set of APIs that allow 16-bit applications to call 32-bit Client NLM APIs and access Ring-0 memory, thus bridging the gap between user applications and the NIOS Client system-level components.

For example, an end user would execute an application running in the DOS/MS-Windows environment to load or unload client system-level components. Under DOS, for example, the command LOAD loads the specified modules into a protected area.

```
> LOAD LSL
> LOAD NE2000
> LOAD IPX
.
.
.
```

In an MS-Windows environment, the interface is graphical instead of command-line, but the underlying principle is the same.

LOAD invokes system-level APIs to actually perform the basic load/unload operations. In this way, DOS/MS-Windows applications provide the user interface for managing the NIOS Client system, though all lower-level system functionality is provided by NLMs and NIOS.

LAN Drivers

The client uses unmodified NetWare OS-compatible (3.x and 4.x) LAN drivers. This provides a huge pool of proven, certified LAN drivers for the client environment.

Plus, the burden on third-party LAN adapter manufacturers is greatly reduced since the NIOS Client does not introduce another LAN driver interface to which developers have to write.

Topology Support Modules

Topology Support Modules (TSMs) are components of the NetWare OS LAN driver architecture which provide an intermediate layer between the LAN driver and the Multiple Link Interface (MLI) of the LSL module. There is a separate TSM for each supported topology type (for example, ETHERTSM.NLM, TOKENTSM.NLM). (See Figure 3.2.)

Note: The MLI is an API that allows the LSL to remain independent of physical media.

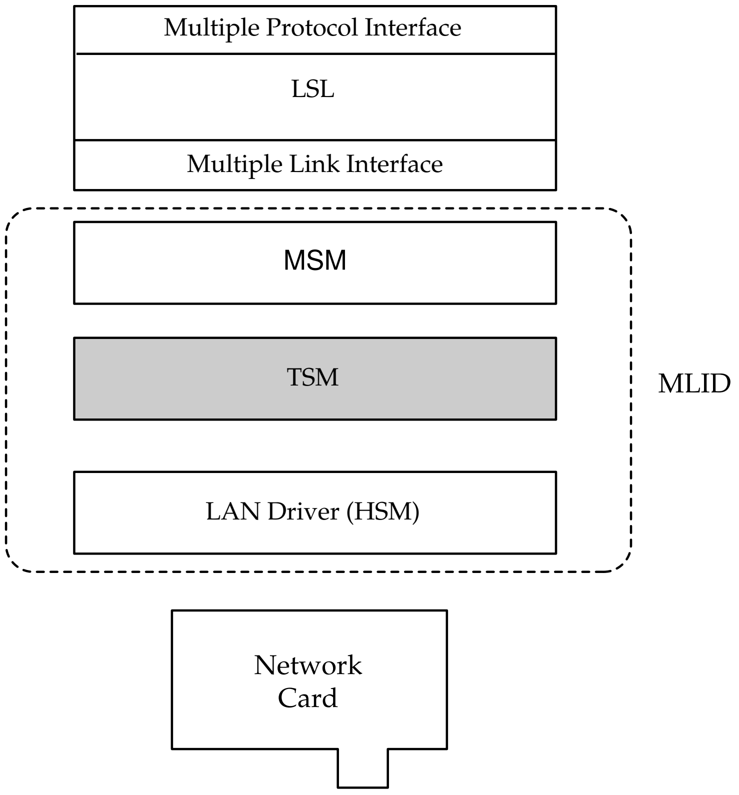

Figure 3.2: Topology Support Module

The client uses customized TSM modules which use a different packet-receive mechanism than the TSMs written for the NetWare OS. Instead of using the LSL buffer pool approach of the NetWare OS TSMs, the NIOS Client uses a Receive Look-Ahead approach, allowing protocols to preview packet header information and provide buffers directly for incoming packets. The Receive Look-Ahead method is much more efficient in a client environment than the traditional buffer pool method.

The client provides backward compatibility to the NetWare OS TSMs. A TSM written for the NetWare OS can be used on the NIOS Client.

Link Support Layer

The Link Support Layer (LSL) is the central component of any Open Data-Link Interface (ODI) implementation. (See Figure 3.3.)

The core of the NIOS Client's LSL is actually a ported version of the LSL used by the NetWare OS compliant with the ANSI "C" LSL interface. Because it does not use any OS-specific APIs, the unmodified core LSL can be used on other platforms supported by NIOS.

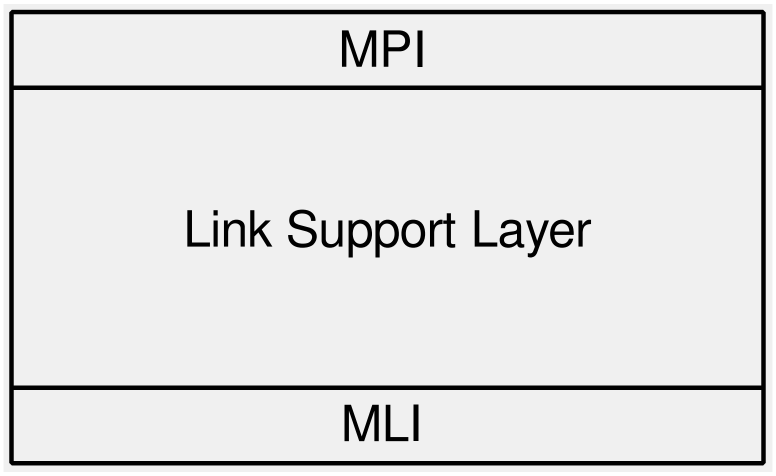

Figure 3.3: Link Support Layer

Internetwork/Sequenced Packet Exchange Protocols

The NLM (IPX.NLM) that implements the IPX and SPX protocols for the DOS/MS-Windows NIOS Client comprises a number of functional components. (See Figure 3.4.) The core IPX and SPX protocols are OS-independent and need not be modified to work on other NIOS-supported platforms.

The 16-bit IPX (IPXODI.COM) could bind to only one LAN driver at a time. The NIOS Client's IPX, however, is multiple- board aware; it can bind to more than one LAN driver at a time. While servers have always offered this capability by using internal routers, no such technology has been generally available on client workstations.

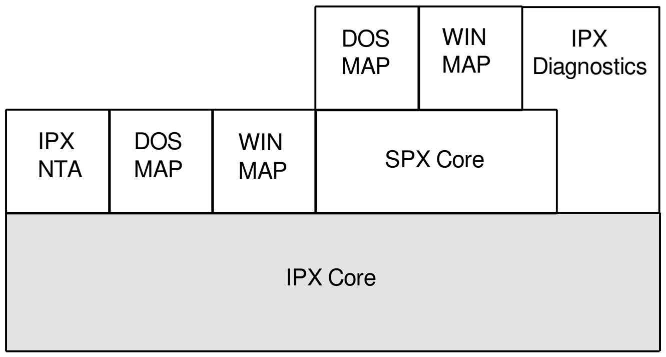

Figure 3.4: IPX.NLM architecture

The NIOS Client bypasses the need for cumbersome internal routers by demanding that client applications be multiple board-aware. An application queries IPX to find which boards are registered, and uses a new **GetLocalTarget** to select the best board on which to send and receive packets.

Built on top of the core IPX API are four API mappers: two that emulate the 16-bit IPX/SPX APIs, and two that emulate the 16-bit MS-Windows IPX/SPX interface. (The MS-Windows 16-bit interface was previously provided by VIPX.386.)

Additionally, an IPX diagnostics module emulates the Diagnostic/GNMA functionality currently available with NetWare clients.

IPX.NLM also includes the IPX Transport Service Agent (TSA) that interfaces with the TRAN.NLM.

Transport Service Interface

In an attempt to evolve to an entirely transport-independent Requestor/Shell, the NIOS Client contains a transport independent layer called the Transport Service Interface (TSI). This layer is made up of a transport manager called TRAN.NLM and individual Transport Service Agents (TSAs). NTAs offer a consistent API to the Transport Service Users (TSUs) to each relevant transport protocol. (See Figure 3.5.)

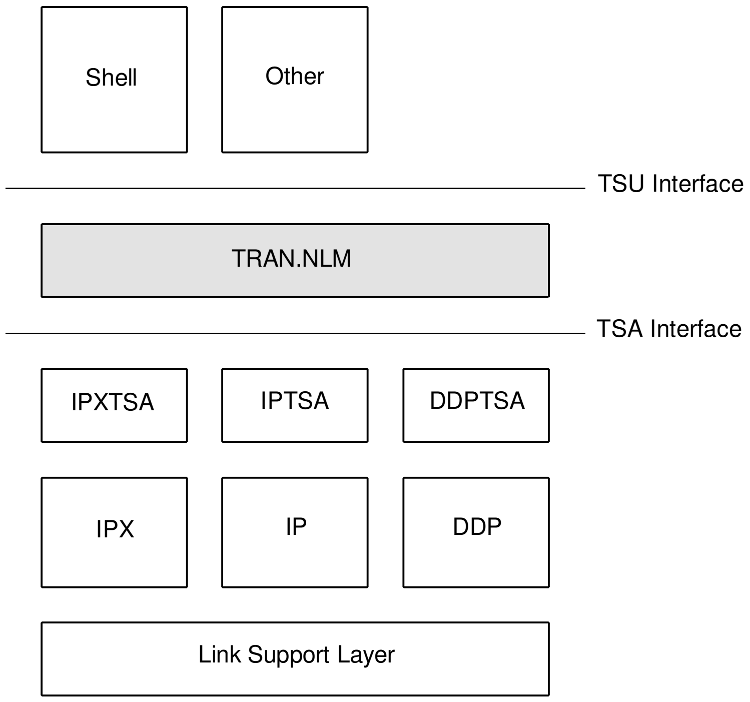

Figure 3.5: Transport Service Interface (TSI)

An example of aTSU is the NIOS Client Shell. Before the NIOS Client, NetWare Shells spoke only IPX. But the NIOS Client Shell runs on any transport. This is only possible because the TSI masks protocol differences from the Shell. Instead of issuing an IpxSendPacket request, the NIOS Client Shell would send a generic SendPacket. The TSI would translate it into a protocol-specific request.

Each supported protocol will have its own TSA. A TSA is a mapper to a protocol. For example, the IPX TSA is linked with IPX.NLM to provide an interface for the IPX protocol.

Requestor/Shell

NETX.NLM, the NIOS Client Shell, is a true DOS Int 21h shell (as opposed to a Redirector). Though harder to implement, Shells are significantly faster than Redirectors. The NIOS Client Shell is analogous to the 16-bit NETX.EXE module, except that it supports the advanced features introduced with the 16-bit VLM Requestor, such as NetWare Directory Service (NDS) support, improved packet burst protocol, auto-reconnect, and Personal NetWare support.

The Requestor/Shell module consists of many sub-components, some of which are OS-dependent and some -independent. Many of the source code modules for NETX.VLM can be reused without modification, simply being linked in for a new OS implementation. However, a significant portion of the Shell will be OS-dependent due to its close ties with the underlying desktop OS.

NETX.NLM introduces a number of improvements over existing 16-bit shells, such as improved file caching, flexible local/global network resource mappings, increased usage of new NCPs, ease of use, and a number of other performance improvements.

NDIS 3.0 Shim Module

ODINSUP is the module which allows ODI to run NDIS 3.0-compliant network layer protocols. It works like this:

The NIOS Client loads the Microsoft ProtMan module (roughly equivalent to NetWare's LSL) and related protocols along with the standard ODI protocols. ODINSUP takes the output of ProtMan and maps it to the ODI LAN Driver instead of to an MS MAC (Medium Access Control) module. In effect, ODINSUP.NLM provides an NDIS MAC driver interface to NDIS-compliant protocol, though an ODI LAN driver is actually used to drive the LAN. (See Figure 3.6.)

This module provides dual connectivity between LAN Manager/Windows for Workgroups and NetWare services.

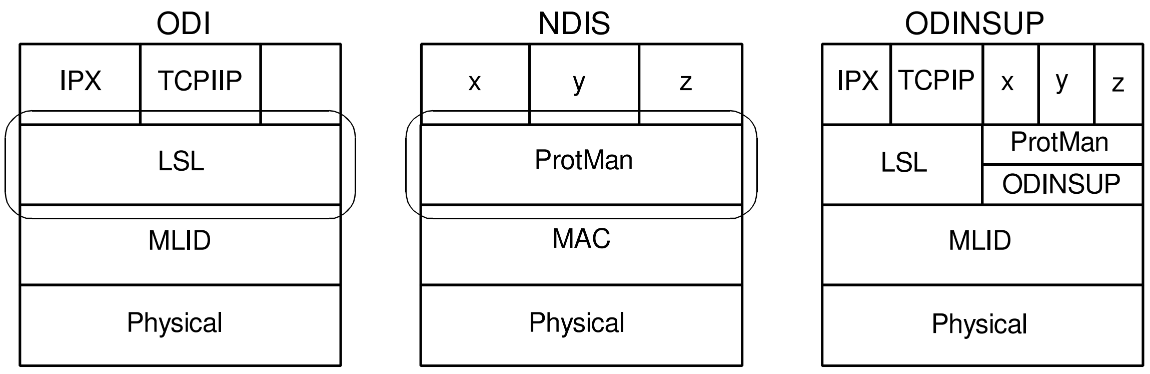

Figure 3.6: NIOS Client ODINSUP capability

Source Routing Module

IBM Token-Ring Source Routing bridges Token-Ring rings together into one logical ring.

Currently, Novell implements this technology with a ROUTE.NLM module in the NetWare OS, and source route modules for all other client environments. This provides the necessary protocols to allow Token-Ring ODI LAN drivers to work correctly on a Source Routed network.

Because the NIOS Client ROUTE.NLM generally uses only NetWare OS LAN driver APIs, it is expected that ROUTE.NLM can be used unmodified on the NIOS Client.

System Debugger

Because the DOS/MS Windows NIOS Client environment operates in protected mode under DOS, no off-the-shelf debugger can be used to debug NIOS Client system components. Novell has developed the DEBUG.NLM module to assist in developing NIOS Client NLMs.

This module will be made available to third-party developers in the DOS/MS Windows NIOS Client SDK.