Windows 2000 Server

Appendix B - Windows 2000 Network Architecture

Microsoft® Windows® 2000 network architecture is composed of software components that provide networking abilities to the Windows 2000 operating system. This appendix describes the components, protocols, and interfaces within Windows 2000. In addition, it introduces the networking concepts that provide a foundation for the other chapters in this book to build upon.

In This Appendix

Related Information in the Resource Kit

- For more information about SNA protocols, see "Interoperability with IBM Host Systems" in the *Microsoft® Windows® 2000 Server Resource Kit Internetworking Guide*.

Overview of Windows 2000 Network Architecture

This chapter describes software and hardware components and the connections between them that allow computers to function as a network. Windows 2000 network components are arranged in layers. Each layer has specific tasks to perform and within each layer more than one component can perform a similar task.

The Windows 2000 network layers are described in the following sections from the bottom of the network architecture model up to the top. The layers are:

Network Driver Interface Specification (NDIS) Layer NDIS is the layer that provides a communication path from a network transport to a physical device, such as a network adapter. NDIS acts as a boundary layer between network adapters and network protocols and manages the binding between these components. NDIS adds support for connection-oriented network media such as ATM and Integrated Services Digital Network (ISDN) and continues to support traditional connectionless network media such as Ethernet, Token Ring, and Fiber Distributed Data Interface (FDDI). This layer contains the miniport drivers that interface directly with the network adapter.

Network Protocol Layer The network protocols provide services for clients. These services allow applications or clients to send data over a network. Network protocols include TCP/IP, ATM, NWLink Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX), NetBEUI, Infrared Data Association (IrDA), AppleTalk and Data Link Control (DLC). Systems Network Architecture (SNA) protocols are available with the addition of Microsoft® SNA Server.

Transport Driver Interface Layer The transport driver interface (TDI) provides a standard interface between network protocols and clients of these protocols (such as applications, network redirectors or networking Application Programming Interfaces (APIs)).

Network Application Programming Interface Layer The network application programming interface (API) provides standard programming interfaces for network applications and services. They support Winsock, NetBIOS, Telephony API (TAPI), Messaging API (MAPI), WNet API and other services.

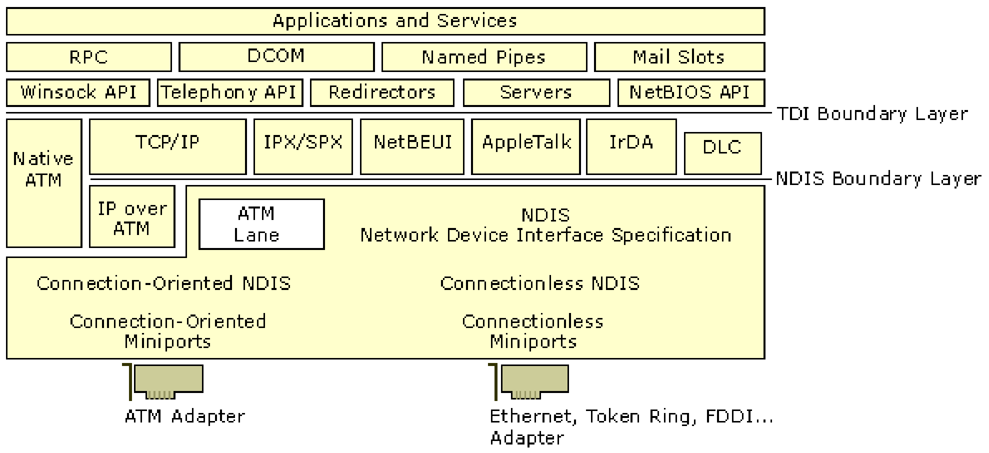
Interprocess Communications Layer Interprocess communications (IPC) support client/server computing and distributed processing. Some of the services that they support are remote procedure calls (RPC), Distributed Component Object Model (DCOM), named pipes, mailslots, and Common Internet File System (CIFS).

Basic Network Services Layer Basic network services support network user applications by providing services. These include network address management, name services, file services and advanced network services such as Internet Protocol Security (IPSec) and Quality of Service (QoS).

The International Organization for Standardization has a model for computer networking, also called the Open Systems Interconnection (OSI) Reference Model. This model is useful to describe network layers. The OSI model defines a modular approach to networking, with each layer responsible for some discrete aspect of the networking process. The OSI model does not correspond exactly to most existing network architectures. However, models assist in understanding how networks function by providing a structure to use as a comparison. For more information about the OSI model, see "OSI Model" in this book.

Network communication begins when an application program attempts to access resources on another computer. Data and requests move from layer to layer within a computer. Each layer is able to communicate with the layer immediately above it and the layer immediately below it. If the packet is not meant for use by the current layer, it passes the packet to an adjacent layer. Packets travel down the network protocol stack of the first computer. If the destination is on another networked computer, the packets of data are sent across the physical media (such as wiring, fiber optic cable or satellite). The data is passed upward through the lower layers of the second computer up to the same layer that started the exchange of data.

Figure B.1 represents a model of Windows 2000 network architecture.



If your browser does not support inline frames, click here to view on a separate page.

Figure B.1 Windows 2000 Network Architecture

Software components are represented by rectangles. These components are in horizontal layers. Components that are on the same horizontal level provide similar functionality. The top layer of the diagram is where user applications reside. In order to communicate with other networked computers, additional software and hardware support is needed. Each layer below the applications and services layer provides services that are necessary to create packets of data, arrange for their delivery and send them across the physical media to another computer.

Boundary layers are interfaces between functional layers in the Windows 2000 network architecture model. Creating boundaries as breakpoints in the network layers helps standardize programming for developers. Because the functionality between the layers is well-defined, developers need to program only to a boundary layer instead of having to code from the top (an application program) to the bottom of the protocol stack (network adapter). If software is correctly written to a boundary layer, then support on the other side of that layer already exists and does not need to be written. Both the transport device interface (TDI) and Network Driver Interface Specification (NDIS) are boundary layers. In the diagram, DLC and Native ATM both have a gap between the top edge of DLC, Native Asynchronous Transfer Mode (ATM) and the TDI layer because they do not interface through the TDI layer.

Binding is the linking of network components on adjacent layers of the protocol stack. Binding occurs at boundary layers and other adjacent layers. Binding enables communication between the various layers.

Components are portions of software that perform specific tasks. A network component can bind to one or more network components above or below it. When adding network software, Windows 2000 binds all needed and associated components together. During binding, an information file (.inf) contains the instructions necessary to establish the required binding relationships.

Figure B.2 shows two protocols binding to two network adapters inside the same computer.

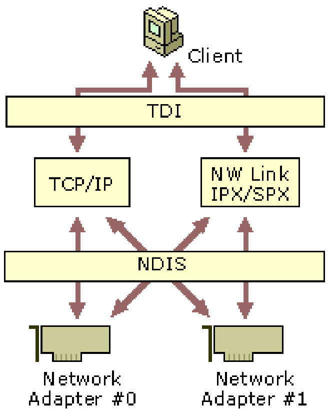


Figure B.2 Binding

Network Driver Interface Specification

The Network Driver Interface Specification (NDIS) is a specification for network driver architecture that allows transport protocols such as TCP/IP, Native ATM, IPX, and NetBEUI to communicate with an underlying network adapter or other hardware device. The network adapter can then send or receive data over the network. NDIS permits the high-level protocol components to be independent of the network adapter by providing a standard interface to the network protocols. Because Windows 2000 network architecture supports NDIS, it requires that network adapter drivers be written to the NDIS specification.

Windows 2000 NDIS provides both a standard connectionless interface and also defines a connection-oriented interface. Microsoft® Windows NT® version 4.0 network architecture supported traditional connectionless network standards such as Ethernet, Token Ring, and FDDI. Connectionless networking does not negotiate, manage, and maintain a connection before transmitting data. Connectionless networking, also known as a datagram service, is a "best-effort" delivery service. There is no guarantee that messages won't be lost, duplicated or delivered out of order. These services are usually provided by the network protocols if required.

Windows 2000 continues to support traditional connectionless networking, but adds support for connection-oriented mediums, such as Asynchronous Transfer Mode (ATM) and Integrated Services Digital Network (ISDN). In connection-oriented networking, a connection is created and all data is sent in sequence over a virtual circuit. Windows 2000 negotiates connections using a call manager. A call manager is a portion of software that can initiate and maintain connections, creating virtual circuits (VCs) between two network endpoints. Virtual circuits act as conduits for the transmission of data, allowing greater control of bandwidth, latency, delay variation, and sequencing. These services provide greater support for distributed voice, data, and video applications.

Windows 2000 NDIS is implemented by a file called Ndis.sys, referred to as the "NDIS wrapper." The NDIS wrapper is code that surrounds all NDIS device drivers. The NDIS wrapper provides a uniform interface between protocol drivers and NDIS device drivers, and contains supporting routines that make it easier to develop NDIS drivers. NDIS allows an unlimited number of network adapters in a computer and an unlimited number of protocols bound to one or more network adapters.

Figure B.3 shows the NDIS wrapper architecture.

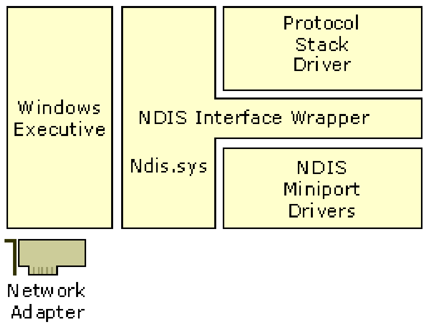


Figure B.3 NDIS Wrapper (Ndis.sys)

The NDIS wrapper defines the way protocols communicate with network adapters. The protocols communicate with the NDIS wrapper rather than the network adapter. This is an example of the modular structure of a layered network model. The network adapter is independent from the transport protocols.

NDIS New Features

Windows 2000 NDIS includes many new features, such as connection-oriented NDIS and new intermediate and miniport driver support. Other Windows 2000 NDIS features include:

- Connection-Oriented NDIS
- Wake-On-LAN
- Media-Sense
- Network Plug and Play
- TCP/IP Task Offload

Connection-Oriented NDIS

Connection-oriented NDIS (CoNDIS) is the portion of NDIS that supports connection-oriented media. CoNDIS supports connection-oriented media such as dial-up networking, ATM and network streaming over connection-oriented media. Connection-oriented NDIS provides the support to establish, maintain, and close connections.

Wake-On-LAN

Wake-On-LAN controls the wake-up of computers based on network events. It is a subset of the OnNow Power Options initiative. In order for Wake-On-LAN to function, the network adapter must be Wake-On-LAN–capable and the device driver must support Wake-On-LAN. A network adapter may be put into a low power mode when the system requests a power level change. The user or system can initiate the request. For example, the user might want to put the system into sleep mode, or the system might request sleep mode based on keyboard or mouse inactivity.

Unless initiated by a user, all overlying network components must agree to the request before the network adapter can be turned off. If there are any active sessions or open files over the network, the turn-off request can be refused by any or all of the components involved.

Many events enable a system to wake up without user intervention. The system may be able to wake up from a lower power state based on network events specified by the networking software. This capability means that any standard Windows network access (such as connections to shared folders, Winsock connections, and service and management applications) can wake the system from lower power states. This is done by:

- Receipt of a previously registered packet.
- Receipt of a magic packet. A magic packet is a packet that contains 16 contiguous copies of the receiving network adapter's Ethernet address.
- A connection event, such as plugging in a network cable.

Network drivers and hardware that do not support Wake-On-LAN can still be used on Windows 2000. Systems having network adapters with no Wake-On-LAN capabilities can be suspended and resumed based on user activity, but not resumed based on network events.

Media Sense

Media Sense is the capability of a network adapter to indicate when it does or does not have a connection to the physical network medium. Most Windows 2000 network technologies support Media Sense. Protocols and applications can receive these notifications and act accordingly. For example, an icon can be displayed indicating the media is disconnected, an event can be logged, and TCP/IP can manage addresses with the knowledge of the state of the network.

Network Plug and Play

Network Plug and Play is a combination of hardware and software support that enables a computer system to recognize and adapt to hardware configuration changes with little or no user intervention. A user can add or remove Plug and Play devices dynamically. No intricate knowledge of computer hardware is necessary. For example, a user can dock a portable computer and use the docking station's Ethernet card to connect to the network without changing the configuration. Later, the user can undock that same computer and use a modem to connect to the network, again without making any manual configuration changes.

TCP/IP Task Offload

TCP/IP Task offload allows tasks normally performed by the transport layer to be processed by the network adapter. This reduces the overhead required of the system CPU for these tasks. This allows the system CPU to do more work, possibly increasing the throughput to the network. A special query is made by the transport driver to find out if the network adapter supports the offload of the computation of TCP/IP checksums, TCP/IP segmentation (large send), Fast Packet Forwarding and IPSec Offload. If one or more of these conditions is detected, the transport can request that the network adapter furnish these services.

TCP/IP Checksum Task Offload Capability

TCP/IP checksums verify the integrity of the data packet. TCP/IP queries the miniport to determine its ability to perform checksum calculations. If the miniport is capable of handling offloads, then it performs these calculations. These computations can consume many CPU cycles. This can include send and receive checksum calculations for TCP, User Datagram Protocol (UDP), and IP. The miniport driver requests that the network adapter perform the calculations rather than requiring the CPU to process this request. This can result in enhanced performance.

TCP/IP Segmentation Offload

TCP/IP segmentation (large send) is the creation of TCP packets from data that is too large for transmission over network media. TCP/IP splits data into small segments, adds IP and TCP headers, and creates TCP packets. TCP/IP segmentation can now be performed by NDIS miniports and a capable network adapter. The adapter must be able to calculate IP and TCP checksums for send packets and have an appropriate miniport driver. Offloading these calculations from the CPU results in greater performance for the system.

Fast Packet Forwarding

Fast Packet Forwarding allows multiport network adapters (FastEthernet, FDDI, or similar single-port network adapters) to use Windows 2000 to route packets from one port to another port without passing the packet to the host processor. This increases throughput to the network and reduces work for the CPU.

IPSec Offload Capability

Internet Protocol Security (IPSec) is an Internet Engineering Task Force (IETF) standard for security at the packet processing layer of IP networks. IPSec provides two security services:

- Authentication Header. Allows sender authentication.
- Encapsulating Security Payload. Supports both sender authentication and payload encryption.

The IPSec protocol information associated with each of these services is inserted into the packet in a header that follows the regular IP header. Included in this information is the Security Parameter Index, which is a 32-bit value used to distinguish between different Security Associations (SAs) terminating at the same destination and using the same IPSec protocol.

The work of encrypting and decrypting each packet can be assigned to the network adapter through the use of NDIS and the associated miniport drivers. With proper configuration of security policy in Windows 2000, outgoing IP packets are authenticated and encrypted before transmission to the network, and incoming IP packets are validated and decrypted.

For more information about new features in NDIS, see the Platform Software Development Kit (SDK). For more information about IPSec, see "Internet Protocol Security" in this book.

NDIS Driver Types

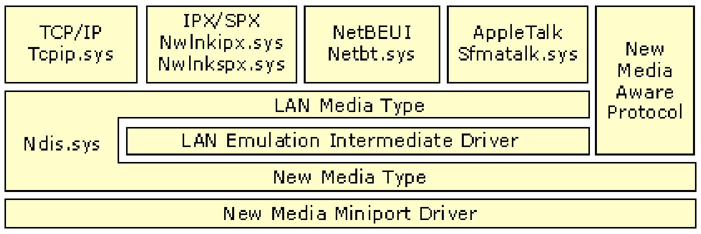
NDIS drivers allow the transport protocols to communicate with the hardware layer. These drivers can be implemented in different configurations, described in the following sections.

Intermediate Drivers

The *intermediate driver* are found just below the transport protocols and above the miniport drivers in the network protocol stack. There are two types of intermediate drivers that are used: the LAN Emulation Intermediate driver and the Filter driver.

LAN Emulation Intermediate Driver

The LAN Emulation Intermediate Driver translates packets from the overlying connectionless transport's local area network (LAN) format to the connection-oriented format (such as ATM) below. Therefore, the transport protocols appear to communicate with a LAN network adapter (Ethernet) but in reality they are communicating with a different hardware device. The LAN Emulation Intermediate driver translates the packets coming and going to the protocol layer above. It converts these packets into packets that can be sent over a different medium. Figure B.4 illustrates the LAN Emulation Intermediate driver architecture.



If your browser does not support inline frames, click here to view on a separate page.

Figure B.4 LAN Emulation Intermediate Driver

Currently, the only supported application for this driver is LAN Emulation for ATM. However, this driver configuration can be used for other types of "new media" in future configurations.

For more information about NDIS, see the Platform Software Development Kit (SDK).

Filter Driver

Filter drivers perform special operations (such as compression, encryption and tracing) on packets being transported through them. Figure B.5 shows the filter driver architecture.

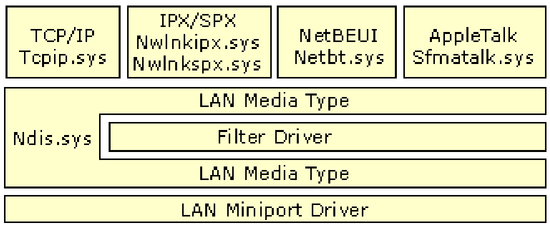


Figure B.5 The Filter Driver

As packets are transported through the filter driver, compression and encryption can be performed. Several services utilize this type of intermediate driver, such as the packet scheduler in *Quality of Service* (QoS) and *Network Load Balancing*.

Miniport Drivers

A *miniport driver* is a driver that connects hardware devices to the protocol stack. The miniport driver is connected to an intermediate or protocol driver and a hardware device. A miniport driver handles the hardware-specific operations necessary to manage a network adapter or other hardware device. They implement sending and receiving data on the network adapter. Microsoft produces some intermediate drivers for Windows 2000. This is helpful to hardware manufacturers because it is not necessary for them to implement added functionality. To add additional functionality, hardware vendors can also create intermediate drivers. In this chapter, miniport drivers are also referred to as miniports.

Windows Driver Model (WDM) is a specification for device drivers. WDM makes device drivers compatible between Windows 2000 and Microsoft® Windows® 98. To help reduce the effort necessary for hardware vendors to support all Windows platforms, WDM enables devices designed for either Windows 2000 or Windows 98 to be installed and used with computers running under either operating system. WDM developers write smaller code pieces (miniports) that talk to their hardware directly and call the appropriate class driver to do the bulk of the common tasks.

Miniport drivers can be serialized or deserialized. Serialized drivers rely on NDIS to sequence calls to miniport functions and to manage send queues. Deserialized miniport drivers on multiprocessor systems can perform faster by serializing access to their internal data structures rather than by allowing NDIS to perform this function. They also queue all incoming send packets rather than using NDIS. This can result in a better full duplex performance. However, deserialized miniport drivers are more difficult to design and require more testing and debugging.

Common Miniport Drivers

The most common miniport drivers are:

- Connectionless miniport drivers

- NDISWAN miniport drivers
- Non-NDIS Lower Interface miniport drivers
- Connection-oriented miniport drivers

Connectionless Miniport Drivers

Connectionless miniport drivers control network adapters for connectionless network media such as Ethernet, FDDI, and Token Ring.

Figure B.6 shows the architecture of connectionless miniport drivers.

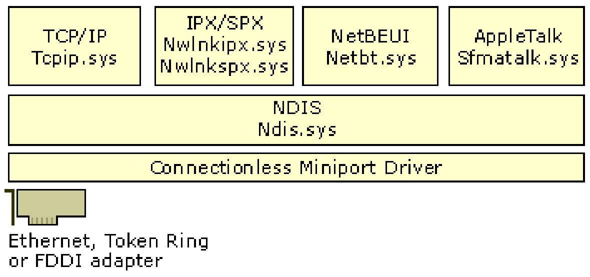


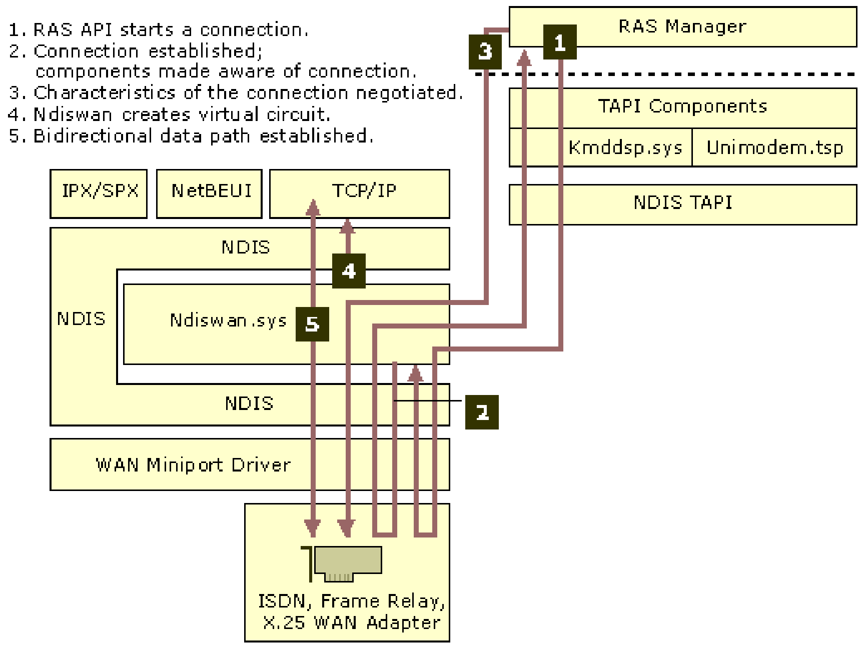
Figure B.6 Connectionless Miniport Driver Architecture

The most common use of connectionless miniports is for LAN-based network adapters. Network mediums such as Ethernet, Token Ring and FDDI do not support connection-oriented communications. Packets are sent to or received from a destination without any existing connection.

NDISWAN Miniport Drivers

There are many types of NDISWAN miniports. NDISWAN miniports are used with ISDN, Frame Relay and X.25 to provide dial-up networking. Ndiswan.sys is a Microsoft driver that provides Point-to-Point Protocol (PPP encapsulation), compression, and encryption. Ndiswan.sys communicates with the Routing and Remote Access service.

Figure B.7 illustrates the NDISWAN miniport driver architecture and usage.


1. RAS API starts a connection.
2. Connection established; components made aware of connection.
3. Characteristics of the connection negotiated.
4. Ndiswan creates virtual circuit.
5. Bidirectional data path established.
RAS Manager
TAPI Components
Kmddsp.sys
Unimodem.tsp
NDIS TAPI
IPX/SPX
NetBEUI
TCP/IP
NDIS
NDIS
Ndiswan.sys
NDIS
WAN Miniport Driver
ISDN, Frame Relay, X.25 WAN Adapter
1
2
3
4
5

If your browser does not support inline frames, click here to view on a separate page.

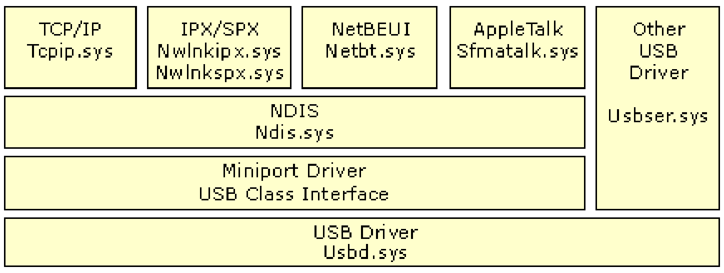
Figure B.7 NDISWAN Miniport Driver Architecture and Usage

Figure B.7 shows an example of NDISWAN usage. The numbered arrows show the steps for the Routing and Remote Access service to set up a PPP session using an NDISWAN miniport driver.

1. The Routing and Remote Access service initiates a connection by passing information down through the TAPI components, (kmddsp.tsp, NDIS TAPI) to the WAN miniport driver (NDISWAN).
2. A reply is passed back to the Routing and Remote Access service to inform it that a connection is established by the same path. All components are made aware that the call attempt was completed.
3. The RAS Manager negotiates the characteristics of the PPP connection. This includes compression protocols and frame types.
4. Ndiswan.sys creates a virtual circuit for the appropriate protocol.
5. Data is both sent and received through the connection that has been established by Ndiswan.sys.

Non-NDIS Lower Interface Miniport Driver

A Non-NDIS lower interface driver is a connection-oriented miniport driver that interfaces to network protocols on the top, but on the bottom can interface to devices, such as Universal Serial Bus (USB) and Institute of Electrical and Electronics Engineers (IEEE) 1394 devices. Figure B.8 shows the Non-NDIS miniport driver architecture.



If your browser does not support inline frames, click here to view on a separate page.

Figure B.8 Non-NDIS Miniport Driver Architecture

Connection-Oriented Miniports

The connection-oriented miniport transports data on a particular connection rather than to a particular destination. It is a new NDIS driver architecture. Connection-oriented miniports differ from connectionless miniports because a connection must be established between two points before data can be exchanged. Connection-oriented miniports support media such as ATM.

Certain applications rely on connection-oriented communication. Since data is sent over a connection, it remains in sequence and all data follows the same path. Because all data follows the same path, QoS parameters can be controlled more easily than connectionless data transfer. QoS requirements for the connection are negotiated by a call manager if the medium supports QoS. Call managers create and maintain connections. There are two types of call managers:

- A stand-alone call manager is a discrete software entity separate from the miniport driver.
- An integrated call manager has code that is an integral part of the connection-oriented miniport driver.

Connection-oriented miniports support many types of data transmission. Data, voice and videoconferencing using data streaming are made possible with connection-oriented miniport drivers. NDIS can offload synchronization and queue management details to the miniport driver. NDIS connection-oriented miniport drivers can expose a connection to the network protocols. For a diagram and additional information about connection-oriented miniport drivers supporting ATM, see "ATM" later in this chapter.

Network Protocols

Protocols are specifications for standardized packets of data that make it possible for networks to share information. Windows 2000 supports many different protocols. The packets of information are moved up and down the protocol stack, and across the transmission media. Network protocols include:

- Transmission Control Protocol/Internet Protocol (TCP/IP)
- Asynchronous Transfer Mode (ATM)
- NetWare Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX)
- NetBIOS Enhanced User Interface (NetBEUI)
- AppleTalk
- Data Link Control (DLC)
- Infrared Data Association (IrDA)

Note The Systems Network Architecture (SNA) protocols are not included in Windows 2000. SNA protocols are available with the addition of Microsoft SNA Server. SNA Server is a separate product that supports interoperability with IBM midrange and mainframe computers. For more information about SNA protocols, see "Interoperability with IBM Host Systems" in the *Windows 2000 Internetworking Guide*.

TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) has been adopted by Microsoft as the strategic enterprise transport protocol for Windows 2000. The Windows 2000 TCP/IP suite is designed to make it easy to integrate Microsoft enterprise networks into large scale corporate, government, and public networks, and to provide the ability to operate over those networks in a secure manner.

Several factors have lead to the success of TCP/IP. The protocol is routable, which means that data packets can be switched (routed to a different subnet) by use of the packet's destination address. TCP/IP's ability to be routed allows greater fault tolerance. If a network failure occurs, packets are transported by a different route. Another factor contributing to the success of TCP/IP is the massive interest in the Internet. TCP/IP is the standard for computer interconnectivity.

Windows 2000 TCP/IP includes several performance improvements for networking within high-bandwidth LAN and wide area network (WAN) environments. These features include the following:

Large Window Support

Large window support improves performance of TCP/IP when there are large amounts of data in transit or unacknowledged between two connected. In TCP-based communication, the window size is the maximum number of packets that can be sent in a streamed sequence before the first packet must be acknowledged. Large window support allows for more data packets to be in transit on the network at one time and increases effective bandwidth.

Selective Acknowledgments

Selective acknowledgments is a TCP option that allows the receiver to selectively notify and request from the sender only those packets that were missing or corrupted during initial delivery. Selective acknowledgments allow networks to recover quickly from a state of congestion or temporary interference by requiring only lost packets to be resent. In previous TCP/IP implementations, if a receiving host failed to receive a single TCP packet, the sender might retransmit not just the corrupted or missing packet, but all subsequent packets. With selective acknowledgments, fewer packets are sent so better utilization of the network results.

RTT Estimation

Round Trip Time Estimation (RTT) is a technique of estimating packet transit times and adjusting for the optimum retransmission time for packets. Round Trip Time is the amount of time it takes for a round-trip communication between a sender and receiver on a TCP-based connection. Because performance depends on knowing how long to wait for a missing packet, improving the accuracy of RTT estimation results in better retransmission time-out values being set on each host. Better timing particularly improves performance over long round-trip network links, such as WANs that span large distances (continent-to-continent) or use either wireless or satellite links.

IP Security

Internet Protocol Security (IPSec) is an encryption process that allows data to be scrambled to make it virtually impossible to view its contents. IPSec uses cryptography-based security to provide integrity, data origin authentication, protection against replays, confidentiality, and limited traffic flow confidentiality. Because IPSec is provided at the IP layer, its services are available to the upper-layer protocols in the stack, and are transparently available to existing applications.

IPSec enables a system to select security protocols, decide which algorithm to use for the service, and establish and maintain cryptographic keys for each security relationship. IPSec can protect paths between hosts, between security gateways, or between hosts and security gateways. IPSec policy can be configured locally on a computer, or can be assigned through Windows 2000 Group Policy mechanisms using the Active Directory™ directory service.

When IPSec is used to encrypt data, network performance is generally reduced due to the processing overhead of encryption. One method of reducing the processing overhead is to offload the processing to a hardware device. Since NDIS supports task offloading, it is feasible to include encryption hardware on network adapters.

For more information about IPSec, see "Internet Protocol Security" in this book.

For more information about TCP/IP, see "Introduction to TCP/IP" and "Windows 2000 TCP/IP" in this book.

Generic Quality of Service

Generic Quality of Service (GQoS) is implemented in Winsock, so Windows 2000 GQoS can run on any network that supports TCP/IP. GQoS ensures the quality of a connection. QoS allows developers to deploy real-time applications over IP networks while providing acceptable levels of bandwidth, latency, and jitter. GQoS allows TCP/IP to provide the benefits of ATM in a TCP/IP environment.

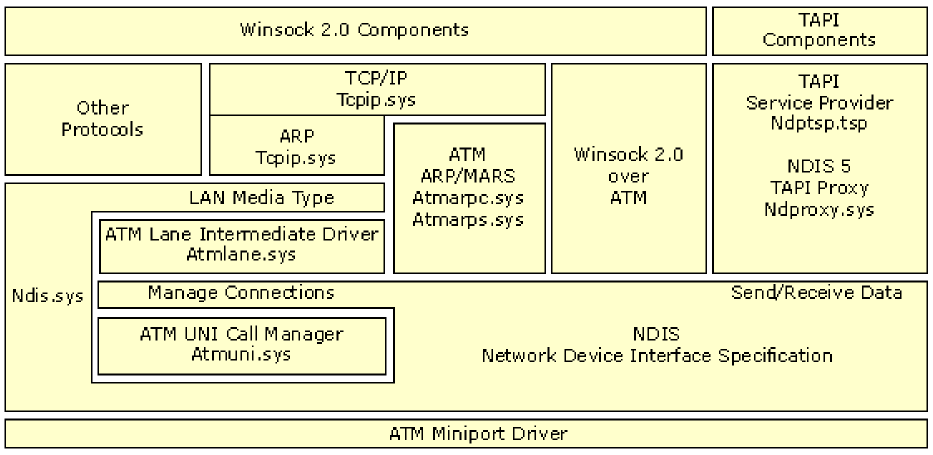
For more information about QoS, see "Quality of Service" in this book.

ATM

The Asynchronous Transfer Mode (ATM) protocol is a connection-oriented protocol that is ideal for voice, video and data communications. ATM is a high-speed networking technology that transmits data in cells of a fixed length. ATM is a native connection-oriented transport protocol. It is composed of a number of related technologies including software, hardware, and connection-oriented media. A cell is a fixed-length packet containing 53 bytes of information. Since the number of bytes—and consequently the transit time—of the cell is constant, the cells can be switched at a constant interval.

An ATM endpoint establishes a connection or virtual circuit prior to sending any data on the network. It then sends cells along this path toward the destination. This virtual circuit is a direct path from one endpoint to another. While establishing the connection, the ATM endpoint also negotiates a Quality of Service contract for the transmission. This contract spells out the bandwidth, maximum delay, acceptable variance, and other parameters the virtual circuit (VC) provides, and this contract extends from one endpoint to the other. Since the virtual circuit is connection-oriented, the data arrives at the receiving end in proper order and with the specified service levels. ATM is an excellent compromise for the transmission of both voice and data on a network. ATM provides a guaranteed Quality of Service on a LAN, a WAN, and a public internetwork.

Figure B.9 illustrates the connection-oriented miniport driver architecture as implemented in ATM.


Winsock 2.0 Components
TAPI Components
Other Protocols
TCP/IP Tcpip.sys
ARP Tcpip.sys
ATM ARP/MARS Atmarpc.sys Atmarps.sys
Winsock 2.0 over ATM
TAPI Service Provider Ndptsp.tsp
NDIS 5 TAPI Proxy Ndproxy.sys
LAN Media Type
ATM Lane Intermediate Driver Atmlane.sys
Ndis.sys
Manage Connections
Send/Receive Data
ATM UNI Call Manager Atmuni.sys
NDIS Network Device Interface Specification
ATM Miniport Driver

If your browser does not support inline frames, click here to view on a separate page.

Figure B.9 ATM Using Connection-Oriented Miniports

ATM is supported by Windows 2000 architecture with the following components. This diagram is used for:

- LANE (LAN Emulation)
- IP over ATM
- PPP over ATM
- Native ATM through Winsock 2.0

For more information about ATM, see "Asynchronous Transfer Mode" in the *Windows 2000 Internetworking Guide*.

LANE LAN Emulation (LANE) is a method by which other protocols (not just TCP/IP) that only understand connectionless media can communicate over ATM. It allows ATM to utilize both legacy networks and applications. Traditional LAN-aware applications and protocols can communicate over an ATM network without modification.

LANE consists of two primary components: the LANE client (Atmlane.sys) and the LANE services. The LANE client allows LAN protocols and LAN-aware applications to function as if they were communicating with a traditional LAN. The LANE client communicates LAN commands to the network protocols and native ATM commands to the ATM protocol layer. The LANE services are a group of ATM components, usually on a switch that supports LAN emulation.

IP Over ATM IP over ATM is a means of carrying IP packets over an ATM network. IP over ATM uses the connection-oriented properties of ATM to overcome the connectionless nature of IP. It functions in a manner similar to LANE. A central IP server (called an ATMARP server) maintains a database of IP and ATM addresses, and provides configuration and broadcast services. IP over ATM is a group of components that doesn't reside in one place. The services are not usually on an ATM switch. IP over ATM server services are provided with Windows 2000 and can reside on a Windows 2000 server.

IP over ATM is a small layer between the ATM protocol and the TCP/IP protocols. The client emulates standard IP to the TCP/IP protocol at its top edge and uses native ATM commands to the ATM protocol layers underneath.

IP over ATM is handled by two primary components: the IP over ATM server (Atmarps.sys) and the IP over ATM client (Atmarpc.sys). The IP over ATM server is composed of an ATMARP server and Multicast Address Resolution Service (MARS). The ATMARP server provides services that emulate standard IP functions, while MARS provides broadcast and multicast services. Both services maintain IP address databases.

ATM Over xDSL Digital Subscriber Line (xDSL) technology is a means by which plain old telephone service (POTS) can be used to send ATM cells over a pair of copper wires to the central station of a phone company. ATM over xDSL offers high-speed network access from the home and small office environment. Several standards are being developed in these areas, including asymmetric digital subscriber line (ADSL) and universal ADSL (UADSL). These technologies use the local loop, the copper wires that connect the local central office in a user's neighborhood to the customer's phone jack. In many areas, this local loop connects directly to an ATM core network run by a telephone company.

ATM over xDSL service preserves the high-speed characteristics and QoS guarantees available in the core ATM network without changing protocols. This creates the potential for an end-to-end ATM network to the residence or small office.

Point-to-Point Protocol (PPP) over this end-to-end architecture adds functionality and usefulness. PPP allows necessary features such as authentication, encryption and compression. To support these architectures (such as residential broadband, PPP over ATM), Windows 2000 has additional components. Ndttsp.tsp is a TAPI service provider that allows NDIS proxy to interface with call control through TAPI. Ndproxy.sys provides call control over connection-oriented media.

Native ATM Access Through Winsock 2.0 Applications can directly use Winsock 2.0 to gain access the ATM protocols natively. Applications that using native ATM can access QoS guarantees such as bandwidth, and latency.

For more information about ATM, see "Asynchronous Transfer Mode" in the *Windows 2000 Internetworking Guide*.

NWLink

NWLink is a Microsoft-compatible IPX/SPX protocol for Windows 2000. NWLink does not allow a computer running Windows 2000 to access files or printers shared on a NetWare server, or to act as a file or print server to a NetWare client. To access files or printers on a NetWare server, a redirector must be used, such as the Client Service for NetWare on Microsoft® Windows® 2000 Professional, or the Gateway Service for NetWare on Microsoft® Windows® 2000 Server.

NWLink is useful if there are NetWare client/server applications running that use Winsock or NetBIOS over IPX/SPX protocols. A Microsoft-compatible NetWare client, NWLink can be run on a Windows 2000 Server or Windows 2000 Professional computer to access the server portion on a NetWare server.

NetWare NetBIOS Link (NWNBLink) contains Microsoft enhancements to NetBIOS. The NWNBLink component is used to format NetBIOS-level requests and pass them to the NWLink component for transmission on the network.

For more information about NetWare IPX/SPX, see "Interoperability with NetWare" in the *Windows 2000 Internetworking Guide*.

NetBEUI

NetBEUI (NetBIOS Extended User Interface) was originally developed as a protocol for small departmental LANs of 20 to 200 computers. NetBEUI is not routable since it doesn't have a network layer. NetBEUI is included with Windows 2000 Server and Windows 2000 Professional. It is primarily a legacy protocol to support existing workstations that have not been upgraded to Windows 2000.

For more information about NetBEUI, see "NetBEUI" in the *Windows 2000 Internetworking Guide*.

AppleTalk

AppleTalk is a protocol suite developed by Apple Computer Corporation for communication between Macintosh computers. Windows 2000 includes support for AppleTalk which allows Windows 2000 to be a router and a dial-up server. Support is natively provided as a service for file sharing and printer sharing.

For more information about AppleTalk, see "Services for Macintosh" in the *Windows 2000 Internetworking Guide*.

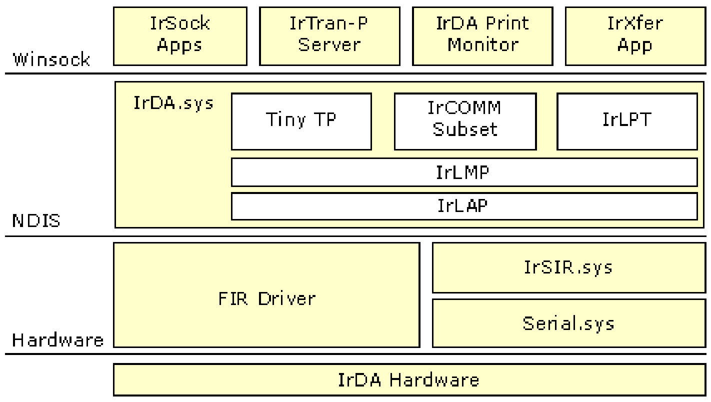
DLC

Data Link Control (DLC) was originally developed for IBM mainframe communications. The protocol was not designed to be a primary protocol for network use between personal computers. The other use of DLC is to print to Hewlett-Packard printers connected directly to networks. Network-attached printers use the DLC protocol because the received frames are easy to disassemble and because DLC functionality can easily be coded into read-only memory (ROM). DLC's usefulness is limited because it doesn't directly interface with the Transport Driver Interface layer. DLC needs to be installed only on those network machines that perform these two tasks, such as a print server sending data to a network Hewlett Packard printer. Clients sending print jobs to a network printer do not need the DLC protocol. Only the print server communicating directly with the printer needs the DLC protocol installed.

For more information about DLC, see "Data Link Control" in the *Windows 2000 Internetworking Guide*.

IrDA

Infrared Data Association (IrDA) has defined a group of short-range, high speed, bidirectional wireless infrared protocols, generically referred to as IrDA. IrDA allows a variety of devices to communicate with each other. Cameras, printers, portable computers, desktop computers, and personal digital assistants (PDAs) can communicate with compatible devices using this technology. The IrDA protocol stack is accessed using NDIS connectionless drivers. Figure B.10 illustrates the IrDA protocol architecture.



If your browser does not support inline frames, click here to view on a separate page.

Figure B.10 IrDA Architecture

The components of IrDA are:

Winsock Winsock is an API that allows Windows-based applications to access the transport protocols. The IrDA protocol stack is made available to applications by using Winsock.

IrTran-P IrTran-P is a bidirectional image transfer protocol. Windows 2000 IrTran-P receives data only and is used for cameras with infrared capability. Many cameras have digital ports and can beam infrared data to a receiving computer. That data is then placed in a user-specified (or default) directory.

IrDA Print Monitor IrDA Print Monitor is a software component that interfaces with an IrDA-connected printer to make that printer appear as any other printer to Windows 2000 users.

IrXfer IrXfer is an IrDA file transfer application. Files can be dragged and dropped from the desktop to another computer. The Windows 2000 implementation of IrXfer has bidirectional transfer capabilities.

Tiny TP Tiny TP is a flow control mechanism for IrDA. Tiny TP acts as a regulator to control the rate of data input or output. This prevents an overflow of data from occurring and creating data errors.

IrDA.sys IrDA.sys is the transport protocol stack that supports IrDA. It provides support for applications through Winsock to the NDIS layer.

IrCOMM IrCOMM is a software component that supports IrTran-P. IrCOMM uses Winsock, and is interfaced by default to the IrTran-P server. The IrTran-P server must be disabled if other applications need to use the IrCOMM port.

IrLPT IrLPT is the protocol support that is used by IrDA Print Monitor. IrLPT enables printing directly from IrDA devices to IrDA printers.

IrLMP Infrared Link Management Protocol is used to multiplex various connections over one IrDA link. Multiplexing is the technique of splitting data from various sources into time slices and sending the data slices in sequence to a destination.

IrLAP Infrared Link Access Protocol is a media access control software component that determines which component can access the media during each time slice.

FIR Driver A Fast Infrared driver (FIR) is a miniport driver provided by a hardware vendor to link hardware devices on the lower side of the protocol stack to the transport protocol above, such as TCP/IP or IPX/SPX. FIR devices can exchange data up to 4 megabytes (MB) per second. All FIR devices are also required to support serial transmission using Serial Infrared driver (SIR).

IrSIR.sys Serial Infrared driver is a Microsoft-provided miniport driver. It is an alternate driver to the Fast Infrared driver and can be used only in combination with Serial.sys. The maximum data transfer rate is 115.2 kilobytes per second (Kbps). In combination with Serial.sys, IrSIR.sys provides support for serial ports.

Serial.sys Serial.sys is used to connect infrared devices to the IrSIR.sys driver above in the protocol stack and the hardware device below. It is a software driver that sends and receives data from a hardware device and presents it to the IrSIR.sys driver in a format that conforms to the requirements of IrSIR.sys.

Transport Driver Interface

The Transport Driver Interface (TDI) is a common interface for drivers (such as the Windows 2000 redirector and server) to use to communicate with the various network transport protocols. This allows services to remain independent of transport protocols. Unlike NDIS, there is no driver for TDI, which is a specification for passing messages between two layers in the network architecture.

Microsoft developed TDI to provide greater flexibility and functionality than is provided by existing interfaces (such as Winsock and NetBIOS). All Windows 2000 transport providers directly interface with the Transport Driver Interface. This allows the TDI to provide a consistent interface for the transport protocols. The TDI specification describes the set of functions and call mechanisms by which transport drivers and TDI clients communicate. The specific software requirements on both sides of the interface are provided by adherence to the TDI specification. Some applications, such as NetBEUI, do not directly interface to the TDI layer.

Emulator Modules

Emulator Modules provide a single common interface to all Windows 2000 transport drivers. This simplifies the task of transport driver development since only the interface needs to be coded. The transport drivers provide the Transport Driver Interface. Therefore, they can only be used by applications that can use TDI. Some older applications are written to use older existing interfaces. Windows 2000 includes emulator modules for the most popular existing network interfaces (such as NetBIOS). Emulator modules provide a mapping layer between network interfaces and TDI-compliant protocols.

Network Application Programming Interfaces

An Application Programming Interface (API) is a set of routines that an application program uses to request and carry out lower-level services performed by the operating system. Windows 2000 network APIs include:

- Winsock API
- NetBIOS API
- Telephony API
- Messaging API
- WNet API

Winsock API

Winsock is an API that allows Windows-based applications to access the transport protocols. Winsock in Windows 2000 is a protocol-independent networking API. Winsock is the Windows 2000 implementation of the widely-used Sockets API, the standard for accessing datagram and session services over TCP/IP, NWLink IPX/SPX NetBIOS, and AppleTalk. Applications written to the Winsock interface include File Transfer Protocol (FTP) and Simple Network Management Protocol (SNMP). Winsock performs the following:

- Provides a familiar networking API for programmers using Windows or UNIX.
- Offers binary compatibility between the heterogeneous, Windows-based TCP/IP stack and utility vendors.
- Supports both connection-oriented and connectionless protocols.

Windows 2000 includes Winsock 1.1 support. Winsock 2.0 extends the Winsock 1.1 interface to provide access to networks using protocols other than TCP/IP, such as NetWare and AppleTalk. Winsock 2.0 provides the following enhancements over Winsock 1.1:

- Name registration and resolution.

  Winsock 2.0 provides an interface that applications can use to access many different namespaces, such as Domain Name System (DNS), Novell Directory Services (NDS), and X.500.
- Support for real-time multimedia communications.

  Winsock supports several multimedia enhancements, including Quality of Service (QoS)
- Protocol-independent multipoint and multicast.

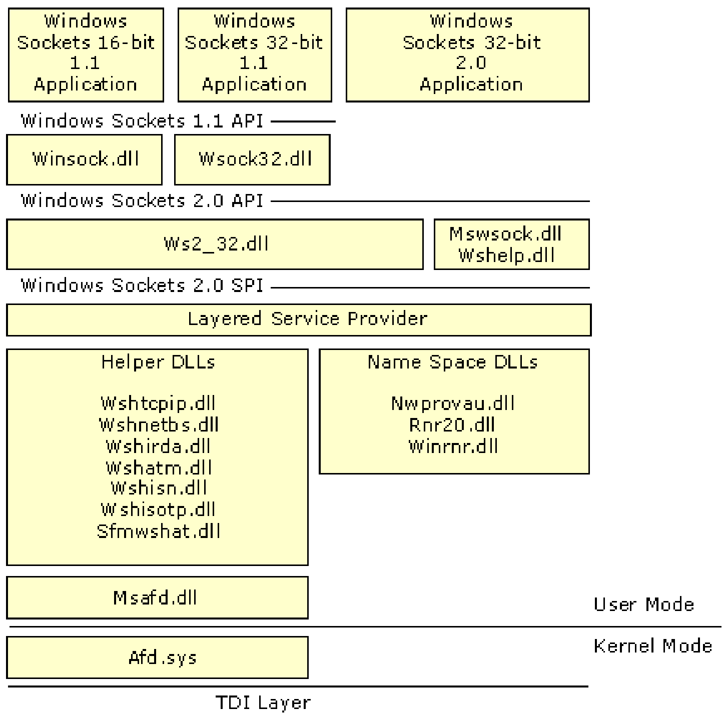
  Winsock 2.0 enables applications to take advantage of the multipoint and multicast capabilities of transport stacks.

Winsock Architecture

Winsock 2.0 is a Windows Open Systems Architecture (WOSA)–compliant interface that enables a front-end application and a back-end service to communicate. The Winsock 2.0 interface includes the following components:

- The Winsock 1.1 Application Programming Interface (API)
- The Winsock 2.0 Application Programming Interface (API)
- The Winsock 2.0 Transport Service Providers
- Layered Service Providers

Figure B.11 shows the Winsock 2.0 architecture.



If your browser does not support inline frames, click here to view on a separate page.

Figure B.11 Winsock 2.0 Architecture

Winsock Files

Table B.1 contains a list of files that Winsock uses to function. The table lists the files in order of the layer that they support and gives a brief description of their function.

Table B.1 Winsock Files

Winsock DLLs	Description
Winsock.dll	16-bit Winsock 1.1
Wsock32.dll	32-bit Winsock 1.1
Ws2_32.dll	Main Winsock 2.0
Mswsock.dll	Microsoft extensions to Winsock. Mswsock.dll is an API that supplies services that are not part of Winsock.
Ws2help.dll	Platform-specific utilities. Ws2help.dll supplies operating system–specific code that is not part of

	Winsock.
Wshtcpip.dll	Helper for TCP
Wshnetbs.dll	Helper for NetBT
Wshirda.dll	Helper for IrDA
Wshatm.dll	Helper for ATM
Wshisn.dll	Helper for Netware
Wshisotp.dll	Helper for OSI transports
Sfmwshat.dll	Helper for Macintosh
Nwprovau.dll	Name resolution provider for IPX
Rnr20.dll	Main name resolution
Winrnr.dll	LDAP name resolution
Msafd.dll	Winsock interface to kernel
Afd.sys	Winsock kernel interface to TDI transport protocols

Winsock 1.1 API

Winsock 1.1 API is a thunk layer. A thunk layer translates the output from a component into a form another component can use. Winsock 1.1 layer commands are converted to Winsock 2.0 layer commands to allow backward compatibility for legacy applications.

Winsock 2.0 API

Winsock 2.0 API is the interface for Winsock 2.0. For example, it helps Winsock 2.0 to add new APIs (such as Generic Quality of Service). Winsock 2.0 API is located between the Winsock 2.0 dynamic link library (DLL) and a Winsock 2.0 application.

Winsock 2.0 SPI Transport Service Providers

Transport service providers give applications a consistent interface for accessing multiple transport protocols. Located above the transport service provider, the Winsock 2.0 DLL takes requests from applications and sends those requests to the transport service provider. The Winsock 2.0 DLL also provides traffic management. The transport service provider can support one or more transport protocols.

Layered Service Provider Layer

An optional Layered Service Provider layer can be inserted between the Winsock 2.0 DLL and the underlying protocol stack if required by an application. It can extend the underlying protocol stack by providing additional services such as authentication, encryption, or proxy server services.

Winsock Helper DLLs

Winsock helper DLLs provide specific software components to assist Winsock 2.0. Transport protocols such as TCP, ATM, and IrDA have DLLs that supply the necessary program code to support Winsock.

Winsock 2.0 Name Resolution Providers

Name resolution providers enable server and client applications to use a consistent interface for multiple name services. Services register with the Winsock DLL, and client applications send requests for the names of those services to the Winsock DLL. The Winsock DLL manages registration and loading of name resolution providers and sends name resolution operations to the correct provider. Finally, the provider implements an interface with existing name services, such as DNS.

Generic Quality of Service and Resource Reservation Protocol

Connectionless networks (such as Ethernet networks) make only a best effort to deliver packets to their destination. There is no guarantee that packets will arrive, or that they will arrive in the correct order. Instead, protocols such as TCP/IP were developed to ensure retransmission of lost packets and to ensure that out-of-order packets could be reassembled in the correct order. This is sufficient for most applications, such as e-mail. However, for newer applications, such as real-time audio and video, packets must arrive on time and in order or the transmission might be garbled.

Connection-oriented networks enable applications to request certain levels of service, such as bandwidth and reliability, for specific connections. Additionally, they enable computers to set up several different connections with several different qualities of service. For example, on a connection-oriented network, two simultaneous connections can support both a high-delay low-bandwidth connection to send e-mail and a high-bandwidth, low-delay connection for a videoconferencing application.

Windows 2000 makes different service levels possible through its Generic Quality of Service (GQoS) APIs and its support for the Resource Reservation Signaling Protocol (RSVP). Applications can request different network characteristics for a connection. RSVP then handles those requests by attempting to make bandwidth reservations for that connection.

Generic Quality of Service

The GQoS APIs in Winsock 2.0 provide access to most QoS levels of service. The underlying QoS providers make it possible to utilize these levels of service directly from the GQoS APIs. Applications can make calls to GQoS APIs and request attributes such as:

- Peak bandwidth (average or peak bit rate available).
- Latency (the maximum acceptable delay between transmission of a bit and its receipt by the receiver).
- Delay variation (the difference between a packet's minimum and maximum delay).

Figure B.12 shows the architecture of GQoS.

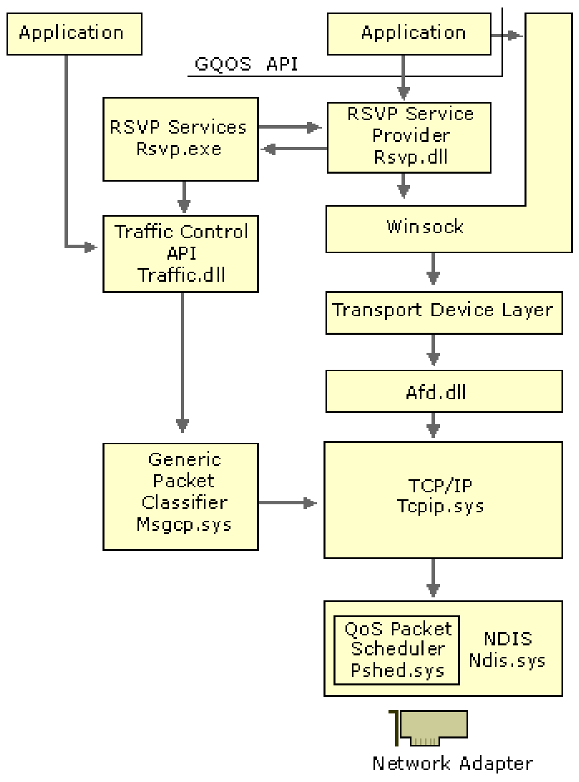


Figure B.12 GQoS Architecture

The GQoS components are:

- RSVP Service Provider

  The QoS component that invokes nearly all QoS functions and services. RSVP Service Provider (rsvpsp.dll and rsvp.exe) starts traffic control and implements, maintains, and handles RSVP signalling for all of Windows 2000 QoS functions.
- Traffic Control API

  A programmatic interface for the traffic control components that regulate network traffic on local hosts. It regulates traffic internally (within the kernel) and on the network. (It also prioritizes and queues packets based on transmission priority.)
- Generic Packet Classifier (GPC)

  Classifies and prioritizes packets, it has the ability to provide lookup tables and classification services within the network stack.
- Afd.sys

  This Winsock Kernel Interface provides access to the TDI transports.
- QoS Packet Scheduler

  This traffic control module regulates how much of the data an application is allowed to transmit at one time, thereby enforcing QoS parameters that are set for a particular flow.

Quality of Service Admission Control Service

The QoS Admission Control Service is responsible for regulating subnet usage for QoS-enabled components.

Figure B.13 illustrates the QoS Admission Control Service architecture.

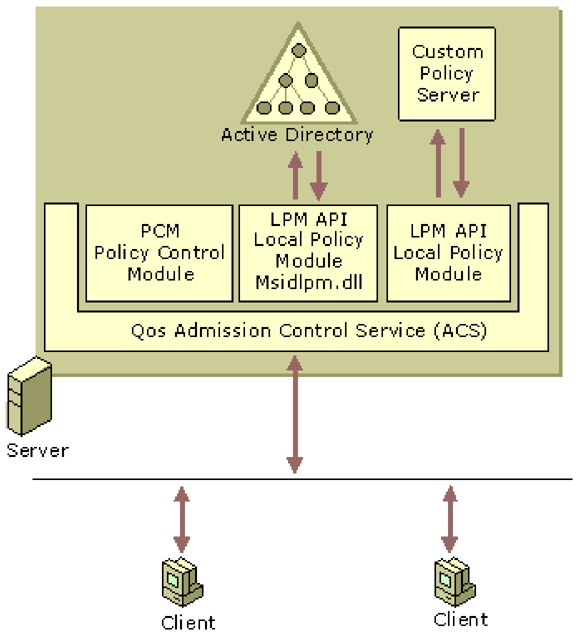

Active Directory
Custom Policy Server
PCM Policy Control Module
LPM API Local Policy Module Msidlpm.dll
LPM API Local Policy Module
Qos Admission Control Service (ACS)
Server
Client
Client

Figure B.13 QoS Admission Control Service

The QoS ACS administers subnet bandwidth resources that are necessary to ensure QoS transmission of data. The QoS ACS operates on a Windows 2000 Server residing on a subnet. On the shared segments, all QoS reservation messages are routed through the QoS ACS, so subnet clients can share their bandwidth and the administration of bandwidth allocation can be centralized. The QoS ACS sends messages, called *beacons*, to let other clients on the network know it is present and ready to receive subnet bandwidth reservation requests. QoS Admission Control Service components govern QoS-enabled applications. The QoS ACS must be installed in a server that does not have QoS components present.

The QoS ACS components are:

- Quality of Service Admission Control Service (QoS ACS)

  A QoS component that regulates subnet usage for QoS-enabled applications. The QoS ACS exerts its control over QoS-aware applications or clients by placing itself within the RSVP message path. The QoS ACS intercepts Resource Reservation Protocol (RSVP) and Reservation (RESV) messages and passes the messages with the user information to Local Policy Modules for authentication. RSVP messages are sent to request transmission characteristics and RESV messages confirm that the transmission characteristics can be granted.

- Policy Control Module (PCM)

  Mediates the interaction between the QoS ACS and LPMs. PCMs send user information to each LPM and gathers all the responses, then performs logical checks on the information. The PCM gathers the information and sends it as one response to QoS ACS.

- Local Policy Module (LPM)

  The API that communicates with the QoS Admission Control Service. The LPM API also specifies how LPMs are registered and initialized within the construct of QoS ACS.

- Active Directory

  Provides a single point of management for Windows-based policies, user accounts, clients, servers, and applications. In QoS, Active Directory stores information about the levels of service that GQoS uses.

- Custom Policy Server

  A third-party component that can be used to store policies for GQoS levels of service.

Applications request QoS levels of service. The RSVP signaling provider negotiates with the network for the requested levels of service. The packet classifier and scheduler determine when to send the packets and with what priority. Finally, the QoS-aware router forwards the packets as requested.

For more information about QoS and RSVP, see "Quality of Service" in this book.

Telephony API

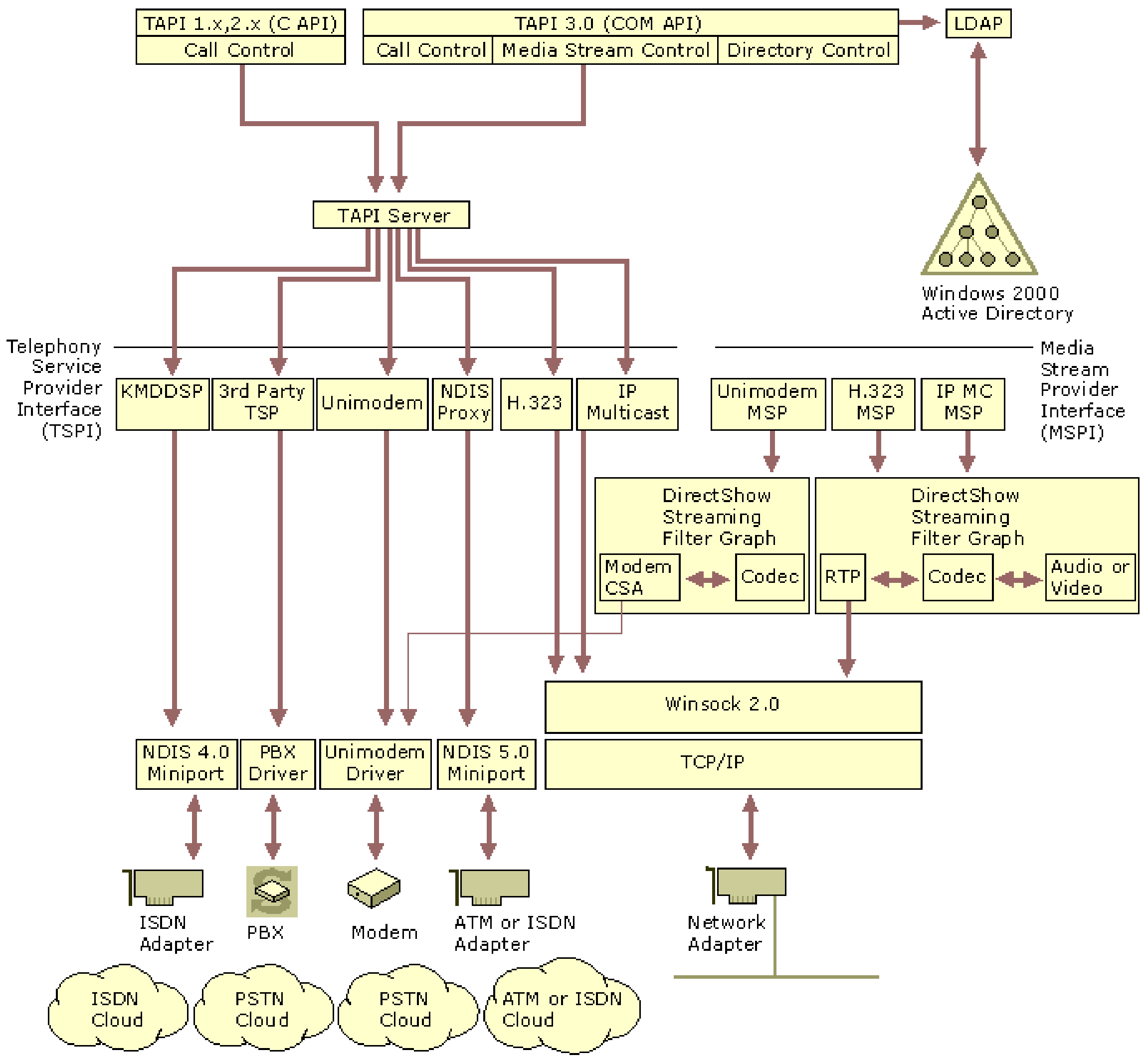
Telephony is a technology that integrates computers with telephone networks. With telephony, people can use their computers to take advantage of a wide range of communications features and services over a telephone line.

The Telephony API (TAPI) allows programmers to develop applications that provide personal telephony to users. TAPI supports both speech and data transmission, and allows for a variety of terminal devices. It supports complex connection types and call-management techniques such as call conferencing, call waiting, and voice mail. TAPI allows all elements of telephone usage, from the simple dial-and-speak call to international e-mail, to be controlled within applications developed for Microsoft® Win32® application programming interfaces.

Many applications can utilize telephony:

- Multicast multimedia IP conferencing.
- Voice calls over the Internet.
- Call center applications capable of tracking multiple agents.
- Basic voice calls on the Public Switched Telephone Network (PSTN).
- Private Branch Exchange (PBX) control.
- Interactive voice response (IVR) systems.
- Voice mail.

Figure B.14 shows the major components of the TAPI architecture.



If your browser does not support inline frames, click here to view on a separate page.

Figure B.14 TAPI Architecture

TAPI 3.0 COM API The TAPI 3.0 API is implemented as a suite of Component Object Model (COM) interfaces. This allows developers to write TAPI-enabled applications in any COM-aware language, such as Java, Microsoft® Visual Basic, or Microsoft® Visual C++®. TAPI 3.0 supports two classes of service providers: telephony and media.

TAPI 2.1 COM API TAPI 2.1 COM API provides compatibility with TAPI 3.0. TAPI 2.1 is a translation layer that maps 16-bit addresses to 32-bit addresses and passes requests along to Tapi32.dll.

Tapi32.dll Tapi32.dll is a thin marshaling layer, that transfers function requests for Tapisrv.exe. When needed, it loads and calls service provider user interface DLLs.

TAPI Server TAPI server is implemented by Tapisrv.exe, the core component of TAPI. TAPI server runs as a separate service process in which all telephony service providers run. API functions communicate with the TAPI server to send service requests to telephony service providers.

Telephony Service Providers Telephony service providers are dynamic-link libraries that carry out low-level and device-specific actions needed to complete telephony tasks through hardware devices such as fax adapters, ISDN adapters, telephones, and modems. Applications link to and call functions in the TAPI dynamic-link library only; they never call the service providers directly.

Media Service Providers The Microsoft H.323 Telephony Service Provider and its associated Media Service Provider (MSP) allow TAPI-enabled applications to engage in multimedia audio/video sessions with any H.323-compliant terminal on a local area network (LAN) or the Internet. Examples include:

- The multicast conference service provider. Uses IP multicast techniques to provide efficient multiparty audio and video conferencing facilities over IP networks, intranets and the Internet.
- The NDIS Proxy Service Provider. Offers a TAPI interface to wide area network (WAN) devices, such as ISDN or ATM.
- TAPI, which has a Remote Service Provider to support client/server telephony. The Remote SP provides TAPISRV telephony service extensions for client access to TAPI devices that reside on networked server machines.
- The TAPI Kernel-Mode Service Provider. Communicates with NDIS components in order to provide a TAPI interface to NDISWAN drivers.
- Unimodem 5. A Telephony Service Provider that provides an abstraction for modem devices so that applications can operate transparently across a wide variety of modems.
- Third-party hardware and software vendors. Can write Telephony Service Providers that are compatible with TAPI for additional applications.

For more information about TAPI, see "Telephony Integration" in the *Windows 2000 Internetworking Guide*.

NetBIOS API

NetBIOS is a standard application programming interface in the personal-computing environment. NetBIOS is used for developing client/server applications. NetBIOS has been used as an interprocess communication (IPC) mechanism since its introduction. It is included with Windows 2000 to support legacy applications.

A NetBIOS client/server application can communicate over various protocols:

- NetBEUI Frame protocol (NBF)

  NetBEUI is a NetBIOS-compatible transport. NBF was designed for small networks of 50 to 200 computers. Since NetBEUI has no networking layer, it is not routable. It is included with Windows 2000 to support legacy networks.
- NetBIOS over TCP/IP (NetBT)

  Since NetBEUI is not routable, alternate means were created to transport it through routers. RFCs 1001 and 1002 defined methods by which NetBEUI can be transported in IP packets. With this technique, NetBEUI networks can be connected.
- NWLink NetBIOS (NWNBLink)

  NetWare NetBIOS Link (NWNBLink) contains Microsoft enhancements to NetBIOS. The NWNBLink component is used to format NetBIOS-level requests and pass them to the NWLink component for transmission on the network.

NetBIOS uses the following components:

- Netapi32.dll. Linked to the NetBIOS emulator to allow communication with TCP/IP, NetBEUI, and NWLink. It shares the address space of the NetBIOS user-mode application.
- NetBIOS emulator. Provides the NetBIOS mapping layer between NetBIOS applications and the TDI-compliant protocols.

The Winsock APIs can also be used to access the NetBIOS protocols.

For more information about NetBIOS, see "NetBEUI" in the *Windows 2000 Internetworking Guide*.

Messaging API

The Messaging Application Programming Interface (MAPI) is an industry standard that enables applications to interact with many different messaging services using a single interface. MAPI is a set of API functions that allow messaging clients, such as Microsoft Exchange, to interact with various message service providers, such as Microsoft Mail, and Fax.

For more information about MAPI, see the Platform Software Development Kit (SDK).

WNet API

WNet APIs provide Windows networking (WNet) functions that allow networking capabilities for applications. Also known as the Win32 APIs, applications that are created by using Wnet APIs function independently from the network on which they operate. By using WNet APIs, applications can be developed that run successfully on all platforms while still able to take advantage of unique features and capabilities of any particular platform.

Network services are one of many categories of services that the Win32 API's can provide. Requests for network services are provided by the Multiple Provider Router. Multi Provider Routing receives WNet commands, determines the appropriate redirector, and passes the command to that redirector. Multi Provider Routing then routes the requests for network service to the appropriate provider for transmission over the network.

For more information about WNet APIs, see the Platform Software Development Kit (SDK).

Other Network APIs

Windows 2000 uses many APIs in addition to the ones that have been covered:

- The firewall API is a low-level API that allows greater and more complex filtering than the Filter Driver API.
- The Dynamic Host Configuration Protocol (DHCP) API allows applications to request DHCP options.
- The Multicast Address Dynamic Client Allocation Protocol (MADCAP) API allows applications to request multicast addresses.
- The DHCP Server Call API allows custom extensions to a DHCP server.
- The IP Helper API (IPHLP API) is a public API that provides TCP/IP information to Win32 applications.
- The Multiple Provider Router API (MPR API) is used to configure and administer routers.
- The Filter Driver API allows configuration of packet filtering for IP traffic.
- The Message Queuing API provides loosely-coupled, reliable communications. Message Queuing can be used for transaction processing to communicate transaction status.

For more information about network APIs, see the Platform Software Development Kit (SDK).

Interprocess Communication

Interprocess communication (IPC) allows bidirectional communication between clients and servers using distributed applications. IPC is a mechanism used by programs and multi-user processes. IPCs allow concurrently running tasks to communicate between themselves on a local computer or between the local computer and a remote computer.

IPC mechanisms are used to support distributed processing. Applications that split processing between networked computers are called distributed applications. The split portions of a distributed application can be located on the same machine or on separate machines. A client/server application uses distributed processing, in which processing is divided between a workstation (the client) and a more powerful server. The client portion is sometimes referred to as the front end and the server portion is referred to as the back end. The client portion of a client/server application can consist of just the user interface to the application. However, the application can be split at various places in the distributed application. It runs on the client workstation and takes a lesser amount of processing power. For example, the client portion might handle only screen graphics, mouse movements, and keystrokes.

Multi-tier applications are an extension of the basic client/server model. This common distributed business model is sometimes called the three-tier model. It is composed of three parts: the client tier, the components tier and the server tier.

For example, your company's payroll department uses an application to print paychecks. When a payroll employee runs a client application, the application starts a business-rules server in the component tier. The component tier is composed of various components such as business rules, transaction management, and other business logic components.

The business rules server organizes the appropriate components for the task. The server application connects to a database server in the database tier and retrieves employee records such as salary information. The business-rules server transforms the payroll information into the final output and returns it to the client.

The application server can then process information that the client computer normally does in the client/server model. The application server handles the business or application logic and communicates with the back-end database, usually with structured query language (SQL). The server or back-end database often utilizes large amounts of data storage, computing power, and specialized hardware. The component parts make it easier to manage, deploy, and update these objects.

The goal of distributed processing is to move the actual application processing from the client to the servers that have the power to run large applications. While running, the client portion formats requests and sends them to the server for processing. Servers run the request and pass the result back to the client.

There are a number of ways to establish this connection. The Windows 2000 operating system provides many different Interprocess Communication (IPC) mechanisms.

Distributed Component Object Model

In addition to supporting component object model (COM) for interprocess communication on a local computer, Windows 2000 supports the distributed component object model (DCOM). DCOM is a system of software objects designed to be reusable and replaceable. Objects are software components that can perform and support applications. The objects support sets of related functions such as sorting, random-number generation, and database searches. Each set of functions is called an interface, and each DCOM object can have multiple interfaces. When applications access an object, they receive an indirect pointer to the interface functions. The pointer has information on the location of an object. After receiving this pointer, the calling application doesn't need to know where the object is or how it does its job since the pointer directs the calling application to it.

DCOM allows processes to be efficiently distributed to multiple computers so that the client and server components of an application can be placed in optimal locations on the network. Processing occurs transparently to the user because DCOM handles this function. Thus, the user can access and share information without needing to know where the application components are located. If the client and server components of an application are located on the same computer, DCOM can be used to transfer information between processes. DCOM is platform independent and supports any 32-bit application that is DCOM-aware.

Advantages of Using DCOM

DCOM is a preferred method for developers to use in writing client/server applications for Windows 2000. With DCOM, interfaces to software objects can be added or upgraded, so applications aren't forced to upgrade each time the software object changes. Objects are software entities that perform specific functions. These functions are implemented as dynamic-link libraries so that changes in the functions, including new interfaces or the way the function works, can be made without rewriting and recompiling the applications that call them.

Windows 2000 supports DCOM by making the implementation of application pointers transparent to the application and the object. Only the operating system needs to know if the function called is handled in the same process or across the network. This frees the application from concerns with local or remote procedure calls. Administrators can choose to run DCOM applications on local or remote computers, and can change the configuration for efficient load balancing.

Your application might support its own set of DCOM features. For more information about configuring your application to use DCOM, see your application's documentation.

DCOM builds upon remote procedure call (RPC) technology by providing an easy-to-use mechanism for integrating distributed applications on a network. A distributed application consists of multiple processes that cooperate to accomplish a single task. Unlike other interprocess communication (IPC) mechanisms, DCOM gives you a high degree of control over security features, such as permissions and domain authentication. It can also be used to start applications on other computers or to integrate Web browser applications that run on the Microsoft® ActiveX® platform.

Remote Procedure Call

Remote Procedure Call (RPC) is an interprocess communication technique to allow client and server software to communicate. The Microsoft RPC facility is compatible with the Open Group's Distributed Computing Environment (DCE) specification for remote procedure calls and is interoperable with other DCE-based RPC systems, such as those for HP-UX and IBM AIX UNIX–based operating systems. The RPC facility is compatible with the Open Group specification.

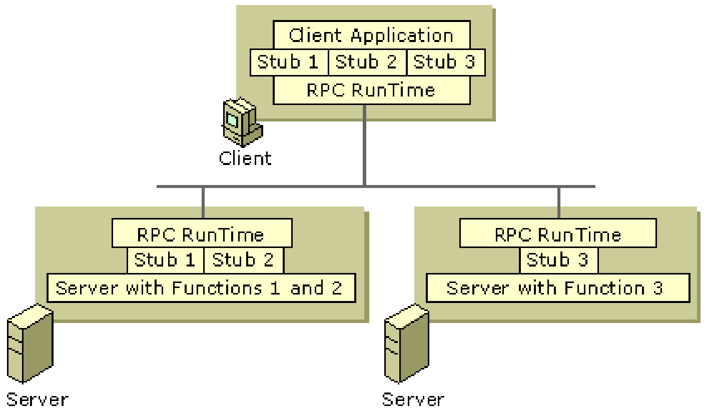
The Microsoft RPC mechanism is unique in that it uses other IPC mechanisms, such as named pipes, NetBIOS, or Winsock, to establish communications between the client and the server. With the RPC facility, essential program logic and related procedure code can exist on different computers, which is important for distributed applications.

RPC is based on the concepts used for creating structured programs, which can be viewed as having a backbone to which a series of ribs can be attached. The backbone is the mainstream logic of the program that rarely changes. The ribs are the procedures that the backbone calls upon to do work or perform functions. In traditional programs, these ribs are statically linked to the backbone and stored in the same executable file. RPC places the backbone and the ribs on different computers.

Windows 2000 uses dynamic link libraries (DLLs) to provide procedure code and backbone code. This enables the DLLs to be modified or updated without changing or redistributing the backbone portion.

Client applications are developed with specially-compiled stub libraries that are provided by the application program. In reality, these stubs transfer the data and the function to the RPC run-time module. This module is responsible for finding the server that can satisfy the RPC command. Once found, the function and data are sent to the server, where they are picked up by the RPC run-time component on the server. The server builds the appropriate data structure, and calls the function.

The function interprets the call as coming from the client application. The server may impersonate the client. For security, RPC can use NTLM, Kerberos, or Secure Sockets Layer (SSL). When the function is completed, any return values are collected, formatted, and sent back to the client through the RPC run-time module. When the function returns to the client application, it has the appropriate returned data or an indication that the function failed. Figure B.15 illustrates Remote Procedure Calls.

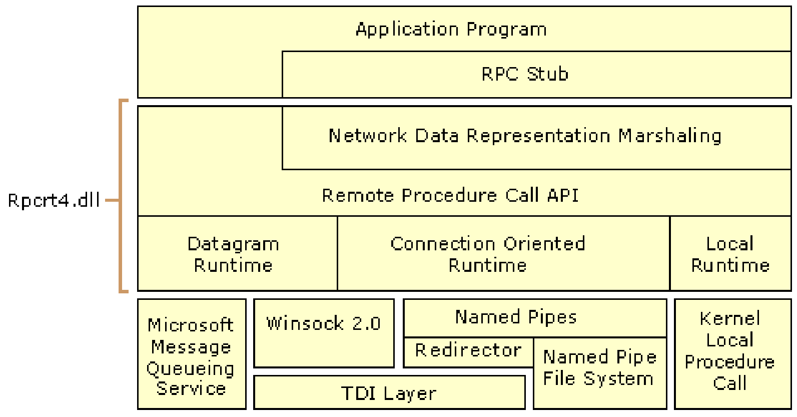


If your browser does not support inline frames, click here to view on a separate page.

Figure B.15 Remote Procedure Calls

If the client and server portions of the application are on the same computer, local procedure calls (LPCs) can be used to transfer information efficiently between processes. This makes RPC a flexible and portable IPC choice.

Figure B.16 illustrates the software components of RPC.


Application Program
RPC Stub
Rpcrt4.dll
Network Data Representation Marshaling
Remote Procedure Call API
Datagram Runtime
Connection Oriented Runtime
Local Runtime
Microsoft Message Queueing Service
Winsock 2.0
TDI Layer
Named Pipes
Redirector
Named Pipe File System
Kernel Local Procedure Call

If your browser does not support inline frames, click here to view on a separate page.

Figure B.16 Windows 2000 RPC Client and Workstation Architecture

The components of RPC are:

RPC Stub Part of an application executable file or a DLL that is generated by the Microsoft Interface Description Language (MIDL) compiler specifically for each interface.

Network Data Representation Marshaling Marshaling is the process of packaging and unpackaging parameters into Network Data Representation (NDR) format for communication over the network.

Remote Procedure Call APIs A series of protocol-independent APIs responsible for establishing connections and security as well as registering servers, naming, and endpoint resolution.

Datagram Runtime A connectionless RPC protocol engine that transmits and receives requests using connectionless protocols, such as UDP.

Connection-Oriented Runtime A connection-oriented RPC protocol engine that transmits and receives requests using connection-oriented protocols, such as TCP.

Local Runtime A local RPC protocol engine that transmits and receives RPC requests between processes on the local computer.

RPC Name Resolution

The function of RPC name resolution is to allow clients to find RPC servers. Servers on a Windows 2000 network send messages to the Active Directory directory service by a locator, which is a software service that runs on RPC servers. The locator exports bindings, interfaces, protocols, and endpoints for servers running on the local computer. This server information is stored in Active Directory in System\RPCServices. When a client wants to find a server, it queries Active Directory. The client receives information, including server names, protocols, and endpoints that were placed in Active Directory by the server. With this information, the client can directly connect to the server. Windows 2000 still supports named pipes, mailslots and broadcast messaging for name resolution by NetBIOS.

For more information about RPC name resolution, see "Service Publication in Active Directory" in the *Microsoft® Windows® 2000 Server Resource Kit Distributed Systems Guide*.

Named Pipes and Mailslots

Named pipes and mailslots are high-level interprocess communication mechanisms used by networked computers. Named pipes and mailslots are written as file system drivers, so implementation of named pipes and mailslots differs from implementation of other IPC mechanisms. Local processes can also use named pipes and mailslots. As with all other file systems, remote access to named pipes and mailslots is accomplished through the Common Internet File System (CIFS) redirector. A redirector intercepts file input/output (I/O) requests and directs them to a drive or resource on another networked computer. The redirector allows a CIFS client to locate, open, read, write, and delete files on another network computer running CIFS.

Named Pipes

Named pipes provide connection-oriented messaging by using pipes. Connection-oriented messaging requires that the communication occur over a virtual circuit and maintain reliable and sequential data transfer. A pipe is a portion of memory that can be used by one process to pass information to another. A pipe connects two processes so that the output of one can be used as input to the other. This technique is used for passing data between client and server. Named pipes are based on OS/2 API calls, which have been ported to the WNet APIs. Additional asynchronous support has been added to named pipes to pass data between client/server applications. Named pipes is included to provide backwards compatibility with Microsoft® LAN Manager and related applications.

The Windows 2000 operating system provides special APIs that increase security for named pipes. Using a feature called impersonation, the server can change its security identity to that of the client at the other end of the message. A server typically has more permissions to access databases on the server than a client requesting services. When the request is delivered to the server through a named pipe, the server changes its security identity to the security identity of the client. This limits the server to only those permissions granted to the client rather than its own permissions, thus increasing the security of named pipes.

Mailslots

Mailslots are a connectionless, high-level interprocess communication mechanism between networked computers, often used to locate and provide notification of services and computers. That is, mailslots are a broadcast service used for message delivery. The delivery of a message is not guaranteed, although the delivery rate on most networks is high.

The Windows 2000 operating system supports only second-class mailslots, not first-class mailslots. First-class mailslots are connection-oriented. Second-class mailslots provide connectionless messaging for broadcast messages.

A mailslot can be created on any networked computer. When a message is sent to a mailslot, the sending application specifies in the mailslot message structure whether the message is a first-class or second-class delivery. Connectionless messaging is most useful for identifying other computers or services on a network, such as the Browser service offered in the Windows 2000 operating system.

Mailslots are included to provide backward compatibility with LAN Manager applications.

Common Internet File System

The *Common Internet File System* (CIFS) is the standard way that computer users share files across corporate intranets and the Internet. An enhanced version of the Microsoft open, cross-platform Server Message Block (SMB) protocol, CIFS is a native file-sharing protocol in Windows 2000.

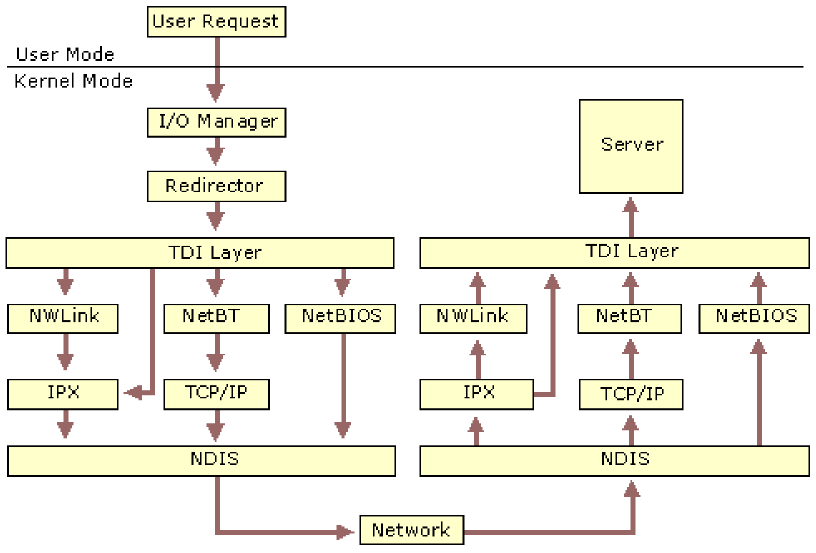
CIFS defines a series of commands used to pass information between networked computers. The redirector packages requests meant for remote computers in a CIFS structure. CIFS can be sent over a network to remote devices. The redirector also uses CIFS to make requests to the protocol stack of the local computer. The CIFS messages can be broadly classified as follows:

- Connection establishment messages consist of commands that start and end a redirector connection to a shared resource at the server.
- Namespace and File Manipulation messages are used by the redirector to gain access to files at the server and to read and write them.
- Printer messages are used by the redirector to send data to a print queue at a server and to get status information about the print queue.
- Miscellaneous messages are used by the redirector to write to mailslots and named pipes.

Some of the platforms that CIFS supports are:

- Microsoft Windows 2000, Microsoft® Windows NT®, Microsoft® Windows® 98, Microsoft® Windows® 95
- Microsoft® OS/2 LAN Manager
- Microsoft® Windows® for Workgroups
- UNIX
- VMS
- Macintosh
- IBM LAN Server
- DEC PATHWORKS
- Microsoft® LAN Manager for UNIX
- 3Com 3+Open
- MS-Net

CIFS complements Hypertext Transfer Protocol (HTTP) while providing more sophisticated file sharing and file transfer than older protocols, such as FTP. CIFS is shown servicing a user request for data from a networked server in Figure B.17.



If your browser does not support inline frames, click here to view on a separate page.

Figure B.17 CIFS Architecture

When there is a request to open a shared file, the I/O calls the redirector, which in turn requests the redirector to choose the appropriate transport protocol. For NetBIOS requests, NetBIOS is encapsulated in the IP protocol and transported over the network to appropriate server. The request is passed up to the server, which sends data back to satisfy the request.

Components in the redirector provide support for CIFS, such as:

- Rdbss.sys

  All kernel-level interactions are encapsulated in this driver. This includes all cache managers, memory managers, and requests for remote file systems so the specified protocol can use the requested server.
- Mrxsmb.sys

  This mini-redirector for CIFS has commands specific to CIFS.
- Mrxnfs.sys

  This mini-redirector for the Network File System (NFS) provides support for NFS. Mrxnfs.sys is included in Services for Unix.

In Windows NT 4.0, Windows Internet Name Service (WINS), and Domain Name System (DNS) name resolution was accomplished by using TCP port 134. Extensions to CIFS and NetBT now allow connections directly over TCP/IP with the use of TCP port 445. Both means of resolution are still available in Windows 2000. It is possible to disable either or both of these services in the registry.

Features that CIFS offers are:

Integrity and Concurrency CIFS allows multiple clients to access and update the same file while preventing conflicts by providing file sharing and file locking. File sharing and file locking is the process of allowing one user to access a file at a time and blocking access to all other users. These sharing and locking mechanisms can be used over the Internet and intranets. They also permit aggressive caching and read-ahead and write-behind without loss of integrity. File caches of buffers must be cleared before the file is usable by other clients. These capabilities ensure that only one copy of a file can be active at a time, preventing data corruption.

Optimization for Slow Links The CIFS protocol has been tuned to run well over slow-speed dial-up lines. The effect is improved performance for users who access the Internet using a modem.

Security CIFS servers support both anonymous transfers and secure, authenticated access to named files. File and directory security policies are easy to administer.

Performance and Scalability CIFS servers are highly integrated with the operating system, and are tuned for maximum system performance.

Unicode File Names File names can be in any character set, not just character sets designed for English or Western European languages.

Global File Names Users do not have to mount remote file systems, but can refer to them directly with globally significant names (names that can be located anywhere on the Internet), instead of ones that have only local significance (on a local computer or LAN). Distributed File Systems (DFS) allows users to construct an enterprise-wide namespace. Uniform Naming Convention (UNC) file names are supported so a drive letter does not need to be created before remote files can be accessed.

Basic Network Services

Network services support application programs and provide the components and APIs necessary to access files on networked computers. Both the server service and the workstation service also assist in accessing I/O requests.

Server Service

The server service is located above the TDI and is implemented as a file system driver. The CIFS server service interacts directly with other file-system drivers to satisfy I/O requests, such as reading or writing to a file. The server service supplies the connections requested by client-side redirectors and provides them with access to the resources they request. Figure B.18 shows the server service receiving a data request.

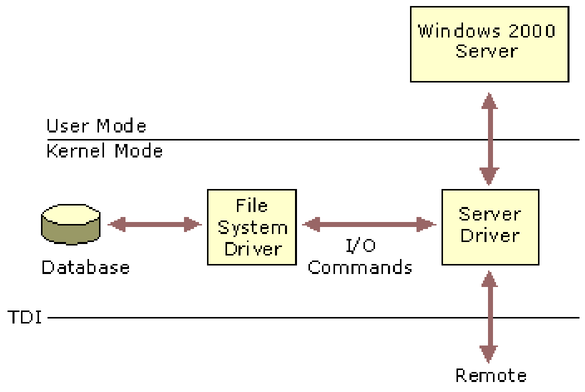


Figure B.18 Server Service

When the server service receives a request from a remote computer asking to read a file that resides on the local hard drive, the following steps occur:

1. The low-level network drivers receive the request and pass it to the server driver.
2. The server service passes the request to the appropriate local file-system driver.
3. The local file-system driver calls lower-level, disk -device drivers to access the file.
4. The data is passed back to the local file-system driver.
5. The local file-system driver passes the data back to the server service.
6. The server service passes the data to the lower-level network drivers for transmission back to the remote computer.

The server service is composed of two parts:

- Server service is a component of Services.exe. Services.exe is the Service Control Manager, where all services start. Unlike the workstation service, the server service is not dependent on the Multiple Uniform Naming Convention Provider (MUP). The MUP selects the appropriate UNC provider to handle the requests.
- Srv.sys is a file system driver that handles the interaction with the lower levels of the protocol stack and directly interacts with various file system devices to satisfy command requests, such as file read and write.

Workstation Service

All user-mode requests from the MUP go through the workstation service. This service consists of two components:

- The user-mode interface, which resides in Services.exe in Windows 2000.
- The redirector (Mrxsmb.sys), is a file-system driver that interacts with the lower-level network drivers by means of the TDI interface.

The workstation service receives the user request, and passes it to the kernel-mode redirector. The workstation service is shown in Figure B.19.

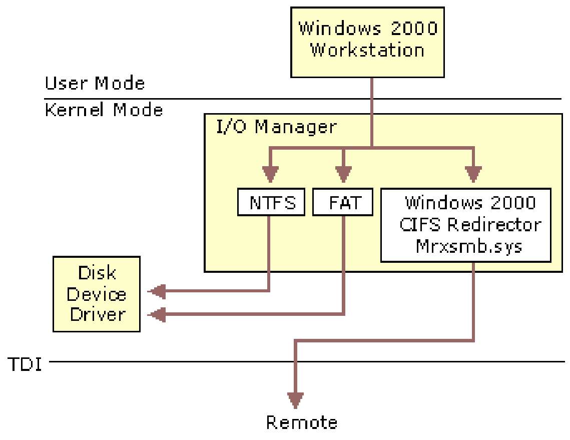


Figure B.19 Workstation Service

Windows 2000 Redirector

The redirector is a component that resides above TDI and through which one computer gains access to another computer. The Windows 2000 operating system redirector allows connection to Windows 98, Windows 95, Windows for Workgroups, LAN Manager, LAN Server, and other CIFS servers. The redirector communicates to the protocols by means of the TDI interface.

The redirector is implemented as a Windows 2000 file system driver. Implementing a redirector as a file system has many benefits:

- Allows applications to call a single API (the Windows 2000 I/O API) to access files on local and remote computers. From the I/O manager perspective, there is no difference between accessing files stored on a remote computer on the network and accessing those stored locally on a hard disk.
- Runs in kernel mode and can directly call other drivers and other kernel-mode components, such as cache manager. This improves the performance of the redirector.
- Can be dynamically loaded and unloaded, like any other file-system driver.
- Can coexist with other redirectors.

Figure B.20 shows the network architecture of the Windows 2000 Redirector.

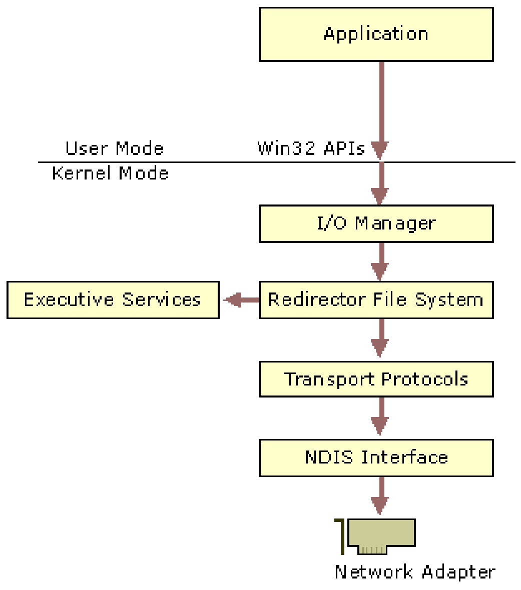


Figure B.20 Windows 2000 Redirector

Interoperating with Other Networks

Besides allowing connections to Windows 98, Windows 95, peer-to-peer networks, LAN Manager, LAN Server, and MS-Net servers, the Windows 2000 redirector can coexist with redirectors for other networks, such as Novell NetWare and UNIX networks.

Providers and the Provider-Interface Layer

For each additional type of network, such as NetWare or UNIX, you must install a component called a provider. The provider is the component that allows a computer running Windows 2000 Server or Windows 2000 Professional to communicate with the network. The Windows 2000 operating system includes several providers: Client Services for NetWare and Gateway Services for NetWare.

Client Services for NetWare is included with Windows 2000 Professional and allows a computer running Windows 2000 Professional to connect as a client to the NetWare network. The Gateway service, included with Windows 2000 Server, allows a computer running Windows 2000 to connect as a client to the NetWare network and provide gateway services between Microsoft network-based clients and Novell NetWare servers. Other provider DLLs are supplied by the appropriate network vendors.

Accessing a Remote File

When a process on a Windows 2000 computer tries to open a file that resides on a remote computer, the following steps occur:

1. The process calls the I/O manager to request that the file be opened.
2. The I/O manager recognizes that the request is for a file on a remote computer, and passes the request to the redirector file-system driver.

3. The redirector passes the request to lower-level network drivers that transmit it to the remote server for processing.

Network Resource Access

Applications reside above the redirector and server services in user mode. Like all other layers in the Windows 2000 networking architecture, there is a unified interface for accessing network resources, which is independent of any redirectors installed on the system. Access to resources is provided through the Multiple Uniform Naming Convention Provider (MUP) and the Multi-Provider Router (MPR).

Multiple Universal Naming Convention Provider

When applications make I/O calls containing Uniform Naming Convention (UNC) names, these requests are passed to the Multiple UNC Provider (MUP). MUP selects the appropriate UNC provider (redirector) to handle the I/O request.

Universal Naming Convention Names

UNC is a naming convention for describing network servers and the share points on those servers. UNC names start with two backslashes followed by the server name. All other fields in the name are separated by a single backslash. A typical UNC name appears as: *\\server\share\subdirectory\filename*.

Not all of the components of the UNC name need to be present with each command; only the share component is required. For example, the command **dir** *\\servername\sharename* can be used to obtain a directory listing of the root of the specified share. One of the design goals of the Windows 2000 networking environment is to provide a platform upon which others can build. MUP is a vital part of allowing multiple redirectors to coexist in the computer. MUP frees applications from maintaining their own UNC-provider listings. If there are multiple redirectors present there must be a means of deciding which one to use. MUP's function is to act as an arbitrator to decide the most appropriate redirector to use. Figure B.21 shows the Windows 2000 architecture of MUP.

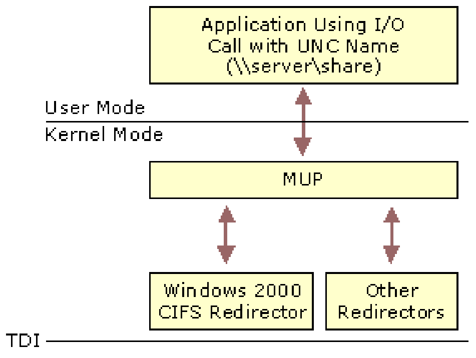

Application Using I/O Call with UNC Name (\\server\share)
User Mode
Kernel Mode
MUP
Windows 2000 CIFS Redirector
Other Redirectors
TDI

Figure B.21 MUP Architecture

MUP is a driver, unlike the TDI interface, which merely defines the way a component on one layer communicates with a component on another layer. MUP also has defined paths to UNC providers (redirectors).

I/O requests from applications that contain UNC names are received by the I/O manager, which passes the requests to MUP. If MUP has not seen the UNC name during the previous 15 minutes (this is only an approximate time period and is subject to change), MUP sends the name to each of the UNC providers registered with it. MUP is a prerequisite of the workstation service.

When a request containing a UNC name is received by MUP, it checks with each redirector to find out which one can process the request. MUP looks for the redirector with the highest registered-priority response that claims it can establish a connection to the UNC. This connection remains as long as there is activity. If there has been no request for 15 minutes (this is only an approximate time period and is subject to change) on the UNC name, then MUP negotiates to find another appropriate redirector.

Multi-Provider Router

Not all programs use UNC names in their I/O requests. Some applications use WNet APIs, which are the Win32 network APIs. The Multi-Provider Router (MPR) was created to support these applications.

MPR is similar to MUP. MPR receives WNet commands, determines the appropriate redirector, and passes the command to that redirector. Because different network vendors use different interfaces for communicating with their redirector, there is a series of provider DLLs between MPR and the redirectors. The provider DLLs provide a standard interface so that MPR can communicate with them. The appropriate DLLs take the request from MPR and communicate it to their corresponding redirector. Figure B.22 illustrates the Multi-Provider Router architecture.

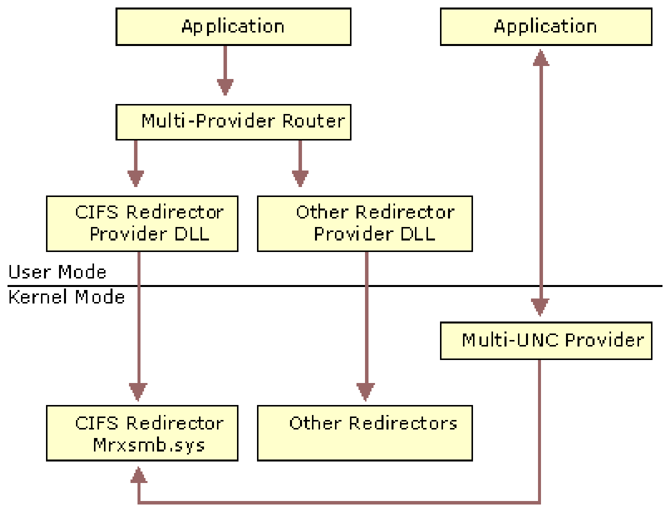


Figure B.22 Multi-Provider Router

The provider DLLs are supplied by the network-redirector vendor and are automatically installed when the redirector is installed.

Note The acronym MPR is also used for the Multi-Protocol Routing, a series of routing components supplied with Windows NT 4. In

Windows 2000, Multi-Protocol Routing has become the Routing and Remote Access Service.

Additional Resources

- For more information about Windows 2000 network programming, see the Microsoft Platform Software Development Kit (SDK) link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources .
- For more information about NDIS device driver development, see the Microsoft Platform Software Development Kit (SDK) link on the Web Resources page at http://windows.microsoft.com/windows2000/reskit/webresources .