C H A P T E R 2 6

Client/Server Communications

About this chapter

Each network environment has its own peculiarities. This chapter describes those aspects of network communication that are relevant to the proper functioning of your database server, and provides some tips for diagnosing network communication problems. It describes how networks operate, and provides hints on running the network database server under each protocol.

Network database server only
The material in this chapter applies only to the network server. You do not need to read this chapter if you are using the personal database server.

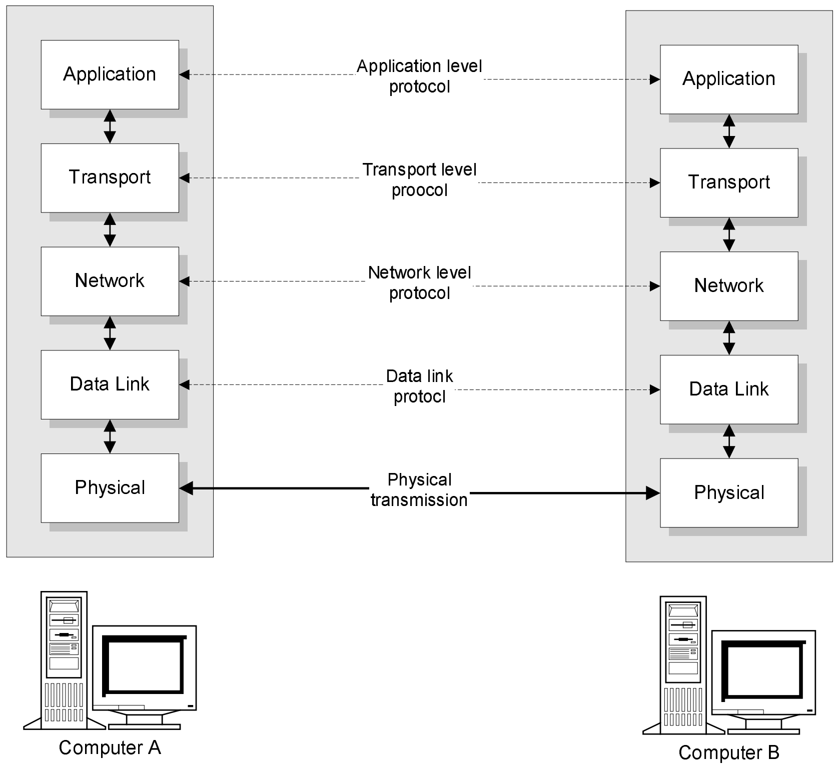
Contents

Network communication concepts

Applications in a local area network communicate using a set of rules and conventions called an application protocol. Each application is isolated from the lower-level details of how information gets transported across the network by lower-level protocols, which form a **protocol stack**.

This section provides a brief description of how protocol stacks work. Hints regarding specific network issues later in this chapter assume a basic understanding of how protocol stacks operate.

The protocol stack



The figure shows the layers of a protocol stack, based on a simplification of the OSI Reference Model of network communications.

The OSI (Open Systems Interconnection) Reference Model, developed by the International Organization for Standardization (ISO), is a step towards international standardization of network protocols. Most current networking software and hardware conforms to some extent, but not exactly, to this model. Even though conformance is incomplete, the OSI model is a valuable aid in understanding how network communications work.

How information is passed across a network

When one network application sends information to another network application (such as the database server), the information passes down one protocol stack, across the network, and up the other protocol stack to the other application.

Protocol stacks have layers

The protocol stack isolates the different functions needed for reliable data transfer. Each layer of the protocol stack is connected to the layers above and below it by an **interface**.

Each layer of a protocol stack treats information passed to it by the layer above it merely as data, labeling that data in such a way as to be identified and deciphered by the equivalent layer on the other computer. Only the physical layer is responsible for actually placing data onto the wire—all other layers provide some well-defined level of functionality, such as error detection, correction, encryption and so on.

Each layer has a protocol

Although actual data transmission is vertical (down one stack, up the other), each layer is programmed as if it were in direct communication with the corresponding layer on the other stack (peer-to-peer communication). The rules and conventions that govern each level of peer-to-peer communication are called a **protocol** for that level. There exist transport protocols, network protocols, data link protocols, and application level protocols, among others.

Protocol stack compatibility

The protocol stacks on each side of the communication must be compatible at each level for network communications to work properly. If they are not compatible, the layer on the receiving stack does not understand the information being passed to it by its corresponding layer on the sending stack.

Software that manages lower levels in a protocol stack is often called a **driver**. The different layers of a protocol stack have the following functions:

The physical layer

Your **network adapter**, or **network card**, and the network wiring form the physical layer for network communication. The higher layers in the protocol stacks ensure that software does not have to be aware of the details of network wiring or network adapter design.

The data link layer

The job of a data link layer is to handle the safe passing of information across the network at the level of individual sets of bits. At higher levels of the protocol stack, data is not described in terms of individual bits. It is at the data link layer that information is broken down to this elementary level.

The data link layer's interface with the network layer above it in the protocol stack commonly conforms to one of two specifications: the Network Driver Interface Specification (**NDIS**) developed jointly by IBM and Microsoft, and the Open Device Interface (**ODI**) developed by Novell.

ODI and NDIS data link layers can be made compatible with each other using a **translation driver**.

For more information, see "Working with multiple protocol stacks" on page 692.

The network layer

The network layer takes a **packet** of information from the transport layer above it and gets it to the corresponding network layer on the receiving protocol stack. Issues of routing are handled by the network layer.

Information at a higher level is broken down into packets of a specified size (in numbers of bytes) for transmission across a network.

Examples of network layer protocols

Internet Protocol (IP) and Novell's IPX are widely-used network layer protocols. Adaptive Server Anywhere has interfaces to both network layer protocols and transport layer protocols. Adaptive Server Anywhere has an interface directly to the IPX network protocol.

The transport layer

The principal task of the transport layer is to guarantee the transmission of information between applications. It accepts information directly from a network application, such as a database server, splits it up if necessary, passes the packets to the network layer and ensures that the pieces all arrive correctly at the other end, where it assembles the packets before passing them up the stack to the application layer.

Examples of transport protocols

Novell's SPX, Microsoft and IBM's NetBEUI, and Named Pipes are widely-used transport protocols. The TCP/IP suite of protocols includes more than one transport layer. NetBIOS is an interface specification to the transport layer from IBM and Microsoft that is commonly (but not necessarily) paired with the NetBEUI protocol.

Adaptive Server Anywhere supports the NetBIOS interface to the transport layer. In addition, Adaptive Server Anywhere has an interface to Named Pipes for same-machine communications only.

Adaptive Server Anywhere applies its own checks to the data passed between client application and server, to further ensure the integrity of data transfer.

The application layer

Database servers and client applications are typical application layers in a protocol stack, from a networking point of view. They communicate using an application-defined protocol. This protocol is internal to Adaptive Server Anywhere programs.

Typical data for transmission includes a SQL query or other statement (from client application to database server) or the results of a SQL query (from database server to client application).

Passing information down the protocol stack

The client library and database server have an interface at the transport level (in the case of NetBIOS or Named Pipes), or at the network level (in the case of IPX and TCP/IP), passing the information to the network communications software. The lower levels of the protocol stack are then responsible, independent of the application, for transmitting the data to the equivalent layer on the other computer. The receiving layer hands the information to Adaptive Server Anywhere on the receiving machine.

The database server and the client library perform a set of checks and functions to ensure that data passed across the network arrives correctly, in a proper form.

Compatible protocol stacks

For two protocol stacks to be compatible, they must be operating the same transport layer (say NetBIOS) on each stack, and the same network protocol on each stack (say IP). If one stack employs a NetBIOS interface to the transport layer, so must the other. A client application running on a protocol stack employing NetBIOS cannot communicate with a database server using a TCP/IP protocol stack.

At the data link layer, ODI-based protocol stacks can be made compatible with NDIS-based protocol stacks using translation drivers, as discussed in "Working with multiple protocol stacks" on page 692.

Real world protocol stacks

The OSI model is close enough to current networking software and hardware to be a useful model for thinking about networks. There are inevitable complications because of the different ways in which software companies have designed their own systems. Also, there are complications when an individual computer is running more than one protocol stack, as is increasingly common.

Common protocol stacks

The TCP/IP protocol is available on all major operating systems. In addition, some widely used network software packages implement their own protocol stacks. Here are some networking software vendors together with their most common proprietary protocol stacks.

- **Novell** Novell NetWare typically operates using an SPX transport level atop an IPX network protocol (often written as SPX/IPX). Novell's IPX requires an ODI data link layer. The NetWare client installation installs this protocol stack on Novell NetWare client computers. Adaptive Server Anywhere has an interface directly to the IPX protocol, and does not rely on the services of the SPX layer.

  NetWare, IPX, and ODI are often used interchangeably when discussing network protocols.
- **Microsoft** Microsoft Windows NT 3.5 and later comes with networking software for NetBIOS, IPX, and TCP/IP. Windows 95 installs IPX as default networking software, and also comes with NetBEUI software. Windows for Workgroups typically uses NetBIOS on top of NetBEUI as the transport level protocol. The NDIS data link layer protocol was jointly developed by Microsoft and IBM, and is employed by all Microsoft's higher-level protocols.
- **IBM** IBM LAN Server and other IBM networking software use a NetBIOS interface to a NetBEUI transport layer on top of an NDIS data link layer. The NDIS data link layer protocol was jointly developed by Microsoft and IBM, and is employed by all IBM's higher-level protocols.

In each case, the installed data link layer may be changed from ODI to NDIS (or vice versa) by other networking software if more than one network protocol is installed. In these cases a translation driver ensures that both ODI and NDIS data link layer interfaces are available to the higher levels of the protocol stack. Working with more than one protocol stack is discussed in more detail in the next section.

Working with multiple protocol stacks

For two network applications (such as a client application and a database server) to communicate, the client computer and the server computer must have compatible protocol stacks at each level. When each computer has a single network protocol installed, it is fairly straightforward to ensure that each is running the same stack. However, when each computer is running multiple protocols, the situation becomes more complex.

At the transport and network layers, there is no problem with more than one protocol running at once; as long as there is a compatible path through the protocol stacks, the two applications can communicate.

ODI and NDIS interfaces

There are two widely-used interfaces between the data link layer and the network layer above it: ODI and NDIS. The data link layer communicates directly with the network adapter, so only one data link driver can be installed at a time. However, because most data link layers do not modify the information placed on the network wire, an ODI data link driver on one computer is compatible with an NDIS driver on another computer.

Each network layer is written to work with one, and only one, of ODI or NDIS. However, translation drivers are available that enable network level protocols that require an NDIS (or ODI) data link interface to work with ODI (or NDIS) data link layers. Consequently, a protocol stack built on an ODI data link layer can be compatible with a protocol stack based on an NDIS data link layer as long as the upper layers are compatible.

ODI and NDIS translation drivers

For example, Novell provides a driver named *odinsup*, which enables support for NDIS network protocols on an ODI data link layer, as well as a driver named *odi2ndi*, which translates in the other direction. Microsoft provides an ODI to NDIS mapper called *odihlp.exe* with Windows for Workgroups and Windows 95.

The translation drivers enable NDIS-based protocol stacks to be compatible with ODI-based protocol stacks on other computers, and to coexist on the same computer. You can think of the network adapter driver, the NDIS or ODI interface, and (if present) the translation driver as together forming the data link layer. For instance, you can configure a computer with an NDIS driver to communicate (via *odi2ndi*) with a Novell NetWare server using IPX on ODI drivers and, at the same time, communicate (via the NDIS driver) with a Windows NT computer using TCP/IP on an NDIS driver.

Troubleshooting tip
Not all translation drivers achieve complete compatibility. Be sure to get the latest available version of the driver you need. Although we provide some tips concerning network troubleshooting, the primary source of assistance in troubleshooting a particular protocol stack is the documentation for the network communications software that you install.

Supported network protocols

Properly configured Adaptive Server Anywhere servers run under the following networks and protocols:

- ♦ **Windows NT and Windows 95** NetBIOS, TCP/IP, or IPX protocols.
- ♦ **NetWare** All Novell networks using the IPX or TCP/IP protocols.
- ♦ **UNIX** TCP/IP protocol.
- ♦ **QNX** QNX messages or the TCP/IP protocol.

In addition, two protocols are provided for communication from a client application running on the same machine but built for a different operating system.

- ♦ **Windows NT** Named Pipes is used to communicate with Windows 3.x client applications running on the same machine as the server. Named Pipes are not used for network communications.
- ♦ **Windows 95** DDE is used to communicate with a Windows 3.x client application running on the same computer. This protocol is not used for network communications.

The client library for each platform supports the same protocols as the corresponding server. In order for Adaptive Server Anywhere to run properly, the protocol stack on the client and server computers must be compatible at each layer.

Using the TCP/IP protocol

TCP/IP is a suite of protocols originally implemented by the University of California at Berkeley for BSD UNIX. TCP/IP is gaining widespread use with the expansion of the Internet and the World-Wide Web. Different TCP/IP implementations rely on particular data link drivers in order to work correctly. For example, TCP/IP for NetWare relies on ODI drivers.

Unlike NetBEUI and IPX, the TCP/IP protocol is not associated with any specific software vendor. There are many implementations of TCP/IP available from different vendors, and many different programming interfaces to TCP. Consequently, Adaptive Server Anywhere supports only certain TCP/IP implementations on each platform. For details, see the subsections below for each platform. Because all TCP/IP implementations do implement the same protocol suite, they are all compatible.

UDP is a transport layer protocol that sits on top of IP. Adaptive Server Anywhere uses UDP on top of IP to do initial server name resolution and TCP for connection and communication after that, while SA used UDP for everything.

Using TCP/IP with Windows 95 and Windows NT

The TCP/IP implementation for Windows 95 and Windows NT is written to the Winsock 1.1 standard. Your TCP/IP software must support this standard.

Windows 95, Windows NT 3.5 and later ship with TCP/IP software that uses NDIS network drivers. Install TCP/IP Protocol from the Windows NT Control Panel, Network Settings.

This software allows a database server for Windows NT or a client application to use Windows NT TCP/IP.

Using TCP/IP with UNIX

The UNIX database server supports TCP/IP. This enables non-UNIX clients to communicate with a UNIX database server.

For performance reasons, you should consider running the Socket process on each QNX node using the TCP/IP link.

Using TCP/IP with Windows 3.x

The TCP/IP implementation for Windows 3.x is written to the Winsock 1.1 standard. Your TCP/IP software must support this standard.

In order to use TCP/IP under Windows 3.x, or under Windows for Workgroups, you need to obtain TCP/IP software that conforms to the Winsock 1.1 standard from a TCP/IP vendor, or obtain Microsoft TCP/IP from Microsoft.

Generally, TCP/IP software for Windows uses NDIS drivers If you are running Novell NetWare in addition to TCP/IP, you will need a translation driver to ensure that both NDIS-based and ODI-based network protocols can run, as discussed in "Working with multiple protocol stacks" on page 692. Windows for Workgroups comes with the *odihlp* translation driver.

Tuning TCP/IP performance

Although the default packet size for TCP/IP is 512 bytes, a larger packet size may improve query response time, especially for queries transferring a large amount of data between a client and a server process. You can set the packet size using the `-p` parameter on the database server command line, or by setting the CommBufferSize connection parameter in your connection profile.

☞ For information on the database server command line, see "The database server" on page 12 of the book *Adaptive Server Anywhere Reference Manual*. For information on CommBufferSize, see "CommBufferSize connection parameter" on page 42 of the book *Adaptive Server Anywhere Reference Manual*.

Using the IPX protocol

IPX is a network level protocol from Novell. Adaptive Server Anywhere for NetWare, Windows NT, Windows 95, and Windows 3.x can all employ the IPX protocol. This section provides some tips for using IPX under different operating systems.

Connecting via IPX

Some machines can use the NetWare bindery. These machines are NetWare servers or Windows NT or Windows 95 machine where the Client Service for NetWare is installed. A client application on one of these machines does not need to use broadcasts to connect to the server, if the server to which it is connecting is also using the bindery.

Applications and servers running on machines not using the bindery must connect using either of the following:

♦ **An explicit address** You can specify an explicit address using the HOST communication parameter.

  ☞ For more information, see "HOST parameter" on page 56 of the book *Adaptive Server Anywhere Reference Manual*.

♦ **Broadcast** You can broadcast over a network by specifying the DOBROADCAST communication parameter.

  ☞ For more information, see "DOBROADCAST parameter" on page 55 of the book *Adaptive Server Anywhere Reference Manual*.

Using IPX with Windows NT

Windows NT 3.5 ships with IPX network software that uses NDIS network drivers. This software can be installed from the Windows NT Control Panel, Network Settings. This software allows a Windows NT database server or a client application to use Windows NT IPX running on NDIS, while also allowing access to a NetWare file server, even though the NetWare file server will generally be using an ODI driver.

You must install both NWLink IPX/SPX Compatible Transport and Client Service for NetWare to use this feature. In some environments, problems have been found when the default setting (Auto-detect) has been chosen for the frame type. In this case, the frame type settings for the client and the server should be the same.

Using IPX with Windows 95

The IPX/SPX network protocol is the default network installed by Windows 95. If you installed Windows 95 without network support, you can install it later from the Control Panel, Network Settings.

Using IPX with Windows 3.x

IPX support for Windows 3.1 is generally associated with Novell NetWare. You must have the Novell support for Windows 3.1 installed and have Windows 3.x set up for Novell NetWare. This setup is done by the Novell NetWare requestor installation, and places three drivers into your *system.ini*: *vipx.386*, *vnetware.386*, and *netware.drv*. Many network adapters require an

```
EMMExclude=XXXX-YYYY
```

line in the *system.ini* file that excludes memory used by the network adapter.

The Adaptive Server Anywhere installation installs two Novell dynamic link libraries (DLLs): *nwcalls.dll* and *nwipxspx.dll*. You should check to see that there is not an older version of any DLLs in the Windows system directory or in the PATH before the Adaptive Server Anywhere directory.

Using IPX with Windows for Workgroups

Windows for Workgroups comes with NetBEUI as its default transport protocol. Microsoft NetBEUI employs a NetBIOS interface.

You can use both NetBEUI-based Windows for Workgroups network software and IPX-based Novell network software on a single network adapter board by installing NetWare support as an additional network.

NetWare support

Installing NetWare support allows a client application to use an NDIS-based protocol stack while also allowing access to a NetWare file server, even though the NetWare file server will generally be using an ODI driver. Installation is carried out using the Network Setup utility in the Network Program Manager group. See *networks.wri* in your Windows for Workgroups directory for details on installing NetWare support. You can view or print the file with the Write accessory in the Accessories Program Manager group.

IPX/SPX Compatible Transport

Windows for Workgroups can also support IPX communication even when the machine is not configured as a NetWare client, through the IPX/SPX Compatible Transport that comes with Windows for Workgroups. IPX/SPX Compatible Transport is installed by default on machines with enough memory. A Windows for Workgroups client can then communicate with Adaptive Server Anywhere using the NetBIOS protocol or IPX protocol.

IPX/SPX Compatible Transport is only necessary when Windows for Workgroups is not configured for NetWare and you would like a client application to communicate with a database server using IPX.

Tuning IPX performance

IPX parameters are found in the **Protocol IPXODI** section of your *net.cfg* file. The Novell IPX driver has only one parameter that affects performance:

♦ **IPX PACKET SIZE LIMIT** The IPX PACKET SIZE LIMIT is usually 576. The default maximum packet size is 512 bytes plus 30 bytes for IPX addresses.

If you are fetching long records in your multi-user system, you may want to increase the IPX packet size and specify the new packet size less 30 bytes as the packet size parameter (`-p`) on the server command lines.

Using the NetBIOS protocol

NetBIOS is an interface for interprocess communications, not a protocol specification. Programs written to the NetBIOS interface should operate over a variety of protocol stacks.

Protocol stacks that implement the NetBIOS interface must be compatible between client and server machines. For instance, a NetBIOS interface using SPX/IPX as the communication mechanism is incompatible with another NetBIOS implementation using NetBEUI.

As Windows 95, Windows for Workgroups and IBM networking software all use NetBIOS with NetBEUI as a standard protocol, configuration of the NetBIOS protocol stack for these environments is carried out by the network software installation.

Using NetBIOS with Windows NT

Windows NT 3.5 and later ships with NetBIOS interfaces to a number of different protocol stacks (NetBEUI, NetWare IPX/SPX transport, TCP/IP) that use NDIS network drivers. To use NetBIOS, install **NetBIOS Interface** from the Windows NT Control Panel, Network Settings.

Adaptive Server Anywhere works with any of these protocol stacks, but you must be sure to use compatible protocol stacks on the client and server sides of the communication.

You can have more than one NetBIOS interface active at one time. Each interface appears to as a different LAN adapter number. Adaptive Server Anywhere can simultaneously use different LAN adapter numbers, and so can simultaneously communicate on multiple NetBIOS interface protocol stacks.

Using Named Pipes

Named pipes are a facility for interprocess communication. Named pipes can be used for communication between processes on the same computer or on different computers.

Adaptive Server Anywhere for Windows NT uses local named pipes. This allows Windows 3.x applications running on Windows NT to communicate with a database server on the same machine.

Adaptive Server Anywhere does not use named pipes for client/server communications between different machines. You do not need to install remote named pipe support for local named pipes to work.

Using QNX messages

QNX messages is the native QNX network protocol. You can use QNX messages to communicate between a client application and a QNX database server.

By default, both server and client start all available protocols when they start up. If you are using QNX messages only, you may wish to disable the TCP/IP protocol on your client and server command lines.

Preventing the client from starting TCP/IP

If you experience lengthy delays when an application is first connecting to the database, it may be due to the client trying to start the TCP/IP link to the database. You can prevent the client from starting the TCP/IP link in the following ways:

- Set the SQLCLIENT environment variable. For example:

  ```
  export SQLCLIENT="dbclient -x qnx"
  ```

- If you have the source code for the application, you can achieve the same effect by calling the **db_client_start_line** function in the database library. For example:

  ```
  db_client_start_line( "dbclient -x QNX" );
  ```

Preventing the server from starting TCP/IP

You can prevent the server from starting the TCP/IP link by using the `-x` switch on the server command line. For example:

```
dbsrv6 -x qnx -n dbname asademo.db
```

Troubleshooting network communications

Network software involves several different components, increasing the likelihood of problems. Although we provide some tips concerning network troubleshooting here, the primary source of assistance in network troubleshooting should be the documentation and technical support for your network communications software, as provided by your network communications software vendor.

Ensure that you are using compatible protocols

If you have more than one protocol stack installed on the client or server computer, you should ensure that the client and the database server are using the same protocol. The `-x` command line switch for the server selects a list of protocols for the server to use, and the CommLinks connection parameter does the same for the client application.

You can use these options to ensure that each application is using the same protocol.

By default, both the database server and client library use all available protocol stacks. The server supports client requests on any active protocol, and the client searches for a server on all active protocols.

🖎 For more information about the `-x` switch, see "The database server" on page 12 of the book *Adaptive Server Anywhere Reference Manual*. For information about the CommLinks connection parameter, see "CommLinks connection parameter" on page 43 of the book *Adaptive Server Anywhere Reference Manual*.

Ensure that you have current drivers

Old network adapter drivers are a common source of communication problems. You should ensure that you have the latest version of the NDIS or ODI driver for your network adapter, as appropriate. You should be able to obtain current network adapter drivers from the manufacturer or supplier of the card.

Network adapter manufacturers and suppliers make the latest versions of drivers for their cards available. Most card manufacturers have a Web site from which you can download the latest versions of NDIS and ODI drivers.

You may also be able to obtain a current network adapter driver from the provider of your networking software.

When you download Novell client software, it has ODI drivers for some network adapters in addition to the Novell software that is used for all network adapters.

Switch off your computer between attempts

Some network adapter boards do not reset cleanly when you reboot the computer. When you are troubleshooting, it is better to turn the computer off, wait a few seconds, and then turn it back on between attempts.

Diagnose your protocol stack layer by layer

If you are having problems getting your client application to communicate with a database server, you need to ensure that the client and the database server are using compatible protocol stacks.

A helpful method of isolating network communication problems is to work up the protocol stack, testing whether each level of communication is working properly.

Verify that the data link layer is working

If you can connect to the server computer in any way, then the data link layer is working, regardless of whether the connection is made using the same higher-layer protocols you will be using for Adaptive Server Anywhere.

For example, you may want to try to connect to a disk drive on the computer running the database server from the computer running the client application.

Having verified that the data link layer is working, the next step is to verify that other applications using the same network and transport layers as Adaptive Server Anywhere are working properly.

Testing a NetBIOS protocol stack

If you are using Windows for Workgroups, Windows 95, or Windows NT, and you are using the native protocol, try using the **chat** or **WinPopup** application. This tests whether applications on the client and server computers can communicate with each other.

You should ensure that the applications that come with your networking software are running properly before testing Adaptive Server Anywhere.

Testing a TCP/IP protocol stack

If you are running under TCP/IP, there are several applications that you can use to test the compatibility of the client computer and server computer TCP/IP protocol stack. The **ping** utility provided with many TCP/IP packages is useful for testing the IP network layer.

Using ping to test the IP layer

Each IP layer has an associated address—a four-integer period-separated number (such as 191.72.109.12). Ping takes as an argument an IP-address and attempts to send a single packet to the named IP-protocol stack.

First, determine if your own protocol stack is operating correctly by "pinging" yourself. If your IP-address is 191.72.109.12, you would enter:

```
ping 191.72.109.12
```

at the command line prompt and wait to see if the packets are routed at all. If they are, the output will appear similar to the following:

```
c:> ping 191.72.109.12

Pinging 191.72.109.12 with 32 bytes of data:
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
...
```

If this works, it means that the computer is able to route packets to itself. This is reasonable assurance that the IP layer is set up correctly. You could also ask someone else running TCP/IP for their IP address and try pinging them.

You should ensure that you can ping the computer running the database server from the client computer before proceeding.

Using telnet to test the TCP/IP stack

To further test the TCP/IP stack you can start a server application on one computer, and a client program on the other computer, and test whether they can communicate properly.

There are several applications commonly provided with TCP/IP implementations that can be used for this purpose: here we show how to use **telnet** to test the TCP/IP stack.

1 Start a telnet server process (or **daemon**) on one machine. Check with your TCP/IP software for how to do this. For a typical command line telnet program you would type the following instruction at the command prompt:

    ```
    telnetd
    ```

2 Start the telnet client process on the other machine, and see if you get a connection. Again, check with your TCP/IP software for how to do this. For command line programs, you would typically type the following instruction:

```
telnet server_name
```

where *server_name* is the name or IP address of the computer running the telnet server process.

If a telnet connection is established between these two machines, the protocol stack is stable and the client and server should be able to communicate using the TCP/IP link between the two computers. If a telnet connection cannot be established, there is a problem. You should ensure that your TCP/IP protocol stack is working correctly before proceeding.

Diagnosing wiring problems

Faulty network wiring or connectors can cause problems that are difficult to track down. Try recreating problems on a similar machine with the same configuration. If a problem occurs on only one machine, it may be a wiring problem or a hardware problem.

For information on detecting wiring problems under NetWare, see your Novell NetWare manuals. The Novell LANalyzer program is useful for tracking down wiring problems with Ethernet or TokenRing networks. Your NetWare authorized reseller can also supply you with the name of a Certified NetWare Engineer who can help diagnose and solve wiring problems.

A checklist of common problems

For a description of network communication parameters, see "Network communications parameters" on page 54 of the book *Adaptive Server Anywhere Reference Manual*.

The following list presents some common problems and their solutions.

If you receive the message Unable to start — server not found when trying to start the client, the client cannot find the database server on the network. Check for the following problems:

- The network configuration parameters of your network driver on the client machine are different from those on the server machine. For example, two Ethernet adapter cards should be using a common frame type. For Novell NetWare, the frame type is set in the *net.cfg* file. Under Windows 95 and Windows NT the settings can be accessed through the Control Panel Network Settings. In Windows for Workgroups, it can be accessed through Network Settings in the Windows Setup.
- Under the TCP/IP protocol, clients search for database servers by broadcasting a request. Such broadcasts will typically not pass through gateways, so any database server on a machine in another (sub)network, will not be found. If this is the case, supply the host name of the machine on which the server is running using the `-x` command-line option.
- Your network drivers are not installed properly or the network wiring is not installed properly.
- The network configuration parameters of your network driver are not compatible with Adaptive Server Anywhere multi-user support. See "Configuring your network adapter board" on page 706 for a description of configuration parameters that affect performance and may affect operation of the client and server.
- If your network communications are being carried out using TCP/IP, and you are operating under Windows for Workgroups or Windows NT, check that your TCP/IP software conforms to the Winsock 1.1 standard.
- If you receive the message Unable to initialize any communication links, no link can be established. The probable cause is that your network drivers have not been installed. The server and the client try to start communication links using all available protocols unless you have specified otherwise using the `-x` option. Check your network documentation to find out how to install the driver you wish to use.

Configuring your network adapter board

Network adapter boards have configuration settings that allow them to use different interrupt request (IRQ) levels. The network adapter board may work only when it is using the same IRQ level as another adapter board in your computer (for example, your parallel card). Performance may suffer when the network adapter board shares an IRQ level.

The drivers for some adapter boards have parameters. The most important type of parameter to look out for is one that controls buffer size or number of buffers. For example, some versions of the NDIS driver for an SMC PLUS Ethernet adapter have these configuration parameters:

```
ReceiveBuffers = 12
ReceiveBufSize = 768
```

The default packet size for Adaptive Server Anywhere is 512 bytes. The maximum buffer size used by the adapter board should be at least 600 to allow for protocol information in the packet. The computer running the database server might need more than the default number of buffers used by a driver.

Evaluating network performance

The *dbconsol* utility provides some statistics for evaluating network performance. These are gathered at both the server and the client ends of the communication.

Server statistics can be accessed from Sybase Central and the Windows NT Performance Monitor. Client statistics can be displayed by running the console utility (**dbconsol**) and pressing F3.

The console has a **test mode**. In test mode, it sends packets to the server as rapidly as it can receive response packets from the server. You can activate the test mode by choosing **Test Mode** from the **Command** menu.