CLIENT/SERVER SYSTEMS

APPENDIX F

Preview

Today client/server computing is a fact of life. The Internet—along with its intranet and extranet derivatives—is perhaps the most pervasive example of client/server computing, and it has taken center stage with regard to application development. Because of the Internet's wide reach and acceptance, you should know what client/server computing is, what its components are, how the components interact, and what effects client/server computing has on database design, implementation, and management.

Data Files Available on www.cengagebrain.com:

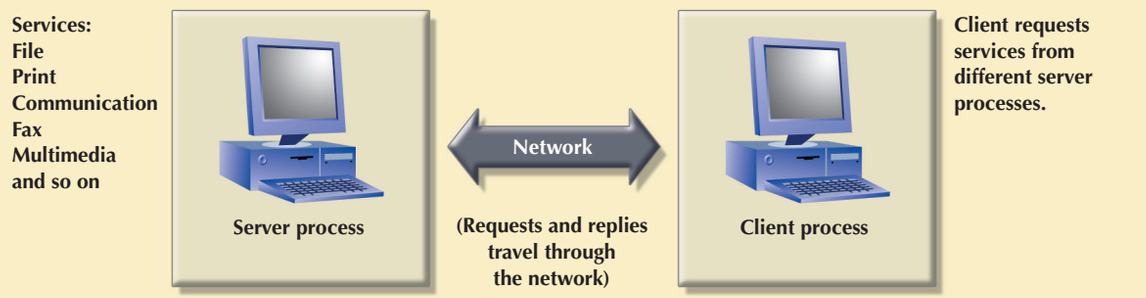
Appendix	Available formats	Questions and Problems	Available formats
There are no data files for this appendix.			

F.1 What Is Client/Server Computing?

Client/server is a term used to describe a computing model for the development of computerized systems. The model is based on the distribution of functions between two types of independent and autonomous processes: servers and clients. A **client** is any process that requests specific services from server processes. A **server** is a process that provides requested services for clients. Client and server processes can reside in the same computer or in different computers connected by a network.

When client and server processes reside on two or more independent computers on a network, the server can provide services for more than one client. In addition, a client can request services from several servers on the network without regard to the location or the physical characteristics of the computer in which the server process resides. The network ties the servers and clients together, providing the medium through which clients and servers communicate. (See Figure F.1.) As you examine Figure F.1, note that the services can be provided by a variety of computers on the network. For example, one computer may be dedicated to providing file and print services, another may provide communication and fax services, some may be used as web servers, and others may provide database services.

FIGURE F.1 Basic client/server computing model



The key to client/server power is where the request processing takes place. For example, in a client/server database, the client requests data from the database server. The actual processing of the request (selection of the records) takes place in the database server computer. In other words, the server selects the records that match the selection criteria and sends them over the network to the client. Information processing can be divided among different types of (server) computers ranging from workstations to mainframes.

The extent of the separation of data-processing tasks is the key difference between client/server systems and mainframe systems. In mainframe systems, all processing takes place on the mainframe, and the (usually dumb) terminal is used only to display the data screens. In that environment, there is no autonomy—the terminal is simply an appendage to the mainframe. In contrast, the client/server environment provides a clear separation of server and client processes, and both processes are autonomous. The relationship between servers and clients is many-to-many; one server can provide services to many clients, and one client can request services from many servers.

Depending on the extent to which the processing is shared between the client and the server, a server or a client may be described as fat or thin. A **thin client** conducts a minimum of processing on the client side, while a **fat client** carries a relatively larger proportion of the processing load. A **fat server** carries the bulk of processing burdens, while a **thin server** carries a lesser processing load. Thus, thin clients are associated with fat servers; conversely, fat clients are associated with thin servers.

Finally, client/server systems may also be classified as two-tier or three-tier. In a **two-tier client/server system**, a client requests services directly from the server. In a **three-tier client/server system**, the client's requests are handled by intermediate servers that coordinate the execution of the client requests with subordinate servers.

To understand why client/server computing is such a powerful player in the modern computing arena, you must examine its evolution, its architecture, and its functions.

F.2 The Evolution of Client/Server Information Systems

In the mid-1970s, corporate data resided safely within big, expensive mainframes that were driven by complex, proprietary operating systems. Dumb terminals, connected to front-end processors, communicated with the mainframe to produce the required information. The mainframe and its accompanying devices were jealously guarded, and access was rigorously restricted to authorized personnel. That computing style, partly dictated by available hardware and software and partly made possible by a relatively static data environment, suited the usually large companies that could afford the high cost of such computing. The centralized computing style imposed rigid control on the applications, strict limits to end-user data access, and a complex MIS department bureaucracy.

With the introduction of microcomputers in the 1980s, users were able to manipulate data locally with the help of relatively easy-to-use software such as spreadsheets and microcomputer-based database systems. However, the data on which the software operated still resided in the mainframe. Users often manually reentered the necessary data to make them accessible to the local application. This "manual download" of information was not very productive and was subject to the data anomalies discussed in Chapter 1, Database Systems. In the early 1980s, many managers' desks were home to a dumb terminal and a PC. A substantial portion of the information game required the concurrent use of both devices. Few MIS department managers viewed the PC as a first-class citizen in their information delivery infrastructure.

The use of the PC grew steadily over the years and eventually replaced the dumb terminals on end users' desks. Communications and terminal emulation programs allowed the PC to connect to and integrate with the MIS data center. The PCs connected to the mainframe were usually referred to as **intelligent terminals**. By this time, the electronic download of data from the mainframe to the PC was the standard way to extract required data from the mainframe to be manipulated by the local PC. Given that data access environment, the end users' data were only a snapshot of the company's changing mainframe data. Therefore, current mainframe data had to be downloaded frequently to avoid outdated reports or inaccurate query results.

Using their PCs, end users could create their own databases and reports, thus relying less on the MIS department's centralized control and services. Unfortunately, that new end-user freedom caused the proliferation of snapshot versions of the corporate database. This scenario created so-called islands of information that were independent of the MIS department. Data sharing between the islands was unsophisticated. When users needed to share data, they would simply copy the data to a disk and walk to the coworker's office, disk in hand. That data-sharing approach was later labeled the **sneakernet**.

It was no surprise that the PC's introduction caused data security, data replication, and data integrity problems for corporate MIS departments. However, because PCs yielded many end-user information benefits, their growth could not be controlled easily. Consequently, to retain some semblance of control, MIS departments encouraged the development of departmental PC users' groups to share information electronically.

The new willingness to share information electronically was made possible, in large part, by a company known as Novell Data Systems, which introduced Netware/86 (originally called ShareNet) in 1983. The Novell software and hardware allowed MIS managers to connect PCs through a local area network (LAN). The LAN made it possible for end-user PCs to share files via a central PC that acted as a network file server. Netware/86

became the first widely accepted **network operating system** (**NOS**) for IBM personal computers and compatibles.

As PC microprocessor and data storage technology advanced rapidly in the 1990s, new PCs began to rival mainframe processing power. That trend accelerated in the late 1990s and into the first few years of the 21st century. The new PC power made it possible and, based on relative computing cost, even desirable for MIS development teams to shift much of their work to PC-based application development tools. A welcome result of that operational shift was that the end user and the MIS specialist were now working on a common PC platform. As more PCs were integrated into the corporate data centers, MIS department needs grew closer to those of the end user. As operating systems and network technology matured, even some mission-critical applications and business functions were moved from the mainframe to the PC platform.

The evolution from mainframe computing to PC-based client/server information systems generated many changes in key aspects of information management. Some of those differences are highlighted in Table F.1.

TABLE F.1 Contrasting Mainframe and Client/Server Information Systems

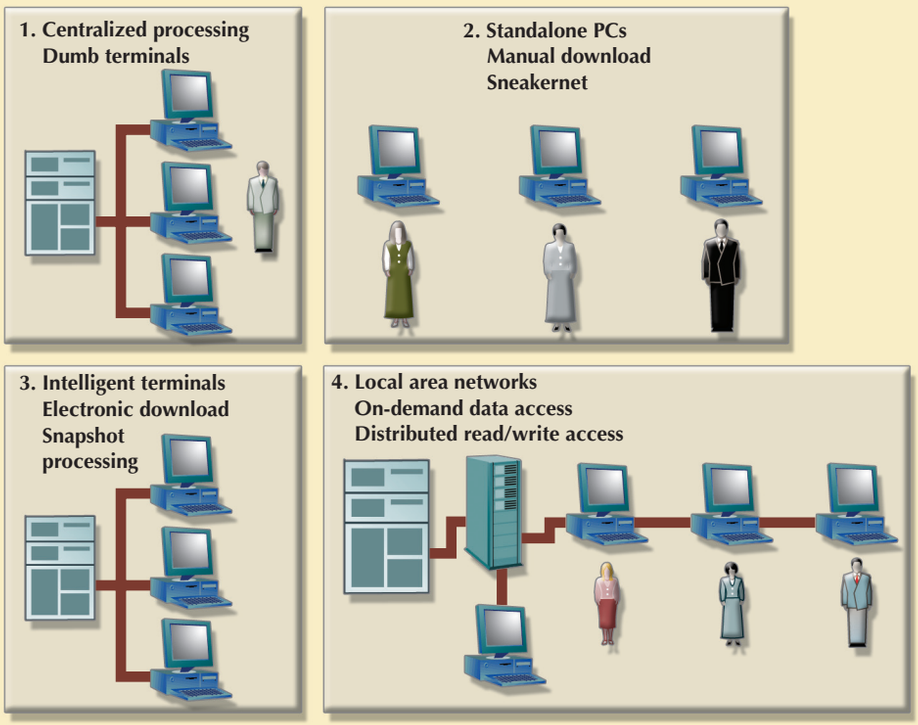
ASPECT	MAINFRAME-BASED INFORMATION SYSTEM	PC-BASED CLIENT/SERVER INFORMATION SYSTEM
Management	Centralized	Distributed/decentralized
Vendor	Single-vendor solution	Multiple-vendor solution
Hardware	Proprietary	Multiple vendors
Software	Proprietary	Multiple vendors
Security	Highly centralized	Decentralized
Data manipulation	Very limited	Extensive and very flexible
System management	Integrated	Few tools available
Application development	Overstructured Time-consuming Creation of application backlogs	Flexible Rapid application development Better productivity tools
End-user platform	Dumb terminal Character-based Single task Limited productivity	Intelligent PC Graphical user interface (GUI) Multitasking OS Better productivity tools

Figure F.2 summarizes the preceding discussion by depicting the four stages of the information systems' evolution from the mainframe to the PC-based infrastructure required for client/server computing.

The general forces behind the move to PC-based client/server computing are:

- *The changing business environment*. Businesses must meet global competitive pressures by streamlining their operations and by providing an ever-expanding array of customer services. Information management has become critical in this competitive environment, making fast, efficient, and widespread data access key to survival. The corporate database has become a far more dynamic asset than it used to be, and it must be available at a relatively low cost.
- *The growing need for enterprise data access*. When corporations grow, especially when they grow by merging with other corporations, it is common to find a mixture of disparate data sources in their systems. In such a multiple-source data environment, managers and decision makers need fast, on-demand data access and easy-to-use tools to integrate and aggregate data. Client/server computing makes it possible to mix and match data as well as hardware. In addition, the Internet's inherent client/server structure makes it relatively easy to access both external and internal data sources.

FIGURE F.2 Evolution of the computing environment



- *The demand for end-user productivity gains, based on the efficient use of data resources*. Client/server computing supports end users' demands for better ad hoc data access and data manipulation, better user interfaces, and better computer integration.
- *Technological advances that have made client/server computing practical*. The change to PC-based information systems was also driven by advances in microprocessor technology and storage capacity, data communications and the Internet, database systems, operating systems and GUI interfaces, and sophisticated application software.
- *Growing cost/performance advantages of PC-based platforms*. PC platforms often offer unbeatable price/performance ratios compared to mainframe and minicomputer platforms. PC application costs, including acquisition, installation, and use, are usually lower than those of similar minicomputer and mainframe applications. (In complex client/server system implementations, PC-based training and support costs might be higher than those in a mainframe environment. Purchasing hardware and software from multiple sources can also become a major management headache, especially when system problems occur. Yet for many organizations, the dollar cost comparison between PC-based client/server systems and mainframe systems favors PC-based systems.)

F.3 Client/Server Architecture

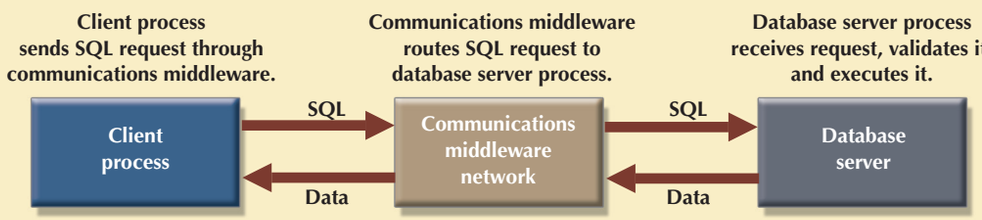
The client/server infrastructure, known as the client/server architecture, is a prerequisite to the proper deployment of client/server systems. The **client/server architecture** is based on hardware and software components that interact to form a system. That system includes three main components: clients, servers, and communications middleware.

- The client is any computer process that requests services from the server. The client is also known as the **front-end application**, reflecting that the end user usually interacts with the client process.
- The server is any computer process providing services to the clients. The server is also known as the **back-end application**, reflecting that the server process provides the background services for the client process.
- The **middleware** is any computer process through which clients and servers communicate. The middleware, also known as communications middleware or the communications layer, is made up of several layers of software that aid the transmission of data and control information between clients and servers. The communications middleware is usually associated with a network. All client requests and server replies travel through the network in the form of messages that contain control information and data.

F.3.1 How Client/Server Components Interact

To illustrate how the components interact, let's examine how a client requests services from a database server. Examine Figure F.3, noting that the application processing has been split into two main, independent processes: a client and a server. The communications middleware makes it possible for the client and server processes to work together. As you examine Figure F.3, also note that the communications middleware becomes the supporting platform on which clients and servers rest. Although the communications middleware is a key component in the system, its presence exacts a price by creating substantial additional overhead; adding system failure points; and, in general, adding complexity to the system's implementation.

FIGURE F.3 Interaction between client/server components



In Figure F.3, for example, the client process is in charge of the end-user interface, some portion of the local data validation, some processing logic, and data presentation. The communications middleware ensures that the messages between clients and servers are properly routed and delivered. SQL requests are handled by the database server, which validates the requests, executes them, and sends the results to the clients.

The server and client do not need to be in different computers. They can reside in the same computer and share the same processor, assuming the operating system allows it, assuming the use of a multitasking operating system. However, most client/server implementations place the client and server processes in separate computers. Figure F.4 illustrates a client/server system with two servers and three clients.

Given the environment shown in Figure F.4, a database server process runs on an HP computer, while an **imaging server** process runs on an IBM computer. The three client processes run under three different operating systems: Windows, Linux, and Apple Mac OS. The client and server processes are connected through a network. The front-end applications in the client computers request data and images from the back-end processes (database and imaging servers). The network and supporting software form the communications middleware through which clients and servers communicate. Note that the communications can take place between clients and servers as well as between servers. Remember from Chapter 12, Distributed Database Management Systems, that that scenario is typical of distributed database environments, in which requested data can be stored in different locations.

FIGURE F.4 An example of client/server architecture

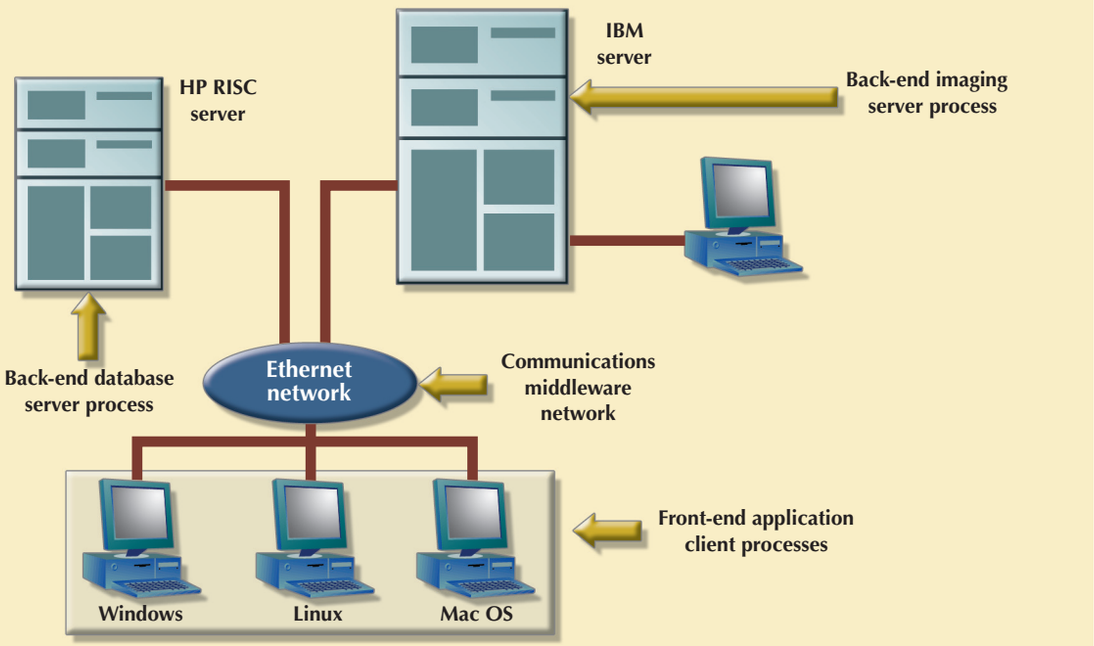


Figure F.4 illustrates a complex, yet common client/server environment in which the server processes are running under two different operating systems, the client processes are running under three different operating systems, and the system contains three different hardware platforms. In that scenario, the communications middleware (network and supporting software) becomes the integrating platform for all components. The following section examines the communications middleware components in greater detail.

F.3.2 Client Components

As mentioned earlier, the client is any process that requests services from a server process. The client is proactive and will, therefore, always initiate the conversation with the server. The client includes hardware and software components. Desirable client hardware and software features are:

- Powerful hardware
- An operating system capable of multitasking
- A graphical user interface (GUI)
- Communications capabilities

Because client processes typically require a lot of hardware resources, they should be stationed on a computer with sufficient processing power, such as a fast 64-bit processor workstation. Such processing power facilitates the creation of systems with multimedia capabilities. Multimedia systems handle multiple data types, such as voice, images, and video. Client processes also require large amounts of hard disk space and physical memory. You should have as much memory and hard disk space available as possible.

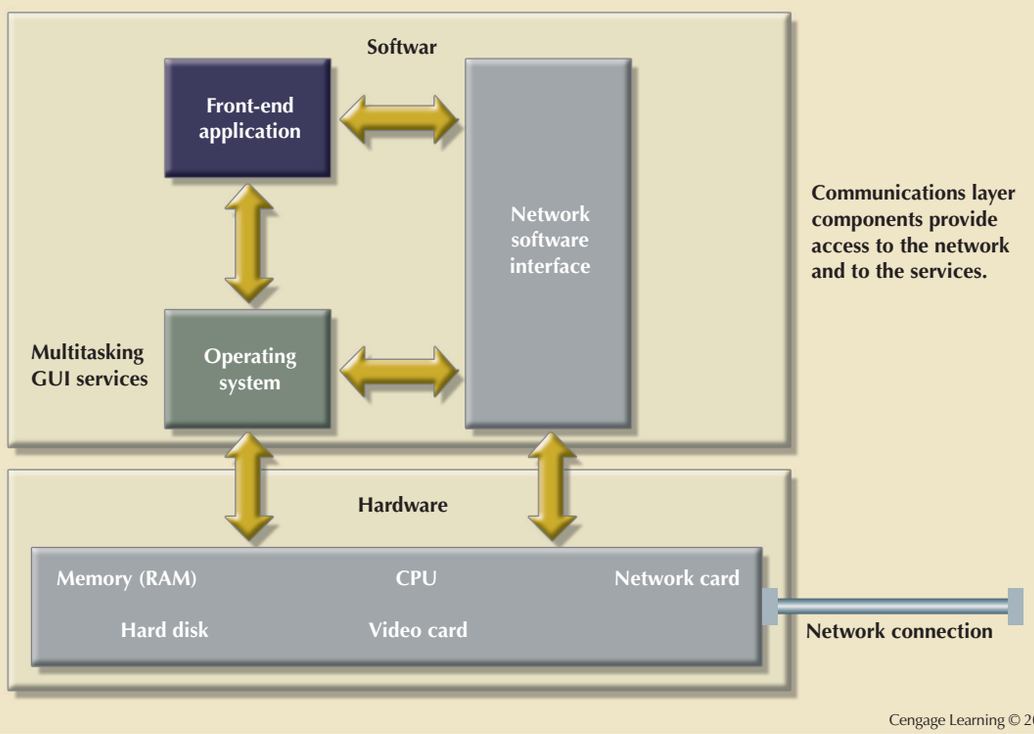
The client should have access to an operating system with at least some multitasking capabilities. Various versions of Microsoft Windows are the most common client platforms as of this writing. Windows provides access to memory, preemptive multitasking capabilities, and a graphical user interface. Those capabilities, in addition to the abundance of applications developed for the Windows interface, make Windows the platform of choice in the majority of

client/server implementations. However, although the Windows operating system is popular at the client side, other operating systems—such as Microsoft Windows Server and the many "flavors" of UNIX, including Linux—are better suited to handle the client/server processing that is largely done on the server side.

To interact efficiently in a client/server environment, the client computer must be able to connect and communicate with other computers in a network environment. Therefore, the combination of hardware and operating system must also provide adequate connectivity to multiple network operating systems (NOSs). The reason for requiring a client computer to be capable of connecting with and accessing multiple network operating systems is simple: services may be located in different networks.

The client application, or front end, runs on top of the operating system and connects with the communications middleware to access services available in the network. Several third-generation programming languages (3GLs) and fourth-generation languages (4GLs) can be used to create the front-end application. Most front-end applications are GUI-based to hide the complexity of the client/server components from the end user. Figure F.5 depicts the basic client components.

FIGURE F.5 **Client components**



As you examine Figure F.5, note that the front-end application interacts with the operating system to access the multitasking and graphical user interface capabilities provided by the operating system. The front-end application also interacts with the network software component of the communications middleware to access the services located in the network. The hardware components of the communications middleware (network cable and network board) physically transport the requests and replies between clients and servers. While the request is being processed by the server, the client is free to perform other tasks.

F.3.3 Server Components

As mentioned, the server is any process that provides services to client processes. The server is reactive because it waits for the client's requests. Servers typically provide:

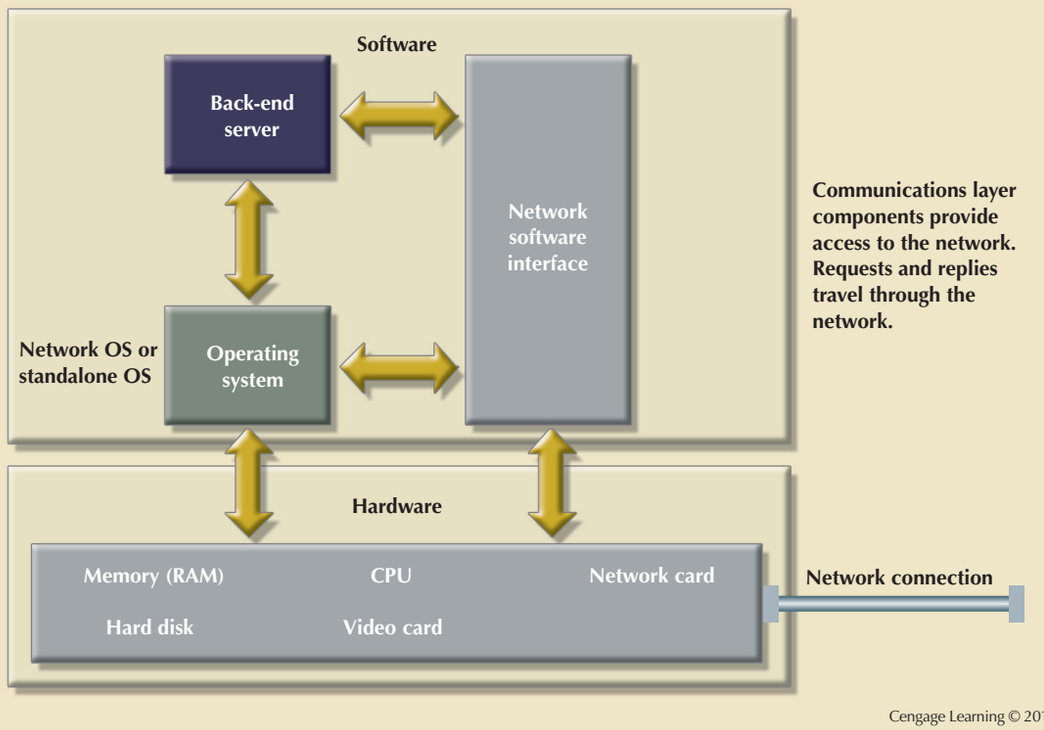
- *File services* for a LAN environment in which a computer with a big, fast hard disk or an array of disks is shared among different users. A client connected to the network can store files on the file server as if it were another local hard disk.
- *Print services* for a LAN environment in which a PC with one or more printers attached is shared among several clients. A client can access any one of the printers as if it were directly connected to its own computer. The data to be printed travel from the client's PC to the print server PC, where they are temporarily stored on the hard disk. When the client finishes sending the print job, the data are moved from the hard disk on the print server to the appropriate printer.
- *Fax services* that require at least one server equipped (internally or externally) with a fax device. The client PC need not have a fax machine or even a phone line connection. Instead, the client submits the data to be faxed to the fax server, with the required information, such as the fax number or name of the recipient. The fax server schedules the fax, dials the fax number, and transmits the fax. The fax server should also be able to handle any problems that occur in the process.
- *Communications services* that let client PCs connected to the communications server access other host computers or services to which the client is not directly connected. For example, a communications server allows a client PC to dial out to access a bulletin board or a remote LAN location.
- *Database services*, which constitute the most common and most successful client/server implementation. The client sends SQL requests to a database server. The server receives the SQL code, validates it, executes it, and sends only the results to the client. The data and the database engine are located on the database server computer. The client is required to have only the front-end application to access the database server.
- *Transaction services*, which are provided by transaction servers that are connected to the database server. A transaction server contains the database transaction code or procedures that manipulate the data in the database. A front-end application in a client computer sends a request to the transaction server to execute a specific procedure stored on the database server. No SQL code travels through the network. Transaction servers reduce network traffic and provide better performance than database servers.
- *Miscellaneous services* that include CD-ROM, DVD, video, and backup.

A common misconception is that the server process must run on the computer that contains the network operating system. This is not necessarily so. Unless circumstances dictate otherwise, separation of the server process and the NOS is highly recommended. That separation allows the server process to be located on any of the network's computers and still be available to all client computers. For example, suppose you have a CD-ROM server in a Novell NetWare network. If the server software requires the CD-ROM server process to be run on the same computer with the NetWare operating system, the host computer will be severely taxed. The host must double as a file server and a CD-ROM server. If the product does not require such "doubling," it can be installed on any PC in the network, thereby effectively distributing the workload. Both types of products use the network services (IPX or TCP/IP) provided by Novell NetWare to transport the messages between clients and the server. Each solution is subject to advantages and disadvantages. The best solution always depends on the specific circumstances.

Like the client, the server also has hardware and software components. The hardware components include the computer, CPU, memory, hard disk, video, and network card. The computer that houses the server process should be a more powerful computer than the "average" client computer because the server process must be able to handle concurrent requests from multiple clients. Server components are illustrated in Figure F.6.

The server application, or *back end*, runs on top of the operating system and interacts with the communications middleware components to "listen" for the client's requests for services. Unlike the front-end client process, the server process need not be GUI-based. Keep in mind that the back-end application interacts with the operating system (network or standalone) to access local resources (hard disk, memory, CPU cycles, and so on). The back-end server

FIGURE F.6 Server components



constantly "listens" for the client's requests. Once a request is received, the server processes it locally. The server knows how to process the request; the client tells the server only what it needs done, not how to do it. When the request is met, the answer is sent back to the client through the communications middleware.

Server hardware characteristics depend on the extent of the required services. For example, a database server for a network of 50 clients may require a computer with the following minimum characteristics:

- Fast CPU (Pentium Xeon, AMD Opteron 64-bit, or multiprocessor)
- Fault-tolerant capabilities:
  - Dual power supply to prevent power supply problems
  - Standby power supply to protect against power line failures
  - Error checking and correcting (ECC) memory to protect against memory module failures
  - Redundant array of independent disks (RAID) to protect against physical hard disk failures
- Expandability of CPU, memory, disk, and peripherals
- Bus support for multiple add-on boards
- Multiple communications options

In theory, any computer process that can be clearly divided into client and server components can be implemented through the client/server model. When properly implemented, the client/server architectural principles for process distribution are translated into the following server process benefits:

- *Location independence*. The server process can be located anywhere in the network.
- *Resource optimization*. The server process can be shared by several client processes.
- *Scalability*. The server process can be upgraded to run on more powerful platforms.

- *Interoperability and integration*. The server process should be able to work in a plug-and-play environment.

Those benefits, in addition to the hardware and software independence principles of the client/server computing model, facilitate the integration of PCs, midrange computers, and mainframes in a nearly seamless environment.

F.3.4 Communications Middleware Components

The communications middleware software provides the means through which clients and servers communicate to perform specific actions. In the client process, the communications middleware software also provides the specialized services that insulate the front-end applications programmer from the internal workings of the database server and network protocols. In the past, applications programmers had to write code that would directly interface with the specific database language (generally, a variation of SQL) and the specific network protocol used by the database server. If the same application were to be used with a different database and network, the application's routines had to be rewritten for the new database and network protocols. Clearly, that condition is undesirable, which is where middleware is valuable.

Although middleware can be used in different scenarios, such as e-mail, fax, or network protocol translation, most first-generation middleware used in client/server applications is oriented toward providing transparent data access to several database servers. The use of database middleware yields:

- *Network independence* by allowing the front-end application to access data without regard to the network protocols.
- *Database server independence* by allowing the front-end application to access data from multiple database servers without having to write code that is specific to each database server.

The use of database middleware makes it possible for the programmer to use generic SQL sentences to access different and multiple database servers. The middleware layer isolates the programmer from the differences among SQL dialects by transforming generic SQL sentences into the database server's expected syntax. For example, a problem in developing front-end systems for multiple database servers occurs because applications programmers must have in-depth knowledge of the network communications and the database access language characteristics of each database in order to access remote data. The problem is aggravated because each DBMS vendor implements its own version of SQL (with differences in syntax, additional functions, and enhancements with respect to the SQL standard). Furthermore, the data might reside in a nonrelational DBMS that doesn't support SQL, thus making it harder for the programmers to access the data. Given such cumbersome requirements, programming in client/server systems can be more difficult than programming in traditional mainframe systems. Database middleware eases the problem of accessing multiple sources of data in multiple networks and releases the programmer from the details of managing the network communications.

To accomplish its functions, the communications middleware software operates at two levels:

- The *physical* level deals with the communications between client and server computers (computer to computer). In other words, it addresses how the computers are physically linked. The physical links include the network hardware and software. The network software includes the network protocols. Recall that network protocols are the rules that govern how computers must interact with other computers in a network. They ensure that computers are able to send and receive signals to and from each other. Physically, the communications middleware is, in most cases, the network. Because the client/server model allows the client and the server to reside on the same computer, it may exist without the benefit of a computer network.
- The *logical* level deals with the communications between client and server processes (process to process), that is, with how the client and server processes communicate. The logical characteristics are governed by **interprocess** (or process-to-process) **communication** (**IPC**) protocols that give the signals meaning or purpose. It is at this level that most client/server conversation takes place.

To illustrate the two levels at which client/server communications take place, let's use an analogy. Suppose you order a pizza by phone. You start by picking up the phone, dialing the number, and waiting for someone to answer. When

the phone is answered, you identify yourself, tell the clerk what type of pizza you want, how many pizzas you want, and other details. In turn, the clerk asks for your address, provides price information, and gives you an estimated delivery time. That simple transaction required both physical- and logical-level actions:

- The physical-level actions included the telephone changing your voice to analog signals and the subsequent movement of those signals through phone lines to the phone company's central PBX and from there to the phone installed at the pizza place.
- The logical-level actions were handled by you and the clerk. Because you and the clerk spoke the same language, you requested the service in a format that the clerk understood and the two of you discussed the details of the transaction successfully.

Other than requiring that you know how to use the phone, the physical details of the phone connection are hidden. The phone company handled all of the physical details of your conversation, whereas you and the pizza clerk handled all of the logical details.

F.3.5 The OSI Model

Although the preceding analogy helps you understand the basic client/server interactions, you should know some details of computer communications to better understand the flow of data and control information in a client/server environment. Consider the **Open Systems Interconnection** (**OSI**) network reference model as an illustration of those details. That model, published in 1984, was developed by the International Organization for Standardization (ISO) in an effort to standardize the diverse network systems. The OSI model is based on seven layers, which are isolated from one another. No layer needs to know the details of another layer in order to operate. The OSI model, shown in Table F.2, was designed to let each layer provide specific services to the layer above it.

TABLE F.2 The OSI Network Reference Model

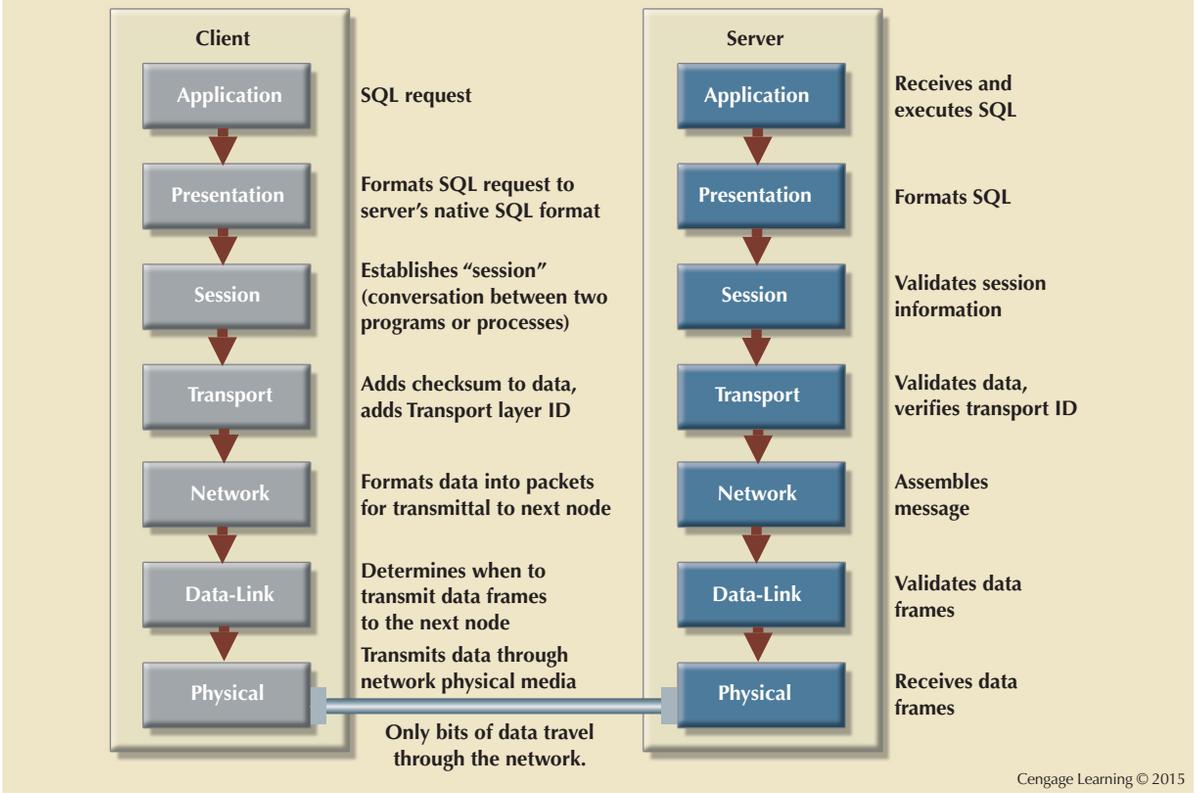
LAYER	DESCRIPTION
Application	End-user applications program. Client: front-end application such as e-mail or a spreadsheet. Server: back-end application such as a file server, a database server, or e-mail.
Presentation	Provides formatting functions for Application layer protocol conversion, compression, encoding, and so on.
Session	Establishes and controls communication between applications. Ensures security, delivery, and communications recovery.
Transport	Provides error recognition and recovery, ensures that all data are properly delivered, and adds Transport-layer-specific ID.
Network	Provides end-to-end routing of packets. Splits long messages into smaller units.
Data-Link	Creates frames for transmission and controls the shared access to the network physical medium (cable). Includes error checking, correction, and so on.
Physical	Provides standards dealing with the electrical details of the transmission (network cards, cable types, voltages, and so on). Physically transmits frames of data through the cable or other media.

As you examine the OSI network reference model, note how data flow in a network. The objective of the bottom layers is to hide the network complexity from all of the layers above. In short:

- The Application and Presentation layers provide end-user application-oriented functions.
- The Session layer ensures and controls program-to-program communications.
- The Transport, Network, Data-Link, and Physical layers provide network-oriented functions.

To better illustrate the functions contained within the OSI reference model, let's examine how a client requests services from a database server in a network. Figure F.7 depicts the flow of information through each layer of the OSI model.

FIGURE F.7 **Information flow through the OSI model**

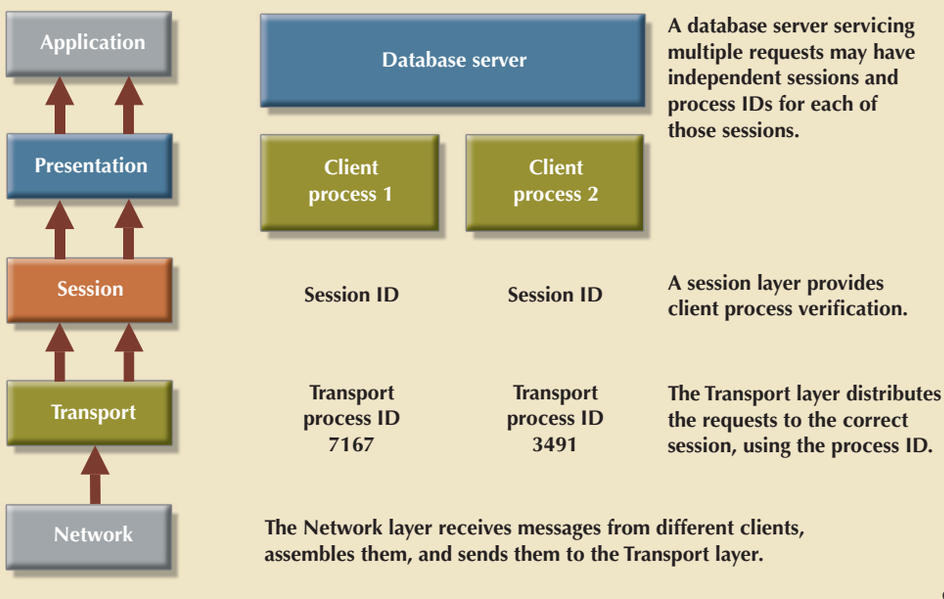


Using Figure F.7 as a guide, you can trace the data flow.

1. The client application (*Application layer*) generates a SQL request.
2. The SQL request is sent down to the *Presentation layer*, where it is changed to a format that the SQL server engine can understand. Actions include translating ASCII characters, indicating single- and double-precision numbers, and specifying date formats (for example, mm/dd/yyyy instead of dd/mm/yyyy).
3. The SQL request is handed down to the *Session layer*. The Session layer establishes the connection of the client process with the server process. If the database server process requires user verification, the Session layer generates the necessary messages to log on and verify the end user. At this point, usually at the beginning of the session, the end user may be required to enter a user ID and a password to access the database server, after which additional messages may be transmitted between the client and the server processes. The Session layer will identify which messages are control messages and which are data messages.
4. After the session is established and validated, the SQL request is sent to the *Transport layer*. The Transport layer generates some error validation checksums and adds some Transport-layer-specific ID information. For example, when several processes run on the client, each process may be executing a different SQL request or each process may access a different database server. The Transport layer ID helps the Transport layer identify which data correspond to which session.
5. Once the Transport layer has performed its functions, the SQL request is handed down to the *Network layer*. The Network layer takes the SQL request, identifies the address of the receiving node (where the server is located), adds the address of the next node in the path (if any), divides the SQL request into several smaller packets, and adds a sequence number to each packet to ensure that they are assembled in the correct order.

6. The packet is handed to the *Data-Link layer*. The Data-Link layer adds more control information. That control information depends on the network and on which physical media are used. This information is added at the beginning (header) and at the end (trailer) of the packet, which is why the output of this process is called a **frame**. Then the Data-Link layer sends the frame to the next node. The Data-Link layer is responsible for sharing the network medium (cable) and ensuring that no frames are lost.
7. When the Data-Link layer determines that it is safe to send a frame, it hands the frame down to the *Physical layer*, which transforms the frame into a collection of ones and zeros (bits) and then transmits the bits through the network cable. The Physical layer does not interpret the data; its only function is to transmit the signals.
8. The signals transmitted by the Physical layer are received at the server end by its *Physical layer*, which passes the data to the *Data-Link layer*. The Data-Link layer reconstructs the bits into frames and validates them. At this point, the Data-Link layers of the client and the server computer may exchange additional messages to verify that the data were received correctly and that no retransmission is necessary. Then the Data-Link layer strips the header and trailer information from the packet and sends the packet up to the *Network layer*.
9. The Network layer checks the packet's destination address. If the final destination is some other node in the network, the Network layer identifies it and sends the packet down to the Data-Link layer for transmission to that node. If the destination is the current node, the Network layer assembles the packets and assigns appropriate sequence numbers. Then the Network layer generates the SQL request and sends it to the *Transport layer*.
10. The Transport layer provides additional validation checks and then routes the message to the proper session by using the transport ID. Figure F.8 illustrates how the transport process ID correctly distributes network requests for a database server process that serves multiple clients.

FIGURE F.8 **Transport process ID**



11. Most of the client/server "conversation" takes place in the *Session layer*. If the communication between client and server processes is broken, the Session layer tries to reestablish the session. The Session layer identifies and validates the request and then sends it to the *Presentation layer*.

12. The Presentation layer provides additional validation and formatting.
13. The SQL request is sent to the database server or *Application layer*, where it is executed.

Keep in mind that although the OSI framework helps you understand network communications, it functions within a system that requires considerable infrastructure. The network protocols constitute the core of the network infrastructure because all data traveling through the network must adhere to some network protocol. In a client/server environment, it is not unusual to work with several different network protocols. Different server processes may support different network protocols to communicate over a network.

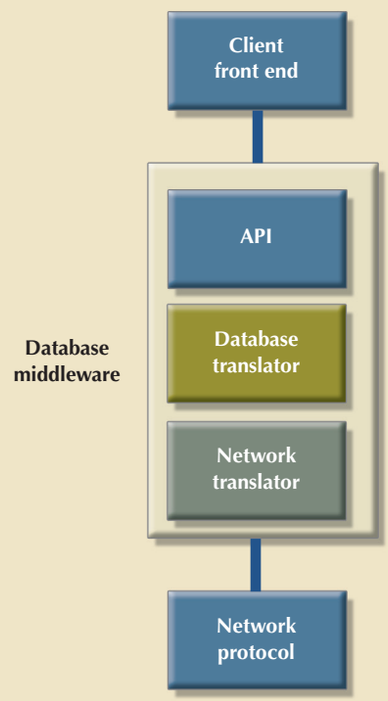
F.3.6 Database Middleware Components

As depicted in Figure F.9, database middleware is divided into the following three main components:

- Application programming interface (API)
- Database translator
- Network translator

Those components (or their functions) are generally distributed among several software layers that are interchangeable in a plug-and-play fashion.

FIGURE F.9 Database middleware components

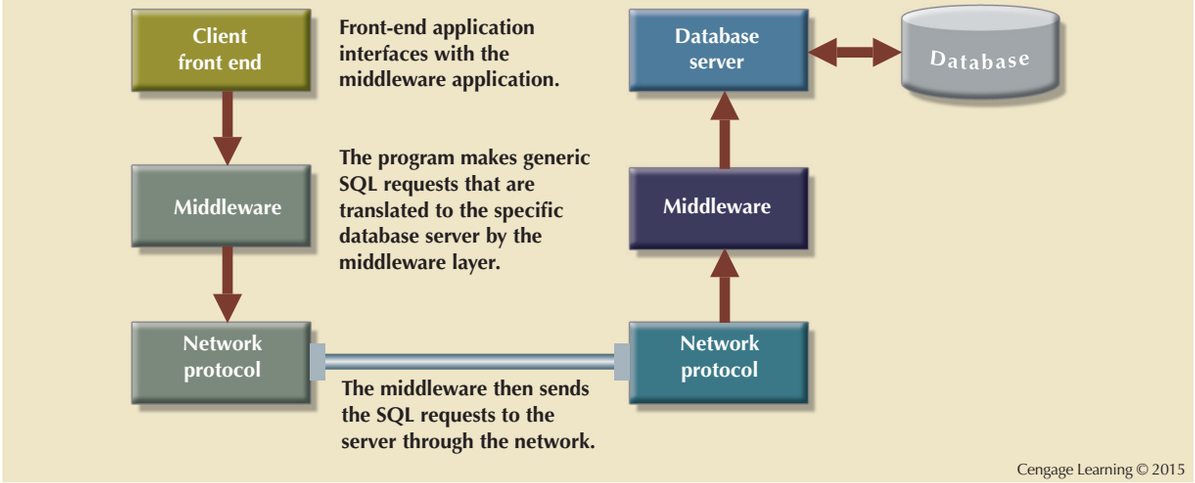


The **application programming interface** (**API**) is public to the client application. The programmer interacts with the middleware through the APIs provided by the middleware software. The middleware API allows the programmer to write generic SQL code instead of code specific to each database server. In other words, the middleware API allows the client process to be independent of the database server. That independence means that the server can be changed without requiring the client applications to be completely rewritten.

The **database translator** translates the SQL requests into the specific database server syntax. The database translator layer takes the generic SQL request and maps it to the database server's SQL protocol. If a database server has some nonstandard features, the database translator layer will opt to translate the generic SQL request into the specific format used by the database server. If the SQL request uses data from two different database servers, the database translator layer will take care of communicating with each server, retrieving the data using the common format expected by the client application.

The **network translator** manages the network communications protocols. Remember that database servers can use any of the network protocols discussed earlier. Therefore, if a client application taps into two databases, one that uses TCP/IP and another that uses IPX/SPX, the Network layer handles all the communications details of each database transparently to the client application. Figure F.10 illustrates the interaction between client and middleware database components.

FIGURE F.10 Interaction between client/server middleware components



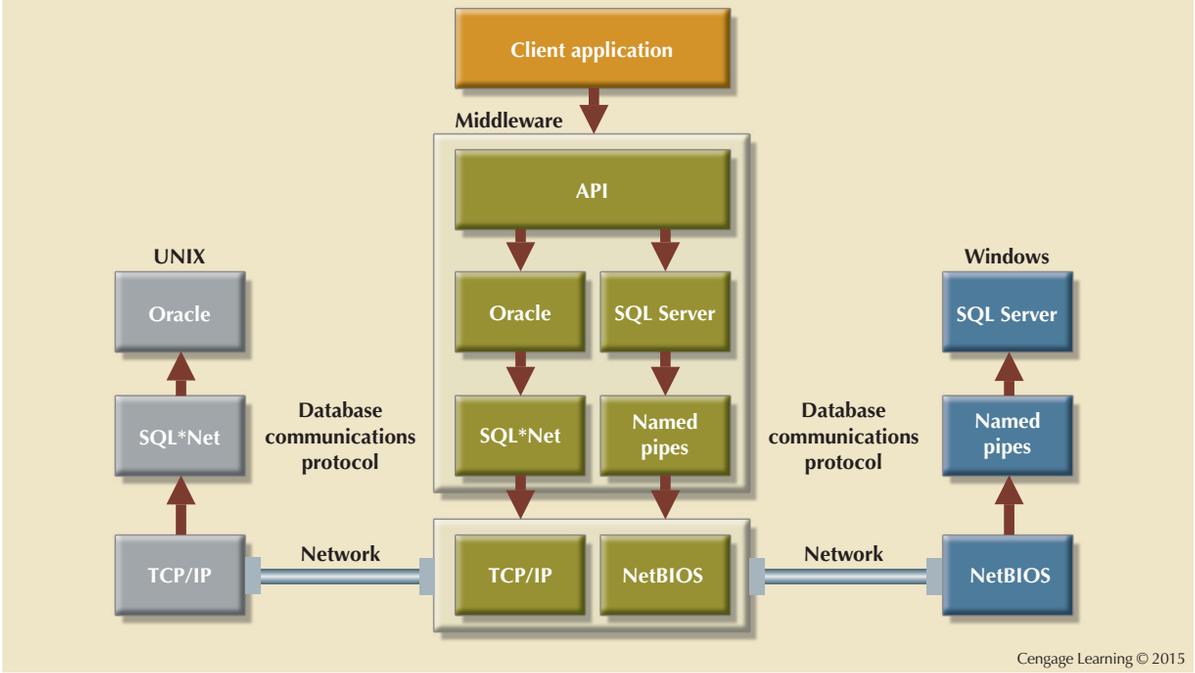
Given the existence of the three middleware components shown in Figure F.9, the three main benefits of using middleware software can be identified. Clients can:

- Access multiple (and quite different) databases
- Be database-server-independent
- Be network-protocol-independent

To illustrate how all of those pieces work together, let's see how a client accesses two different database servers. Figure F.11 shows how a client application requests data from an Oracle database server (Oracle Corporation) and from a SQL Server database server (Microsoft Corporation). The Oracle database server uses SQL*Net as its communications protocol with the client; the SQL Server database server uses Net-Library routines. SQL*Net, a proprietary solution limited to Oracle databases, is used by Oracle to send SQL requests over a network. Net-Library routines provide an interprocess communications (IPC) protocol used in SQL Server to manage client and server communications across the network.

As you examine Figure F.11, note that the Oracle server runs under the UNIX operating system and uses TCP/IP as its network protocol. The SQL server runs under the Windows NT operating system and uses NetBIOS as its network protocol. In this case, the client application uses a generic SQL query to access data in two tables: an Oracle table and a SQL Server table. The database translator layer of the middleware software contains two modules, one for each database server type to be accessed.

FIGURE F.11 Middleware accessing multiple database servers



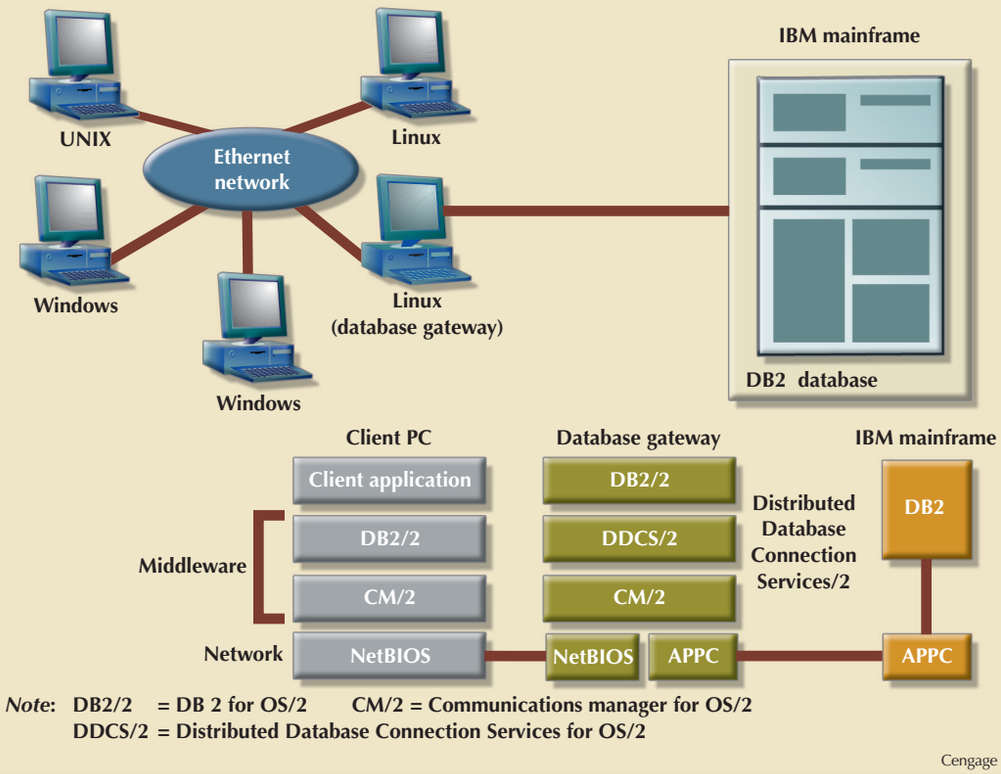
Each module handles the details of each database communications protocol. The network translator layer takes care of using the correct network protocol to access each database. When the data from the query are returned, they are presented in a format common to the client application. The end user or programmer need not be aware of the details of data retrieval from the servers. The end user might not even know where the data reside or from what type of DBMS the data were retrieved.

Another example of how middleware can be used to provide transparent access to databases is shown in Figure F.12. In this case, the network serves several clients that draw their data from an IBM mainframe containing a DB2 database. The clients are Windows Vista, Windows XP, and Linux computers that request data through the network.

Using the bottom of Figure F.12 as a guide, note that the mainframe DB2 database uses the Application Program-to-Program Communications (APPC) protocol to communicate with the computers in the network. A computer in the network is used to translate the TCP/IP requests of the clients into the APPC protocol needed to access the mainframe database. This computer is known as a **gateway**. A gateway computer provides communications translation functions between dissimilar computers and networks. The term *gateway* refers to another type of middleware software; thus, a gateway computer is one that uses gateway middleware. In this case, the middleware software is installed on several computers.

Given the scenario shown in Figure F.12, the client applications request data from the IBM DB2 mainframe database. The DB2 component on the client computer performs some database translator functions. The CM component on the client computer manages the network communications in the network. The gateway computer uses the DB2, DDCS, and CM components to provide database transparency features across the network. The CM on the gateway computer translates the requests from TCP/IP to APPC and sends the requests to the DB2 mainframe database. The middleware components, residing on the client and gateway computers, work together across the network to provide database and network transparency features to all client applications.

FIGURE F.12 Middleware accessing mainframe databases



Note: DB2/2 = DB 2 for OS/2 CM/2 = Communications manager for OS/2
DDCS/2 = Distributed Database Connection Services for OS/2

F.3.7 MIDDLEWARE CLASSIFICATIONS

Database middleware software can be classified according to the way clients and servers communicate across the network. Therefore, middleware software is usually classified as:

- Message-oriented middleware (MOM)
- Remote-procedure-call-based (RPC-based) middleware
- Object-based middleware

Choosing the best-suited middleware depends on the application. For example, RPC-based middleware is probably best for highly integrated systems in which data integrity is paramount, as well as for high-throughput networks. Message-oriented middleware is generally more efficient in local area networks with limited bandwidth and in applications in which data integrity is not quite as critical. Object-based middleware is an emerging type of middleware that is based on object-oriented technology. Although not as widely used as the other two, it promises better systems integration and management.

F.4 Software Infrastructure: Network Protocols

A **network protocol** is a set of rules that determines how messages between computers are sent, interpreted, and processed. Network protocols enable computers to interact in a network and work at different levels of the OSI model. Other terms that are used to label the network protocols are *LAN protocol* and *network transport protocol*. The main network protocols are as follows:

- **Transmission Control Protocol/Internet Protocol** (**TCP/IP**) is the official communications protocol of the Internet, a worldwide network of heterogeneous computer systems. TCP/IP is the main communications protocol used by UNIX systems, is supported by most operating systems at the midrange and personal computer levels, and has become the de facto standard for heterogeneous network connections. Because UNIX is the preferred operating system for medium- and large-scale database servers, TCP/IP is an important player in the client/server arena.
- **Internetwork Packet Exchange/Sequenced Packet Exchange** (**IPX/SPX**) is the communications protocol developed by Novell, one of the world's leading LAN operating systems companies. The IPX/SPX protocol does not behave well when integrated into MANs (metropolitan area networks) or WANs (wide area networks), given their high levels of network traffic. That is why the latest versions of Novell operating systems have adopted TCP/IP as their default network protocol.
- **Network Basic Input/Output System** (**NetBIOS**) is a network protocol originally developed by IBM Corporation and Sytek in 1984 as a standard for PC applications communications. NetBIOS's limitations render it unusable in geographically dispersed internetworks. It is also perceived to be a poorer performer than the IPX/SPX protocol.
- **Application Program-to-Program Communications** (**APPC**) is a communications protocol used in IBM mainframe **Systems Network Architecture** (**SNA**) environments. This protocol allows communications between personal computers and IBM mainframe applications, such as DB2, running on the mainframe. APPC is used in IBM shops to create client/server applications that blend PCs, midrange computers such as the IBM AS/400 and RISC/6000, and mainframe systems.

The TCP/IP protocols are the leading networking protocols in use today. Mainframe network protocols are also used in many companies, especially when the company has a mainframe or minicomputer as its main data repository. As a result of the client/server computing boom, many mainframes and midrange computers are now implementing support for more open, nonproprietary network protocols, such as TCP/IP, to allow direct access from client/server PC-based front-end applications.

The network protocol you select directly affects the software products you can use. For example, an older Novell PC-based database server may be limited to supporting IPX as the network protocol. Most database servers based on UNIX and recent versions of Windows and NetWare use the TCP/IP network protocol. In companies with multiple servers, networks, and clients, the network communications hardware (bridges, routers, and so on) must be able to translate network messages from one protocol to another.

The selection of network topology (covered later in this appendix) and protocols is a critical decision in the development of a client/server system. For commercial software developers, that decision may be market-driven because they want to sell their products to the largest markets as quickly as possible. Therefore, commercial software developers use the network protocols that provide access to the largest number of customers. A commercially developed client/server front-end or back-end application program must support multiple network protocols to communicate with different servers or clients. The network protocol decision may include additional critical variables for MIS systems developers or consultants. For example, does the company already have a network infrastructure in place? Does the company have a mainframe or a wide area network that must be integrated into the system? What type of internal expertise is available? It would not be economically feasible or efficient for commercial

software developers or corporate MIS developers to create applications more than once to support multiple network protocols and multiple database systems.

F.5 Hardware Infrastructure: Cabling and Devices

The primary hardware components of network infrastructure are cabling and devices that permit and regulate network communications.

F.5.1 Network Cabling

Usually, cables are used to physically connect computers and to transmit data between them. There are three main types of network cabling systems: twisted pair, coaxial, and fiber-optic cable. There are also some wireless connection alternatives.

- **Twisted pair cable** is chosen for most installations because it is easy to install and carries a low price tag. Twisted pair cable resembles typical telephone cable and is formed by pairs of wires that are twisted inside a cover. The wires are classified as shielded twisted pair (STP) or unshielded twisted pair (UTP). Quality requirements of twisted pair cable depend on the intended use. Quality is classified by a system that grades the cable's quality and reliability on a scale from category 1 (lowest) to category 6 (highest). The scale reflects cable resistance to electromagnetic interference, electrical resistance, speed, and so on. STP or UTP category 5 or above is recommended for client/server system implementations.
- **Coaxial cable** uses copper cables enclosed in two layers of insulation or shielding. This cable comes in a variety of types and is similar to the cable used for cable TV. The most common varieties used in local area network installations are thicknet and thinnet. Thinnet is cheaper and easier to install than thicknet, but thicknet allows greater distances between computers.
- **Fiber-optic cable** is the most expensive option, but it offers the highest data transmission quality and allows greater distances between computers. This cable is free of electromagnetic interference because it uses laser technology to transmit signals through glass cables. Fiber-optic cable is recommended for the connection of critical network points, such as a connection between two database server computers.
- Wireless communications media, such as satellite and radio, are gaining popularity in connecting remote sites and in providing an alternative to cables in office networks. These media possess great potential in replacing conventional cables in the long run. Several standards (including 802.11n, 802.11b, 802.11g, 802.11a, and others) allow wireless networks to achieve high transmission speeds.

F.5.2 Network Communications Devices

Network communications devices include network interface cards (NICs), hubs, repeaters, concentrators, bridges, routers, and other devices. Those devices allow you to extend and connect networks, even dissimilar ones. They also allow you to mix different cable media within the same network. Because networks are crucial components of client/server architecture, you should know the following basic device descriptions:

- **Network interface cards** (**NICs**) are electronic boards that allow computers to communicate within a network. An NIC interfaces with the physical cable to send and receive signals through the cable medium. In the case of wireless networks, the **wireless adapter**, sometimes called a wireless NIC, allows a computer to communicate using a wireless network.
- A **bridge** is a device that connects similar networks. The bridge, which allows computers in one network to communicate with computers in another network, operates at the OSI model's Data-Link layer and allows two or more networks to be managed as a single logical network.

- The **repeater** is a device used in Ethernet networks to add network segments to the network and to extend the reach of the network. This device, which regenerates the signal and retransmits the signal to all segments, operates at the OSI model's Physical layer.
- A **hub** is a special repeater that allows computers to be added to a network that conforms to a star configuration. A hub will retransmit the packet through all ports (computers); therefore, only one transmission takes place at a time. In the case of wireless networks, a network **access point** allows the connection of wireless devices to a wired or wireless network.
- A **switch** is an intelligent device that connects computers. Unlike a hub, a switch allows multiple simultaneous transmissions between two ports (computers). Therefore, switches have greater throughput and speed than regular hubs.
- A **router** is an intelligent device used to connect *dissimilar* networks. Routers operate at the Network layer and allow a network to span different protocols, topologies, and cable types. A router is frequently used to divide a network into smaller subnetworks. A router also can be programmed to support specific network protocols and provide multiple functions, such as packet filtering and address blocking.
- The **concentrator** is a device that resembles a network wiring closet. It provides multiple functions, such as bridge, router, repeater, and network segmentation, in a single box. Concentrators support different network topologies, cabling, and protocols. Some also provide network management capabilities.

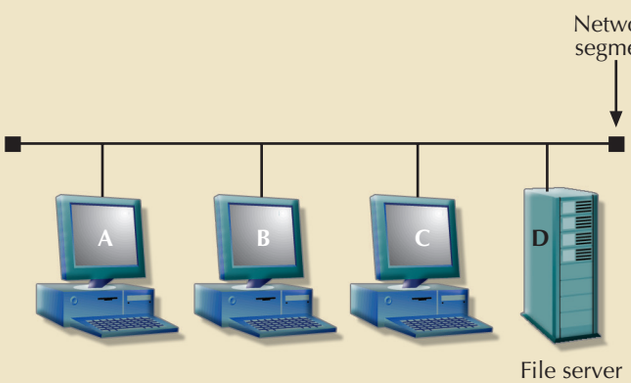
Network devices are used to extend and expand a network's "reach" and to connect existing networks with similar or dissimilar ones. The preceding list is far from exhaustive; it represents only a sampling of the most frequently used network devices. New network communications devices, designed to combine and enhance the capabilities of existing devices, appear at a dizzying rate. Keeping abreast of the new network technology is a full-time, never-ending job in the network world.

F.6 Network Topologies

The term *network topology* refers to the way data travel along the network. Network topology is closely related to the way computers are connected physically. There are three main network topologies, as follows:

- **Bus topology** requires that all computers be connected to a main network cable. In this case, messages traveling through the network are handled by all computers in the bus until they reach their final destination. For example, if A sends a message to D, the message travels through B and C before it reaches D. See Figure F.13.

FIGURE F.13 Bus topology

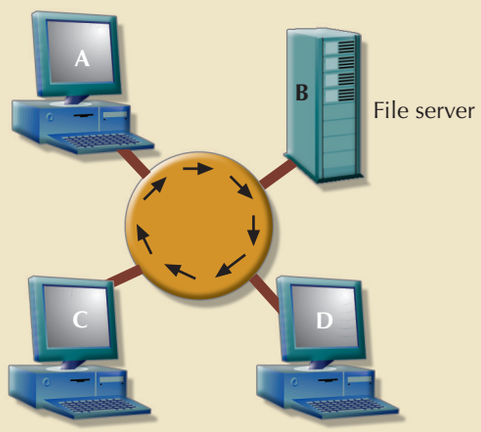


Cengage Learning © 2015

Bus topology, usually implemented through Coaxial cable, is widely used in medium- to small-sized networks. The principal disadvantage of bus topology is that if node B or C breaks down, the entire network segment goes down. (A **network segment** is a single section of cable that connects several computers.)

- **Ring topology** computers are connected to one another through a cabling setup that, as the name implies, resembles a ring. Messages are sent from computer to computer until they reach their final destination. IBM uses the ring topology in its token ring network. The token ring implementation uses a device called a **multiple access unit** (**MAU**) as a wiring concentrator through which the network's computers are connected physically. Ring topology, shown in Figure F.14, is more flexible than bus topology because computers can be added to or disconnected from the ring without affecting the rest of the computers.

FIGURE F.14 **Ring topology**



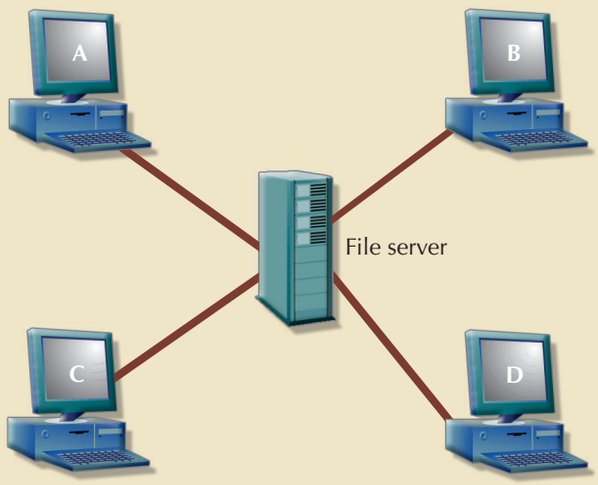
Cengage Learning © 2015

The computers logically exchange messages by passing them along the ring. For a message to be sent from a computer in a token ring network, it must use a token. The **token**, which resembles a baton in a relay race, travels through the ring from computer to computer. Only the computer with the token can transmit at a given time.

- **Star topology** allows all computers to be connected to a central computer, as shown in Figure F.15. Like ring topology, star topology allows workstations to be added to or removed from the network without affecting the operation of the rest of the computers. Unlike ring topology, however, star topology computers are not connected to one another. Instead, they are connected to a central computer. Therefore, all network messages travel through the central computer.

  The network topology is independent of the cabling system used. Any network can use coaxial, twisted pair, or fiber-optic cable.

FIGURE F.15 Star topology



F.7 Network Types

Networks are usually classified by the extent of their geographical area coverage: local area, campuswide, metropolitan area, and wide area networks.

- A **local area network** (LAN) typically connects PCs in an office, a department, a floor, or a building. The LAN is the most frequently encountered network type and is preferred when workgroups are connected. There are two main LAN types: Ethernet and token ring. **Ethernet** is based on a bus or star topology that can use coaxial, twisted pair, or fiber-optic cabling. Most Ethernet LANs transfer data at a speed of l00 Mbps (one hundred million bits per second). **Token ring networks** are based on a ring topology that can use shielded twisted pair (STP), unshielded twisted pair (UTP), or fiber-optic cabling. Token ring networks can transfer data at speeds of 4 Mbps to 16 Mbps. Today, token ring networks have fallen out of favor, and are being replaced by most organizations with Ethernet networks. In addition, there are **wireless LANs** (**WLANs**) that can be configured according to several different standards.
- A **campuswide network** (**CWN**) is the typical college or university network in which buildings containing LANs and often WLANs are (usually) connected through a main network cabling system known as a **network backbone**.
- The **metropolitan area network** (**MAN**) is used to connect computers across a city or metropolitan area. The MAN is designed to cover much more territory than the CWN. It can even be used to connect CWNs located within a city or metropolitan area.
- A **wide area network** (**WAN**) is used to connect computer users across and between countries. The MAN and WAN generally make use of telephone and specialized communications companies to connect networks in sites separated by great distances.

F.8 Network Standards

Because client/server computing is focused on the *sharing* of resources, adherence to network standards is crucial. Fortunately, the Institute of Electrical and Electronics Engineers (IEEE) developed standards to provide uniformity among networks. Those IEEE standards specify the technical details that define network topology and data transmission across shared media. In addition, the IEEE standards yield the rules that govern network cabling, cable distances between computers, devices used in networks, and so on. Three important IEEE network standards are:

- **IEEE 802.3**: Ethernet networks.
- **IEEE 802.5**: Token ring networks.
- **IEEE 802.11 and 802.16**: Wireless networks.

F.9 The Quest For Standards

Standards ensure that dissimilar computers, networks, and applications can interact to form a system. But what constitutes a standard? A *standard* is a publicly defined method to accomplish specific tasks or purposes within a given discipline or technology. Given the use of standards, it is possible to use a TV set to receive video from different broadcasters, to use a DVD player manufactured by different companies located in different countries, and so on. Standards make networks practical.

There are several organizations whose members work to establish the standards that govern specific activities. For example, the **Institute of Electrical and Electronics Engineers** (**IEEE**) is dedicated to defining standards for network hardware. Similarly, the **American National Standards Institute** (**ANSI**) has created standards for programming languages such as COBOL and SQL. The **International Organization for Standardization** (**ISO**) produced the Open Systems Interconnection (OSI) reference model to achieve network systems communications compatibility.

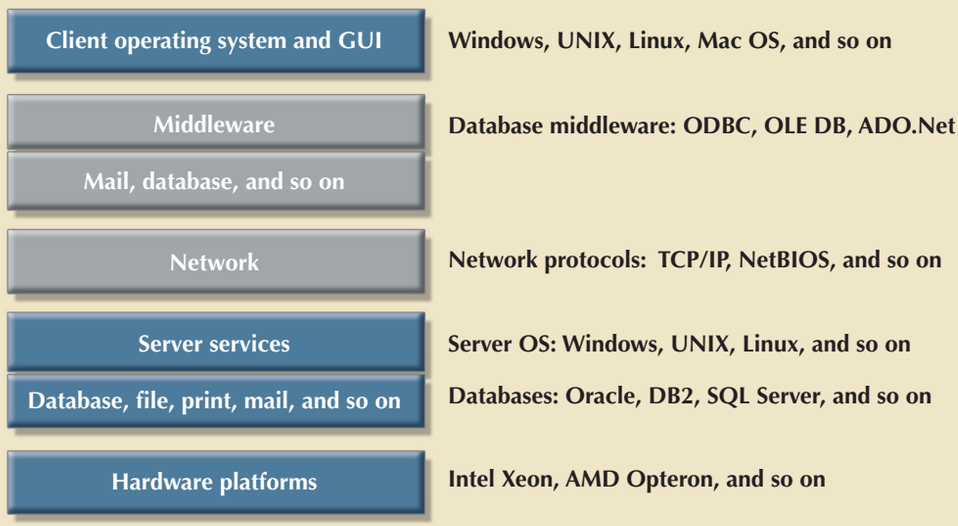
Truly universal standards for all client/server components do not yet exist. There are many different standards from which to choose. There are standards for the user interface, data access, network protocols, interprocess communications, and so on.

For example, a system might use ODBC, OLE DB, or ADO.Net database middleware. **Open Database Connectivity** (**ODBC**), developed by Microsoft Corporation and the de facto standard for database middleware, is designed to provide Windows applications with an API that is independent of the data source. ODBC also provides applications programmers with a generic format for data access. A specific ODBC driver (for example, an Oracle ODBC driver or a SQL Server ODBC driver) must be used for each database being accessed. ODBC requires the database communications protocol to be present—for example, TCP/IP or SQL*Net—for communication to take place with the database server. ODBC also provides the capability to access database-specific options if they are required by the client application. Microsoft also offers OLE DB and ADO.Net as other alternatives for database connectivity.

An application that does not use a single standard can still be a client/server application. The point is to ensure that all components (server, clients, and communications middleware) are able to interact as long as they use the same standards. What really defines client/server computing is that application processing is split into client and server components.

Figure F.16 shows some of the options available to client/server systems developers. Ultimately, the objective is to have options that allow systems to interact regardless of the selection made from this list, thus producing a plug-and-play environment.

FIGURE F.16 Client/server options



Ultimately, standards must be developed that provide systems interoperability at all levels. Recent technological advances have removed some major systems integration barriers, thus setting the stage for realizing a client/server environment that was just a dream only a few years ago: standards-based systems that function seamlessly across operating systems, graphical user interfaces, networks, and hardware platforms.

F.10 Client/Server DBMSs

A database management system (DBMS) lies at the center of most client/server systems in use today. To function properly, the client/server DBMS must be able to:

- Provide transparent data access to multiple and heterogeneous clients, regardless of the hardware, software, and network platform used by the client application.
- Allow client requests to the database server (using SQL requests) over the network.
- Process client data requests at the local server.
- Send only the SQL results to the clients over the network.

A client/server DBMS reduces network traffic because only the rows that match the query are returned. Therefore, the client computer resources are available to perform other system chores such as managing the graphical user interface. Client/server DBMSs differ from other DBMSs in terms of where the processing takes place and what data are sent over the network to the client computer. However, client/server DBMSs do not necessarily require distributed data.

Client/server systems change the way in which data processing is approached. Data may be stored in one site or in multiple sites. When the data are stored in multiple sites, client/server databases are closely related to distributed databases. (See Chapter 12.) Distributed client/server database systems (DDBMSs) must have the following characteristics:

- The *location* of data is transparent to the user. The user does not need to know what the data location is, how to get there, or what protocols are used to get there.

- Data can be *accessed* and *manipulated* by the end user at any time and in many ways. Powerful applications stored on the end user's side allow access and manipulation of data in ways that had never before been available. The data request is processed on the server side; the data formatting and presentation are done on the client side.
- The *processing* of data (retrieval, storage, validation, formatting, presentation, and so on) is distributed among multiple computers.

The distinctions between client/server systems and DDBMSs are sometimes blurred. The client/server system distributes data processing among several sites. The DDBMS distributes data at different locations. In other words, client/server systems and DDBMSs involve mainly complementary functions, as well as some overlapping functions. In fact, DDBMSs use distributed processing to access data at multiple sites. Therefore, the DDBMS resembles a client/server implementation.

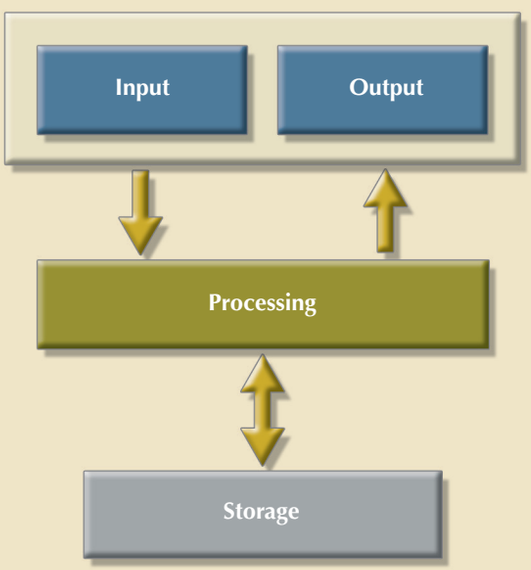
F.11 Client/Server Application-Processing Logic

Because the division of the application-processing logic components is a prime client/server characteristic, the two key questions that every client/server systems designer must answer are these:

- How is the division to be made?
- Where in the system should the results of that division be placed?

To answer those questions, you must first look at application-processing logic components. (See Figure F.17.)

FIGURE F.17 Application logic components



Cengage Learning © 2015

Figure F.17 illustrates that an application's logic can be divided into three main components: input/output, processing, and storage.

- The *input/output* (*I/O*) component works to format and present data in output devices such as the screen, and manages the end-user input through devices such as the keyboard. For example, the input logic shows a menu screen, waits for the end user to enter data, and then responds to the data entry. (Within this I/O component, the application uses *presentation logic* to manage the graphical user interface and data formatting.)
- The *processing* component refers to the application code that performs data validation, error checking, and so on. The processing component's logic represents the business rules and the data management logic for data retrieval and storage. For example, the processing logic "knows" that a sales transaction generates an invoice record entry, an inventory update, and a customer's accounts receivable entry. The processing logic performs several functions, including managing input and output, enforcing business rules, managing information flows within a business, and mapping the real-world business transactions to the actual computer database. Therefore, the processing component can be further divided into three functional subcomponents, as follows:
  1. *I/O processing logic* manages data entry validation and basic error checking.
  2. *Business logic* is applied through the code that represents the business rules.
  3. *Data management logic* determines which data are needed for each business transaction. For example, a sales transaction might require vendor, customer, and product data.
- The *storage* component uses *data manipulation logic* to deal with the actual data storage to and retrieval from the physical storage devices. For example, data manipulation logic is used to access the files and to check for data integrity.

In short, the three main client/server application logic components can be subdivided into the following five main functional logic components:

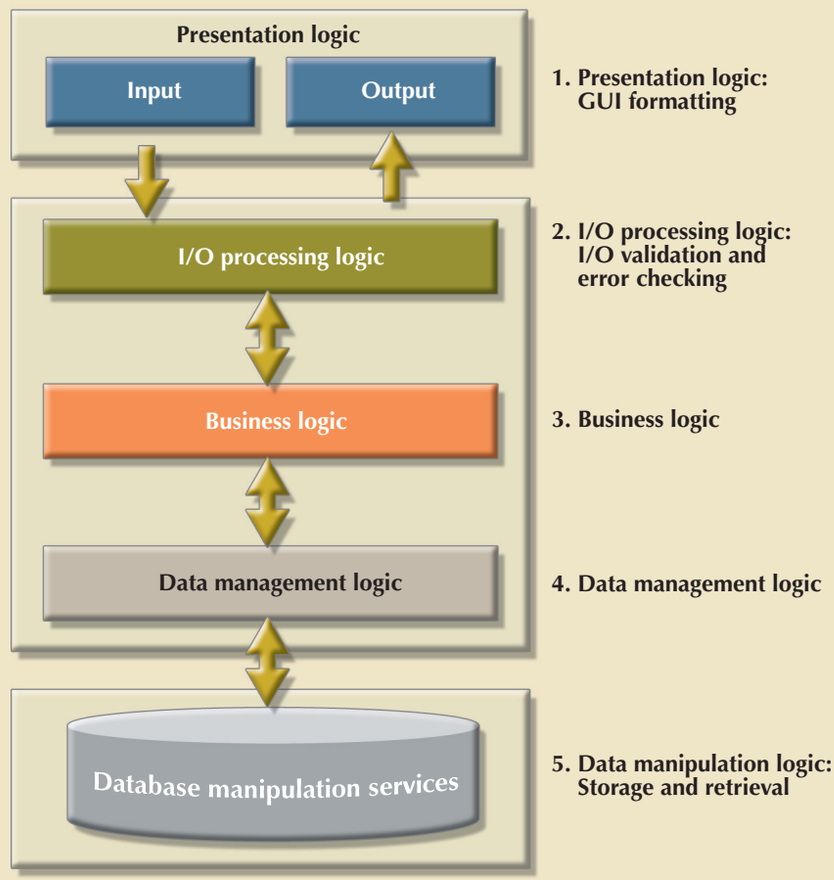
- Presentation logic
- I/O processing logic
- Business logic
- Data management logic
- Data manipulation logic

Figure F.18 shows the five functional components that form the basis for splitting the application logic processing in the client/server model.

Although there is no methodology to dictate the precise distribution of the logic components shown in Figure F.18 among clients and servers, the client/server architectural principles of process distribution (autonomy, resource maximization, scalability, and interoperability) and hardware and software independence may be used as a guide.

So where should each component be placed? With the probable exception of the presentation logic, which normally goes on the client side, each of the remaining components may be placed on the server, thus becoming a service for all of the clients. Given that arrangement, even the purest mainframe environment can be classified as client/server. It may even be argued that a mainframe resembles a primitive client/server incarnation in which the mainframe provides services to dumb, rather than intelligent, terminals. However, if the objective is to create a distributed environment, the pure mainframe yields few advantages compared to the naturally distributed client/server architecture, in which clients are not necessarily entirely dependent on a single server. What's more, the pure mainframe's architecture is not designed to allow the distribution of the various functional areas because the other services cannot be split out of the mainframe to be placed on other computers. (That is not, of course, referring to mainframes that are part of a client/server setup!) In fact, given a pure mainframe environment, none of the processing logic components is split: only the presentation logic can be kept on the client. The price of such architectural rigidity is high because, as has been illustrated several times in this appendix, the greatest distributed processing benefits are obtained when the processing logic is split between server(s) and client(s).

FIGURE F.18 Application functional logic components



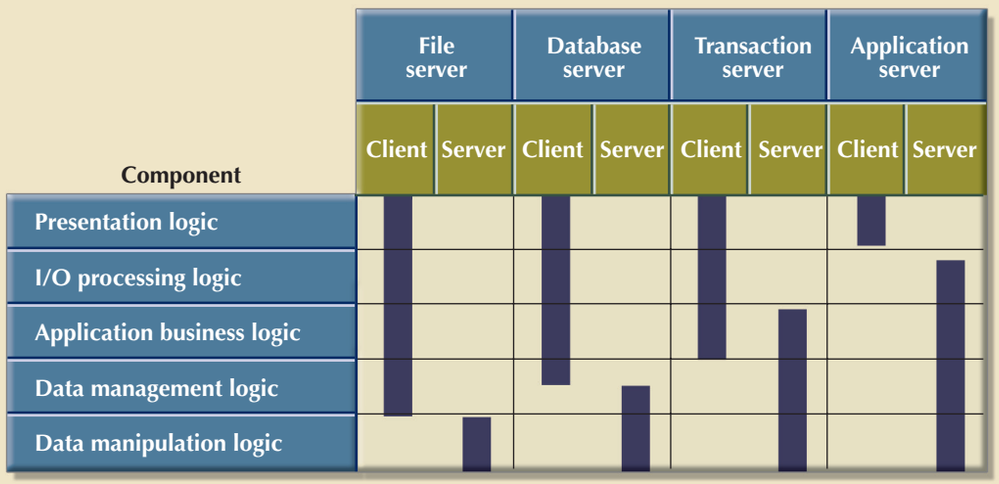
The location combinations reflect different computing styles. For example, all components for a typical home computer are located on a single PC. The pure mainframe style reflects a condition in which only the data presentation takes place on the client side, whereas all other processing takes place on the mainframe side. It is not practical to put each component on a unique server—saving only the presentation logic for the client side—though it can be done.

Although it is possible to select any combination of logic component locations, practical considerations require that specific services such as file, print, communications, and fax be logically identified and separated, and that a decision then be made on the placement of each component. The following placement is typical:

- The *presentation logic* is always placed on the client side because it is required for end-user interaction. The GUI usually provides the services to the front-end application services.
- The *I/O processing logic* may be placed on the client side or on the server side. Although it is most commonly located on the client side in the client/server model, it may be placed on the server side when a fat server/thin client implementation exists. (Naturally, the latter scenario is the norm when the mainframe model is considered.) If a three-tier client/server system is used, the intermediate servers usually contain all of the I/O processing logic, thus making it available to all clients.

- The *business logic* can also go to either the client or the server. However, it is usually located on the client side. This logic component can also be split into client and server subcomponents. If a three-tier client/server system is used, the intermediate servers usually contain all of the business logic. Given this three-tier arrangement, changes in business logic are available to all clients.
- The *data management logic* can also be placed on either the client or the server side. However, it is normally placed on the client side or on an intermediate business logic server. The data management logic can also be split into client and server subcomponents, as is done in database middleware. Or in the case of distributed databases, the subcomponents can be placed within multiple server computers.
- The *data manipulation logic* is most commonly located on the server side. However, the data manipulation logic can also be divided among several computers in the distributed database environment.
- The split and distribution of the application-processing components are also a function of the architectural style. Figure F.19 shows the likely distribution of application-processing components within the four basic client/server architectural styles: the file server model, the database server model, the transaction server model, and the application server model.

FIGURE F.19 **Functional logic splitting in four client/server architectural styles**



As you examine Figure F.19, keep in mind that the server side provides services for many clients. Further, the server column represents one or more server computers. Examine Figure F.19 with the following details in mind:

- The *file server* architectural style reflects a setup in which the client does most of the processing, whereas the server side manages only the data storage and retrieval. If a client application wants to select some database table rows, the actual selection of the records takes place in the client rather than in the server.
- The data management logic is split between the client and the server computers in the *database server* architectural style. For example, the client sends a SQL request to the server and the server executes it locally, returning only the requested rows to the client. Keep in mind that there may be many servers and that a client may access many servers concurrently. If the client application executes a transaction that requires access to multiple servers, the client computer must address all of the transaction management details. Therefore, each SQL transaction must travel from the client to the server, thus increasing network traffic.

- The *transaction server* architectural style permits the sharing of transaction details between the client and the server. For example, if the server side has some knowledge about the transaction details, some of the business logic must reside on the server. This architectural style is favored when the application transaction details are known beforehand (that is, they are not ad hoc) and do not change very often. In this scenario, some business logic is stored on the server in the form of SQL code or some other DBMS-specific procedural language. Such stored code, usually known as stored procedures (see Chapter 8, Advanced SQL) is verified, compiled, and stored in the DBMS. The client application merely calls the stored procedure, passing it the necessary parameters for its execution. No code travels through the network, and the transaction server can be connected to many database servers.
- The *application server* architecture makes it possible to enjoy the benefits of client/server computing even when the client computers are not powerful enough to run some of the client/server applications. This architectural style allows any application to reside on a powerful computer known as the application server and then be executed and shared by many less powerful clients. In this case, all of the processing is done on the application server side, and the client computers deal just with the application output presentation. The application server architectural style is favored when it is necessary to use remote control computers over a network or when office workers are likely to require access to their office desktop PCs through their home phones.

The use of one architectural style does not preclude the use of another. In fact, it is possible to create several server "layers" by daisy chaining the server processes. For example, an application server may access a transaction server that, in turn, may access multiple database servers. It is possible to have several client/server computing styles supported concurrently within the same network. Such flexibility makes it imperative that the client/server network infrastructure be carefully planned to enable it to support diverse client/server information requirements. The client/server model's ability to work with multiple servers is also the reason it works so well as an integrating platform on which personal computers, minicomputers, and mainframes can be brought together in a seamless fashion.

NOTE

The web application server represents a new computing style that integrates all of the architectural styles presented in this appendix. A web server acts like a file server; the web server transmits files to clients for execution. At the same time, the web server acts like a transaction server to coordinate database access by multiple clients. The web server is also able to provide session status control for each client as it accesses the server, effectively behaving like an application server. (The web application server model is discussed in detail in Chapter 14, Database Connectivity and Web Technologies.)

F.12 CLIENT/SERVER IMPLEMENTATION ISSUES

Implementing client/server systems is best described as a challenge. The development of client/server systems differs greatly in process and style from the traditional information systems development methods. For example, the systems development approach, oriented toward the centralized mainframe environment and based on traditional programming language, can hardly be expected to function well in a client/server environment that is based on hardware and software diversity. In addition, modern end users are more demanding and are likely to know more about computer technology than users did before the PC made its inroads. Therefore, MIS department managers are constantly racing the knowledge clock to assimilate new technologies that are based on multiple platforms, multiple GUIs, multiple network protocols, and so on. In addition, MIS managers must cope with rapid application development as well as the issues that arise from greater end-user autonomy in information management.

This section explores some of the managerial and technical issues involved in the development and implementation of client/server systems. Discussion begins by examining how the client/server and traditional data-processing models differ. Next, you will examine the management issues that arise from the adoption of the client/server model. Then some basic technical issues will be presented. Finally, a basic framework will be developed within which you can approach the development and implementation of client/server systems.

F.12.1 Client/Server Versus Traditional Data Processing

You already know that the new client/server computing model environment is more complex technically than the traditional data-processing model because the former may be based on multiple platforms, operating systems, and networks. Yet the basic technical characteristics of the client/server model cannot explain why it has set the stage for another information shake-up. Instead, it is more important to note that the client/server model changes the way you look at the most fundamental data-processing issues. The client/server model's impact is far greater than the measure of its technological prowess alone.

Client/server computing expands the reach of information systems, thus changing how things are done and creating information-aware end users who will not settle for less than information autonomy. End users traditionally relied on the MIS department for information; they now create their own information by tapping into a common data pool and producing their own queries and reports to support their decision making.

This new view of the information world creates a paradox. On the one hand, end users have declared their independence from the MIS department. But on the other hand, end users have become very dependent on the client/server infrastructure (servers, networks, middleware, and client front ends) that is managed by—you guessed it—the MIS department. Clearly, client/server computing has introduced major changes from traditional data processing.

- *From proprietary to open systems*. Traditional data-processing architecture is typically based on single-vendor solutions. Integrating multiple-vendor products within this architecture was a difficult and often impossible task. The new client/server environment demands systems that are easily integrated—systems that are open to other systems.
- *From maintenance-oriented coding to analysis, design, and service*. Given the traditionally centralized mainframe environment, most of the MIS department's focus was on application maintenance. Although client/server systems do require substantial infrastructure maintenance, the MIS department that manages such systems spends the greater portion of its time on end-user support functions. Traditional systems development life cycles dedicated most of their time to limited-use application coding and maintenance. The client/server environment changes the role of programmers by letting them use sophisticated 4GL, CASE, and other development tools to free them from coding. The price tag for using the new tools is that programmers must spend more time on systems analysis and design because errors tend to be very costly. (For one thing, end-user autonomy means that errors will be more widespread.) In short, the focus changes from coding to design.
- *From data collection to data deployment*. Instead of focusing on centralized data storage and data management, the client/server-based MIS department must concentrate on making data more easily and efficiently available to end users.
- *From a centralized to a more distributed style of data management*. Typical data management in the traditional mainframe environment was tightly structured and required rigid procedural enforcement. The new client/server environment requires a more flexible data management style. Characterized by a more decentralized management and decision-making approach, client/server computing forces a shift in focus toward the solution of end-user information problems and customer needs.
- *From a vertical, inflexible organizational style to a more horizontal, flexible organizational style*. Traditional MIS department structures will be flattened. Direct data access empowers individual users to be more information-independent from the MIS department. Consequently, the MIS department must modify or restructure its activities to accommodate people with diverse PC, GUI, and network skills.

The change in the data-processing environment brought about by client/server computing may also be evaluated by examining information systems components.

- *Hardware*. Information systems are no longer single-vendor-dependent; instead, they are likely to integrate many different hardware platforms.
- *Software*. Traditional systems consisted of procedural language routines written in a 3GL such as COBOL or FORTRAN and supported character-based applications only. All of the processing was done by the mainframe. New client/server systems are the result of the integration of many routines created by and supported by graphical user interfaces, databases, networks, and communications. The new systems split application processing into many subcomponents that integrate seamlessly. Usually, these new systems are created through the use of languages such as Visual Basic, C++, and Java.
- *Data*. Traditionally, data were centralized within a single repository. New systems tend to distribute data among multiple computers, thus putting data closer to the end user. In addition, multiple data formats (sound, images, video, text, and so on) are available.
- *Procedures*. Traditional systems were based on centralized procedures that were very rigid and complex. New distributed systems have made the procedures more flexible and decentralized.
- *People*. Client/server computing changes people's roles and functions. Updated skills are required to support and use the new technology, thus demanding intensive training and retraining to stay up to date. Such changes are not limited to the MIS department, but are spread throughout the organization.

F.12.2 Managerial Considerations

You have seen how client/server systems change the data-processing style and how those changes affect the organization. You will now look at some of the managerial issues that result from the introduction of client/server systems. Those issues are based on managing multiple platforms, hardware, networks, operating systems, and development environments and on dealing with multiple vendors.

- *Management and support of communications infrastructure*. One of the most complex issues in client/server environments is management of the communications infrastructure (network hardware and software). Managers must deal with several layers of network equipment to make sure that equipment from multiple vendors works together properly. The situation is especially complex because there are no comprehensively integrated client/server network management tools. Mainframe systems administrators are often afraid of the changes induced by the client/server environment because they are used to the integrated management tools that mainframe systems provide. Managers cannot count on equivalent comprehensive monitoring and management tools to support the client/server environment.
- *Management and support of applications*. Client/server applications are characterized by the distribution of processing among multiple computers. Each of those computers may be running a different operating system. An enterprise client/server system may support multiple GUIs (Windows and Apple Macintosh) on the client side, several operating systems (Novell NetWare, UNIX, or Windows Server on the server side, and Windows 8, Windows 7, Windows Vista, and Apple Macintosh on the client side), and appropriate versions of middleware components. Managers must ensure that all of the components maintain current version levels at all stages: client application modules, middleware components, network components, and the back-end server side. Fortunately, there are software tools that use the network to distribute and update software automatically at the client computers. End-user support may be enhanced by creating a Help Desk to give end users a central point of support for all of their computer needs. Help Desk personnel staffing and training must be the priority of MIS management.

- *Control of escalating and hidden costs*. Client/server systems generally are expected to reduce MIS costs. Part of such cost reduction is based on the economy of scale enjoyed by the personal computer industry. (You can afford to sell complex database software for $299 when you expect to sell 2 million copies.) Although cost reductions are expected when client/server solutions are compared to an all-mainframe alternative, there are significant startup costs. In fact, costs associated with the adaptation of resources (personnel and computer systems) to the client/server environment might be higher than expected because:
  - It may be difficult and expensive to find personnel who have the right mix of wide-ranging skills.
  - Training (or retraining) of data-processing staff, managerial personnel, and end users can be time-consuming and expensive.
  - The acquisition of sophisticated new hardware and software technologies is expensive.
  - Establishing new procedures or adapting existing procedures to the new system can be cumbersome.

The initial costs associated with client/server computing must be treated as an investment that is likely to yield good returns through subsequent savings in new systems development, increased flexibility, and improved customer service benefits. Nevertheless, the hidden costs of maintaining and supporting the client/server environment must be carefully determined in the planning stage. (One hidden cost that managers often overlook is the "people cost" of retraining—retraining not only the data-processing personnel, but also the managers and end users. Such educational investments help minimize the culture shock associated with the freedom of information management fostered by client/server computing.) The cost of implementing client/server computing must include the following:

- *Managing people and cultural changes*. Dealing with the psychological impact of the employees' changing roles is a never-ending task. Managers must involve the end user in the implementation of the client/server infrastructure. In the long run, the effectiveness of the system depends on whether end users put it to good use. Although the garbage-in-garbage-out (GIGO) phenomenon is encountered in any system, the wide reach of client/server computing and the power of its applications make it especially easy for users to make bigger mistakes—and make them faster. Managers also must understand that not all end users are equal. Some will be eager to learn how to use SQL or a graphical query tool to get data or to produce a report, whereas others may still be very dependent on the MIS department for the required information. The MIS department must develop and implement a gradual and progressive educational plan.
- *Managing multiple vendor relationships*. In the past, mainframe MIS managers could dial a single phone number to find solutions to hardware and software problems. In contrast, the client/server environment forces the MIS manager to deal with multiple vendors. Therefore, managers often must develop partnership-like relationships with vendors to ensure that the multiple-vendor environment works satisfactorily.

F.12.3 Client/Server Development Tools

In today's rapidly changing environment, choosing the right tools to develop client/server applications is a critical decision. As a rule of thumb, managers tend to choose a tool that has long-term survival potential. However, the selection of a design or application development tool must also be driven by the system requirements. Once such requirements have been delineated, it is appropriate to determine the characteristics of the tool you want to have. Client/server tools include:

- GUI-based development
- A GUI builder that supports multiple interfaces
- Object-oriented development with support for code reusability
- A data dictionary with a central repository for data and applications
- Support for multiple databases (relational, network, hierarchical, and flat file)
- Data access regardless of data model (using SQL or native navigational access)
- Seamless access to multiple databases
- Complete systems development life cycle support from planning to implementation and maintenance

- Team development support
- Support for third-party development tools (CASE, libraries, and so on)
- Prototyping and rapid application development (RAD) capabilities
- Support for multiple platforms (operating systems, hardware, and GUIs)
- Support for middleware protocols (ODBC, IDAPI, APPC, and so on)
- Support for multiple network protocols (TCP/IP, IPX/SPX, NetBIOS, and so on)

There is no single best choice for any application development tool. For one thing, not all tools will support all GUIs, operating systems, middleware, and databases. Managers must choose a tool that fits the application development requirements and that matches the available human resources, as well as the hardware infrastructure. Chances are the system will require multiple tools to make sure that all or most of the requirements are met. Selecting the development tools is just one step. Making sure the system meets its objectives at the client, server, and network levels is another issue.

F.12.4 An Integrated Approach

The development of client/server systems is based on the premise that those systems are effective in helping management reach the organization's goals. Client/server-based systems should never be developed because the available tools are so technically advanced or because management wants to ride a new technology wave. Remember that client/server technology is one possible road to an objective; it is not the objective.

If a thorough study of the client/server system's technical and human dimensions indicates that its use can help achieve desired ends, a marketing plan should be developed before the client/server design and development effort is started. The objective of that plan is to build and obtain end-user and managerial support for the future client/server environment. Although there is no single recipe for the process, the overall idea is to conceptualize client/server systems in terms of their scope, optimization of resources, and managerial benefits. In short, the plan requires an integrated effort across all departments within the organization. If the client/server decision is being made for the first time, such an effort includes the following six phases:

1. *Information systems infrastructure self-study*. The objective is to determine the actual state of the available computer resources. The self-study will generate at least the following:
   - A software and hardware inventory.
   - A detailed and descriptive list of critical applications.
   - A detailed human resources (personnel and skills) inventory.
   - A detailed list of problems and opportunities.
2. *Client/server infrastructure definition*. The output of phase 1, combined with the company's computer infrastructure goals, is the input for the design of the basic client/server infrastructure blueprint. This blueprint will address the main hardware and software issues for the client, server, and networking platforms.
3. *Selection of a window of opportunity*. The next phase is to find the right system on which to base the client/server pilot project. After identifying the pilot project, you must define it very carefully by concentrating on the problem(s), the available resources, and a set of clear and realistic goals. Describe the project in business terms rather than in technological jargon. When defining the system, make sure to plan carefully for costs. Try to balance the costs with the effective benefits of the system. Also make sure to select a pilot implementation that provides immediate and tangible benefits. A system that takes two years to develop and another three to generate tangible benefits is not acceptable.

4. *Management commitment*. Top-to-bottom commitment is essential when you are introducing new technologies that affect the entire organization. You also need managerial commitment to ensure that the necessary resources (people, hardware, software, money, and infrastructure) will be available and dedicated to the system. A common practice is to designate a person to work as a guide, or an agent of change, within the organization's departments. The main role of this person is to ease the process that changes people's roles within the organization.
5. *Implementation*. Guidelines to implementation should include at least:
   - Using "open" tools or standards-based tools.
   - Fostering continuing education in hardware, software, tools, and development principles.
   - Looking for vendors and consultants to provide vendor-specific training and implementation of designs, hardware, and application software.
6. *Review and evaluation*. Make sure that the systems conform to the criteria defined in phase 3. Continuously measure system performance as the system load increases, because typical client/server solutions tend to increase the network traffic and slow down the network. Careful network performance modeling ensures that the system performs well under heavy end-user demand conditions. Such performance modeling should be done at the server end, the client end, and the Network layer.

KEY TERMS

access point, F-21
American National Standards Institute (ANSI), F-24
application programming interface (API), F-15
Application Program-to-Program Communications (APPC), F-19
back-end application, F-6
bridge, F-20
bus topology, F-21
campuswide network (CWN), F-23
client, F-2
client/server architecture, F-5
coaxial cable, F-20
concentrator, F-21
database translator, F-16
Ethernet, F-23
fat client, F-2
fat server, F-2
fiber-optic cable, F-20
frame, F-14
front-end application, F-6
gateway, F-17
hub, F-21
IEEE 802.3, F-24
IEEE 802.5, F-24
IEEE 802.11 and 802.16, F-24
imaging server, F-6
Institute of Electrical and Electronics Engineers (IEEE), F-24
intelligent terminals, F-3
International Organization for Standardization (ISO), F-24
Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX), F-19
interprocess communication (IPC), F-11
local area network (LAN), F-23
metropolitan area network (MAN), F-23
middleware, F-6
multiple access unit (MAU), F-22
network backbone, F-23
Network Basic Input/Output System (NetBIOS), F-19
network interface cards (NICs), F-20
network operating system (NOS), F-4
network protocol, F-19
network segment, F-22
network translator, F-16
Open Database Connectivity (ODBC), F-24
Open Systems Interconnection (OSI), F-12
repeater, F-21
ring topology, F-22
router, F-21
server, F-2
sneakernet, F-3
star topology, F-22
switch, F-21
Systems Network Architecture (SNA), F-19
thin client, F-2
thin server, F-2
three-tier client/server system, F-3
token, F-22
token ring networks, F-23
Transmission Control Protocol/Internet Protocol (TCP/IP), F-19
twisted pair cable, F-20
two-tier client/server system, F-3
wide area network (WAN), F-23
wireless adapter, F-20
wireless LANs (WLANs), F-23

REVIEW QUESTIONS

1. Mainframe computing used to be the only way to manage enterprise data. Then personal computers changed the data management scene. How do those two computing styles differ, and how did the shift to PC-based computing evolve?
2. What is client/server computing, and what benefits can be expected from client/server systems?
3. Explain how client/server system components interact.
4. Describe and explain the client/server architectural principles.
5. Describe the client and the server components of the client/server computing model. Give examples of server services.
6. Using the OSI network reference model, explain the function of the communications middleware component.
7. What major network communications protocols are currently in use?
8. Explain what middleware is and what it does. Why would MIS managers be particularly interested in such software?
9. Suppose you are currently considering the purchase of a client/server DBMS. What characteristics should you look for? Why?
10. Describe and contrast the client/server computing architectural styles that were introduced in this appendix.
11. Contrast client/server data processing and traditional data processing.
12. Discuss and evaluate the following statement: There are no unusual managerial issues related to the introduction of client/server systems.

PROBLEMS

1. ROBCOR, a medium-sized company, has decided to update its computing environment. ROBCOR has been a minicomputer-based shop for several years, and all of its managerial and clerical personnel have personal computers on their desks. ROBCOR has offered you a contract to help the company move to a client/server system. Write a proposal that shows how you would implement such an environment.
2. Identify the main computing style of your university computing infrastructure. Then recommend improvements based on a client/server strategy. (You might want to talk with your department's secretary or your advisor to find out how well the current system meets their information needs.)