CHAPTER 26
Client/Server Technology

ON HASHIDA
University of Tsukuba

HIROYUKI SAKATA
NTT DATA Corporation

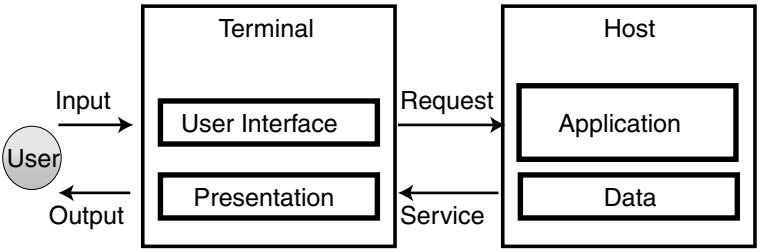

Figure 1 Host-Centered Processing System.

1. INTRODUCTION

1.1. Concept of C/S Systems

In a traditional, centralized mainframe system, a user interacts with an application via a dumb terminal that has a keyboard for entering commands and a display to show the results. All applications and databases are placed on the mainframe (host), and the terminal does not have any ability to process applications. That is, the functions of the user interface and the presentation are placed on the terminal and other applications are placed on the host side, as shown in Figure 1.

In a client/server (C/S) system, by contrast, a user advances processing by using a program called a server, from a workstation or personal computer connected to the server computer by network. The program that receives services from the server is called a client. As in the host-centric system, the functions of user interface and presentation are placed on the user (client) side. Databases that are used for processing requests are placed on the server side. Each application is divided into several processes that run on the client and server sides and cooperate for processing the application. The difference between the two systems is the way of distribution of applications. As shown in Figure 2, some applications can be placed on the user side in the C/S system. Figure 3 shows the physical and logical structures for a host-centric system and a C/S system.

In the physical structure of the host-centric system, applications and databases are placed on a mainframe. That is, the physical structure is almost the same as the logical structure described above. On the other hand, in the C/S system, all the processes that comprise the service are not necessarily placed in the same computer. In many cases, they are distributed in separate computers connected to each other via network.

The features of C/S systems are:

- They contain the basic functional components: presentation, application service, and database.
- Generally, each application service is divided into several processes.
- In many cases, the service is offered by cooperative work of several processes placed on separate computers.

1.2. Roles of Client and Server

There is a many-to-one relationship between clients and servers, but the relationship is relative. In some cases, a client may pass a reference to a callback object when it invokes a service. This lets

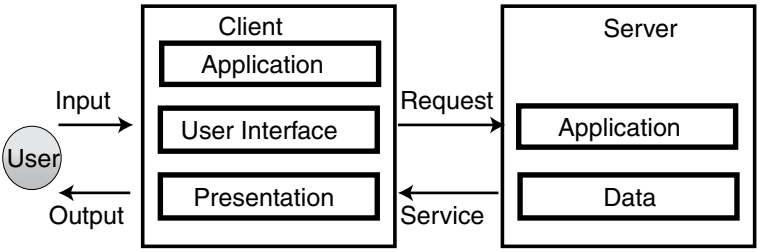

Figure 2 Relationship of Client and Server.

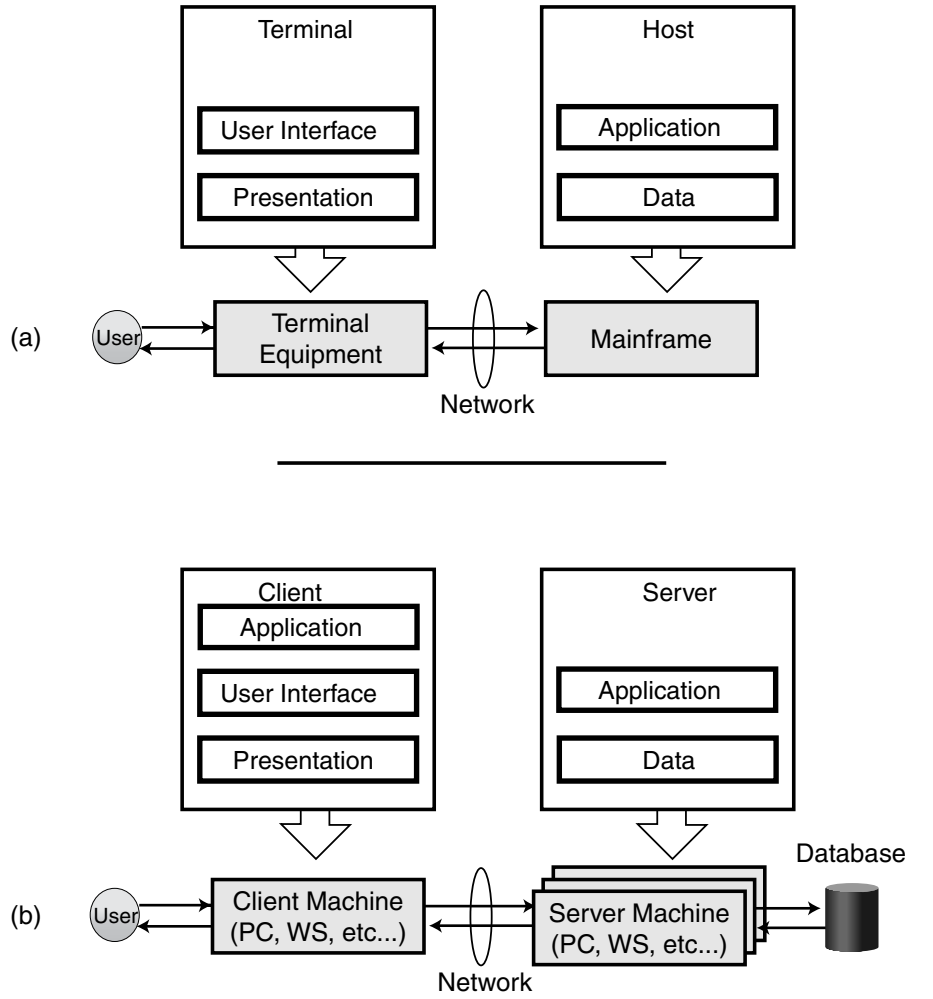

Figure 3 Logical Structure vs. Physical Structure. (*a*) Host processing model. (*b*) C/S processing model.

the server call back the client, so the client becomes a server. For example, in an Internet banking system, a user program (WWW browser) is a client and a program located at the bank that offers services to the client is a server. However, the role of the program located at the bank is not always fixed to the server. To offer a fund transfer service to the user, the program located at bank A asks for a fund acceptance process that is located at bank B. In this case, the program at bank A becomes a client and the program at bank B becomes a server. Thus, the roles of the client and server change dynamically depending on the circumstances. In this chapter, ''client'' means a program that is located on the user side and provides the user interface and presentation functions, and ''server'' means a program that receives requests from the client and provides services to the client.

1.3. Computer Technology Trends

1.3.1. Web Technologies

A C/S system is realized by a network over which clients and servers work together to accomplish a task. To develop the network applications, it is important for a developer to select an appropriate communication protocol from the viewpoint of cost and performance. The communication protocol

is a set of rules by which reliable end-to-end communication between two processes over a network is achieved. In recent years, the TCP/IP protocol stack has been adopted for constructing many network applications and has become the standard protocol for the Internet. World Wide Web (or Web) technologies using HTTP (Hypertext Transfer Protocol), an application layer protocol based on TCP/IP, have been applied to C/S systems. At present, Web technologies are the most widespread for C/S applications. The client side of the Web application is called the browser. Microsoft's Internet Explorer and Netscape's Navigator are the most widely used browsers.

Figure 4 shows the progression of the Web structure of C/S applications:

- At the initial stage, the main service of the Web system was distribution of static information such as a text or an image stored in the storage device of the server machine (the Web server). (Figure 4[*a*]).
- In the middle stage, in addition to distribution of information, more interactive services were provided to the user by applications added to the Web server. For example, in an online shopping service, monthly expenditure is calculated using inputs of the purchase amount of daily shopping (Figure 4[*b*]).
- In the recent stage, external systems such as legacy systems are placed on the back end of the Web server and more advanced services can be provided. An Internet banking service is realized by this structure (Figure 4[*c*]).

The advantage of a C/S system using the Web technologies is that users can unify various kinds of operations and the maintenance of the client software becomes easier. In the future, the proliferation of the Internet means that Web technologies will gain more popularity for realizing C/S systems.

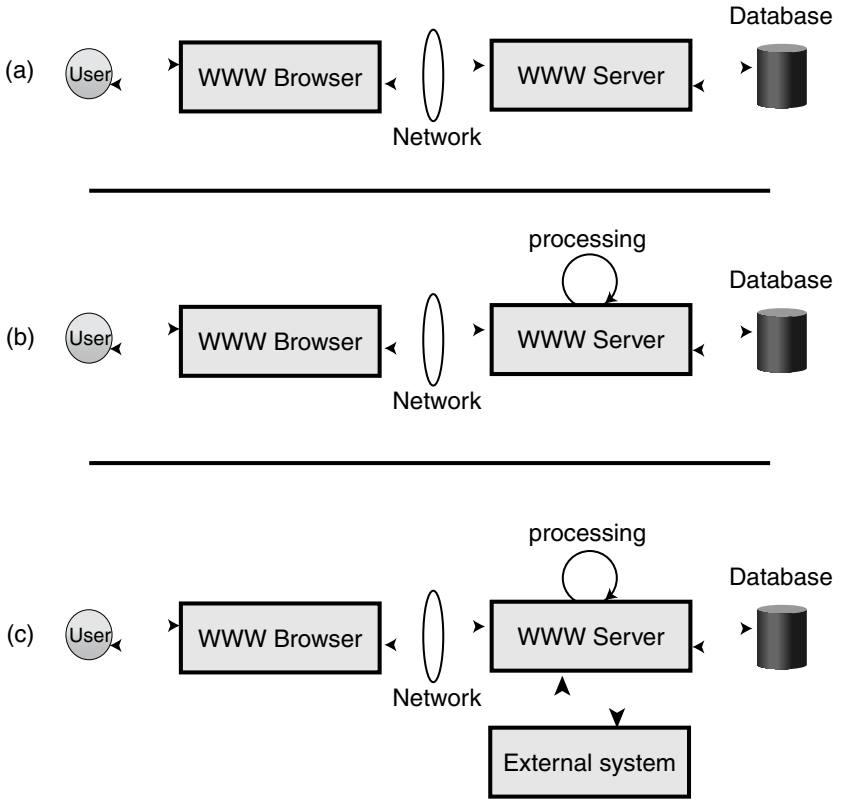

Figure 4 Progressive Stages of Web-Style C/S System. (*a*) WWW information-providing system. (*b*) WWW information-processing system. (*c*) WWW information-processing system with external systems processing.

1.3.2. Open System Technologies

A C/S system is one of distributed models of computing in the environment of heterogeneous computers. This is because open system technologies such as communication protocols and developing languages are required. For example, the adoption of the Java language as the developing language has increased, as has the adoption of CORBA (common object request broker architecture) as interconnection technology. Java is a new object-oriented programming language from Sun Microsystems and a portable operating system environment that enables the writing of portable components. CORBA is a standard architecture created by the OMG (Object Management Group) for a message broker called object request broker, a software component that enables communication among objects in a distributed environment. The common feature of Java and CORBA is platform independence. The most commonly used operating system on which C/S systems are developed is UNIX, which mainly works on workstations or Windows NT, which mainly runs on server-class personal computers. Java and CORBA are designed for running on both operating systems and cooperate mutually. In the future, technology like Java or CORBA that works in heterogeneous computing environments will become more and more important.

Due to the recent progress in hardware technologies, the processing capacity of a personal computer has become almost equal to that of a workstation. Therefore, there are too many choices among various technologies to allow appropriate system components to be selected. The system configuration has become more complex because of a combination of distributed components as well as heterogeneity.

2. FEATURES OF C/S SYSTEMS

2.1. Advantages

The advantages of C/S systems derive from their distributed processing structure, which C/S systems adopt, where processing is handled by two or more cooperating geographically distinct processors.

1. *Performance tuning:* In a distributed processing environment, the service requests from a client are not concentrated on a single server but can be distributed among several servers. Therefore, it becomes possible to distribute required workload among processing resources and improve the performance, such as response time to the client. Furthermore, adopting an efficient algorithm of request distribution such as an adaptive algorithm taking account of current loads of all servers makes it possible to improve the throughput (the amount of work processed per unit time) of the system.
2. *Availability improvement:* Clients are sensitive to performance of the system, but they do not perceive the physical structure of the distribution for requests. In other words, the group of networked servers seems to the client like one server. In a distributed processing environment, even if one server breaks, other servers can substitute for the broken server and continue to process its tasks. This can guarantee the high availability of services to the client compared to a host-centric processing environment.
3. *Scalability and cost-efficiency:* Scalability is the capacity of the system to perform more total work in the same elapsed time when its processing power is increased. With a distributed system, the processing capacity of the system can be scaled incrementally by adding new computers or network elements as the need arises, and excessive investment in a system can be avoided at the introduction stage of the C/S system. On the other hand, with a host-centric system, if the load level is going to saturate the system, the current system will be replaced by a more powerful computer. If further growth is planned, redundant computer capacity will need to be built into the current system to cope with future growth in requirements.

2.2. Disadvantages

1. *Strong dependence on the network infrastructure:* Because a distributed environment is largely based on a networking infrastructure, the performance and availability of the distributed system are strongly influenced by the state of the network. For example, if a network failure, such as of the router or transmission line, occurs, performance and availability for services may seriously deteriorate. The system designer and manager should design the fault-tolerant system and plan the disaster recovery taking account of the networking infrastructure.
2. *Security:* From the viewpoint of security, the possibility exists that confidential information stored on the database may leak out through the network. The system manager should take security measures such as authentication, access control, and cryptographic control according to the security level of information.
3. *Complexity of system configuration:* C/S systems are composed of several components from multiple vendors. Distributed applications often have more complex functionality than cen-

tralized applications, and they are built from diverse components. Multitier architectures, which will be explained in the next section, provide a nearly limitless set of choices for where to locate processing functions and data. In addition to choices about locations, there are also more hardware and software choices, and more data sharing occurs. These cause the difficulty in developing a C/S system and the complexity of managing it.

3. C/S SYSTEM ARCHITECTURES

3.1. Functional Elements of C/S Systems

Since vendors began releasing RDBMS (relational database management system) products in the 1970s, the processing model in which various business data are divided into distributed databases and are accessed via network has been widely adopted. Client/server systems are composed of the various functional elements associated with data processing.

The logical functions of C/S systems are roughly divided into three layers:

1. *Presentation logic:* This function contains all the logic needed to manage screen formats, the content of windows, and interaction with the user, that is, the user interface. In recent years, the use of graphical user interface (GUI) features such as buttons has become general.
2. *Application logic:* This is a generic name for application functions that do not belong to other layers. It includes business logic and the flow of control among application components.
3. *Data logic:* This contains all the logic relating to the storage and retrieval of data, and enforcing business rules about data consistency. Generally, databases are treated and are maintained by a DBMS (database management system).

In addition to these application-functional layers, some functions are needed to support interactions between clients and servers via networks. All the distributed software that supports interaction between application components and network software is called middleware. Middleware is a generic term for all the software components that allow us to connect separate layers or components and put them into a complete distributed system. It provides an application programming interface (API) that isolates application codes from the underlying network communication formats and protocols. It also supplies intermediate system services such as security, naming, directory, messaging, and transaction management services:

1. *Naming service:* In C/S systems, names of clients and servers must be unique within the range in which they are used. A naming service uniquely names a physical entity such as a client or server and associates logical structure such as a tree with physical entities.
2. *Directory service:* This is a directory service that provides a way for clients to locate servers and their services on the network and controls the address of messages and processing requests through a tree structure.
3. *Security service:* This service provides authentication, authorization, access control, and user account management.
4. *Messaging service:* This service provides the ability to send and receive messages between and among applications and users.
5. *Transaction service:* This service manages distributed transaction processing, such as consistency control, that maintains a database in a consistent state and enables recovery control from failures.

3.2. Two-Tier Architecture

Client/server architectures can be characterized by how the applications are distributed between the client and the server. In a two-tier architecture, the application logic is placed on either the client or the server undivided or split into two parts, which are placed on the client and the server respectively. In most cases, the presentation logic and the application logic are placed on the client side and the data logic is placed on the server side. Examples of two-tier C/S systems are file servers and database servers with stored procedures. Figure 5 shows a two-tier architecture.

The advantage of the two-tier architecture is ease of development. Because the two-tier architecture is simple, a developer can create applications quickly using a 4GL (fourth-generation language) development environment such as Microsoft's Visual Basic or Inprise's Delphi. However, as C/S systems grew up to run mission-critical or enterprise-wide applications, shortcomings of the two-tier architecture emerged:

1. *Performance:* Performance may deteriorate markedly as the number of clients, the size of the database, or the volume of transferred data increases. This deterioration is caused by a lack of capacity of resources such as the processing unit, the memory area, and the network bandwidth

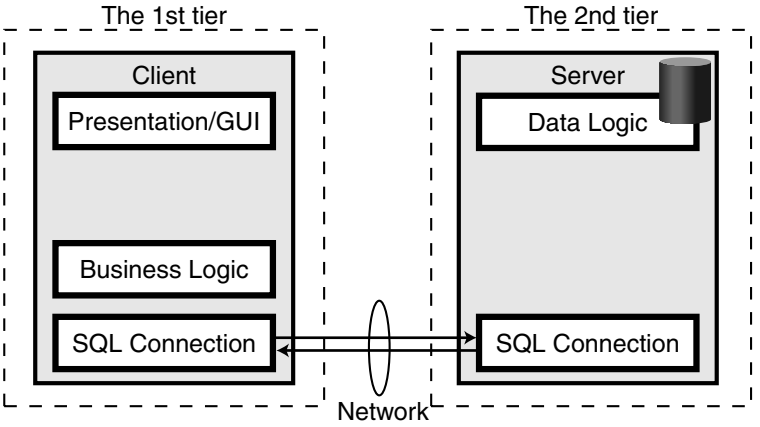

Figure 5 Two-Tier Architecture.

when the processing load is concentrated on the database server. Placing a part of application logic on the database server allows network traffic between the client and the database server to be reduced and the performance even improved. For example, a set of database access commands is compiled and saved in the database of the server. This scheme is called stored procedure. Some DBMS packages support this scheme. But there remains the problem that the application maintenance becomes complicated.

2. *Maintenance:* Because the business logic is placed on the client side in most two-tier C/S systems, version control of applications becomes complex. For example, in the case of an online shopping system, the business logic (or rule), such as tax rate, is implemented on the client side. Although this structure is effective in reducing network traffic between the client and the server, a system manager has to replace the application of all the clients every time the business rule or version of application is changed. This becomes a big problem in the total cost of ownership of two-tier C/S systems: costs of maintaining, operating, and managing large-scale systems accommodating many users and services.

3.3. Three-Tier Architecture

To solve some problems of the two-tier architecture and improve the reliability and performance of the system, a three-tier architecture has been adopted. In a three-tier architecture, the business logic is separated from the presentation and data layers and becomes the middle layer between them. The presentation logic resides in the first tier (the client side), and the data logic in the third tier (the server side), and the business logic in the second tier (the middle) between both tiers. This solves the problems occurring with the two-tier architecture. Figure 6 shows a three-tier architecture where the business logic and some connection functions are placed in the second tier.

In a three-tier architecture, the second tier plays the most important role. Many functions such as data access and connection with other systems are located in the second tier and can be used by any client. That is, the second tier becomes the gateway to other systems. The second tier is often called the application server.

The advantages of three-tier architecture derived from the application server are:

1. *Reduction of network traffic:* Concentrating the function of accessing database and other systems on the application server makes it possible to reduce network traffic between the client and the server. That is, instead of interacting with the database directly, the client calls the business logic on the application server, and then the business logic accesses the database on the database server on behalf of the client. Therefore, only service requests and responses are sent between the client and the server. From the viewpoint of network traffic, comparisons of two-tier architecture and three-tier architecture are shown in Figure 7.
2. *Scalability:* Adding or removing application servers or database servers allows the system to be scaled incrementally according to the number of clients and volume of requests.
3. *Performance:* Because workloads can be distributed across application servers and database servers, this architecture can prevent an application server from becoming a bottleneck and can keep the performance of the system in a permissible range.

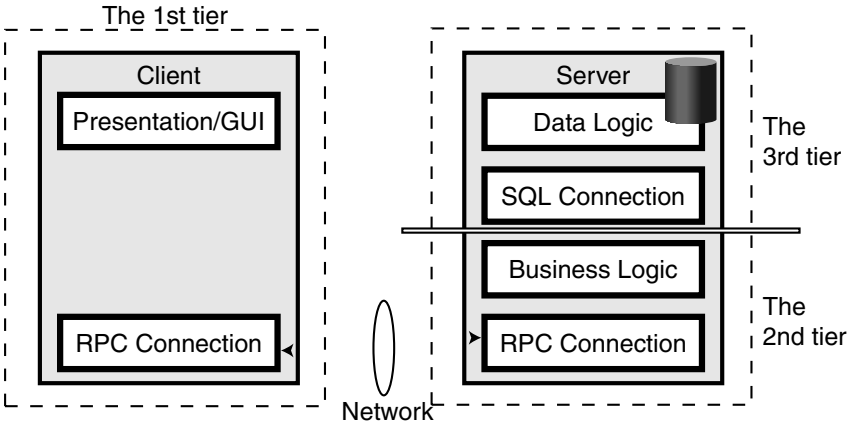

Figure 6 Three-Tier Architecture.

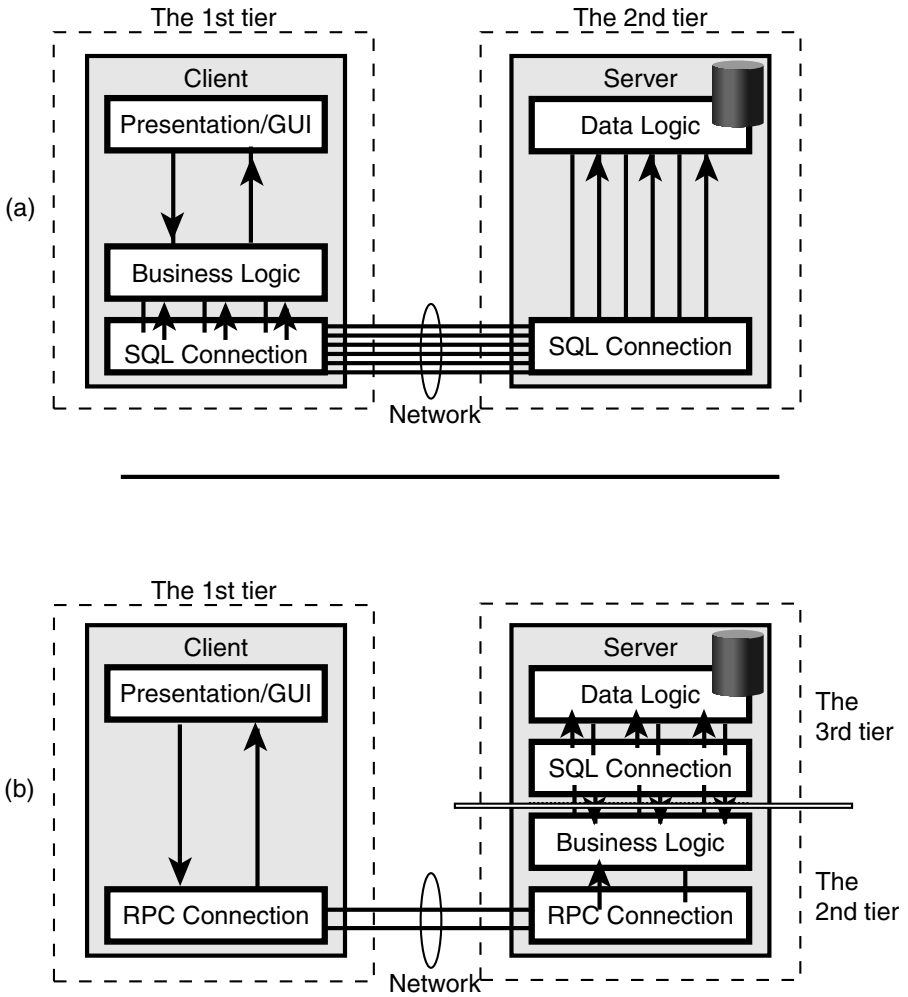

Figure 7 Network Traffic Comparison. (*a*) Two-tier architecture model. (*b*) Three-tier architecture model.

4. *Availability:* Even if an application server fails during processing the business logic, the client can restart the business logic on other application servers.

4. FUNDAMENTAL TECHNOLOGIES FOR C/S SYSTEMS

4.1. Communication Methods

The communication method facilitating interactions between a client and a server is the most important part of a C/S system. For easy development of the C/S systems, it is important for processes located on physically distributed computers to be seen as if they were placed in the same machine. Socket, remote procedure call, and CORBA are the main interfaces that realize such transparent communications between distributed processes.

4.1.1. Socket

The socket interface, developed as the communication port of Berkeley UNIX, is an API that enables communications between processes through networks like input/output access to a local file. Communications between two processes by using the socket interface are realized by the following steps (see Figure 8):

1. For two processes in different computers to communicate with each other, a communication port on each computer must be created beforehand by using ''socket'' system call.
2. Each socket is given a unique name to recognize the communication partner by using ''bind'' system call. The named socket is registered to the system.
3. At the server process, the socket is prepared for communication and it is shown that the server process is possible to accept communication by using ''listen'' system call.
4. The client process connects to the server process by using ''connect'' system call.

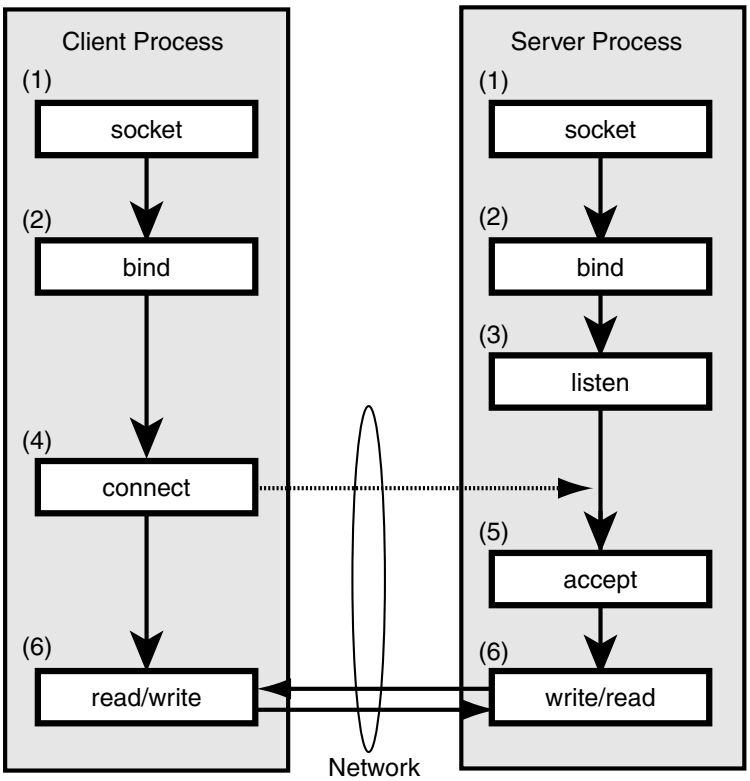

Figure 8 Interprocess Communication via Socket Interface.

5. The communication request from the client process is accepted by using ''accept'' system call in the server process. Then the connection between the client process and the server process is established.
6. Finally, each process begins communicating mutually by using ''read'' or ''write'' system calls.

4.1.2. Remote Procedure Call

Because the application programming interface (API) for socket programming uses system calls, the overhead associated with an application that communicates through the socket interface is rather small. However, because API is very primitive, socket programming depends on operating systems and the development of a system with a socket interface becomes complicated. For example, where programmers develop a system in which communications between processes on different platforms (such as Windows and UNIX) are required, they should master several versions of socket API supported by each platform.

One of the earliest approaches to facilitating easier use of sockets is realized by a remote procedure call (RPC). An RPC is a mechanism that lets a program call a procedure located on a remote server in the same fashion as a local one within the same program. It provides a function-oriented interface, and necessary preparation for communication is offered beforehand. An RPC is realized by a mechanism called a stub. The functions of stubs at client and server sides are to mimic the missing code, convert the procedure's parameters into messages suitable for transmission across the network (a process called marshaling) and unmarshal the parameters to a procedure, and dispatch incoming calls to the appropriate procedure.

Communications between two processes by using an RPC is realized by the following steps (see Figure 9):

1. The client program invokes a remote procedure function called the client stub.
2. The client stub packages (marshals) the procedure's parameters in several RPC messages and uses the runtime library to send them to the server.
3. At the server, the server stub unpacks (unmarshals) the parameters and invokes the requested procedure.
4. The procedure processes the request.
5. The results are sent back to the client through the stubs on both sides that perform the reverse processing.

The sequence from 1 to 5 is concealed from application programmers. One of the advantages of an RPC is that it hides the intricacies of the network and these procedures behave much the same as ordinary procedures to application programmers.

4.1.3. CORBA

An object is an entity that encapsulates data and provides one or more operations (methods) acting on those data; for example, ''the bank object'' has data from his client's account and operations by

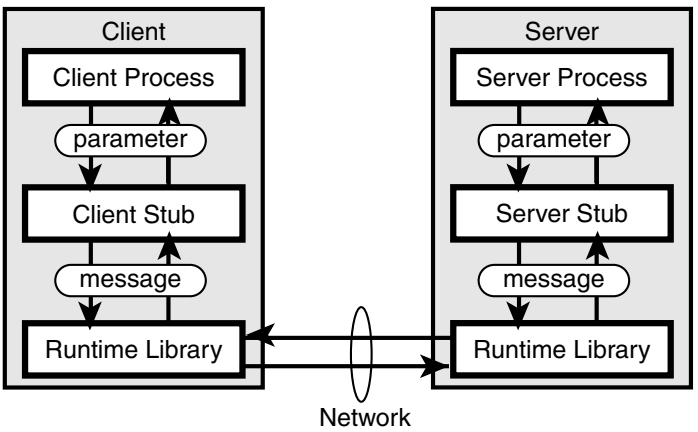

Figure 9 Interprocess Communication via an RPC.

which to manipulate them. Object-oriented computing is enabling faster software development by promoting software reusability, interoperability, and portability. In addition to enhancing the productivity of application developers, object frameworks are being used for data management, enterprise modeling, and system and network management. When objects are distributed, it is necessary to enable communications among distributed objects.

The Object Management Group, a consortium of object technology vendors founded in 1989, created a technology specification named CORBA (Common Object Request Broker Architecture). CORBA employs an abstraction similar to that of RPC, with a slight modification that simplifies programming and maintenance and increases extensibility of products.

The basic service provided by CORBA is delivery of requests from the client to the server and delivery of responses to the client. This service is realized by using a message broker for objects, called object request broker (ORB). An ORB is the central component of CORBA and handles distribution of messages between objects. Using an ORB, client objects can transparently make requests to (and receive responses from) server objects, which may be on the same computer or across a network.

An ORB consists of several logically distinct components, as shown in Figure 10.

The Interface Definition Language (IDL) is used to specify the interfaces of the ORB as well as services that objects make available to clients. The job of the IDL stub and skeleton is to hide the details of the underlying ORB from application programmers, making remote invocation look similar to local invocation. The dynamic invocation interface (DII) provides clients with an alternative to using IDL stubs when invoking an object. Because in general the stub routine is specific to a particular operation on a particular object, the client must know about the server object in detail. On the other hand, the DII allows the client to dynamically invoke a operation on a remote object. The object adapter provides an abstraction mechanism for removing the details of object implementation from the messaging substrate: generation and destruction of objects, activation and deactivation of objects, and invocation of objects through the IDL skeleton.

When a client invokes an operation on an object, the client must identify the target object. The ORB is responsible for locating the object, preparing it to receive the request, and passing the data needed for the request to the object. Object references are used for the ORB of the client side to identify the target object. Once the object has executed the operation identified by the request, if there is a reply needed, the ORB is responsible for returning the reply to the client.

The communication process between a client object and a server object via the ORB is shown in Figure 11.

1. A service request invoked by the client object is handled locally by the client stub. At this time, it looks to the client as if the stub were the actual target server object.
2. The stub and the ORB then cooperate to transmit the request to the remote server object.
3. At the server side, an instance of the skeleton instantiated and activated by the object adapter is waiting for the client's request. On receipt of the request from the ORB, the skeleton passes the request to the server object.
4. Then the server object executes the requested operations and creates a reply if necessary.
5. Finally, the reply is sent back to the client through the skeleton and the ORB that perform the reverse processing.

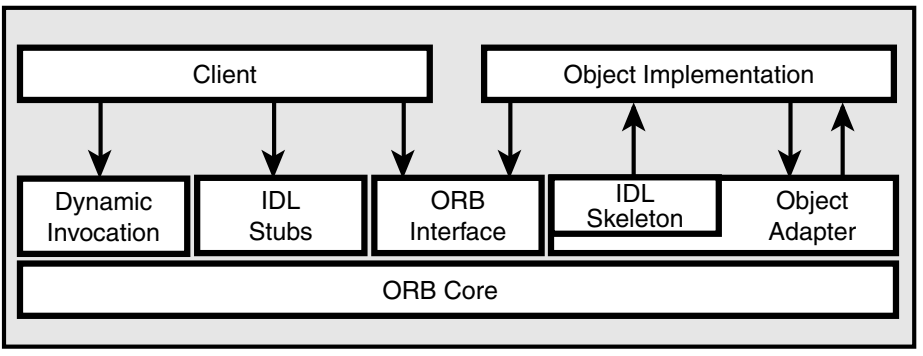

Figure 10 The CORBA Architecture.

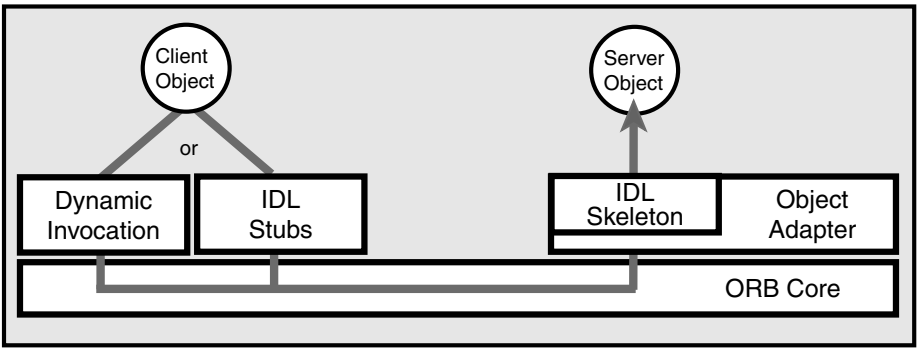

Figure 11 Communication between Objects in the CORBA Environment.

The main features of the CORBA are as follows:

1. *Platform independence:* Before the ORB transmits the message (request or result) from the client object or the server object into the network, the ORB translates the operation and its parameters into the common message format suitable for sending to the server. This is called marshaling. The inverse process of translating the data is called unmarshaling. This mechanism realizes communications between objects on different platforms. For example, a client object implemented on the Windows operating system can communicate with a server object implemented on the UNIX operating system.
2. *Language independence:* Figure 12 shows the development process of a CORBA C/S application. The first step of developing the CORBA object is to define the interface of the object (type of parameters and return values) by using the intermediate language IDL. Then the definition file described in IDL is translated into the file containing rough program codes in various development languages such as C, C++, Java, and COBOL.

IDL is not a programming language but a specification language. It provides language independence for programmers. The mapping of IDL to various development languages is defined by OMG. Therefore, a developer can choose the most appropriate one from various development languages to implement objects. Also, this language independence feature enables the system to interconnect with legacy systems developed by various languages in the past.

4.1.4. Other Communication Methods

In addition to Socket, RPC, and CORBA, there are several methods for realizing communication between processes.

- Java RMI (remote method invocation) is offered as a part of the JAVA language specification and will be a language-dependent method for the distributed object environment. Although there is the constraint that all the environments should be unified into Java, the developing process with it is a little easier than with CORBA or RPC because several functions for distributed computing are offered as utility objects.
- DCOM (distributed component object model) is the ORB that Microsoft promotes. A DCOM is implemented on the Windows operating system, and the development of applications can be practiced by using a Microsoft development environment such as Visual Basic.

4.2. Distributed Transaction Management

In a computer system, a transaction consists of an arbitrary sequence of operations. From a business point of view, a transaction is an action that involves change in the state of some recorded information related to the business. Transaction services are offered on both file systems and database systems.

The transaction must have the four properties referred to by the acronym ACID: atomicity, consistency, isolation, and durability:

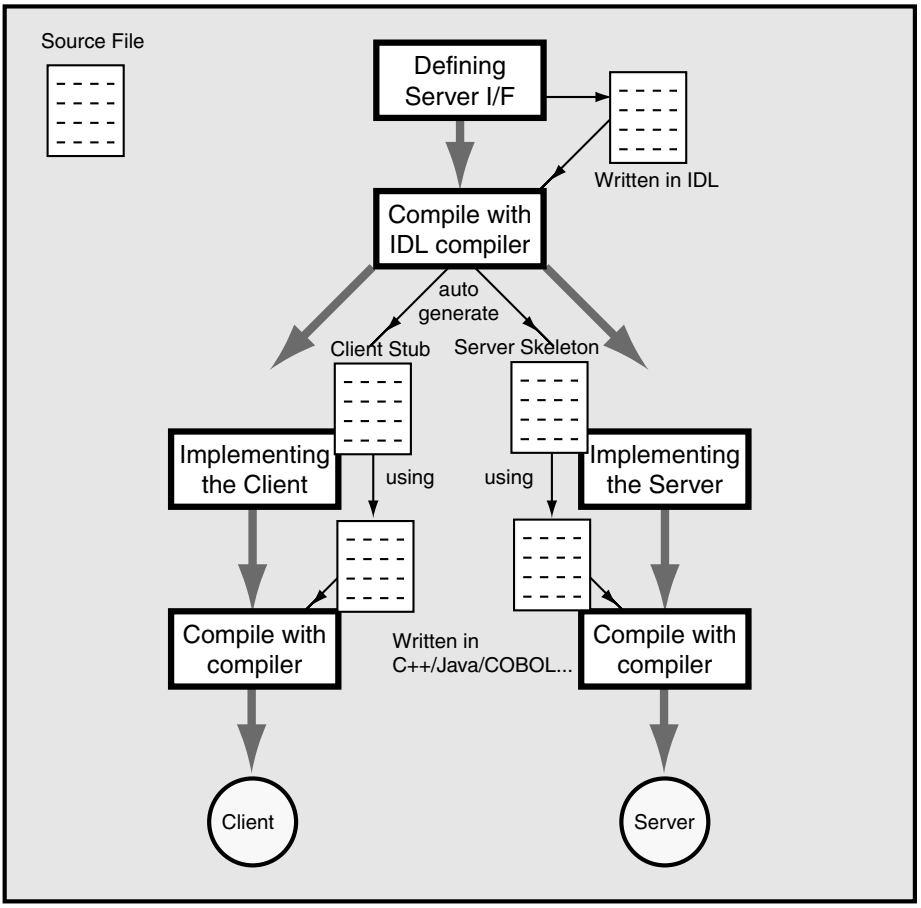

Figure 12 Development Process of a CORBA C/S Application.

- *Atomicity* means that all the operations of the transaction succeed or they all fail. If the client or the server fails during a transaction, the transaction must appear to have either completed successfully or failed completely.
- *Consistency* means that after a transaction is executed, it must leave the system in a correct state or it must abort, leaving the state as it was before the execution began.
- *Isolation* means that operations of a transaction are not affected by other transactions that are executed concurrently, as if the separate transactions had been executed one at a time. The transaction must serialize all accesses to shared resources and guarantee that concurrent programs will not affect each other's operations.
- *Durability* means that the effect of a transaction's execution is permanent after it completes commitments. Its changes should survive system failures.

The transaction processing should ensure that either all the operations of the transaction complete successfully (commit) or none of them commit. For example, consider a case in which a customer transfers money from an account A to another account B in a banking system. If the account A and the account B are registered in two separate databases at different sites, both the withdrawal from account A and the deposit in account B must commit together. If the database crashes while the updates are processing, then the system must be able to recover. In the recovery procedure, both the updates must be stopped or aborted and the state of both the databases must be restored to the state before the transaction began. This procedure is called rollback.

In a distributed transaction system, a distributed transaction is composed of several operations involving distributed resources. The management of distributed transactions is provided by a transaction processing monitor (TP monitor), an application that coordinates resources that are provided by other resources.

A TP monitor provides the following management functions:

- *Resource management:* it starts transactions, regulates their accesses to shared resources, monitors their execution, and balances their workloads.
- *Transaction management:* It guarantees the ACID properties to all operations that run under its protection.
- *Client/server communications management:* It provides communications between clients and servers and between servers in various ways, including conversations, request-response, RPC, queueing, and batch.

Available TP monitor products include IBM's CICS and IMS/TP and BEA's Tuxedo.

4.3. Distributed Data Management

A database contains data that may be shared between many users or user applications. Sometimes there may be demands for concurrent sharing of the data. For example, consider two simultaneous accesses to an inventory data in a multiple-transaction system by two users. No problem arises if each demand is to read a record, but difficulties occur if both users attempt to modify (write) the record at the same time. Figure 13 shows the inconsistency that arises when two updates on the same data are processed at nearly the same time.

The database management must be responsible for this concurrent usage and offer concurrency control to keep the data consistent. Concurrency control is achieved by serializing multiple transactions through use of some mechanism such as locking or timestamp.

Also, in case the client or the server fails during a transaction due to a database crash or a network failure, the transaction must be able to recover. Either the transaction must be reversed or else some previous version of the data must be available. That is, the transaction must be rolled back. ''All conditions are treated as transient and can be rolled back anytime'' is the fundamental policy of control in the data management.

In some C/S systems, distributed databases are adopted as database systems. A distributed database is a collection of data that belong logically to the same system but are spread over several sites of a network. The main advantage of distributed databases is that they allow access to remote data transparently while keeping most of the data local to the applications that actually use it.

The primary concern of transaction processing is to maintain the consistency of the distributed database. To ensure the consistency of data at remote sites, a two-phase commit protocol is sometimes used in a distributed database environment. The first phase of the protocol is the preparation phase, in which the coordinator site sends a message to each participant site to prepare for commitment. The second phase is the implementation phase, in which the coordinator site sends either an abort or a commit message to all the participant sites, depending on the responses from all the participant sites. Finally, all the participant sites respond by carrying out the action and sending an acknowledgement message. Because the commitment may fail in a certain site even if commitments are completed in other sites, in the first phase of the two phase commit protocol, temporary ''secure'' commitment that can be rolled back anytime is done at each site. After it is confirmed that all the sites succeeded, the ''official'' commitment is performed.

The two-phase commit protocol has some limitations. One is the performance overhead that is introduced by all the message exchanges. If the remote sites are distributed over a wide area network, the response time could suffer further. The two-phase commit is also very sensitive to the availability of all sites at the time of update, and even a single point of failure could jeopardize the entire transaction. Therefore, decisions should be based on the business needs and the trade-off between the cost of maintaining the data on a single site and the cost of the two-phase commit when data are distributed at remote sites.

5. CAPACITY PLANNING AND PERFORMANCE MANAGEMENT

5.1. Objectives of Capacity Planning

Client/server systems are made up of many hardware and software resources, including client workstations, server systems, and network elements. User requests share the use of these common resources. The shared use of these resources gives rise to contention that degrades the system behavior and worsens users' perception of performance.

In the example of the Internet banking service described in the Section 7, the response time, the time from when a user clicks the URL of a bank until the home page of the bank is displayed on

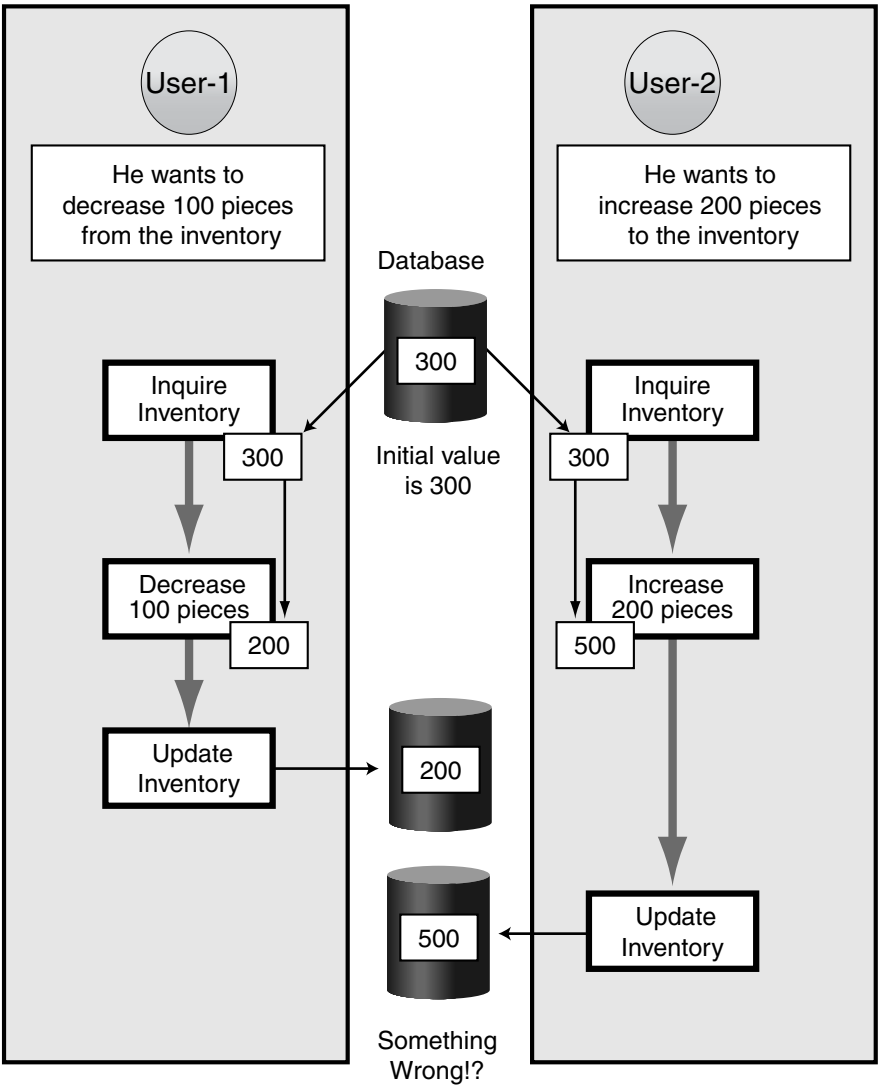

Figure 13 Inconsistency Arising from Simultaneous Updates.

the user's Web page, affects user's perception of performance and quality of service. In designing a system that treats requirements from multiple users, some service levels may be set; for example, average response time requirements for requests must not exceed 2 sec, or 95% of requests must exhibit a response time of less than 3 sec. For the given service requirements, service providers must design and maintain C/S systems that meet the desired service levels. Therefore, the performance of the C/S system should be evaluated as exactly as possible and kinds and sizes of hardware and software resources included in client systems, server systems, and networks should be determined. This is the goal of capacity planning.

The following are resources to be designed by capacity planning to ensure that the C/S system performance will meet the service levels:

- Types and numbers of processors and disks, the type of operating system, etc.
- Type of database server, access language, database management system, etc.
- Type of transaction-processing monitor

- Kind of LAN technology and bandwidth or transmission speed for clients and servers networking
- Kind of WAN and bandwidth when WAN is used between clients and servers

In the design of C/S systems, processing devices, storage devices, and various types of software are the building blocks for the whole system. Because capacity can be added to clients, servers, or network elements, addition of capacity can be local or remote. The ideal approach to capacity planning is to evaluate performance prior to installation. However, gathering the needed information prior to specifying the elements of the system can be a complicated matter.

5.2. Steps for Capacity Planning and Design

Capacity planning consists of the steps shown in Figure 14, which are essentially the same as those for capacity planning of general information systems.

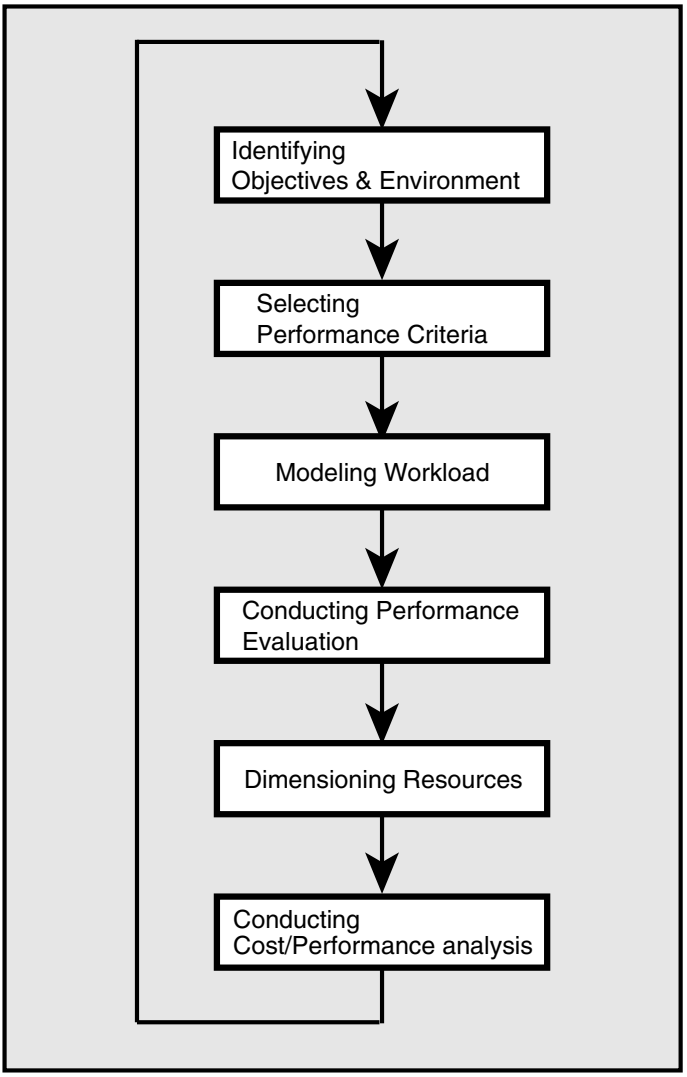

Figure 14 The Steps in Capacity Planning and Design.

Because new technology, altered business climate, increased workload, change in users, or demand for new applications affects the system performance, these steps should be repeated at regular intervals and whenever problems arise.

5.3. Performance Objectives and System Environment

Specific objectives should be quantified and recorded as service requirements. Performance objectives are essential to enable all aspects of performance management. To manage performance, we must set quantifiable, measurable performance objectives, then design with those objectives in mind, project to see whether we can meet them, monitor to see whether we are meeting them, and adjust system parameters to optimize performance. To set performance objectives, we must make a list of system services and expected effects.

We must learn what kind of hardware (clients and servers), software (OS, middleware, applications), network elements, and network protocols are presented in the environment. Environment also involves the identification of peak usage periods, management structures, and service-level agreements. To gather these information about the environment, we use various information-gathering techniques, including user group meetings, audits, questionnaires, help desk records, planning documents, and interviews.

5.4. Performance Criteria

1. *Response time* is the time required to process a single unit of work. In interactive applications, the response time is the interval between when a request is made and when the computer responds to that request. Figure 15 shows a response time example for a client/server database access application. As the application executes, a sequence of interactions is exchanged among the components of the system, each of which contributes in some way to the delay that the user experiences between initiating the request for service and viewing the DBMS's response to the query. In this example, the response time consists of several time components, such as (a) interaction with the client application, (b) conversion of the request into a data stream by an API, (c) transfer of the data stream from the client to the server by communication stacks, (d) translation of the request and invocation of a stored procedure by DBMS, (e) execution of SQL (a relational data-manipulation language) calls by the stored procedure, (f) conversion of the results into a data stream by DBMS, (g) transfer of the data stream from the server to the client by the API, (h) passing of the data stream to the application, and (i) display of the first result.
2. *Throughput* is a measure of the amount of work a component or a system performs as a whole or of the rate at which a particular workload is being processed.
3. *Resource utilization* normally means the level of use of a particular system component. It is defined by the ratio of what is used to what is available. Although unused capacity is a waste of resources, a high utilization value may indicate that bottlenecks in processing will occur in the near future.

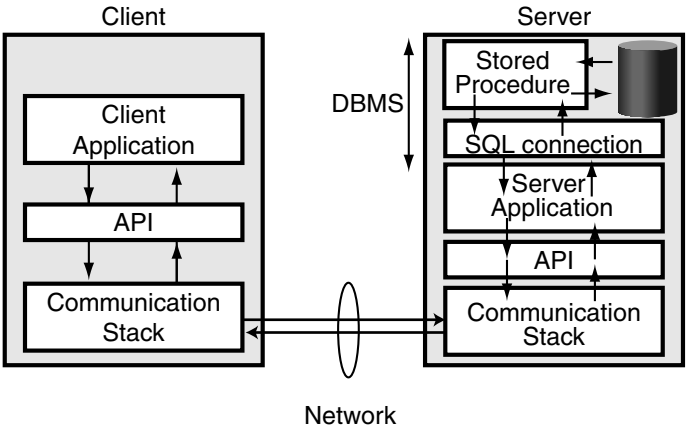

Figure 15 Response Time Example for a Database Access.

4. *Availability* is defined as the percentage of scheduled system time in which the computer is actually available to perform useful work. The stability of the system has an effect on performance. Unstable systems often have specific effects on the performance in addition to the other consequences of system failures.
5. *Cost–Performance ratio:* Once a system is designed and capacity is planned, a cost model can be developed. From the performance evaluation and the cost model, we can make an analysis regarding cost–performance trade-offs.

In addition to these criteria, resource queue length and resource waiting time are also used in designing some resources.

Response time and availability are both measures of the effectiveness of the system. An effective system is one that satisfies the expectations of users. Users are concerned with service effectiveness, which is measured by response time for transaction processing, elapsed time for batch processing, and query response time for query processing. On the other hand, a system manager is concerned with optimizing the efficiency of the system for all users. Both throughput and resource utilization are measures for ensuring the efficiency of system operations. If the quality of performance is measured in terms of throughput, it depends on the utilization levels of shared resources such as servers, network elements, and application software.

5.5. Workload Modeling

Workload modeling is the most crucial step in capacity planning. Misleading performance evaluation is possible if the workload is not properly modeled.

5.5.1. Workload Characterization

Workload refers to the resource demands and arrival intensity characteristics of the load brought to the system by the different types of transactions and requests. A workload consists of several components, such as C/S transactions, web access, and mail processing. Each workload component is further decomposed into basic components such as personnel transactions, sales transactions, and corporate training.

A real workload is one observed on a system being used for normal operations. It cannot be repeated and therefore is generally not suitable for use as a test workload in the design phase. Instead, a workload model whose characteristics are similar to those of real workload and can be applied repeatedly in a controlled manner, is developed and used for performance evaluations.

The measured quantities, service requests, or resource demands that are used to characterize the workload, are called workload parameters. Examples of workload parameters are transaction types, instruction types, packet sizes, source destinations of a packet, and page-reference patterns. The workload parameters can be divided into workload intensity and service demands. Workload intensity is the load placed on the system, indicated by the number of units of work contending for system resources. Examples include arrival rate or interarrival times of component (e.g., transaction or request), number of clients and think times, and number of processors or threads in execution simultaneously (e.g., file reference behavior, which describes the percentage of accesses made to each file in the disk system) The service demand is the total amount of service time required by each basic component at each resource. Examples include CPU time of transaction at the database server, total transmission time of replies from the database server in LAN, and total I/O time at the Web server for requests of images and video clips used in a Web-based learning system.

5.5.2. Workload Modeling Methodology

If there is a system or service similar to a newly planned system or service, workloads are modeled based on historical data of request statistics measured by data-collecting tools such as monitors.

A monitor is used to observe the performance of systems. Monitors collect performance statistics, analyze the data, and display results. Monitors are widely used for the following objectives:

1. To find the frequently used segments of the software and optimize their performance.
2. To measure the resource utilization and find the performance bottleneck.
3. To tune the system; the system parameters can be adjusted to improve the performance.
4. To characterize the workload; the results may be used for the capacity planning and for creating test workloads.
5. To find model parameters, validate models, and develop inputs for models.

Monitors are classified as software monitors, hardware monitors, firmware monitors, and hybrid monitors. Hybrid monitors combine software, hardware, or firmware.

If there is no system or service similar to a newly planned system or service, workload can be modeled by estimating the arrival process of requests and the distribution of service times or processing times for resources, which may be forecast from analysis of users' usage patterns and service requirements.

Common steps in a workload modeling process include:

1. Specification of a viewpoint from which the workload is analyzed (identification of the basic components of the workload of a system)
2. Selecting the set of workload parameters that captures the most relevant characteristics of the workload
3. Observing the system to obtain the raw performance data
4. Analyzing and reducing of the performance data
5. Constructing of a workload model.

The basic components that compose the workload must be identified. Transactions and requests are the most common.

5.6. Performance Evaluation

Performance models are used to evaluate the performance of a C/S system as a function of the system description and workload parameters. A performance model consists of system parameters, resource parameters, and workload parameters. Once workload models and system configurations have been obtained and a performance model has been built, the model must be examined to see how it can be used to answer the questions of interest about the system it is supposed to represent. This is the performance-evaluation process. Methods used in this process are explained below:

5.6.1. Analytical Models

Because generation of users' requests and service times vary stochastically, analytical modeling is normally done using queueing theory. Complex systems can be represented as networks of queues in which requests receive services from one or more groups of servers and each group of servers has its own queue. The various queues that represent a distributed C/S system are interconnected, giving rise to a network of queues, called a queueing network. Thus, the performance of C/S systems can be evaluated using queueing network models. Figure 16 shows an example of queueing network model for the Web server, where the client and the server are connected through a client side LAN, a WAN, and a server-side LAN.

5.6.2. Simulation

Simulation models are computer programs that mimic the behavior of a system as transactions flow through the various simulated resources. Simulation involves actually building a software model of

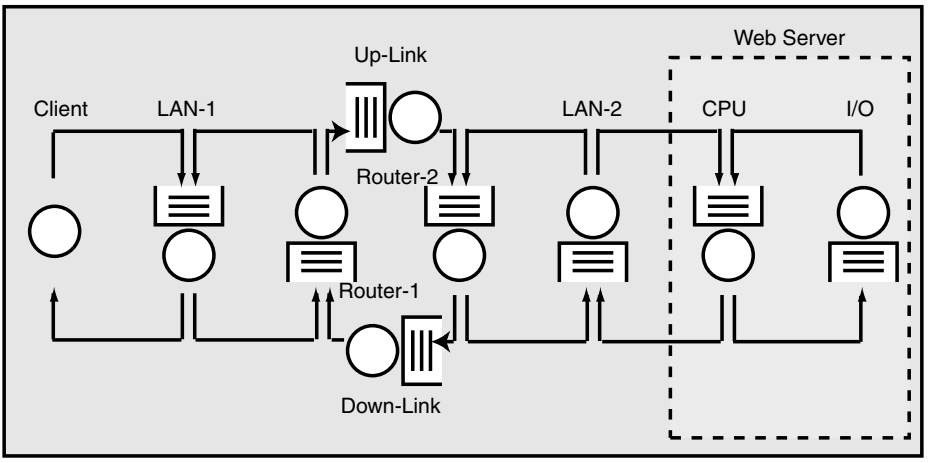

Figure 16 An Example of a Queueing Network Model for a Web Server.

each device, a model of the queue for that device, model processes that use the devices, and a model of the clock in the real world. Simulation can be accomplished by using either a general programming languages such as FORTRAN or C or a special-purpose language such as GPSS, SIMSCRIPT, or SLAM. For network analysis, there are special-purpose simulation packages such as COMNET III and BONeS are available.

5.6.3. Benchmarking

A benchmark is a controlled test to determine exactly how a specific application performs in a given system environment. While monitoring supplies a profile of performance over time for an application already deployed, either in a testing or a production environment, benchmarking produces a few controlled measurements designed to compare the performance of two or more implementation choices.

Some of the most popular benchmark programs and most widely published benchmark results come from groups of computer hardware and software vendors acting in consort. RISC workstation manufacturers sponsor the Systems Performance Evaluation Cooperative (SPEC), and DBMS vendors operate the transaction Processing Council (TPC). The TPC developed four system-level benchmarks that measure the entire system: TPC-A, B, C, and D. TPC-A and TPC-B are a standardization of the debit/credit benchmark. TPC-C is a standard for moderately complex online transaction-processing systems. TPC-D is used to evaluate price/performance of a given system executing decision support applications. The SPEC developed the standardized benchmark SPECweb, which measures a system's ability to act as a web server for static pages.

5.6.4. Comparing Analysis and Simulation

Analytic performance modeling, using queueing theory, is very flexible and complements traditional approaches to performance. It can be used early in the application development life cycle. The actual system, or a version of it, does not have to be built as it would be for a benchmark or prototype. This saves tremendously on the resources needed to build and to evaluate a design. On the other hand, a simulation shows the real world in slow motion. Simulation modeling tools allow us to observe the actual behavior of a complex system; if the system is unstable, we can see the transient phenomena of queues building up and going down repeatedly.

Many C/S systems are highly complex, so that valid mathematical models of them are themselves complex, precluding any possibility of an analytical solution. In this case, the model must be studied by means of simulation, numerically exercising the model for the inputs in question to see how they affect the output measures of performance.

6. MAINTENANCE AND ADMINISTRATION OF C/S SYSTEMS

Although system maintenance and administration is a complicated task even for a single centralized system, the complexity increases significantly due to the scalability, heterogeneity, security, distribution, naming, and so on in a C/S system. Efficient network and system-management tools are critical for reliable operation of distributed computing environments for C/S systems. In recent years, open and multivendor technologies have been adopted in construction of C/S systems. To administrate and maintain those systems totally, a standardized system management is needed for equipment from different vendors.

6.1. Architecture of System Management

6.1.1. OSI Management Framework

The OSI has defined five functional areas of management activities that are involved in distributed system management:

1. *Configuration management:* This involves collecting information on the system configuration and managing changes to the system configuration. Inventory control, installation, and version control of hardware and software are also included in this area.
2. *Fault management:* This involves identifying system faults as they occur, isolating the cause of the faults, and correcting them by contingency fallback, disaster recovery and so on.
3. *Security management:* This involves identifying locations of sensitive data and securing the system access points as appropriate to limit the potential for unauthorized intrusions. Encryption, password requirements, physical devices security, and security policy are also included in this area.
4. *Performance management:* This involves gathering data on the usage of system resources, analyzing these data, and acting on the performance prediction to maintain optimal system

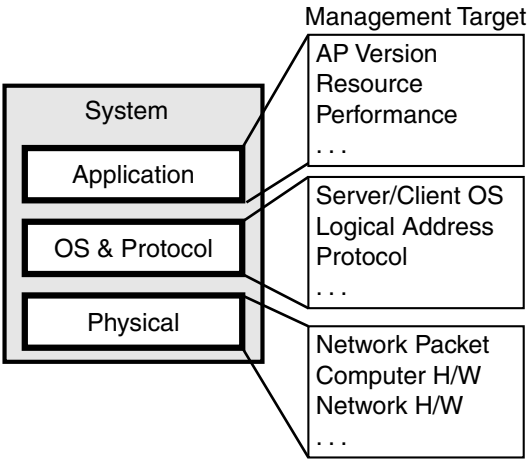

Figure 17 System Management Items.

performance. Real-time and historical statistical information about traffic volume, resource usage, and congestion are also included in this area.

5. *Accounting management:* This involves gathering data on resource utilization, setting usage shares, and generating charging and usage reports.

6.1.2. System Management Architecture

The items that are managed by system management can be classified into three layers: physical layer, operating system and network protocol layer, and application layer, as shown in Figure 17.

1. The physical layer includes client devices, server devices, and network elements, including LANs, WANs, computing platforms, and systems and applications software.
2. The operating system and network protocol layer includes the items for managing communication protocols to ensure interoperability between some units. In recent years, the adoption of TCP/IP standard protocol for realizing the Internet has been increasing. In a system based on the TCP/IP protocol stack, SNMP can be used for system management.
3. The application layer includes the items for managing system resources, such as CPU capacity and memory.

The system management architecture consists of the following components. These components may be either physical or logical, depending on the context in which they are used:

- A *network management station* (NMS) is a centralized workstation or computer that collects data from agents over a network, analyzes the data, and displays information in graphical form.
- A *managed object* is a logical representation of an element of hardware or software that the management system accesses for the purpose of monitor and control.
- An *agent* is a piece of software within or associated with a managed object that collects and stores information, responds to network management station requests, and generates incidental messages.
- A *manager* is software contained within a computer or workstation that controls the managed objects. It interacts with agents according to rules specified within the management protocol.
- A *management information base* (MIB) is a database containing information of use to network management, including information that reflects the configuration and behavior of nodes, and parameters that can be used to control its operation.

6.2. Network Management Protocol

An essential function in achieving the goal of network management is acquiring information about the network. A standardized set of network management protocols has been developed to help extract the necessary information from all network elements. There are two typical standardized protocols

for network management: simple network management protocol (SNMP), developed under Internet sponsorship, and common management information protocol (CMIP), from ISO (International Organization for Standardization) and ITU-T (International Telecommunication Union-Telecommunication Standardization Sector).

SNMP is designed to work with the TCP/IP protocol stack and establishes standards for collecting and for performing security, performance, fault, accounting, and configuration functions associated with network management. The communication protocol for the SNMP is UDP, which is a very simple, unacknowledged, connectionless protocol. CMIP is designed to support a richer set of network-management functions and work with all systems conforming to OSI standards. Both SNMP and CMIP use an object-oriented technique to describe information to be managed, where the software describing actions is encapsulated with the rest of agent code within the managed object. CMIP requires considerably more overhead to implement than SNMP.

Because SNMP is the most widely implemented protocol for network management today, an SNMP management system is described below. An SNMP management system consists of the following components, as shown in Figure 18:

1. An SNMP agent is a software entity that resides on a managed system or a target node, maintains the node information, and reports on its status to managers.
2. An SNMP manager is a software entity that performs management tasks by issuing management requests to agents.
3. An MIB is a database containing the node information. It is maintained by an agent.

SNMP is an asynchronous request/response protocol that supports the following operations (Version 2):

- Get: a request issued by a manager to read the value of a managed object
- GetNext: a request made by a manager to traverse an MIB tree

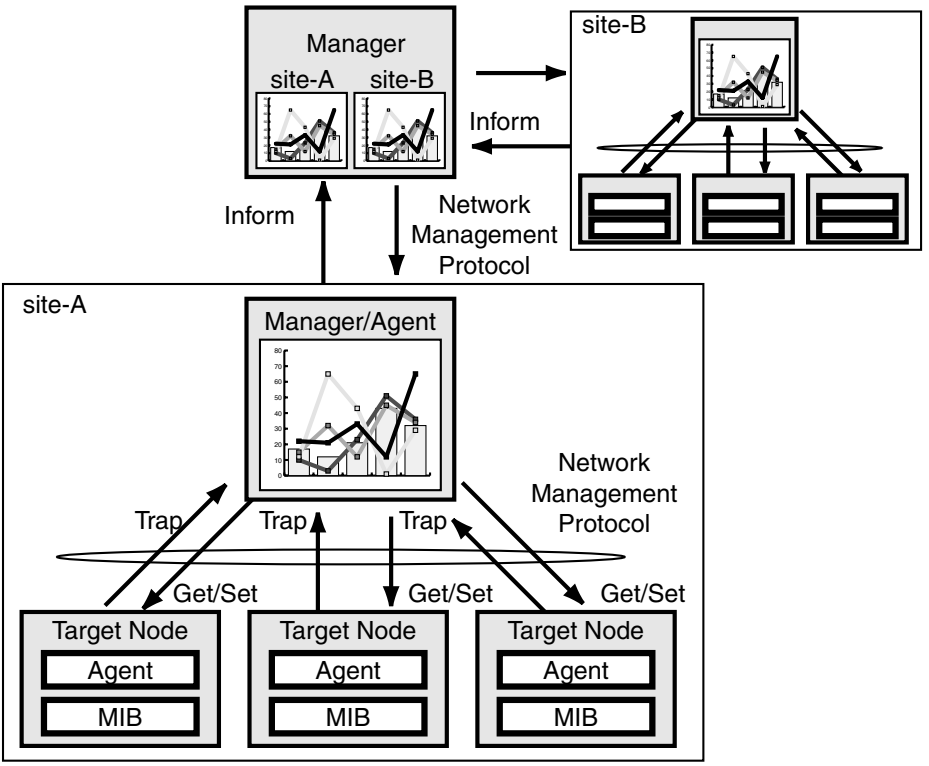

Figure 18 SNMP Management System.

- GetBulk: a command issued by a manager, by which an agent can return as many successor variables in the MIB tree as will fit in a message
- Set: a request issued by a manager to modify the value of a managed object
- Trap: a notification issued from an agent in the managed system to a manager that some unusual event has occurred
- Inform: a command sent by a manager to other managers, by which managers can exchange management information

In this case, the managed system is a node such as a workstation, personal computer, or router. HP's OpenView and Sun Microsystem's SunNet Manager are well-known commercial SNMP managers.

System management functions are easily decomposed into many separate functions or objects that can be distributed over the network. It is a natural idea to connect those objects using CORBA ORB for interprocess communications. CORBA provides a modern and natural protocol for representing managed objects, defining their services, and invoking their methods via an ORB. Tivoli Management Environment (TME) is a CORBA-based system-management framework that is rapidly being adopted across the distributed UNIX market.

6.3. Security Management

C/S systems introduce new security threats beyond those in traditional host-centric systems. In a C/S system, it is more difficult to define the perimeter, the boundary between what you are protecting and the outside world. From the viewpoint of distributed systems, the problems are compounded by the need to protect information during communication and by the need for the individual components to work together. The network between clients and servers is vulnerable to eavesdropping crackers, who can sniff the network to obtain user IDs and passwords, read confidential data, or modify information. In addition, getting all of the individual components (including human beings) of the system to work as a single unit requires some degree of trust.

To manage security in a C/S system, it is necessary to understand what threats or attacks the system is subject to. A threat is any circumstance or event with the potential to cause harm to a system. A system's security policy identifies the threats that are deemed to be important and dictates the measures to be taken to protect the system.

6.3.1. Threats

Threats can be categorized into four different types:

1. *Disclosure or information leakage:* Information is disclosed or revealed to an unauthorized person or process. This involves direct attacks such as eavesdropping or wiretapping or more subtle attacks such as traffic analysis.
2. *Integrity violation:* The consistency of data is compromised through any unauthorized change to information stored on a computer system or in transit between computer systems.
3. *Denial of service:* Legitimate access to information or computer resources is intentionally blocked as a result of malicious action taken by another user.
4. *Illegal use:* A resource is used by an unauthorized person or process or in an unauthorized way.

6.3.2. Security Services

In the computer communications context, the main security measures are known as security services. There are some generic security services that would apply to a C/S system:

- *Authentication:* This involves determining that a request originates with a particular person or process and that it is an authentic, nonmodified request.
- *Access control:* This is the ability to limit and control the access to information and network resources by or for the target system.
- *Confidentiality:* This ensures that the information in a computer system and transmitted information are accessible for reading only by authorized persons or processes
- *Data integrity:* This ensures that only authorized persons or processes are able to modify data in a computer system and transmitted information.
- *Nonrepudiation:* This ensures that neither the sender nor the receiver of a message is able to deny that the data exchange occurred.

6.3.3. Security Technologies

There are some security technologies fundamental to the implementation of those security services.

6.3.3.1. Cryptography Cryptographic systems or cryptosystems can be classified into two distinct types: symmetric (or secret-key) and public-key (or asymmetric) cryptosystems. In a symmetric cryptosystem, a single key and the same algorithm are used for both encryption and decryption. The most widely used symmetric cryptosystem is the Data Encryption Standard (DES), which is the U.S. standard for commercial use. In a public-key cryptosystem, instead of one key in a symmetric cryptosystem, two keys are employed to control the encryption and the decryption respectively. One of these keys can be made public and the other is kept secret. The best-known the public-key cryptosystem is RSA, developed by Rivest, Shamir, and Adleman at MIT (1978).

The major problem in using cryptography is that it is necessary to disseminate the encryption/decryption keys to all parties that need them and ensure that the key distribution mechanism is not easily compromised. In a public-key cryptosystem, the public key does not need to be protected, alleviating the problem of key distribution. However, a public key also needs to be distributed with authentication for protecting it from frauds. Public-key cryptosystems have some advantages in key distribution, but implementation results in very slow processing rates. For example, encryption by RSA is about 1000 times slower than by DES in hardware and about 100 times slower than DES in software. For these reasons, public-key cryptosystems are usually limited to use in key distribution and the digital signature, and symmetric cryptosystems are used to protect the actual data or plaintexts.

Data integrity and data origin authentication for a message can be provided by hash or message digest functions. Cryptographic hash functions involve, instead of using keys, mapping a potentially large message into a small fixed-length number. Hash functions are used in sealing or digital signature processes, so they must be truly one-way, that is, it must be computationally infeasible to construct an input message hashed to a given digest or to construct two messages hashed to the same digest. The most widely used hash function is message digest version 5 (MD5).

6.3.3.2. Authentication Protocol In the context of a C/S system, authentication is the most important of the security services because other security services depend on it in some way. When a client wishes to establish a secure channel between the client and a server, the client and the server will wish to identify each other by authentication. There are three common protocols for implementing authentication: three-way handshake authentication, trusted-third-party authentication, and public-key authentication. One of the trusted third-party protocols is Kerberos, a TCP/IP-based network authentication protocol developed as a part of the project Athena at MIT. Kerberos permits a client and a server to authenticate each other without any message going over the network in the clear. It also arranges for the secure exchange of session encryption keys between the client and the server. The trusted third-party is sometimes called an authentication server.

A simplified version of the third-party authentication in Kerberos is shown in Figure 19. Kerberos protocol assumes that the client and the server each share a secret key, respectively K_c and K_s, with the authentication server. In Figure 19, $[M]K$ denotes the encryption of message M with key K.

1. The client first sends a message to the authentication server that identifies both itself and the server.
2. The authentication server then generates a timestamp T, a lifetime L, and a new session key K and replies to the client with a two-part message. The first part, $[T, L, K, ID_s]K_c$, encrypts the three values T, L, and K, along with the server's identifier ID_s, using the key K_c. The second part, $[T, L, K, ID_c]K_s$, encrypts the three values T, L, and K, along with the client's identifier ID_c using the key K_s.
3. The client receives this message and decrypts only the first part. The client then transfers the second part to the server along with the encryption $[ID_c, T]K$ of ID_c and T using the session key K, which is decrypted from the first part.
4. On receipt of this message, the server decrypts the first part, $[T, L, K, ID_c]K_s$, originally encrypted by the authentication server using K_s, and in so doing recovers T, K, and ID_c. Then the server confirms that ID_c and T are consistent in the two halves of the message. If they are consistent, the server replies with a message $[T + 1]K$ that encrypts $T + 1$ using the session key K.
5. Now the client and the server can communicate with each other using the shared session key K.

6.3.3.3. Message Integrity Protocols There are two typical ways to ensure the integrity of a message. One uses a public-key cryptosystem such as RSA to produce a digital signature, and the other uses both a message digest such as MD5 and a public-key cryptosystem to produce a digital

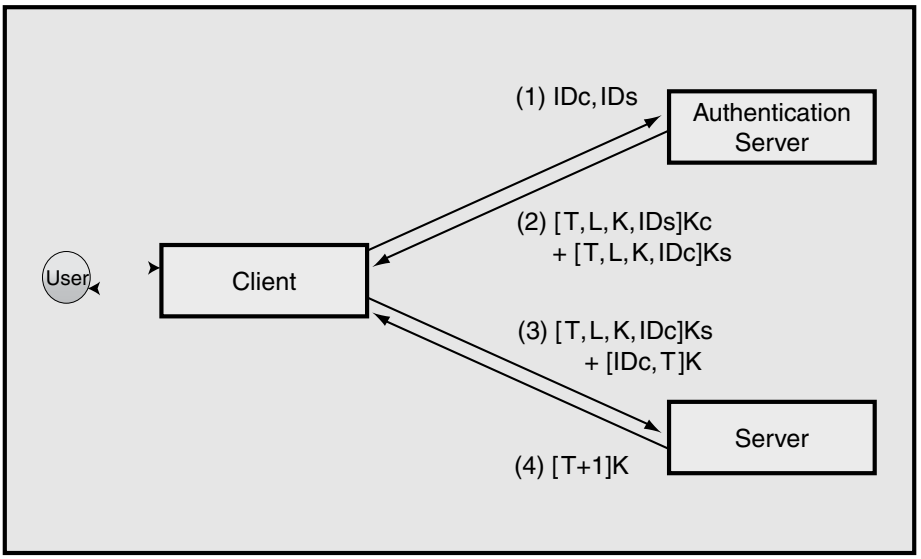

Figure 19 Third-Party Authentication in Kerberos.

signature. In the latter type, a hash function is used to generate a message digest from the message content requiring protection. The sender encrypts the message digest using the public-key cryptosystem in the authentication mode; the encryption key is the private key of the sender. The encrypted message digest is sent an appendix along with the plaintext message. The receiver decrypts the appendix using the corresponding decryption key (the public key of the sender) and compares it with the message digest that is computed from the received message by the same hash function. If the two are the same, then the receiver is assured that the sender knew the encryption key and that the message contents were not changed en route.

6.3.3.4. Access Control Access control contributes to achieving the security goals of confidentiality, integrity, and legitimate use. The general model for access control assumes a set of active entities, called subjects, that attempt to access members of a set of resources, called objects. The access-control model is based on the access control matrix, in which rows correspond to subjects (users) and columns correspond to objects (targets). Each matrix entry states the access actions (e.g., read, write, and execute) that the subject may perform on the object. The access control matrix is implemented by either:

- *Capability list:* a row-wise implementation, effectively a ticket that authorizes the holder (subject) to access specified objects with specified actions
- *Access control list* (ACL): a column-wise implementation, also an attribute of an object stating which subjects can invoke which actions on it

6.3.3.5. Web Security Protocols: SSL and S-HTTP As the Web became popular and commercial enterprises began to use the Internet, it became obvious that some security services such as integrity and authentication are necessary for transactions on the Web. There are two widely used protocols to solve this problem: secure socket layer (SSL) and secure HTTP (S-HTTP). SSL is a general-purpose protocol that sits between the application layer and the transport layer. The security services offered by the SSL are authentication of the server and the client and message confidentiality and integrity. The biggest advantage of the SSL is that it operates independently of application-layer protocols. HTTP can also operate on top of SSL, and it is then often denoted HTTPS. Transport Layer Security (TLS) is an Internet standard version of SSL and is now in the midst of the IETF standardization process. Secure HTTP is an application-layer protocol entirely compatible with HTTP and contains security extensions that provide client authentication, message confidentiality and integrity, and nonrepudiation of origin.

6.3.3.6. Firewall Because the Internet is so open, security is a critical factor in the establishment and acceptance of commercial applications on the Web. For example, customers using an Internet

banking service want to be assured that their communications with the bank are confidential and not tampered with, and both they and the bank must be able to verify each other's identity and to keep authentic records of their transactions. Especially, corporate networks connected to the Internet are liable to receive attacks from crackers of the external network. The prime technique used commercially to protect the corporate network from external attacks is the use of firewalls.

A firewall is a collection of filters and gateways that shields the internal trusted network within a locally managed security perimeter from external, untrustworthy networks (i.e., the Internet). A firewall is placed at the edge of an internal network and permits a restricted set of packets or types of communications through. Typically, there are two types of firewalls: packet filters and proxy gateways (also called application proxies).

- A packet filter functions by examining the header of each packet as it arrives for forwarding to another network. It then applies a series of rules against the header information to determine whether the packet should be blocked or forwarded in its intended direction.
- A proxy gateway is a process that is placed between a client process and a server process. All incoming packet from the client is funneled to the appropriate proxy gateway for mail, FTP, HTTP, and so on. The proxy then passes the incoming packets to the internal network if the access right of the client is verified.

7. A PRACTICAL EXAMPLE: INTERNET BANKING SYSTEM

Here we will explain an Internet banking system as a practical example of a C/S system. An Internet banking service is an online financial service that is offered on the Internet. The system offers various financial services, such as inquiry for bank balance, inquiry for payment details, and fund transfer to customers via the Internet.

We will explain the flow of transactions in C/S processing through an example of fund transfer service between a customer's account in bank A and another account in bank B. Figure 20 shows the flow of processing for the example.

Step 1: A customer connects to the Internet from a desktop personal computer in his home or office and accesses the site of bank A, which offers the Internet banking service, by using a Web browser.

Step 2: At the top menu of the Web page, the user is asked to enter his or her user ID and password already registered. According to the indication on the Web page, the user inputs user ID and password into the input field on the Web page.

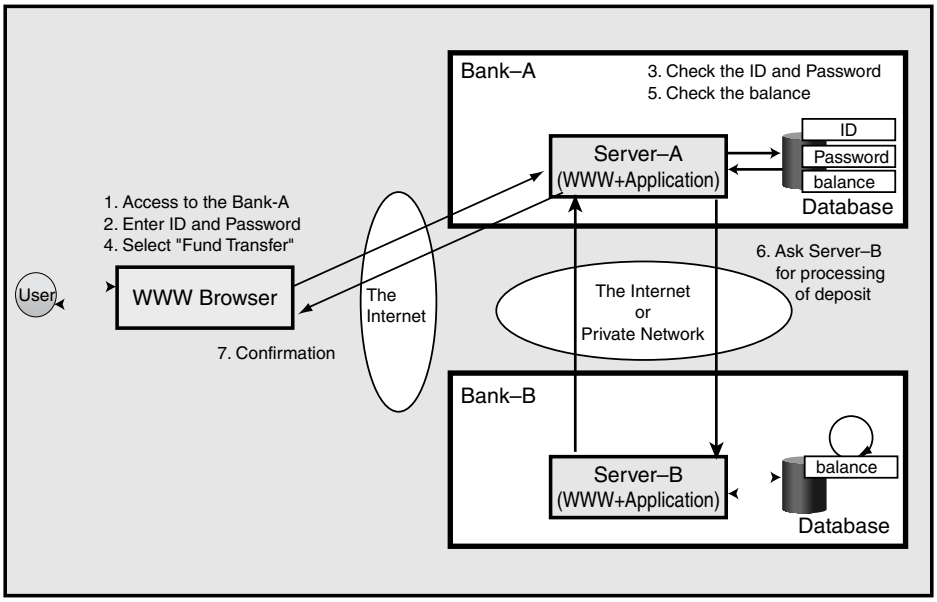

Figure 20 Internet Banking System.

Step 3: The entered user ID and password are sent back to the server via the Internet by the secured communication protocol HTTPS. The server receives that information and checks information on the database that accumulates information about the client and other various data. If the user ID and password correspond to those already registered, the customer is permitted to use the service.

Step 4: The customer selects ''fund transfer'' from the menu displayed on the Web page.

Step 5: The server receives the above request, searches the database, and retrieves the customer's account information.

Step 6: After confirming that there is enough money for payment in the customer's account, the server asks another server in bank B for processing of deposit. In this case, the server of bank A becomes ''client'' and the server of bank B becomes ''server.''

Step 7: After the server of bank A confirms that the processing was completed normally in the server of bank B, it updates (withdraws) the account information of the customer in the database. The acknowledgement that the request for fund transfer is successfully completed is also displayed on the customer's Web page.

ADDITIONAL READING

Crowcroft, J., *Open Distributed Systems,* UCL Press, London, 1996.

Davis, T., Ed., *Securing Client/Server Computer Networks*, McGraw-Hill, New York, 1996.

Edwards, J., *3-Tier Client/Server at Work,* Rev. Ed., John Wiley & Sons, New York, 1999.

Menascé, D. A., Almeida, V. A. F., and Dowdy, L. W., *Capacity Planning and Performance Modeling: From Mainframes to Client–Server Systems,* Prentice Hall, Englewood Cliffs, NJ, 1994.

Orfali, R., Harkey, D., and Edwards, J., *Client/Server Survival Guide,* 3d Ed., John Wiley & Sons, New York, 1999.

Renaud, P. E., *Introduction to Client/Server Systems,* John Wiley & Sons, New York, 1993.

Vaughn, L. T., *Client/Server System Design and Implementation,* McGraw-Hill, New York, 1994.