Client-Server Computing in a Large Network of PCs

by Frederick L. Hiltz

Client-server architecture and the network of personal computers are a handsome couple. They complement each other, forming a strong foundation for the family of application, system, and maintenance programs and databases that establishes an information system. After discussing why they combine so well, I present examples from the information system at Brigham and Women's Hospital, as seen by programmers and system managers.

Client-Server Model

The client-server model for distributing an information system across many processes or many computers may be compared with two other popular models: peer-to-peer networks and the mainframe model. A client is the active partner, making requests and receiving responses from a server that remains passive, working only as requested. Less restrictive is the peer-to-peer network, where any process may potentially use any facility (device, data, program) on the network. The mainframe model, even less restrictive, gives all processes a single large environment containing all facilities. Of course the mainframe model does not require a mainframe computer; it often appears on one or a cluster of minicomputers.

In return for its architectural restriction, the client-server model on a network of PCs offers benefits for configuration, management, and programming.

Configuration—Clients are typically PC workstations, "seats" where the user enjoys a modern interface and substantial computing power dedicated to her own task, at little more cost than the terminals used in other models. Each new client on the network adds but a small load to its servers, and thus little impact on other users. When necessary, a new server is quickly and easily added, with no downtime and no effect on the users except improved performance. Completely new services bring their own servers with them. In comparison, peer-to-peer networks are easy to expand but difficult to coordinate. The mainframe model, while relatively easy to coordinate, is expensive and disruptive to expand.

Management—Because client workstations need not store data or programs, the system manager may install or replace them as easily as terminals. Servers may reside at central sites where a professional staff handles security, backup, maintenance, and reconfiguration without the users' concern. Failure of a client affects only one user until his workstation is replaced. A large PC network contains many servers; failure of one affects only those using its services until the staff corrects the problem at a central site. By comparison, a peer-to-peer network of more than a few computers presents a nontrivial management task—just ask an Internet-host administrator. Mainframe systems, on the other hand, offer superior security and management tools, although their failures often affect a large number of users.

Programming—The programmer in a client-server environment works with a simple model that enforces the clean interfaces between components so often advocated and so seldom implemented elsewhere. When designing a server, she abstracts the service into a set of client requests and server responses. She publishes their form and the means for transmitting them (for example, a message to an object or a remote procedure call), and she writes the server to return exactly one response to each request. The program deals with one thing at a time, has no side effects, and retains little if any information from one request to another. Therefore, it is easy to write and to maintain.

The designer of a client program takes advantage of the published services that are useful to the application. His program invokes services by the request-response model without concern for how the server generates responses. If he needs a new service, he does not add logic to the client, but writes a new server.

An ideal division of an application between client and server would place the user-interface functions in the client and the database access functions in the server so that each can operate most efficiently. In practice, the division of tasks is one of many application-design decisions that take into consideration performance, costs, available languages, and preexisting software components.

The client-server architecture applies at all levels: system, network, and application. In M systems the most common servers are the vendors' database global and lock managers, written to answer requests from clients running their products on the local computer or a remote computer. In mixed-vendor

networks, the industry standard DDP protocol and the ANSI standard Open MUMPS Interconnect protocol—both client-server—handle globals and locks. Among applications, background batch servers like the Department of Veterans Affairs' DHCP (Decentralized Hospital Computer Program) Task Manager are common, and the concept can readily be extended, as the following examples demonstrate.

Brigham and Women's Hospital

Brigham and Women's Hospital (BWH) is a 720-bed teaching affiliate of the Harvard Medical School. In late 1994, it is completing the replacement of 14 minicomputers and 1,700 terminals, organized on the mainframe model, with a large network of PCs to serve the medical and administrative needs of the entire enterprise.

The New Platform—Called TNP by its friends, the platform supports 4,100 PCs on a Token Ring network in a client-server architecture.[1,2,3] Client workstations run DOS on 486-class PCs with no disk. (Some run Windows, which requires a disk, but the disk stores no data or application programs.) A Novell network operating system provides initial program loading, file servers, and DOS applications. The database servers, 36 large PCs, maintain 43 gigabytes of M programs and globals. Specialized servers, some described later, have been installed to provide new services.

This platform gives the hospital a large boost in capacity, computing power, and ease of use, offering a fast full-screen color user interface at very reasonable cost. In current dollars, its capital cost for the entire system is $3,500 per seat, compared with $3,239 for the system it replaces.[4] Information systems operating costs are less than 1 percent of the BWH budget.[5]

The PCs where users work are pure clients, loading programs from servers. As the programs run, they access the database and all other components of the system via requests to servers. At a fundamental level, these workstations have only two "I/O ports": the user interface and a network connection. Because they have no local storage, the workstations are deployed as easily as terminals and the hospital need not manage thousands of copies of programs.

Servers, located in the computer room, are maintained, backed up, and secured by professional operations staff. Few of the processes running on these servers stand alone; most double as clients, sometimes requesting services from other servers.

Speedy response from TNP depends on extensive buffering of data and M routines in caches at the servers, routers, and clients. The many copies of data in the caches are not always identical, although each presents a consistent view of the database to its client.[6] The LOCK command, and someday, transaction processing, requires the copies to be made identical; consequently these are time-consuming operations.

TNP is not designed to run nonstop, but to be tolerant of normal operational procedures and of failures. Daily, the backup program copies shadow data sets to tape without interrupting service. Failure of a workstation affects only its user until a replacement is installed. Servers fail too: an option on the system operator's menu changes the shadow PC into a replacement server; meanwhile the screens on its client workstations ask the user to wait (4 to 6 minutes) until the application resumes with no loss of data. TNP tolerates most network failures. Token Ring survives one break in the ring, sending packets "the other way around." There are redundant routers as a secondary path between clients and servers when routers and some wiring hubs fail.

Client-Server Communication—Some applications, once loaded and running on the client, communicate via their own specialized protocols to their servers:

- The popular *Scientific American* Medicine program supplies medical text and images from dedicated CD-ROM servers to DOS client viewing programs.
- M programmers see the 36 servers and 43 gigabytes of the M database as one big computer, using M's extended reference programming syntax to partition the big computer according to application. Every reference to a global or routine is handled by the DTM language processor through a complex protocol that passes the reference through routers to the server where the global or routine resides, while managing several data buffers along the way.[7] DTM keeps track of the locations of globals and routines invisibly to the application programmer. Most routines are stored on the same server as their principal data files, but many applications refer to data from several servers, and large or busy databases are often partitioned among many servers, all transparent to the M program, its programmer, and its users.
- During conversion from the minicomputers to TNP, certain activities require a connection between them, using a protocol written in C on the "gateway" servers and in M on the clients. To connect the workstation as a programming terminal to the minicomputers, the M program simply says `do^%Gatecall`, which implements the protocol. Other entries to `%Gatecall` support a great deal of data transmission, accomplished on servers dedicated to maintaining identical copies of selected globals on both systems.
- System managers maintain the TNP users' identifications and privileges through M routines that implement Novell's

protocol and programming interface to their network authorization server. A pleasant consequence is just one password to remember for all network functions.

- Visual Basic processes on Windows workstations manage the user interface but call on programs written in M for database services at a higher level than global references and locks, for example to obtain a doctor's patient list. The Visual Basic program makes a remote procedure call to an M function, which returns the result via a BWH protocol.

The second mechanism for client-server communication is the M global itself, often organized as a double-ended queue, where clients place requests at the front and servers remove requests from the back:

- The classical background batch processors work this way, running on dedicated "compute servers." Because they don't slow down users, they can and often do run during the day. Background processes do their printing as clients to Novell's network print servers.
- An expert system evaluates selected clinical events and alerts physicians about life-threatening ones. The evaluation program runs as a subroutine for immediate feedback while physicians are writing orders at a workstation.[8] The same program runs in a background process, evaluating laboratory results placed in the queue as the technicians enter them.
- An M program can start another M process on any computer in the network through an extrinsic function that mimics the JOB command but puts the information in a queue. The target computer's caretaker program takes the information from the queue and performs the actual JOB command.
- M programs use the hospital's radio paging system to page someone or to change a pager's status by calling standard M extrinsic functions that do not do the work themselves, but place requests on a queue for a dedicated server, connected to the radio page controller, to perform.

Watchdog functions monitor dozens of server computers and hundreds of server processes. Without them the network would be unmanageable. The watchdog, designed and programmed along with the server, typically monitors some of the server's globals and locks for "liveness" and tries to repair or restart failed server processes. If unable, the watchdog alerts the operations staff by a request (yes, on another queue) to the server that maintains the central network monitor.

In summary, the client-server architecture has proven to be an effective and productive model in a large network of small computers. Powerful workstations are managed as easily as dumb terminals, and server computers are installed at small cost and no fanfare when needed to distribute the computing load or for new services. Programmers enjoy a coherent environment in which to create the hospital's computing services.

Frederick L. Hiltz, Ph.D., is a principal software designer at the Center for Applied Medical Information Systems Research of Brigham and Women's Hospital, where he applies the enterprise network to hospital information requirements.

Endnotes

1. P. Roberts, "Client-Server at Brigham and Women's Hospital: An Enterprise of PCs," in R Khanna, ed., *Integration of Personal Computers into Enterprise-Wide Client/Server Computing Environments* (Englewood Cliffs, New Jersey: Prentice Hall, 1994).
2. P.A. McManus et al., "Distributed Design for a Large-Scale Workstation Network," *MUG Quarterly* 19:4 (April 1990).
3. J. Glaser et al., "A Very Large PC LAN as the Basis for a Hospital Information System," *Journal of Medical Systems* 15:2 (1991).
4. Roberts.
5. Brigham and Women's Hospital 1993 Annual Report, 24–25.
6. P.D. Beaman and J.J. Althouse, "An Efficient MUMPS Distributed Database Using a High Level LAN Interface," *MUG Quarterly* 19:3 (1989).
7. Ibid.
8. J.M. Teich et al., "Design of an Easy-to-Use Physician Order Entry System with Support for Nursing and Ancillary Departments," Proceedings of 16th SCAMC (New York: McGraw-Hill, 1992): 99–103.