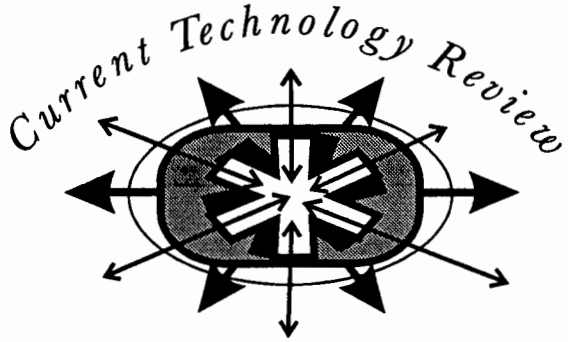


Client-Server Computing

TIME-SHARED COMPUTING

Two decades ago, the data processing industry was using powerful mainframes with the users sharing CPU cycles and data storage facilities. Access to the mainframes was tightly controlled by MIS departments and the only method of accessing the data from mainframes was through punched cards or primitive terminals. We saw evolution of pioneering data retrieval and analysis methodologies in the years to come; yet the techniques were arduous and constrained due to centralized resources and user-hostile interfaces. Thus, when PCs came to the computing industry in the early 1980s, their growth was phenomenal, as they provided limited CPU cycles and data storage facilities on user desktops, user control over desktops, and user-friendly PC-based software.

Alok Sinha

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

PCs, however, were still incapable of handling data processing needs of most large businesses. Local area networks (LANs) provided the solution by connecting PCs and mainframes. Thus, the PC where present, was used as a user-friendly "terminal" to the host. At the same time, we saw the evolution of LAN and PC-based file and print Servers (e.g., Novell Netware started in 1983 and Microsoft MS-Net-based LAN software, such as IBM PC-LAN started in the 1984–85 year). In 1989, in a survey by Infonetics of Santa Clara, 20% of PC LAN buyers cited the desire to share printers, while 22% cited the sharing of large mass storage devices as buying motivation [6]. As we will discuss in the next section, file and printer sharing does not lead to Client-Server applications.

Today, both PCs and LANs have a large installed base in corporations[1]; so have PC-based word-processors, spreadsheets, and database programs. Thus, while the PC-based software has allowed a variety of processing to be done on the desktop, PC LANs have allowed sharing of files and peripherals to take place at the department level in corporations.

We have also seen Unix workstations getting larger installed bases in corporations. These are typically used for "specialized applications." Worldwide Unix workstation revenue from "the commercial market" is expected to rise from $0.8 billion in 1990 to $4.6 billion in 1994, while the revenue from "the technical market" is expected to grow from $6.6 billion in 1990 to $10.7 billion in 1994 [20].

Most of the "Mission critical applications" in corporations, however, have remained mainframe-(and minicomputer-) centric. "Mission critical applications" can be loosely defined to be those information processing and analysis applications whose output is used by corporations for strategic decisions. Almost universally, finance management systems are considered "mission critical applications." The "criticality" of one set of applications over another is often determined by the business of the corporation (e.g., transaction processing systems are often the most "critical" applications for the airline companies).

Indeed, today the Network industry in general and the LAN and Database Server Industry in particular is very excited about "downsizing" mainframe applications to PC Servers and LANs using the Client-Server Computing paradigm. Their enthusiasm is causing MIS shops to take more serious notice of this new paradigm. Of course, this has resulted in the development of new network operating systems and database architectures to support the Client-Server Computing model.[2]

Client-Server Computing Paradigm

Definitions

In the Client-Server computing paradigm, one or more Clients and one or more Servers, along with the underlying operating system and interprocess communication systems, form a composite system allowing distributed computation, analysis, and presentation. We will call such a composite system a "Client-Server System" or simply CSS. In such a system, a Client is a process which interacts with the user and has the following characteristics:

[A] It presents the User Interface (UI). This interface is the sole means of garnering user queries or directions for purposes of data retrieval and analysis, as well as the means of presenting the results of one or more queries or commands. Typically, the Client presents a Graphical User Interface (GUI) to the user (e.g., Microsoft Windows-based interfaces).

As a CSS can consist of multiple Clients, multiple User Interfaces may exist in a CSS, but each Client will have a single consistent UI, e.g., a CSS may have both Microsoft Windows or Presentation Manager (PM) Clients. A CSS may have interfaces in addition to the User Interface for administrative control and system management.

[B] It forms one or more queries or commands in a predefined language for presentation to the Server. The Client and the Server may use a standard-based language such as SQL or a proprietary language known within the CSS. Each user query or command need not necessarily map a query to the Server from the Client.

A Client may use caching and optimization techniques to reduce queries to the Server or perform security and access control checks. A Client may also check integrity of queries or commands requested by the user. Sometimes it may not be necessary to send a query to the Server at all. In such cases, Client may itself perform the data processing requested by the user and satisfy the user query or command. It is however, recommended that Client applications do not provide these functionalities and we ask that they foist these on the Server application. For example, if the Server manages security, it is much more difficult for intruders to break in.

[C] It communicates to the Server via a given Interprocess communication methodology and transmits the queries or commands to the Server. An ideal Client completely hides the underlying communication methodology from the user.

[D] It performs data analysis on the query or command results sent from the Server and subsequently

[1]PC-installed bases in U.S. government, businesses, and education were estimated to be 33.4 million units in 1990 and are expected to grow to 69 million units by 1994. 34.2% of PC-installed bases in 1990 are connected through LANs; 65.2% of PC-installed bases are expected to be connected by LANs in 1994. DataQuest (Mar. 1991).

[2]Network operating systems (NOS) provide network functionalities besides normal (local) file-system functionalities. These typically have network-aware modules embedded within the operating system kernel. Microsoft Lan Manager Server or 4.3 BSD Unix is a very good example of a NOS.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

presents them to the user. The nature and extent of processing on the Client may vary from one CSS to another.

The Client characteristics [B] and [D] set it apart from dumb terminals connected to a Host, since it possesses intelligence and processing capability.[3] On the other hand, Client characteristic [D] must not be confused with a PC connected to a LAN, which downloads all the necessary files from a Server or a Host and does all the processing locally.

In a CSS, a Server is a process, or a set of processes all of which must exist on one machine which provides a service to one or more Clients. It has the following characteristics:

[A]. A Server provides a service to the Client. The nature and extent of the service is defined by the business goal of the CSS itself. Thus, an accounting CSS Server provides an accounting data retrieval and processing service to the Client.

A service provided by a Server may require minimal Server-based computation (e.g., print servers or file Servers) to intensive computations (e.g., database servers or Image-Processing servers).

[B]. A Server merely responds to the queries or commands from the Clients. Thus, a Server does not initiate a conversation with any Client. It merely acts either as a repository of data (e.g., file Server) or knowledge (e.g., database Server) or as a service provider (e.g., print Server).

[C]. An ideal Server hides the entire composite Client-Server system from the Client and the user. A Client communicating with a Server should be completely unaware of the Server plaform (hardware and software), as well as the communication technology (hardware and software). For example, a DOS-based Client should be able to communicate with a Unix or OS/2-based Server in the same manner, regardless of the operating system on the Server(s) and LAN technology connecting the Client to the Server(s).

It is advisable, and desirable, that in a multiserver environment, the Servers communicate with one another to provide a service to the Client without its knowledge of the existence of multiple Servers or intra-server communication. Thus, in such a distributed processing environment, the Client should be unaware of the locale of one or more Servers servicing the Client query or command.

Thus, a Client/Server architecture divides an application into separate processes operating on separate machines connected over a network, thus forming a "loosely coupled" system.[4] An application designer divides the user-defined task into subtasks to be completed either by the Client or by the Server(s) within the constraints posed by business goal of the CSS itself and funtionalities provided by the underlying network operating system. The more advanced the network operating system, the smaller (code-wise) the application will be. For example, Microsoft LAN Manager provides a rich set of functionalities geared toward developing CSS. Thus, a CSS running on top of LAN Manager will itself contain less code—which leads to reduced development time. If, however, the same CSS were to run on top on a network operating system which merely provides file/printer sharing, CSS code can easily double.

Industry Perspective

At present, most corporate applications are either mainframe-centric or PC-server-centric. In mainframe-centric environments, users interact with applications running on mainframes through terminals or PC-based terminal-emulators which have the following characteristics:

- They may present a proprietary User Interface. Typical terminals or terminal-emulators present a non-GUI interface (e.g., IBM 3270 terminals or emulators). In recent years, the terminal-emulator industry has attempted to present graphical interfaces while hiding non-graphical and proprietary interfaces from the users.
- Typically, all user key strokes and cursor positions are transmitted to the mainframe. Thus, no local intelligence or processing is required, except that required for the transmission of the user commands or queries.
- Simple terminals are typically hardwired to the mainframe or to a local terminal controller, which in turn is connected to mainframes via cluster controllers, while PC-based terminal emulators are either remotely connected to the mainframes through modems or connected to the mainframe through a LAN (Figure 1)
- Typically, all results returned from the mainframes are in terms of cursor positions and characters or strings to be displayed at certain positions on the screen. All computations as well as UI control and rendering is done by the mainframe. This causes excessive loading on expensive mainframe resources, such as storage systems and CPU cycles, and is the main disadvantage of mainframe-centric systems.
- Mainframe-centric systems accord tight administrative control as well as comprehensive system management and performance management facilities.

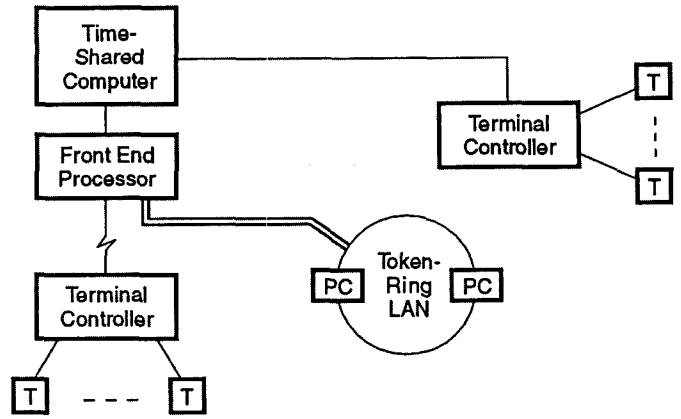
In PC-Server-centric environments, on the other hand, PCs share applications, and data reside on one or more PC-based Servers. This environment provides flexibility to the individual user, but administrative control and system management tools are minimal. These systems have the following characteristics:

- Typically, the PC-based Server is used to share printers and share common applications and data

[3]See section on X-Windows for differentiation between CSS and X-Windows Client/Server.

[4]Ideally, the Client and the Server may exist on the same machine without any network involvement; nevertheless, almost all Client-Server Systems of interest have a network subsystem.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.



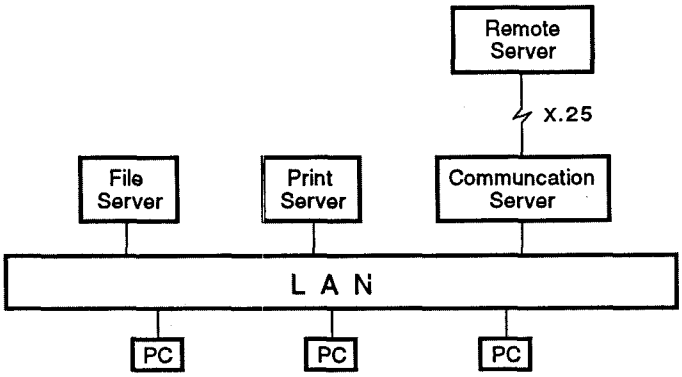


Figure 1. Terminals connected to a time-shared computer

Figure 2. PCs connected to PC-Servers over LAN

(files only). The file Servers provide a range of services to share data among one or more PC applications (Figure 2).

- Each application presents UI and has complete control of the interface as well as rendering of the results. Recent years have seen an explosive growth of GUI-based applications.
- Usually, all user commands or queries are processed on the PC itself. Thus, there must be a large RAM in the PC to enable it to run sophisticated (and thus large) applications. This works against corporate desire to provide cost-effective desktops to the users. Furthermore, the PC-based application may interact with the file Server for accessing (shared or private) data. This causes high volume network traffic since data file(s) must be transported from the file Server onto the PC's local memory. This is particularly true for PC-network-based database programs. Together, these two drawbacks of PC-based applications form the main disadvantage of PC-server-centric systems.

A CSS provides an ideal solution to the drawbacks of both mainframe-centric and PC-Server-centric systems. The most important features of a CSS are

- Desktop intelligence, since the Client is responsible for UI. It transforms the user queries or commands to a predefined language understood by the Server and presents the results returned from the Server to the user.
- Sharing the Server resources (e.g., CPU cycles, data storage) most optimally. A Client may request the Server to do intensive computation (e.g., image processing) or run large applications on the Server (e.g., database servers) and simply return the results to the Client.
- Optimal network utilization as the Clients communicate with the Server through a predefined language (e.g., SQL) and the Server simply returns the results of the command as opposed to returning all the data files.
- Providing an abstraction layer on the underlying operating systems and communication systems such as LAN, allowing easy maintenance and portability of the applications for years to come.

Comparisions with Time-Shared Systems. It is widely acknowledged in the computer industry that CSS systems are being widely considered by corporations for new applications. Often, MIS professionals are confronted with the question "Is Client/Server computing cheaper?" McCarthy et al. [3] present the costs involved in a CSS system and compare it to the typical costs of a mainframe-centric (or time-shared system centric) application. They have broken down the life cycle cost of a CSS into five categories and forecast the cost of developing an application based on the Client/Server computing model, as opposed to the time-sharing model shown in Table 1.

Thus, they summarize:

- Time-sharing has the edge in application development cost through 1992 due to (1) incomplete and incompatible development tools in Client-Server environment and (2) additional complexity of building and testing split applications.
- CSS holds a hardware cost advantage over minicomputers.
- System administration is more difficult in a CSS, since (1) the management tools are immature or

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

missing, (2) a CSS composite consists of multivendor-supplied components, (3) Time-sharing systems allow central administration.

- Software maintenance in a CSS is expected to be lower than in a time-sharing system due to (1) greater reliance on packages as opposed to in-house developments, (2) utilization of new software technology such as object-oriented programming.[5]
- Adoption of the graphical user interface in a CSS will help cut user training costs by as much as 30 to 40%.

Technology

The major components of a Client-Server System are:

- LAN: This is the backbone of the communication subsystem of a CSS and provides low-level communication mechanisms to the network applications.
- LAN-based PC Servers: The Server can be either a packaged product (e.g., file Server, database Server) or a custom-designed Server to meet the needs of the business.
- Mainframe connectivity via PC Server, if desired: This gives the Clients easy access to the vast resources of the existing mainframes. This feature is crucial to the success of a CSS, since it provides a natural migration path to the users and applications downsized from the mainframes.
- Higher-level connectivity support (e.g., RPC) and Client-Server dialogue (e.g., SQL) support
- GUI.

While the Client-Server paradigm itself is not new to the computer industry, it is only now that most of the components of a CSS are commercially available. With the emergence of industry standards in user interfaces such as Microsoft Windows, OSF Motif, IBM Presentation Manager, the development of the Client software has become relatively simple and less time-consuming; these new interfaces, however, have caused escalation of developer training costs.

Similarly, LAN software vendors have provided higher-level communication support. This has been a catalyst in rapid development of the Client-Server systems. The high-level interfaces present a less daunting task of developing and testing the communication subsystems, and allow simpler designs.

Connectivity Interfaces

Low-Level Methods. Traditionally, the emphasis of most early LAN software systems (e.g., Microsoft MS-Net) was on providing such services as file-Sharing and printing. Thus they provided limited and low-level programming interfaces for the LAN system. For backward compatibility reasons, these interfaces are present in the most evolved (or advanced) versions of the corresponding LAN software systems.

Most applications which used (or use) these methods have long development and testing cycles. Software maintenance and portability of these applications are arduous at best. Furthermore, these necessitate business houses to train their programmers to be specialized network programmers. Due to low overhead, however, associated with the transport mechanism itself, high data transmission throughput can be achieved in each case.

Network Basic Input/Output System (NetBIOS). This was originally developed by Sytek, Inc. as a "high-level" programming interface to the IBM PC Network on a broadband adaptor card in August 1984. It was located on the IBM PC Network LAN Adaptor (LANA) as an "extention" to BIOS ROM. Soon, it became a *de facto* session layer standard for the PC LAN industry. In the 1984–85 year, Microsoft developed Microsoft Network (MSNet) software for IBM based on NetBIOS, which formed the basis of a wide array of LAN software systems. Most noticeable were IBM PCLP, original versions of DEC Pathworks and UB Net/One, and 3com 3+Share.

NetBIOS represents a programming interface at the Session Layer as per the OSI model[6] (Figure 3). The programming interface and some implementation specifications have been defined by IBM [9], leaving implementation details to each vendor. Currently, NetBIOS support is provided by all major LAN vendors (e.g., Novell, Microsoft, Banyan) on DOS, Windows, OS/2, and Unix platforms.

Applications interact with NetBIOS support through a data structure called Network Control Block (NCB) (see Figure 4); An application must specify values for *Command* field and may specify values for zero or more fields in the NCB data structure, depending on the NCB Command. Finally, an application "submits" a NCB to the NetBIOS support. The exact interface varies with operating system e.g., while in the DOS environment, an application can call function INT

[5]provided object-oriented tools are available

[6]See Tanenbaum, A. S. *Computer Networks*, Prentice-Hall, Englewood Cliffs, N.J. 07632 1980, for a detailed explanation of the OSI layers.

Table 1.
Relative Cost of Development of a Client/Server System as Compared to a Time-Shared System.

Cost Category / Year	Application Development	Hardware Acquisition	Systems Administration	Software Maintenance	Training
1990	+ 30%	− 20%	+ 30%	0%	+ 10%
1992	+ 10%	− 30%	+ 20%	− 10%	0%
1994	− 10%	− 40%	+ 10%	− 25%	+ 10%

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.