Journal of Information Systems Education

Winter 1993-94

Client-server Computing: Getting Your Feet Wet

ABSTRACT: Tremendous growth of client-server (C-S) computing is expected throughout the 1990s. Computer information systems (CIS) curricula are faced with the ongoing challenge of providing client-server instruction while technologies and methodologies are still evolving. To keep pace with this dynamic environment, faculty must begin building initial client-server experiences into the CIS curriculum. The purpose of this paper is to describe one institution's experiences getting its "client-server feet wet." First, fundamental concepts of client-server are considered. Second, basic questions about introducing C-S to the CIS curriculum are addressed. Next, the introduction of client-server computing at Western Carolina University is described. Finally, future client-server changes are considered.

Lynn R. Heinrichs
Western Carolina University
School of Business
Cullowhee, NC 28723

KEYWORDS: *Client-server, Computer Information Systems Curriculum, Database Management Systems*

INTRODUCTION

Tremendous growth of client-server (C-S) computing throughout the 1990s is expected. In the 1991 bonus issue of Information Week, client-server computing was identified as one of nine technologies for the 1990s. (1) More recently, Stewart Schuster, Vice President of Marketing for Sybase, Inc., predicted that client-server computing "will have pervaded the entire enterprise" by the year 2000. (2)

Many examples can be found already to support client-server growth predictions. For example, Sprint implemented a C-S system to reduce the amount of time required by customer representatives to access customer information from hours to minutes. (3) American President Companies replaced dumb terminals at three West Coast ports of its world-wide shipping business with local area networks (LANs). Access to information on custom regulations and currency conversion, formerly controlled by a central mainframe, was expedited by moving localized data to the LANs. (4) John Wiley and Sons used a client-server scheme combining token-ring, AS/400, personal computer (PC), and LAN server technologies to support accounting, book/distribution/fulfillment, and complimentary copy fulfillment. (5) Such examples routinely can be found in information systems literature.

The arrival of client-server computing places yet more pressure on the information systems curriculum to make some fundamental changes. Understanding client-server involves more than simply learning a new technology. As David Litwack explains:

It involves rethinking the definition of basic concepts. Applications flow differently and integrate closely with personal productivity tools. Transactions more closely resemble the way people work, rather than being limited to a sequence of flat screens. (6)

Because of this shift in the way applications are designed and developed, IS faculty will face a long and steep learning curve that must begin with some basic client-server experimentation.

This article describes the experiences of one IS curriculum getting started with client-server computing. First, C-S definitions and concepts are presented. Second, basic curricular questions are considered. Next, the introduction of C-S computing to the CIS curriculum at Western Carolina University is described. Finally, future client-server changes are considered.

CLIENT-SERVER DEFINED

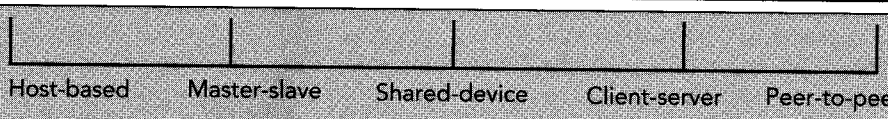
As shown in Figure 1, client-server represents one architecture on a distributed processing continuum. A tendency sometimes exists to think of clients and servers in terms of hardware configurations. For example a Paradox SQL Link reference manual, defines client in terms of software "residing on user workstations." (7) McGoveran considers this perspective to be too narrow:

A server is a logical process which provides services to requesting processes. . . Processes which request services from a server are called the clients of the server. There is no concept of a client without a server. . . the term client should not be confused with hardware, i.e. processors connected to hardware "servers." (8)

McGoveran's client-server perspective allows for client and server processes to exist on the same machine.

Key strengths of the client-server approach include: (1) savings in host processing resources, (2) scalability of client and server platforms, and (3) interoperability of multiple servers. However, client-server applications may suffer from data integrity problems, performance overhead, more complex information systems management, and increased system administration costs. (9)

Figure 1: DISTRIBUTED PROCESSING CONTINUUM



Host-based. All application processing is done on a central host computer. User access is through dumb terminals

Master-slave. Slave computers capable of limited application processing are directed by a master computer.

Shared-device. Personal computers (PCs) perform application processing while a server coordinates requests for peripherals such as disk drives and printers.

Client-server. Application processing is divided between client and server, but controlled by the client. Clients request services from the server.

Peer-to-peer. All computers are equal. Any computer can act as either a client or server by requesting or providing services.

Winter and Dove have identified four generations of client-server architectures (Table 1). (10) Each generation differs in terms of the messages and data exchanged between client and server. The **first generation** involved an application, acting as a server, receiving terminal inputs from another program, acting as a client. The two programs could exist on different machines. **Second generation** C-S architectures involved file servers providing clients with the capability to read and write files. In the **third generation**, the server responded to requests from clients for database operations. The **fourth generation** differs significantly from the other three. Rather than treating messages and services as data and operations on the data, the fourth generation C-S supports large complex objects and conversation models that specify interactions between clients and servers.

The differences in architectures across the four generations described by Winter and Dove document the confusion that can exist when using the term "client-server." Client-server does not mean the same thing to all people. According to McCauley, solutions characterized as C-S can range from the simple to complex. (11) Simple solutions use the server to provide remote data management functions while complex solutions partition business logic across both the client and server.

CLIENT-SERVER CURRICULUM DECISIONS

Because client-server solutions can vary in complexity, the best approach for introducing C-S to the computer information systems (CIS) curriculum is not straightforward. Differences in opinion may exist among faculty regarding what to teach and where to teach it.

In plotting a C-S strategy for the CIS curriculum at Western Carolina University, faculty sought answers to the following three questions:

1. **What scope of client-server solutions should be taught?**
2. **Where should hands-on C-S coverage be included?**
3. **How should C-S coverage be delivered?**

Since client-server technology will continue to evolve throughout the decade, the answers to these questions will require review on an ongoing basis. The intent of the initial effort described here was to gain some early C-S learning experiences that would facilitate an on-going decision-making process.

WHAT: CLIENT-SERVER EMPHASIS

As described earlier, client-server solutions can range from the simple to complex. When considering the inclusion of C-S experiences in the CIS curriculum, the logical place to start was determining the scope of solutions to be emphasized. For example, should solutions involve characteristics of simple, third-generation (database server) architectures or complex, fourth-generation (information server) systems?

Since the immediate objective was to gain some C-S experience, not necessarily attack large, complex problems, solutions that involved remote data management properties seemed the most appropriate place to start. Once these fundamental experiences were in place, the curriculum could be revised to involve more comprehensive solutions involving the partitioning of logic across client and server.

WHERE: TARGET COURSE FOR COVERAGE

Computer information systems majors at Western Carolina University complete eight CIS courses in addition to the business core. The eight courses and their prerequisites are listed in Table 2. The criteria for selecting the most appropriate course for introducing hands-on C-S skills were: (1) compatibility of C-S coverage with existing course content and (2) faculty expertise in teaching hands-on C-S skills.

Based upon the two criteria given, the most likely candidates for inclusion of hands-on client-server work appeared to be either Application Development II or Database Management Systems. The first option, **Application Development II**, focuses on the development of business applications using third and fourth generation tools. Applications stress the use of interactive user interfaces, reusable program modules, and random-access file organizations. Although this course is undergoing a slow transition to include new development methodologies, technologies, and environments, introduction of C-S skills would have required dramatic changes for which faculty expertise and available resources did not exist. At some point in the future, this course will be a likely target for development of comprehensive C-S applications that split processing logic and data across client and server.

The second option, **Database Management Systems**, provides students with the concepts and skills needed to complete the development of a database application. A comprehensive project is typically developed using a PC relational database product. Interactive SQL is also used apart from the project as a query tool.

Table 1: GENERATIONS OF CLIENT-SERVER

No.	Characteristic	Message exchange	Data exchange
1	Terminal emulation	Terminal input/output	Lines, screen images
2	File server	Record, file request Status, error codes	Records, blocks
3	Database server	SQL request Standardized responses	Relations
4	Information server	Application-defined conversations	Objects

Table 2: EXISTING CIS REQUIREMENTS

Course Id	Course Title	Prerequisites
CIS 251	Management Information Systems	None
CIS 256	Structured Programming	CIS 251
CIS 258	Application Development	CIS 256
CIS 358	Application Development	CIS 258
CIS 365	Hardware, System Software, and Communications	CIS 256
CIS 453	Database Management Systems	CIS 358
CIS 455	Systems Analysis and Design	CIS 453
CIS 465	Information Resource Management	All CIS courses

This option was chosen as the best for gaining some C-S learning experiences for several reasons. First, since the initial emphasis was to be on remote data management, experimenting with client-server solutions in the database management system course was logical. Second, because many C-S products rely on SQL and the relational model, topics already taught in the database course, content could be easily adjusted to incorporate hands-on client-server experiences.

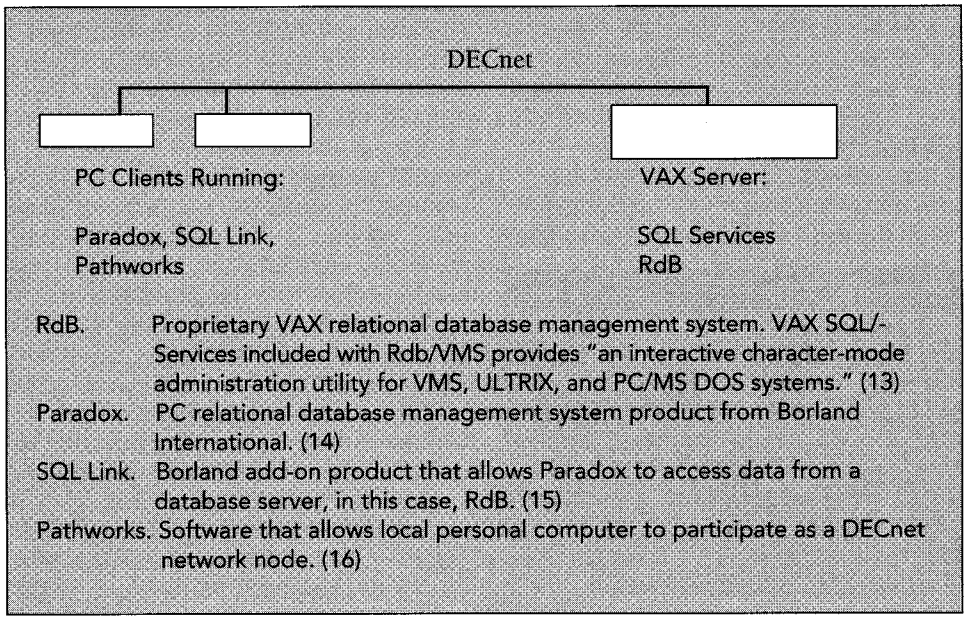
HOW: CLIENT-SERVER PLATFORM FOR DELIVERY

Salemi defines four possible platforms for building client-server systems: (1) personal computers, (2) RISC and other UNIX workstations, (3) minicomputers, and (4) mainframes. (12) Since resources currently available to the School of Business included a local area network (Netware) with Ethernet access to the University's VAX minicomputer, the first and third platforms were the only real options worth considering for initial C-S work.

PC Platform. Using this approach, PCs on the local area network would act as clients requesting services from a dedicated server also on the LAN. Although the appropriate hardware and communications were present for this option, a C-S database management system (DBMS) was needed. Commitment to such a software product seemed premature given the limited background of faculty with C-S solutions. For the purpose of building a foundation of experiences on which future decisions could be made, a less expensive and less traumatic choice was desirable.

Minicomputer Platform. The less expensive option, although less flexible, was the minicomputer platform. For this option, the existing VAX could act as a server using the proprietary DBMS, RdB. On the client side, Borland's add-on product, SQL Link, could be incorporated with Paradox to provide access to RdB data. Since both Paradox and RdB were already accessible and familiar to faculty, using the minicomputer platform had great appeal. The cost of SQL Link, the product needed for providing the interface between Paradox and RdB, was nominal. Because of this small resource commitment and the existing familiarity with the two DBMS products, the minicomputer platform (Figure 2) was a logical choice for gaining some C-S experience.

Figure 2: CLIENT-SERVER PLATFORM



REVISING THE DBMS COURSE

Selection of the minicomputer platform for delivering solutions provided an easy transition to client-server computing in the database management system course. Minimal changes were needed to course schedule and content coverage to provide some basic C-S experiences.

BEFORE CLIENT-SERVER

Using Kroenke's database text, the DBMS course traditionally covers concepts and skills associated with the development of databases and database applications. (17) As shown by the objectives in Table 3, hands-on experiences have typically included assignments using both Paradox and SQL.

Paradox coverage. Consistent with Kroenke's coverage of both database and application design, students were required to build a Paradox application using the PAL programming language that provides maintenance (change, delete, add rows), query, and report generation functions for a multi-table database. Applications were menu driven using an object/action strategy as defined by Kroenke. Using this approach, the highest level menu allows users to first pick an object (e.g. PART, SUPPLIER) to be processed. At the next menu level, users select the action (e.g. ADD, CHANGE, FIND) to be taken regarding the object.

SQL coverage. The SQL component of the course had been taught with several different products on both the PC and VAX minicomputer. Typically, students use interactive SQL queries to: (1) create a multi-table database, (2) maintain database data, (3) complete single-table queries, and (4) perform multiple-table queries.

AFTER CLIENT-SERVER

The addition of hands-on client-server coverage has allowed for integration of the Paradox and SQL course components. Paradox's PAL programming language has continued as the vehicle for providing database application function. However, the database targeted for processing by an application is located on the VAX rather than the PC. Application procedures that previously operated on Paradox tables are now required to access and update RdB data. To provide the link between Paradox and RdB, embedded SQL statements are used within PAL programs.

Although the specific problems to be solved by students differ each semester, sample Paradox scripts similar to those required of students are shown in Figures 3

Table 3: COURSE OBJECTIVES BEFORE CLIENT-SERVER

Upon completion of the course, the student will be able to do the following:

- Describe differences between file and database processing.
- Describe the database development process.
- Compare and contrast different approaches for modeling database requirements.
- Use object diagrams for specifying database requirements.
- Use relation diagrams for specifying a database logical design.
- Compare/contrast major database models including the relational, hierarchical, and network.
- Design and implement a database application using Paradox.
- Access and manipulate database data using SQL.

and 4. As previously mentioned, Paradox applications developed by students typically require basic maintenance, query, and report functions. In these examples, an RdB table, PARTS, contains four attributes describing parts purchased from suppliers: part number, part name, weight, and color. Figure 3 demonstrates retrieval of data from PARTS that is used subsequently by Paradox for generation of a simple report. Figure 4 shows a Paradox script for retrieving a PARTS row, changing its contents, and updating the RdB database.

Using SQL Link to achieve remote data management required additional content be added to the DBMS course. Coverage of the following concepts is critical for helping students make the transition from Paradox-only applications to development using SQL Link:

1. Creation of replica tables. To complete any embedded SQL query using Paradox SQL link, replica tables must be created in Paradox that duplicate the structures of the original RdB tables. Replicas (not visible by examining the sample scripts) are created using an SQL Link utility program. Students must learn how to create and change the replica tables.
2. Manipulation of data using local table. As shown in the Figure 4 sample script, data retrieved from an RdB table is stored in a local Paradox table for manipulation. Changes are made to the local table and then rewritten to the RdB table. Students must understand the relationship between the local and remote tables to maintain data integrity.
3. Use of embedded SQL statements. If students previously have not had exposure to embedded SQL, using Paradox SQL Link requires basic coverage of embedded SQL commands. All SQL commands within a PAL program must be blocked by the SQL and ENDSQL markers. Additionally, several new commands are available for such purposes as error handling (e.g., SQLERRORCODE and SQLERRORMESSAGE), transaction processing (e.g. SQLCOMMIT and SQLSTARTTRANS), and connection management (SQLMAKECONNECT AND SQLBREAKCONNECT).
4. Diagnosis of errors. With introduction of the SQL Link, determining the source of a data manipulation or updating error can be complicated since a problem can originate from Paradox or RdB. SQL Link provides some diagnostics not available in Paradox that help students locate database problems.

Although many textbooks exist on the market that cover fundamentals of Paradox, few are available that provide much help in using SQL Link. To provide instructional resources for students, faculty had to assume the responsibility of preparing handouts from a variety of sources. At this point in time, lack of instructional materials would likely be a problem no matter what C-S platform was being used.

FUTURE CHANGES

Although the use of SQL Link has been suitable for introducing basic remote data management skills, complex C-S application development involving the distribution of processing logic across both client and server still needs to be addressed by the CIS curriculum at Western Carolina University. As previously mentioned, the greatest obstacles to achieving fourth generation client-server development is (1) available resources and (2) faculty expertise.

The most immediate objective in moving client-server computing forward is to acquire a comprehensive development environment for building C-S applications. Although the current minicomputer platform has been adequate for developing some initial learning experiences, Salemi sees few advantages in using this proprietary approach except when mission-critical data is already stored on the minicomputer. (12) Given that a PC-based environment could (1) provide greater long-term flexibility for delivering C-S instruction and (2) exist with currently available hardware and communications, the CIS faculty believe that future resource acquisitions should establish client-server

Figure 3: SAMPLE REPORT SCRIPT

```
;PAL Script to Produce a Master Parts List
;
;Retrieve all rows from remote PARTS table and
 store in local ANSWER table
;
;SQL
  SELECT * FROM PARTS
ENDSQL
;
;Copy rows from ANSWER table into local table
 TPARTS for which
;Master List Report has been defined
;
;EMPTY "Tparts"
ADD "Answer" "Tparts"
;
Print Master List Report
;MENU {Report} {Output} {Tparts} {1} {Printer}
```

Figure 4: SAMPLE UPDATE SCRIPT

```
;PAL Script to update remote Parts Table
CLEARALL
CLEAR
@2,1
??"Enter part number to update: "
ACCEPT "A2" TO PIN
;
; Retrieve rows from remote PARTS table and
 store in local ANSWER table
;
SQL
 SELECT * FROM PARTS WHERE PNUM =
 "~PIN~"
ENDSQL
;
; Copy row from ANSWER table into local table
 TPARTS for which
; part data entry form has been defined
;
EMPTY "Tparts"
ADD "Answer" "Tparts"
;
; Edit part data using default form
;
MENU {Modify} {Edit} {Tparts}
MENU {Image} {Pickform} {F}
WAIT TABLE UNTIL "F2"
;
; Store data in variables for updating remote table
;
Name = [Pname]
Color = [Color]
Weight = [Weight]
DO_IT!
;
; Update remote table using Paradox query
;
MENU {Ask} {Parts}
MOVETO [Pnum]
TYPEIN "~PIN"
MOVETO [Pname]
TYPEIN "changeto ~Name"
MOVETO [Color]
TYPEIN "changeto ~Color"
MOVETO [Weight]
TYPEIN "changeto ~Weight"
DO_IT!
```

instruction on a PC platform.

To create the desired PC-based environment, a client-server, relational database management (RDBMS) system product is required. The CIS faculty have established the following general requirements for selecting appropriate software. The product chosen must:

- Execute in a Netware operating system environment.
- Provide access to data through SQL queries.
- Provide Windows-based, front-end tools for application development.
- Support object-oriented application development.

Although the existing family of Borland products would meet the defined requirements, other vendors also are being considered with the hope that a software grant can be obtained. Once the RDBMS has been determined, the CIS faculty will face the problem of upgrading hardware. Only 25% of existing laboratory PCs are capable of participating as clients in the future environment.

CONCLUSIONS

Client-server growth will continue throughout this decade and beyond. Although C-S technologies and methodologies are still evolving, CIS curricula need to begin the client-server learning processing by gaining some initial hands-on experiences. Although different learning approaches are suitable for different institution, this paper described one school's successful experimentation in getting its "client-server feet wet." These initial experiences are proving invaluable in making long-term, client-server decisions.

REFERENCES

1. DePompu, Barbara. "More Power at Your Fingertips," *Information Week*, December 30, 1991 (Bonus Issue), pp. 22-23.

2. "Remapping the Computing Environment for the 90s: The Impact of Client/Server," *Sybase Executive Summary*, 3(1), Winter 1993, p. 5+.

3. Rinaldi, Damian. "Mining Business Nuggets," *Software Magazine*, November Special 1992, pp. 14-15+.

4. O'Leary, Meghan. "Work in Progress," *CIO*, November 1, 1992, pp. 52-60.

5. Client/Server Computing with the AS/400. A Duke Communication International White Paper published as a supplement to News 3X/400, 1992.

6. Litwack, David and Beggs, John. "Should Client/Server Development Be Done by a Separate Group?" *Software Magazine*, September 1992 (Client/Server Special Edition), p. 70.

7. Buzzard, James. "The Client-Server Paradigm: Making Sense Out of the Claims," *Data Based Advisor*, August 1990, pp. 72-79.

8. *Paradox SQL Link User's Guide*. Borland International, 1991.

9. McGoveran, David. *Evaluating an RDBMS for Client/Server Applications*. Alternative Technologies, 1992.

10. Winter, Richard and Dove, John. "Age of the Information Server," *Database Programming & Design*, June 1991, pp. 48-55.

11. McCauley, James. "Client-Server: Achieving the Promise," Borland International sponsored presentation at the Annual Meeting of the Decision Sciences Institute, San Francisco, CA, November 22, 1992.

12. Salemi, Joe. *PC Magazine Guide to Client/Server Databases*. Emeryville, CA: Ziff-Davis Press, 1993.

13. *RdB*. Version 4.0. Computer Software. Digital Equipment Corporation, 1992.

14. *SQL Link*. Version 1.0. Computer Software. Borland International, 1990.

15. *Paradox*. Version 3.5. Computer Software. Borland International, 1990.

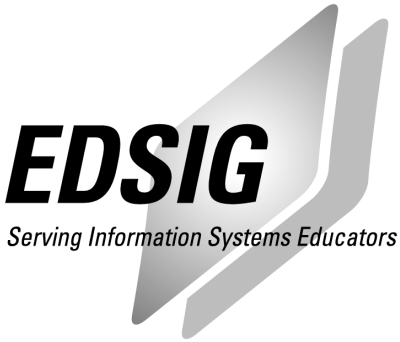
16. *Pathworks for DOS*. Version 4.1. Computer Software. Digital Equipment Corporation, 1991.

17. Kroenke, *David. Database Processing: Fundamentals, Design, Implementation*. SRA, 1992.

AUTHOR'S BIOGRAPHY

Lynn R. Heinrichs is an Assistant Professor of Computer Information Systems and Network Administrator for the School of Business at Western Carolina University. She received her BS and MS degrees from University of Illinois-Urbana and the Ed.D. from Northern Illinois University. She teaches in the areas of application development, database management, and telecommunications. Her research interests include client-server application development, object-oriented development tools, and curriculum design.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096