A Study Of Technologies For Client/Server Applications

Wei Pan Feinstein
Advisor
Dr. J. Scott Hawker
Department of Computer Science
University of Alabama

Abstract-The growing size and complexity of software systems has exposed many shortcomings of traditional software engineering models. This has increased interest in Client/Server development. The driving force of Client/Server computing is the fundamental belief that personal computers connected to small servers provide the best price-performance. One of the most important decisions a developer needs to make is to choose the optimum development technologies and tools. This has become more feasible with the advent of the powerful specialized Client/Server development technologies and tools. The purpose of this paper is to identify the various Client/Server technologies that are available, and to explain how the technologies are used, and compare and contrast these various technologies in the Client/Server model. In order to give a better overview of these technologies and tools, this paper will utilize a simulated e-commerce online book-order model. The technologies will be used to implement this model.

1. Introduction

The advent of powerful and inexpensive PCs has made it possible to shift from mainframe-centric systems towards Client/Server systems. This shift has been accelerated by the fact that the Client/Server model makes cooperative computing practical and manageable by supporting the division of applications into functions and services that need to be shared by many users [1]. The Client/Server model in its simplest form allows developers to split the processing load between two logical processes: the client and server (See Figure 1). The owner of the resources or services is called the server while the user is called the client. Both the client and the server may exist on the same physical computer, such as local database servers that run native to Windows.

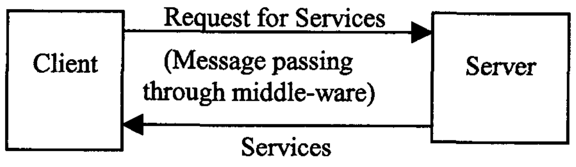

Figure 1 Client/Server Model

Client/Server computing is a common model for distributing resources. Many of the Client/Server systems run the client and server processes on separate computers. From the client view obtaining a service just requires sending a request to the server (or middleman) and receiving a reply back.

In a distributed Client/Server system, middleware is the glue that binds the clients and the servers together into a cohesive system. Middleware is an enabling software layer that provides a transparent means of accessing information between clients and servers. Middleware insulates the user application from the intricacies of the various operating environments on which the application is running.

The exploding popularity of the World Wide Web makes the Web-based Client/Server architecture one of the most important applications of the Client/Server model. The Web browser and Web server play the central roles in this model.

The three-tier Client/Server model is currently the most commonly used method in the Client/Server arena. The three tiers are the user interface tier, the business rules tier, and the database tier [2]. The interface tier is responsible for interfacing the user to the application. It often provides graphical application information displays, gathers and manages user input to the application. The business rules tier is responsible for controlling the application execution and enforcing business rules. The database tier is responsible for interfacing to the underlying DBMS (Database Management System) that provides permanent physical data storage.

The three-tier model makes the application less fragile by further insulating the client from changes in the other parts of the application [2]. It also gives more flexibility in changing an application for either the client or server.

This paper will study these and other Client/Server technologies and demonstrate how the technologies can work together to build large, complex applications. Section 2 provides an overview of the problem and some background. Section 3 details and contrasts various Client/Server architectural styles that leverage Client/Server technology. Section 4 highlights the key results and describes possible follow-up work.

2. Overview of the Problem

2.1Overview of Previous Work

The Client/Server model has been in use for a number of years. A large number of client/server technologies and development tools are available in the current market. Below is the summarized research in this area.

The evolution from mainframe-based architecture to the Client/Server architecture has been discussed widely [6]. The three-tired Client/Server architecture has also been of significant interest. An excellent review of the three-tier model and related issues is found in the work by Edwards's [7]. Numerous articles discussing the middleware technology CORBA are found in the literature. Tim Berners-Lee and several other people [5] discuss the background and architecture of CORBA thoroughly in the work. An electronic banking system was designed and implemented as a case study by using CORBA technology [2]. In a work by Berg the use of Java RMI from an applet is discussed [9]. A Java-CORBA solution is found in a study available on the World Wide Web [10]. Using a stock market simulation [11] develops a comparison of client server middleware including CORBA, DCOM, and Java/RMI.

The Web-based Client/Server technologies have been addressed widely. First-generation Web-based applications for enterprise computing typically use traditional Client/Server architecture with CGI programs [12]. In the work by Vetter [12] the server-side web applications is also addressed by using servlet and ASP. Case studies implementing a database retrieval system are implemented in servlet by Karl Moss [14].

To date there has been no comprehensive study of the Client/Server technologies but rather they have been studied and addressed separately in wide variety of papers under different situations. It would be very helpful to the researcher as well as practitioner to develop a comprehensive treatment of these important new technologies.

2.2 Overview of the E-Commerce Project

It is difficult for the practitioner to maintain an awareness of the multitude of options available and select the appropriate technology and tool to develop a solution. It is therefore important and very useful to develop a case study to help the practitioners understand what is available. The project designed for this paper is similar to the AMAZON.COM online bookstore. In this project customers can retrieve a book inventory to find out which books are available and how much they cost. Books can also be ordered online by completing an order form. After the order form is submitted the book inventory is updated automatically. The project is a simplified simulation of a real world case.

3. Client/Server Architectures

3.1 Introduction

This section details alternate Client/Server architectures. They are as follows: Section 3.2.1 studies shifting functionality between client and server; the thin-client model and fat-client model will be addressed. Section 3.2.2 applies middleware communication technologies, such as CORBA, RMI and DCOM to the distributed Client/Server model. Section 3.2.3 applies Web-based Client/Server technology. Section 3.2.4 applies three-tier Client/Server technology. Section 3.2.5 designs distributed three-tier Client/Server architectures. The major functions required to accomplish this design are listed as follows:

1. Initiate session, such as launch application and Web URL
2. Display the welcome page.
3. Connect to the bookstore inventory.

4a. Form the page of inventory content.
4b. Display the contents of the inventory
5a. Form the customer order form.
5b. Display the customer order form
5c. Collect order data and submit order information.

6. Validate the customer's input data, such as the book amount that customer orders has to be an integer and

order quantity has to be less than that in stock.
7. Update the bookstore inventory.
8a. Form the order-complete notice.
8b. Display the order-complete page and ask if order will continue. If so, repeat.

Both static and dynamic diagrams are drawn for each of the models. Static diagrams show the structure and connection of components in the system and basic data flow. The dynamic diagram traces the message flow sequence when the client and server communicate with each other to execute transactions online.

3.2 Client/Server Architectures

3.2.1 Basic Client/Server Architecture

Client/Server architecture has two basic models; depending on which side the workload is assigned to in the Client/Server model. In the thin-client and fat-server model, the client computer is responsible for nothing more than presentation management; the server is responsible for the majority of functionality. (See Figure2) The functions implemented by client and server are included in client-side and server-side respectively. The fat-client and thin-server model is the opposite; the server is responsible only for managing the database, the client does almost everything else. (See Figure 3)

The balance of the job load between the client and server can be varied from application to application. Invariably there are always tradeoffs on this issue. For example, the design decision can be to move the user-input validation from the server to the client; this would reduce network traffic. Another design decision could be to move the order form provider from server to client; this would shorten customers' waiting time. These are design issues to be determined at the application design phase.

3.2.2 Distributed Client/Server Architecture

Middleware technology plays the central role in the distributed Client/Server architecture. Based on basic Client/Server architecture, we expand this model by adding CORBA (Common Object Request Broker Architecture), RMI (Remote Method Invocation), and DCOM (Distributed Component Object Model) technologies into the architecture. CORBA functions as the middleware to connect the client program and server program together based on the generic Client/Server model (See Figure 4). The client machine and the server machine are physically connected across networks. The network communication protocol is TCP/IP. The ORB (Object Request Broker) is the middleman as its name implies. It needs to be installed on both client machine and server machine. The ORB connects a client application with the object it wants to use, the client program only needs to know the object's name and server ORB is IIOP (Internet Inter-ORB Protocol), a standard protocol for communication between ORBs on TCP/IP network protocol [13]. IIOP enables ORB-based communication across networks. The client and server programs reside individually on the client machine and server machine. In order to represent the basic Client/Server model, the database resides on the server side locally.

The service interface is defined in IDL (Interface Definition Language) at the application implementation stage. This interface presents the clients the services that the server provides. By compilation of the IDL file the compiler generates a server skeleton and client stub. The actual communication between the client and server applications is through the client stub and server skeleton. Client stub and server skeleton act like local server or local client, the client stub is responsible for getting remote object references. They are also responsible for marshalling and unmarshalling parameters to accomplish language-independence as well as interacting with ORB. Client stub and server skeleton are included in the implement applications on both client and server side.

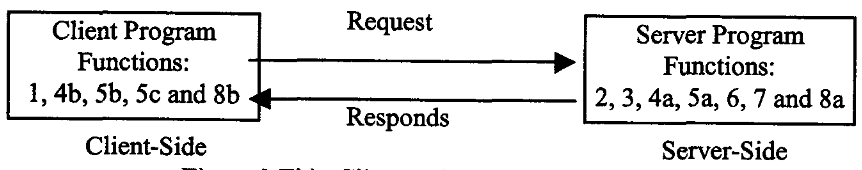

Figure 2 Thin-Client and Fat-Server Architecture

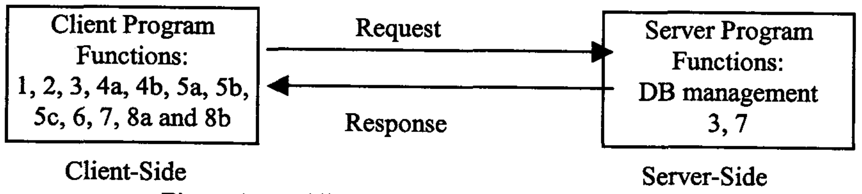

Figure 3 Fat-Client and Thin-Server Architecture

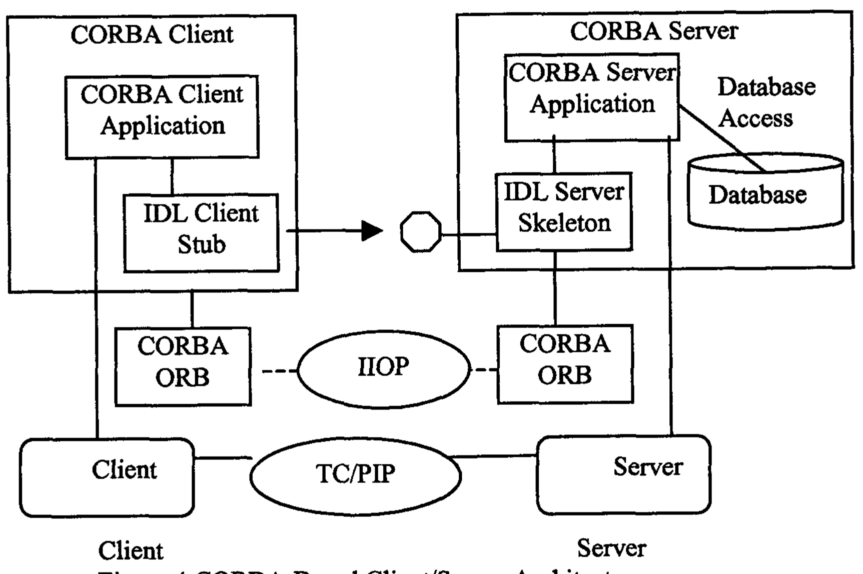

Figure 4 CORBA-Based Client/Server Architecture

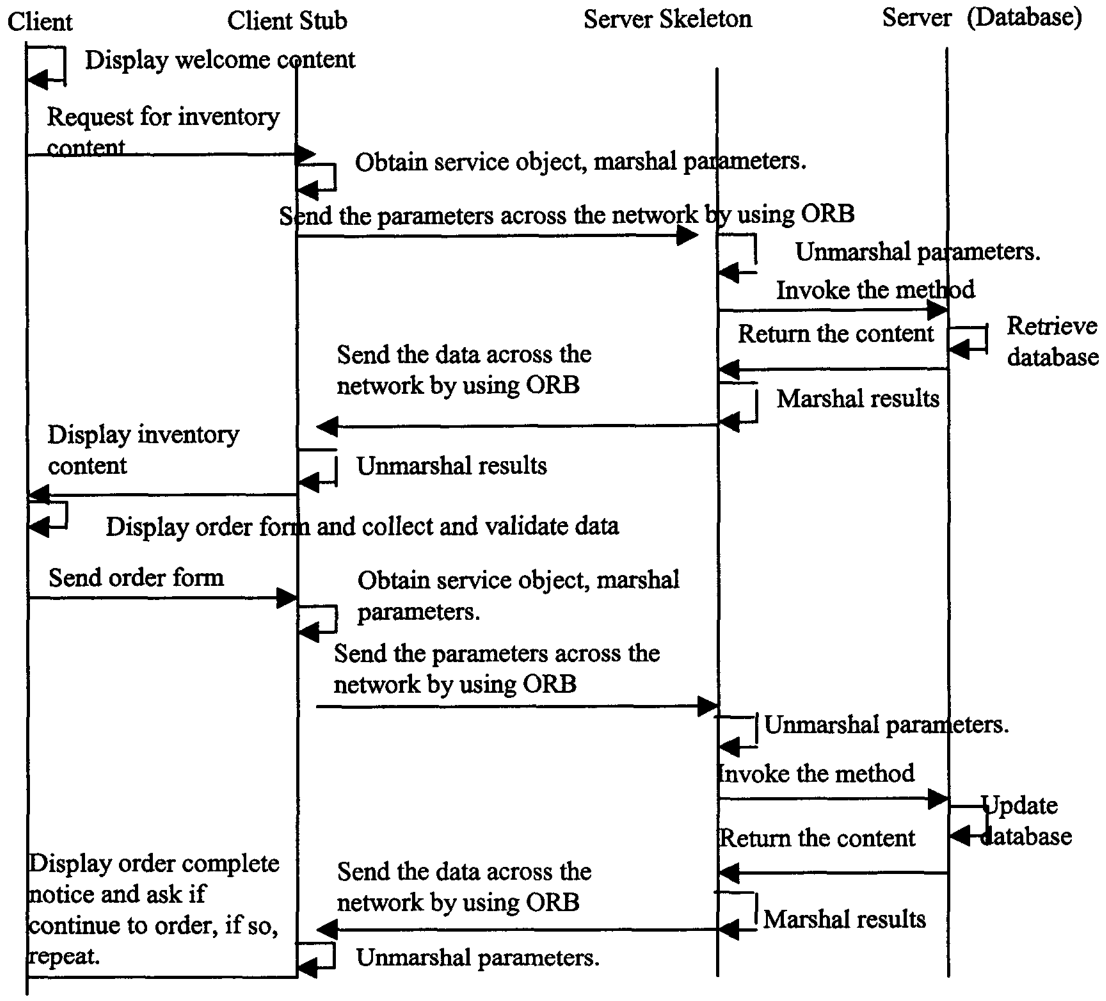
Client
Client Stub
Server Skeleton
Server (Database)
Display welcome content
Request for inventory content
Obtain service object, marshal parameters.
Send the parameters across the network by using ORB
Unmarshal parameters.
Invoke the method
Return the content
Retrieve database
Send the data across the network by using ORB
Marshal results
Display inventory content
Unmarshal results
Display order form and collect and validate data
Send order form
Obtain service object, marshal parameters.
Send the parameters across the network by using ORB
Unmarshal parameters.
Invoke the method
Update database
Return the content
Display order complete notice and ask if continue to order, if so, repeat.
Send the data across the network by using ORB
Marshal results
Unmarshal parameters.

Figure 5 Message Tracing of CORBA-Based Client/Server

The client and server applications can be implemented in many languages, such as C++ and Java.

Message tracing of the CORBA-based Client/Server architecture is shown in Figure 5. It is based on the basic Client/Server architecture, therefore, it is more focused on the internal communication part between the client and the server with middleware involved. The client is responsible for providing all the forms and representing pages for the customers as well as validating the customer-input data.

3.2.3 Web-Based Client/Server Architecture

Because of the growing popularity of the World Wide Web, adapting the Client/Server model into this environment is very important. Java is inherently a network-capable language. The platform independence of a Java program, combined with the automatic on-demand deployment of a Java applet provided by the Web, makes it widely used on the Web. Figure 6 is an HTML with applet embedded Web-based Client/Server architecture.

When a Java application is compiled, the result of compilation is a file of Java bytecode. The bytecode resides on the server; when the client accesses an HTML page that has Java applet embedded program, the bytecode is transmitted to the client. This bytecode is then parsed through an interpreter that resides on the client. This interpreter is called JVM (Java Virtual Machine). Figure 7 is the message-tracing diagram based on the pure HTML web-based Client/Server architecture. The client application is responsible for providing and presenting the forms and pages for the user and checking validation for providing and presenting the forms and pages for the user and checking validation for the user input; the server application is responsible for accessing the database. To simplify the diagram, the Java applet is placed with the JVM after it is downloaded.

Java RMI is Sun Microsystem's CORBA-like architecture. One advantage of RMI over CORBA is that the RMI application has less overhead and better performance than CORBA. A disadvantage is that RMI is a Java-only solution. DCOM is Microsoft proprietary distributed Client/Server middleware technology. Refer to my thesis for details of these two technologies.

3.2.4 Three-Tiered Client-Server Architecture

Dividing the application logic and database server into separate parts leads to the three-tiered Client/Server model. In the Web-based environment, there are several technologies that implement this model. CGI (Common Gateway Interface), ASP (Active Server Pages), and Servlet are among those technologies accessing databases via ODBC (Open Database Connectivity) or JDBC (Java Database Connectivity) or by using embedded SQL.

The traditional Web-based Client/Server architecture is with CGI scripts (See Figure 8). CGI is a standard for interfacing external applications with information servers, such as HTTP or Web server [23]. A plain HTML document that the Web browser retrieves is static, which means it exists in a constant state: a text file that does not change. A CGI program, on the other hand, is executed in real-time, so that it can output dynamic information.

This three-tier CGI Client/Server model needs the Web browser and Web server installed on both the client machine and server machine separately. The Database server resides on the same platform as the business application server. ODBC is the interface used in this architecture for the server application connecting to the database. CGI engine needs to be installed on the middle-tier application server machine

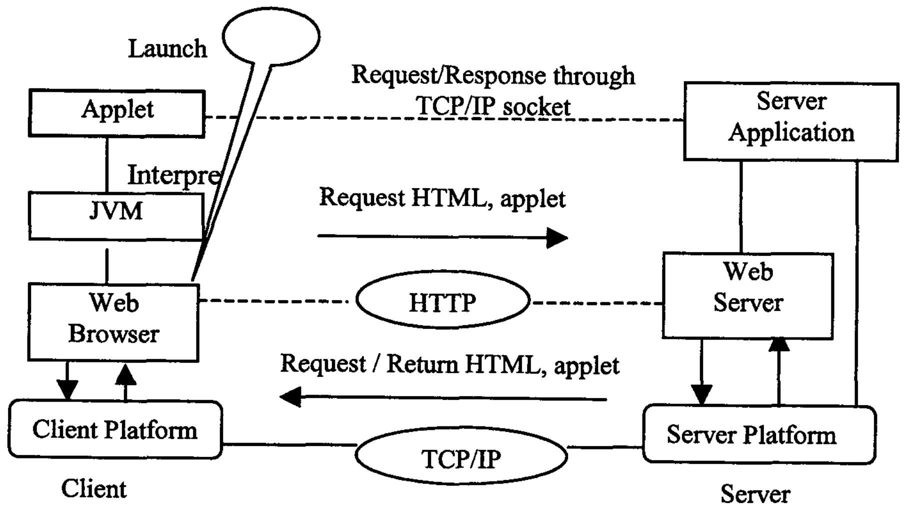

Figure 6 HTML with Applet Web-Based Client/Server Architecture

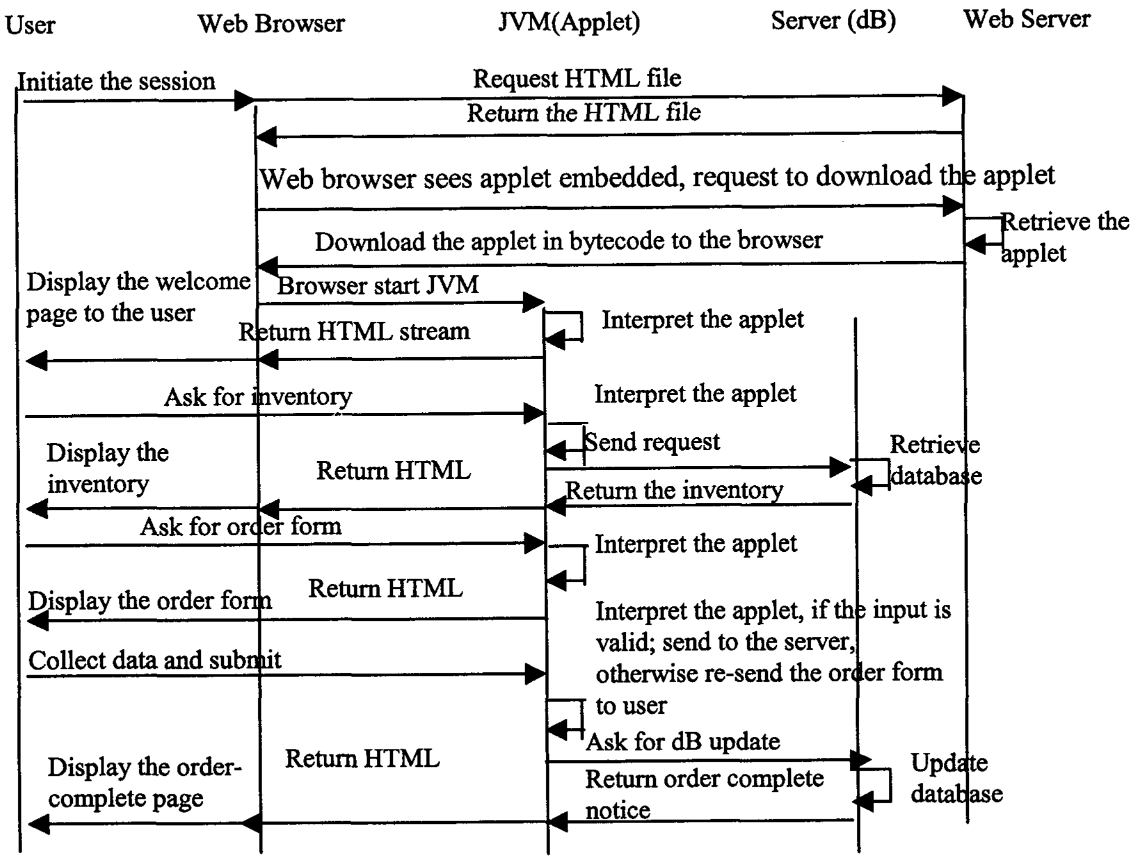

Figure 7 Message-Tracing of the HTML with Applet in Web-Based Client/Server Architecture

to execute the CGI code. Figure 9 is the message tracing of the three-tier web-based Client/Server model. The client is responsible for presenting to users the responding pages and forms. The server does the rest of work.

ASP is Microsoft's solution to server side scripting; it is a good substitute for CGI [3]. Servlet is a server-side component in Java, which dynamically extends the functionality of a server. Refer to my thesis for details of these two technologies.

There are other kinds of three-tier Client/Server technologies, such as SSJS (Server Side JavaScript) and ColdFusion. Because the Client/Server architectures with these technologies are similar with those studied above, they are not addressed in detail in this paper.

The database-centered architecture is currently being widely used Report generators such as BrioQuery or CrystalReport are used to create the user interface, to collect user input and to present the results from the database. The application logic defines the business rules by using embedded SQL to query the database or by using ODBC.

3.2.5 Design of Distributed Three-Tier Client/Server Architecture

In the previous sections several different paradigms of the Client/Server model have been addressed. Initially the basic Client/Server structure was considered by using both thin-client fat-server and fat-client thin-server models. The distributed Client/Server structure was then considered by applying CORBA, DCOM, and RMI middleware technologies. Next the basic Client/Server model was expanded from LAN to the World Wide Web. The final consideration was to expand two-tier Client/Server model to three-tier Client/Server model by separating the application logic and database server. Technologies such as CGI, ASP, and Servlet were used in this paradigm.

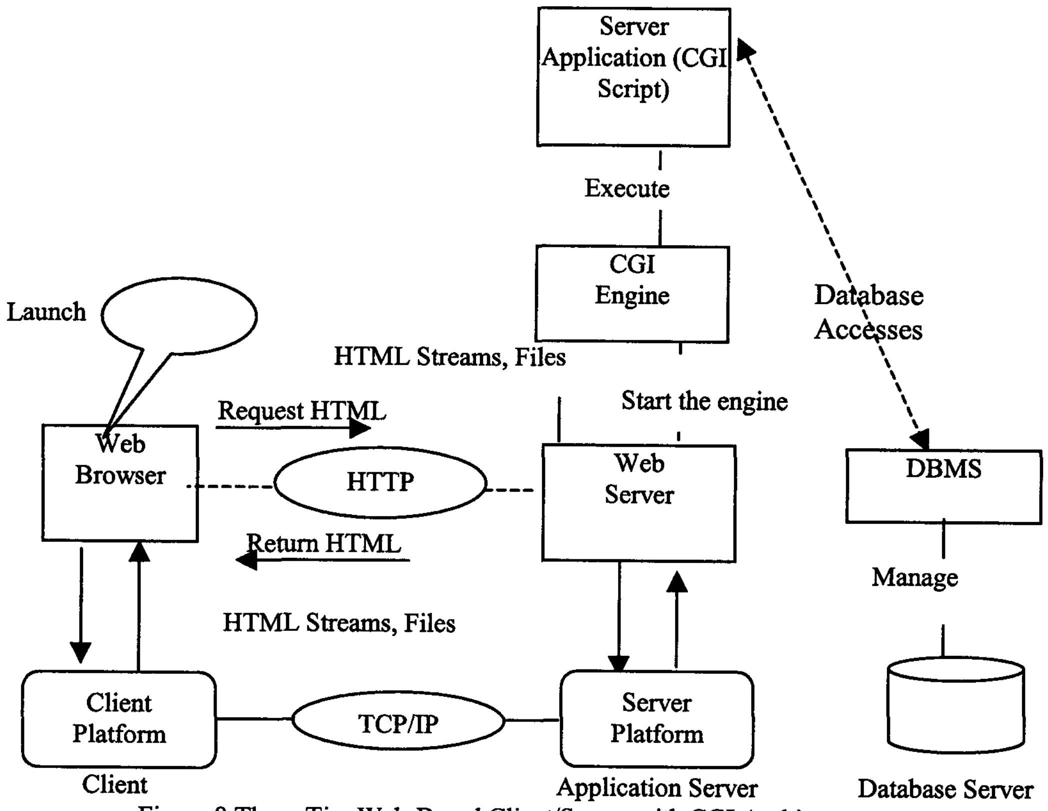

Figure 8 Three-Tier Web-Based Client/Server with CGI Architecture

The three-tier Client/Server model discussed above was applied on two platforms. The database server was local to the application server. To have a real distributed three-tier Client/Server system it is necessary to distribute the database server so that it is located on an independent machine in the network. This improves scalability and robustness and allows the server application to access multiple databases.

To distribute a database server database, database middleware is required on the application server machine. It plays the role as a translator between the database server and its clients. A distributed three-tier Web-based Client/Server system will now be considered by combining the Client/Server technologies that have been introduced in the previous sections. Its model is a distributed three-tier Web-based system with ASP technology. In this example Oracle database is the database server (See Figure 10).

From Figure 10, we can see that the first tier (client) and second tier (application server) are almost identical to the three-tiered architecture with CGI shown in Figure 8. The major difference is that the database middleware needs to be installed on the application server machine. SQL*Net is also installed on the third tier or database server machine. SQL*Net distributes the workload associated with databases, and supports queries and updates against remote databases [4]. SQL*Net is also the basic communication protocol between the Oracle database and the application server database middleware. SQL or ODBC is used as API for the application server programs to access the database server. TCP/IP is the network communication protocol between the application server and database server. Notice in Figure 10 how the application server-tier serves Web-based client requests by, in turn, being a client to middleware-based database services.

The simulated online book order case study is a simple problem. As we showed earlier in this section the case study could be designed and implemented by using much less complex Client/Server technologies. The above design diagram of distributed three-tier Client/Server models offers solution for much more complex e-commerce applications. It actually is overqualified for this case study. There are more comprehensive designs could be used to implement more complicated online e-commerce project, such as by using CORBA technology on all of the three tiers.

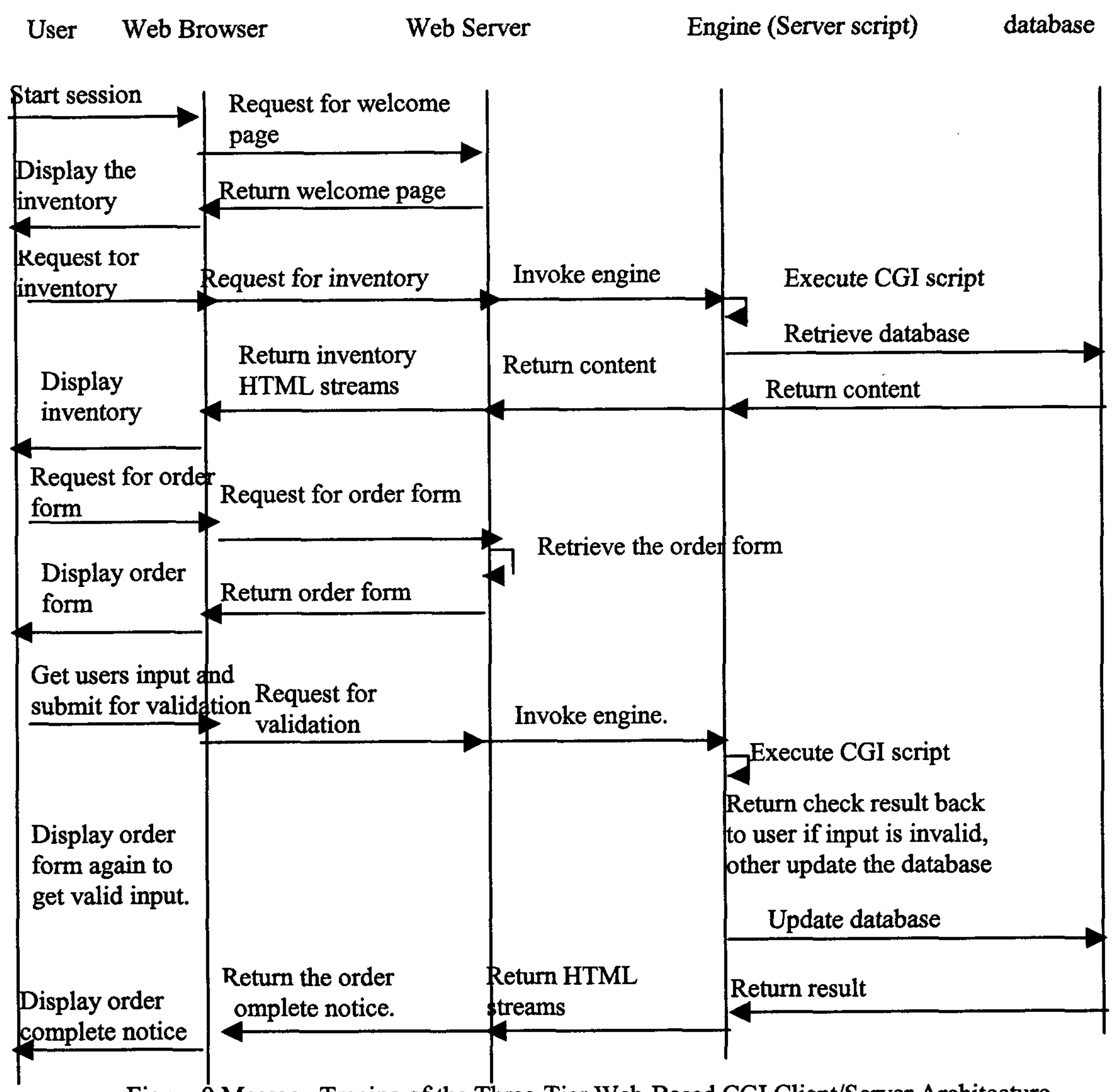

Figure 9 Message Tracing of the Three-Tier Web-Based CGI Client/Server Architecture

The motivation for this presentation is to show that this complex structure can be easily scaled up to accommodate much more complex applications, such as changing existing database server or adding more database servers.

4. Conclusions

In this work an overview of the current major Client/Server technologies is given. These technologies are illustrated by using a simulated online e-commerce book order system. The methodology employed was through the use of diagrams to give a visual view of the Client/Server structures. Message flow was traced in this simulated model for selected Client/Server technologies. This paper incrementally introduced various Client/Server technologies, including the basic Client/Server model, middleware, Web-based with applets, and distributed three-tier model. All of these technologies were brought together in complicated, yet understandable Client/Server architecture. The last comprehensive design, the Web-based three-tiered distributed Client/Server design was accomplished by separating client user interface, business logic, and database across the network. Each tier is isolated from the rest of the application system. This architecture and the technologies they integrate make application development, modification, and updating easier and make application scalability and robustness possible.

It is anticipated that the study of these important Client/Server technologies will be useful to the researchers

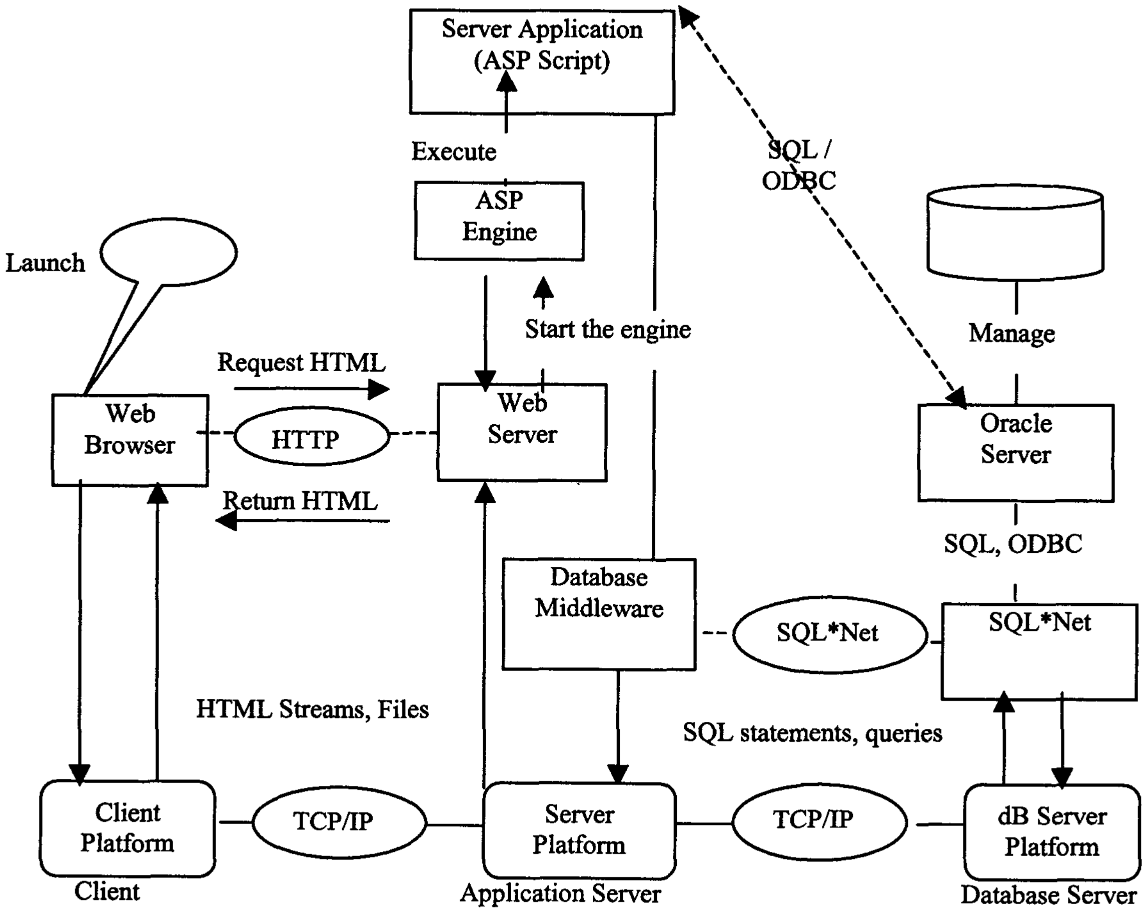

Figure 10 Distributed Three-Tier Web-Based with ASP Technology and Oracle Database Server Client/Server Architecture.

as well as the practitioners. It should provide a useful instrument to help developers choose the appropriate technologies and development tools.

References

[1] Abhijit Chaudbury, H. Raghav Rao. Introducing Client/Server Technologies in Information Systems Curricula. *The DATA BASE for Advances in Information Systems*, Fall 1997 Vol. 28, No. 4.

[2] Jeremy Rosenberger. *Teach Yourself CORBA in 14 days.* SAMs Publishing, 1998.

[3] Bill Hatfield. *Active Server Pages for Dummies.* IDG BOOKS, 1998.

[4] Kevin Loney. *ORACLE DBA Handbook 7.3 Edition.* Osborne McGraw-Hill, 1997.

[5] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The World-Wide Web. *Communication of The ACM*, August 1994, Vol.37, No.8.

[6] David S. Linthicum. *David Linthicum's guide to Client/Server and Intranet Development.* John Wiley & Sons, Inc, 1997.

[7] Jeri Edwards and Deborah Devoe. *3-Tier Client/Server At Work.* Wiley Publishing, 1997.

[8] David M. Kroenke. *Database Processing.* Sixth Edition, Prentice Hall, 1998.

[9] Cliff Berg. How do I Use Java Remote Method Invocation From An Applet. *Dr. Dobb's Journal*, March 1997.

[10] Tom Albertson. Distributed object application development: The Java-CORBA solution. *Tech focus*, March, 1998.

[11] Gopalan Suresh Raj. A Detailed Comparison of CORBA, DCOM and Java/RMI. http://www.execpc.com/~gopalan/misc/compare.html

[12] Ron Vetter. Web-based Enterprise Computing. *Computer* May 1999, Vol.32, No.5.
[13] Visibroker for C++ Reference. Version 3.3 white paper.
[14] Karl Moss. *Java Servlets*. Mc Graw Hill Publishing, 1998.