Client – Server Architecture: An Overview

D. Njoku,[1] K.I. Nkuma-Udah[2] and C.G. Onwugbufor[3]

[1]Department of Computer Science, Imo State Polytechnic, Umuagwo, Imo State, Nigeria
[2]Department of Biomedical Technology, Federal University of Technology, Owerri, Nigeria
[3]Department of ICT, Imo State Polytechnic, Umuagwo, Imo State, Nigeria

(Received October 15, 2013; Revised December 01, 2013; Accepted December 10, 2013)

Client-server architecture (client/server) is a network architecture in which each computer or process on the network is either a *client* or a *server*. Servers are computers or processes dedicated to managing disk drives (*file servers*), printers (*print servers*), or network traffic (*network servers*). Clients are personal computers (PCs) or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

The most basic form of the client/server architecture involves two computers: one computer, the server, is responsible for storing some sort of data and handing it to the other computer, the client, for user interaction. The user can modify that data and save it back to the server. The most popular form of a client-server architecture is the web system.

The client/server architecture assigns processing responsibility where it logically belongs. A simple system can be broken into two layers: a server where data and common processing occurs and a client where user-specific processing occurs. This kind of architecture is more commonly known as a *two-tier architecture*. The other architecture is the three-tier client/server architecture.

A client/server application is a piece of software that runs on a client computer and makes requests to a remote server. Many such applications are written in high-level visual programming languages where UI, forms, and most business logic reside in the client application. Specific types of client applications include: web browsers, E-mail clients, and online chat clients. And specific types of server applications include: web servers, FTP servers, database servers and E-mail servers.

Key words: Client, Server, Overview, Architecture, Computer, Tier

1. Introduction

Client-server architecture (client/server) is a network architecture [1] in which each computer or process on the network is either a *client* or a *server*. Servers are computers or processes dedicated to managing disk drives (*file servers*), printers (*print servers*), or network traffic (*network servers*). Clients are personal computers (PCs) or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

The Client/server architecture approach introduced replacement of file server by database server. User queries could be answered directly by using a relational database management system. The client/server architecture significantly decreased network traffic by providing a query response rather than total file transfer. It allows multi-user updating through a GUI front end to a shared database. Remote Procedure Calls (RPCs) or standard query language (SQL) statements are typically used to communicate between the client and server.

The Client/Server architecture is based on hardware and software components that interacts to form a system. This system includes three main components: Clients, Servers and Communication middleware [2]. Usually, the client needs to know of

*Corresponding author email: domokey08@yahoo.com

the existence and the address of the server, but the server does not need to know the address or even the existence of the client prior to the connection being

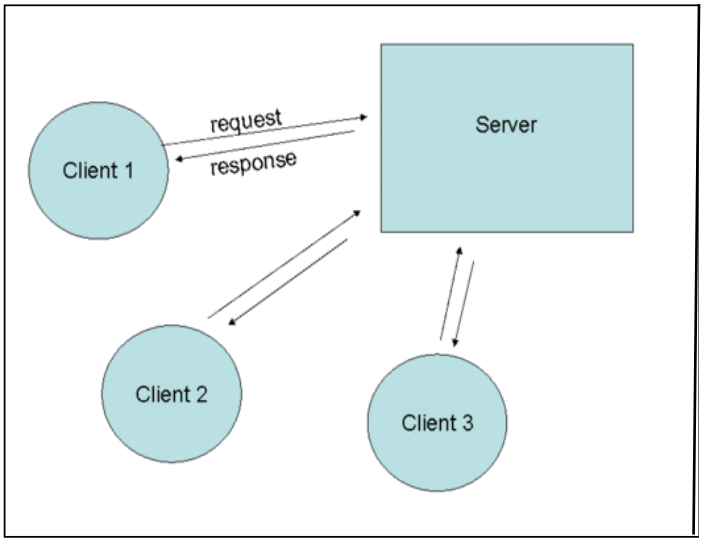

Figure 1: Client—Server Architecture

established. Once a connection is established, both sides can send and receive information.

1.1 The Client

The client is any computer or process [3] that requests services from the server. It is also known as the front-end- application. These include applications: that run on computers and rely on servers for files, devices and processing power. An example of a client is the e-mail client, which is an application that enables you to send and receive e-mail and PCs with MS Windows operating system.

1.2 The Server:

The server is any computer process providing services to the clients. It is also known as the back-end application, reflecting the fact that the server process provides the background services for the client process. Examples of server are computers or processes that manage network resources such as disk drives (file servers), printers (print servers) and network traffic (network servers)

Most of the Net Applications use the Client Server architecture [4]. These terms refer to the two processes which will be communicating with each other to exchange some information. One of the two processes acts as a client process and another process acts as a server.

2.0 Basic Client/Server Architecture

The most basic form [5] of the client/server architecture involves two computers: one computer, the server, is responsible for storing some sort of data and handing it to the other computer, the client, for user interaction. The user can modify that data and save it back to the server. The most popular form of a client-server architecture is the web system.

The Web implements this simple form of client/server architecture for multiple client machines. Your computer, the client, uses a Web browser to display HTML documents stored across the Internet on a Web server. There are four software components to the Web system:

- A browser such as Netscape that displays HTML documents on a client machine.
- A server program running on the server that hands HTML documents to client browsers.
- The HTML documents stored on the server machine.
- The communications protocol that handles the communication of data between the client and server.

Figure 3 shows how this architecture fits together.

The client/server architecture provides one with a logical breakdown of application processing. In an ideal environment, the server side of the application handles all common processing, and the client side handles user-specific processing. With the Web, the server stores the HTML documents that are shown to all the clients. Each cli-

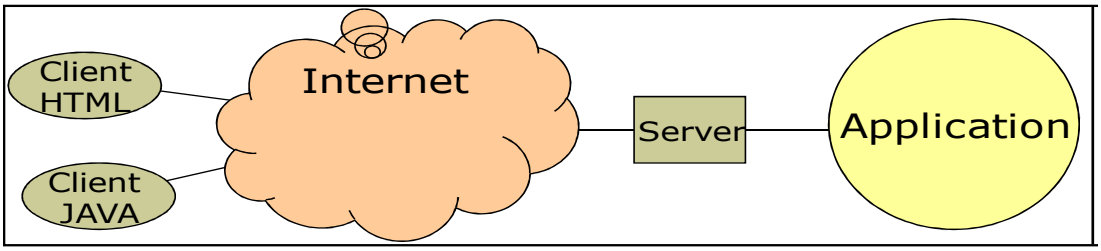

Figure 2: Applications of Client — Server

ent, on the other hand, has different display needs. For example, a user at a dummy terminal is limited to using the character-mode Lynx client. A Windows user, on the other hand, can use a GUI browser to use the power of the graphical interface to display the document using full multimedia. The presentation of the HTML documents presented by the server is thus left up to the client.

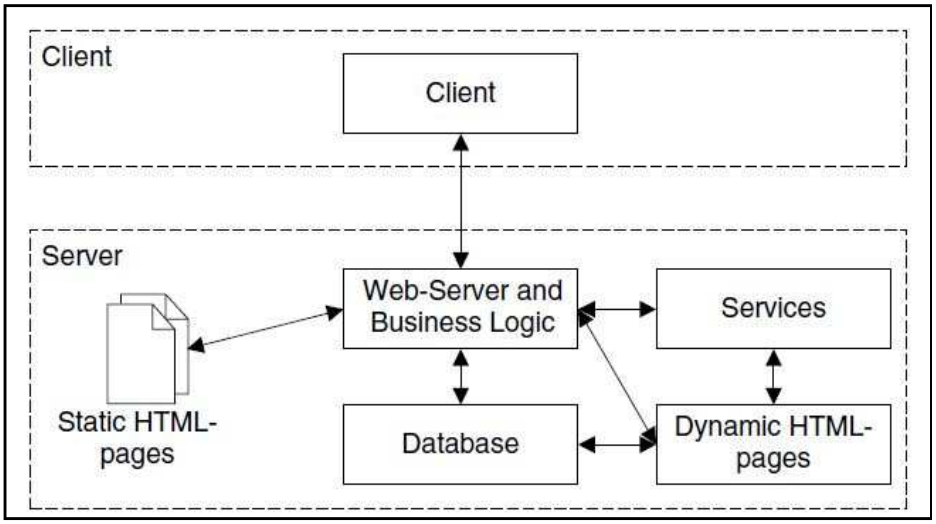

Figure 3: The client/server architecture of the Web

2.1 Client/Server Communication

Internet applications communicate using Internet Protocol (IP) sockets [6]. IP is a basic networking protocol on top of which other protocols exist to serve varying purposes. Each computer in an IP network has an *IP address*, which is a 32-bit number usually broken into four 8-bit quads. An IP address looks like this: 206.11.201.18. The first two numbers (the high-order bits) form the network address. The low-order bits specify which computer on that network the address is for. Using this multinetwork addressing scheme, computers on different networks can communicate with each other.

When one wants to send some information from his/her machine to another's, the sender's application uses the receiver's machine's IP address to send that data. If the receiver's machine has the address 199.199.181.120, for example, the sender's machine first checks whether it knows where that specific IP address is. Because the sender's machine is a simple client on the 206.11 network, it is very unlikely that it has any idea where the receiver's machine is. But it *does* know of a *default gateway* machine to which it sends data for all unknown computers.

When a gateway receives data addressed for a specific IP, it in turn checks to see whether it knows about the specific computer in question. In this case, the sender's default gateway is likely a router for his local network. It probably knows about the existence of machines only on the local network. It thus forwards his data onto its default gateway, which is responsible for knowing about a lot of networks. Although this router also does not know where the 199.199.181.120 machine is, it does have a specific gateway for the 199.199 network. It therefore forwards the data to that gateway. After traveling through a series of gateways, the data eventually reaches a machine that knows exactly where the receiver's machine can be found.

On any given machine, one may have a lot of applications communicating with other computers on the network. The packet sent has to be able to tell the receiver's machine exactly which application it is destined for. It does this using the final piece to IP addressing: the port number. Just as an apartment number tells the post office which apartment in a building a specific letter should be sent to, a port number tells a computer which application should receive an IP packet.

2.2 Internet Protocol

However, all one really needs to know about IP specifically is how to address the data he/she wants to send [6]. In fact, the port number just described is not even part of the IP. IP simply describes how to get a packet from point A to point B. Any network application written will instead be coded against higher level protocols which, in turn, handle IP management. The two most common protocols are: TCP/IP (Transmission Control Protocol/Internet Protocol) and UDP/IP (User Datagram Protocol/Internet Protocol)

TCP/IP, referred to in older texts as DARPA

Internet Protocols, is actually a suite of protocols that provides applications with a reliable data communication layer. When you send a TCP/IP packet, you know either that the packet will reach its destination or that you will be informed of any problems with the transmission of the data. All this is done by encapsulating packet transmission inside a network session. When you want to communicate with another computer using TCP/IP, you create a connection that allows you to send multiple packets. The target application is listening to the network on a target port. Your client application connects to that listen port and negotiates a new private port through which data can be transmitted for as long as the connection is open.

The downside to TCP/IP is the overhead required to manage all that error handling as well as the need to take up ports on both machines to maintain a constant connection. UDP/IP, on the other hand, is a protocol for transmitting packets across the network without the reliability overhead incurred by TCP/IP. If you use UDP/IP, your application sends individual packets (called *datagrams*) to the target computer's listen port and hopes for the best. Sometimes the packets get to their destination, sometimes they do not.

3.0 Types of Client/Server Architecture

The client/server architecture assigns processing responsibility where it logically belongs [7]. A simple system can be broken into two layers: a server where data and common processing occurs and a client where user-specific processing occurs. This kind of architecture is more commonly known as a *two-tier architecture*. However, for much more complex applications and systems, the client absorbs an inordinate amount of the system's processing needs leading to a *fat client*. Although fat client architecture is as capable as any other client/server configuration, it is harder to scale as applications grow over time. The solution to the problem of the fat client is a three-tier client/server architecture that creates another layer of processing across the network.

3.1 Two-tier Architecture

In this architecture, client directly interact with the server. This type of architecture may have some security holes and performance problems. Internet Explorer and Web Server work on two tier architecture. Here security problems are resolved using Secure Socket Layer (SSL). In two tier client/server architectures, the user interface is placed at user's desktop environment and the database management system services are usually in a server that is a more powerful machine that provides services to the many clients. Information processing is split between the user system interface environment and the database management server environment. The database management server supports for stored procedures and triggers. Software vendors provide tools to simplify development of applications for the two tier client/server architecture.

Two-tier is the simplest of the architecture types, consisting of only the server and the client application. The server, also known as the database, houses the information of a network, while the client requests to access the information. In a two-tier model, this request is direct, thus easy to develop and maintain. Though this architecture type is more common, it is usually only implemented in smaller, less demanding conditions.

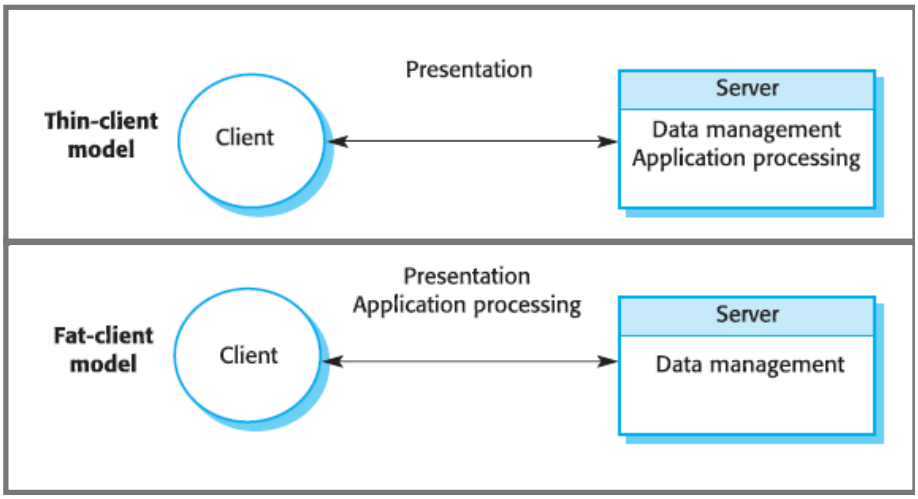

Figure 4: Two-tier Client-Server Architecture

In the two-tier, the two application layers are mapped onto two computer systems – the client and the server. The Client can be: thin client or fat client. For thin-client model, all of the application processing and data management is carried out on the server. The client is simply responsible for running the presentation software. However, in the case of fat-client model, the server is only responsible for data management. The software on the client implements the application logic and the interactions with the system user.

3.2 Three-tier Architecture

In this architecture, one more software sits in between client and server. This middle software is called middleware. Middleware are used to perform all the security checks and load balancing in case of heavy load [8]. A middleware takes all requests from the client and after doing required authentication it passes that request to the server. Then server does required processing and sends response back to the middleware and finally middleware passes this response back to the client. If you want to implement a 3-tier architecture then you can keep any middle ware like Web Logic or WebSphere software in between your Web Server and Web Browsers.

The three tier architecture is introduced to overcome the drawbacks of the two tier architecture. In the three tier architecture, a middleware is used between the user system interface client environment and the database management server environment [9]. These middleware are implemented in a variety of ways such as transaction processing monitors, message servers or application servers. The middleware perform the function of queuing, application execution and database staging. In addition the middleware adds scheduling and prioritization for work in progress.

The three tier client/server architecture is used to improve performance for large number of users and also improves flexibility when compared to the two tier approach [10]. However, a drawback of three tier architectures is that the development environment is more difficult to use than the development of two tier applications.

4. Applications of Client - Server

A client/server application is a piece of software that runs on a client computer and makes requests to a remote server [11]. Many such applications are written in high-level visual programming languages where UI, forms, and most business logic reside in the client application. Often such applications are database applications that make database queries to a remote central database server.

Specific types of client applications include: web browsers, E-mail clients, and online chat clients. And specific types of server applications include: web servers, FTP servers, database servers and E-mail servers. Most web services are also types of servers.

4.1 Client Applications

Web Browsers

A web browser is a computer program application found on all modern computers [12]. They are also recently a common feature of many mobile devices and tablet computers like the iPad or Android tablets. Web browsers are used by people to find and look at web sites on the Internet. The first web browser was created in 1990. Many different web browsers are available for free.

All web browsers can go to websites but each browser has good

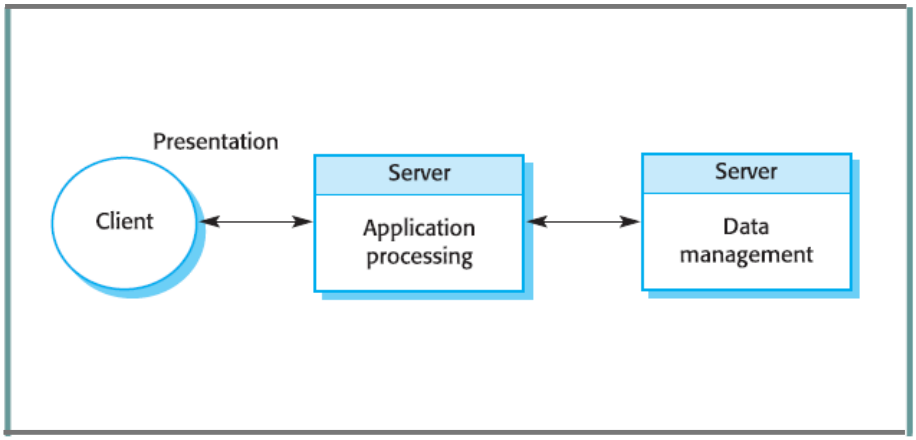

Figure 5: Three-tier Client-Server Architecture

things and bad things about it. For example, some browsers focus on data security and keeping computers safe from viruses. Other browsers are made so that web pages appear on screen faster. Some popular web browsers available to download include: Camino, Internet Explorer, Google Chrome, Mozilla Firefox, Opera, Safari

E-mail clients

Electronic mail (or e-mail or email) is an Internet service that allows those people who have an e-mail address (accounts) to send and receive electronic letters [12]. Those are much like postal letters, except that they are delivered much faster than snail mail when sending over long distances, and are usually free.

Like with regular mail, users may get a lot of unwanted mail. With e-mail, this is called spam. Some programs used for sending and receiving mail can detect spam and filter it out nearly completely.

To send or receive an email in the traditional way, you need a device (computer, phone etc.) connected to the Internet and an e-mail program (simply called mailer). Several formats exist for email addresses. The most common, called RFC 2822, looks like user@domain.com. E-mail messages are sent mostly by text, and sometimes by HTML style.

Some companies let you send and receive emails for free from a remote website. Gmail, Hotmail and Yahoo! are among the many that do this kind of service, which is known as "web mail". Microsoft invented its own "communication protocol" (or set of rules) for sending and receiving mail, called "Exchange". Exchange protocol works entirely differently from the traditional method.

Online chat clients

An online chat client is a software program designed for use in online chat rooms [13]. It might also be associated with Instant Messaging (IM) programs, though IM programs are designed to use while cruising the Web. This makes standard IM programs more akin to a plug-in than a fully featured chat client. What IM programs and a chat client have in common is that they both allow real-time communication between people who are online at the same time. As soon as a participant types a message and hits “send” in a chat client or IM program, the recipient sees the reply inside his or her own chat client or IM program.

However, similarities end there. While an instant message communication is usually between two people who are cruising the Web, a chat client makes use of a separate network designed to allow *numerous* people to talk to each other in real time. Using a chat client, a participant can discuss one of countless subjects with dozens or even hundreds of other people.

Various “chat rooms,” each with their own subject matter, are located at unique network addresses. The user chooses what subject she or he would like to discuss, then joins the chat room. Chat rooms operate 24/7, and at any given time, a particular chat room might be jumping with hundreds of participants or virtually empty.

4.2 Server applications

Web servers

A web server is a computer with a boot device or other disk containing a web site [11]. The term may also refer to an application that helps a computer to perform web server functions. If one site runs more than one server they must use different port numbers. Alternatively, several hostnames may be mapped to the same computer in which case they are known as "virtual servers".

Apache, IIS, and nginx are three popular web servers. There are many others including some for practically every platform. Servers differ mostly in the "server-side" features they offer such as server-side includes, and in their authentication and access control mechanisms. All decent servers support CGI and most have some binary API as well.

FTP Server

An FTP server is a software.html application running the File Transfer Protocol (FTP) [14,15]. FTP is a standard network protocol used to transfer computer files from one host to another host over a TCP-based network, such as the Internet. FTP is built on a client-server architecture and uses separate control

and data connections between the client and the server [16]. FTP users may authenticate themselves using a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS). SSH File Transfer Protocol (SFTP) is sometimes also used instead, but is technologically different.

The first FTP client applications were command-line applications developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix, and Linux operating systems [17,18]. Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into productivity applications, such as Web page editors.

Database servers

A database server is a computer program that provides database services to other computer programs or computers, as defined by the client–server model [15]. The term may also refer to a computer dedicated to running such a program. Database management systems frequently provide database server functionality, and some DBMSs (e.g., MySQL) rely exclusively on the client–server model for database access.

Most of the Database servers works with the base of Query language. Each Database understands its query language and converts it to Server readable form and executes it to retrieve the results.

Some examples of proprietary database servers are Oracle, DB2, Informix, and Microsoft SQL Server. Examples of GNU General Public Licence database servers are Ingres and MySQL. Every server uses its own query logic and structure. The SQL query language is more or less the same in all relational database servers. DB-Engines lists over 200 DBMSs in its ranking.

E-mail servers

An e-mail server (also known as a mail transfer agent or MTA, a mail transport agent, a mail router or an Internet mailer) is an application that receives incoming e-mail from local users (people within the same domain) and remote senders and forwards outgoing e-mail for delivery [15]. It is a computer within a network that works as a virtual post office. It usually consists of a storage area where where e-mail is stored for local users, a set of user definable rules which determine how the mail server should react to the destination of a specific message, a database of user accounts that the mail server recognizes and will deal with locally, and communications modules which are the components that actually handle the transfer of messages to and from other mail servers and email clients.

Generally the person(s) responsible for the maintenance of the e-mail server (editing users, monitoring system activity) are referred to as the postmaster. Most mail servers are designed to operate without any manual intervention during normal operation.

An e-mail server is the computerized equivalent of your friendly neighborhood mailman. Every email that is sent passes through a series of mail servers along its way to its intended recipient. Although it may seem like a message is sent instantly - zipping from one PC to another in the blink of an eye - the reality is that a complex series of transfers takes place. Without this series of mail servers, you would only be able to send emails to people whose email address domains matched your own - i.e., you could only send messages from one example.com account to another example.com account.

Mail servers can be divided into two main categories: outgoing mail servers and incoming mail servers. Outgoing mail servers are known as SMTP, or Simple Mail Transfer Protocol, servers. Incoming mail servers come in two main varieties. POP3, or Post Office Protocol, version 3, servers are best known for storing sent and received messages on PCs' local hard drives. IMAP, or Internet Message Access Protocol, servers always store copies of messages on servers. Most POP3 servers can store messages on servers, too, which is a lot more convenient.

References

[1] Omran A. Bukhres and Ahmed K. Elmagarmid, Object-Oriented Multidatabase Systems, 1996: A Solution for Advanced Applications, Prentice Hall.

[2] David J. DeWitt, Philippe Futtersack, David Maier, and Fernando Velez, 1990, “A Study of Three Alternative Workstation-Server Architectures for Object Oriented Database Systems”, Proceedings of the 16th VLDB Conference, Brisbane, Australia, pp. 107-121.

[3] Nieh, Jason; Novik, Naomi &., Yang, S. Jae (December 2005). "A Comparison of Thin-Client Computing Architectures". *Technical Report CUCS-022-00* (New York: Network Computing Laboratory, Columbia University). Retrieved November 11, 2011.

[4] "Distributed Application Architecture". Sun Microsystem. Retrieved 2009-06-16.

[5] Benatallah, B.; Casati, F.; Toumani, F. (2004). "Web service conversation modeling: A cornerstone for e-business automation". *IEEE Internet Computing* **8**: 46. doi:10.1109/MIC.2004.1260703.

[6] Dustdar, S.; Schreiner, W. (2005). "A survey on web services composition". *International Journal of Web and Grid Services* **1**: 1. doi:10.1504/IJWGS.2005.007545.

[7] Rulifson, Jeff (June 1969). *DEL*. IETF. RFC 5. Retrieved 30 November 2013.

[8] Shapiro, Elmer B. (March 1969). *Network Timetable*. IETF. RFC 4. Retrieved 30 November 2013.

[9] Sturgis, Howard E.; Mitchell, James George; Israel, Jay E. (1978). "Separating Data from Function in a Distributed File System". Xerox PARC. Missing or empty |title= (help) Harper, Douglas. "server". *Online Etymology Dictionary*. Retrieved 30 November 2013.

[10]"Separating data from function in a distributed file system". *GetInfo*. German National Library of Science and Technology. Retrieved 29 November 2013.

[11]Nieh, Jason; Novik, Naomi; Yang, S. Jae (December 2005). "A Comparison of Thin-Client Computing Architectures" (PDF). *Technical Report CUCS-022-00* (New York: Network Computing Laboratory, Columbia University). Retrieved 30 November 2013.

[12]Tolia, Niraj; Andersen, David G.; Satyanarayanan, M. (March 2006). "Quantifying Interactive User Experience on Thin Clients" (PDF). *Computer* (IEEE Computer Society) **39** (3).

[13]d'Amore, M. J.; Oberst, D. J. (1983). "Microcomputers and mainframes". "Proceedings of the 11th annual ACM SIGUCCS conference on User services - SIGUCCS '83". p. 7. doi:10.1145/800041.801417. ISBN 0897911164.

[14]Otey, Michael (22 March 2011). "Is the Cloud Really Just the Return of Mainframe Computing?". *SQL Server Pro*. Penton Media. Retrieved 1 December 2013.

[15]Barros, A. P.; Dumas, M. (2006). "The Rise of Web Service Ecosystems". *IT Professional* **8** (5): 31. doi:10.1109/MITP.2006.123.

[16]Yongsheng, H.; Xiaoyu, T.; Zhongbin, T. (2013). "An Optimization Model for the Interconnection among Peers of the P2P Network". *Journal of Applied Sciences* **13** (5): 700. doi:10.3923/jas.2013.700.707.

[17]Forouzan, B.A. (2000). TCP/IP: Protocol Suite. 1st ed. New Delhi, India: Tata McGraw-Hill Publishing Company Limited.

[18]Kozierok, Charles M. (2005). "The TCP/IP Guide v3.0". Tcpiguide.com.