INTERNATIONAL JOURNAL OF RESEARCH AND REVIEWS IN SOCIAL AND APPLIED SCIENCES

ISSN: 2931 – 7853 | www.caarnjournals.com | Volume 9. Issue 1. (September 2022)

NETWORK ARCHITECTURE: TIERS OF CLIENT SERVER ARCHITECTURES

BENJAMIN, UBOKOBONG EFFIONG
Department of Computer Science,
Akwa Ibom State Polytechnic,
Ikot Osurua, Ikot Ekpene
E-mail: benjaminubokobong52@gmail.com

EKPO, EDET GEORGE
Department of Electrical/Electronic Engineering Technology,
Akwa Ibom State Polytechnic, Ikot Osurua, Akwa Ibom State, NIGERIA
Email: georgeekpo@gmail.com

ABSTRACT

This paper examined network architecture: tiers of client server architectures. Network architecture has to do with the design of a network. The network architecture determines the physical components and the configuration to be used. Client-Server architecture is an architectural deployment style that describes the separation of functionality into layers with each segment being a tier that can be located on a physically separate computer. They evolved through the component-oriented approach, generally using platform specific methods for communication instead of a message-based approach. This architecture has different usages with different applications. It can be used in web applications and distributed applications. The two tier and N-tier client/server architectures are discussed and the criteria for selecting a given network architecture is also examined in this paper.

Keywords: Client, Server, Network, Architecture, Browser

INTRODUCTION

Network architecture is the design of a computer network. It is a framework for the specification of a network's physical components and their functional organization and configuration, its operational principles and procedures, as well as communication protocols used. In telecommunication, the specification of a network architecture may also include a detailed description of products and services delivered via a communications network, as well as detailed rate and billing structures under which services are compensated.

Client-Server architecture is an architectural deployment style that describe the separation of functionality into layers with each segment being a tier that can be located on a physically separate computer. They evolved through the component-oriented approach, generally using

ISSN: 2931 – 7853 | www.caarnjournals.com | Volume 9. Issue 1. (September 2022)

platform specific methods for communication instead of a message-based approach. This architecture has different usages with different applications. It can be used in web applications and distributed applications (Kratky & Reichenberger, 2013).
The strength in particular is when using this architecture over distributed systems. In this paper, the tiers of client-server architecture are discussed.

Types of Network Architecture

The client server architectures are of three types
1. Two-tier Architecture
3. Three-tier (N-tier) Architecture

Two-tier Architecture

A traditional client/server application consists of two tiers: the client tier and the server tier. The client provides the user interface and typically runs on PCs and other desktop machines that are connected to a LAN. The server stores shared data and can run on anything from a PC to a mainframe. The two-tier client/server approach has two options: place the business logic on the client (fat client) or place the business logic on the database server (fat server). Both of these options have potential drawbacks. When business logic is placed on the client in the form of application code, the code will have to be changed each time the business logic changes. The revised application code must then be recompiled and distributed to all of the workstations; this can be time consuming and inefficient. Another disadvantage of fat clients is the increased potential for poor response time. Structured Query Language (SQL) commands must be used to query, insert, update, or delete records in a relational database. Placing the business logic on the client requires sending large blocks of SQL to the server for processing; the increased network traffic can result in poor performance - especially if the database is located remotely and the client must send SQL over the Wide Area Network (Linthicum, 2017).
Another drawback of having SQL code embedded in the client application is that the front-end developers must have intimate knowledge of the structure and contents of the relational database. In a typical organization, there are several core databases, each having thousands of tables; it often takes a new developer a few months to learn the tables and columns appropriate to use for different applications. To remedy some of these problems, the business logic can be placed on the database server in the form of triggers and stored procedures. When the business logic is located on the database server, the SQL is physically located in the same place as the data it manipulates, thus preventing the need to send large blocks of SQL over the network. The client application calls a database procedure, passing it any necessary parameters, and the result is then passed back to the client application. A benefit of fat server is that it helps centralize business logic (Sovik and Sandman, 2017).

ISSN: 2931 – 7853 | www.caarnjournals.com | Volume 9. Issue 1. (September 2022)

Two-tier client/server applications do not scale well to support thousands of users. The database servers simply can't manage thousands of client connections. The host operating system tends to thrash and die when you scale to high user loads. This is because of the one-client, one-connection requirement of two tier client/server. In two-tier client/server, each client application running has its own thread to the database that uses host operating system resources on the server. Thousands of client connections would require high levels of system resources. Connectivity above a certain threshold will result in poor performance and perhaps crash the system (Davis, 2016).

Three Tier (N-tier) Client Server Network Architecture

The three client/server network architecture is divided into 3 different tiers which are: Presentation tier, Logic tier, and Data tier. Each tier is developed and maintained as an independent tier.

1. **Presentation Tier**: This is the topmost level of the application. The presentation layer provides the application's user interface (UI). Typically, this involves the use of Graphical User Interface for smart client interaction, and Web based technologies for browser-based interaction. The presentation tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network (Dewire, 2013). In the presentation layer the user interaction takes place as defined before. The user enters the address in the web browser and in the browser the URL is decoded into protocol/host/file, i.e. host name converted to IP address. Then an issue request is sent to remote server using appropriate protocol (usually HTTP). Also a returned HTML from the logic tier might be accepted. In the presentation layer interaction with client side scripts (e.g. using DHTML) is supported and user inputer of variety controls on the form are accepted (Dewire, 2013).
2. **Logic Tier:** This tier of network architecture is also known as **business logic, data access tier, or middle tier.** The logic tier is pulled out from the presentation tier and, as its own layer; it controls an application's functionality by performing detailed processing. Logic tier is where mission-critical business problems are solved. The components that make up this layer can exist on a server machine, to assist in resource sharing. These components can be used to enforce business rules, such as business algorithms and legal or governmental regulations, and data rules, which are designed to keep the data structures consistent within either specific or multiple databases. Because these middle-tier components are not tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules require. For example, simple edits can be placed on the client side to minimize network round-trips, or data rules can be placed in stored procedures (Fong, & Hiu, 2009). In the logic Tier the application's functionality is done by performing detailed processing of data from presentation layer. Server such as Appache (or IIS) or

ISSN: 2931 – 7853 | www.caarnjournals.com | Volume 9. Issue 1. (September 2022)

Server Script (such as PHP) can be used to support this. With Server (Apache or IIS) the appropriate action to be taken is identified, such as fetching a file, or passing request to an interpreter. Also it sends an output back to caller in MIME package. As such support for thousands for concurrent users, multithreading (allow multiple processor to run concurrently) and caching (holding results in a temporary store to reduce recalculation) is achieved) (Sultan, 2010). With Server script (example in PHP) interacting with server such as accessing input or generating input is done. It interprets the requests according to business rules and past transactions from this client, and requests appropriate data from the persistence layer. It also computes the derived data and creates HTML (or GIF…) for the page (Dewire, 2013).

3. **Data Tier:** This tier consists of database servers, is the actual DBMS access layer. It can be accessed through the business services layer and on occasion by the user services layer. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance. This layer consists of data access components (rather than raw DBMS connections) to aid in resource sharing and to allow clients to be configured without installing the DBMS libraries and ODBC drivers on each client. An example would be a computer hosting a database management system (DBMS), such as a Microsoft SQL Server database.

The interaction with the database is done using standard languages such as SQL queries using database specific protocol over TCP/IP. The data structures (for example tables) are defined and modified themselves, that insertion, updating and deleting of data for example. Data maintenance should be maintained with backup and recovered. Access to compilation of queries should be optimized, with indexing or replication of tables. An example of technology using this would be .NET that is built into the .NET framework, as ADO.NET contains a mechanism to query data out of the database and return it to the caller in a connected or disconnected fashion.

Real life Application of Three Tier Network Architecture

Real life example of three tier client/server architecture would be in Emails done using 3 Tier Architecture. Reading e-mail using a Web-based interface, such as Hotmail, uses a three-tier architecture (Zhang, 2013). The three tiers are:

1. Presentation Layer: The client's web browser that sends HTTP requests to the Web server.
2. Logic Layer: The Web server:
    a. sends HTTP responses to the Web client
    b. Translates the client's HTTP requests into SMTP packets which are then sent to the Mail server.

ISSN: 2931 – 7853 | www.caarnjournals.com | Volume 9. Issue 1. (September 2022)

3. Data Layer: The Mail server performs the following functionality, and when performed it is transformed to the Logic Layer.

a) When completed, an e-mail message is sent by the sender's e-mail client as an SMTP packet to the local mail server.
b) The mail server's message transfer agent next reads the packet's destination address and sends it over the Internet to the receiver's mail server.
c) The destination mail transfer agent then stores the message in the receiver's mail box.
d) When the receiver next accesses e-mail, his or her user agent contacts the local mail server which then downloads the message to the receiver's client computer.

Components Interconnections of 3-Tier Client Server Architecture

The 3 tier network architecture is characterized by the functional decomposition of applications, service components, and their distributed deployment, providing improved scalability, availability, manageability, and resource utilization. During an application's life cycle, the three-tier approach provides benefits such as reusability, flexibility, manageability, maintainability, and scalability. Each tier is completely independent from all other tiers, except for those immediately above and below it. You can share and reuse the components and services you create, and you can distribute them across a network of computers as needed. You can divide large and complex projects into simpler projects and assign them to different programmers or programming teams. You can also deploy components and services on a server to help keep up with changes, and you can redeploy them as growth of the application's user base, data, and transaction volume increases (Oluwatosin, 2014). Logic layer is moved outside the presentation layer and into the business layer as it enhances reuse. As applications grow, applications often grow into other realms. Applications may start out as a web application, but some of the functionality may later be moved to a smart client application. Portions of an application may be split between a web site and a web or windows service that runs on a server. In addition, keeping logic helps aid in developing a good design (Oluwatosin, 2014).

Benefits of the 3-tier Client Server Network Architecture

The main benefits of the 3-tier architectural style are:

1. Maintainability. Because each tier is independent of the other tiers, updates or changes can be carried out without affecting the application as a whole.
2. Scalability. Because tiers are based on the deployment of layers, scaling out an application is reasonably straightforward.
3. Flexibility. Because each tier can be managed or scaled independently, flexibility is increased.
4. Availability. Applications can exploit the modular architecture of enabling systems using easily scalable components, which increases availability.

ISSN: 2931 – 7853 | www.caarnjournals.com | Volume 9. Issue 1. (September 2022)

Criteria for Deciding Client/Server Network Architecture

Deciding which architecture is best for an application is not an easy task. There are many options to choose from, many of which will work for any given application. There are many factors to consider and trade-offs to weigh. In general, a two-tier application is cheaper and faster to build but it does not provide as many robust features as an n-tier application.

Consider the 3-tier architectural style if the processing requirements of the layers in the application differ such that processing in one layer could absorb sufficient resources to slow the processing in other layers, or if the security requirements of the layers in the application differ. For example, the presentation layer should not store sensitive data, while this may be stored in the business and data layers. The 3-tier architectural style is also appropriate if you want to be able to share business logic between applications, and you have sufficient hardware to allocate the required number of servers to each tier (Renaud, 2013).

The following heuristics should be examined when making the choice between two-tier and n-tier client/server (Sovik and Sandman, 2017).

1. If the number of users exceeds 100 or the number of users is expected to grow to more than 100 during the life of the application, scalability issues will arise with a two-tier application.
2. If the application uses business logic that is common to other applications in the organization, it may be worth the investment to place the business logic in middle-tier processes so that it can be accessed by multiple applications.
3. According to the Gartner Group, if an application is complex, involving more than 50 distinct business processes, n-tier should be used. As applications become more complex, two-tier applications become exponentially more difficult to develop.
4. If the application is needed for a specific purpose for a short amount of time, it is not worth the investment of time and money to create an n-tier application. However, if the life of the application is expected to be more than three years, an n-tier approach should be used (Edwards, 2012).
5. If there is a plan to change to a different database in the future, it may be worth the investment to create an n-tier application, Modifying an n-tier application to a new database vendor is easier than modifying a two-tier application.
6. A mission-critical application needs to be continuously available and thus requires a comprehensive disaster recovery plan. Disaster recovery is much more complicated in a distributed n-tier environment.
7. An n-tier application can access multiple heterogeneous data sources and even provide two-phase commit controls across them. A two-tier application cannot support heterogeneous databases. If the application needs to access heterogeneous databases, n-tier is the only option (Edwards, 2012).

ISSN: 2931 – 7853 | www.caarnjournals.com | Volume 9. Issue 1. (September 2022)

8. If the application needs to be deployed urgently, a two-tier architecture will get the job done faster. N-tier applications are more complex and take longer to create. An n-tier application can take up to twice as long to create compared with a two-tier approach.
9. N-tier applications can be configured to have better response time than two-tier client/server applications. When databases are located remotely and the WAN time is significant, the n-tier application can be much faster than a two-tier application.
10. Applications with more than 50,000 transactions per day should be developed using an n-tier approach (Edwards, 2012).
11. The power of the middleware comes with a price. Usually, a consultant that specializes in the kind of middleware you want to implement will be needed for a few months to get things going. Employing an on-site consultant can be very expensive. The licenses for the middleware products are also expensive.

CONCLUSION

This paper has examined network architecture: tiers of client/server network architectures. The two types of client/server architecture are discussed. 2-tier architecture is used to describe client/server systems where the client requests resources and the server responds directly to the request, using its own resources. This means that the server does not call on another application in order to provide part of the service. In 3-tier architecture, there is an intermediary level, meaning the architecture is generally split up between a client, i.e. the computer, which requests the resources, equipped with a user interface (usually a web browser) for presentation purposes, the application server (also called middleware), whose task it is to provide the requested resources, but by calling on another server, the data server, which provides the application server with the data it requires. For usage that will require thousands of users 3-tier client/server network architecture have been proven to be better.

RECOMMENDATIONS

1. More research should be encouraged on network architecture.
2. Network engineers should be trained on the best network architecture to use depending on the number of users.

REFERENCES

Davis, P. (2016). *Securing client/server computer networks*. McGraw-Hill, USA

Dewire, D. T. (2013). *Client/server computing*. McGraw-Hill, Singapore

Edwards, J. (2012). *Three tier client server at work*. New York, NY: Wiley Computer Publishing.

Fong, J., & Hui, R. (2009). Application of middleware in the three tier client/server database design methodology. *Journal of the Brazilian Computer Society, 6*(1), 50-64.

ISSN: 2931 – 7853 | www.caarnjournals.com | Volume 9. Issue 1. (September 2022)

Kratky, S., & Reichenberger, C. (2013). Client/Server Development based on the Apple Event Object Model. *Atlanta*.

Linthicum, D. (2017) Moving to n-tier RAD. *DBMS*, 10(3), 3-8.

Oluwatosin, H. (2014) Client-Server Model. *IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. IX, PP 67-71*

Renaud, P. (2013) I*ntroduction to Client/Server Systems: A Practical Guide for Systems Professionals*, Wiley & Sons.

Sovik, J. and Sandman, T. (*2017)* Two-tier vs. N-tier client/server architectures: toward selection heuristics. *College of Business, CSUS pp. 642-644*

Sultan, N. (2010). Cloud computing for education: A new dawn? *International Journal of Information Management*, *30*(2), 109-116.

Zhang, H. (2013). Architecture of Network and Client-Server model. *arXiv preprint arXiv:1307.6665*.