1Lt Luis Sepulveda

CSCS 689

16 Jan 2020

Locations of forked code: https://github.com/LsepulvedaAFIT/AFIT-CSCE689-HW1-S

Homework 1 questions:

1. Based on what you've read so far, would you define this application as a distributed system? Why or why not? Use the design goals listed in Ch 1 to justify your position.

I would not classify this application, as it is currently running, as a distributed system. A few changes would have to be made in order define the application as a distributed system as described in chapter 1. First off, the server and client are only both running on the localhost, so it lacks the geographic distribution characteristic normally applied to a distributed system. The application would have to be on two different computers which are geographically separated to satisfy this fundamental condition. Secondly, looking at the “supporting resource sharing” goal, this application only supports the limited sharing of commands from the client to server and information from the server to client. To achieve this goal, the application must be enhanced to support sharing files, services, and storage facilities. Furthermore, this simple application also lacks the Interoperability and Portability design goals given that it was coded to primarily to work on a Linux Debian system. This goal could be achieved by creating a web installer that could remotely install the application on cross-platform systems.

On the positive side, the application communicates over TCP connects which theoretically allows it to be geographically scalable. Additionally, the simple user interface implementation helps hide the communication latencies by pausing the client for a second after sending commands. This allows the server messages to be displayed all at once on the client side. Lastly, I would argue this application is could be size scalable by continuously replicating the server as clients increase with size. With the help of the GitHub repository, the source code can be easily download by many different Linux servers and executed as necessary.

2. Based on your Chapter 2 reading, what architectural style(s) does your software leverage? Explain why it fits.

This application leverages the layered architecture model as it implements the TCP/IP stack for communication between the server and client with the use of socket programming. Additionally, the object-based architecture style is used when creating individual clients as they invoke method calls to the server triggering command executions. Lastly, the direct coordination model is established between to the server and client in order to properly exchange information.

3. Based on your Chapter 2 reading, what system architecture does your software leverage?

This application leverages the centralized system architecture by implementing the basic Client-Server model. The tcpserver provides the service and accepts tcpclient connections that request services. The request-reply behavior is used for communication between the two groups.

4. What steps would you need to take to "evolve" your code into the following types of systems. List at least three major tasks each:

A. A three-tiered architecture pulling data from an SQL database.

1.) Develop a SQL database that can be connected to the server

2.) Augment the server code to interface the SQL database through query commands. Implement a user interface with the client to allow the client to make plain text data request.

3.) Create an interpreter (middleware) in the server that can convert client requests into SQL query commands and ensure the proper data format can be interchanged between the client and database

B. A node of a peer-to-peer system (structured or unstructured).

1.) Develop binding code to merge the server and client objects, in order to create individual nodes. This will allow the node to act as a client and server at the same time.

2.) Develop file sharing processes that allows a node to search for a specific file stored by other nodes or super peers and initiate a copy transfer (broker the interaction).

3.) Implement a policy that dictates the rules and authentication method for joining the network to avoid unauthorized users (i.e. spam bots who introduce unwanted file to the network)

C. An edge-server system

1.) Create a content provider that distributes the content out the edge servers (tcpserver in our case)

2.) Augment the server code to allow connections to the content provider in addition to the client connections.

3.) Implement a procedure which allows for automatic updates of content between the edge server and the content provider.