Using Twisted as a Potential Framework for an Application Server Herd

Calvin Liu
CS 131 Programming Languages

1. Abstract

The purpose of this project is to try and make an application server herd through the Twisted framework and evaluate the efficiency and feasibility of this style of development.

2. Introduction

The way this project works is that a user can learn about the location of another user by asking the whereabouts of a 3rd party person. This 3rd person is used to relay the information, but we want to limit the number of transmissions without talking to a database.

Doing this project, it is easy to see that Twisted is a good framework to use for an application herd as the libraries allow people to use the protocols and functions and if extra functionality is needed then you can override functions to meet necessary specifications. Twisted gives us a framework that allows us to create a server and client based program who functionality can be limited through extra programming in python.

3. Twisted Approach

For the project, we have to meet the requirement of:

1. Alford talks with everybody but Bolden and Hamilton
2. Bolden talks with Parker and Powell
3. Hamilton talks with Parker

These specifications could be satisfied using 3 different protocols: IAMAT, WHATSAT and AT. The main python server that exists is chatserver.py. IAMAT tells the server where the client is. WHATSAT queries the server how much information you want to receive about a certain server within a certain distance. The AT response is used to tell the client information, but since there is flooding, we have to limit the amount of response messages each server gets based on who they are allowed to talk to or else the flooding with flood everyone with messages.

3.1. Type Checking

Since python is a dynamic type checking language, type errors will be checked at run time and any error will be reported. This is helpful in debugging in why a server might go down due to an error. This can be overcome by catching the error and doing whatever you want to do with that error.

3.2. Memory Management

Memory management is handled through python's garbage collector. The garbage collector is able to optimize the amount of memory being used, which is necessary when enacting something like an application server herd. The key to solving the problem of if there is a memory leak or a server needs more memory to handle the flooding then you can shut down one of the servers that has multiple edges going to it, leaving other edges the flooding can take to alert other servers and relay the location information.

3.3. Multi-threading

There is no multi-threading in Twisted, but there is an asynchronous programming style that uses event loops by using call back functions and allows spawning of multiple processes. This is not the best way to handle something that can be handled with multi-threading because it uses CPU resource consistently. If we were to go with a true multi-threading approach, then we can divide up the CPU resource in order to achieve parallel programming every time a new person makes a connection. With the Twisted approach, there is a also a problem of if there is a server error then the whole server will crash, but with a multi-thread approach, only a thread will crash. Even though we are not using multi-threading though, asynchronous and distributive programming is still achieved with Twisted.

3.4. Comparison to Node.js

Both Twisted and Node.js allow the use of event-driven networking framework and after some research it seems that Node.js is faster than Twisted. Other than that, the difference is apparent in that one uses javascript while python is used for Twisted.

4. Conclusion

In conclusion it is evident that Twisted is a goo framework to use for an application server herd. In mychatserver.py I tried to parse the command protocols based on the first argument and then execute the necessary functionalities. A dictionary was needed to save the information of each server that entered the “network” of servers.

5. Running

To run the code, run ./run.sh and then telnet to the corresponding port numbers. Unfortunately I was not able to make 100% of the code work so there might be some bugs, but it is evident that python can be used for creating an application server herd.

6. References

"[Introduction to Perspectives]." *The Modern Language Journal* 94.2 (2010): 315-17. Brown University. Web. 6 Mar. 2015.

"Memory Management." *Memory Management — Python 2.7.9 Documentation*. Python.org, n.d. Web. 06 Mar. 2015.