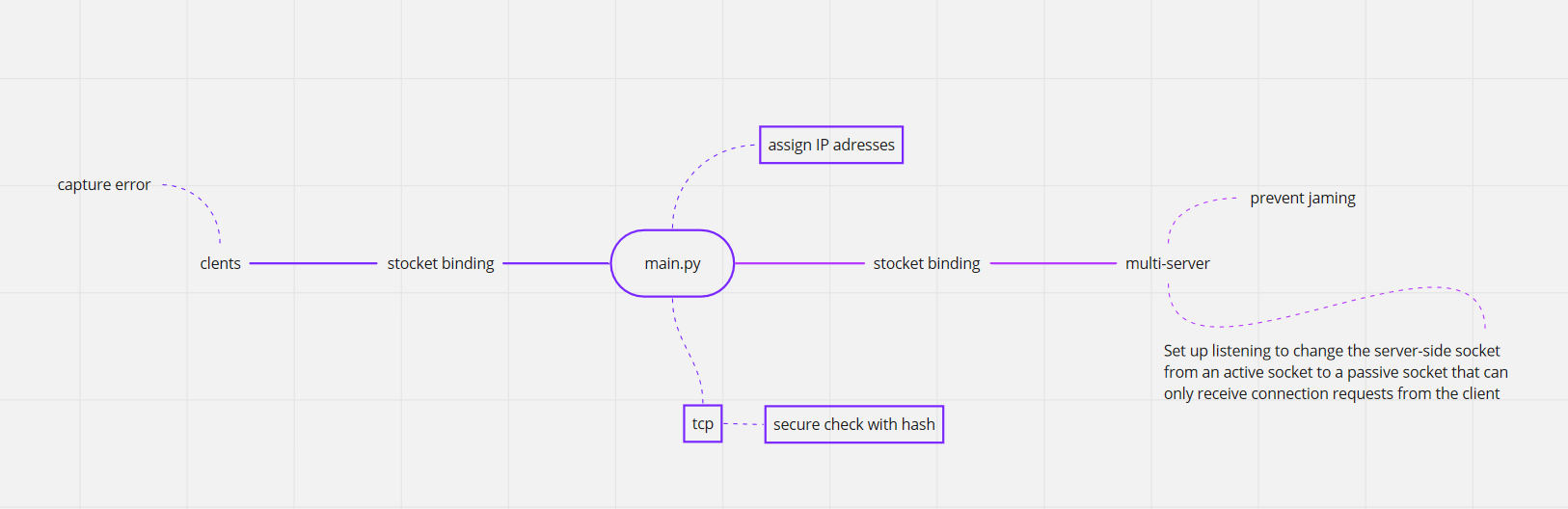
ABSTRACT

When transferring large volumes of data, reliability and flexibility become imperative considerations. Based on this reposition, the project is crafted to help student practice evaluating the structure of large efficient flexible and trusty file sharing systems.

INTRODUCTION

This project, focusing on building a Large Efficient Flexible and Trusty (LEFT) Files Sharing system resembling various cloud platform, requires students to have a basic understanding on filing sharing technics, network coding and python. In modern society, information is becoming more and more important, and the technology for exchanging, transferring and storing information over the network is constantly advancing. Introducing new architectures, like google cloud platform did back in 2011 which is applying APP Engine (GAE) [1], can lead to more functionality. Moreover, it is also important to improve fault tolerance to reduce transmission costs.

Methodology



A brand-new customized protocol that is highly compactable with the system was designed. Initially the virtual machines are expected to be connected through a unique ethernet created under Microsoft KM-TEST however soon assigning IP addresses became a problem and after realizing HTTP protocol was no allowed to use, no other choices but to build this projects’ own protocol.

Some of the functions are showing in the mind map, other functions include:

Adding messages using the update method of the hash object for clients.

Checking the client file update, the main file update is still filename and file size, if one of these two changes, the file is considered to have changed, and check the file save format. If a new file is found to be added, update the new file name and size to file\_change\_client. If it is a folder, then recursive traversal is used, and the last saved is full\_path.

Using the rsa module to generate asymmetric keys, including public and private keys. Pass the public key to the server, and the stringified sha256 value of that public key. Accept the key passed by the server and decrypt it.

Applying struct module to send fixed length data to the server to prevent sticky packets.

Tcpserver is able to Serialize symmetric keys encrypted with public keys using pickle for network transmission.

Implementation

There are no server problems.

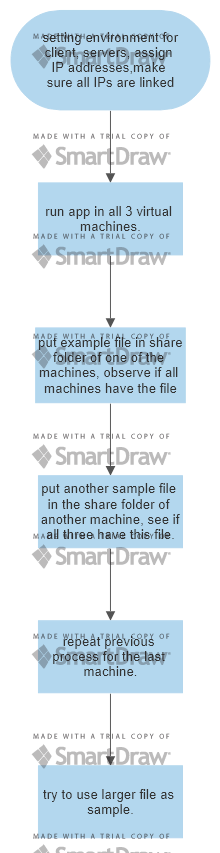
Basically, the program goes by:

The main program will first get the command line data and then assign the ip port, open multiple servers and pair the port numbers.

The client program will first import the module, create the socket and connect the ip address and port. The server will do the same.

Then the server will encrypt the file, and the client will decrypt it to achieve the download function.

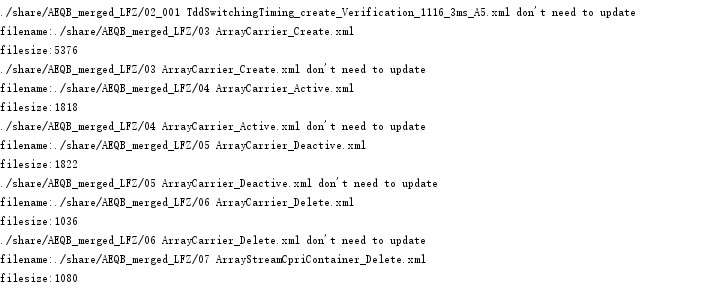
(Please See next page for flowcharts that illustrated the running procedure.)

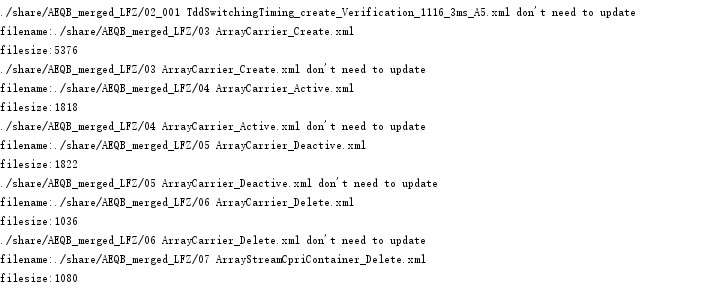


Testing and results

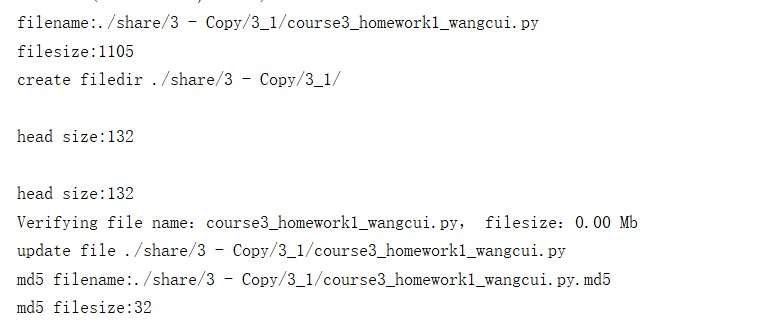
Simply drag a file under the share folder of one of the machines. Observe if then all three machines all have the file.

Fig below showed processing mechanics.





Here is the result of transformating a sample file, see that file have been converted into md5.



Conclusion

The project was successfully completed with majority of the tasks accomplished. Problems regarding IP assigning as well as this inconvenience of not able to use http was meet but were solved by using materials provided on CDSN. Moreover, thankfully other developers on codingame hand stackoverflow offered many advices on module calling and function constructing thus this program can be finished on time. in future I will be sure to make good use of these online resources.

Reference lists

[1] Wei-Tek Tsai; Peide Zhong; Jay Elston; Xiaoying Bai; Yinong Chen,

“Service Replication Strategies with MapReduce in Clouds” in *2011 Tenth International Symposium on Autonomous Decentralized Systems*: IEEE Xplore, April 2011.