# Design Document

Overview

Our implementation of this project is centered around python code with the use of the mininet API. We used mininet’s default single switch topology as it provides us with two hosts connected by a switch. We then assigned each of them an IP address as this made it easier to set up the connection between S (server) and R (client). To create the server and have the client connect to it, we used Python’s socket API.

The Server

The server listens on its port for a connection. Once a client is connected, the server then waits until the client sends it a message. Then, based on the message it will return to the client appropriately. If the server encountered an error while parsing the request, the server will send back an error message as described in the protocol documentation. Otherwise it will handle the request messages as described in the protocol. The server only recognizes Request messages as described in the protocol, it will ignore Response messages.

Sorting

If a “Sorted” request is received, the server looks at the next word on that line. If that word doesn’t match “Name”, “Quantity”, or “Date” exactly, the server returns an error saying that the server did not understand the request. If it is a valid request, the server reads the inventory.txt file and creates an array of 3-tuples, where each tuple is one item in the inventory. The array is then sorted and converted into a string format, where each item is in the form “Name,Quantity,Date” and each item is separated with a ‘;’. The date is stored in “YY-MM-DD” format. Once the conversion to string is complete, the server sends a data Response message as described in the protocol.

Update

Updating the inventory required an Update keyword along with 2 other data entries: the name of the item to be updated and the new value of the quantity field. After the server receives this request, it reads the file Inventory.txt, uses split functions to acquire the names of every object and searches for the right item name. After that it rewrites all the items but with a change of quantity to the desired item. Then it proceeds to send a success message to the client. In a situation where the server is unable to find the item name, it sends the client an error message.

Delete

The delete function enables the client to remove a particular item within the inventory file maintained in the server. This function takes a single parameter that represents the name of the item that the client wants to remove. Upon receipt of this parameter, the function reads through the file named *Inventory.txt*, simultaneously searching for an item with the same name as the one provided by the client. After doing so, it rewrites all of the items except for the item with the same name as that provided by the client. If an item was found with the same name, the function returns a success message to the client. Otherwise, an error message is returned informing the client that an item with the name they provided was not located in the inventory file.

The Client

The client connects to the server using sockets. It tries to connect to the IP address that was assigned with mininet and connects to the server’s port number. Then, the client can send a request to the server. After sending the request to the server, it waits for its response. Once the response is received, the client parses it. If it finds data in there, it turns the text into a file and saves it to receiver/inventory.txt. Otherwise, it just prints the error or success message.