Client

* parse command
* if valid continue
* Open and parse dfc.config for server connections
* fork or loop for each file
* handle command

Put

* check it file exist locally
* split up file in chunks
* add correct tags (from who, time, chunk, filename, method)
* now connect to servers
* wait for one second before determining unavailable
* if connection succeeds send packet to server if succeeds go to next
* If filename has exact same time as the file then store both
* otherwise go back through successful connections and delete data and list name and print put failed and if delete tails print that too
* end connections and return

Get

* Connect to servers
* wait for one second before determining unavailable
* Create message delimiter with tags (method, filename)
* If multiple connections fails or doesn’t receive correct data multiple times, then can’t reconstruct and print that it is incomplete
* If success take out response message delimiters and reconstruct in correct order
* Check if multiple files
* Write to local

List

* Connect to servers
* wait for one second before determining unavailable
* Create message with tag (method)
* Compare all lists
* if name missing multiple in multiple servers mark that file as incomplete
* print file names
* return

Server

* parse arguments
* set up folder if it doesn’t exist
* set up socket on given port
* add server to dfc.config file
* loop
* listen for connections and fork when one is accepted
* open new socket
* close original socket
* parse command
* if valid continue to handling commands
* close new socket
* end process
* remove from dfc.config file

Put

* parse packet
* Hash filename to store and add filename to list file
* If file exists and has exact same time as the file then store both (no way this will happen)
* Hash that filename with a 2 on the end and if that exists do a 3 and so on
* If put successfully respond with an acknowledgment
* return

Get

* Parse packet
* Hash filename and look for it
* If exists send it packet back and check for duplicates
* otherwise send back doesn’t exist
* send acknowlgement
* return

List

* open list file
* if empty send back an empty file
* make a packet with list as a deliminator
* send packet
* return