Phase 2 Design Document

This is the document for the Phase 2 of the project. Phase 2 requires to send one file from server side into chunks to client side and client receives those chunks, join them and save it as an image onto the laptop.

We have written two codes – one for the client side and other for the server side. Each code has one class – client() and server() respectively.

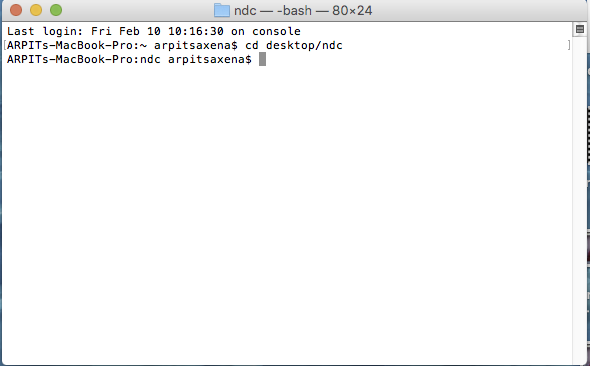
1. Client() – This class contains two functions and is responsible for the function of the client side. It receives message from the server side, it checks those messages and decides what next message will be. It receives packet size and number of packets. It also receives an image from the server in packets. It assembles those packets and reconstruct the image on the specified path.
2. Server() – This class is mentioned in the code written for server side. It has four functions and is responsible for the function of the server side. It sends the message to client with the number of packets and the packet size. It breaks down the file into packets and then send those packets to client.

Functions:

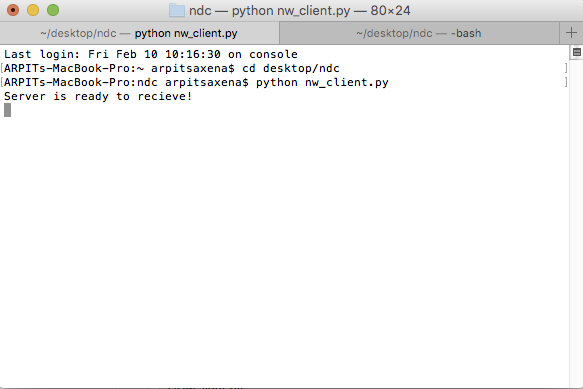
1. \_\_init\_\_ - this is init function defined for both the classes and it holds the initialization of global variables. We also created socket within the init function for both client and server.
2. clientRecv – It receives message from the server, check those messages and accordingly save upcoming messages to some variables. It includes number of packets and total file size. Moreover, it receives the image from the server in form of packets. It joins all the packets and save it as an image on to the specified path.
3. serverSend – This function opens an image and decode it using base64 library. It further divides the image into several chunks of fixed size and send them to the client.
4. send\_msg – This function is responsible for sending out the messages to the client. It takes the message as the argument.
5. make\_packets – The primary role of this function is to divide the image into chunks. It takes the whole file data as the argument.

Step by Step Execution:

1. Open the terminal and change the current directory to the desired directory.



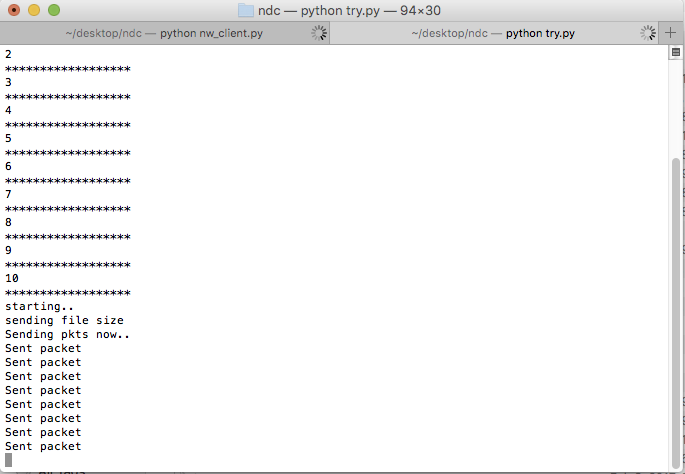
1. Run the client code by writing the command – python followed by the filename and the moment we hit enter, code will execute and client is waiting to receive the message from server side.



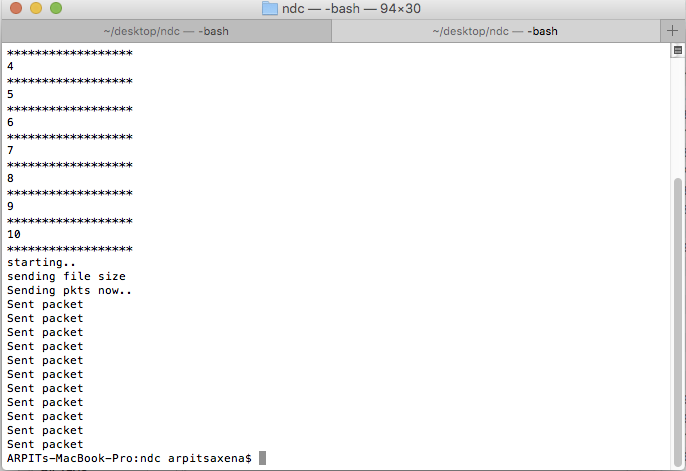
1. Open another terminal to execute the server code.



1. We will again use the command – python followed by the filename. Next we hit the enter and it will be converting the image into chunks – the numbers it is displaying are the number of packets created.



1. Now every packet is sent from the server side to client side.



1. This is the client side which received total number of packets, each packet size. Then it will receive all the packets, assemble them and save it back as the image on the relative folder.

