CSC/ECE 573 Section 001
Spring 2017
Project 1

Peer-to-Peer with Centralized Index (P2P-CI) System for Downloading RFCs

Name 1: Harish Pullagurla Student ID1: 200178872

Name 2: Venkata Surya Subrahmanyam Nukala Student ID2: 200158956

Project Objective

Development of a Peer to Peer system for file transfer with a centralized server

How to use the Code :-

1. Run the Central server file – Server.py
   It starts listening on a TCP Socket Stream on a well-known Port Number '5555'
   It prints a message "Started listening on IP 192.168.179.1 Port : 5555" upon successfully creating a socket.

2. Run the client file - Client.py
   It prompts the user to enter the Host Name which is the Name by which this host is to be registered with the Server.
   It is expected to give unique names for each of the client that is registering with the Centralized Server.
   The user is expected to enter the upload server details, the socket at which the client would be open, listening to other clients for registering. Upload TCP socket stream is started.

3. Operations Thread is initialized in the Client file, which is responsible for communicating with other sockets that are listening. This happens through a set of pre-defined commands that are decoded by each of the corresponding files for decoding the operations.

4. Select from the set of operations that are available at the client side. This generates commands to communicate with the server

5. Suggested usage sequence :-

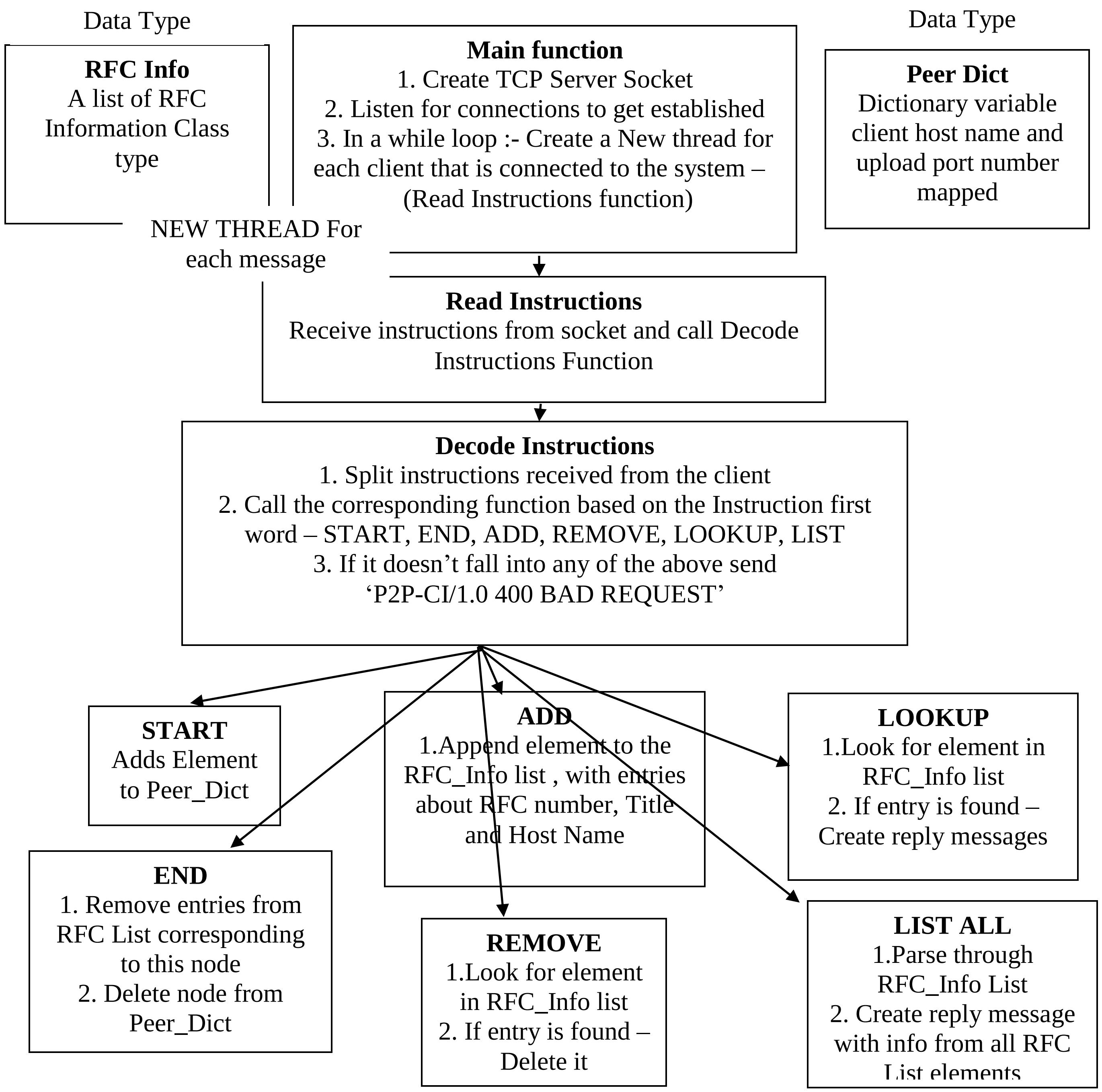
   a. Add a file with Client into the centralized server, by giving the RFC number and title of files present in the Client folder
   b. Try to Lookup for the reference by querying the RFC number available with the Central Server.
   c. Try to Get the list of all RFC's present with the Central Server
   d. Get RFC's present with other clients
   e. Remove File from RFC
   f. Exit from the system – removes entry from the Central Server

6. Server prints each of messages it receives and makes note in the backend about the available files with each of the peers

Code Structure:-

Server. py

The primary task of this file is to listen on a well know port number and maintain a Centralized Index of all the files that are available with the server
This file decodes the predefines set of instructions to complete the tasks and reply back with the relevant instructions.



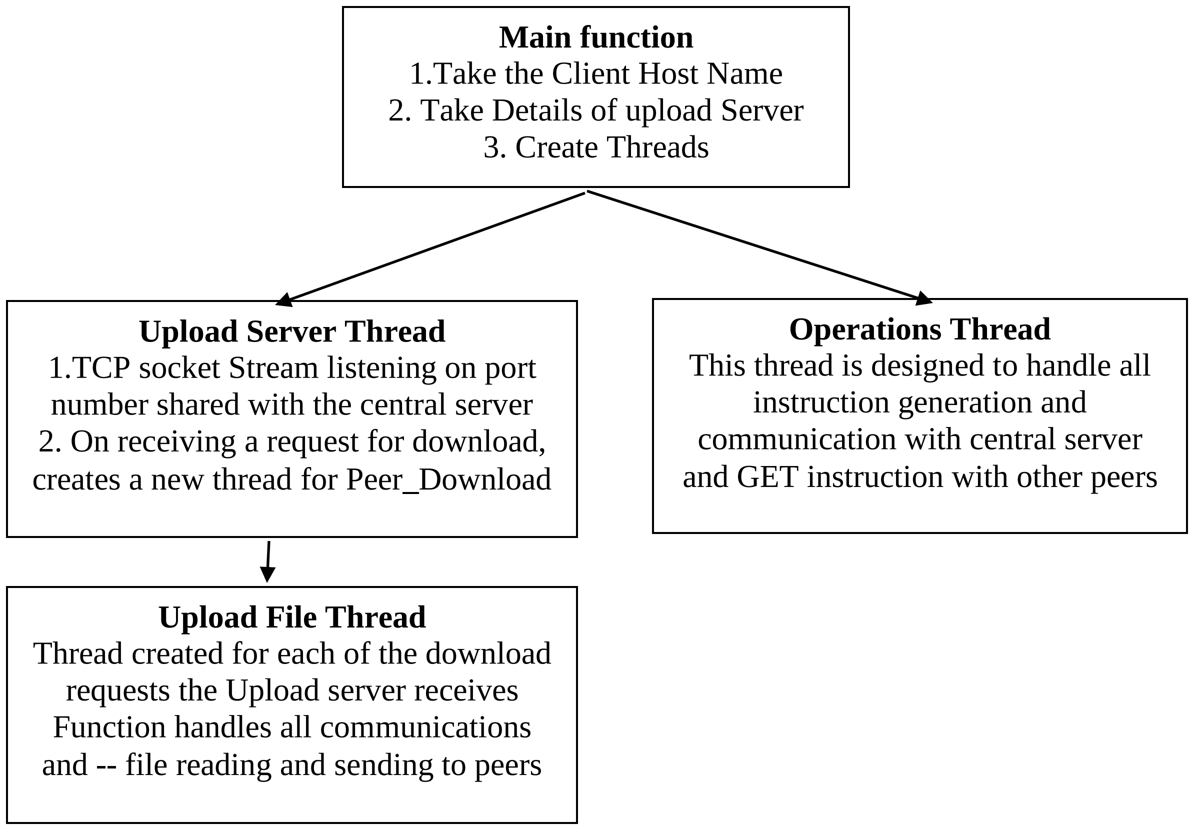
Data Type
RFC Info
A list of RFC Information Class type
Main function
1. Create TCP Server Socket
2. Listen for connections to get established
3. In a while loop :- Create a New thread for each client that is connected to the system – (Read Instructions function)
Data Type
Peer Dict
Dictionary variable client host name and upload port number mapped
NEW THREAD For each message
Read Instructions
Receive instructions from socket and call Decode Instructions Function
Decode Instructions
1. Split instructions received from the client
2. Call the corresponding function based on the Instruction first word – START, END, ADD, REMOVE, LOOKUP, LIST
3. If it doesn't fall into any of the above send 'P2P-CI/1.0 400 BAD REQUEST'
START
Adds Element to Peer_Dict
ADD
1.Append element to the RFC_Info list , with entries about RFC number, Title and Host Name
LOOKUP
1.Look for element in RFC_Info list
2. If entry is found – Create reply messages
END
1. Remove entries from RFC List corresponding to this node
2. Delete node from Peer_Dict
REMOVE
1.Look for element in RFC_Info list
2. If entry is found – Delete it
LIST ALL
1.Parse through RFC_Info List
2. Create reply message with info from all RFC List elements

Clinet.py

Client has 2 basic tasks

1. Communicate with the central server to get the required information
2. Communicate with the upload server of other peer to download files that are present with it

Code Structure:-



Main function
1.Take the Client Host Name
2. Take Details of upload Server
3. Create Threads
Upload Server Thread
1.TCP socket Stream listening on port number shared with the central server
2. On receiving a request for download, creates a new thread for Peer_Download
Operations Thread
This thread is designed to handle all instruction generation and communication with central server and GET instruction with other peers
Upload File Thread
Thread created for each of the download requests the Upload server receives Function handles all communications and -- file reading and sending to peers

Operations Thread :- This thread handles all communication , instruction generation that is to be send to the centralized server

Upon starting the thread – a start message with client host name and port number is sent to the Central server to register with it , by adding an entry into the peer dictionary.
In our design a function Message Central server – handles all communication to the server. This function creates a new tcp socket each time sends the message to the server, closes the socket.

After registering client gets to choose from one of the following options for actions to be taken / instruction generation
1. Add RFC - To Add File with Client into Server List
2. Remove RFC - Remove the file from Server List
3. Lookup RFC - a specific RFC for download from other Peers
4. List All - available with Central Server

5. Exit Client Listing from Central Server
6. GET RFC - To Communicate with Other Clients Present

Corresponding messages are created when each of the options is selected and Message Server functions is called which transmits the message
When Option 6 – **GET RFC** is selected , a separate function - download file is called that communicates with another peers upload server, to get file data iteratively as multiple packets.

Thus running – Server and multiple instances of Client program could mimic a Peer to Peer System.