Alp Kural - 71959
Computer Engineering

Comp 416 – Application Layer Programming Project

Project Description

The NFTNet project is centered around the development of an application layer protocol that interacts with the CoinGecko API to facilitate the extraction of Non-Fungible Token (NFT) information. This project combines the principles of application layer protocols, client/server model, application software development, socket programming, and multithreading to create a robust and efficient solution for accessing NFT-related data from CoinGecko.

Understanding The Sequence of the Application

1. NFTNet clients connect to the server, and the server displays information about the connected sockets.
2. Clients send queries to the server following the protocol specifications.
3. Upon receiving a query, the NFTNet server extracts the relevant data and prepares a request to be sent to the CoinGecko API.
4. The NFTNet server establishes a connection with the CoinGecko API and retrieves the necessary information.
5. The server communicates the results to the respective client using the communication protocol.
6. Clients acknowledge the receipt of information following the communication protocol and present the received data to the users.
7. Clients can then express their intention to send more queries or terminate the connection.
8. The server has the option to disconnect a client's socket if no further messages are received from the client(s).

Application Layer Server-Client Communication Protocol (NFTNet Protocol)

Application Layer Server-Client Communication Protocol is important due to the fact that it standardizes the messaging between the server and the client into a structured form.

Supported Structure of the Client Queries:

- Query Type 1 = “GET /NFT Data API /NFTNet 1.0”
- Query Type 2 = “GET /NFT Search API /{ID} /{attribute}* /NFTNet 1.0”

- Request type = {GET}
- API Name = {NFT Data API, NFT Search API}
- ID = {Set of valid IDs for NFTs}
- attribute = {Set of valid attributes for NFTS}
- NFTNet Version = {NFTNet 1.0}

PS. ":" notation is supported to concatenate the queries which allows client to send more than one query at a time.

i.e., Query 1:Query 2

Supported Structure of the Server Responses:

- "NFTNet 1.0 400 Bad Request", if the client query is not understood by the server and the query not comply with the NFTNet Communication Protocol.
- "NFTNet 1.0 200 OK Content-Length: {Length of the Response String} Response: {Response String}", if the server fulfills the requested client query.

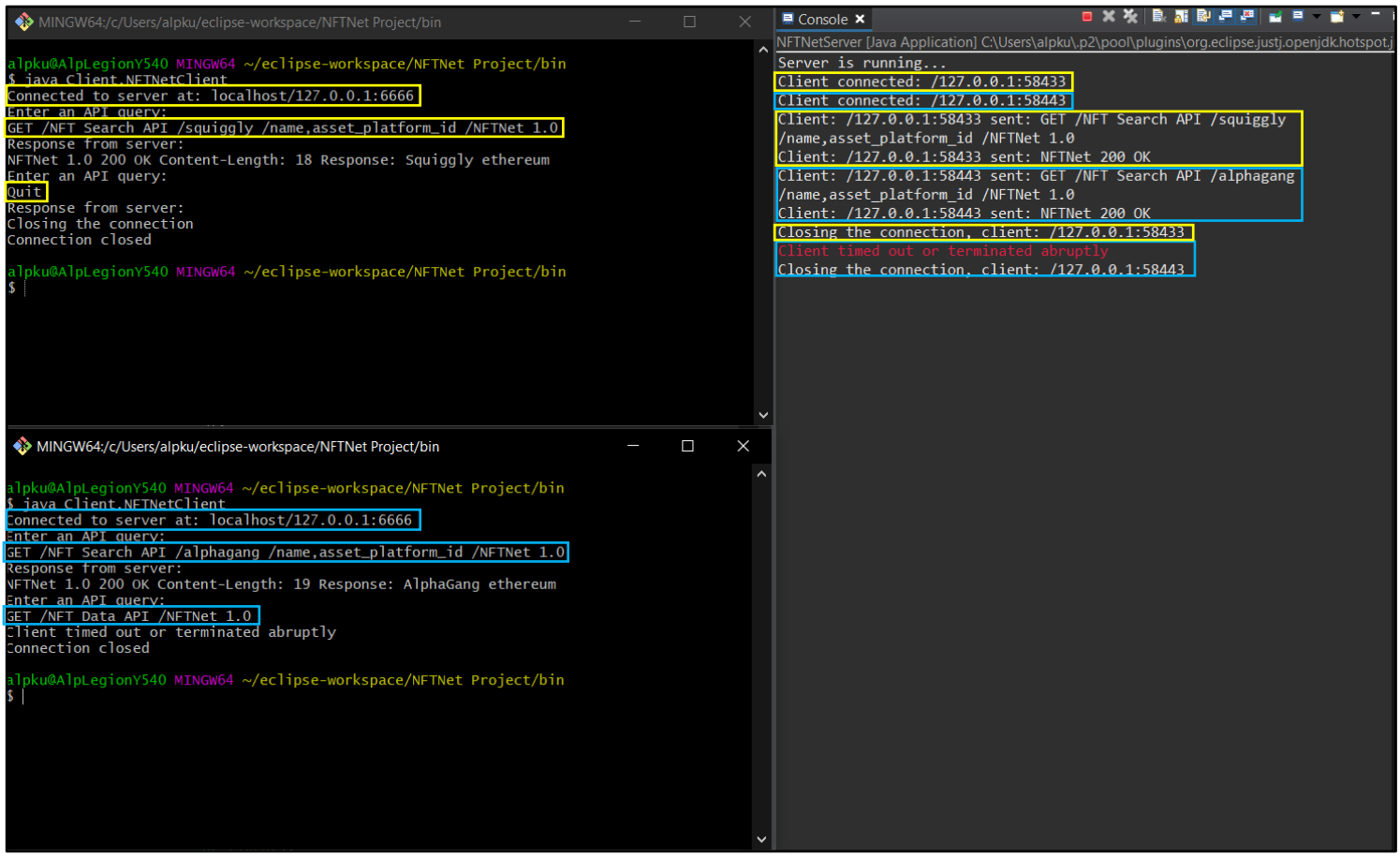
Upon successful receive of the server response, client sends "NFTNet 200 OK" acknowledgement message to the server regarding status of the received response.

Runtime Snapshots of the NFTNet Application

Server-Client Interactions and Test Case Scenarios

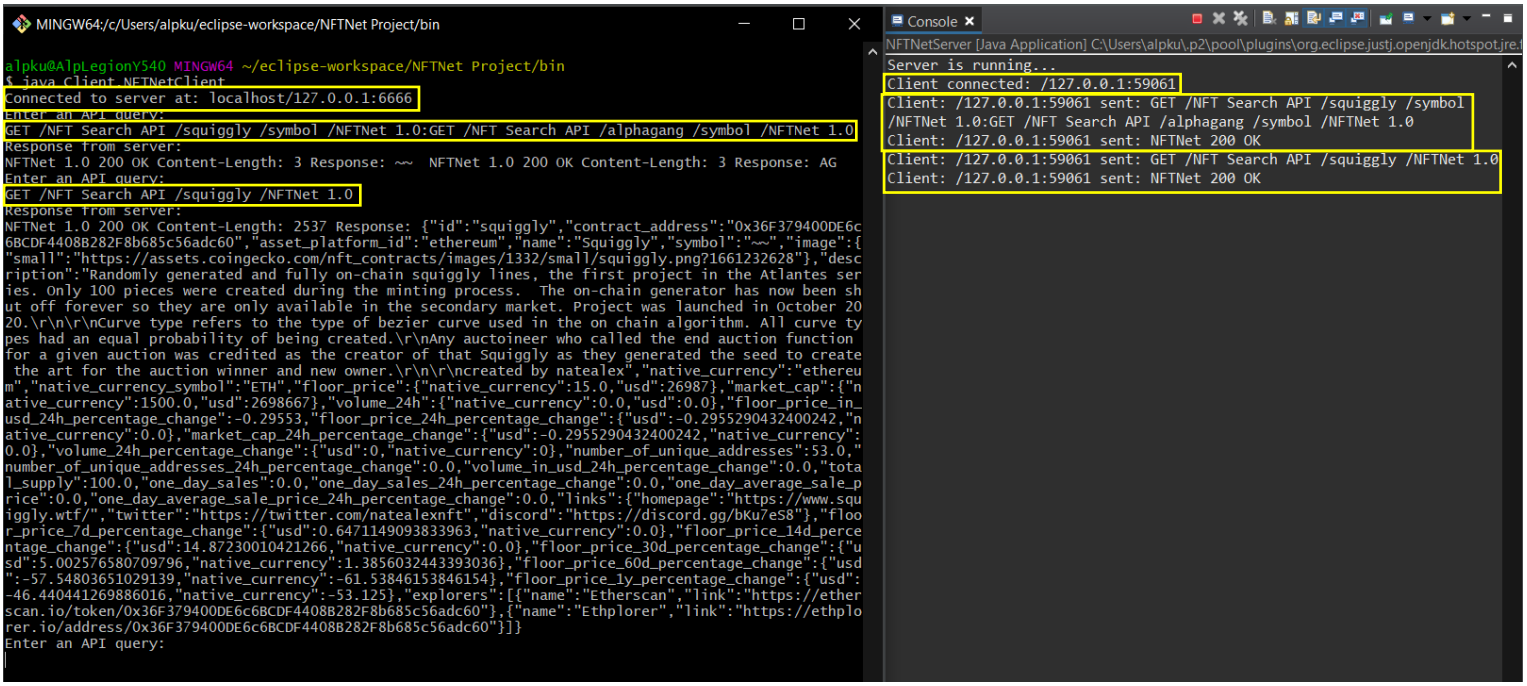
Scenario #1

Two clients get connected to the server and both of them sends one "GET /NFT Search API /{ID} /{attribute}* /NFTNet 1.0" type query and the server returns the responses to the clients. After being done with the querying client 1 labeled as yellow in the snapshot quits the NFTNet service application, whereas, client 2 which is labeled with light blue does not quit the application. After a non-responsive two minutes of a time server timeouts the connection of the client. Therefore, the next query the client 2 conducts results in an error message.



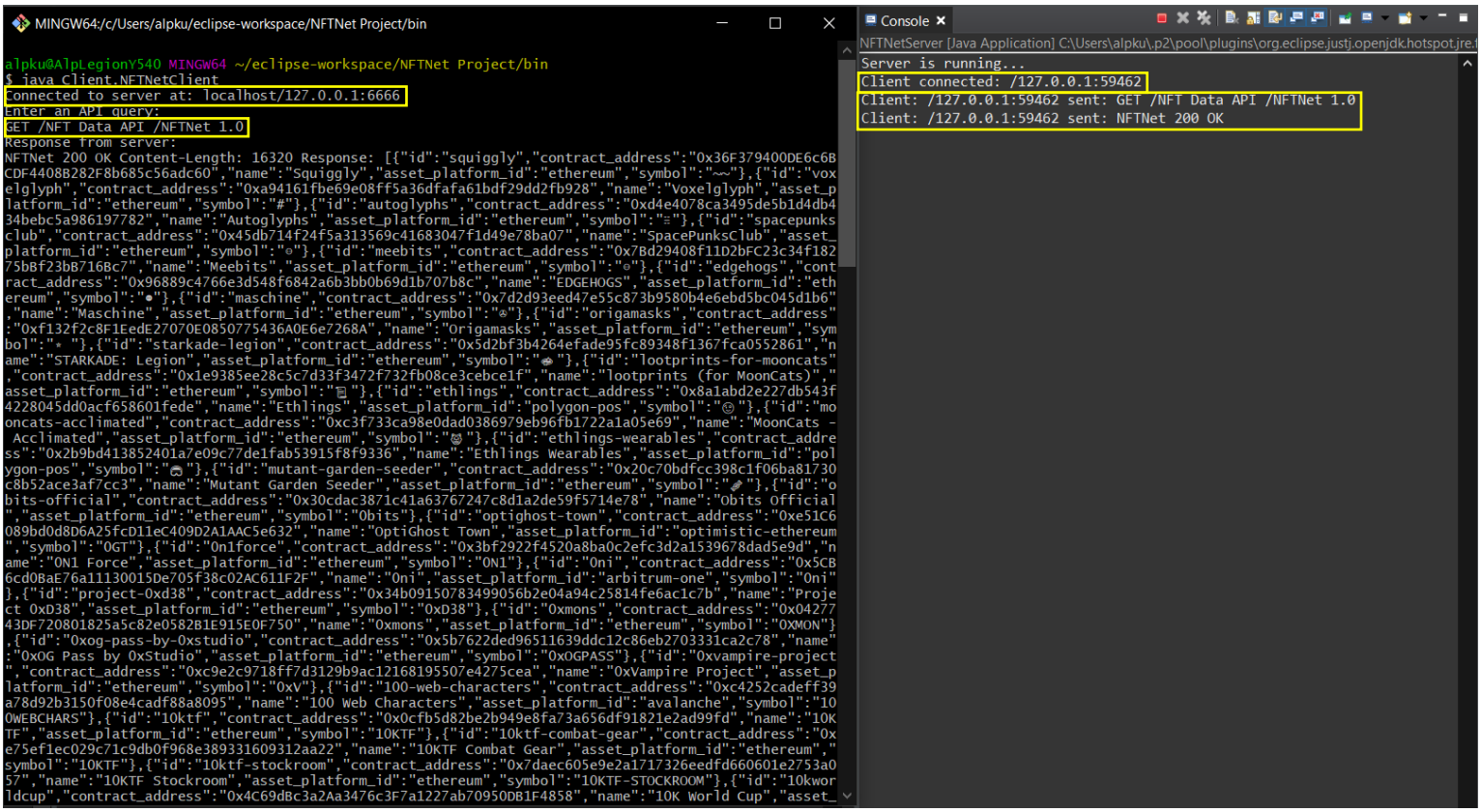
#Scenario 2

One client gets connected to the server and sends two “GET /NFT Search API /{ID} /{attribute}* /NFTNet 1.0” type query requests in a single prompt and the server returns the responses to the both queries in a single line. Then the client sends yet another “GET /NFT Search API /{ID} /{attribute}* /NFTNet 1.0” type query but this time without the specified attributes, then the server in response returns the all the specifications of the given NFT ID.



#Scenario 3

One client gets connected to the server and sends “GET /NFT Data API /NFTNet 1.0” type query to the server and the server returns the NFT list response to the client.



Alp Kural - 71959
Computer Engineering

#Scenario 4

One client gets connected to the server and sends a query not complying with the NFTNet Server Client Communication Protocol and the server returns the bad request response to the client. Then client sends yet another query with the proper protocol structure, however with a non-existent attribute of the NFT objects. Therefore, server returns “error” for the value of the non-existent attribute. Then client sends another query in the concatenated query format and since one of the queries in the concatenated query prompt is not valid the server returns bad request corresponding to the non-complying query.