Comp416: Computer Networks - Project:1 NFTNet CoinGecko-Api Based Application Layer Protocol Report

Batuhan Arat, 68665

Brief Description of the Project

In this project, I have design a network application layer that has server side and client side with an CoinGecko API. CoinGecko hasn an api to investigate NFTs. I have used 2 endpoints of the api.

1) List all the nft's id, name, contract address, platform id and their symbol with endpoint

https://api.coingecko.com/api/v3/nfts/list

2)For given id of the nft, exhibit its name , asset platform id and price in USD with endpoint

https://api.coingecko.com/api/v3/nfts/id

Only the server talks to the API. Client only talks with server with obeying protocol rules. By following protocol rules, client can request information from the server. Server takes this request, talks with api, and return the information to the client.Server is designed to be multithreaded. That means that more than one client can separately talk with the server.

Client Server Application Layer Protocol Design

I have design a simple protocol. There are certain types of commands that server accepts in order to answer certain questions.For example;

If client writes; '**help' :** Server returns what type of commands are appropriate.

If client writes; '-**list' :** Server use first endpoint to talk with api, and return the information about all the nft's that are present in CoinGecko system.

If client writes; '**-show <id>**': Server use second endpoint to talk with api, and return the specific nft's information that is requested by the id.

If client writes; '**bye**': Server close the socket of the client.

If client writes anything else other than the commands above, server warns the client with the command is not applicable.

Whenever, I have send output to the client, I have to finish with `os.println("END_OF_RESPONSE");` in order to tell client that my message is over. That is needed because at the socket, server transmit the outputs as chunks whenever it saw a newline. My protocol is rely on tcp, therefore I have to send message from the client in order to get an response from the server. I want to get full response when I give the message from the client. If I have did not implement this strategy, my response would keep coming if it include newline. That means that even I write a different command, I would have still been receiving the previous command's response whenever I type something until it finishes. So, in my logic client checks `END_OF_RESPONSE` string is encountered, if it is not it keeps reading the output from the server.

Client:

Client has 2 classes as MultithreadedClient and ConnectionToServer.

MultithreadedClient class uses ConnectionToServer class to connect to server, getting messages from server and send messages to server. It reads the messages from the user terminal input and writes instructions to the user.

While it is not receiving ‘bye’ command, it keeps reading the terminal. Every response is also kept reading until reached ‘`END_OF_RESPONSE`' as protocol suggest.

If client is unresponsive for 1 minute, server close the socket of client. Before doing that server sent a message to client as ‘`TIME_OUT`' . Client is responsible for if `TIME_OUT` is received, close the socket and quit the application

Server

Server opens a welcoming socket with port 4444 and starts to listen for any client. When client is connected, it opens a new socket. To establish a time-out for unresponsive client, after the accepting connection I have set timeout as 1 minute in the socket.Then in order to serve to all client as multithreaded fashion, I start the server thread.

Server thread is responsible for the initialize the api, take input from the client and send output to the client. It should accepts inputs those are suitable for the protocol. It uses the api to requests that wants to list all nfts and specific nft.

Our api is just a RESTful api that opens connection at the endpoint and get the information. `getListOfNFTs` method is for getting the list of the nft’s and it is already returning information which is suitable in json form. I encapsulate it as JSONArray after i scan the whole information then return it to the server.

`getNFTbyId` method accept a string which should be an id. Its endpoint includes the id to be set the connection. This information is not in the json form, so I have add additional ‘[’ , ’]’ to beginning and the end of the object, than encapsulate it as JSONArray and return it to the server.

Server thread gets those responses and parse it to the suitable way that it readable by client. While we are reading values with a given key at the json, we have to be careful about null values. I have return a null string if the values is null with `safeGetString` method. Also, `showWithIDResponse` method which is responsible for parsing the json needs additional parse method `safeGetJson` which is needed for getting usd information of the

nft. Usd information has encapsulated it another json, therefore that should be taken first.

Server also set timeout like this `s.setSoTimeout(60000);` after it's welcoming socket accepting the connection. Time out should be implemented on the accepted socket, not the welcoming socket. When it is timed out, send a string to the client to inform when client try to write something.

In the pdf, it is written that we should also take into account that exceeding max number of api calls per minute which is 50. We should check the error code of the response where we make the request to the api. Then catch the created exception at the server thread

```
int responseCode = conn.getResponseCode();
        if (responseCode == 429) {
            throw new IOException("Your api calls should not exceed 50 per minute");
        }
```

```
catch (IOException e) {
            if(e.getMessage() !=null) {
                System.err.println(e.getMessage());
            }else {
                throw new RuntimeException(e);

            }
        }
```

Test Scenarios

Client Cases

In this scenario, writing wrong command, help command, first api call, second api call with correct argument, second api call with wrong argument and exit statement are tested.

```
/Users/batuhanarat/Library/Java/JavaVirtualMachines/openjdk-19.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=64
Client connected from 'local '/127.0.0.1:64145
Client connected from 'remote 'localhost/127.0.0.1:4445
Enter a message for the server (type 'bye' to quit, 'help' for commands):
list
You entered a wrong command in terms of protocol. Write 'help' to see commands

Enter a message for the server (type 'bye' to quit, 'help' for commands):
help
Use '-list' command to see all the NFTs in the CoinGecko platform.
Use '-show id' to see specific nft's information.You should replace the id by the corresponding id of the nft.
Use 'bye' command to exit.

Enter a message for the server (type 'bye' to quit, 'help' for commands):
-list
Id: squiggly    Name: Squiggly  Contract Address: 0x36F379400DE6c6BCDF4408B282F8b685c56adc60    Platform Id: ethereum   Symbol: ~~

Id: voxelglyph  Name: Voxelglyph    Contract Address: 0xa94161fbe69e08ff5a36dfafa61bdf29dd2fb928    Platform Id: ethereum   Symbol: #

Id: autoglyphs  Name: Autoglyphs    Contract Address: 0xd4e4078ca3495de5b1d4db434bebc5a986197782    Platform Id: ethereum   Symbol: ≋

Id: meebits Name: Meebits   Contract Address: 0x7Bd29408f11D2bFC23c34f18275bBf23bB716Bc7    Platform Id: ethereum   Symbol: ⚇

Id: spacepunksclub  Name: SpacePunksClub    Contract Address: 0x45db714f24f5a313569c41683047f1d49e78ba07    Platform Id: ethereum   Symbol: ⚇

Id: edgehogs    Name: EDGEHOGS  Contract Address: 0x96889c4766e3d548f6842a6b3bb0b69d1b707b8c    Platform Id: ethereum   Symbol: ●

Id: maschine    Name: Maschine  Contract Address: 0x7d2d93eed47e55c873b9580b4e6ebd5bc045d1b6    Platform Id: ethereum   Symbol: ⊛

Id: origamasks  Name: Origamasks    Contract Address: 0xf132f2c8F1EedE27070E0850775436A0E6e7268A    Platform Id: ethereum   Symbol: ⭐

Id: starkade-legion Name: STARKADE: Legion  Contract Address: 0x5d2bf3b4264efade95fc89348f1367fca0552861    Platform Id: ethereum   Symbol: 💫

Id: lootprints-for-mooncats Name: lootprints (for MoonCats) Contract Address: 0x1e9385ee28c5c7d33f3472f732fb08ce3cebce1f    Platform Id: ethereum   Symbol: 🟫
```

```
Id: acacia-girl Name: Acacia Girl   Contract Address: 0x916d30768be5dcad473085138369fb7a7e7574e0    Platform Id: ethereum   Symbol: AG

Id: real-agents Name: REAL AGENTS   Contract Address: 0x33497c777dcccd4d209ad4518e6c6b23ddc45705    Platform Id: ethereum   Symbol: AGENT

Id: ag3dnft Name: AG3DNFT   Contract Address: 0x4bafc595a9ff4a5f4936689a0389c148a65456a2    Platform Id: binance-smart-chain    Symbol: AGN

Id: agame3dnft  Name: AGame3DNFT    Contract Address: 0x9d8f8ea82b12a6e48d8423f0ec9907ad599233a8    Platform Id: binance-smart-chain    Symbol: AGN

Id: alpha-gardeners-pass    Name: Alpha Gardeners Pass  Contract Address: 0x88f753c48008b938dd7f78751d3106d04ee432e3    Platform Id: ethereum   Symbol: AGP

Id: ape-hater-club  Name: Ape Hater Club    Contract Address: 0x9370045ce37f381500ac7d6802513bb89871e076    Platform Id: ethereum   Symbol: AHC

Id: avalanche-hills-muscle-cars Name: Avalanche Hills Muscle Cars   Contract Address: 0x66f703e48f68c03fffee0eaee7be2fe411cb3713    Platform Id: avalanche  Sy

Id: ape-invaders-genesis    Name: Ape Invaders Genesis  Contract Address: 0x3ed7dfaca773b90da1bc3fa8b04656c917d33afb    Platform Id: ethereum   Symbol: AI

Id: asm-aifa-genesis    Name: ASM AIFA Genesis  Contract Address: 0x26437d312fb36bdd7ac9f322a6d4ccfe0c4fa313    Platform Id: ethereum   Symbol: AIFABOX


Enter a message for the server (type 'bye' to quit, 'help' for commands):
-show wrongid
You entered a false id, check again

Enter a message for the server (type 'bye' to quit, 'help' for commands):
-show asm-aifa-genesis
Nft Name: ASM AIFA Genesis  Asset Platform Id: ethereum Price in USD: 1033.92

Enter a message for the server (type 'bye' to quit, 'help' for commands):
bye
ConnectionToServer.. Connection Closed

Process finished with exit code 0
```

In this scenario time-out scenario is tested

```
/Users/batuhanarat/Library/Java/JavaVirtualMachines/openjdk-19.0.1/Contents/
Oppened up a server socket on Batuhan-MacBook-Pro.local/127.0.0.1
A connection was established with a client on the address of /127.0.0.1:5216
Client socket is timed out for being unresponsive 1 minute.
Socket Input Stream Closed
Socket Out Closed
Socket Closed
Closing the connection
```

```
Client connected from 'local '/127.0.0.1:52164
Client connected from 'remote 'localhost/127.0.0.1:4449
Enter a message for the server (type 'bye' to quit, 'help' for commands):
help
Use '-list' command to see all the NFTs in the CoinGecko platform.
Use '-show id' to see specific nft's information.You should replace the id
Use 'bye' command to exit.

Enter a message for the server (type 'bye' to quit, 'help' for commands):
-list
Client socket is timed out for being unresponsive 1 minute.
ConnectionToServer.. Connection Closed

Process finished with exit code 0
```