COMP416: Computer Networks
Project 1 - Part 2
CoinNet: CoinGecko-API Based Application Layer Protocol

Due: 04/11/2022 (Late submissions will not be accepted).

Submission of the project deliverables is via Blackboard.

This is an individual project. You are not allowed to share your codes with each other.

This project work is about the **application layer** of the network protocol stack. It involves application layer protocol principles, design and implementation of a client/server protocol, application layer software development, socket programming, and multithreading.

Through this project, you are going to develop an application layer protocol by interacting with the application layer abstract programming interface (API) of CoinGecko (https://www.coingecko.com/en/api/documentation). You are required to work with the following APIs for cryptocurrency information extraction from the CoinGecko server:

1. List cryptocurrencies data API
2. Search for a cryptocurrency price API

These are two separate APIs that allow users to interact with the CoinGecko aggregated data and retrieve the required information related to cryptocurrencies.

Project Overview:

In this project, you are asked to develop a CoinGecko API based application layer protocol based on the client/server model.

CoinNet server uses TCP connection to interact with the clients. Fig.1 shows connections for a sample CoinNet server and client interactions. As shown in the figure, only the CoinNet server interacts with the CoinGecko API web server and provides the CoinNet client with the information required.

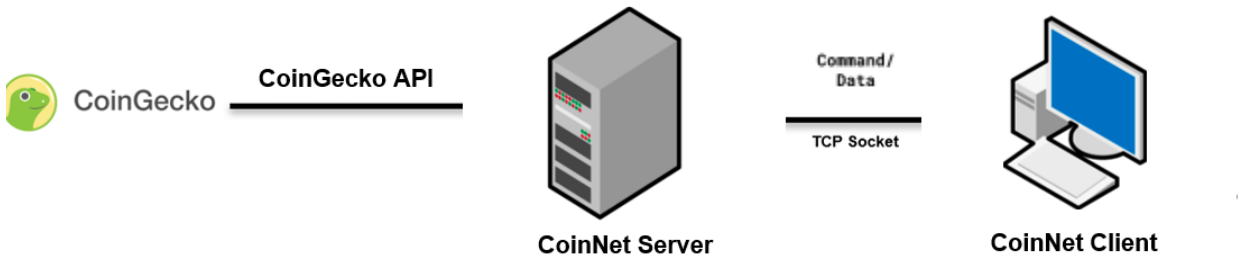

Figure 1. The CoinNet Client/Server connections.

Implementation Details:

The CoinNet application layer protocol has four main components:

- Interaction of the CoinGecko API and the server
- Server side of the application
- Client side of the application
- Client-Server Communication Protocol

It is pertinent to note that the mentioned APIs allow 50 calls/minute. That means a maximum of 50 API calls (method invocation) can be done each minute. The API will give a different response (error message) when trying to make more than 50 calls in a minute. The developed application and testing including for demonstration must keep these limits in perspective.

Methods:

1. List cryptocurrencies data API can be called with no parameters and returns a JSON array containing information about all the cryptocurrencies. Each cryptocurrency is represented as

a JSON object with *id*, *symbol*, and *name* fields. The endpoint for this API is https://api.coingecko.com/api/v3/coins/list.

2. Search for a cryptocurrency price API (https://api.coingecko.com/api/v3/simple/price?ids=idsOfCryptosCommaSeperated&vs_currencies=try). This API requires the ids of the cryptocurrencies that we need to get their price in addition to the currency of the price. In case of requesting the price for more than one cryptocurrency, we put their ids comma separated in the *ids* parameter. The parameter *vs_currencies* is used to determine the currency of the price and we can send a list of comma-separated currencies.

CoinNet Sequence overview:

1. CoinNet Client/s connect with the server and display the connected socket information.
2. Client/s send the query to the server formatted as per the protocol specifications.
3. The CoinNet server upon receiving the query, extracts the relevant part and formulates the request to be sent to the API server.
4. The CoinNet server establishes a connection with the required CoinGecko API and fetches the required information (JSON)
5. The CoinNet server communicates the results to the concerned client per the communication protocol.
6. Client acknowledges receiving the information as per the communication protocol and displays the received information for the user.

7. Clients may then indicate their willingness to send additional queries or terminate the connection.
8. The server may timeout the socket of a client if it does not receive any further messages from the client/s.

CoinNet Server side overview:

The server for the CoinNet application should:

1. Establish a connection with the respective CoinGecko web servers using the APIs and download the specified required information.
    a. Ask for a list of cryptocurrencies
    b. Ask for the price for a cryptocurrency or more by passing the corresponding ids. The server will fix the currency to be Turkish Lira (try) which is why the client will not send this parameter.
2. Allow multiple clients to connect using multi-threading. Each thread is responsible for serving a client.
3. Attach a timeout with the connection socket

Based on the client's request, the server will parse the client input and use the API to extract the relevant information from the web server. This information will then be passed to the client.

The process at the server side would be:

1. Create a welcoming socket
2. Accept incoming client connections at the welcoming socket, simultaneously using multi-threading.
3. Decipher the requests coming in from the clients and query the required information from the web server. (The protocol for forwarding requests has to be designed and developed by yourself and explained in your report.)
4. Send the requisite information to the client.

5. Terminate connection on a timeout or upon receiving a terminate “type” message from the client.

CoinNet Client Side Overview:

This application envisions multiple clients interacting with a single server. For this application, the clients should:

1. Be able to submit requests to the server.
2. Be able to receive data from the server
3. Display the data, acknowledge reception of data, generate a new request, or submit a ‘no request’ status

Client-Server Interaction

The client-side must take care of the parameters to be sent to the server.

The process on the client side will follow the given steps:

1. Initiate connection with the server
2. Pass the requests to the server
3. Receive the information sent over by the server
4. Display the information in the appropriate manner.
5. Terminate the connection in case appropriate data is received and no other request is forwarded within the timeout duration. (Appropriate tests for demonstration purposes should be developed.)

Protocol Design

All communications between the client and server must follow a set protocol. A communication protocol is a system of rules that allows two or more entities of a communications system to exchange information. The communication protocol defines the format for exchanging various types of messages. The entire payload is divided into fields, each of which represents a specific purpose. These fields themselves may be spread over bits or bytes. Each field has a defined set of values that indicate its significance and purpose which are publicly announced. For example, the socket address of the destination

may be embedded within the payload as the last 16 bytes. These last 16 bytes would then be considered to always contain the value of the destination address. As an example, a structure of a message may be divided into the following fields:

Dest Address (16 bytes)	Sequence Number (1 byte)	Data (8 bytes)	Msg Type (1 byte)	Origin Address (16 bytes)	Checksum value (4 bits)	Ack Required (4 bits)

Fig. 1: Example of a Protocol message fields

The entire packet structure in the example message structure above has been divided into 7 different fields with the data occupying the largest chunk. For example, the message type may indicate a query from the client or a response from the server depending on its value. A value of **0XFF (**Binary 11111111) may indicate a query from the server and **0x00 (** Binary 00000000) may be set to reflect a client query. The receiving side extracts the relevant information from a package by interpreting the various fields.

This project envisions you designing your own communication protocol for message exchange between the client and the server. **Please make sure that the designed protocol must adhere to the TCP payload restrictions.** All the communication between the client and the server after the initial handshake and assigning of the connected port must happen according to this protocol.

The following requirements must be met for protocol definition:

1. Message types
2. Packet Structure
3. Packet Field values

You are required to envision the interaction between the client and the server and create a complete protocol for this communication. Each client and server should have access to a range of communication messages (query, respond, terminate, etc.) which may be chosen by specifying a particular value of a field. The client and the server will handle the received messages according to their types. Similarly, they will construct messages adhering to the protocol.

Each of the requirements including message type packet structure and packet field values must be clearly defined in the code as well as the project report.

Initialization Phase

At the start, the server allows connection for multiple clients. After the connection, the client/s sends a query message to the server for obtaining the required information. The server queries the CoinGecko server using the API for the required information. Once received, the server reformats the received information according to the communication

Here are some useful links:

https://docs.oracle.com/javase/7/docs/api/java/io/DataInputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/DataOutputStream.html
https://docs.oracle.com/javase/7/docs/api/java/io/FilterOutputStream.html

Execution Scenario

The client and server are expected to reside on a single machine for simplicity of both execution and demonstration. The project envisions you connecting multiple clients and one server.

Project deliverables:

You should submit your source code and project report (in a single .rar or .zip file) via Blackboard.

Report: The report should start with a very brief description of the project. This should be followed by an explanation of the philosophies, especially the **client-server application layer protocol design and specifications.** Following this, you should provide an overview of the programming of the client and the

server side including connection with the API and data transmitted back to the client. Instead of attaching complete code, it is strongly recommended to attach code Snippets and images of the results of their execution. **The report should explicitly describe the test scenarios developed to evaluate the full range of features required for the CoinNet.**
Source Code: A .zip or .rar file that contains your implementation in a single Eclipse or Intellij IDEA project. If you aim to implement your project in any IDE rather than the mentioned one, you should first consult with TA and get confirmation.

Demonstration:

You are required to demonstrate the execution of the CoinNet Application for the defined requirements. Your demo sessions will be announced by the TA. Attending the demo session is required for your project to be graded. **The on-time attendance at the demo session is considered a grading criterion.**

During your demonstration, you will be asked to demonstrate **at least two** clients being connected to the server at the same time. During this time, you will be asked to display all the operations of the application through a series of questions. Please prepare the codes in an appropriate manner to answer the range of features as defined for the CoinNet application.

You have the creative freedom to present any additional features you may have built into the application in any way you deem feasible.

Good Luck!