Serhat Coşkun – 64152
Networking – Assignment 2

Introduction

In this project specification of the Structured Messaging Protocol, an example of its usage through the Coin Gecko Application, and details of running the Coin Gecko application will be explained.

Readme

The project consists of 2 java files named server and Client with java extension. These files implement the chat application's client and server. To run the application either compile using the javac <filename> command. The general run time of the application is as follows:

1- Run the Server application using default port 9998 just typing java Server or run on the desired port by typing java Server <port-number> <timeout>.

2- Run the Client applications within the same port as the server using the same syntax java Client <port-number> <server-IP>. Note to run on the same computer server-IP needs to be localhost.

3- Send a message from the client

4- Send a message from Server

5- Repeat 3 and 4. To exit, type the exit command from either of the applications.

2* If any client does run in <timeout> seconds server times out, with a descriptive message.

3* If the client does not sends a message for <timeout> seconds, the server timeouts and informs the client by sending a timeout command. If the client tries to send a message at this point it gracefully exits with information about the server's timeout.

Protocol Design: Structured Messaging Protocol (SMP)

Structured Messaging Protocol (SMP) is an application protocol that aims to be the choice of protocol for application developers. The main audience is the developers who want to have simple, lightweight messaging or control protocol that is easy to integrate with external API sources. The design of the SMP follows a similar approach to HTTP, but it emphasizes the following principles:

- Easy integration: with the rise of JSON message format most of the APIs are using JSON to respond to requests. To comply with this trend, SMP structures **Message Data** as JSON, hence there is no need for further processing from the server side.
- Structured requests: any SMP request must be in the form of JSON. All types of messages must use JSON to send requests, responses, or special messages. In contrast to
- Lightweight: SMP is lightweight and designed for small-size data transfers, it can transfer at most 2Mb of data.
- Un-opinionated interaction: since specifics of the commands are only at the level of request/response/termination/error. Users of SMP can design more complex interactions by utilizing the Message Data section. Under Message Type=Request, the Message Data section behaves like a terminal. Thus, users can easily extend the SMP functionality.

Structure Messaging Protocol's messages have the following structure:

Message Type [1 byte]	Sequence Number [1 byte]	Message Data [16 bytes]

The Message Data field represents the information or command that will be sent. It has a size of 64 bits and must comply with a JASON-like format. This approach enables the protocol to seamlessly integrate with different API sources. In addition, messages longer than 16 bytes, will be split into multiple packets, where the position of the packet is specified by the sequence number.

The Message Type field specifies the type and associated behavior of the message. Currently, there are 4 message types:

1. Request: This type specifies that message should be treated as a command, by the receiver. The receiver can interpret the Message Data according to its specification. In this project, an example of how to specify a request command structure will be given.
2. Response: Under this type, data is simple JSON that can be parsed by the client.
3. Termination: This signals the intentional termination of the connection between the client, and server. Under this message type, the data section can include the reason for termination to inform the server.
4. Error: This typically signals the occurrence of an error that resulted in enforced termination of the connection. The termination message type data section can be used to comment on the error.

The Sequence Number is a field that represents the position of a packet about the initial packet. It is of type UnsignedInt8, hence takes only one byte. As a natural extension of the 1-byte structure, at most 2^8 packets can be sent for single data transfer. In addition, the zeroth packet informs the requester about the number of packets that will be sent. Through this mechanism total number size of the data that can be sent is equal to 2 MB.

SMP in Practice

For the implementation of the application server and client, SMP is utilized. Since most of the details of the SMP are already mentioned, specifics of the different message types will be explained.

Request

In the Coin Gecko Application client-side requests information from the server side. As an un-opinionated protocol, SMP does not implement the specifics of the interaction between the server and client side. However, a common and stable approach is to use the following format to transfer commands in the data section.

Data section: {command: theCommand, arguments: {arg1: value1, arg2: [multiValule1, multiValue2]}}

In this section, the client will specify the command to the command object, and the argument to the arguments object. This approach enables an easy and interpretable method of requesting. Specifically, for the Coin Gecko application, the user will have the following 2 commands:

1. **Pricing**: {command: pricing, arguments: {coinId: [id1, id2, id3]}}
2. **Listing**: {command: listing}

Serhat Coşkun – 64152
Networking – Assignment 2

Response

In the Coin Gecko, the Application response side will be returning the original JSON string from the Coin Gecko external server directly since this is encouraged by the SMP. It is the responsibility of the client side to interpret the JSON for its usage.

Termination

For the termination signal, the client will be sending an empty JSON.

Error

For the error signal, the server will be sending an empty JSON.

Overview of the Implementation