Application Protocol

Overview

This document describes the application protocol utilized in a client-server program for conducting basic checks and calculations with integers. The protocol facilitates communication over TCP between the server and multiple clients.

Design Overview

Server

The server is implemented in Python and employs the socket and select modules to manage multiple client connections concurrently. Users must authenticate before utilizing the server's services. The server processes commands such as calculations, finding maximum values, and prime factorization. It ensures secure communication, proper error handling, and efficient resource management.

Client

The client is implemented in Python and provides an interactive command-line interface. Users can authenticate and send requests to the server, including mathematical calculations and other supported commands.

Application Protocol Definition

1. Message Format:

   Messages exchanged between the client and server follow the format:

   < message_type> <message_body>\

   - <message_type> A single-digit integer indicating the type of message.
   - <message_body> The content of the message, formatted based on the message type. Each message ends with a delimiter (\).

2. Message Types:

Client-to-Server Messages:

Authentication Requests:

- <message_type>: 0 - Signaling to the server that an authentication message has been transmitted
- <message_body>: The username / password entered by the client.

Service Request:

- <message_body>: The payload of the message adjusted to the <message_type> requested by the client.
- **Available requests:**

  Mathematical Calculation:

  - 1 num1 op num2\
  - Preforms num1 op num2
  - Available operations are: (+, -, *, /, ^).

  Find Maximum:

  - 2 num1,num2,num3,…\
  - Finds the maximum value in a list of integers.

  Prime Factorization:

  - 3 num\
  - Gets the prime factors of the given Int.

  Quit:

  - 4\
  - Terminates the session with the server.

Server-to-Client Responses:

Authentication:

- On success: Hi <username>, good to see you.
- On failure: N

Mathematical Calculation:

- On success: response: <result>.
- On failure: error: <message>.

Find Maximum:

- the maximum is <value>.

Prime Factorization:

- the prime factors of <num> are: <factor1, factor2,...>.

Error Handling:

- Invalid input or protocol violations result in client disconnection.

Server-to-Client Messages:

Welcome Message:

- <message_body>: A welcome message sent by the server when the client initially connects.

Login Response:

- On success: Sends a greeting message to with the given username.
- On failure: error: N - indicating failed login

Service Response:

- <message_body>: The result of the requested service or an error message.

3. Running the Programs:

1. Prepare a text file with user credentials in the format: ‘username password’ in each line

2. Start the server - python numbers_server.py users.txt [port], port is optional if not provided than 1337 is used

3. python numbers_client.py [hostname] [port], if not provided than localhost and 1337 are used.

Example Usage:

- python numbers_server.py users.txt 1337
- python numbers_client.py 127.0.0.1 1337

•

Sample Interaction:

1. Login:

```
Welcome! Please log in.
Please enter username in the format: 'User: username'
User: alice
Please enter password in the format: 'Password: password'
Password: password123
Hi alice, good to see you.
```

If login fails:

```
Welcome! Please log in.
Please enter username in the format: 'User: username'
User: alice
Please enter password in the format: 'Password: password'
Password: wrongpassword
Failed to login.
```

2. Commands:

Calculation:

- Please enter your command: calculate: 5 + 3
- response: 8.

Find Maximum:

- Please enter your command: max: (3 7 2 10)
- the maximum is 10

Prime Factorization:

- Please enter your command: factors: 28
- the prime factors of 28 are: 2, 7

3. Quit:

Please enter your command: quit

1.