Unreliable HTTP Server and Client Testing

1. Introduction

The task was to create an unreliable HTTP server using Python that simulates various outcomes when a request is made to the /getbalance endpoint. These outcomes occur with specific probabilities:

- **20% chance**: The server times out and does not respond.
- **20% chance**: The server returns an HTTP 403 (Forbidden) error.
- **10% chance**: The server returns an HTTP 500 (Internal Server Error).
- **50% chance**: The server successfully returns an HTTP 200 response with fake financial data.

The server also logs each request (including the outcome and the client IP) and prints these logs via a /**getlogs** endpoint in JSON format. A client was developed to test the server by sending repeated requests to the /**getbalance** endpoint and fetching the logs using /**getlogs**.

2. Implementation Overview

2.1 Server Implementation

The server was implemented using Python's http.server module, which provides a simple HTTP request handler. The logic was extended to simulate random events on the /getbalance route and serve logs on the /getlogs route.

- **Timeout Handling**: When a timeout is simulated, the server simply sends again request to get back response.
- **Error Codes (403, 500)**: For error cases, the server responds with the appropriate HTTP error code (403 or 500) using the **send_error** method.
- **Successful Response (200 OK)**: For successful requests, the server responds with fake financial data in HTML format and a 200 OK status code.
- **Logging**: Each request (including the client IP, timestamp, and outcome) is logged in both a log file and in-memory for the /getlogs response.

2.2 Client Implementation

A Python client was implemented to send repeated requests to the server. The client uses the requests library and has the ability to retry in case of a timeout.

- **Request Behavior**: The client sends requests to the /**getbalance** endpoint and logs each outcome. If a timeout occurs, the client automatically retries the request, which can result in multiple log entries for a single request from the client.

- **Log Fetching**: After sending a batch of requests, the client retrieves the logs from the /**getlogs** endpoint and displays them.

2.3 Client Script (client.sh)

A shell script (client.sh) was created to make it easier to invoke the client. The script accepts the server's IP address and port as arguments, ensuring that the client can connect to servers running on different machines or ports.

3. Testing Procedure

To evaluate the reliability and performance of the system, I increased the number of client requests to **100** to test the distribution of the outcomes. Here's a breakdown of the testing setup:

- **Total Requests**: 100 requests were sent to the /**getbalance** endpoint.
- **Expected Outcome Distribution**:
  - 20% timeout (around 20 requests)
  - 20% HTTP 403 (around 20 requests)
  - 10% HTTP 500 (around 10 requests)
  - 50% HTTP 200 OK (around 50 requests)

The client also fetched the logs after the requests to analyze the results and compare the actual distribution to the expected probabilities.

4. Test Results

4.1 Observed Outcome Distribution

After sending 100 requests to the server, the observed distribution of outcomes was close to the expected probabilities:

Outcome	Expected Count	Actual Count
Timeout	~20	21
HTTP 403	~20	20

HTTP 500	~10	12
HTTP 200 OK	~50	47

4.2 Conclusion on Outcome Distribution

The observed counts are relatively close to the expected probabilities. As noted, with a small sample size (e.g., 100 requests), there will always be some variance, but as the number of requests increases, the actual distribution will converge more closely to the expected 20%, 20%, 10%, and 50% probabilities.

Example from logs:

```
2024-10-23 23:55:31,705 - 192.168.100.59 - 200 OK
2024-10-23 23:55:32,716 - 192.168.100.59 - 403 Forbidden
2024-10-23 23:55:33,727 - 192.168.100.59 - 500 Internal Server Error
2024-10-23 23:55:34,738 - 192.168.100.59 - 200 OK
2024-10-23 23:55:38,745 - 192.168.100.59 - Timeout occurred
2024-10-23 23:55:39,766 - 192.168.100.59 - 200 OK
2024-10-23 23:55:43,815 - 192.168.100.59 - Timeout occurred
2024-10-23 23:55:44,824 - 192.168.100.59 - 403 Forbidden
```