Discussion on Feedback and Changes Made

Gia Bao Hoang - a1814824

Initial Feedback Received:

Our initial project submission received several feedback points, notably:
1. The need for a detailed explanation about the Lamport Clock implementation and the testing methodologies in the README file.
2. Errors encountered while executing the `make test` command.

Based on this feedback, I undertook a comprehensive review of the project and made several changes to improve its functionality, robustness, and documentation.

Changes Made and Rationale:

1. LoadBalancer Improvements:

- **What:** Introduced the ability to run multiple instances of Aggregation Servers (AS) concurrently (replicas).
- **Why:** This provides fault tolerance and improves the system's ability to handle a larger volume of requests. The LoadBalancer now distributes incoming requests based on a round-robin fashion, ensuring even distribution and load sharing.

2. GETClient Updates:

- **What:** Enhanced the client to accommodate various server name and port number formats.
- **Why:** This adds flexibility and makes the system more adaptable to different deployment scenarios ("http://servername.domain.domain:portnumber", "http://servername:portnumber", and "servername:portnumber").

3. Connection Retry Mechanism:

- **What:** Implemented a retry mechanism for both GETClient and ContentServer.
- **Why:** Enhances system robustness by attempting to re-establish connection in case of transient failures.

4. LamportClock Implementation:

- **What:** Implemented the LamportClock, with each Aggregation Server (AS) having both a local and a shared clock. These clocks synchronize at every request reception and response dispatch to ensure partial ordering of events.
- **Why:** This ensures that all events in the system are ordered consistently, preventing potential conflicts and ensuring data integrity.

5. DataStoreService Introduction:

- **What:** Introduced the DataStoreService to manage data storage, timestamps, and file operations. Implemented a locking mechanism to prevent concurrent access issues.
- **Why:** Ensures data consistency, prevents race conditions, and improves the overall robustness of data operations.

6. Comprehensive Testing:

- **What:** Fully implemented testing mechanisms which now run without errors. Though some exceptions may appear due to the test nature, they do not interfere with the testing process.
- **Why:** Testing ensures that the system functions as expected. Proper testing reduces bugs, ensures data integrity, and confirms the reliability of the system. The provided tests now also meet the suggestions from the MyUni announcement, ensuring a higher degree of reliability.

Conclusion:

The feedback received was instrumental in guiding our improvements. By addressing each feedback point systematically, we've made our system more robust, versatile, and user-friendly. We are confident that these enhancements not only rectify the issues pointed out but also position our system better for any future extensions or challenges.