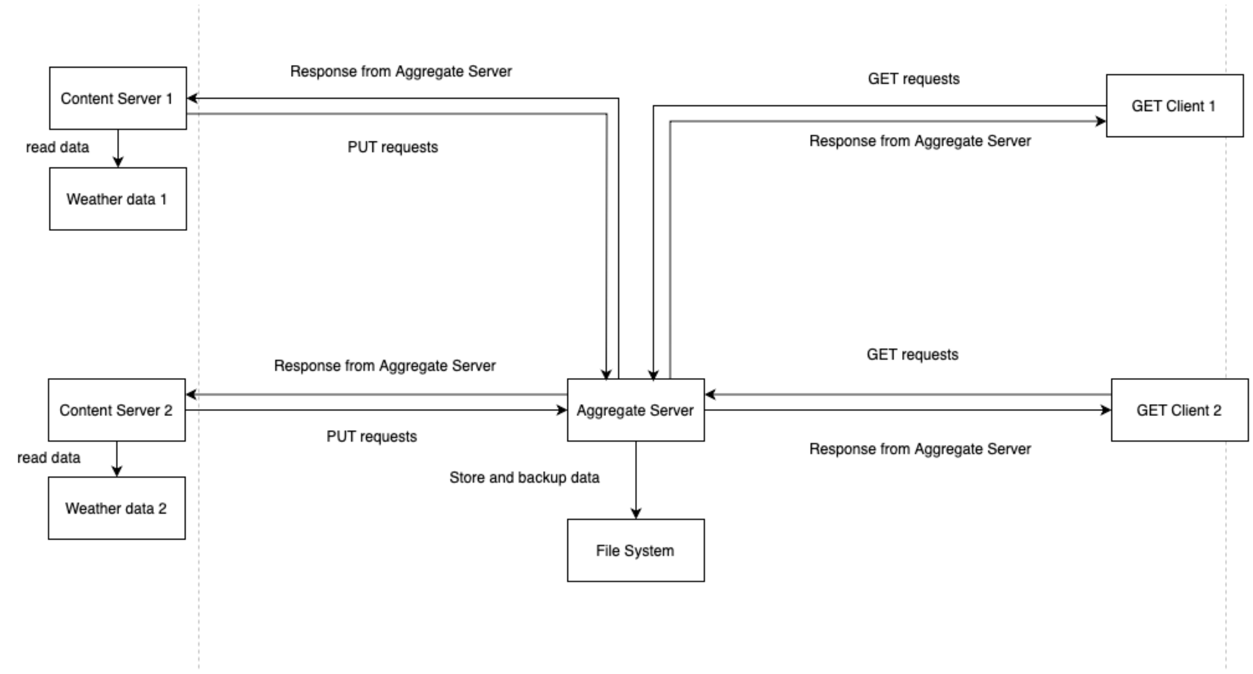
Name: Lap Van Quan

ID: 1895135

Design Sketch

1. System design diagram



2. Functional analysis

Aggregation Server

- Accept **PUT** requests from Content Servers.
- Accept **GET** requests from Clients.
- Maintain persistent weather data storage.
- Remove outdated/expired data (older than 30s or disconnected servers).
- Handle concurrency (multiple GET/PUT requests at once).
- Implement Lamport clock for ordering requests.
- Return correct HTTP status codes (201, 200, 204, 400, 500).
- The server can recover from failures (crash, unintended shutdown).

Content Server

- Read weather data from a local file.
- Convert the data into **JSON format**.
- Have Lamport clock to stay synchronize with server.
- Handle acknowledgments from Aggregation Server.

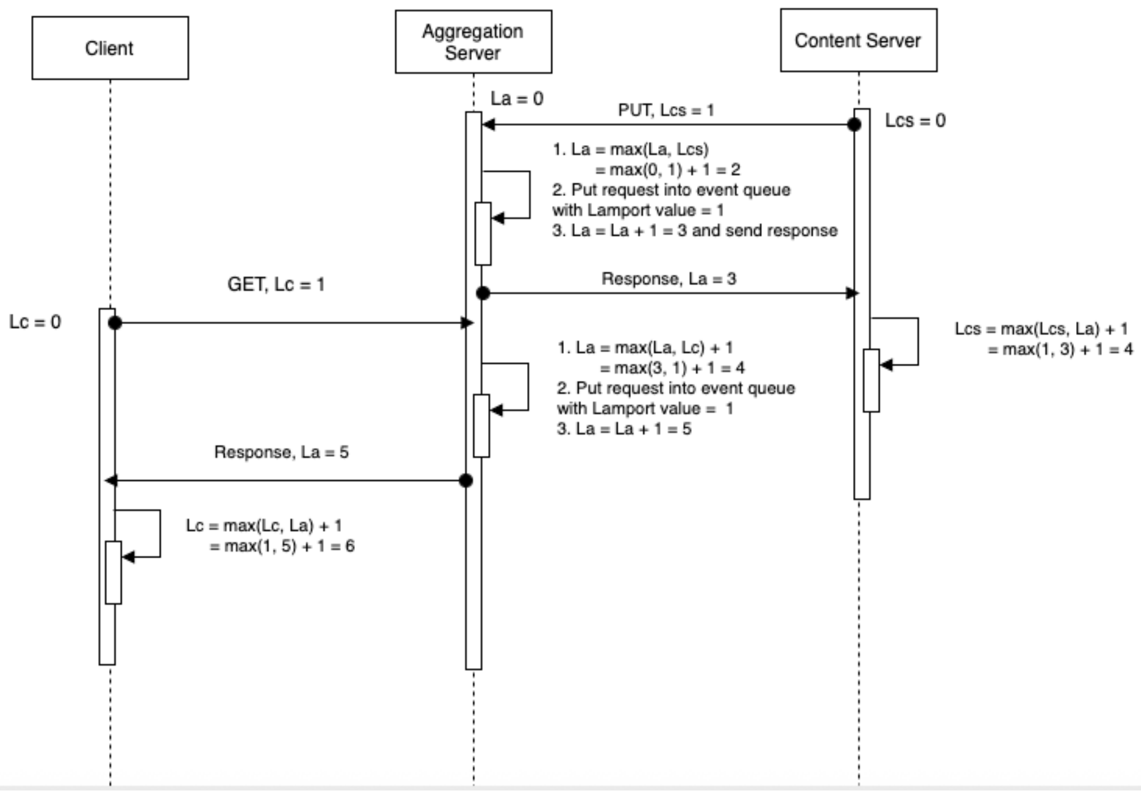
- Retry or recover from failures.

GET Client

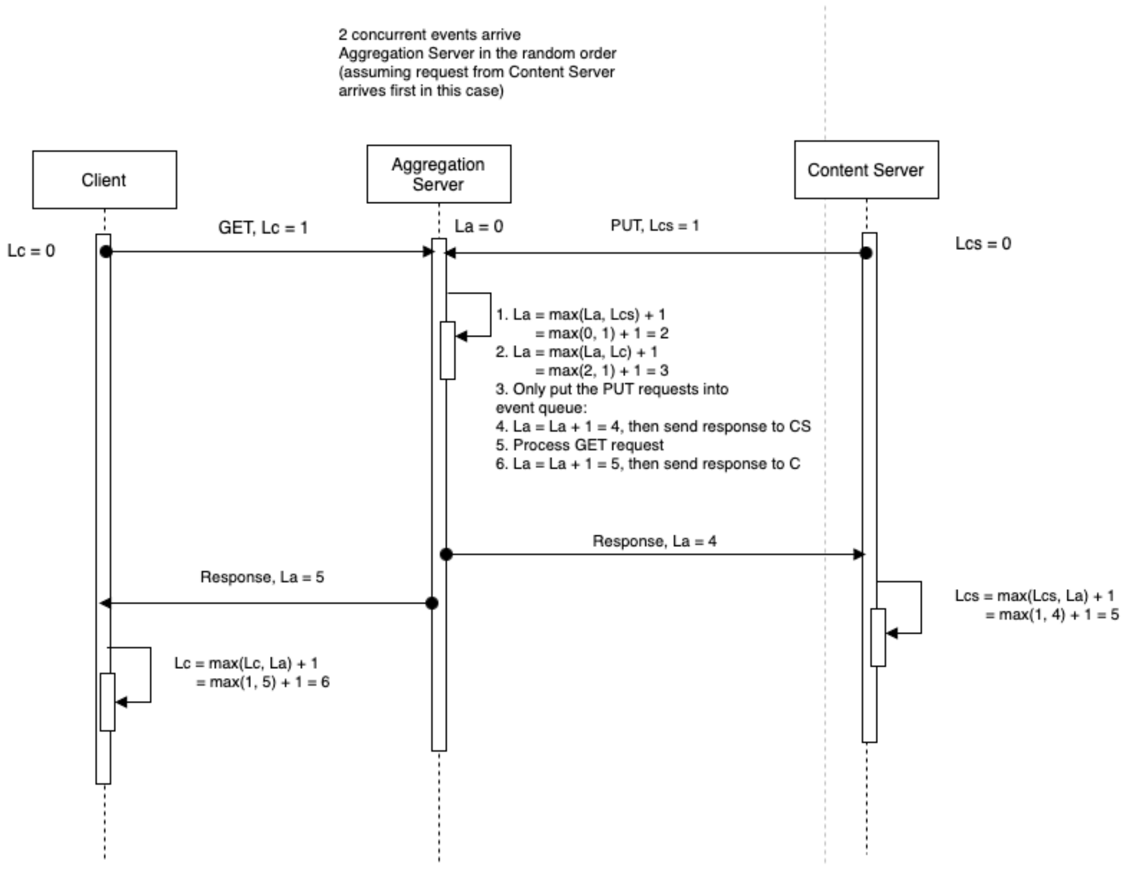
- Send GET request with Lamport clock to Aggregation Server.
- Receive JSON response from Aggregation Server.
- Have Lamport clock to stay synchronize with server
- Parse JSON and display data line-by-line.
- Handle failures, retry when server unavailable.

3. How the Aggregation Server will handle sequential and concurrent requests

Handle sequential requests



Handle concurrent requests



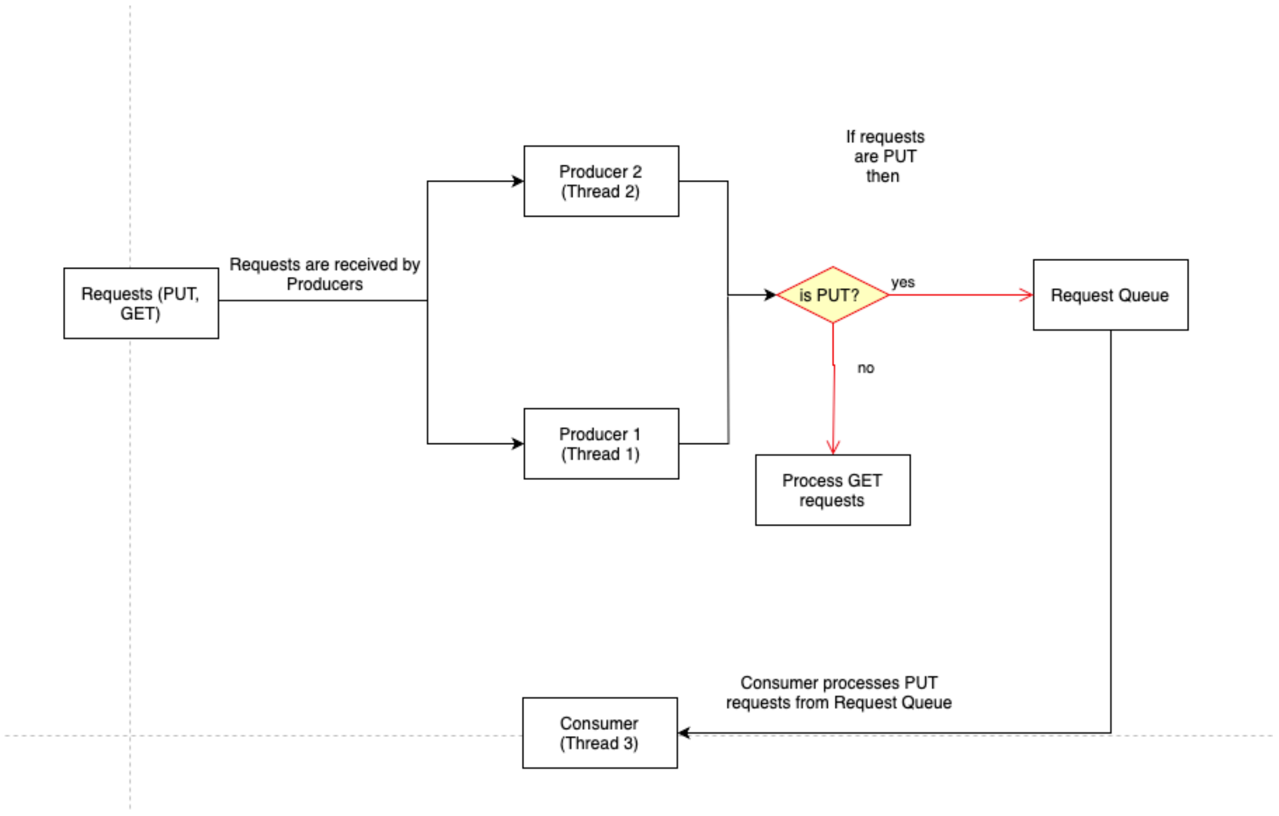
4. Multi thread interactions

- To ensure thread interactions are safe, I use data structure for concurrent (e.g. ConcurrentHashMap,…) and Producer - Consumer approach to solve concurrent requests.
- Additionally, I also have another thread for managing the expiration of data when the Aggregation Server could not connect to Content Servers in 30 seconds.

5. How Producer - Consumer approach works in my Aggregation Server

- Each Producers and Consumer are in separate thread.
- There is also another thread for managing the expiration of data.
- When a request arrives Aggregation Server, the Producer will receive the request. and classify its type to have a corresponding process.
- If it is a PUT request, the Producer will enqueue it into the Request Queue.
- If it is a GET request, the Producer will send back the response with weather data to client.
- At the same time on another thread, the Consumer dequeue request from Request Queue and process it.

- Processing PUT request by this way will ensure the PUT requests maintaining their arriving order and avoid blocking the Producers threads from receiving other requests.
- The thread used for managing expiration of weather data will run in background every 10 seconds to remove expiry data.



6. Testing

- I used the Integration Test to test the interactions between Aggregation Server, Content Servers and GET Clients.
- I also used Unit Test to test the Lamport Clock class and Json Util class separately to ensure it works as expected.