Architecture Overview

The system consists of three main components:

1. **Aggregation Server**
2. **GET Clients**
3. **Content Servers**

Aggregation Server

Responsibilities:

- Handle HTTP requests from GET Clients and Content Servers.
- Manage and store weather data.
- Maintain Lamport clocks for ordering PUT requests and consistency.
- Expire and remove old or stale data.

Components:

- **Request Handler**: Processes incoming HTTP requests (GET and PUT).
- **Data Storage**: Persists weather data (could be a file or a database).
- **Lamport Clock Manager**: Manages Lamport clock timestamps for requests.
- **Data Expiry Manager**: Periodically checks and removes stale data.
- **Error Handling**: Manages responses and errors.

GET Clients

Responsibilities:

- Request weather data from the Aggregation Server.
- Display the data in a human-readable format.

Components:

- **HTTP Client**: Sends GET requests and receives responses.
- **JSON Parser**: Parses JSON responses and formats data for display.
- **Lamport Clock Manager**: Manages Lamport clock timestamps for requests.

Content Servers

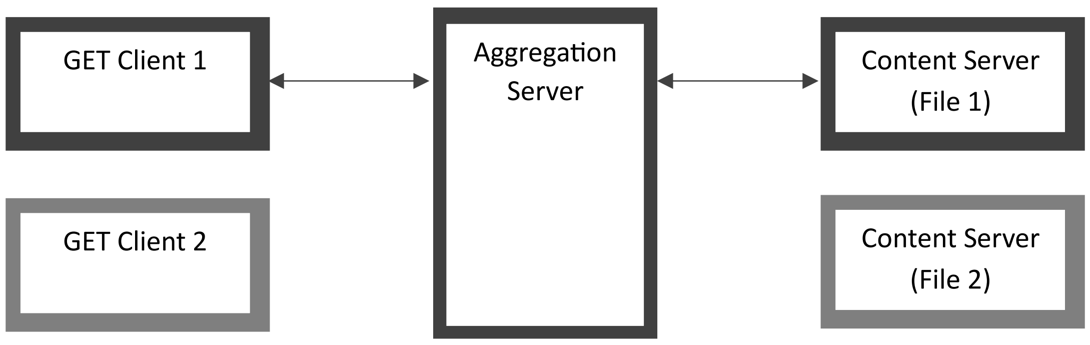
Responsibilities:

- Read weather data from local files.
- Send weather data to the Aggregation Server using PUT requests.

Components:

- **File Reader**: Reads and converts local weather data files to JSON.
- **HTTP Client**: Sends PUT requests with weather data.
- **Lamport Clock Manager**: Manages Lamport clock timestamps for requests.

Design Sketch



Detailed Design Considerations

Aggregation Server

Multi-threaded Interaction:

- **Request Handling**: Use a thread pool to handle incoming requests. Each request is processed in a separate thread to handle multiple simultaneous GET and PUT requests.
- **Data Consistency**: Synchronize access to shared data structures (e.g., weather data storage) using synchronization mechanisms (e.g., synchronized blocks or ReentrantLock).
- **Deadlock Prevention**: Ensure that locks are always acquired and released in a consistent order. Avoid nested locks where possible.

Components Interaction:

- **Request Handler** receives requests and forwards them to appropriate handlers (GET or PUT).
- **Lamport Clock Manager** ensures the proper ordering of requests and maintains consistency.
- **Data Storage** interacts with the Request Handler to read and write weather data.
- **Data Expiry Manager** runs periodically in a separate thread to remove stale data.

Data Flow:

- **GET Request**: Request Handler fetches data from Data Storage, applies Lamport clock validation, and sends it back.
- **PUT Request**: Request Handler updates Data Storage with new data, applying Lamport clock validation and handling concurrency.

GET Clients

HTTP Client connects to Aggregation Server, sends GET requests, and receives responses.

JSON Parser processes the received data for display.

Content Servers

File Reader reads the data from local files and converts it to JSON format.

HTTP Client sends PUT requests to the Aggregation Server with the weather data.

Testing Considerations

Unit Testing:

- Test individual components like JSON parsing, HTTP request handling, and file reading.

Integration Testing:

- Test interactions between GET Clients and Aggregation Server.
- Test PUT requests from Content Servers and verify data storage and retrieval.

Concurrency Testing:

- Simulate multiple concurrent GET and PUT requests to test data consistency and concurrency control.
- Ensure Lamport clocks are correctly implemented and handle ordering.

Failure Testing:

- Test server recovery after crashes or network failures.
- Verify that stale data is correctly expired and removed.