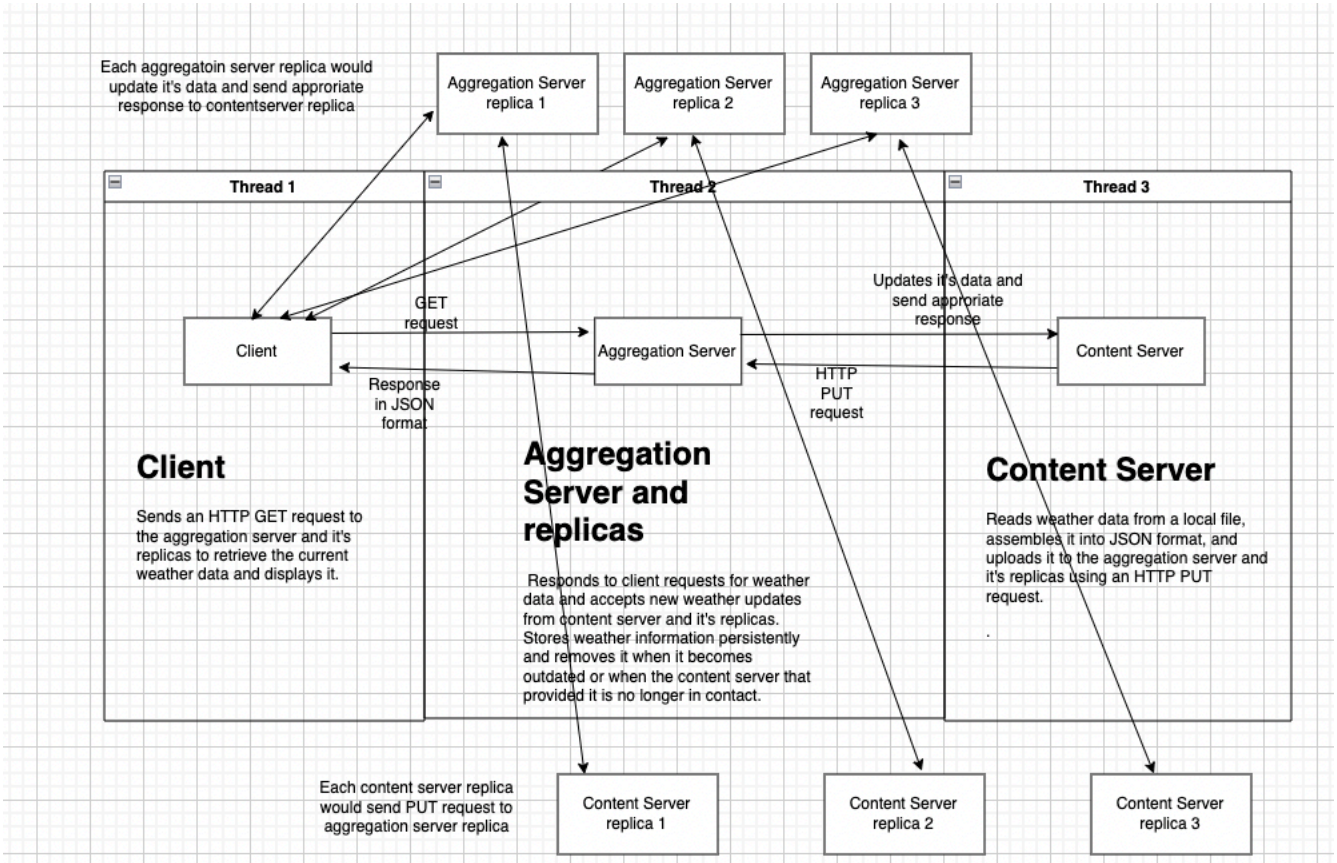

Making Multi-threaded interaction safe

Using locks for Aggregation server	Using semaphores for coordination b/w client and content server	Lamport clocks
We use a lock for the shared data structure of Aggregation server to ensure only one thread is accessing this at a time.	use a semaphore to ensure that the content server uploads new weather data to the aggregation server before GET client retrieves this data.	we can use Lamport clocks to ensure that the aggregation server processes PUT and GET requests from the content server and GET client in the correct order. Each component in the system would maintain its own Lamport clock, which would be updated as the component communicates with other components or processes events. The local Lamport clocks would be sent with every message or request, allowing the receiving component to update its own Lamport clock accordingly.

Testing

Unit testing	Integration Testing	Concurrency Testing	Failure Testing	Performance Testing
Test Aggregation server's ability to to process PUT and GET requests, update its stored weather data, and send back appropriate responses test the content server's ability to read weather data from a local file, assemble it into JSON format, and upload it to the aggregation server using an HTTP PUT request test the GET client's ability to send an HTTP GET request to the aggregation server, retrieve the current weather data, and display it.	test if the content server is able to successfully upload new weather data to the aggregation server and if the GET client is able to retrieve and display this updated weather data.	Test how the system handles multiple clients attempting to GET simultaneously and multiple content servers attempting to simultaneously PUT. Ensuring that the system is able to process these requests concurrently without any race conditions or unsafe mutations.	Test how the system handles errors and failures, such as a content server losing connection or sending invalid data. Ensure that the system is able to recover from these failures reliably and predictably.	Test the performance of the system under different loads, such as a large number of clients sending requests simultaneously or a large amount of weather data being uploaded by content servers. Ensure that the system is able to handle these loads without any significant degradation in performance.