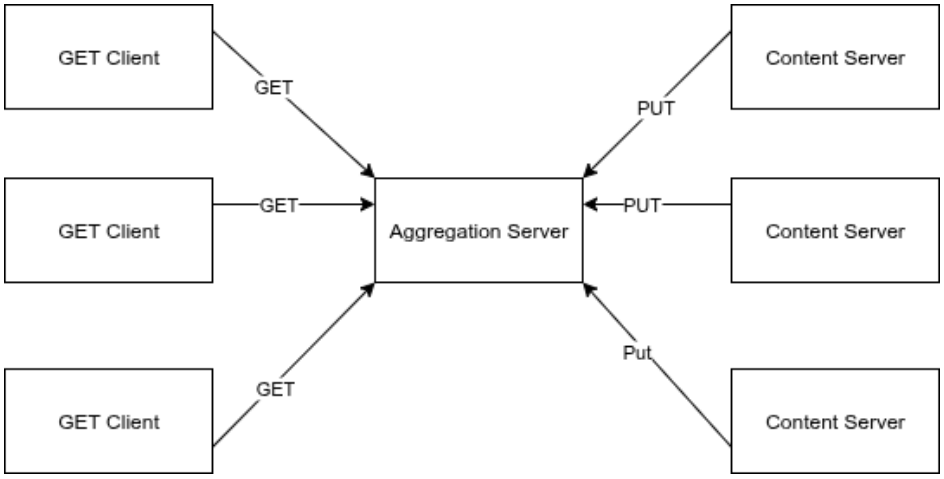
Design

Components

Basic Interactions:



Aggregation Server:

Accepts GET “/lamport” which returns the servers lamport clock

Accepts PUT “/” which takes a station data and stores it.

Accepts GET “/” which returns station data

Maintain consistent up to date JSON datastore:

- Assignment Description stated a single file to be used to store data however I plan to use multiple files to increase performance as with multiple files different station’s data can be read and written to at the same time. Tutor stated this was a valid plan.

Removes data that is older then 30 seconds

- Assignment Description also stated that data not modified in the past 20 updates should be removed, this would not be hard to implement i.e. use a LRU cache and after an update remove the data of the least recently used station. However, I do not believe this makes sense in the context of a weather station and overcomplicated implementation unnecessarily. Hence, only data older then 30 seconds will be deleted.

Content Server:

Initial Connect to server get lamport clock

Read Local Weather File

Parse to JSON

Send to Aggregation Server via PUT

Confirm Acknowledgement and update lamport clock.

In this implementation the data will just constantly be sent to the aggregation server every 5 seconds, however, in a real world use case the Content Server would probably wait until the file has changed and the Content Server Thread is notified to send the data.

GET Clients:

Initial Connect to server get lamport clock

Connect to server via socket

Send Get Request

Display Weather Data

Update lamport clock

Information Flow

There are three main Data transactions that occur, storing data, getting data and removing the stale data.

Storing Data: Content Server & Aggregation Server

Content Server:

1. Content server reads file and serialises to JSON.

2. Sends PUT with Lamport timestamp to Aggregation Server

Aggregation Server:

3. Aggregation server receives PUT
4. Validates JSON
5. Updates Clock and datastore
6. Send a 200 HTTP response

Fetching Data: GET Client & Aggregation Server

GET Client:

1. GET client sends request + Lamport clock.

Aggregation server:

2. Updates Lamport clock.

3. Reads persistent file.

3. Sends aggregated JSON.

GET Client:
4. Client displays data;

Data Expiry: Aggregation Server

Option 1: