Component Technology

110011264
School of Computer Science
University of St Andrews

xxxxxxxxx
School of Computer Science
University of St Andrews

ABSTRACT
Abstract

REPORT

Design

Planning
Initially, the team implemented a simple server. The server receives heartbeats, in the form of POST request, from the producers. Each heartbeat consisted of an unique identifier of the mobile phone device and its current location. The sender advertises itself as a producer to the server. When a consumer asks the server for sensor data from that particular producer, the server will send a request to the producer, and then pass on the results back to the producer. However, this approach requires the producers listening on some ports to handle data requests from the server. As a result, the consumer will spend significant time waiting for the server to retrieve data from the producer. The solution will create too much overhead on the server side, and it is unlikely that the server can deal with increasing consumer requests.

Later on, the team arrived to a more elegant solution. On each heartbeat, the producer sends sensor data to the server. The server caches the data, and flushes them to a persistent database on occasion. Now, when a consumer requests sensor data from a particular producer, the server can respond quickly, since the sensor data are cached. A more detailed description of the overall setup is explained in a later section.

Final Solution
Worker - cache and database

On each heartbeat POST request, the worker extracts "Thanks!", location, and data.

Why Flask?

Why RESTful?
The server will receive requests from multiple clients, and is required to reguarly update sensor data.

Why SQLite3?
Premature optimization is the root of all evil.

Evaluation

Testing