Smart Phone Grid

Undergraduate Student:Austin Aske
Faculty Advisor: Xinghui Zhao

Goals

- Create an actor framework for iOS
- Utilize sensors found in mobile devices
- Implement priority message passing within the Actor concurrency model
- Ability to dynamically allocation device priority
- Implement a Weather application using the newly created framework

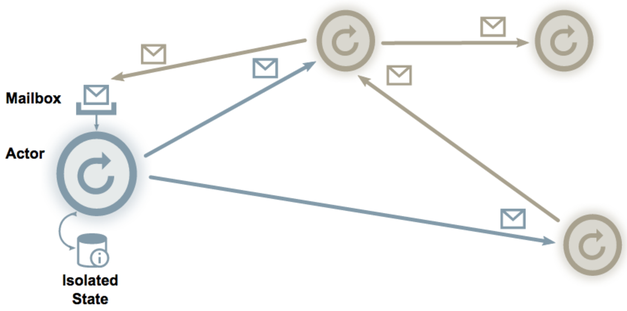
Actor Model

The Actor Concurrency model was theorized by Carl Hewitt in 1973 and motivated by the complexities found in parallel programming. An Actor in the model is the fundamental building block and adheres to a few basic theoretical rules.

An Actor is:
1. Block of computation
2. Uniquely identified
3. Isolated from external data
4. Has a “mailbox” for incoming messages

An Actor Can:
1. Create more actors
2. Send messages to known actors
3. Process messages
4. Update current state

A Message is:
1. Immutable
2. Sent asynchronously



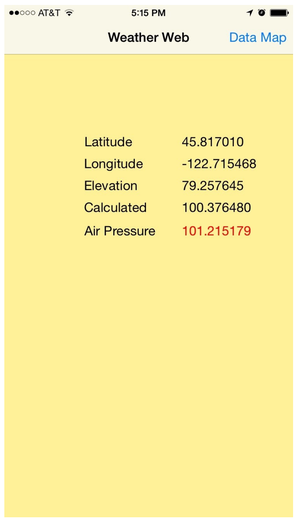
Weather Web Implementation

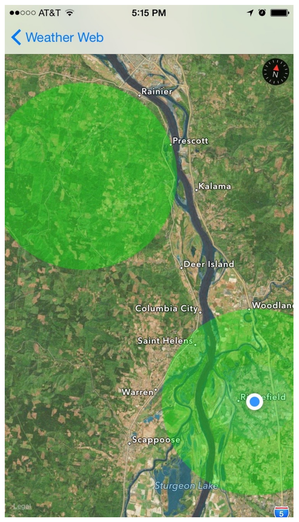
Application Overview:
Weather Web was created to take advantage of air pressure data from smart phones to better model weather trends. Air pressure measurements along with elevation are a decent indicator of weather conditions, but with millions of measurement points makes possible 3D time sensitive models. The data could be processed locally on the phones or used to feed more complex weather prediction algorithms found in current forecasting.

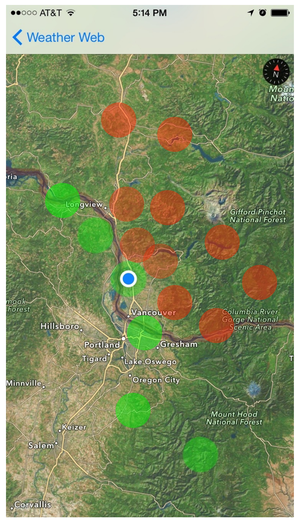
Design:
The iPhone and iPad application follows the MVC design pattern while using Actors computation model. While implementing Actors two major obstacles arose, isolating each actor’s state, and allowing each actor asynchronous behavior. Using Objective-C, which has message passing built into the language, to implement Actors became a natural extension.

Functionality:
Weather Web periodically sends your device’s barometer, elevation and gps information to the server for archiving. Meanwhile, showing the user the current measurements and calculated air pressure for comparison.

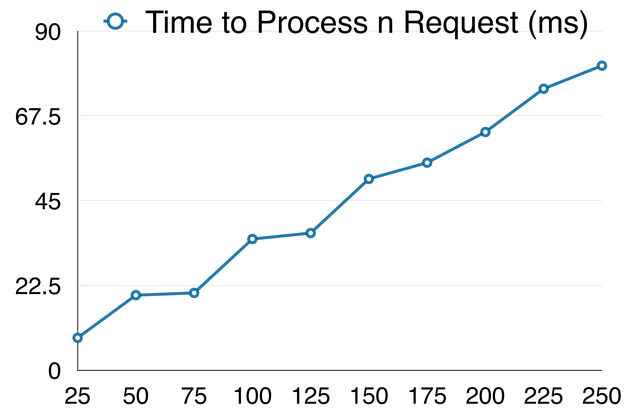
Golang Server:
The server built in Golang, acts as a master actor where all the request are handled. Currently each message can be of three different types; a new set of data to push to the database, a request to retrieve the map data, or a message to transfer to another iOS device.



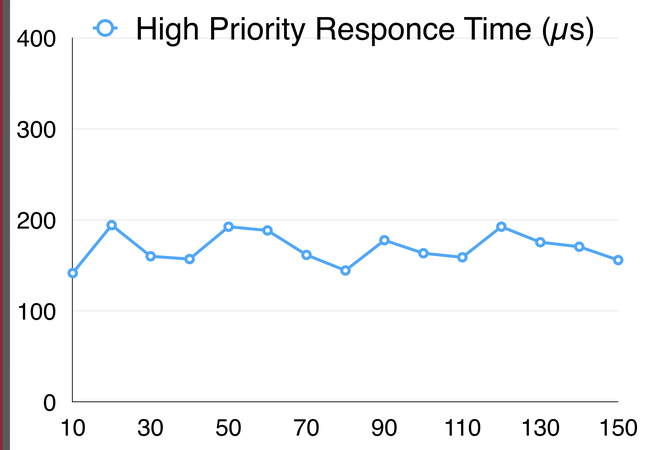




Results



The graph above was the result of measuring the total server side processing time for an increasing number of request per second. The results are linear, showing optimistic scaling characteristics.



The above graph illustrates the response time for a high priority request during n number of lower priority request. The fact that the data trends towards a constant means that the priority queue implementation was successful.