Data Broker for Collaborative Sensor Platform

Salil Raghunath Rajadhyaksha (email: srr7890@rit.edu)

Advisor: Dr. Leon Reznik

Rochester Institute of Technology, Department of Computer Science

Introduction
- There are roughly 2 billion smartphones being used worldwide as of 2016. These devices come mounted with a plethora of sensors. Such crowd funded sensor data could be used for a variety of purposes like research, analytics and even possible path breaking tasks like early detection of earthquakes.
- The purpose of this project is to develop a centralized broker for sensor data that can be accessed globally to acquire crowd sourced sensor data. The broker also masks the identity of the data source from the data users.

Technologies Used
- The data source which registered with the broker and passed on sensor information to it was developed as an Android application.
- The broker was developed using Java and the architectural paradigm used was the layered architecture. The broker services were exposed as RESTFUL API's.
- Mongo DB labs, the cloud based version of Mongo DB was used to persist the data of the broker.

Implementation
- **Step 1:** Volunteers willing to supply data to the broker register themselves with the data broker.
- **Step 2:** A data requester communicates its request to the broker specifying the location whose data it is interested in, the sensor information it needs and the quality of data it is expecting.
- **Step 3(a):** The broker checks its local repository for suitable sensor data with the requested data quality. If data is present locally, it is promptly returned to the requester.
- **Step 3(b):** If the broker fails to find suitable data in its repository but finds a source in its registry that could be accessed to satisfy this request, it contacts the source.
- **Step 4:** After the source returns the necessary information the broker completes the request.

Architecture
- **Identify suitable device and fetch data:** If the suitable data is not present locally, the data broker requests data from the data source.
- **Return recently fetched data:** The data broker sends the newly fetched data to the data requester along with DQ parameters.
- **Return locally available data:** If the data is present locally, the data is promptly returned to the data requester.

Experiment Results
- The broker was deployed on a Tomcat instance and the data source Android application was installed on four Android phones for the purpose of these experiments.
- The experiment results proved that the turn around time increased drastically in cases where the data had to be fetched from the server.
- It can thus be concluded that the efficiency of the broker depends greatly on its capability to maintain optimum data quality in its local repository which in turn would enable it to respond to a greater number of queries without the need to contact any data source.

Conclusion
- The broker was deployed on a Tomcat instance and the data source Android application was installed on four Android phones for the purpose of these experiments.
- The experiment results proved that the turn around time increased drastically in cases where the data had to be fetched from the server.
- It can thus be concluded that the efficiency of the broker depends greatly on its capability to maintain optimum data quality in its local repository which in turn would enable it to respond to a greater number of queries without the need to contact any data source.

Future Work
- Developing the data source for IOS platform.
- Increasing fault tolerance of the broker by replicating the server.

References