Service-oriented middleware for mobile sensing

Goals

Modern mobile devices are equipped with various complex sensors – accelerometers, location, video, audio, temperature, altitude, etc. Not all of them provide scientific-grade precision or update frequency. However these disadvantage are often compensated by ubiquity and availability of such devices, allowing unprecedented deployment scale for scientific experiments.

It was decided that a producer must send the data only to one source. Implementing data broadcasting through the producer would not be acceptable, since mobile devices are often limited in bandwidth.

Design

Components

Registry

Broker

Producer

Consumer

Implementation

Key technologies

As described in the brief, the sensor data producer is deployed on a mobile device. This dictated a number of constraints that had to be considered while developing the system.

The technologies used to develop the producer had to be easy to deploy on a mobile device. In recent years, it has become increasingly popular to develop mobile web apps – applications that use device's browser as the execution environment. Such applications have an advantage of being easy to distribute and deploy on various platforms. HTML5 and Javascript, used to develop web apps, provide rich APIs for accessing sensor data – accelerometers, audio, video, etc. Thus, the project aimed to utilise technologies that are supported by browsers of modern mobile platforms.

WebSockets and WAMP

WebSockets is a protocol for full-duplex (two-way) communication through TCP. Additionally, compared to regular HTTP requests, WebSockets have a low header overhead (2 bytes compared to kilobytes in HTTP). Two-way communication and low overhead (thus, decreased latency and bandwidth usage) are advantageous for a data streaming platform.

WebSockets are supported by most of the modern desktop and mobile browsers (through Javascript APIs, without utilising any plugins), which makes it easy to use WebScokets-based communication in web-apps. Various libraries implement the WebSockets for other languages – Python, Java, Go, etc.

REST

The system operates through a REST API, the registry offering various methods to clients (e.g. '/register_producer', '/request_brokers') which are called via GET requests.

Components

Registry

Broker

Producer

Consumer

Testing

Reproducing tests

Future work

Conclusion