Smart Route Planning Using Open Data and Participatory Sensing

Vivek Nallur, Amal Elgammal*, and Siobhán Clarke

FutureCities, Distributed Systems Group, Trinity College, Dublin, Ireland,
{vivek.nallur, amal.elgammal, siobhan.clarke}@scss.tcd.ie

Abstract. Smart cities are not merely the infusion of technology into a city's infrastructure, but also require citizens interacting with their urban environment in a smart and informed manner. Transportation is key aspect of smart cities. In this paper, we present a smart route planning open-source system; SMART-GH utilizes open data and participatory sensing, where citizens actively participate in collecting data about the city in their daily environment, e.g., noise, air pollution, etc. SMART-GH then augments the routing logic with sensor data to answer queries such as 'return the least noisy route'. SMART-GH enables citizens to make smarter decisions about their daily commute, and subsequently improve their quality of life.

Keywords: participatory sensing, open-data, open-source, smart-city-routing

1 Introduction

There are a plethora of route-planning applications available to citizens today. Web-based applications (*e.g.*, Google Maps, Live Maps, Yahoo! Maps, etc.), specialized device-based applications (*e.g.*, TomTom, Garmin, etc.) and smartphone-based apps (*e.g.*, iOS Maps, Google Maps, etc.), all allow a user to plan a route from one point to another, calculating the fastest possible route, or shortest route. There are some that even offer to calculate the fastest route using 'current' traffic data. However, all of these applications rely on a centralized source of data, usually controlled by a single entity.

As cities get smarter, and more sensors are available in the urban environment, route planning can be augmented to use this sensor data. For example, elderly citizens, parents with young children, etc. would like to know which streets have the least air pollution or least noise, along their route? Currently, there are two roadblocks to answering such questions: (i) how do we get sensor data about city-scale locations into the hands of the citizens? and (ii) how can this aggregated data be utilized by individuals?

As an answer, this paper presents a case-study of an application that combines *participatory sensing* [4], *open-data* from government sources, and inte-

* Assistant Professor, Faculty of Computers and Information, Cairo University

grates it with open-source routing libraries to provide a *smarter* route. An application that provides a social good, if created without participatory sensing, would require considerable investment in a sensor network from either the government, or an enterprise. Even if this were feasible, smart-city applications due to their pervasive nature, raise issues of privacy [3] from government databases. Not only does open-source software and open-data help in the provision of a social good, but it also alleviates *some* of the privacy issues that are inevitable, in a smart-city application. The rest of this paper is organized as follows: Section 2 introduces concepts of aggregating city-level data, while Section 3 presents the structure and behaviour of our smart-city application. Discussion and future work are addressed in Section 4, followed by concluding notes in Section 5.

2 Participatory Sensing, Open Data and Smart Cities

2.1 Opportunistic/Human-Centred/Urban/Participatory Sensing

Traditionally, wireless sensor networks have been used by researchers and city administrators to collect data about specific aspects of, and specific locations, in a city. This method has the obvious disadvantages of: (i) being expensive to deploy, (ii) high maintenance cost due to field-damage, and (iii) requiring owner permission for installation

Given the penetration of smartphones [9], and the possibility of using smartphones as sensors, at least two of the three disadvantages above can be eliminated. Ethically, the third disadvantage continues to persist, but can be alleviated through education and transparency. Citizens must be motivated to donate data to the city, and thereby help create a city-level aggregation of the desired data. However, most citizens are only motivated to participate, if they perceive a benefit to contributing their time and data [4]. We posit that such sensor data, made available by individuals, can be leveraged easily and harnessed to provide a social good. For example, individuals with breathing difficulties can use localized pollution data to determine how to plan their commute to work. In this paper, we describe a case-study of a novel application that leverages open-data contributed by individuals, and allows them to reap the benefits of such contributions by enabling smart route planning, which incorporates participatory as well as open data sources into the routing algorithm.

2.2 Open Data

Many public institutions and governments release data being generated from their research and data gathering bodies. The EU releases its data from the EU Open Data Portal[1]. Ireland does not yet have a centralized portal that covers all departments, but it has announced a national action plan[2] that lays out a

[1] https://open-data.europa.eu/en/data/

[2] http://www.per.gov.ie/minister-brendan-howlin-td-publishes-irelands-first-open-government-partnership-national-action-plan/

roadmap for open-data to be released. Most open-data available in Ireland is through city councils' websites. Dublin city council releases data available on Dublin city through its *Dublinked* initiative[3]. Use of *open-data* is expected to provide macro-level insights into various aspects of a government, and a city's life. In particular, from this paper's perspective, it allows us to leverage existing environmental data that is available about Dublin City, and incorporate it into routing decisions. **Note:** the use of Dublin city in this paper is merely illustrative, and the system that will be described is in no way tied to any particular city.

2.3 Smart Cities

The term *smart city* is frequently used to connote the use of information technology by city authorities, in various domains ranging from smart-grids to waste-management to intelligent-transport, etc. to enable a better quality of life for citizens. The basic idea is that cities could (and should) actively leverage the latest technologies, to make informed decisions at the macro-level, and enable citizens to make intelligent individual choices about city resources. About 180,000 people move to cities everyday and it is predicted that by 2050, about 70% of the world's population will be living in urban agglomerations [8]. This will result in huge pressure on resources such as energy, water as well as exacerbate problems of congestion, waste-management, etc. In Dublin alone, the cost of congestion is estimated at 4.1% of the GDP [6]. In such a scenario, judicious and smart use of technology to ease problems in transport, healthcare, energy-usage and waste-management, etc. is of utmost importance. However, current technical solutions cannot simply be transplanted into a city, typically due to the following problems:

- **Multi-ownership:** Different parts of a city's infrastructure are owned by different agencies, and hence technical solutions developed by one are not necessarily interoperable with others. Also, different agencies have different goals, and it is difficult to create a system that can account for all possible goals
- **Scale:** Most decision-making mechanisms do not scale to city-scale entities, when information and goals are effectively de-centralized
- **Availability of fine-grained and yet aggregate data:** To be able to take decisions at a city-scale, fine-grained and up-to-date data needs to be made available on a continuous basis. This is currently difficult to achieve. Either data is not available, or decision-makers are flooded with data, but do not have a macro-level perspective.

In this paper, we focus on a smart-city application that circumvents the data availability issue. This application assumes that there will be no single data stream, allows individuals to contribute data, and then uses the aggregated data to provide a service back to individual citizens.

[3] http://www.dublinked.ie/

2.4 Issues

Smart-City applications tend to use technologies that are pervasive in nature. That is, sensors and sensor-networks that surround the user throughout the day. A smartphone is a perfect example of a sensor-based device that is both, uniquely identifiable as well as pervasively present. A smart-city application that uses participatory sensing is uniquely positioned to identify individuals and their habits, in ways that the user did not anticipate. In such a situation, it is imperative that the application make extra efforts to ensure that user-privacy is protected.

While governments are sometimes willing to open up data sources, there have been criticisms [7] about their intent in doing so. A major critique is the utility of releasing raw data, when the citizen is unable to interpret such data and make informed decisions. Helbig *et al.* report that most government websites are essentially data dumps with little thought given to usability, quality of data, or consequences of use [5]. In such cases, open-data merely becomes a way for public authorities to pay lip-service to openness, without actually achieving anything concrete.

3 Smart GraphHopper

Our smart-city application called Smart GraphHopper[4] (*Smart-GH* for short) uses GraphHopper[5], which is an open-source routing library that uses OpenStreetMap (OSM). The original GraphHopper can be used to plan either the *fastest*, or *shortest* routes[6] in the same way as its commercial counterparts (GoogleMaps, etc.). GraphHopper uses the *length* and *maxspeed* data embedded inside the OSM to calculate routes. However, Smart-GH extends GraphHopper to allow citizens to compare routes by evaluating different available sensor data, e.g., noise data, air pollution data, etc. It also enables better distributed deployment capabilities.

Participatory Sensing: A smart-city application relies on city-wide data. Participatory data could be a key enabler in providing the application with a large set of data points. For this purpose, we use NoiseTube [2], which is a mobile app, developed by *Vrije Universiteit Brussel*. NoiseTube uses sensors on smartphones and measures the ambient noise in the user's surroundings.

Open Data: Air quality monitoring data is available from Dublin City Council's *Dublinked* portal[7]. The city council maintains fixed sensor stations covering the major areas of Dublin city (centre, north, south, east, west).

In the next sub-sections, the structure, behaviour and technological choices to build and deploy Smart-GH are discussed in more detail.

[4] `https://github.com/DIVERSIFY-project/SMART-GH`

[5] `https://github.com/graphhopper/graphhopper/`

[6] In this paper, we refer to the original GraphHopper as *GraphHopper*, and to our extensions to transform it into a smart and distributed GIS system as *Smart-GH*.

[7] http://dublinked.com/datastore/datasets/dataset-279.php

3.1 Processing Data

Smart-GH supports both volunteer data as well as open data. The three essential elements required for *any sensor data* to be integrated into Smart-GH are: (i) *location* in the form of GPS coordinates, (ii) *value* read by the sensor and (iii) *timestamp*. Smart-GH requires each city to have a `config` file, that lists all the sensor-types that are available. In this case, Dublin city has two sensor types: (a) Noise (b) Air Pollution.

Noise data is stored on mobile phones as XML files that can be automatically uploaded to the server. Researchers in Trinity College Dublin (TCD) installed NoiseTube on their smart phones and started collecting noise data during their daily commute. Using NoiseTube's *API*, Smart-GH periodically pulls data about noise levels on Dublin's streets.

Air pollution data from Dublin City Council is available as a set of excel sheets, which were converted into `csv` format. Smart-GH currently supports three file formats: `xml`, `json`, and `csv`. These file reading mechanisms are inserted via a plugin mechanism, and can be extended to handle any other file-format, as desired.

Parsing and Filtering: For each sensor-type, the `config` file lists the name of a datasource, parser and a filter. The Parser is responsible for connecting to the datasource (web service returning json, or csv files), and collecting all the relevant data. The Filter is used to prevent fluctuations in the day-to-day collection of sensor data, from affecting the final value. The currently used filter is exponential weighted moving average, however any other filtering mechanism may also be simply plugged in, via a config file. There is a default no-op filter .

Reverse-Geocoding: Typically, each sensor reading is linked to a GPS location. To link them to the relevant OSM map ways (e.g. streets, roads, etc.), the Parser needs to obtain concrete way-ids for each GPS location in the respective files, a process known as *Reverse-Geocoding* [1]. For this, the Parser calls a geocoding web service[8], which takes a GPS-coordinate as input and returns concrete OSM way-ids.

Data Storage: The OSM information is now associated with the sensor value, and stored as a hash in Redis. This is done for both, the noise as well as the air pollution sensor. However, collecting the air pollution sensor data is less computationally intensive, since the sensor locations are fixed and *Reverse-Geocoding* becomes a one-off process.

Figure 1 presents a UML sequence diagram capturing the behaviour of the actors involved and the flow of logic. As shown in the figure, there are six system actors: 'Data Retrieval and Processing Daemon', 'Air Pollution Sensor', 'Parser', 'Filter', 'Reverse geocoding Web service' and 'Redis', and one human actor: 'Noisetube Participator'. The scenario is started by the 'Data Retrieval and Processing Daemon' getting the noise and air pollution readings from 'Noisetube

[8] `http://services.gisgraphy.com/street/streetsearch`

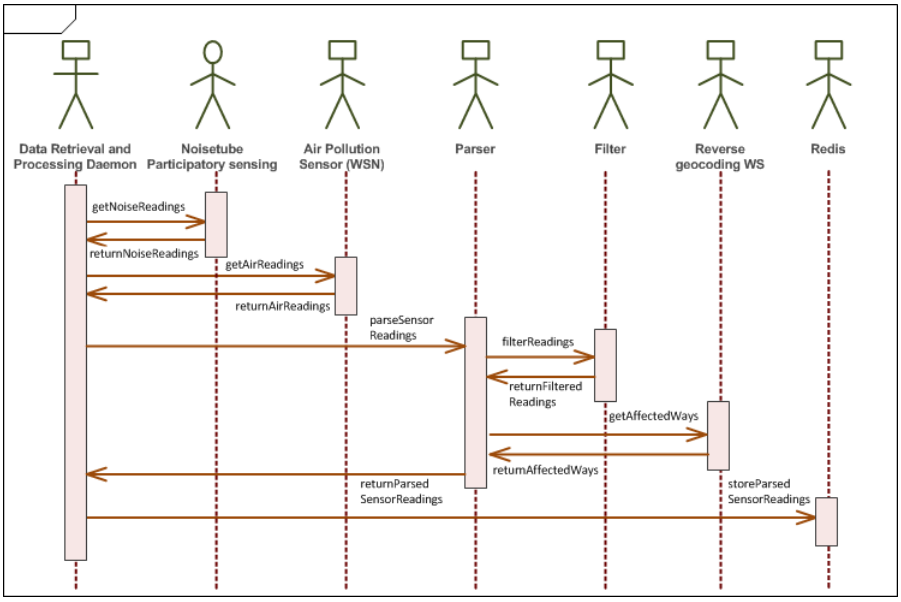

Fig. 1: UML sequence diagram of sensor data reading and parsing

Participatory Sensing' and 'Air Pollution Sensor', respectively. Figure 2 presents a finer-grained UML component diagram of this sensing component.

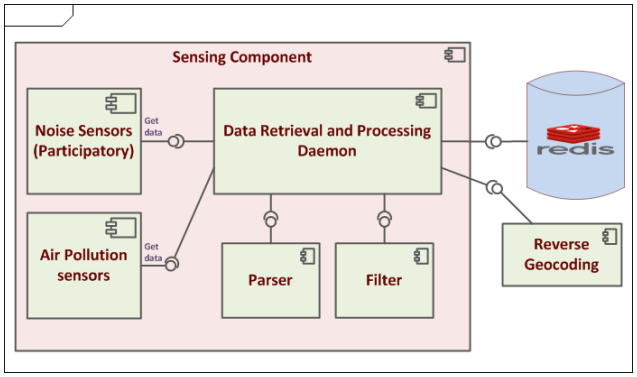

Fig. 2: UML component diagram of Smart-GH sensor data reading and parsing

3.2 Marrying the Sensors to Routing

As discussed above, Smart-GH extends the typical usage of GIS applications by integrating sensor data from various sources, which will then be used as the

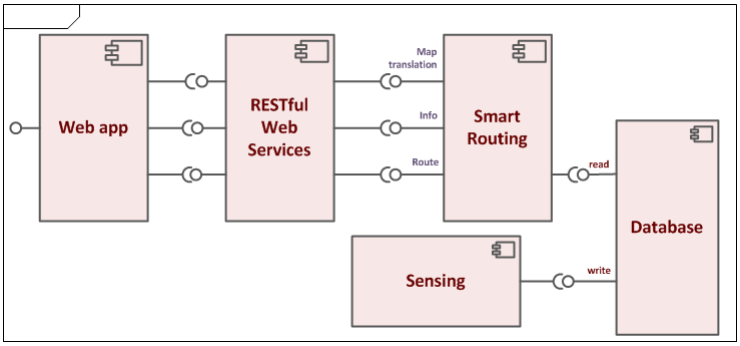

Fig. 3: UML component diagram of Smart-GH

basis of routing decisions. Smart-GH utilizes the GraphHopper routing engine for route planning. GraphHopper is a fast, efficient routing library that implements various routing algorithms, which given two (up to five) GPS locations, can calculate either the *fastest* or the *shortest* route, connecting these points. The fastest/shortest route is calculated based on speed limits/distance tags associated with OpenStreetMaps (OSM) ways (streets, roads). The problem of finding a fastest or shortest route becomes a minimization problem: minimum total time or minimum distance, respectively.

Enabling routing based on collected sensor readings required extension of the original GraphHopper library. First, the communication protocol was modified to allow arbitrary sensor types to be included as the user's preferred routing mechanism. In this case, 'noise' and 'air-pollution' are two additional values that are available to the user. Second, we modified the existing algorithms to perform routing based on stored sensor readings in Redis, instead of OSM tags. Figure 3 shows a UML component diagram of Smart-GH after augmenting it to enable smart routing. As shown in the figure, Smart-GH constitutes four components: 'Sensing', 'Database', 'Routing' and 'Web app'. The 'Sensing' and 'Database' represent the data sensing and parsing component and Redis database, as explained previously in Section 3.1.

Figure 4 presents a screenshot of the Smart-GH web interface, which visualizes a Bike route from Smithfield to School street(two places in Dublin), where the weighting is selected as 'Least Noisy', by dynamically visualizing noise data as a heatmap overlay. GraphHopper has a very simple (and limited) web interface that allows a user to only query for the fastest driving (car) route between two GPS locations. An *Android App*[9] has also been developed for SMART-GH. Both apps have been designed and developed to meet the following functionalities:

– Enable the user to request sensor-based weights on routes. Current options for Dublin city are: fastest, shortest, least noisy, least air pollution. We maintain a city configuration file for each city, capturing, among others, the types

[9] https://github.com/DIVERSIFY-project/SMART-GH/blob/master/SMART-GH-Android/platforms/android/ant-build/CordovaApp-debug.apk

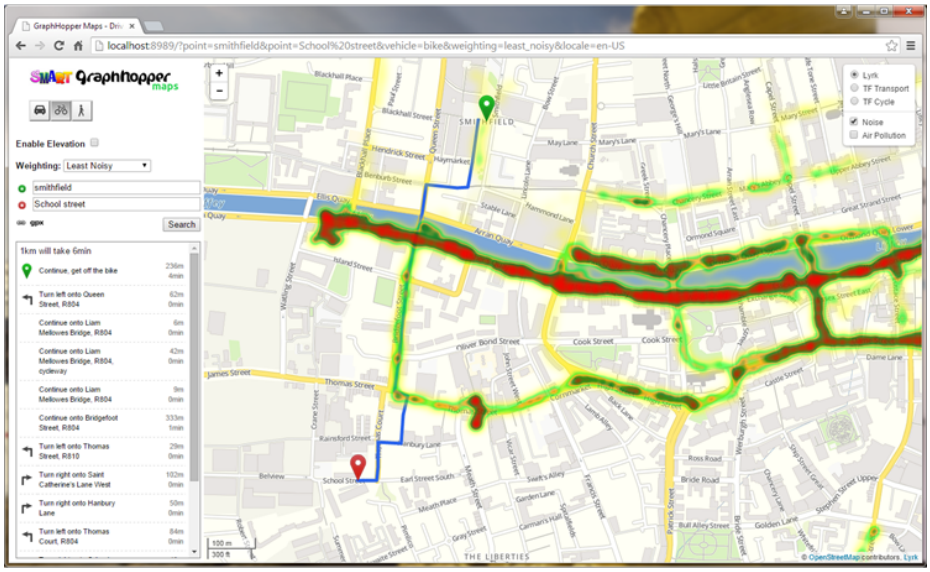

Fig. 4: Screenshot of Smart-GH web interface

of sensors available for this city. Based on available sensor data for each city, the web-interface is automatically configured to include the weighting specified in the relevant city config file. The appropriate config file is parsed based on the loaded map. For example, since `dublin.osm` map is loaded in Figure 4, `dublin.config` is parsed accordingly. Since `dublin.config` has two sensors corresponding to noise and air pollution data, the interface is amended to the drop-down list as shown in Figure 4.

– Support more than one vehicle type on the same running session. As shown in Figure 4, car, bike and foot are supported. These configuration parameters are specified in Smart-GH config file.

We modified the interaction model between the components to be *asynchronous* and *stateless*, and then wrapped the routing algorithms into Restful Web Services (WS). This allows the system to be scaled horizontally, to meet increased (web)traffic.

4 Discussion and Future Work

4.1 Performance

The deployment configuration shown in Figure 5 shows the sensor datastore on a separate machine, with Redis serving up the data. One of GraphHopper's biggest strengths is *speed*. Due to its unique caching strategy, it is able to provide routes over long distances ($> 100km$) very fast. Accessing sensor data, is an out-of-process procedure, and is therefore much slower than if only OSM tags are

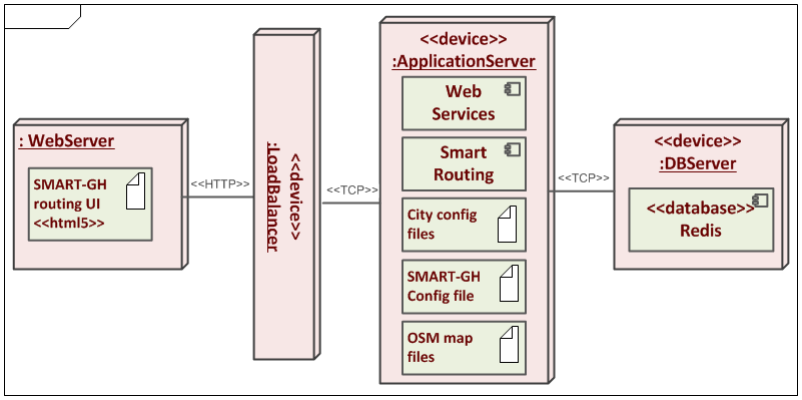

Fig. 5: UML deployment diagram of the distributed deployment of Smart-GH via WSs

used. However, in a smart-city usage scenario, route demand is rarely greater than 100km, while access to sensor data provides much more value. Also, due to data-retention/privacy laws, it might not be possible to always hold sensor data on Smart-GH's servers. We believe that flexibility afforded by an out-of-process access to a datastore is an acceptable trade-off to the performance-loss of in-process routing.

4.2 Security/Privacy

Participatory Sensing, in some ways, depends on the *kindness of strangers* to be effective. To acquire a critical mass of data and ensure continuous updates, it is essential to gain the trust of citizens by ensuring data anonymity and security. Even with an innocuous application like NoiseTube that makes no conscious effort to uniquely identify its users, it is fairly easy to correlate *usernames* (from the data collected) to the routes they have taken, and thus find out where they work, and reside. In the data-processing performed by Smart-GH, we strip away **all** user information, and work **only** with street locations and noise-levels. For a smart-city application, privacy is an aspect that must be explicitly and aggressively engineered in, since the default protocol of releasing open-data does not ensure this. Given that our entire software chain is made of open-source components, it is easy to verify that the data is anonymized, as early as possible.

4.3 Sensor-X

Currently, Smart-GH knows all the sensor-types that are present in a city. Although, for a new sensor-type, a new parser and filter can simply be plugged in through the `config` file, the routing algorithm still needs to know which sensor data it is dealing with. In the ideal case, users must be able to extend Smart-GH to handle different sensors without having to modify the routing algorithm. This

will allow different subsets of users in a city to add different kinds of sensors (and corresponding data) without having to contact Smart-GH developers. In other words, the application should be *self-extending*.

5 Conclusion

Smart cities use digital technologies to enhance performance, well-being and improve the quality of living. This will inevitably require the active participation of citizens. Transport, Energy, Healthcare, Water and Waste are key areas of smart cities' concerns. Smart-GH uses raw open-data collected about air quality, as well as noise measurements collected by individuals, and aggregates them to produce actionable information in the transport domain. This, we believe, is in the highest tradition of open-source and open-data where small individual contributions are combined to produce a social good. Citizens usually invest a substantial chunk of time in commuting, and enabling citizens to improve the quality of this time would directly improve their well-being, and their quality of life. To the best of our knowledge, Smart-GH is the first GIS system that adds such benefits to the traditional usage of common GIS applications.

Acknowledgment

This work was partially supported by the EU Project Diversify FP7-ICT-2011-9

References

1. M. Cayo and T. Talbot. Positional error in automated geocoding of residential addresses. *International Journal of Health Geographics*, 2(1), 2003.
2. E. D'Hondt, M. Stevens, and A. Jacobs. Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. *Pervasive and Mobile Computing*, 9(5):681–694, 2013.
3. R. Gallagher. Operation Auroragold: How the NSA hacks cellphones worldwide. `https://firstlook.org/theintercept/2014/12/04/nsa-auroragold-hack-cellphones/`, 12 2014.
4. M. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, 2007.
5. N. Helbig, A. Cresswell, G. Burke, and L. Luna-Reyes. The dynamics of opening government data: A white paper. *Centre for Technology in Government, State University of New York, Albany*, http://www.ctg.albany.edu/publications/reports/opendata/opendata.pdf, 2012.
6. IBM Institute for Business Value. Smarter cities for smarter growth, 2010.
7. R. Kitchin. *Four critiques of open data initiatives: The Programmable City.* Nov. 2013.
8. World Health Organization. Urbanization and health. *Bulletin of the World Health Organisation*, 88(4), 2010.
9. P. Zheng and L. M. Ni. Spotlight: The rise of the smart phone. *IEEE Distributed Systems Online*, 7(3), 2006.