CDME – Crowd-Sourced Data Mapping Engine

System that Analyzes, Mapps & Publishes Crowd-Sourced Data on Enviorenment Facts

S. Ruwanpathirana and I. Perera
Department of Computer Science & Engineering
University of Moratuwa
Moratuwa, Sri Lanka
ssrpsathira@gmail.com, indika@cse.mrt.ac.lk

***Abstract*—Availability of records on environmental factors like noise, temperature, and precipitation is important in making critical decisions concerning public safety and wellbeing. Traditional methods involving dedicated human personnel and equipment in capturing these data have been reliable, but extremely costly and time consuming. We propose a data collecting and visualizing framework based on crowdsourcing that is readily available, extensible, and virtually incurs zero cost. The crowd-sourced data mapping engine (CDME) presents an extensible back-end web application and a noise data collecting mobile application targeting analyses on noise pollution, which poses a significant concern especially in urban areas. The mobile applications runs as a service and updates the server with periodic noise data. The server accepts data updates and provides analytical functions such as graphs and heat maps.**

Keywords—CDME, Crowd-Sourcing, Mobile Device, Client-Server, Data Mapping, Data Analyzing, Environmental Facts, Monitoring, Noise Pollution

I. INTRODUCTION

Crowd-sourced data mapping engine (CDME) enables voluntary mobile users to act as data collecting agents and provides a web based analytical interface on data. The types of data, e.g., are noise levels, speed limit violations on roads, weather conditions, and public feedback on government services. Irrespective of the data type, the analytic engine provides an application program interface (API) that a client could use to access CDME services. The server is capable of producing visualizations along a few dimensions such as graphs and heat maps of the collected data.

Sri Lanka being a developing country is showing increasing noise levels especially in urban areas due to industrialization and rising number of vehicles. Previous studies show that noise, 65 dB(A) or above in particular, has significant long term effects on human health such as reduced memory rehearsal, increased aggression, decreased strategic thinking, progressive loss of hearing, and even insomnia [1]. These factors motivate us to couple CDME with noise level monitoring to enable study of noise pollution in Sri Lanka. We introduce a mobile client therefore to collect noise levels that will be processed and visualized by CDME. An added benefit of the mobile client is that a participating user gets the opportunity to observe his or her exposure to noise overtime. The usefulness of such people-centric data have been demonstrated in medical projects such as [2], in which children were equipped with sensors for air pollution to understand the factors affecting asthma. Therefore, we believe that measuring the impact of noise pollution, not only from a geographical point of view, but also through people's exposure will form a new social perspective and lead to further epidemiological studies to better understand the increasing urban noise pollution.

The rest of the paper is organized as follows. Section 2 describes the Android client, NoisyGlobe, which runs on user's mobile devices to collect environmental sound data. Section 3 discusses about CDME server, which acts as the analytical and visualization engine. Issues and possible improvements related with the project are presented in section 4. Section 5 presents the summary this work.

II. CDME NOISE CLIENT

NoisyGlobe is the mobile application that collects environmental sound levels using the device's external microphone, which is generally used to capture voice input when making phone calls. The application is initially targeted for Android based mobile devices since it covers a large percentage – 84% – of the total mobile devices at present [3]. NoisyGlobe will run as a background service without interrupting user's other applications.

A typical mobile device is not intended by design to be used as a sound level meter, thus requiring microphone calibration to reduce possible errors. While it would be ideal to be able to calibrate each device we propose calibrating at device's model level for practical reasons where CDME server could automatically adjust receiving sound levels without burdening the user. A mechanism for such automatic calibration is given under future work. The application is currently tested on HTC One X mobile device which runs Android 4.1.2 Ice-cream Sandwich operating system that has a device specific sound level offset.

A. Sound pressure level calculation

In CDME Android MediaRecorder provides an inbuilt function (getMaxAmplitude()) to get the maximum voltage value induced by the external microphone and is converted to dB using the following equation when recording external sound.

978-1-4799-1740-2/15/$31.00 ©2015 IEEE

Fig. 1. NoisyGlobe enviorenmental sound level measuring

Fig. 2. NoisyGlobe GPS location of the user

$$SPL\ dB = \log\left({Max\ Amplitude}/{Reference\ Value}\right) \quad (1)$$

We define a reference amplitude value when converting actual sound pressure value to dB and for the HTC One X, this reference value is taken as 0.6 by calibrating against an industrial sound level meter.

B. *Graphical representation*

When measuring the environmental sound level, the user is presented with a real time graphical representation of sound values, which will provide a better understanding of sound level variation and the amount of noise exposure. Also, the user can render a noise level graph using the historical values as shown in Fig. 1.

The user can also see his or her GPS location in Google maps while monitoring and recording environment sound level as shown in Fig. 2. This will help the user to identify noisy areas according to the geological information shown in the map and it will help to analyze and predict noise exposures in other similar areas on the map.

C. *Context awareness*

The user is not expected to have expertise on sound level monitoring and therefore he may use the application in different positions during the sound level monitoring process. This presents the challenge of ensuring the accuracy of sound levels and achieve this with a context awareness module in the NoisyGlobe application that will avoid recording when if the device is not in a suitable position.

A good evaluation of context awareness of the mobile sound level meter could be found in [4] and [5], which state that if the mobile sound level meter is held in our palm, then its accuracy is within 2.7dB of the reference sound level. If the phone is in a shirt pocket or carried around the waist, the accuracy is within 3.4 dB of the reference. The additional error of 0.7dB was small compared with the actual noise level of about 67dB. If the phone is carried in a backpack, then the accuracy is within 4.1dB of the reference sound level. The outcome of this evaluation suggests that carrying the mobile sound level meter in a shirt pocket, around the waist, or in a bag doesn't a lower the noise level as one would expect being in a closed environment. Therefore, the best position to record sound is when the device is in an open context like on the palm of the user.

Modern smart mobile devices come with a proximity sensor for sensing the distance to a barrier from the devices screen (display side). Android provides access to the proximity sensor of the device with its SensorManager and it's generally treated as a Boolean sensor meaning that the distance reading is used to indicate if the device is too close to a barrier based on a preset threshold value. Generally, the sensor value will trigger within 2-5 cm from a barrier [6]. NoisyGlobe determines based on this whether the device is in an open space that will reduce the probability of monitoring inaccurate noise values.

CDME android client is also aware about the device's external loudspeaker state and pauses sound monitoring when the loudspeaker is turned on and resumes automatically when it's turned off. Also, the client responds to active calls in the same way. This active response system of CDME improves the accuracy of collected data.

D. *Data storage and uploading*

The NoisyGlobe client is responsible for storing and uploading monitored sound level values to CDME server where the data are analyzed and mapped with specific user and location. It uses a MySQL Lite database for storing sound level values, geological information like longitude and latitude, current time stamp, and the user's IMEI number.

The NoisyGlobe application runs a self-maintenance routine periodically to remove old entries that have been already uploaded and yet in the local device. The design of CDME allows capturing noise data even when no connection to the back end is available. The recorded values are stored locally and are uploaded when a data connection is available. Therefore, it's necessary to perform this cleanup to reduce memory usage.

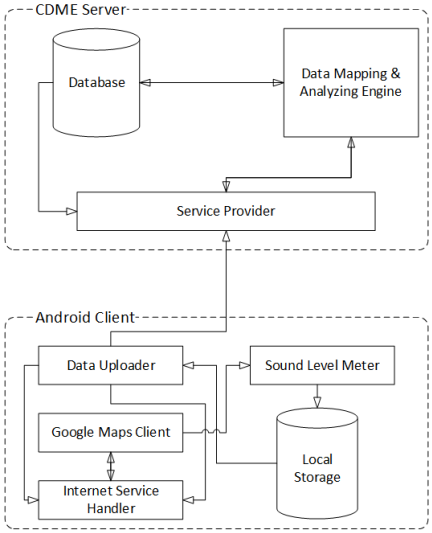

Fig. 3. Overall system architecture

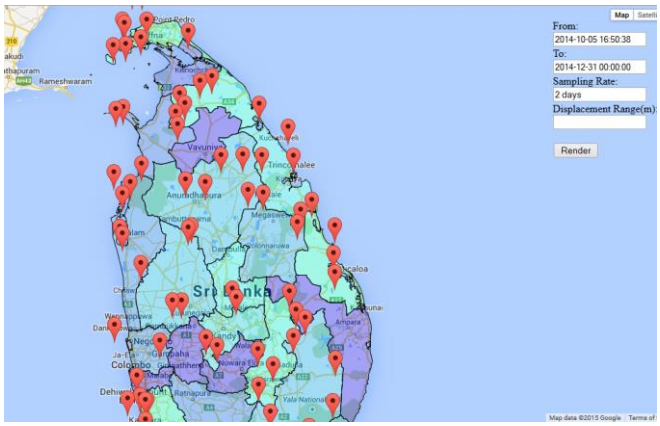

Fig. 4. Heat map of sound levels from 2014-10-05 16:50:38 to 2014-12-31 00:00:00

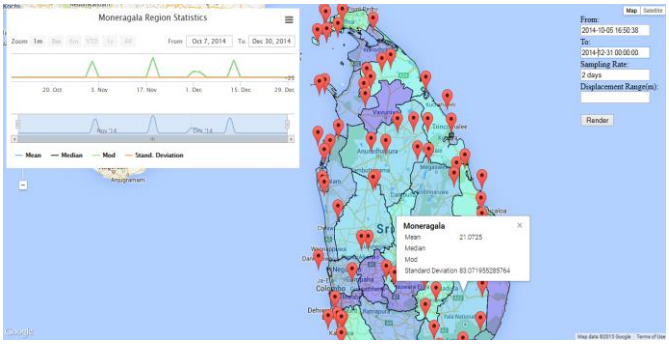

Fig. 5. Moneragala district statistics within the given time range and for the given sampling rate of data

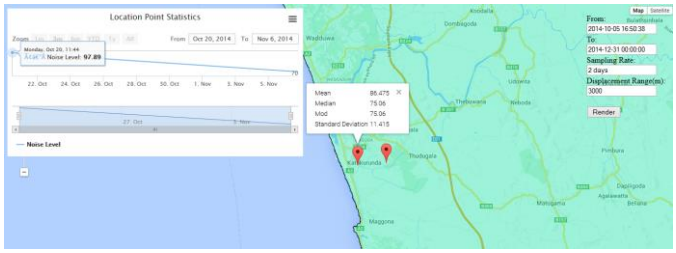

Fig. 6. Specific location statistics clustered with nearby other location due to given displacement range 3000m.

The client also captures the application usage time of the user, and the overall distance he has travelled for measuring and monitoring environment sound level values. These are used for the purpose of user profiling and rankings which will appreciate and promote user's efforts. Currently, this user profiling module is not included in the system and it's mentioned under future work, but the system uploads necessary data to the server to be used in future.

III. CDME – Server

The system is designed around the common client-server architecture, where the client could be any data collecting system conforming to the CDME server API. The client in this project is the Android application – NoisyGlobe – that collects environmental sound and uploads them to CDME server for analyzing.

NosiyGlobe client consists of 4 modules – sound level meter, data uploader, Internet service handler, and Google maps client – and a local storage system. Sound level meter collects environmental sound level, subjected to the context aware criterion described earlier, and stores them in local storage. The user can decide the mode of operation – foreground application vs. background service – for this module if necessary. Data uploader is a background thread that uploads data stored in the local storage to CDME server. The choice to use Wi-Fi, mobile data, or both is configurable to avoid potential data charges that could incur while on mobile networks. The actual transfer of data is handled by the Internet service handler module. The Google maps client tags geo-location coordinates with sound data and visualizes them in real time.

CDME server stores, analyzes, and visualizes the data it receives from a compatible client like NoisyGlobe. The service provider is responsible for invoking the appropriate functionality in data mapping and analyzing engine for user requests. The user can control visualization and graph generation through parameters provided in the Web interface.

The main feature related with visualization in CDME server is the generation of heat-maps for collected data. This happens in real time based on the parameter values of "From Date" and "To Date", which represent the window of time to consider. The heat-map represents a district with its average sound level for the given period of time using a color range varying from dark blue

to dark red. If no time range is given for rendering the heat-map the engine considers all the sound level values since the commencement of the system. An example of a heat map generated by CDME system is shown in Fig. 4.

The user can view overall statistics of a district by clicking on it in the map, which will generate a pop up balloon and a line graph of noise data. The generation is done in real time based on the value given in the "Sampling Rate" input box.

Fig. 5 shows an example of the overall statistics of Moneragala district and the generated graph using a 2 day sampling rate for the given date and time range.

Moreover, the system can generate statistics based on the distance to a given location. The user can specify this through the "Displacement Range" parameter and the system will collect noise values within a radius of this to produce the graph as in Fig. 6. Results of this could be further filtered by specifying a time range from the available options.

IV. Future Work

Despite the feasibility and other benefits, there are shortcomings to using crowd-sourcing for data collection. The primary concern is ensuring the accuracy of data. Despite having a context aware module, it's still possible to record noise from nearby human conversations, especially when the device is in a crowded place. Noise interference from user movements such as walking or cycling would require filtering too since the mobile device's microphone may get interfered by the wind blowing into it [4]. Performance is another concern since the GPS availability and GPRS connectivity tend to vary with weather (e.g., wind and rain). Thirdly, power consumption is important, especially with mobile devices. Data upload from the phone via GPRS can draw a large amount of power, particularly when the phone is far from a cell base station. A challenge is therefore to reduce the use of these radios without significantly impacting the application experience. One solution might be to provide an option to vary the periodic update frequency. For example, the application may select to update every second or once a day depending on the power saving option the user has chosen [4].

A few improvements that would add more value this solution are given below.

A. *User feedbacks*

Integrate a user feedback module for the NoisyGlobe client, which will help the user to define the noise source and the level of annoyance they face during monitoring. The server counterpart of this module will recognize and process these feedbacks and compute an accuracy value for the collected data.

B. *Heat maps for other countries*

The system uses a dynamically generating KMZ layer on Google maps to show the heat map for Sri Lanka. Many other countries include KMZ files specific to their regions, so it will add a global presence to CDME if these get integrated.

C. *Power consumption of the system*

NoisyGlobe application uses the mobile device's GPS module to obtain geographical coordinates and Wi-Fi or the mobile data connection to upload recorded data. These units tend to draw relatively high amounts of power thus requiring an optimal usage scheme based on a proper power consumption study of the NoisyGlobe application.

D. *A-Weighting sound level values, and voice cancellation*

NoisyGlobe doesn't implement frequency dependent hearing of the human ear. A signal of the same pressure is interpreted differently for changing frequency levels in human ear [3]. A so called A-weighting has to be applied to the signal to normalize the value. Also, the sound levels may get interfered by user conversations, which would need to be filtered out from the original sound. Thus, it would benefit to have a separate module to cancel nearby voices from the recorded environmental sound.

E. *Callibration offsets of the mobile devices*

NoisyGlobe uses the microphone in regular smart phones or mobile devices to capture environmental sound. The downside to this approach is that these microphones aren't designed to be used as sound level recording devices and aren't calibrated against any industrial grade sound meter. While the latter is possible to do, it's not practical for a voluntary user to dedicate time and resource on adjust his or her phone. A less ideal, but feasible alternative is to have calibration data for different mobile models be stored in CDME and use them automatically to scale incoming data. Each mobile device has an IMEI number, which can be used as an identifier to figure out correct reference data.

F. *User profiling module*

In NoiseMap [3], there is a ranking mechanism of users, which will encourage the user to engage frequently with application and enjoy his or her contribution. NoisyGlobe would also benefit from having such a ranking system for users.

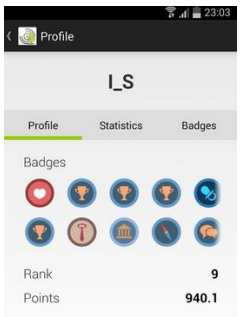

Fig. 7. Different badges for user's acheivements

Fig. 8. Overall rankings from existing users of the system

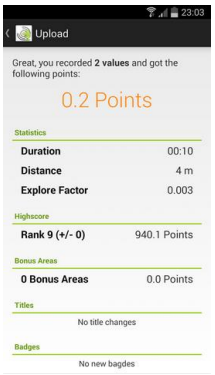

Fig. 9. Provide points for user contribution

V. RELATED WORK

There are a few systems intended for noise level monitoring in the literature and we present brief descriptions of them comparing to CDME.

A. NoiseMap

NoiseMap [3] too uses an Android client to monitor the environmental sound levels. It gathers data on loudness and transfers them to the open urban sensing platform da_sense. Da_sense allows users to access and control their data, generate real-time noise maps and data graphs. Public data is made available using either a web service or a JavaScript API [3].

NoiseMap samples the incoming sound to translate the discrete digital signal to a dB full scale (dbFS) value. This value has to be translated to dBSPL were SPL stands for sound pressure level. To translate dBFS to dBSPL a calibrated value has to be added as given below.

$$dBSPL = dBFS + xcal \quad (2)$$

The stored data can be used to generate noise maps on Google maps both in the Android application and the Web portal. The Web interface presents graphs and noise related data in a user friendly manner. Also, the generated data points are clustered depending on the zoom level. The clustering algorithm used is DBSCAN [7] as it has provided a good and fast geographic clustering. Every data point is selectable and the tool tip gives additional information of the sensor node and average noise level. The graph for each sensor node is also accessible from the tool tip. It allows for a quick overview of all the data collected, average loudness, and a graph showing the data over time. It's also possible to select a single location to show a 24 hour noise graph of the area [3].

However, this system doesn't provide user with dynamically generated heat maps, and statistical analyses parameterized by user input as in CDME. Also, the visualizations are limited to location points, and sound level variation is limited to a 24 hour graph only. Moreover, the NoiseMap client doesn't come with a context awareness module for refining the data collected from the client.

B. NoiseTube

This system has initially targeted devices which ran Symbian/S60 operating system [8], but they have launched recently an Android client. The mobile application contains a real-time signal processing algorithm which measures the loudness level of the microphone recording the environmental sound (at 22500 Hz, 16 bits) over 1 second at a chosen interval. An A-weighting filter is then applied to the recorded sound and the equivalent sound level $(Leq)^2$, measured in dB(A), is computed [8].

The server component is implemented using Ruby on Rails, MySQL, Google Maps and Google Earth.

Users can specify the source of a noise (e.g., cars, aircraft, and neighbors) and provide an annoyance rating or any additional contextual information in the form of free words (tags). And it allows users to describe their location using pre-configured ("favorite") place tags (such as "home", "work", the name of the subway station) in addition to GPS positioning [8].

C. Ear-Phone

The mobile application is developed and tested on Nokia N95 mobile devices and HP iPAQ mobile devices. Noise levels are assessed on the mobile phones before being transmitted to the central server [9-11].

This system uses a signal processing module which applies A-Weighting to calculated sound levels from the environment and the client uses a calibration mechanism for the device which is done manually as mentioned in [10]. And they have used Java threading for separate modules as implemented in CDME noise client.

The server component consists of a MySQL database and PHP server-side scripting. They have used the MySQL database to store both the collected noise level data and the reconstructed noise level data[9, 10].

The central server has a signal reconstruction module as stated in [9] because the urban sensing framework can't guarantee that noise measurements are available at all times and locations. Thus, it assumes that the client won't provide a continuous sound level signal for each and every location, which has unintentionally increased the complexity of the server. In contrast, CDME noise client gathers noise levels only for stable locations determined by the Android GPS provider services.

D. NoiseSpy

NoiseSpy application is written in Native Symbian C++. They have chosen this language as it enables the phone software to access all the development APIs which aren't available by JME, and to avoid potential bugs such as memory leak in the JME audio API [4, 12].

This system uses last valid GPS coordinates, which is a weak point in the system where the coordinates would be out of date in the absence of the GPS services causing incorrect data to be sent to the server. Also, it creates a log file for each measurement of sound in the phone memory, which would unnecessarily increase memory usage. CDME client avoids these pitfall by using GPS updated coordinates when only the services are available, and storing data in a SQLite database in the device memory.

Unlike in CDME's noise client, it uses a user profiling module which, will allow the user to visualize their personal data

on web interface provided by the system. Also, the system is focused on journey based data collection where as in CDME's noise client data is collected and uploaded in stable positions.

The method of plotting the data is to divide the area into a grid of squares and to place data points into the correct square based on GPS position. Furthermore, users can view the same journey in a variety of formats, including an overlay on Google Maps, Ordnance Survey OpenSpace, and Google Earth KML, which automatically opens GoogleEarth to view a 3D model of the measurements.

In [1] they have used KML layers to represent the analyzed data using their system in Google Earth. This will help the user to easily observe the noise exposure statistics data over time and around the globe, which is the main motivation behind CDME as well to include KML or KMZ layers for representing analyzed data.

VI. Summary

Participatory environment facts monitoring has become a popular method where mobile devices equipped with an assortment of sensors have proven to be very successful as client devices. The proposed system, CDME, is an extensible data analytics and mapping engine, which could be used with a number of data types such as noise, temperature, wind speed, and precipitation. In contrast to the existing systems, CDME presents dynamically generated heat maps of the collected data along with graphs that are customizable based on user parameters. CDME also comes with a client application, NoisyGlobe, to collect noise data. It provides context aware noise recording as a background service by default and exhibits low resource utilization.

References

[1] C. A. Kardous and P. B. Shaw, "Evaluation of smartphone sound measurement applications," The Journal of the Acoustical Society of America, vol. 135, pp. 186-192, 2014.

[2] E. Wahlgren, "Pocket protector," The Journal of Life Sciences, 2008.

[3] I. Schweizer, F. Probst, R. Bärtl, T. Darmstadt, M. Mühlhäuser, and A. Schulz, "Noisemap - real-time participatory noise maps," 2011.

[4] E. Kanjo, "NoiseSPY: a real-time mobile phone platform for urban noise monitoring and mapping," Mobile Networks and Applications., vol. 15, pp. 562-574, 2010.

[5] E. Commission, "Assessment and management of environmental noise," 2002.

[6] J. Corbett, G. Rambaldi, P. Kyem, D. Weiner, , R. Olson, J. Muchemi, M. Mccall and R. Chambers "Mapping for change - the emergence of a new practice," Participatory Learning and Action, vol. 54, pp. 13-19, 2006.

[7] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications," Data Mining and Knowledge Discovery 2, vol. 2, pp. 169-194, 1998.

[8] N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, and L. Steels, "Citizen noise pollution monitoring," presented at the Proceedings of the 10th Annual International Conference on Digital Government Research: Social Networks: Making Connections between Citizens, Data and Government, Puebla, Mexico, 2009.

[9] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-phone: an end-to-end participatory urban noise mapping system," presented at the Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, Stockholm, Sweden, 2010.

[10] R. Rana, C. T. Chou, N. Bulusu, S. Kanhere, and W. Hu, "Ear-Phone: a context-aware noise mapping using smart phones," Autonomous Systems Laboratory, University of New South Wales, Portland State University, 2013.

[11] DEFRA. Noise mapping england. Available: http://www.noisemapping.org/

[12] Audacity. Free, cross-platform sound editor and recorder. Available: http://audacity.sourceforge.net