Middleware for crowd sensing systems

Student Name: Manan Wason
Roll Number: 2013056

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering

on 18 Nov, 2016

BTP Track: Research

BTP Advisor
Dr. Pushpendra Singh

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **"MEW: Middleware for Crowd-Sensing Systems "** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Pushpendra Singh**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

..... **Place & Date: .....**
Manan Wason

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

..... **Place & Date: .....**
Dr. Pushpendra Singh

Abstract

Nowadays, mobile devices are loaded with many useful sensors and have high computational capabilities. The sensors available on these devices provide valuable data that can be harnessed in many ways. Due to abundance of mobile devices, it has become a challenge to identify optimal data providers.There have been many approaches already proposed to identify a group of mobile nodes that can serve a particular request like collecting accelerometer data or wifi strength, something that makes use of the sensor data from this group of mobile nodes. But, very little work has been done to identify an optimal node within that group. An optimal node can be defined as one that has the best score based on its history to serve requests and vital stats like battery level, data connection quality etc. In my project I address this problem and try to develop a middleware to achieve the same. I also harness the middleware's capabilities to develop real world applications using the data collected by optimal nodes. This semester I was primarily working on developing applications that use the data collected using crowd sensing through the nodes assigned by the middleware.

Keywords: Android, Crowd Sensing, RabbitMQ

Acknowledgments

I would like to express my gratitude towards my advisor, Dr. Pushpendra Singh for his constant support and guidance. Any progress in this project would not have been possible without his supervision.

Work Distribution

I was working on this BTP alone under the guidance of Dr.Pushpendra Singh and Ms.Garvita Bajaj.

Contents

Chapter 1

Introduction

There are two end-points of the spectrum of conscious human involvement for information collection(In our case mobile sensor data), namely participatory sensing, and opportunistic sensing. A participatory approach incorporates people into significant decision stages of the sensing system, such as deciding what data is shared and to what extent privacy mechanisms should be allowed to impact data fidelity. Consequently, a participatory system design focuses on tools that assist people to share, publish, search, interpret and verify information collected using a custodians device. Participatory sensing places demands on involved device custodians (e.g., manually support data collection) that restrict the pool of willing participants. Opportunistic Sensing refers to a paradigm for signal and information processing in which a network of sensing systems can automatically discover and select sensor platforms based on an operational scenario. With opportunistic sensing the participant mobile node- the custodian, may not be aware of active applications. Instead a custodians device (e.g., cell phone) is utilized whenever its state (e.g., geographic location in our case) matches the requirements of an application. This state is automatically detected; the custodian does not knowingly change the device state for the purpose of meeting the application request.

Thus, a strong motivation for opportunistic sensing is to increase the scale and scope/diversity of applications that may otherwise not be supported. Opportunistic sensing shifts the burden of supporting an application from the custodian to the sensing system, automatically determining when devices can be used to meet application requests.

Since the data provided by multiple participants is often redundant, it is sufficient to collect data from a subset of devices and our system lays emphasis on effective utilization of mobile sensors by devising a smart way of querying these subset of devices based on resources like battery levels, network bandwidth etc. while using opportunistic sensing paradigm.

1.1 Objectives

- Create a smart opportunistic sensing system, where the sensors present on mobile phones serve as the source of required data.

- Maintain a map of locations for all the participatory nodes of the system.
- Create a score for each participatory node in the system. This score would be based on factors like mobile phones battery life, available sensors, etc. It would be used to decide which node can serve a particular query optimally.

Chapter 2

Literature Review

Several frameworks have been proposed in the literature to allow researchers and application developers to harness the power of crowd sensing. Following are some of the systems proposed to achieve this and key observations regarding them. Each one focuses on optimizing a particular aspect of the whole system.

2.1 AnonySense [1]

- In comparison to embedded static sensors, system that uses sensors of mobile phones, which are rechargeable, is less energy constrained and therefore targets a network lifetime of years rather than weeks or months.
- Push computational complexity and energy burden to more capable nodes if possible.
- Keep a list of geographical locations and the nodes in them.
- Opportunistic tasking and collection requires the node which has been assigned a query, to remain in the sphere of interaction long enough to serve the query back.
- Lazy Uploading: Still if the node moves out of network coverage before serving the query, we keep track of that query, and later whenever the node gets network access back, it can serve.It also tries to assign the same task, if being repeated, to the same nodes.

2.2 CarTel

- [2] Sensors tend to produce more data than the network can promptly deliver to the portal (e.g., due to insufficient bandwidth or lack of connectivity), applications on the portal need a way to specify how to prioritize data(e.g., preferring summaries to be delivered before detailed values).
- Provide a declarative way for applications to express what data is important.

2.3 Real Time Pothole Detection using Android Smartphones with Accelerometers

- [3] Axes are first oriented to the ground.
- Comparision of 4 different algorithms, described below
- Z-THRESH: Upper and lower bounds are set on the Z values to obtain irregular values that symbolise potholes. Gives about 78% accuracy
- Z-DIFF: Unlike thresholding here they looked for difference between values more than a specific threshold to consider a pothole. This served about 92% accuracy.
- STDEV(Z) : Here the authors calculated standard deviation within window sizes and then set thresholds to identify potholes. This served about 81% accuracy
- G-ZERO : When all axes values neared 0g, a pothole was detected because the vehicle was in a free fall. This gives about 83% accuracy

2.4 The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring

- [4] They divided the problem into 5 components.
- Speed: Windows in which the car is not moving, or is moving very slowly, are ignored. This stage rejects events such as door slams, as well most curb ramps.
- High-pass: The high-pass filter removes low-frequency components from the acceleration signal in all the x-axes and z-axes. Such components can be introduced by acceleration, turning, veering, braking, and as well as subtle changes insensor orientation.
- z-peak: Peak acceleration in the z-axis is a prime characteristic of significant road anomalies. This filter rejects all windows with a peak (absolute) z-acceleration less than a threshold tz.
- xz-ratio: Acceleration in the x-axis can help identify road anomalies that span the width of the road (such as railway crossings, speed bumps and expansion joints), and therefore impact both sides of the car equally. Assuming potholes only impact one side of the car, a true pothole event with a large z-peak acceleration should produce a significant peak in the x-axis within some small time window w.
- speed vs. z ratio: At high speeds, even small road anomalies can create high peak acceleration reading. This filter rejects windows where the peak z acceleration is less than a factor ts times the speed of travel.

2.5 Nericell: rich monitoring of road and traffic conditions using mobile smartphones

- [5] They categorise values on the basis of the speed of the vehicle.
- At speeds less than 25kmph they look for sustained drops in z values of the accelerometer.
- At speeds more than 25kmph, they just use Z-thresholding to identify potholes.

2.6 Formulation of a Simple Model to Estimate Road Surface Roughness Condition from Android Smartphone Sensors

- [6] In this study, a simple model has been formulated to estimate road surface roughness condition from Android smartphone sensor data.
- The goal was to explore the use of smartphones, in the field of road maintenance management and continuous monitoring.
- The formulation of the model is based on an experiment and frequency domain analysis, in which it has been found that the sensor data, such as 3 axis acceleration and speed, has a linear relationship with road surface roughness condition.

Chapter 3

Components

3.1 Mobile Nodes

3.1.1 Registration

Any participatory node first registers itself to the system. There are many identifiers available for an android phone. Some are listed below, along with the reasons of not using them.

- TelephonyManager.getDeviceId() :
  - Wi-Fi only devices might just not even have this available.
  - This ID persists across device wipes - if I have this, I wipe my device and give it to a friend, they will be uniquely identified as the same install as me.
- MAC address:
  - Again, some devices might not have Wi-Fi, or might not give you the address with Wi-Fi off.
  - All these hardware identifiers have problems of reliability and availability.
- Android ID:
  - It does not work before Android 2.2.

We are using the Android ID to correctly identify an app instance in a mobile node.

3.1.2 Upload various Parameters

Once the device has registered, it uploads data like battery stats, available sensors, geographical location, wifi state, camera availability, current cellular network information, etc. These can be done by either push mechanism when the devices pushes any change in parameters or by

pull mechanism when the server queries for these parameters in some fixed amount of time. We are following the pull approach in our model. These statistics are then used by the system to calculate a score for each mobile node that is used to later on determine which query can be better served by which particular mobile node.

3.1.3 Query Format

Each query, apart from mentioning the type of sensor data it requires, also specifies these parameters:

- Start Time: From what time should the node start collecting the data.
- End Time: Till when the data should be collected.

3.1.4 Upload new Parameters

The server would take care of polling for new state data of the mobile node. Also, each time a query would be served by a node, it would also send its parameters alongside the data.

3.1.5 Optimize data upload

If there are multiple queries to be served by a single node at any point of time, then try to reduce the number of times a POST request needs to be made to the server, depending on the Max Return Time of the queries. For instance, if I have a query that needs to be served within 2 hours from the time it was received, I would collect the data, but not send it back immediately. Rather wait for some X amount of time, during which if some other query arrives, I would collect the new queries data, and then serve both the queries back together. This would reduce the number of network calls required.

3.2 Server

The server has been made in Java with rabbitmq Hotqueues. Right now the server only registers the queries and allocates devices on the basis of Nice value of the node which we have calculated on the basis of different parameters of the device. The file transfer is actually taken care by the rabbitmq queues. No data is being logged right now on the server, hence we can't serve past queries currently.
There are two types of clients that the server handles:

- The one which requests the server to provide sensor data for a given period and location. The requests also include several other parameters which the server uses to identify an optimal request handler.

- The other client is the one which serves the query. Whenever the server receives a sensor data request, it can use any one of these clients which are currently available.

The server performs the following tasks:

3.2.1 Query processing

For every query received, we assume that some other mechanism will provide a group of clients that satisfy the basic requirements of the query like location and sensor availability. After we have the set of possible clients, each one of them is requested to send some client-specific information as explained above. This information is then used to calculate the Nice value(NV) for each of these clients. The client with maximum NV times the weight is requested to provide the required sensor data. This request has several parameters, some important ones are stated below:

- Type of sensor
- Duration
- Start time (could be empty)
- Maximum return time - This is the deadline for providing the required data. After this, the request will be cancelled.

If this client fails to serve the request, the client which requested the data is notified about that and the request terminates.

Possible optimizations

- The server can also piggyback to other applications that are using the data already. For example if an application is accessing GPS data and the phone doesn't have a high NV, it should be allocated the query anyways. So if there is a possibility of piggybacking on existing usage, application should prefer that node.
- A different approach to handle multiple requests scenario is to group the requests such that an optimal client gets a couple requests depending upon the type of sensor data asked. This is because many sensors are active all the time in a smartphone, therefore if a client is handling some particular request, it can also be asked to serve other requests too and send the data in a bunch. This way, we impose a little more load on the already existing data provider and save the overhead of including another client.

3.2.2 Providing data

When a client sends the required sensor data, it nows needs to be sent to the requesting client. The format chosen to send the data is JSON.

Possible optimizations

- The data received for each query can be saved at the server so that subsequent requests for the same data can be served directly without involving any clients.

Chapter 4

Applications

Here we discuss about some applications that can be made using MEW:

4.0.1 Pothole Detection

We can collect context based accelerometer data from various participatory nodes registered with us. The application developers can apply various thresholding algorithms on this data to get a wide coverage of their pothole detection problem. I have also worked on this problem this semester. I started reading up on some of the existing solutions for this problem. Some of these solutions are listed in [3], [4], [5], [6]. I particularly picked these 4 because of their cited algorithms. Out of these, [5] is considered the base standard for pothole detection algorithms, so I decided to implement this.

4.0.2 Wifi and Network strength analyzer

We can also use croud sourcing to test wifi and network strength across the campus and/or in the whole city. The phones inherently collect wifi and network strength all the time and we can attach these readings to LatLng's and get a perfect map of network strength without employing special resources. This map can then be used to identify dead zones as well.

4.0.3 Noise analyzer

We can use the phone's microphones to get surrounding noise of an area that is geofenced. Then we can use this microphone data to estimate noise pollution in that area. The basic requirement would be geofencing an area and node allocation only in that geofence. This can be used as a real time noise pollution application as well.

4.0.4 Estimating Population in an area

With the onset of wearable devices and wireless headphones, lots of people keep the bluetooth on their phones activated almost always. We can make use of this and calculate the number of people in a particular area. We just need to scan all the bluetooth devices in the near vicinity bounded by a geofence and remove redundant entries. This obviuosly doesn't give the exact number of people but it gives us an estimate which can be extrapolated to get a real figure. This can be used in various places like kumbh melas, disaster prone areas etc. to keep a track of population and manage efficiently.

Chapter 5

Testing

I tested this system by working on developing a pothole detection app with nericell [5] as the baseline for detection.

5.0.1 Experimental setup:

- I initially registered 9 devices to the server. Each of these devices were kept in the same car and in the same orientation.
- A query was sent by a client device to a subset of these 9 devices with start and end LatLng, time and which sensor data has to be collected(Accelerometer in this case). Then we drove on a pre-decided track near our Institute to collect the readings.
- This was repeated with different number of nodes queried with different levels of battery values to identify the workign of Nice Values based allocations.

5.0.2 Results of the experiment

We were able to collect data from these nodes. Now we had to do processing of the data collected using the middleware by applying the Nericell [5] algorithm on it.

5.0.3 Results of the Processing

- We weren't able to get correct number of potholes due to variation in the sampling of accelerometer data in different phones
- We went through some pretty big potholes which were counted as many potholes, which makes the results go bad.
- Due to different sampling rates, I wasn't able to determine correct thresholds as well that would return these values.

5.0.4 Wifi and network strength

I have also developed applications that measure strength of wifi and network at a particular location and map it to location to obtain a correct map of signal strengths but haven't been able to test it yet.

Chapter 6

Conclusion

Currently, there are many systems available that are based on opportunistic sensing paradigm. Some are focused on providing anonymity, others focus on geolocation based dispatching of queries. But they do not focus on how to make the queries smart, in other words, how to determine which exact node(s) is/are best suitable to serve our data request based on the nodes current state. Our system leverages the available state information about various nodes to make an informed decision about whom should the system ask for data, so that the overall system becomes more intelligent.

6.1 Future Work

Since we have a working middleware and query allocation model, we would like to make more optimisations to the system like serving past queries, options of reusing data collected previously. We are also working on making use of MEW to develop some kinds of applications which serve as a proof of concept of the benefits that a developer gains by usign MEW for building applications.

Bibliography

[1] Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, Nikos Triandopoulos, Anonysense: privacy-aware people-centric sensing, MobiSys '08 Proceedings of the 6th international conference on Mobile systems, applications, and services, Pages 211-224, 2008.

[2] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan and Samuel Madden, CarTel: A Distributed Mobile Sensor Computing System, SenSys '06 Proceedings of the 4th international conference on Embedded networked sensor systems, Pages 125-138, 2006.

[3] Artis Mednis, Girts Strazdins, Reinholds Zviedris: Real time pothole detection using Android smartphones with accelerometers, Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference.

[4] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, Hari Balakrishnan, The pothole patrol: using a mobile sensor network for road surface monitoring : MobiSys '08 Proceedings of the 6th international conference on Mobile systems, applications, and services, Pages 29-39, 2008.

[5] Prashanth Mohan, Venkata N. Padmanabhan, Ramachandran Ramjee,Nericell: rich monitoring of road and traffic conditions using mobile smartphones, SenSys '08 Proceedings of the 6th ACM conference on Embedded network sensor systems, Pages 323-336

[6] Viengnam Douangphachanh, Hiroyuki Oneyama:Formulation of a simple model to estimate road surface roughness condition from Android smartphone sensors, Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference, 2014.