Code In The Air: Simplifying Sensing on Smartphones

Code In The Air: Simplifying Sensing on Smartphones

Tim Kaler, John Patrick Lynch, Timothy Peng,
Lenin Ravindranath, Arvind Thiagarajan, Hari Balakrishnan, and Sam Madden
MIT Computer Science and Artificial Intelligence Laboratory

1 Introduction

Modern smartphones are equipped with a wide variety of sensors including GPS, WiFi and cellular radios capable of positioning, accelerometers, magnetic compasses and gyroscopes, light and proximity sensors, and cameras. These sensors have made smartphones an attractive platform for collaborative sensing (aka crowdsourcing) applications where phones cooperatively collect sensor data to perform various tasks. Researchers and mobile application developers have developed a wide variety of such applications. Examples of such systems include BikeTastic [4] and BikeNet [1] which allow bicyclists to collaboratively map and visualize biking trails, SoundSense [3] for collecting and analyzing microphone data, iCartel [2] which crowdsources driving tracks from users to monitor road traffic in real time, and Transitgenie [5], which cooperatively tracks buses and trains.

What do all these applications have in common? Today, anyone who wants to develop a mobile phone crowdsourcing application needs to:

1. Write and debug low-level application software for one or more phone platforms (iPhone OS, Android, Symbian, etc.).
2. Publish the application on an official distribution channel like the iPhone App Store or the Android Market, and incentivize enough volunteers with phones to use the application, a challenging task.
3. Deal with issues of privacy, energy and intermittent network connectivity. For example, a traffic monitoring app that always collects GPS location samples once a second would drain the battery, and users would not want to install it.
4. Filter out irrelevant portions of sensor traces from phones that do not apply to the problem at hand. For example, Transitgenie, which cooperatively tracks public transit, filters out location traces when the user is stationary, walking or indoors.

What if we had a platform with a large *pre-existing* installed base of phone users that enabled researchers and developers to instantly develop and deploy their own applications without having to worry about any of the above concerns? To realise this vision, we are building *Code in the Air*, a platform for developing mobile crowdsourcing applications that deals with all the low-level details. *Code in the Air* comprises four key components:

- A *background phone service* that users install and configure *once* on their phones. Users specify system-wide privacy settings (how much and what data they are willing to share) and energy and bandwidth budgets (the platform should not reduce my battery life by more than 30%, and should not upload more than 30 MB of data).
- A *web service* that enables researchers to develop, visualize and configure applications for the entire installed base of phones running the service. Apps are written entirely as simple *server-side scripts* without requiring a researcher to write or debug a single line of low-level phone code, or handle the low-level details of wireless communication, energy optimization and intermittent server connectivity.
- A *declarative language* for writing server-side scripts that supports a rich set of *filters* to specify conditions when code should be executed, and *actions* to be executed when the filters become true. For example, "Start collecting GPS data sampled every minute when the user starts walking outdoors".
- An *intelligent optimizer* that is energy- and bandwidth-aware, that optimizes scripts written in the high-level language and distributes them to individual phones. It uses the filter conditions to target each script to the most relevant set of phones. For example, it would not distribute a Boston traffic monitoring application to a phone user living in San Francisco. The optimizer also decides the optimal method to use to execute filters, and the optimal filter ordering (e.g. check if user is moving first before turning on the GPS). It uses a simple pre-configured model of sensor sampling energy costs and data rates for each phone device to determine an efficient execution plan.

2 System Description

Figure 1 shows a mock-up of the *Code in the Air* web interface. A developer can quickly create a new crowdsourcing task by typing in or uploading a server-side script. The script is written in a simple high-level language that supports *filters*, *actions* to execute when the filters are or become true, and *targeting conditions* that specify how much data should be collected and from how many phones or users. A script can consist of multiple filters and associated actions for each filter. Researchers can use the UI to add some kinds of filters visually: e.g. they can draw a geographic bounding box, like the one shown in red in the figure, and target the application to phones within that box. They can also select predefined filter conditions and actions from a drop-down menu (not shown in the mock-up). The web interface can also be used (privacy settings permitting) to view, control and configure

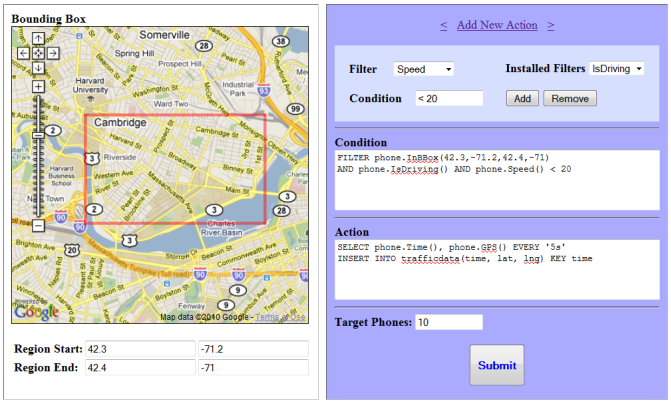

Figure 1. Code in the Air Web Interface.

the phones selected to run a "Code in the Air" task, view its energy consumption and terminate or reconfigure the task.

We are in the middle of implementing *Code in the Air* as a background service for the Android operating system.

3 Demonstration Examples

We will demonstrate two examples of crowdsourcing applications written in our declarative language: remote traffic monitoring and indoor WiFi access point tagging.

Traffic Congestion Monitoring.

```
FILTER phone.InBBox(42.3,-71.2,42.4,-71)
AND phone.IsDriving() and phone.Speed() < 20
ACTION SELECT phone.Time(), phone.GPS() EVERY '5s'
INSERT INTO trafficdata(time, lat, lng) KEY time
TARGET 10 phones
```

Here, *InBBox*, *IsDriving* and *Speed* are all pre-defined filters the system knows about. The script instructs the system to collect time-stamped GPS coordinates every 5 seconds (0.2 Hz) from at least 10 phones detected to be driving within the Boston area, but moving slower than some cutoff speed. The script will populate a server side table, *trafficdata* which can be dumped out to access the data.

The optimizer chooses to execute the three filters in the following order. Whenever fewer than 10 phones are running this script, a server process periodically looks up the last known GPS location of each phone and excludes far away phones unlikely to enter the bounding box (e.g. in San Francisco). It then installs the filters on candidate phones near the box. These phones first use the low-energy accelerometer to check if they are driving, and then sample GPS periodically, first at a low rate to see if the speed and position filters are satisfied. When all three filters are satisfied, the GPS is activated at 0.2 Hz and samples are sent to *trafficdata*.

Demo Details. For this demo, we will ask volunteers to drive a few cars in real-time around the city of Zurich with Android phones that have *Code in the Air* preinstalled. The demo viewers will be able to see the locations of phones near the relevant bounding box that are targeted by the above script, the most recent traffic data from these phones, and the real-time status of whether the accelerometer and GPS are being sampled and what rate. Whenever a driver stops driving or is too far away from the bounding box, the corresponding phone will stop sampling GPS frequently and start sampling the accelerometer. Demo viewers will be able to reprogram the phones live to change the GPS sampling rate or bounding box of interest, or remove or add filters and see how this changes which sensors are sampled.

Indoor Access Point Tagging.

```
FILTER phone.IsIndoors() AND phone.IsWalking()
ACTION SELECT phone.Id(), phone.Time(),
phone.WiFiAPs() AS aps EVERY '2 min'
INSERT INTO seenaps(id, time, aps) KEY aps
TARGET 10 phones

FILTER NEW aps IN seenaps(id, time, aps)
ACTION SELECT phone.User(),
phone.WiFiAps(), phone.Text('Tag')
INSERT INTO aptags(id, aps, locname string) KEY id
LIMIT count(EACH userid) <= 5
TARGET GetPhone(id)
```

This application collaboratively tags indoor locations, associating user-specified location names with WiFi access point signatures for localization. It has two `FILTER` conditions. The first runs on all targeted phones and samples WiFi access points at a low frequency (every 2 minutes) when the user is walking indoors. The second is a simple trigger that is executed whenever the server-side `seenaps` table has a new key i.e. when we see a previously unseen WiFi access point. When this happens, the phone executes `Text()`, which beeps the user and displays a simple predefined form (a text box with a button saying 'Tag') that requests the user to tag the current location. The `LIMIT` clause says that each key in the `aptags` table should have at least 5 rows, which prevents the system from prompting each user more than 5 times (and annoying them). The `TARGET` clause here instructs the action to target a specific phone with the appropriate id.

Demo Details. For this part of the demo, we will ask volunteers to tag locations while walking around the conference venue. We will display tags and associated access points collected from these users, and also show a live view of the sensors being sampled and the rate being used to sample them. Demo participants will see that the system prompts a user for a tag only when he/she enters a new zone where a new access point is seen. If the conference venue does not have access points, we will fall back to remote volunteers performing the task in the MIT CSAIL building.

4 References

[1] S. Eisenman, E. Miluzzo, N. Lane, R. Peterson, G. Ahn, and A. Campbell. The bikenet mobile sensing system for cyclist experience mapping. In *Sensys 2007*, 2007.

[2] "http://icartel.net".

[3] H. Lu, W. Pan, N. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Mobisys 2009*, 2009.

[4] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava. Biketastic: sensing and mapping for better biking. In *CHI 2010*, 2010.

[5] A. Thiagarajan, T. Gerlich, J. Biagioni, and J. Eriksson. Cooperative transit tracking using gps-enabled smartphones. In *Sensys 2010*, 2010.