

SocialCircuits: A Platform for Measuring Social Interaction

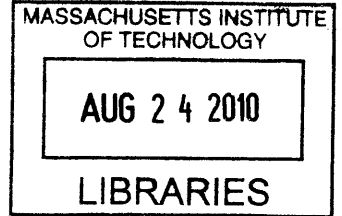
Using Smartphones

by

Iolanthe Chronis

S.B., E.E.C.S. M.I.T., 2008, S.B., Physics M.I.T., 2008

ARCHIVES



Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

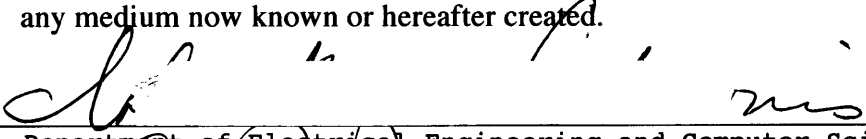
January 2010

[February 2010]


Copyright 2010 Iolanthe Chronis. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole and in part in any medium now known or hereafter created.


Author


Department of Electrical Engineering and Computer Science
May 17, 2010

Certified by


Alex (Sandy) Pentland, Toshiba Professor of Media Arts & Science
Thesis Supervisor

Accepted by


Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

SocialCircuits: A Platform for Measuring Social Interaction

Using Smartphones

by

Iolanthe Chronis

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

January 2010

ABSTRACT

SocialCircuits is a platform capable of measuring the face-to-face and phone-based communication network of a real-world community. This platform uses commodity mobile phones to measure social ties between individuals, and uses long- and short-term surveys to measure the shifts in individual habits, opinions, health, and friendships influenced by those ties.

The flagship experiment using this platform is a yearlong study of an MIT undergraduate dormitory. Some of the key challenges met in building and deploying the platform were mobile phone hardware and software selection, privacy considerations, community selection and recruitment, and minimizing data loss.

Thesis Supervisor: Alex (Sandy) Pentland

Title: Toshiba Professor of Media Arts and Sciences

1. INTRODUCTION

How do you measure the way that people interact with one another? How can you tell why a particular fad catches on in one community, but flops in another? In the past, there has been no way to measure the spread of ideas and opinions through groups of people except for notoriously flakey survey responses. Even new methods of social network data collection like Facebook mining fail to measure face-to-face communication, which is a highly important medium for transfer of trends and tastes.

In the past, specialized electronic sensors and badges have been used to measure face-to-face interaction. For instance, the Sociometric badge [6] was designed to identify human activity patterns, analyze conversational prosody features and wirelessly communicate with radio base-stations and mobile phones. Sensor data from these badges has been used in various organizational contexts to automatically predict employees' self-assessment of job satisfaction and quality of interactions.

Cell phones are a ubiquitous and natural part of our modern social lives. Your cell phone knows more about you than your spouse does. They provide a convenient tool for measuring social connectivity features related to phone calls and text messages. Sensor-enhanced smart phones can even determine personal location and nearness of friends. Several recent projects have used pervasive, mass-market mobile phones as active social sensors. Eagle and Pentland [1] coined the term Reality Mining, and used mobile phone Bluetooth transceivers, phone communication logs, and cellular tower identifiers to identify the social network structure, recognize social

patterns in daily user activity, infer relationships, identify socially significant locations, and model organizational rhythms.

We have built a scalable and reusable platform that transforms a smartphone into an advanced social sensor capable of capturing the relationships and influences within a dense community. This platform is needed because there are not many such data sets for the research community to use. Experiments of this type have a high engineering and deployment costs, and researchers often do not have the skills to build and deploy the platforms.

We deployed our platform on an undergraduate community during the academic year of 2008-2009, as part of a research project instigated by Professor Alex Pentland in support of the Ph.D. thesis research of Anmol Madan and the MEng thesis of Iolanthe Chronis. Experimental design was performed by Professor Pentland and Anmol Madan with help from Professor David Lazar and Dr. Devon Brewer. The design of the software platform was a collaboration between Iolanthe Chronis, Anmol Madan, and Alex Pentland. The creation of the platform was lead by Iolanthe Chronis, supported by UROP students Christopher Palmer, Griffin Chronis, and Xiao Fan. The actual deployment of the platform to participants, including subject recruitment, continual support, and project management was performed by Iolanthe Chronis with the support of UROPs Paul Kominers and Sylvester (Erons) Ohienmhen.

During the course of the experiment we collected over 3 million co-location samples, over 60,000 phone calls, and over 20,000 text message samples, forming the world's largest dataset capturing face-to-face diffusion and social influence behaviors [3, 4]. This document discusses

the platform design, construction, and capabilities, as well as the methodologies for community selection, subject recruitment, and experimental maintenance. It also provides an example of the kind of analysis that can be performed with the data collected.

2. THE SOCIALCIRCUITS PLATFORM

2.1 Platform Capabilities

The platform is a reusable, scalable system that transforms Windows Mobile 6.x smartphones into sociometers. The platform is designed to be deployed in a dense social network for an extended period of time. The platform is designed to collect as much data as possible about the face-to-face and phone based communication between subjects, without sacrificing privacy. The capabilities, described in more depth later, are:

Detect Bluetooth wireless devices in proximity: Bluetooth and other wireless-radio based co-location techniques have been used to identify the nodes and edges in social networks [1].

Bluetooth scanning can establish which subjects are in the same rooms.

Detect Wi-Fi (WLAN 802.11b) access point identifiers: Since most urban areas have a high density of Wi-Fi access points, these identifiers can be used to infer homogeneity and entropy of location and proximity patterns, e.g. is there a cluster of users who tend to visit similar locations frequently?

Capture Phone and SMS logs: The temporal and frequency features extracted from communication logs can be used to infer strength and type of social connection. The actual text or audio of the communication is not stored.

Scan Manager: This component initiates scans for Bluetooth, Wi-Fi, call logging and text logging processes at correct intervals. It also manages daily uploads of experimental data to the central server for tracking of data collection.

Flexible Integration with Web-Based Surveys: The platform integrates with commercial, off-the-shelf web tools for launching surveys for a large number of participants. A flexible, python-based back-end post-processing infrastructure efficiently parses through exported comma-separated value (CSV) survey responses, and generates MySQL tables per user, used for statistical analysis.

On-Device Survey Launcher: The platform supports launching single-screen daily surveys as soon as subjects turned their phones on in the morning. This allows a more fine-grained tracking of changing subject behaviors.

On-Device User Feedback Engine: The On-Device Survey Launcher also has the ability to pull images and data from the central server, allowing use of feedback about other subjects' responses or changing world conditions.

Over-the-Air Application Updater: The platform supports a native updater application that is designed to fetch compressed installer files (.cab format) from a remote server. This component enables seamless remote deployment of software updates, bug fixes, and new experiment modules.

Music Player: In order to study the diffusion of music, a custom music player allows participants to play, share, rate and search through the music library. Participant in our deployments have access to over 1500 independent music tracks of different genres, sourced under the Creative Commons license.

2.2. Operating System Selection

The first step in creating the platform was choosing the phone operating system. There are several smartphone platforms on the market, all with pros and cons. Figure 1 summarizes some of the major factors considered. A more complete description follows.

OS	Development time	Open?	Cross-carrier?	Cool factor	Do we have a platform built?
WinMo	medium	yes	yes	low	yes
Android	low	mostly	yes	high	In process
iPhone	high	no	no	high	yes
Nokia S60	medium	medium	no	low	no
Blackberry	medium	mostly	yes	medium	no

Figure 1: Summary of major pros and cons associated with building a platform on major smartphone operating systems.

2.2.1. Available sensors:

Different phones come with different available sensors which impact how the researchers can measure the activity of the subjects. All phones can track basic call and text message logs, but there are some other potentially essential sensors that a particular type of phone might have:

Absolute Location -- GPS, cell tower triangulation, and Wi-Fi-based localization (e.g. Skyhook) are location-sensing technologies with varying resolutions available on mobile phones. Wi-Fi triangulation and signal strength can also be used to calculate location on a room-level resolution. However, due to the battery constrained 5-minute scan interval, it is not possible to use popular location tracking techniques like particle filters or Kalman filters [2].

Co-Location (Proximity) – Most mobile phones are equipped with class 2 Bluetooth radio transceivers, which can detect Bluetooth devices within a maximum range of 30ft. However, most commodity mobile phones do not provide signal strength information. Also, a device will not be detected unless the Bluetooth radio has to be set to the ‘discoverable’ mode, which is often disabled by default as a security feature. It is necessary to programmatically turn this feature on.

Context and Activity – 3-Axis accelerometers can be used to detect many physical activities, including walking, running, or sleeping [5], although the classification accuracy is reduced if there is no fixed body position for the mobile phone. Accelerometer data can also be used to detect whether the subject is carrying the mobile phone—if the phone hasn’t moved for 13 continuous hours, then it is probably sitting on a desk and not being used by the subject.

Ambient Light/Noise Detection – Some modern smartphones have the ability to sense the level of ambient light, which is useful for inferring the context or location of the subject. For future studies it may be possible to increase the rate of data collection when ambient sensors suggest that the subject is doing something worth measuring more carefully.

2.2.2. Mobile Phone Usability

In order for subjects to use mobile devices over extended periods of time, the phone platform must be user-friendly. Physical size and weight of the device, available device RAM, and operating system interface are all key factors that impact long-term usability. For instance, users have different preferences for phone ergonomics--subjects tend to leave heavy and bulky devices at home, instead of carrying them all the time. If a device has low RAM, when the phone launches scanning process every 5 minutes, it may slow down applications and UI response time—an effect seen with both iPhone and Windows Mobile devices. Finally, the OS itself plays a key role in the usability of the device, and subsequently, the level of engagement participants have in the long-term experiment. iPhone and Android devices have easy-to-use interfaces with thousands of available applications.

2.2.3 Battery Life

Running background scan code every 5 minutes on a phone decreases its battery life by approximately 20%, for instance, but it is essential that this decrease does not cause the total battery life to dip below the 12-hour level, where a fully charged phone will power off before the end of the day. Particularly if the phone has trouble finding signal, like the iPhone does on the MIT campus, the battery could disappear within 3 hours. Windows Mobile and Blackberry have

the advantage that there are many supported models of phone, so it is easy to find one with a sufficient battery life.

2.2.4 Platform Openness

Today, not all phone platforms are equally open—the programmer often does not have access to the sensors and features that are required. For instance, an iPhone implementation of the platform can only run on jailbroken iPhones due to the inability to run background processes using the official Apple SDK.

2.2.5 Cross-Carrier Support

Verizon is an extremely popular carrier in the United States, particularly on the MIT campus, where Verizon users can always expect to have several more bars of service than AT&T or T-Mobile customers. Therefore for any MIT-related deployment where density of subjects is important, it is necessary to support Verizon phones. At the time of the development of SocialCircuits, Android and iPhone were both exclusively GSM-enabled. Android now supports CDMA operators, which makes it suitable for experiments of this nature.

2.2.6 Platform Decision

The Windows Mobile operating system was chosen as the SocialCircuits development platform, despite its failing in the usability and ease of development categories, because at the time it offered the only Verizon-compatible phone with Wifi support. Wifi is crucial because most potential subjects do not have data plans, so Wifi connectivity is essential for uploading data and

updating the software on the phone, and for making the phone a “smartphone”...that is, connected and useful as a mini computer.

In the future, however, studies of this type are probably best run on Android. As you can see from Figure 1, Android is an open platform that currently has popular phones that are compatible with every carrier, and an active community of developers.

2.2.7 Choice of Model

While theoretically in Windows Mobile it is possible to have one codebase that deploys on every phone model running the 6.x operating system, it is convenient to limit the number of models as sharply as possible due to slight but meaningful differences in hardware and design.

SocialCircuits used the SCH-i760 for Verizon phones, the Tytn II Kaiser (aka the Tilt) for GSM (with a few SCH-i780 models), the Alltel 6800, and the Sprint Diamond, all shown in Figure 3.

	SCH-i760	Tytn II Kaiser (Tilt)	Sprint Diamond	SCH-i780	Alltel 6800
Technology	CDMA	GSM	CDMA	GSM	GSM
SD Card	External	External	Internal	External	External
RAM	64MB	128MB	256MB	128MB	64MB
UI	Normal	Normal	TouchFlo	Small screen	Normal
Power Management Registry Keys	Simple	Complex	Simple	Simple	Complex
Number Deployed	~30	~40	4	3	3

Figure 2: Slight differences in model design and construction meant time consuming engineering challenges.

Even these few differences led to time-consuming engineering challenges, summarized in Figure 2. For instance, having one phone with internal rather than external storage meant that every time data was read or written to the storage area, the phone model had to be checked in order to determine the storage path. This was just one more complication and source of potential bugs.

Differences in RAM and UI mean that subjects experience different experiment-related bugs with respect to normal operation of the phone. For instance, on some models, the alarm clock will not ring if a Bluetooth scan ran at the same time, and on low-RAM models, scanning often interrupted on-device games. Debugging is always heavily model-dependent.

The registry key differences affected the power states of the device, which were important to set correctly so that the phone stays mostly powered off while unused. Different models responded differently to changes of registry settings, meaning that a registry-based solution was impractical.

Lastly, differences in model can lead to envy among the subjects—“why can’t Verizon users have a model of phone that’s as cool as the Sprint Diamond?”



TyTn II Kaiser



SCH-i760



Sprint Diamond



SCH-i780

Figure 3: The four phone models chosen to host the SocialCircuits platform.

2.3 Platform Implementation

Building the base platform took about three months, including the scan applications and music player. The Uploader, Updater, On-Device Survey Launcher and Feedback Engine were built as needed over the course of the experiment.

2.3.1 Developing on Windows Mobile

Windows Mobile coding takes place in the .NET world. The SocialCircuits Platform has pieces in both C# and Visual C++. C# uses managed code, meaning that it is easy to develop for but not very powerful. Managed code is an abstraction that ensures security guarantees for .NET developers, if you use only those API's. Our experiment needs some functionality that is not included in those APIs. Some of the capabilities that are not available in the managed code can be 'borrowed' using DLLImport, but for the most part, anything related to lower level device data had to be performed in VC++, which is time consuming and error-prone.

2.3.2 The Application Structure:

At its core, the data mining application is a loop that starts when the phone boots, and calls each sensing application one at a time. Each sensing application is separate executable so that if one fails, it does not bring down the entire scanning application, and so that updating can be done in a piecewise manner.

Battery life was a constant engineering challenge. Like most smart phones, the battery life on i760s and Tilts is not much more than a day when the phone is operating normally. Performing an intensive scan every 5 minutes only hurts that performance. Therefore, every scan had to be analyzed in terms of its usefulness versus its drain on the battery life to ensure that the total battery life stayed over about 12 hours.

2.3.3 Bluetooth

A Bluetooth scan is performed every five minutes. The Bluetooth scanning app is written in visual C++. First, the app checks to make sure that the user's Bluetooth radio is in "discoverable" mode, and turns it on if not. "Discoverable" mode ensures that other devices that perform a Bluetooth scan will be able to find the user's device. This meant that even if the user tried to turn their Bluetooth radio off, it would come back on automatically after no more than five minutes so that each subject is always detectable by all other subjects.

Then the app performs the scan itself. The Bluetooth scan takes about 20 seconds to perform. The scan reports all devices within a distance of around 30ft, but does not report signal strength of any kind. Therefore, Bluetooth acts as a rough indicator of co-location. Performing Bluetooth scans every 5 minutes cut the battery down by about 20%, which still allowed the battery to last for a full day.

2.3.4 Wifi

A Wifi scan is performed every five minutes. The Wifi scanning app, written in C#, reports the name and ESSID (like a mac address for Wifi access points) of each WAP that is in range, as well as a signal strength. If the physical location of each WAP is known precisely, one can theoretically then triangulate to get the user's exact location in the building with an accuracy of around 10ft.

However, keeping the Wifi radio on in order to do scans every 5 minutes lowers the battery life of most phone models to about 4 hours, and turning on and off the Wifi with that frequency is

not only a comparable power drain, it also would demand a potentially buggy set of tests to determine whether the user was using Wifi at the end of the Wifi scan (to determine whether it is appropriate to shut off the Wifi).

Unlike the iPhone and Droid, the default state of the Windows Mobile Wifi radio is off, and turning it on is a fairly intense UI challenge involving navigating at least 3 menus. Also, because of MIT's Wifi policy, the phones have to be registered by the student in order to connect to the network successfully. Also, connectivity in the dormitory is not consistent, with certain areas almost completely devoid of signal. Anything from a closed door to an operational microwave can render the Wifi network inaccessible.

Because of all of these concerns, the Wifi scan app did not turn the Wifi on by default. If Wifi is not on, no scan data is received, so the Wifi data is extremely intermittent. It can be used to get a sense of location outside of the dormitory—if 6 subjects are at a coffee shop together, and one of the subjects' Wifi radio is on, then we can use the Skyhook database of WAP locations to discover the location of all 6 subjects, since the Bluetooth scan will indicate that the subjects are co-located.

2.3.5 Call and Text Logs

The call and text message scanners run every 20 minutes, and have a negligible impact on battery life. The scan had to run every 20 minutes to ensure that if a user deleted a record from their phone records after a day, the scan would still capture it. The call log reports the time of call start and call end, the phone number called, and three flags indicating whether the phone was

roaming, whether the call connected, and how the call ended (dropped, hung up, etc). The text message log reports whether the message is incoming or outgoing, and the time of receipt of the message. Both loggers use a one-way hash function to encode the stored phone numbers. This is an extra safeguard to ensure that there is no possibility of phone numbers that a subject called becoming exposed.

2.3.6 Online Surveys

SurveyMonkey is used to administer online surveys. Starting and ending surveys capture demographic information and baseline data, and monthly surveys track evolving characteristics such as health habits, political opinions, fads, community standards, etc. SurveyMonkey includes useful tools for tracking which subjects have responded to a survey, and emailing the stragglers until everyone has been accounted for.

2.3.7 Music Player

The music player was designed to be a complete phone-based player that allows students to listen to and share music from our library of creative commons tracks. The player was written in C# using the form builder tools that Visual Studio provides. In a previous experiment, we had written a web-based application, but the non-uniformity of Wifi across the MIT campus makes streaming music off a web application frustrating enough that a native application was needed. The music player is intended to allow tracking of which tracks become popular, so that the researchers can analyze musical fads in the context of the social graph measured by the phones.

2.3.7.1 Design Considerations:

Selection of Music: Creative commons tracks were used partly because of the legal issues inherent in allowing users to share their own tracks, and partly to ensure that the popularity of the various tracks not to be influenced by outside media or previous preferences for specific artists. With a library of 1500 tracks that none of the participants had ever heard before, the success of particular tracks can only be based on two factors: quality of tracks, and communication between survey participants. To control for quality of track, we asked the subjects to rate the tracks before they were shared.

Sharing Technology: It was important to be able keep track of who shared which music with whom and when. In order to do this, music was shared not between actual phones, but between the phones and a central server. In order to perform sharing, a phone that had the song needed to send a share request to the server, which would be honored once the other phone checked in with the server.

Playlists: In order to make the functionality of the music player complete enough to replace commercial music players, we needed to support user-created playlists. In order for the application to remember the users' playlists after the application had exited, it was necessary to create a serializable song database. This meant that the application needed to be able to convert between the database object and a string at the opening and closing of the program.

Security: MIT students like to tinker around with technology. We did not want them figuring out how to alter our data or crash our servers, or pretend that they had used the player more than they actually had (since the subjects were reimbursed for their use of the player).

Sharing Meaningfulness: We did not want the users to perform dumps of their whole music library on complete strangers or to give away large quantities of music without listening to it first, so we only allowed users to share songs one at a time and after rating them. The downside of this decision was that sharing music was a fairly slow activity, which made the feature less frequently used.

2.3.7.2 Implementation

Users interact with the program through a graphical user interface. This interface was built using the C# default GUI toolkit, `Systems.Windows.Forms`. There are four main forms in the application, the “Inbox,” “Library,” “Playlists,” and “Now Playing” forms. The user tabs between these forms for the majority of their use of the application, only occasionally interacting with popup forms that appear for particularly complicated actions.

The “Inbox” form, shown in Figure 4, is used to check up on music that other users have shared with this user. From here new music can be downloaded. Music in the Inbox can be played without adding it to the library so that the user can sample it before making a decision about whether to keep it.

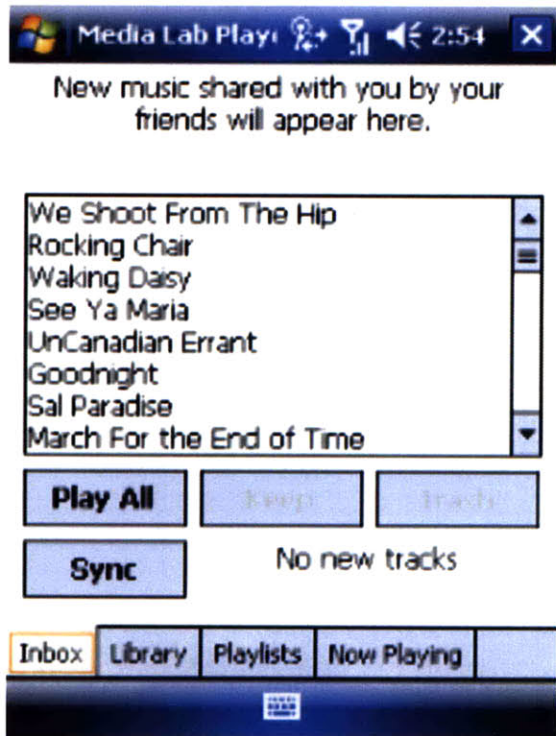


Figure 4: "Inbox" tab for the Music Player application.

The "Library" (Figure 5) and "Playlists" (Figure 6) forms are used to maintain the user's music.

All of the organization functionality in the program is accessed through these two forms.

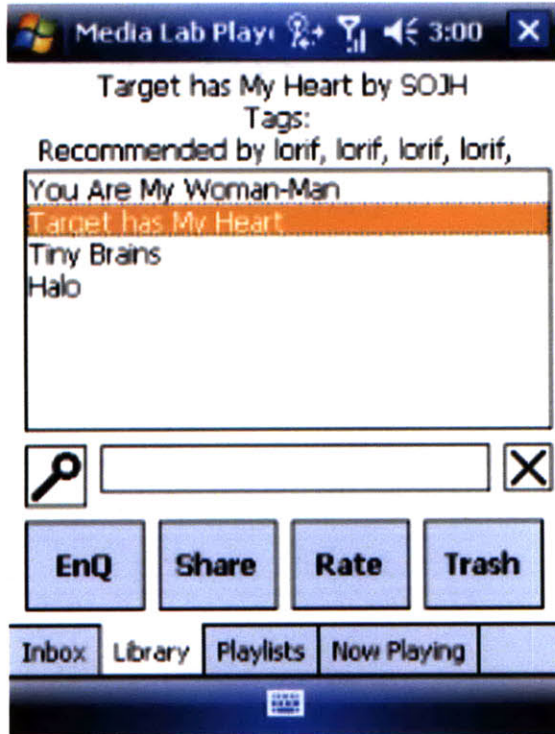


Figure 5: "Library" tab for the Music Player application.

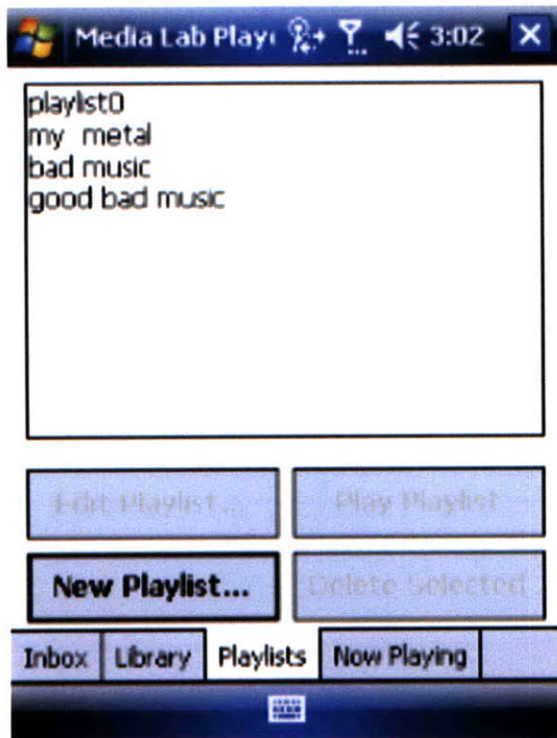


Figure 6: "Playlists" tab for the Music Player application.

From any of these three previous forms the user can add songs and lists of songs to the fourth form, the “Now Playing” form (Figure 7), which provides the functionality the user needs to actually play their music.

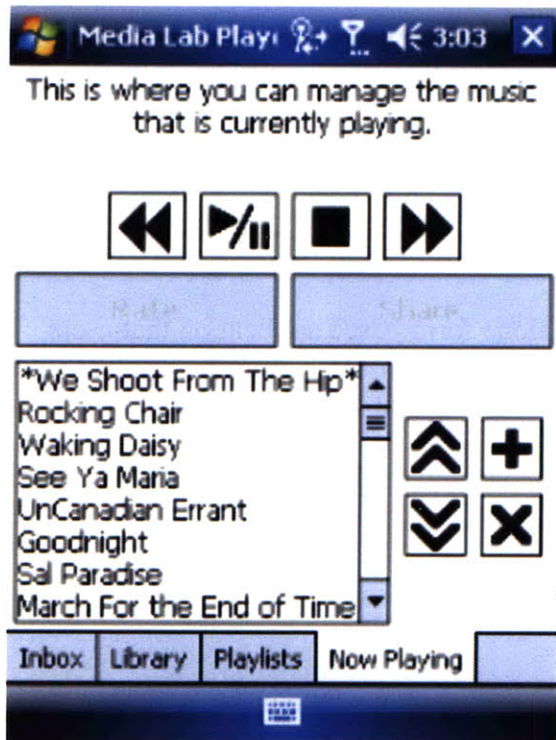


Figure 7: “Now Playing” tab for the Music Player application.

‘Rate’ allows users to assess the relative quality of different songs and is a required action before sharing music. ‘Share’ allows users to send their favorite songs to other users. Sharing also sends the user's rating of the song to the person who receives the music. This means that when a user looks at a song they receive they also know who among their friends liked the music and how much.

The Music Player accesses a central server to handle sharing. The server had a database with a table that stores the list of sharing transactions between individuals. Whenever a user shares a song with a different user a new row is be inserted into this table. Whenever a user navigates to their Inbox tab, the application queries this table to determine if any new shares have occurred where that user was the target. If the user has new songs, they are downloaded off the central server, along with associated metadata.

2.3.8 Updater

The Updater Application is written in C#. It is a standalone form-based application that users can start by looking in their 'Programs' folder. It checks the current version of all the experiment-related applications on the phone, checks the version on the server, and then updates the phone's applications as needed.

The updated applications are distributed through a CAB file system, which is the Windows compressed archive format. All of the experimental phones have development certificates installed on them, allowing the installation of any applications that are also signed by development certificates (the normal method of distributing applications on Windows Mobile devices includes a months-long approval process and costs several hundred dollars. This is perhaps the most obvious scalability problem with the SocialCircuits experimental platform).

The central server keeps copies of all the latest versions of the applications. The filenames indicate the current versions in a way that the client application can read. From a developer's

perspective, updating any given application now just involves making the change to the codebase, re-building the project (which outputs a signed CAB file), renaming the CAB file that Visual Studio builds to include the correct version number, and copying that new version to the repository of current applications. The next time that the subject runs the updater application, the change will occur. A restart will be required if the change is to the Scan application or survey launcher, which run continuously, but otherwise the change takes place immediately. The Updater Application can even seamlessly update itself!

The Updater Application allows the entire experimental codebase to be installed on a given phone in three steps:

1. Install development certificates
2. Install the Updater application
3. Run the Updater application

This is a dramatic increase in efficiency from earlier laborious techniques involving copying over applications one by one via USB.

2.3.9 Uploader

The Uploader application is designed to allow the phones to send the central server data on a daily basis. It turns on the phone's Wifi radio and attempts to connect to a valid network (not linksys). The Uploader sends the server all the data collected since the last successful transfer. It tries 3 times to turn on the Wifi radio and connect to the server before giving up for the day.

The Uploader runs as part of the main scanning loop, so the data files are guaranteed to be closed before the data is transferred. The Uploader does not have any user-facing component—hopefully the subject should not notice that information is being transferred. On the server side, there are tools to analyze which phones that have successfully uploaded data, and what types of errors there are, if any. If properly used, these tools dramatically increase the completeness of the data.

2.3.10 On-Device Survey Launcher

The Survey Launcher is designed to allow collection of data relating to changing daily habits. However, a daily survey runs the risk of being obtrusive, thus making the subject frustrated with the entire experiment and risking the completeness of the survey data. The goal is to collect as many survey responses as possible while interrupting the user's daily life as little as possible.

To support this goal, the survey window is limited to one screen full of questions (usually five or six questions), and it is cancellable twice. The survey first pops up at 9am (or as soon after 9am as the user turns their phone on). The 'Answer Later' allows the user to delay answering until 2pm, and then again until 10pm. If the user does not answer after 3 tries, the survey will never be answered for that day. There is no way to recall the survey.

The survey asks questions about the subject's behavior on the previous day. This period of time is short enough that the user likely remembers the activities of the previous day, but long enough to be sure that the users' activities for the day have been completed (for instance, if a question

asked “how many salads did you eat today,” a respondent can’t very well answer until the day is complete!)

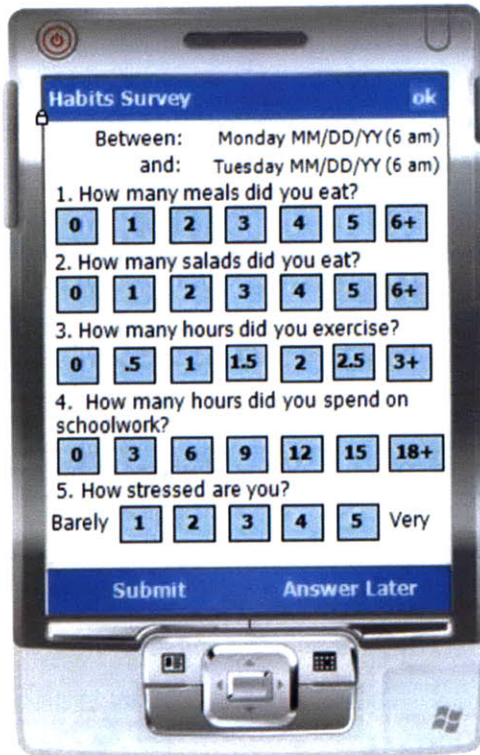


Figure 9: This is a representative survey that would be answered daily on the phone. Results are stored on the storage card and then uploaded to the central server.

The survey is actually two applications on the phone. The SurveyLauncher application starts on phone boot, like the Scan application, and runs constantly. The SurveyLauncher tracks whether the user has answered the survey for that day, and launches the survey at the correct times.

The survey form itself is a separate application, shown in Figure 9, which runs whenever the SurveyLauncher requests that it runs, and exits with a code indicating whether the survey was answered or delayed. It writes its responses to a file.

2.3.11 Feedback Engine

The survey can also be integrated with a feedback engine, which tracks all participants' previous survey responses and presents them to the subject.

In a feedback-enabled survey, the 'Submit' button becomes a 'View Feedback' button. The next screen shows the previous responses to the survey, as depicted in Figure 10.

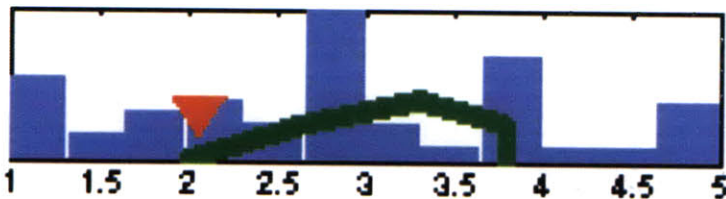


Figure 10: Sample feedback image shown to daily survey takers. The blue represents a histogram of all subjects' responses from the past week. The green represents 5 of the subject's friends as determined by proximity data, and the red represents the subject's responses. All responses are averages of the past week's worth of data.

2.3.12 Bluetooth Scanners

In order to get absolute location from Bluetooth data, Bluetooth scanners can be deployed in known locations within the experimental environment. These scanners monitor the surrounding 30 feet of space and record when experimental phones are present. These are Bluetooth-enabled phones that run the same Bluetooth scanner code that the experimental phones use, and then send

the results to our server over Wifi. These Bluetooth Scanners are designed to only record the existence of phones that are part of the experiment in order to alleviate privacy concerns related to constant monitoring of Bluetooth trails.

2.3.13 Bluetest

In order to test the validity of the Bluetooth co-location data, there is a Bluetest application that pops up every day and asks the user whether they have seen any of 10 subjects in the past minute. The user checks the ones that they have seen, and the application stores that data. The application asks about both subjects that have been detected by Bluetooth scan in the past minute, and subjects that have not.

The goal of the Bluetest is to understand how frequently false positives occur—that is, Bluetooth scans say that the subjects are co-located, but in reality a door or wall or the floor separates the two users. It is unclear to what extent solid objects attenuate the Bluetooth signal for any given experimental environment.

3. THE EXPERIMENT

During the 2008-2009 school year, the platform described above was used to run a continuous experiment in an undergraduate dormitory of about 90 people. The experiment measured diffusion of ideas and behaviors through a dense community in order to understand social influence on a fine-grained level. A timeline of the experiment is on Figure 11.

3.1 Constraints

The budget was \$50,000, which is enough to buy roughly 100 smart phones at \$400/each, and still have enough leftover to provide monetary incentives for sub-experiments and surveys. It is not enough to be able to buy phone or data plans for subjects. Therefore, we gave subjects free phones that they could keep forever, but they had to switch over their existing phone plan.

3.2 Community Selection

We chose a small dorm of around 90 students that was physically and socially distant from the rest of the campus. Over 65% percent residents reported that a majority of their friends and acquaintances lived inside the dorm. The dormitory was close-knit, social, and technophilic.

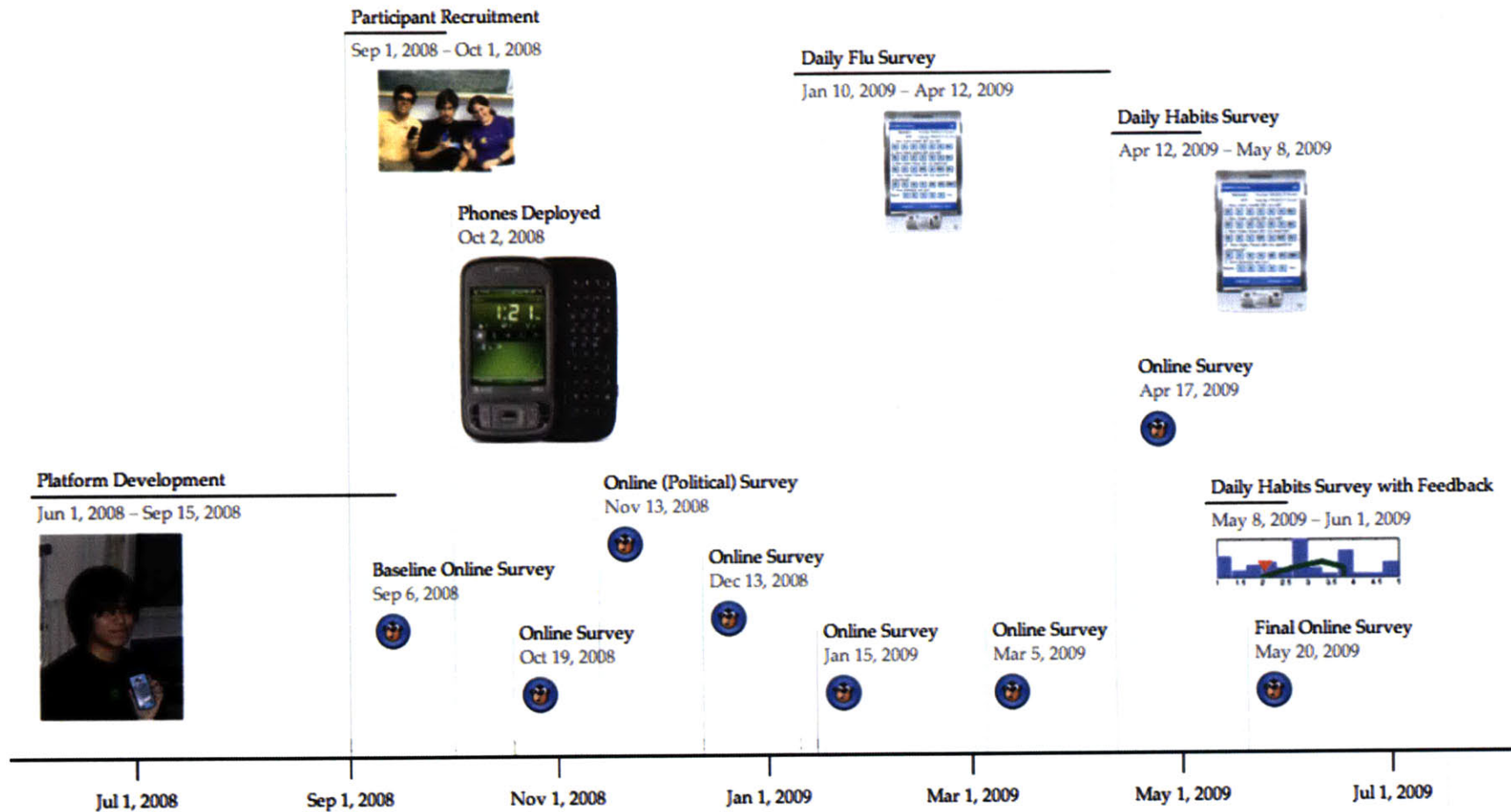


Figure 11: A timeline of deployment of the experiment.

A high rate of participation was achievable at the selected dormitory because Iolanthe Chronis used to live there as an undergraduate, which vastly simplified the process of marketing and explaining the research. She was able to have one-on-one conversations with every member of the community, and the community trusted her to protect their privacy and minimize their time commitment to the experiment. However, this also meant that that researcher was barred from looking at the raw data because she was a member of the community, and could potentially abuse or learn private information from the collected data.

3.3 Participant Recruitment

For an experiment to measure social diffusion in a community, participant density is more important than volume. In our case, it was important to account for as many factors as possible explaining why a particular subject changed their behavior, and hence necessary to capture as many of that subject's friends and acquaintances as possible.

Factors that impede a subject joining the experiment:

1. Too much time commitment
2. Privacy concerns
3. Already having a nice phone
4. Don't like the phone models or OS's offered

Factor 3 (already having a nice phone) is hard to wrestle with, except by selection of community.

We chose one of the undergraduate dormitories with the lowest rent, hoping that people who choose lower rent housing also have less expensive phones. Some members of the community

actually offered to give up their current smart phone for the course of the experiment because they were interested in taking part, but there were also about 4 members of the community with iPhones or Blackberries that could not be converted.

Factor 4 (don't like the phone models or OS's offered) is mostly determined by hardware platform selection, which as I discussed in Section X, was largely determined by other factors. The Windows Mobile phones we were offering were disliked by many members our target community because of their bulky size and overly complicated feature set. In fact, when the experiment ended about half of the subjects switched back to their old, less fancy phones. We lost about 8 subjects due to phone model or operating system concerns, mostly during the course of the experiment.

Factor 1 (too much time commitment) can be mitigated by personal contact with the subjects. An hour was spent with each individual making sure they understood exactly what was involved, and trying to lower energy barriers that would cause them not to join. We lost about 2 subjects due to "just not getting around to it."

Factor 2 (privacy concerns) was a significant issue, and cost us perhaps 10 subjects. Even after IRB approval, a few of our potential subjects were hesitant about participating in the experiment due to privacy concerns. Ironically, it was the very traits that we selected the community for that led to a problem—high social cohesion and co-influence within the community meant that when a few people had concerns, those concerns spread to their network. The most important potential privacy concern in this community was participant re-identification based on mobile phone data

hash, anonymize and remove any personal identifiers from the data at each stage of the collection process.

Another concern was the social pressure applied to participate. The high density required for participation, coupled with the sensitivity of the questions asked during the monthly survey, made a few potential participants uncomfortable. We alleviated this concern by decoupling the survey responses from the act of participation—answering surveys became a lucrative component that was not mandatory for participation in the experiment. All of the subjects filled out the monthly surveys anyway, and were more comfortable about doing so.

3.4 Carriers

For all subjects, we switched their current phone plan and phone number from their current phone to their experimental phone. For subjects whose carrier uses GSM, or SIM technology, switching just involved moving the SIM card from one phone to another. However, subjects whose plan involved a limited data policy had to ensure that they contacted their carrier to block the phone's data, otherwise the phone's automatic data connection policy would result in heavy charges.

Subjects with CDMA-based carriers (Verizon or Sprint) could not manually transfer their plans. In these cases, we helped each subject call up Verizon to have a representative transfer their phone plan. In many cases this also involved calling the subject's parents, re-pitching the experiment, and asking the parent to help us transfer the student's plan.

Switching Verizon plans was a relatively straightforward process at first, but in about the fourth month of the experiment, Verizon changed its policy to disallow people without data plans to switch that plan to a data-enabled phone. This meant that Verizon representatives stopped letting us switch over non-data plans. In the future, this will severely limit the density that is achievable by this type of study, unless the experimental budget is large enough to support the purchase of new plans for Verizon owners.

3.5 Participant Retainment

The study started with 75 people, and ended with 65. Keeping subjects was a constant source of work. We took steps to make sure people stayed happy with the experiment. We gave subjects whatever phone accessories they wanted, from larger batteries to extra styluses to phone covers. We found a company that lets phone owners design their own custom stickers that cover the full body of the phone, partly to allow participants to distinguish which phone belonged to them (imagine living in a dormitory where everyone has one of two different models of phones... for research reasons its in our interest to make sure that subjects don't accidentally grab the wrong phone on the way out the door in the morning). However, the skins also meant that the subjects invested effort and care in their phone—some of the phones looked like Pokemon, some like Van Gogh artwork—each was customized to its owner.

Reasons subjects dropped out:

1. Heightened privacy risk due to personal responsibilities
2. Safety concerns
3. Dropped out or moved out

4. Don't like the phone
5. Too buggy/too much of a time commitment

One subject realized that his/her position in a voluntary medical organization made it unethical to participate in the study—by participating, s/he puts her own privacy at risk, but also that of the people that s/he treats.

One subject looked at the radiation rating for all of the smartphones on the market, and discovered that the model of phone s/he was using had one of the worst available ratings. We do not anticipate this concern being widespread enough to merit including a radiation-free model of phone.

The normal churn of undergraduate life was the biggest source of attrition. People who leave the dormitory or leave MIT can no longer participate in a useful way, because they no longer interact daily with the rest of our subjects. A few subjects joined mid-study, but about 6 or 8 subjects had to leave the experiment due to moving out of the community.

Factors 4 and 5 contributed to a couple more people dropping out, and definitely created an ending deadline for the experiment. Many subjects wanted to switch back to their old phones, and had to be cajoled into remaining in the experiment until years' end, even given the promise of extra cash.

3.6 Data Collection

For the first half of the experiment, there was no automatic uploading system set up, so the data was collected through data dumps that subjects would perform on a computer set up for the purpose in the main community area of the dormitory. The files were uploaded to the central server, but a copy was kept on the phone as a backup.

In the second half of the experiment, the uploading system was deployed. This was both useful for detecting when a phone was not collecting data properly, and necessary for the sub-experiment involving feedback about other people's behavior.

However, the automated uploading system relies on Wifi, which is flakey due to the dorm's connectivity problems and Windows Mobile's poor UI concerning connectivity. Only about half of the phones managed to upload data successfully per day, and some phones never successfully uploaded data due to broken Wifi radios. Therefore all data was backed up on the phone so that it was always retrievable.

Server-side tools reported on how many of the phones were uploading data, and on whether any of the uploading phones had data collection problems.

The SD cards on the phone were the biggest hindrance to successful data collection. The cards are flash memory, and as such have a 30,000 write limit before they start failing. The scan application writes to the cards every 5 minutes, 24 hours a day. This means that the 30,000 write limit was reached after about 3 months. When this happened, many of the cards started failing, which for a time was inexplicable by the researchers. Usually the failure would take the form of

the data collection folder losing the recollection of its contents due to corruption in the folder structure. Later most of the data was recovered using the Linux command ‘ddrescue’ to copy over the contents of the disk, and then using the Linux command ‘strings’ to extract human-readable text. File manipulation tools like ‘grep’ were used to determine which text corresponded to which type of data, and to reconstruct the files.

Because of the possibility of data recovery, the past data loss was not huge. However, a failed disk would cause the phone to silently stop collecting data. Automatic uploading allowed quick discovery of this problem, but much data was lost due to this error in the earlier months of the experiment.

Paying a bounty on bugs was an invaluable way to identify and solve data collection problems quickly. The subjects would receive \$10 for every fresh bug that they found. Suddenly it was clear to all the subjects that they SHOULD absolutely “bother” the researchers with bugs—previously they’d been hesitant to tell us about errors because they felt that they’d be taking up our time. Publicly announcing the winner of the \$10 prize made bug reporting a game.

The other barrier to quick problem resolution is inability to track down individual subjects. Because the subjects are MIT undergraduates, their schedules are extremely busy, hindering debugging phones over email, or scheduling times to meet. This problem was resolved by hiring an “elite squad” of nontechnical study participants to help on the ground. They worked between 5 and 10 hours a week, mostly tracking down people and offering simple advice and bug fixes. Since they lived in the dormitory, even the most cagey individuals could not escape their reach.

Once the data was on the server, it was cleaned and anonymized using python scripts. It should be publicly accessible eventually at the following URL: <http://mob.media.mit.edu/>.

3.7 Sub-experiment compliance

Getting subjects to participate in smaller pieces of the project was also a major challenge. The sub-experiments included the Music Player, the Flu Survey, and the Habits Feedback Survey.

The easiest technique for ensuring user compliance is bribery. The subjects were given \$50 to fill out the monthly surveys each term. They were given \$1/day for filling out the daily surveys.

Ease of participation is also extremely important. The Flu Survey enjoyed a high response rate because participation was easy. Subjects answered ‘yes’ or ‘no’ questions that were easy to consider. The Habits Feedback Survey had a much lower response rate, even given identical incentives and the same number of questions, because remembering complicated questions like “how many salads did you eat yesterday” is more mental effort than many subjects are willing to put forward for \$1.

The biggest compliance challenge was the Music Player. After a brief period of exploration, very few users continued to use the application. Discussion with study participants revealed that the primary reason for the lack of use was the inability of users to integrate the application with their own music libraries. Unfortunately, due both to legal concerns and to an unwillingness to allow prior musical preferences to interfere with track popularity, integration was not possible. Even

with ever-increasing bribes, ending in a promise to distribute \$3000 according to use of the player, and substantial cajoling of subjects, usage of the player never increased to a meaningful level.

4. THE DATA

In total, 65,000 phone calls were recorded, with an average duration of 138 seconds, and about 58% occurring on weekends. The total number of sms messages was 25,000. There were 3.3 million scanned Bluetooth devices, and 2.5 million scanned wifi access points. This is equivalent to 320,000 hours of data.

About 8 to 10 percent of data is missing, usually from phones powering off or an early-experiment sleep-mode bug, or from broken phones.

Figure 12 shows an example of the temporal evolution of social interaction. We have this kind of data for the entirety of the course of the experiment.

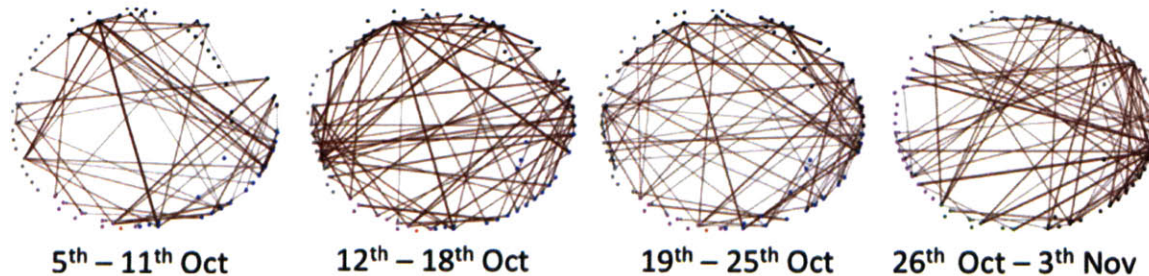


Figure 12: *Temporal evolution of interaction network for the month of October.*

Section 4.1 contains an example analysis of one piece of the data, performed by Anmol Madan.

There are many more such discoveries to be made in the next few years from this data set.

4.1. Example Application: The Adoption of Political Opinions

Political opinions were sampled in this community three times — September, October, and then November (within 3 days of the presidential election). The survey was designed as a Likert scale and included the following questions:

- *‘Are you liberal or conservative?’*

7-point scale, from ‘extremely conservative’ to ‘extremely liberal’

- *‘How interested are you in politics?’*

4-point scale, from ‘not interested’ to ‘very interested’

- *‘What is your political party preference?’*

7-point scale, from ‘strong Democrat’ to ‘strong Republican’

Using the platform, it is possible to model the dynamic exposure to different opinions for every individual. Contact between two individuals is a function of different mobile phone features— e.g. time spent together during the day or in classes, time spent socializing in the evenings or late at night, phone calls and SMS’s exchanged, the count of interactions or the total duration of interaction. It is possible to estimate two types of exposure based on these mobile phone features:

Normalized exposure represents the average of all opinions a person is exposed to on a daily basis, *weighted by the amount of exposure* to different individuals and their self-reported opinions. *Cumulative exposure* represents the magnitude of a particular opinion that a person is

exposed to on a daily basis, and is a function of the *amount* of contact with different individuals and their self-reported opinions.

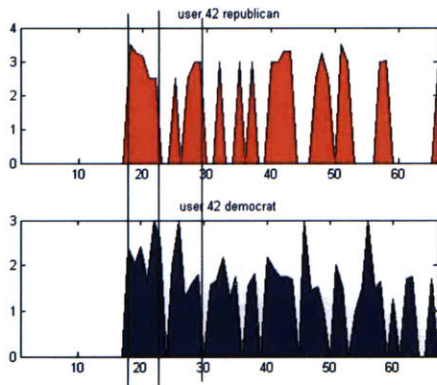


Figure 13: Normalized face-to-face exposure calculated for one individual over 2 months. The upper graph (red) represents exposure to republicans and the lower graph (blue) represents exposure to democrats. The vertical axis represents the intensity of opinion (i.e. 0 = independent, 1 = slight democrat, 2 = democrat, 3 = strong democrat, and similarly for Republicans). The horizontal axis is time over days. Overlaid vertical lines represent the days of the election debates and the day of presidential elections (4th Nov).

Modeling exposure to different opinions allows us to predict whether a person is likely to adopt or reject new opinions. Accounting for the normalized exposure to different opinions helps predict an individual’s future opinions better than using his/her past opinions and community-wide influences. For the ‘interest in politics’, ‘political party preference’ and ‘liberal or conservative’ questions, factoring in automatically-captured exposure explains an additional 15% variance, additional 9% variance and additional 6% variance respectively (over using past opinions alone). Exposure to different political opinions seems to play a bigger role for freshmen. For freshmen, factoring in ‘automatically-captured’ exposure improves the explained variance by 22%, 25% and 30% respectively, for the three questions [4].

5. FURTHER EXPERIMENTS

The success of the SocialCircuits study leads to imitation. The Human Dynamics group will be organizing another similar study starting in February named the Friends and Family study (FnF). The study will take place in a graduate dorm, and will focus on issues of identity and consumer spending behavior.

The Friends and Family study will in some ways well complement the SocialCircuits study.

Where the SocialCircuits study consisted a tight-knit and rapidly evolving community of atypical undergraduates, the FnF study will be deployed in a dormitory for married students, and many of the subjects will have young children. This will provide a sense of generalizability, allowing a comparison between the diffusion behavior in two very different communities.

The FnF study will be launched on the Android platform, allowing a wider range of sensing technologies than were used in the SocialCircuits study. The FnF team hopes to leverage these to develop some techniques for maximizing important data retrieval in the face of limited battery life by increasing rate of data collection when the phone's context changes. For example, if the accelerometer in the phone shows that the phone has not moved for one hour, the phone may be sitting on a desk, and collecting minute-by-minute data may not be important. However, if the noise level that the phone detects increases above a certain volume, than the subject may be in a noisy social setting, and taking more frequent co-location samples is desirable. The FnF study will also be able to use the Android app store to study consumer behaviors.

The Windows Mobile platform described in this platform may be reusable in the future if Microsoft releases a version of the phone that is competitive with the iPhone and Android phones in terms of usability. It is unlikely that the base architecture of Windows Mobile will change sufficiently to render the platform unusable.

6. THE TEAM

The research was performed as part of a project instigated by Professor Alex Pentland in support of the Ph.D. thesis research of Anmol Madan, and the MEng thesis of Iolanthe Chronis.

Experimental design was performed by Professor Pentland and Anmol Madan with help from Professor David Lazar and Dr. Devon Brewer. The design of the software platform was a collaboration between Iolanthe Chronis, Anmol Madan, and Alex Pentland.

Iolanthe Chronis co-designed the platform architecture and lead building of the experimental platform, handled participant selection and recruitment, and in general ran the day-to-day operation and development of the experiment.

Anmol Madan co-designed the platform architecture, handled phone acquisition, online survey creation, COUHES management, and data import and cleansing. He also performed the analysis of changing political opinions described in 4.1.

Griffin Chronis and Xiao Fan built the Music Player application.

Gabe Tobon and Mu He helped with data importing and cleaning.

Paul Kominers and Erons Ohienmhen helped with participant management and debugging. Paul also helped think about our privacy policy, and Erons helped with data import.

Chris Palmer helped build the Uploader and causality survey system, and built tools for analyzing the experimental performance of phones.

7. REFERENCES

- [1] Eagle, N., and Pentland, A. 2006. Reality Mining: Sensing Complex Social Systems. In Personal and Ubiquitous Computing, Vol 10, #4, 255-268.
- [2] Liao, L, Patterson, D.J., Fox, D., and Kautz, H. 2007. Learning and Inferring Transportation Routines. In Artificial Intelligence, 171(5-6), 2007, 311-331.
- [3] Madan A., and Pentland A. 2009. Modeling Social Diffusion Phenomena Using Reality Mining. In AAAI Spring Symposium on Human Behavior Modeling. Palo Alto, CA.
- [4] Madan A., and Pentland A. 2009. Social Sensing to Model the Evolution of Opinions. In submission.
- [5] Munguia Tapia, E., Intille, S., and Larson, K. 2007. Real-Time Recognition of Physical Activities and Their Intensities Using Wireless Accelerometers and a Heart Rate Monitor. In Proceedings of the 11th International Conference on Wearable Computers (ISWC '07). Boston, MA.
- [6] Olguin, D., Waber, B., Kim, T., Mohan, A., Ara, K., Pentland, A. 2009. Sensible Organizations: Technology and Methodology for Automatically Measuring Organizational Behavior. In IEEE Transactions on Systems, Man, and Cybernetics Part B. Vol. 29, February 2009.