A Case For Automatic Sharing over Social Networks

Pravin Shankar
Department of Computer Science*
Rutgers University
mail@spravin.com

Lu Han
Department of Computer Science
Rutgers University
luhan@cs.rutgers.edu

Vancheswaran K Ananthanarayanan
Department of Computer Science
Rutgers University
vanchi@cs.rutgers.edu

Matthew Muscari
Department of Computer Science
Rutgers University
muscarim@cs.rutgers.edu

Badri Nath
Department of Computer Science
Rutgers University
badri@cs.rutgers.edu

Liviu Iftode
Department of Computer Science
Rutgers University
iftode@cs.rutgers.edu

ABSTRACT

The proliferation of sharing on social networks, such as Facebook, Pinterest, Linkedin, and Foursquare, has closely connected people more than ever before. Our hypothesis is that, modulo privacy concerns, people are willing to share a lot more useful information than they are currently doing, but do not have the time and energy to do so. For instance, (1) people would like to share their current availability to receive a call with their most frequent phone contacts, (2) drivers would like to share road conditions with other drivers in the vicinity. By automating the sharing process, we posit that a whole new array of applications become possible. Smart mobile devices possess rich sensors, continuous connectivity and push notifications, which enable them to continuously share information with other devices, in real time. In this paper, we explore the opportunity for *automatic sharing*, and the challenges it introduces. We introduce a novel architecture called **SBone** which enables devices to automatically share useful information with each other, and present two applications that enable the above two scenarios by utilizing SBone.

1. INTRODUCTION

Online social networks such as Facebook, Pinterest, Linkedin, and Foursquare have surged in popularity. People share their personal status, pictures, videos, whereabouts, and links with their friends over these social networks. Facebook, one of the most popular social networks, has over half a billion daily active users and more than 125 billion friend connections. The growth of social networking phenomena has had a huge impact on society, making people more connected than ever before.

Barring some automatic event notifications such as friends' birthdays, today, almost all social network sharing is user initiated. Explicit sharing requires users' time and energy, so it is natural to expect a shift in trend towards automatic sharing. However, today, there are only two well-known examples of automatic sharing: (1). Smartphone users share their location continuously in the background, to get alerted when they are close to each other. (2). With the launch of the new frictionless sharing feature [7], Facebook users can share their activities outside Facebook, such as the songs played, videos watched, and news articles read on other websites. Both these applications are relatively narrow in scope.

People spend a large part of their lives outside online social networks and, until recently, their sharing ability has been limited to online activity. However, with the advent of smart mobile devices that possess rich sensors, continuous connectivity and push notifications, it is now possible to continuously share information with other devices, in real time. Nevertheless, there are several reasons why the trend of sharing on social networks has not transcended from explicit sharing to automatic real-time sharing. First, if naively implemented, automatic sharing can waste precious resources of smart devices, such as battery and network bandwidth. Second, the act of sharing requires users to specify who can access what information, which can often be a daunting task. Finally, and most importantly, people are seldom interested in knowing everything that is shared; they only want specific useful information. This requires a

[1]Pravin Shankar is currently at Green Dot Corporation, Mountain View, CA, USA.

transformation to be applied to the raw data that is shared to convert it to the specific useful piece of information.

If we consider for a moment that all the above challenges were solved and truly automatic, real-time sharing were possible, we could suddenly start to imagine a whole new array of applications, that, today, are just not possible.

- When making a phone call to a frequent contact, the caller does not know the callee's current availability, and if it is currently a good time to call. If this were possible, the caller could know if the person she is calling is driving or in a meeting, and could decide to call them later, thereby saving the callee a lot of inconvenience.

- When we are stuck in traffic due to unexpected events (such as accidents, road closures), we cannot tell other drivers who are about to enter this road segment to avoid it. We can merely watch helplessly as more cars pile into the traffic. If it were possible for cars to automatically share unexpected road conditions with other cars in the vicinity, a lot of traffic-related problems that exist today could be solved.

Our hypothesis is that there are numerous situations where users are willing to share useful information from their devices with other users, but are unable to do so. In some cases, this is due to technical limitations of the devices, and in other cases, it is because the users simply do not have the time and energy to do so. We posit that, by enabling automatic sharing, various new applications can be made possible without increasing the cognitive load on the users.

In this paper, we explore the design space of automatic sharing, the challenges it introduces, and how these challenges can be addressed via our implementation. We start by enumerating the challenges involved in automatic, real-time sharing. We then introduce our novel architecture **SBone** (Social Backbone) that enables devices to automatically share useful information with each other in real-time fashion. Finally, to demonstrate the wide applicability of the SBone architecture, we present two applications, SmartDial and RoadSense, that enable the two scenarios that we mentioned above.

2. RELATED WORK

There is a growing interest in the use of smart devices as rich sensors for environmental sensing, people-centric sensing, and participatory sensing [3, 4, 5]. In [12], the authors utilize the microphone of a smart phone to collect environmental sound for making sophisticated inferences about human activities, locations, and social events. In [14], the authors present CenceMe, which enables social network members to share their sensing presence with their buddies. Sensing presence captures a user's status in terms of his activity (e.g., sitting, walking, etc), disposition (e.g., happy or sad), habits (e.g., at the gym, coffee shop, work) and surroundings (e.g., noisy, hot, bright). Smartphones can also be utilized to detect and report the surface conditions of the roads [6], as well as estimate both a users trajectory and travel time along the route [25]. Smartphones are also increasingly being used as the device of choice for interacting with ubiquitous services [10, 20].

Mobile social networking applications on smart phones are growing in popularity. These applications create and utilize social networks where individuals with similar interests connect with one another through their mobile devices. Popular examples are FourSquare [8], Shozu [22], SociaLight [24], Aka'aki [1], and Path [16]. Users of a mobile social network *manually* push personal status to their friends. In some cases, the applications automatically share user location in the background.

Middleware and platforms that facilitate mobile applications over social networks have been developed, such as MobiClique [17] and MobiSoc [2]. MobiClique is a middleware that enables mobile applications to be developed without requiring a central server, by utilizing existing social networks as overlay. MobiSoc is a middleware that provides a platform for capturing, managing, and sharing the social state of physical communities.

To the best of our knowledge, none of these existing systems provides a platform for automatic, real-time device state sharing between owners without proactive user input. Furthermore, none of them includes personal devices other than smart phones. In SBone, we provide a general platform enabling automatic, real-time, continuous personal device state sharing for different devices, while providing support for higher level applications.

3. CHALLENGES IN AUTOMATIC DEVICE STATE SHARING

3.1 Resource management

Smartphones are powerful but resource constrained devices. Continuously sensing, sharing and receiving of device state can easily drain out the smart phone battery as well as exceed the allowed data limits in typical 3G/4G data plans. Efficient collection and communication of devices state becomes extremely important.

Authors in [11], [13], and [18] propose systems that balance the performance needs of continuous sensing and transmission with the resource consumption on the phone. We can optimize the battery usage through profiling by utilizing the historic usage patterns to estimate the future battery usage and the next charging opportunities [19]. Policies based on predictions and profiling, such as closing the low priority applications, or reducing the frequency of sensing when the battery is lower than a threshold, can help conserve the battery.

Unless the users have an unlimited data plan, they are charged based on their bandwidth usage. In order to provide users with real-time continuous state sharing, efficient data transmission polices need to be defined. The system would need to be cognizant of the user's specific data plan. In the case that the user has a limited data plan, the system should defer data transmission based on predicted WiFi access opportunities. The sensed device data can be buffered on the device, and shared with other devices/users when a WiFi access point is in range.

3.2 Access control

Automatic sharing involves collecting personal state of devices that the owner may or may not want to share with all contacts. Examples of personal state are the current location and activity of the user. Users may not mind sharing personal state of their devices with a select few friends, but would not want anyone else to access this information. Therefore, a system for automatic sharing needs to provide users with fine grained controls that allow them to satisfy their privacy needs, while involving minimal cognitive load on the users. Specifically, the users need to specify what device state is to be revealed to whom and under what conditions.

People do not have the same level of trust with all their virtual friends in a social network. A friend in a social network can refer to an acquaintance, a co-worker, a close friend, or a family member. However, fine-grained access controls are hard to specify for an average user. Typical users in a social network do a bad job when entrusted with complex security decisions [21]. One solution to the problem of specifying access control for state sharing is to exploit a hyper-graph model, which we discuss in Section 4.2.2.

3.3 Data Transformation

Devices generate and collect a large amount of data all the time, however, users are not interested in most of this data. Users only care about some specific useful information, which is a function of the raw sensed data from the device. An example of this transformation is inferring user availability from raw smart phone sensors. Different raw data that could be used as inputs are ring status (Loud, Silent, or Vibrate), call logs, battery status, accelerometer data, and location.

As automatic sharing systems become more widely adopted, this transformation from continuously generated raw device data to useful information will become an important research challenge for the data mining community. Data transformation includes aggregation of multiple data sources, ranging from multiple devices of a user, to devices belonging to multiple users. For example, by applying an aggregation function to the raw data from several drivers in a road segment, it is possible to infer the road conditions in that road segment.

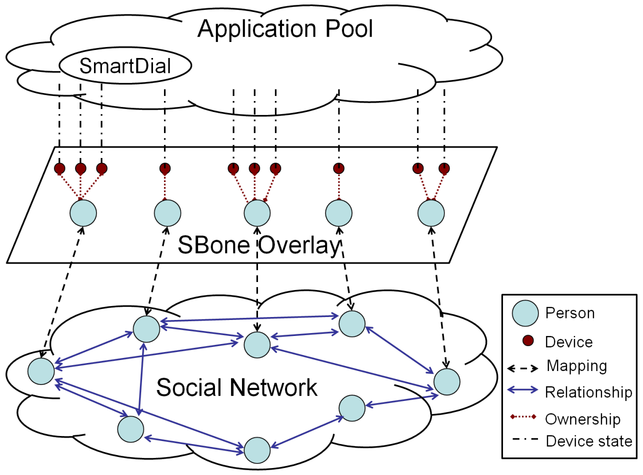


Figure 1: SBone overview: social graph of devices and their owners, and the application pool.

4. SBONE OVERVIEW AND DESIGN

SBone is a platform that facilitates secure, seamless, automatic, real-time state sharing among devices whose owners are connected by social networks. SBone establishes trust between devices by leveraging the relationships that already exist between device owners in the social network. SBone provides this shared personal device state to any application on top, so that the applications can utilize this device state for automatic real-time state inference and sharing.

4.1 SBone Overview

The overview of SBone architecture is shown in Figure 1, where the lower layer can be any underlying social network, middle layer is the the SBone overlay consisting of physical devices and users, and the upper layer is the application pool. The SBone overlay consists of ownership links between personal devices and users who own them. Each personal device has a state vector, which consists of device state that is useful to other devices and upper layer applications, both device-specific (such as battery, ring status, location) and environment-specific (such as traffic, temperature, pollution). Devices are owned by users belonging to one or more social networks. Edges in these social networks represent social relationships between users. SBone leverages these social relationships between users to connect devices.

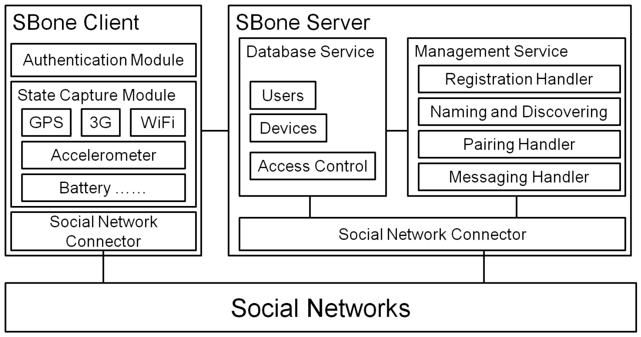


Figure 2: SBone client and server architecture.

4.2 SBone Design

The architecture for SBone client and server are shown in Figure 2. SBone consists of a number of components both at the client side and the server side. A personal device needs to first register with SBone, and specify access control policy for allow SBone to access its state vector. The SBone naming component assigns a globally unique name to the device and maps it to the device's network identifier. Other devices discover a device by either browsing or querying based on relationship and attributes. Once a device finds another device of interest, the pairing component handles secure device-to-device pairing. Finally, SBone allows state sharing between the devices. We will discuss these components in detail in the following sections.

4.2.1 *Registering a Device*

The first step is to register a personal device to one or more social networks that the owner belongs to. SBone must verify that the user owns the device. Systems that support mobile clients typically make use of a trusted out-of-band authentication channel, such as SMS, to allow the user to prove ownership of the device. However, not all devices have the ability to send and receive SMSes. In order to support a broad variety of personal devices and not limited to Smartphones, we make use of location-limited channels, such as wireless, bluetooth, infrared or USB, to authenticate a device.

Once the device is registered, it generates a key pair and a certificate, which are stored on the newly added device. The device's public key is also stored on the SBone server, which plays the role of a certificate authority (CA).

Since a user may remove a device from the system (say when the device is lost, or the user no longer wants to use it), SBone has to maintain a certificate revocation list(CRL). A limitation of using CRL is that devices need to have connectivity with the SBone server at all times. SBone device registration is done by the *Registration Handler* at the SBone Server side, and by the *Authentication Module* at the client side, as shown in Figure 2

4.2.2 *Specifying Access Control Policy*

Once a personal device is registered to the system, the user can specify which sensors of the device can be accessed. As users have different level of trust with their friends in the social networks, we propose to implement a hypergraph of the social network, with each edge to represent a relationship, such as friend, close friend, family member, co-worker, etc. Access control policy is specified per user in the form of a matrix, where the rows are the state vectors, and the columns are the relationships.

Typical users have only a limited number of relationship edges that they are part of. This way, the rules that needed to be specified for each device can be limited to a manageable number. Relationships can inherit other relationships. For example, the close friend relationship inherits the friend relationship, and the sibling relationship inherits the family member relationship. Users can define a precedence order among relationships, which is used between users who are connected to each other by multiple relationships.

There are two steps in the policy specification process: forming the relationship hypergraph, and specifying access control rules. The first step is performed each time a new friendship link is created in the social network. The second step is performed whenever a user adds a new device. To form the relationship hypergraph, the system needs to understand the semantic meaning of an edge between users in a social network. This can be done by a combination of automated and manual techniques. First, the system mines the interactions in the social network to infer inter-personal relationships. Next, the system presents the inferred relationship hypergraph to users, and users can manually make changes.

4.2.3 *Device Management*

Device naming, discovery, and paring are handled by the *Naming and Discovering Handler* and *Pair Handler* component in the SBone server, respectively. SBone identifies each personal device using a name that is specified by the owner when registering the device. This name is unique within the device namespace of the user. Sbone uses the combination of user name and the personal device name to arrive at a globally unique personal name for each device. Each time the device is connected, SBone maps the personal name to a routable address. The SBone client running on the device connects to the SBone server, and specifies this mapping.

Users can access the device namespace of their friends by either browsing or querying, subject to the access control policy. Once devices find each other, they perform secure pairing handled by *Paring Handler*. by exchanging their public key certificates with each other. SBone uses the public keys stored in the SBone server to perform the device-to-device pairing.

4.2.4 *Accessing Device State*

Device state is captured and shared with other devices/users by the SBone *State Capture Module* component at the client-side. Devices may have limited battery, data plan and communication bandwidth. If the state of a device is popular, repeated queries may place undue strain on the device's resources. Therefore, we cache the results of the query on the SBone server, and maintain a staleness indication to indicate how fresh the state information is, using the last updated timestamps.

5. SBONE APPLICATIONS

In this section, we present two representative SBone applications, SmartDial and RoadSense. SmartDial is an Android application that we have implemented as a proof of concept, which automatically shares users availability for answering a call with their most frequent phone contacts. RoadSense is an application that we plan to build, which automatically shares traffic conditions between drivers in the same road segment.

5.1 SmartDial

Smart phones can easily become a social burden when they ring at inopportune times, such as during a meeting, or while the user is driving without a hands-free kit. At these awkward moments, the recipient has to manually put the phone in silent mode or turn it off. SmartDial is designed to prevent these situations from happening, and allow users to share their availability for receiving a call automatically with the most frequently contacted friends in their contact lists. This way, the user's availability is converted into a dynamic state vector that can be shared with desired people. This availability, also referred to as *Personal State or User State*, is inferred from the user's smart phone state that is made available by the SBone. The availability of their contact list friends, who are also using SmartDial, can also be automatically shared.

The usage of SmartDial is illustrated by the following scenario. Bob and Alice are real life friends, and have each other in their phone contact lists. Both Bob and Alice have installed the SmartDial application. During normal operation, the SmartDial application is running in the background, continuously inferring the availability of the user for calls to update their friends without manual intervention. SmartDial classifies user availability as either "Red" (unavailable) or "Green" (available).

Suppose Alice wants to call Bob to schedule an after-work hang-out. She checks Bob's current state via SmartDial, which is shown in Figure 3(a). The SmartDial application shows that Bob is unable to answer phone call

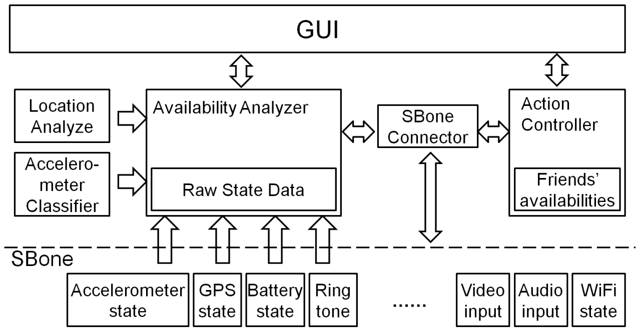


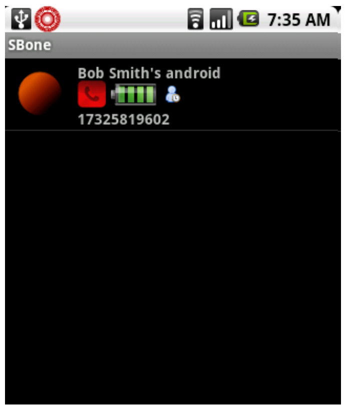
Figure 4: Architecture of SmartDial.

at this time since the state of Bob is shown as *Red*(with a red icon). Instead of calling Bob, Alice can now choose from the two options provided by the SmartDial system, "Send SMS" or "Call Later" as shown in Figure 3(b). "Send SMS" does as the title suggests. Alternately, she can choose the "Call later" option, so that the system will automatically notify her, when Bob becomes available (*Green* state), as illustrated in Figure 3(c). The bar on the top shows the notification that Alice gets once Bob becomes available.

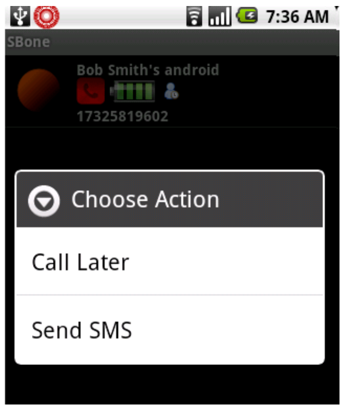
Modern smart phones have rich sensors that can provide a plethora of information about the user's personal context. SmartDial utilizes device context provided by the underlaying SBone platform (GPS/WiFi, accelerometer, ring state, and call state) to infer the users' availability. Conveying this information is completely under the control of the user who can decide to turn it off at any time. Also, the individual state vector is never shared with anyone, only the current availability status (*Red* or *Green*) is shared.

The architecture of the SmartDial is shown in Figure 4. The embedded GPS receiver, which provides the location and velocity, is used as location identifier. GPS state can be used to infer if the user is stationary, walking or in a moving vehicle. The user is very likely to answer the phone if he is stationary, but very unlikely to answer the phone if he is driving. SmartDial *Location Analyzer* (shown in Figure 4) analyzes the GPS/WiFi state obtained from SBone to infer if the user is available. The accelerometer sensor is used to infer if the Smartphone is stationary or in motion, as well as the idle time, if stationary. The ring state of the Smartphone can be in silent, vibrate or ring mode. People usually turn their phones to silent or vibrate mode when they are in a meeting. Therefore, the ring tone status is also used to infer suitability for making a call. Finally, the *Call State* tells if the user is currently in a voice call. After receiving the raw state vector from SBone, SmartDial *Availability Classifier* will infer users' availability, and share this with the most frequently contacted friends in the user's contact lists.

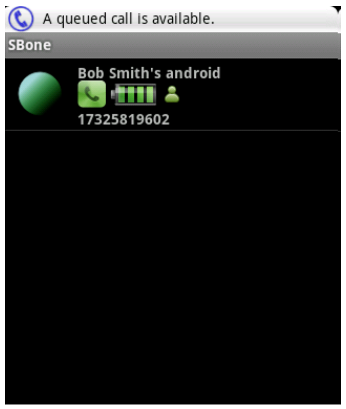
The *Action Controller* handles the user's actions when



(a) Bob is unavailable for answering phone calls.



(b) Alice's choices when Bob cannot answer the phone.



(c) Alice gets a notification (on the top) when Bob becomes available.

Figure 3: SmartDial Usage Scenario.

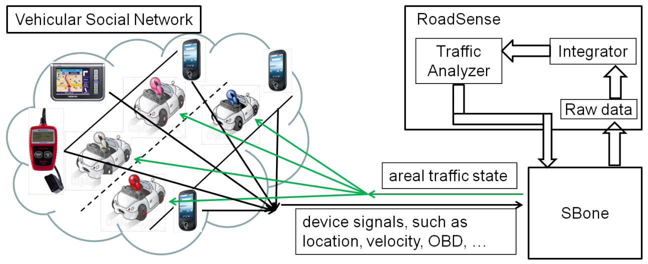


Figure 5: RoadSense overview.

the callee is unavailable. Two options, *SMS* or *Call Later* are provided, as discussed earlier. The *SBone Connector* controls the communication between the SmartDial and the SBone platform.

5.2 RoadSense

Driving is often a physically and mentally exhausting activity, especially when the weather is bad or traffic is congested. Getting to know the possible traffic information ahead of one's way is very useful for drivers to avoid possible hazards and inconveniences.

In this section, we introduce another SBone application, RoadSense, which allows users to get rich, real-time areal traffic state. This areal traffic state is collected, integrated, and inferred from the personal devices carried by users who are driving in the same area and belonging to the same *Vehicular Social Network group*, as described in [23], and [9]. Besides regular traffic information such as velocity and road congestion, RoadSense also automatically updates users with real-time traffic state, such as if the road is slippery or hazardous.

The RoadSense architecture and information flow is shown in Figure 5. The left-side of the Figure illustrates the concept of a Vehicular Social Network group formed by the automobile drivers for real-time traffic information sharing. The right-side of the figure shows the SBone platform that enables automatic state sharing, as well as provides the state to RoadSense for integration and inference.

RoadSense is enabled by three factors top-down, the personal devices carried by the drivers to sense the traffic and road state, the SBone overlay, and the underlay Vehicular Social Networks formed by the drivers.

The first enabler is the underlay Vehicular Social Networks (VSN) [23], [9], which is formed by the drivers who are driving on the same road segment at the same time and are willing to communicate and share.

The second enabler, the SBone platform on top of Vehicular Social Networks, enables traffic and road state from different personal devices to be collected. The RoadSense service processes and integrates the state from SBone to infer the traffic and road state around user's location, and automatically share this state among friends in the Vehicular Social Network continuously.

Finally, personal smart phones, which have been widely used for information collection and exchange in vehicular systems, can provide better real time traffic state. The camera of the smart phone can take pictures/videos about road segments while the driver is driving. Images can be shared to show a "Real-Time Street View" analogous to Google Street View. Other personal devices can also provide practical road and location information besides smart phones. The On-Board Diagnostic (OBD) [15] readers report the condition of the vehicle, such as emission levels, and anti-lock braking (ABS). The OBD state can be used to infer if the automobile is on a slippery road or if the pollution level on the road is higher than the safe threshold.

6. SBONE IMPLEMENTATION

The SBone prototype consists of two components: the *SBone server*, and the *SBone client*. The server manages users and devices by using the Extensible Messaging and Presence Protocol (XMPP) [26] server for registration, naming, discovery, and pairing. We chose XMPP since the protocol provides mechanisms for naming, presence and messaging. We used Jabberd2, an open-source XMPP server. The server stores the user and device information in a Mysql database, which is accessed by the XMPP server.

The SBone client runs a XMPP client, which connects to the SBone server using the owner's social network credentials for registration. The client sends list of resources it wants to be shared to the server, and executes the device sharing lifecycle as described earlier in the paper.

7. CONCLUSION AND FUTURE WORK

In this paper, we presented the design and implementation of SBone, which is a novel scalable platform to facilitate automatic real-time sharing among users/devices connected by a social network. As a proof of concept, we presented the design and implementation of two novel applications, SmartDial and RoadSense, which automatically share real-time personal state using the SBone system.

Our planned future work covers two directions: enhancing SBone, and desiging more SBone applications similar to SmartDial and RoadSense. An example application that we are currently building is integrating gaming consoles with SBone so that they can provide real-time gaming information between users.

For SmartDial, we plan to define fine-grained personal activities besides *availability*, The possible activities are *driving, cycling, away from the phone*, etc. The speaker of the smart phone can be leveraged to detect if the user is a driver or a passenger in the car [27]. The microphone of the smart phone can be used to collect environmental sound to infer the user's activities, such as if they are speaking with a friend, or listening to music [12].

Finally, we plan to incorporate better and more fine-grained privacy controls to our system, and provide a user-friendly interface for users to control their privacy settings when sharing with other users.

8. ACKNOWLEDGEMENT

We thank the anonymous reviewers for their insightful comments and suggestions. This work is supported in part by the National Science Foundation grant CNS-1111811.

9. REFERENCES

[1] aka'aki. http://www.aka-aki.com/.

[2] Borcea, C., Gupta, A., Kalra, A., Jones, Q., and Iftode, L. The mobisoc middleware for mobile social computing: challenges, design, and early experiences. In *Proceedings of the 1st international conference on Mobile Wireless MiddleWARE, Operating Systems, and Applications* (2008), MOBILWARE '08.

[3] Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., and Srivastava, M. B. Participatory sensing. In *Workshop on World-Sensor-Web (WSW'06)* (2006), pp. 117–134.

[4] Campbell, A. T., Eisenman, S. B., Lane, N. D., Miluzzo, E., and Peterson, R. A. People-centric urban sensing. In *Proceedings of the 2nd annual international workshop on Wireless internet* (New York, NY, USA, 2006).

[5] Campbell, A. T., Eisenman, S. B., Lane, N. D., Miluzzo, E., Peterson, R. A., Lu, H., Zheng, X., Musolesi, M., Fodor, K., and Ahn, G.-S. The rise of people-centric sensing. *IEEE Internet Computing 12*, 4 (July 2008), 12–21.

[6] Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., and Balakrishnan, H. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. In *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008)* (Breckenridge, U.S.A., June 2008).

[7] Frictionless sharing. http://en.wikipedia.org/wiki/Frictionless_sharing.

[8] Foursquare. http://www.foursquare.com/.

[9] Han, L., Smaldone, S., Shankar, P., Boyce, J., and Iftode, L. Ad-hoc voice-based group communication. In *Eigth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2010, March 29 - April 2, 2010, Mannheim, Germany* (2010), IEEE Computer Society, pp. 190–198.

[10] Iftode, L., Borcea, C., Ravi, N., Kang, P., and Zhou, P. Smart phone: An embedded system for universal interactions. *Future Trends of Distributed Computing Systems, IEEE International Workshop 0* (2004).

[11] Kim, D. H., Kim, Y., Estrin, D., and Srivastava, M. B. Sensloc: sensing everyday places and paths using less energy. In *SenSys* (2010), pp. 43–56.

[12] Lu, H., Pan, W., Lane, N. D., Choudhury, T., and Campbell, A. T. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services* (New York, NY, USA,

2009).

[13] Lu, H., Yang, J., Liu, Z., Lane, N. D., Choudhury, T., and Campbell, A. T. The jigsaw continuous sensing engine for mobile phone applications. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems* (New York, NY, USA, 2010), pp. 71–84.

[14] Miluzzo, E., Lane, N. D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S. B., Zheng, X., and Campbell, A. T. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems* (New York, NY, USA, 2008), SenSys '08, ACM.

[15] On-board diagnostic. http://epa.gov/obd/.

[16] Path. https://path.com/.

[17] Pietiläinen, A.-K., Oliver, E., LeBrun, J., Varghese, G., and Diot, C. Mobiclique: middleware for mobile social networking. In *Proceedings of the 2nd ACM workshop on Online social networks* (New York, NY, USA, 2009), WOSN '09.

[18] Priyantha, N., Lymberopoulos, D., and Liu, J. EERS: Energy efficient responsive sleeping on mobile phones. In *Proceedings of PhoneSense 2010* (Nov. 2010).

[19] Ravi, N., Scott, J., Han, L., and Iftode, L. Context-aware battery management for mobile phones. In *Proceedings of the IEEE Conference on Pervasive Computing and Communications* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 224–233.

[20] Ravi, N., Stern, P., Desai, N., and Iftode, L. Accessing ubiquitous services using smart phones. In *In Proc. IEEE 3rd Intl Conf. on Pervasive Computing and Communications (PERCOM 05), pp 383 393, Kauai* (2005).

[21] Sasse, M. A., Brostoff, S., and Weirich, D. Transforming the weakest link: in a human/computer interaction approach to usable and effective security. *BT Technology Journal 19*, 3 (July 2001), 122–131.

[22] Shozu. http://www.shozu.com/portal/index.do.

[23] Smaldone, S., Han, L., Shankar, P., and Iftode, L. RoadSpeak: enabling voice chat on roadways using vehicular social networks. In *SocialNets '08: Proceedings of the 1st workshop on Social network systems* (New York, NY, USA, 2008), pp. 43–48.

[24] Socialight. http://www.socialight.com/.

[25] Thiagarajan, A., Ravindranath, L. S., LaCurts, K., Toledo, S., Eriksson, J., Madden, S., and Balakrishnan, H. VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys)* (Berkeley, CA, November 2009).

[26] Xmpp standards foundation. http://www.xmpp.org/.

[27] Yang, J., Sidhom, S., Chandrasekaran, G., Vu, T., Liu, H., Cecan, N., Chen, Y., Gruteser, M., and Martin, R. P. Detecting driver phone use leveraging car speakers. In *Proceedings of the 17th annual international conference on Mobile computing and networking* (New York, NY, USA, 2011), pp. 97–108.