Turn-by-Turn Directions Go Social

Thomas Sandholm
HP Labs
Palo Alto, CA, USA
thomas.e.sandholm@hp.com

Hang Maxime Ung
Ecole Polytechnique
Paris, France
hang.ung@polytechnique.edu

ABSTRACT

In this paper we present the implementation of a system that allows audio-based real-time coordination of a group of users with mobile devices. Use cases include, real-time meeting point coordination, im-to-voice communication, and social sports tracking. The assumption is that at least one person in the group is able to easily enter text from a keyboard-like control, e.g. from a desktop, PC or tablet. This person, who we call the coordinator, can then communicate with one or more people, called operatives, with mobile devices and engaged in an activity that makes it hard or impossible for them to see the screen of the device and to use touch-based input mechanisms. Examples include driving, running, biking, and walking. We scope our work to only look at use cases where the operatives never have to provide any explicit input back to the coordinator apart from automatically detected device properties, such as geolocation. A secondary goal is that the only system requirement both for the coordinator and the operatives is a browser capable of rendering HTML 5 content to allow coordination across a diverse fleet of devices. The main lesson learned from our work is that audio cues can be very useful in a mobile setting to convey system information, activity by friends, as well as direct possibly translated text-to-speech messages. Experiments show that our infrastructure can potentially handle up to 78 people submitting locations in real-time (every 10 seconds) to a coordinator within the same group.

Categories and Subject Descriptors

H5.5 [**Information interfaces and presentation**]: Sound and Music Computing

General Terms

Design, Human Factors

Keywords

Location-based Service, Location-casting, Audiocasting

IWS '11, August 30, 2011, Stockholm, Sweden

1. INTRODUCTION

As mobile devices are becoming as common-place as PCs, and connectivity anywhere, anytime as well as geo-location sensing (e.g. GPS) have become the norm, a new wave of location-based services (LBS) have appeared. Most mobile LBS applications assume that the users are able to interact directly with the device through a touch interface or soft keyboard, and by monitoring the screen. There are, however, many mobile scenarios where this assumption is violated or where it renders the interaction impractical or unsafe, such as when driving a car, biking or running. Applications with voice-based, turn-by-turn directions provide a great example of how one-way voice interactions can be useful in such settings. Collaborative one-way audio applications with the purpose of coordinating and communicating within ad-hoc groups of people have, however, not been explored to a great extent so far. In the work described here, we want to enable live, social, location-based, turn-by-turn inspired audio interactions in order to allow communication in mobile situations where you would not be able communicate effectively today. In our system the text that is read to the mobile client does not come from a road feature database but is rather entered in real-time by your friends. The person entering the text can see where a dynamically created, short-lived group of friends are located in real-time in order to send them coordination messages through the audio channel. Some messages are also auto-generated to make the mobile users aware of each other's whereabouts.

A simple example is when you are in a meeting and want to communicate in real-time with your friend who is driving a car. A phone call might be optimal for the driver while IM or SMS would be optimal for you. In this case our system provides the optimal interface for each user while still allowing them to communicate in real-time.

There are two key challenges in this work:

1. how to seamlessly set up private ad-hoc groups to share location and voice messages across a wide array of smartphone and Web clients, and
2. how to represent real-time activity in a scalable way with high fidelity.

Our main contribution in this paper is lessons learned from implementing and deploying an end-to-end mobile voice coordination system. Some of these lessons relate to the state

of new Web technologies, such as HTML 5 geolocation, audio and real-time sockets on various mobile platforms. Our secondary contribution is a real-time Web and mobile device throughput experiment.

We first review some related work in Section 2, then we give an overview of the main features of the system in Section 3. In Section 4 we present the architecture and technology used to build the system. Section 5 details some use cases, and in Section 6 and Section 7 we discuss lessons learned from some field testing and evaluate the real-time throughput of the system. Finally, in Section 8 we conclude with an outlook on possible future extensions of this work and related use cases to explore.

2. RELATED WORK

There is a wide array of location sharing applications. In line with the analysis made by Tang et al. ([5]), we distinguish earlier purpose-driven location sharing tools such as the "Whereabouts Clock" [1] or "WatchMe" [2], from the more recent, one-to-all and socially driven, location sharing mobile applications, like Foursquare [1]. The latter category of applications, which relies mostly on micro-blogging platforms, is enjoying great success in spite of a persistent debate over privacy concerns [6]. Our approach differs from both categories in that you share your location by joining a group for a limited time; and sharing stops when you close your browser window. From this standpoint, our work is more closely related to real-time online collaborative text pads like Etherpad [2] or TypeWith.me [3].

AsyncVoice [4] is a research project at Ericsson Labs. The project has similar goals to our project of providing voice communication to multiple parties over the Web. However, the system bases the communication on RSS, Atom, and Pubsubhubbub push technology as opposed to real-time HTML 5 WebSocket communication. Furthermore, it sends the actual voice data, not the text across the wire, and it is not based on a TTS engine at its core. All these differences make our solution more lightweight and appropriate for small chatty and frequent real-time messages. A major difference is also our integration with location-casting features.

Sawhney and Schmandt have developed a wearable audio interaction device, the "Nomadic Radio" [3] which provides hands-free audio interaction capabilities for general purpose computer interactions such as reading email, checking your calendar, and reading news. This type of application has been referred to as "everywhere-messaging" [4]. We focus on more specific mobile use cases, where your and your friends' locations are central and need to be shared. Furthermore, we provide our system on standard smart phone Web browser platforms and your device may simply be in your pocket while using the application. We deliver text-to-speech but currently do not address speech recognition since microphone access is still not supported from most mobile Web browsers.

[1]http://foursquare.com
[2]http://etherpad.com
[3]http://typewith.me
[4]http://asyncvoice.com

Glympse [5], is a new app recently introduced on the iPhone and Android markets. It allows you to broadcast your location to friends for a predefined amount of time. Our solution differs from Glympse in that we allow voice communication and sound cues as well as monitoring and coordination of a larger group of mobile users concurrently. We also allow the session to end at any time requested by the sharing party. Architecturally, our solution is built on pure Web technologies, whereas Glympse adopts the common model of having different native apps for each platform. The obvious advantage of levering standard Web technologies is a much smaller effort to port to new devices and platforms, by taking advantage of standard browser capabilities. Furthermore, it allows us the reuse many existing JavaScript libraries developed for and used by widely successful social media applications today. Examples include the Google Maps API [6] and JQuery [7], which in turn have already implemented many features to be portable and render nicely across many platforms.

3. FEATURES

Our system, RESA (REal-time, Social Audio and location casting), allows ad-hoc chat-room-like interactions, where anyone can create an audio-location group and invite members simply by sharing a URL. Two real-time streams of events may be subscribed to, corresponding to whether you are a coordinator or operative. Operatives will continuously send their location to the coordinator stream. Coordinators will send text messages to the operative stream. These messages are then translated in real-time to speech in the language of choice of the operative.

The coordinator has a map interface where the traces of operatives' paths, and their last location updates are visualized. In addition to the map interface there is also an instant messaging interface where text messages can be broadcast to all members of the group or to a subset of members. All interactions are handled through a standard web browser. A screenshot of the coordinator interface is shown in Figure 1. This interface for the coordinators has the following map-related features:

- see traces of all operatives' movements in the group
- have the map automatically adjust to fit all the operatives
- let the map follow a particular user
- see the streetview level map move and rotate based on the followed user's heading
- define a target position

The interface also has these communication features:

- broadcast text messages to all operatives in a range of languages that are then translated and communicated through synthesized voice to operatives

[5]http://www.glympse.com
[6]http://code.google.com/apis/maps/index.html
[7]http://www.jquery.com

- send text messages to individual operatives
- invite new operatives to the group
- monitor when the operatives sent their last position

The operative interface is also offered through a Web interface in a browser. However, this interface is designed to be hands-free and eyes-free. Some status messages are displayed for debugging, but in general all information is communicated with sounds without requiring any touch or visual access to the device (see Figure 2). It is easy to switch into visual mode and act as a coordinator in the same group in order to monitor others visually. Whether you are a coordinator or an operative in a group is just determined by which information stream you subscribe to and what user interface you are presented with. Groups may have any number of both coordinators and operatives, although at least one of each is assumed. Coordinators may invite operatives to join a group and operatives may invite coordinators to join a group. The same user may be both an operative and a coordinator at the same time by opening up the two interfaces in two browser windows. Similarly, a user may be an operative and coordinator in many groups concurrently, although at least the former is not as likely in the use cases we are currently considering.

The sounds communicated fall into three broad categories:

- Coordinator messages. Text messages from the coordinator are converted to speech on the fly using an on-line TTS engine. These messages can be sent to all operatives or restricted to a specific one.
- Client status. Audio tones, denoting success and error, are played when a connection to the server is established and when the connection drops off, respectively. If the GPS and internet connectivity allow accurate enough information to be shared via the coordinator stream, a subtle audio tone is played.
- Operatives' location (optional). The location of operatives is reverse-geocoded into addresses, and broadcast in real time. If a target position is defined, the distance of operatives to this target can also be broadcast.

To monitor a group simply open the coordinator URL [8]:
http://*root*/monitor.html?group=*group*
The *root* parameter is in our case www.crowdee.com/realtime.
To join a group from a mobile device a user may simply open the operative URL:
http://*root*/group.html?group=*group*&name=*user*&pic=*url*
The pic parameter is optional. To test the coordinator interface with simulated operatives you can add the sim parameter to the mobile device (operative) URL above, specifying an integer between 1 and 15. We also provide a URL shortening service that coordinators could use to create and email/SMS clients to visit the URLs above. Devices generate a unique id which is used to identify messages coming

[8]On a desktop/laptop Chrome is the preferred browser and Android 2.2+, webOS 2.0+, or iOS 4.1+ are currently preferred as mobile clients

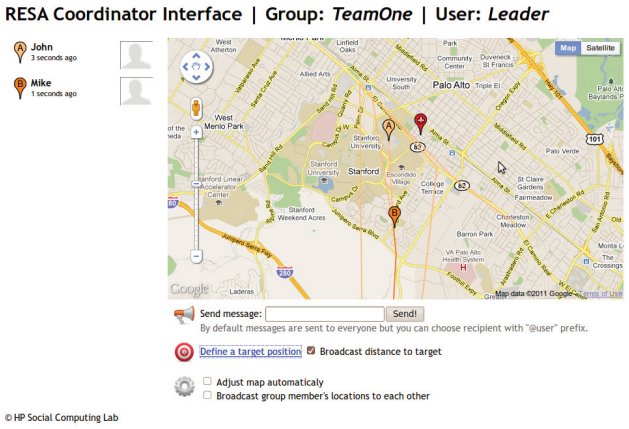

Figure 1: RESA Coordinator Interface. Web interface with ability to track RESA users and send them voice messages.

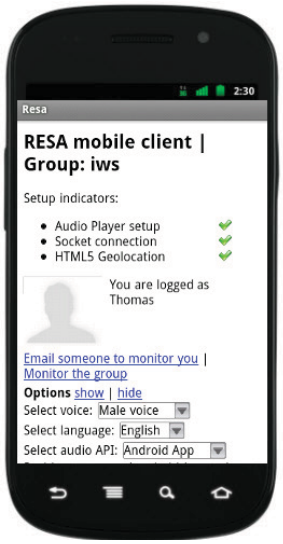

Figure 2: RESA Operative Interface. Mobile Web interface with ability to automatically broadcast location and listen to voice messages.

from the same user. However, group authentication is not done by the service itself but can be done by the clients by using customized messages. Currently only two message primitives are used:

- Move. Signals new coordinates for a user
- Talk. Broadcast of a message from coordinator to client.

4. TECHNOLOGY

The overarching design goal is to provide lightweight ubiquitous access to both the coordinator and the operative clients using state-of-the-art Web technologies. The back-end server is a state-less asynchronous real-time messaging bus that may be federated on many servers. The only restriction is that all messages within a group need to be channeled through the same bus. The main technologies we base our solution on are HTML 5 features such as geolocation API [9], native browser audio (a.k.a. the audio tag), and WebSockets [10]. Given that browsers in general and mobile browsers in particular have very limited support for these relatively new features we also provide fallbacks to older technologies such as flash to support our goal of ubiquitous access. The system architecture can be seen in Figure 3.

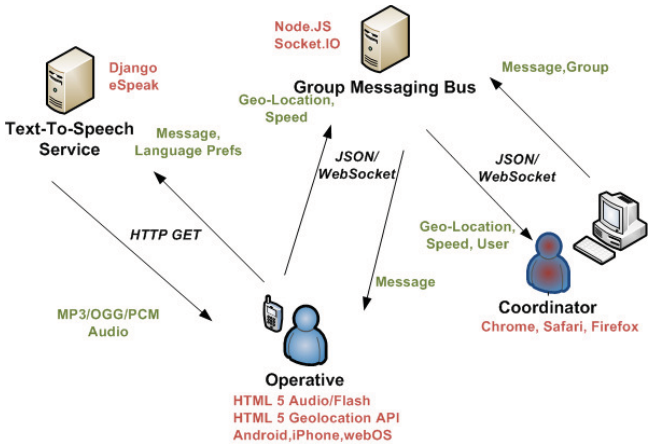

Figure 3: RESA System Architecture. Back-end services are stateless and may thus easily be replicated or loadbalanced to handle more concurrent clients.

Apart from state-of-the art browser implementations of HTML 5, our solution currently also relies on the eSpeak toolkit [11] for text-to-speech, the Socket.IO [12] WebSocket server and client libraries for cross browser WebSocket support, and Node.JS [13] for asynchronous and real-time messaging. We have tested our solution on mobile client platforms such as Android, iPhone and webOS, and PC platforms with Chrome, Safari and Firefox browsers. Both the TTS and the Messaging bus servers are stateless and can easily be replicated in various public cloud deployments.

The WebSocket protocol is fundamentally text-based (UTF-8), but in our case the text always represents valid JSON strings. Broadcasts are made on two channels per group. One channel is used for the coordinator(s) and the other for the operative(s). That way, clients can easily subscribe and unsubscribe to both of these channels depending on the amount of messages they can consume. This design ensures higher scalability in the case of many operatives sending their locations in real-time. It also makes the smartphones connected to slow mobile networks more responsive. The Move messages are sent on the coordinator channel whereas the Talk messages are sent on the operative channel. A Node.JS library was written to extend the Socket.IO WebSocket communication to send real-time messages to groups and to individual users within groups. The clients join groups whenever they send a message designated for a group, and they leave a group when they disconnect from the WebSocket. That way the group membership (being the only state maintained in the server) does not have to be persisted. If the Node.js server goes down it will recreate the group membership data lazily as new messages come in. The group library also caches the last sent message to piggy-back it on disconnect operations so the coordinator(s) know(s) which user has disconnected.

5. USE CASES

Here we discuss some concrete use cases for which the RESA system can and has been used.

5.1 Real-Time Meeting Point Coordination

A number of people are meeting at a remote location for a social lunch, with most attendees being unfamiliar with this location. A coordinator can monitor where everyone is, help people who seem lost or in search for parking, and verify that everyone has arrived.

5.2 IM-to-Voice Real-time Communication

You are in a meeting or in a noisy environment so you cannot make phone calls but you need to communicate in real-time with your friend who is in his car driving. Your friend cannot IM you because of safety concerns while driving. You can then IM your friend while he can get voice SMS delivered in real-time from you.

5.3 Social Sports Tracking

A number of friends decide to bike (or run) in a long distance road race. Some will staff water stations and cheer the bikers on from the sidelines. The spectators can then monitor their friends' locations in real-time to pay extra attention when they pass by, and cheer them on with personalized or group based messages when they are not nearby. Individual bikers can also get updates on where their friends are in the race.

5.4 Real-time Translated Announcements

You are organizing a multi-cultural gathering and you want to communicate some announcements in real time to all event participants who may be spread out across a theme park. Depending on where they are in the park you can send

[9] http://dev.w3.org/geo/api/spec-source-v2.html
[10] http://dev.w3.org/html5/websockets/
[11] http://espeak.sourceforge.net
[12] http://socket.io
[13] http://nodejs.org

them different messages and you can also send personal messages if someone e.g. is approaching a dangerous location. You communicate your messages in English but participants can receive real-time voice messages in their own language.

6. LESSONS LEARNED

We have tested the RESA application in various biking, running and driving scenarios. The first issue we faced was that mobile Web browsers stop running JavaScript when the screen goes into standby. Some phones allow you to extend this time but only up to a very limited maximum time, ranging from 3 minutes (webOS) to 30min (Android). To solve this issue we built lightweight native apps for these platforms that simply load the operative web page on a regular basis. For Android we also use the local TTS engine to save on bandwidth and response time and get a higher quality voice.

The second issue we encountered was related to coverage being spotty and connections dropping intermittently. If location updates are lost for a short period of time it does not affect the user experience very much but if voice messages are lost or location updates are unavailable for longer periods of time the communicating parties may get confused. We therefore added a simple history function that allows all communicating parties who are disconnected for some period of time to request a replay of the last few messages. Note, the replay does not need any user input but can be triggered whenever a new connection is established. Initially we did not report on system status to the operatives so they could move around for a long time without realizing that their connection was lost or that location updates were never sent. We therefore added some subtle sound cues to represent connection establishment, connection dropping, as well as location information being sent.

Finally, the most challenging issue to solve was how to make mobile browsers dynamically load and play audio reliably. We currently adopt five different solutions:

1. HTML 5 Audio tag. The audio tag is inserted dynamically with JavaScript. This works well on Android 2.3+ platforms.
2. jPlayer [14]. jPlayer is a jQuery audio compatibility library. Like with the HTML 5 audio tag we insert the jPlayer tag dynamically in JavaScript. This library works well on iOS 4.1+ devices.
3. Flash. We implemented our own lightweight Flash 9 compatible audio library in ActionScript. This library works well on webOS 2+, and 3+ devices.
4. Android. In our Android wrapper we take advantage of the JavaScript-to-Native bridge to play audio using the native Android media player.
5. Popup. If everything else fails we allow audio to be played in a popup browser window. This works on almost all old and new mobile and desktop browsers, but the user experience is not great. However, given that this interface is supposed to be hands-free and eyes-free it could be a reasonable trade-off on older devices.

[14]http://www.jplayer.org

7. EVALUATION

To evaluate the infrastructure and the real-time experience on actual devices, we ran some location throughput experiments using three mobile devices as well as browser clients. We inject artificial location updates at regular intervals into the same group and coordinator client. The coordinator client for the group was a standard Chrome 7 browser on a 64bit Linux (Ubuntu Lucid Lynx) host inside a corporate firewall (using a http proxy to communicate with our server). As background load, we used Firefox and Chrome browser clients running on 64bit Linux and Windows XP. So the experiment group had updates coming in from a total of 7 clients. The mobile clients used were LG Optimus V running Android 2.2, iPod Touch first generation running iOS 3.1, and HP Pre 3 running webOS 2.2. The mobile clients ran on AT&T's 3G network (iPod and Pre), or Virgin mobile's 3G network (Android).

In the first experiment the mobile clients were configure to send location updates every two seconds, in the second experiment they were configured to send updates every ten seconds. The background updates were configured to generate roughly the same load in the two experiments, 4 updates/s and 4.6 updates/s respectively. The results of the experiments are summarized in Table 1. Each experiment ran for 10 minutes, and a minute-by-minute throughput graph can be seen in Figure 4. The mobile clients drop about 5-7 messages per minute (17-23%) in the two-second-interval experiment and 0-1 message per minute (0-17%) in the ten-second-interval experiment. We note that the HP Pre seems to drop fewest messages, although the differences are marginal. For the ten-second experiment the Pre even pushes too many messages. This behavior is probably due to a drop being overcompensated for as seen in Figure 4 between four and five minutes into the experiment. In conclusion we were able to get good throughput to provide a real-time experience for seven clients under heavy load running on the major mobile phone operating systems. All our clients in the real system are currently configured to not submit more than one location update every two seconds, like in the first experiment.

The maximum load we were able to inject in our experiment was about 7.8 updates per second. This would correspond to a group of 15-16 people submitting updates every 2 seconds, a group of 78 people submitting updates every 10 seconds or a group of 234 people submitting updates every 30 seconds. This shows that our infrastructure can handle reasonably large groups of people broadcasting location updates in real-time.

8. CONCLUDING REMARKS

One major technical challenge that we have yet to address is how to run this service in the background on client devices without native wrapper apps. HTML 5 does have some work in this direction but nothing has been standardized yet apart from Web Workers [15] which allow background threads if you have active foreground threads, but suffers from the same issue as standard JavaScript code if the browser window is not

[15]http://dev.w3.org/html5/workers/

Table 1: Experiment Summary for 2s and 10s Update Intervals

Experiment	Device	Updates/minute
2s	Android	22.5
	iPod	23.7
	Pre	24.8
	Background	167.3
	Total	238.3
10s	Android	5.3
	iPod	5.2
	Pre	6.9
	Background	255.9
	Total	273.3

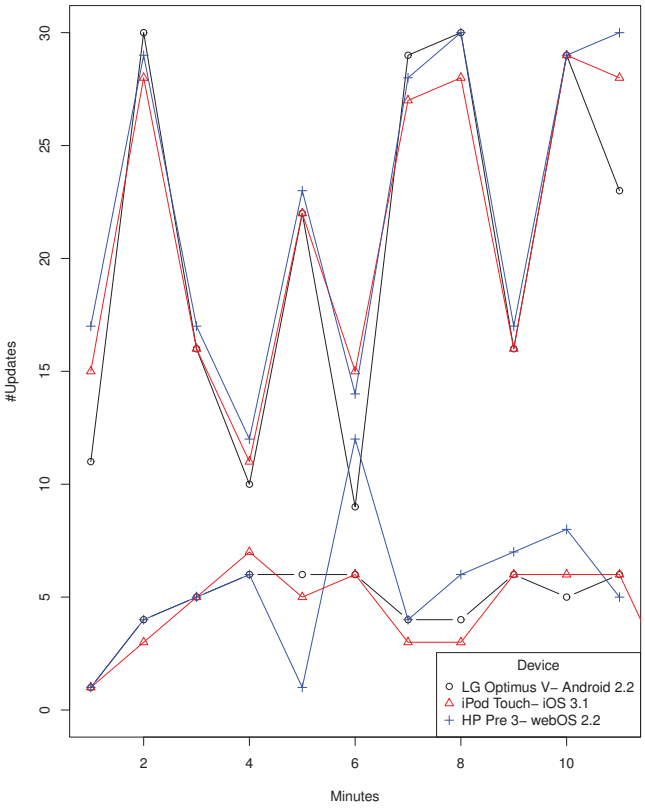

Figure 4: Location Update Throughput for 2s (upper curves) and 10s (lower curves) update interval configuration.

active. As future work we are considering adding an audio sensor to the client application to be able to communicate some information back to the coordinator via ad hoc voice or other sound recordings.

From a use-case perspective we are considering testing and evaluating this solution more in the field and offering it as a public cloud service. Our investigation has shown that the modern mobile platforms have powerful enough technology to both communicate and interact with audio directly from the browser. This opens up a whole new slew of interesting use cases and opportunities to leverage the explosion in smart phone deployments and improvements in ubiquitous connectivity. However, HTML 5 is not ready for this revolution yet so some functionality still needs to fall back on Flash (which now is also becoming available on many mobile platforms e.g. Android 2.2+ and webOS 2.0+) or other workarounds. We have shown in experiments that even with these fallbacks and with the common obstacle of a corporate firewall it is feasible to communicate real-time locations within large groups of people, 16-234 depending on update frequency, just using Web technologies.

Acknowledgments

We thank the members of the Social Computing Group at HP Labs in Palo Alto for helping us test the system, in particular Alex Vorbau and Anupriya Ankolekar for providing insightful ideas and comments on this work.

9. REFERENCES

[1] B. Brown, A. Taylor, S. Izadi, A. Sellen, J. Kaye, and R. Eardley. Locating family values: A field trial of the whereabouts clock. In J. Krumm, G. Abowd, A. Seneviratne, and T. Strang, editors, *UbiComp 2007: Ubiquitous Computing*, volume 4717 of *Lecture Notes in Computer Science*, pages 354–371. Springer Berlin / Heidelberg, 2007.

[2] N. Marmasse, C. Schmandt, and D. Spectre. Watchme: Communication and awareness between members of a closely-knit group. In N. Davies, E. Mynatt, and I. Siio, editors, *UbiComp 2004: Ubiquitous Computing*, volume 3205 of *Lecture Notes in Computer Science*, pages 214–231. Springer Berlin / Heidelberg, 2004.

[3] N. Sawhney and C. Schmandt. Nomadic radio: speech and audio interaction for contextual messaging in nomadic environments. *ACM Trans. Comput.-Hum. Interact.*, 7:353–383, September 2000.

[4] C. Schmandt, N. Marmasse, S. Marti, N. Sawhney, and S. Wheeler. Everywhere messaging. *IBM Systems Journal*, 39(3.4):660 –677, 2000.

[5] K. P. Tang, J. Lin, J. I. Hong, D. P. Siewiorek, and N. Sadeh. Rethinking location sharing: exploring the implications of social-driven vs. purpose-driven location sharing. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 85–94, New York, NY, USA, 2010. ACM.

[6] E. Toch, J. Cranshaw, P. H. Drielsma, J. Y. Tsai, P. G. Kelley, J. Springfield, L. Cranor, J. Hong, and N. Sadeh. Empirical models of privacy in location sharing. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, Ubicomp '10, pages 129–138, New York, NY, USA, 2010. ACM.