Presence-Exchanges: Toward Sustainable Presence-Sharing

Landon P. Cox, Angela Dalton, and Varun Marupadi
Duke University, Durham, NC
{lpcox, angela, varun}@cs.duke.edu

Abstract

Presence-sharing is a promising platform for cooperative location-aware applications, but applications must provide direct benefit to users for their presence information. If not, these services give users strong incentives to free-load to avoid privacy risks and administrative burden. Modeling presence-sharing as an Iterated Prisoner's Dilemma shows that it is not sustainable under these conditions. Thus, to create sustainable presence-sharing, we propose introducing a trusted broker *to transform the game from a Prisoner's Dilemma to a transactional exchange.*

1 Introduction

With the increased availability of BlueTooth-enabled mobile phones and PDAs, *presence-sharing* among independent, mobile users is emerging as a new location-aware service on which to build mobile applications [14]. Example uses of presence-sharing include tagging data such as digital images in support of identity-based file search, mobile social networks [7], and mobile messaging services similar to the "missed connections" feature on `craigslist.com` [12]. While presence-sharing is a promising platform because of its decentralized approach and ease of deployment, users of applications that depend on it must often confront the trade-off between exposing themselves to additional privacy risks and burdening themselves with the administrative overhead of managing that risk.

Mobile users are unlikely to feel comfortable participating in systems where strangers can walk by and know their identity. A simple approach to privacy is to divide the network into those who are trusted to know a user's presence and those who are not, but this is insufficient. There are situations where users do not want to risk leaving a trail of digital footprints, even (and sometimes especially!) for their otherwise trusted friends. Thus, users must craft policies that specify not only *who* may see their presence, but also *where* and *when* they may. These rules are a function of social relations that are difficult to predict and change over time [2]. Furthermore, the privacy risks in these networks are greater than in social networking web sites such as MySpace because of the added dimension of location. Managing risk in such an environment requires continuous user attention and effort.

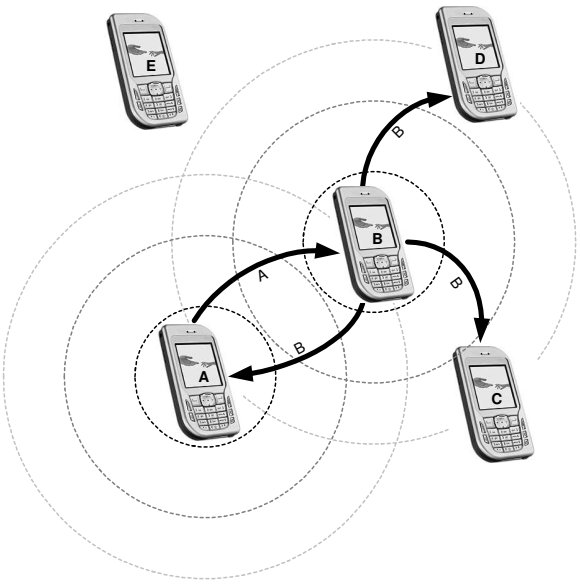
Often compounding the trade-off between privacy and administrative burden is the absence of proportional compensation for these costs. There are applications in which users can expect some direct benefit from sharing their presence, such as in smart environments [4]. For others, such as using presence to tag digital images, a user's presence directly benefits others but not the user herself.

This creates a strong incentive for users to *free-load* by concealing their presence to avoid the privacy-administration trade-off while continuing to benefit from the service. Presence-sharing is not sustainable under these conditions, because the shared resource is not fungible. Access to the most altruistic users' data cannot compensate for those who withhold. Because of this, we believe that application and system designers must take an *incentives-compatible* approach to presence-sharing. A participant's incentives to make her presence known to others must be aligned with the greater goals of the application or system.

The rest of this paper is organized as follows: in Section 2 demonstrate the usefulness of presence-sharing by describing three example applications; in Section 3, we model presence-sharing as an Iterated Prisoner's Dilemma [1] and argue that constraining sharing through brokered *presence-exchanges* is crucial for creating sustainable networks; finally, in Section 4 we make some concluding remarks.

2 Presence-Sharing

Presence-sharing is a subset of location-awareness in which collocated users transmit short-range broadcasts that identify themselves to others. Figure 1 shows such a network. This section describes three example applications



This figure shows a presence-sharing network in which nodes A and B are sharing their presence, but nodes C, D, and E are not. Nodes rely on the short-range radio broadcasts of others to know who is nearby.

Figure 1. Example Presence-Sharing Network

that rely on presence-sharing: file tagging, mobile social networking, and missed connection messaging. Our purpose is not to explore the specific incentives issues of each application, but rather to demonstrate the range of useful services enabled by presence-sharing.

2.1 File Tagging

We began thinking about presence-sharing because of an interest in identity-based file search. Both commercial [9, 15] and research [16] operating systems have begun to embrace search as a first-class data organization tool. These systems typically rely on file names, timestamps, embedded keywords, and computational context [16] to build their search indexes. Unfortunately, there are occasions when conventional sources of meta-data are inadequate or unavailable.

One such occasion is when a user wants to retrieve media files based on the identity of their subject, such as "find the pictures of Bob." Support for this kind of search is both important and hard. The portion of PC users' data composed of personal multi-media files such as digital photographs and video is growing exponentially [13] and at least one study of camera phone use found that images of people comprised over half of all images taken [11]. Furthermore, media files are notoriously difficult to index since they do not contain text and are assigned opaque names by the devices that create them.

Presence-sharing can automatically assign searchable, identity-based attributes to images by associating the presence information available when a picture is taken. MMM2 [3] comes closest to this idea by using presence to suggest users with whom to share an image.

A generalization of the media meta-data application is using presence to organize files on mobile devices. Organizing files on devices such as media players, PDAs, and mobile phones is cumbersome due to fundamental size and form-factor constraints that limit device displays and inputs. Search can ameliorate this problem by allowing users to create data without worrying about how to organize it, relying instead on search engines for retrieval. Ideally, a user could get out a device, do work with it, and then put away the device, with all input data automatically organized by the system. The email service Gmail [8] is an example of a similar approach in another context.

Presence-sharing can be a source highly relevant meta-data, particularly in mobile environments where data's semantics are closely related to the physical context in which it is created. For example, a workshop attendee might take notes during a talk on her PDA. Using presence-informed file meta-data, she could write those notes without worrying about naming a new file or directory and later retrieve her notes by searching for the name of the speaker or the names of other attendees.

2.2 Mobile Social Networking

Presence-sharing can also be used to create mobile social networks. Traditional social network websites, such as `friendster.com` or `myspace.com`, allow users to

create personalized profiles that are linked to their friends'. This allows users to find people with common interests by browsing the profile graph. A mobile social network network provides similar opportunities to meet new people only in physical space rather than over the Internet.

Social Serendipity [7] typifies such a network. Each Serendipity user fills out a profile containing a small photograph, interests, username, and list of friends. Serendipity also associates each profile with a BlueTooth MAC address and mobile phone number. In social situations, a Serendipity server called BlueDar listens for nearby BlueTooth devices and uses the devices' MAC addresses to look up their associated profiles. If there are collocated users with overlapping interests, Serendipity sends a text message containing the profile of each user's match along with a suggestion that they meet. Based on the profile's photo, a user may then look for their match in the room and introduce themselves.

2.3 Missed Connection Messaging

Our final example application is missed connection messaging. The term "missed connection" is derived from a feature of the popular website, `craigslist.com`. This service allows users to post messages for people they encountered in the recent past but were unable to speak to at the time. Cities such as Boston, New York and San Francisco generate hundreds of missed connections on `craigslist.com` each day. Most postings are romantic inquiries, but there are also requests for lost items, such as "did anyone find the laptop I left in my taxi around 2PM," and notifications of found items, such as "I found a set of keys at the coffeeshop."

While popular, this service is by no means ideal. First, users can never be sure if their message has been seen by the intended recipient. Second, authenticating respondents can be difficult. It is not uncommon to see messages in which a respondent is required to provide some detail of the encounter, such as "You were my waitress, please tell me what I ordered." Presence-sharing along with a trusted service mapping devices to profiles could improve both problems.

Users could record the identities of the devices they came into contact with. Then based on the presence information recorded during a social setting, they could ask the service to route messages to the owners of each device. To further ensure that the message reached the correct target, the sender might be allowed to browse the profiles corresponding to the devices she saw or specify that the message only be forwarded to users with specific attributes (male or female, older or younger, etc.). This design could also support messaging users with indirect links. For example, a taxi driver's presence could be used to connect the owner of a lost laptop to the passengers that followed him.

3 The Prisoner's Dilemma

To analyze the incentives faced by presence-sharers, we assume that mobile users are *rational*, but not malicious. Users try to minimize the cost of using a service while maximizing the benefit they receive from it. The primary cost of sharing one's presence is the risk of that information falling into the wrong hands. Importantly, users need not be malicious to be "the wrong hands" since the sensitivity of location privacy depends on time and place. For example, a teenager may not want his friends to know that he and another classmate met for coffee on a Saturday night.

We can model presence-sharing as a Prisoner's Dilemma game. The most common variation of the Prisoner's Dilemma is one in which two agents, A and B, play a game with two actions: *cooperation* and *defection*. Depending on what they do, agents can receive four possible payments at the conclusion of the game: t for "temptation," r for "reward," p for "punishment," and s for "suckered", where $t > r > p > s$.

To play the game, both agents submit their action to a referee without seeing the other's. The referee then hands out payments depending on the actions chosen. If A and B both choose to cooperate, both receive r; if both defect, both get p. If one defects and the other cooperates, the defector is paid the greatest amount, t, and the cooperative agent is paid the least amount, s. The payoff matrix for this game is in Table 1.

What should an agent's strategy be? If B cooperates, it is best for A to defect since a payment of t is better than a payment of r. If B defects, it is also better for A to defect since a payment of p is more than s. Therein lies the dilemma. A and B can both achieve preferable outcome r through cooperation, but their mutual distrust leaves them both with p.

3.1 Incentives in Presence-Sharing

In a presence-sharing network, agents cooperate by revealing their presence and defect by concealing it. $b^{\beta}_{\alpha,\tau,\pi}$ denotes the benefit to agent β of knowing that some other agent α is in place π at time τ. $c^{\beta}_{\alpha,\tau,\pi}$ denotes the cost to β of α knowing that β is in place π at time τ. Finally, $p_{\alpha,\tau,\pi}$ denotes the probability that α is in place π at time τ. Thus, $p_{\alpha,\tau,\pi} * c^{\beta}_{\alpha,\tau,\pi}$ is the expected cost, or privacy risk, relative to α incurred by β when broadcasting her presence in π at τ. Similarly, $p_{\alpha,\tau,\pi} * b^{\beta}_{\alpha,\tau,\pi}$ is the expected benefit to β in π at τ if α cooperates.

There are scenarios in which $c^{\beta}_{\alpha,\tau,\pi}$ may be less than zero. This happens when β benefits from α's knowing that β is in place π at time τ. However, in most cases $c^{\beta}_{\alpha,\tau,\pi} \geq 0$ and we will assume this for the remainder of our analysis. This assumption is reasonable for the applications discussed

	B C	B D
A C	A: r, B: r	A: s, B: t
A D	A: t, B: s	A: p, B: p

This table shows the payoffs for various combinations of cooperation (C) and defection (D) by agents A and B. Payoffs are denoted by r for reward, p for punishment, s for suckered, and t for temptation, where $\mathrm{t} > \mathrm{r} > \mathrm{p} > \mathrm{s}$.

Table 1. Payoff Matrix for the Prisoner's Dilemma

in Section 2: in file tagging, users only benefit from the presence information of others; in mobile social networks and missed connection messaging, users benefit from contact with what is likely to be a small fraction of the entire network.

Our definitions lead to the following payoffs for agent A in place π at time τ for a two-node presence-sharing network with agent B:

$$\begin{aligned} \mathrm{t}_A &= p_{B,\tau,\pi} * b^A_{B,\tau,\pi} \\ \mathrm{r}_A &= (p_{B,\tau,\pi} * b^A_{B,\tau,\pi}) - (p_{B,\tau,\pi} * c^A_{B,\tau,\pi}) \\ \mathrm{p}_A &= 0 \\ \mathrm{s}_A &= -(p_{B,\tau,\pi} * c^A_{B,\tau,\pi}) \end{aligned}$$

If A joins a larger network with members in N, then whenever A's time-space coordinate changes, A plays $|N| - 1$ simultaneous games with each member of N. Thus, for each time increment, A's payoffs are

$$\begin{aligned} \mathrm{t}_A &= \sum_{\alpha \in N} p_{\alpha,\tau,\pi} * b^A_{\alpha,\tau,\pi} \\ \mathrm{r}_A &= \sum_{\alpha \in N} (p_{\alpha,\tau,\pi} * b^A_{\alpha,\tau,\pi}) - (p_{\alpha,\tau,\pi} * c^A_{\alpha,\tau,\pi}) \\ \mathrm{p}_A &= 0 \\ \mathrm{s}_A &= \sum_{\alpha \in N} -(p_{\alpha,\tau,\pi} * c^A_{\alpha,\tau,\pi}) \end{aligned}$$

Reapplying the logic from the original Prisoner's Dilemma to these payoffs yields the same result: A will always defect.

We should explain why administrative costs are absent from our model. The cost of selective access control is difficult to quantify, but a policy based solely on user identities—where cooperation means broadcasting a token that only trusted users can interpret—would have the effect of shrinking N. This might reduce the incentive to defect since a smaller N increases the value of s_A.

However, only perfect access control can make $\mathrm{s}_A = 0$ and $\mathrm{t}_A = \mathrm{r}_A$ and enable cooperation in a single round of the Prisoner's Dilemma. Furthermore, even if perfect access control were possible, its cost would almost certainly be non-zero, while the cost of always defecting would remain zero. Because of this, including administrative costs in these payoffs would complicate, but not alter, our result.

3.2 Enabling Cooperation

Although cooperation in a single round of the Prisoner's Dilemma is impossible, Axelrod demonstrated through his computer tournaments [1] that cooperation can emerge if the game is played over multiple rounds. This is called an Iterated Prisoner's Dilemma (IPD). In an IPD, agents play a round of the Prisoner's Dilemma and remember how their opponent behaved. They then use this history in future encounters with that agent.

In Axelrod's tournaments, agents using a simple "tit-for-tat" (TFT) strategy consistently scored the highest. Under TFT, an agent cooperates the first time it encounters another agent, and thereafter simply mimics what that agent did in their previous interaction. For cooperation to emerge in an IPD, users must be able to retain a history of their previous encounters. Otherwise, the game is reduced to a series of independent, single round Prisoner's Dilemmas in which cooperation is impossible. Thus, for cooperation to emerge in presence-sharing, we must ensure three conditions.

First, there must be repeat interactions between users. If users see each other at most once, there will be no way to punish defectors. Luckily, mobile users are social creatures and are likely to move within a relatively small social network [6].

Second, users must know if their partner has defected. Because users are independent, we cannot eliminate defection, but users must be aware of when it happens. If users "sense" another agent, possibly by overhearing other radio transmissions, but receive nothing in exchange for providing their presence, then it is clear that they have been cheated. However, more subtle forms of defection are also possible. For example, a user could defect by broadcasting a false identity. In doing so, the defector would eliminate any administrative costs and privacy risks, while obtaining the full benefit of the service.

Finally, users must know *who* defected. Strong identities help by disabling the Sybil attack [5]. Otherwise, TFT will alway cooperate and defectors will never be punished. We can provide strong identities through a public key infrastructure [10] or mobile phone numbers.

Crucially, strong identities alone are not sufficient for identifying defectors. This is due to the uniqueness of presence as a shared resource. Agent A cannot know that agent B has defected, because if she did, B's presence would necessarily have been revealed. Knowing which agent has defected—as opposed to simply being out of broadcast range—is equivalent to knowing their presence. This is a contradiction. Thus, to enable cooperation and achieve

	B C	B D
A C	A: r, B: r	A: p, B: p
A D	A: p, B: p	A: p, B: p

This table shows the payoffs for various combinations of cooperation (C) and defection (D) by agents A and B. Payoffs are denoted by r for reward, p for punishment, s for suckered, and t for temptation, where $t > r > p > s$.

Table 2. Payoff Matrix for a Transactional Exchange

sustainability, presence-sharing must be *transformed* from a Prisoner's Dilemma to a more constrained game. Luckily, regulating presence-sharing via *brokered exchanges* provides such a transformation.

3.3 Presence-Exchanges

Under a presence-exchange scheme, users move through the world periodically broadcasting time-, place-, and user-specific *opaque identifiers* (OIDs). To preserve users' privacy, OIDs must not directly identify their broadcaster. They should only be useful once *resolved* to a strong identity. Furthermore, resolving a single OID should not allow a user to resolve past or future OIDs from the same broadcaster. Of course, OIDs must not be completely anonymous either.

Because of this, resolutions pass through a trusted *broker* rather than directly between users. The broker is a single entity known to all users and reachable over the Internet at a well known name, such as `broker.cs.duke.edu`. Users assume that the broker is well-behaved and does not send incorrect or malicious messages. When a user wants to resolve an OID, it registers its interest with the broker. The broker then uses these interests to identify and propose potential trades between users.

Once the broker has proposed an exchange, users must weigh their individual costs and benefits. In this new game, users still cannot identify defectors, but the broker can protect cooperative users in the face of defection. If the broker proposes an exchange and one user defects and one cooperates, the broker will not forward the cooperative user's identity to the defector. This transforms presence-sharing from a Prisoner's Dilemma to a transactional exchange. The payoffs for this new game are in Table 2. Note that users can now cooperate without worrying about being suckered.

There are two primary considerations for designing the broker. First, at minimum the broker must map OIDs to their broadcaster. Otherwise, it cannot coordinate exchanges. Mapping OIDs to users is also crucial for preventing defection by false identity. Second, while the broker must know who wants to exchange resolutions, it should never know any location information related to the exchange. This is necessary to prevent the broker from compiling long-term user histories and to remove what would otherwise be an attractive target for hackers and nosy administrators.

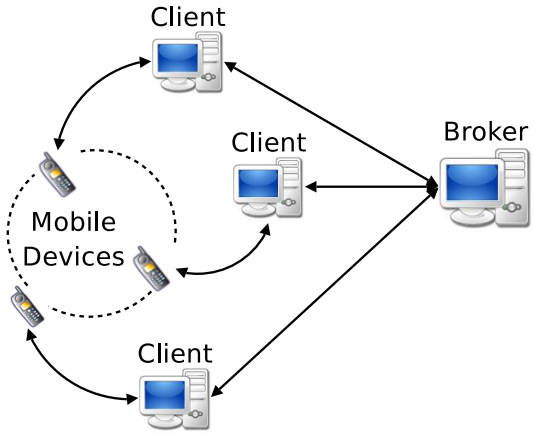


Figure 2. Presence-exchange architecture

There are also interesting design considerations on the users' side. Registering interest with a broker may happen at any time of a user's choosing, but what qualifies as an acceptable resolution latency is application dependent. To be flexible enough to serve the needs of a variety of applications, we can insert an intermediate process called a *client* between the device and broker. Figure 2 shows one possible architecture.

The client is the only entity that communicates directly with the broker; it relays interest in OIDs to the broker and is responsible for accepting or rejecting trades. The client runs on behalf of a particular user, and may run on the user's mobile device, where it can accept OIDs continuously, or on the user's home PC, where it will periodically accept batches of OIDs. The requirements of the application determine this placement.

For example, in a mobile social network, resolution must occur within several minutes. This means that the client must run collocated with the OID logging device, possibly on the device itself. Because communication with the broker over wireless data services like EDGE or GPRS can be expensive, this may not be the best arrangement for applications that can tolerate greater resolution latencies.

In a missed connection messaging system, acceptable message latencies are on the order of hours or days. In this case, the client can probably run on a home PC where communication with the broker is less expensive. Similarly for file meta-data, the only latency requirement is that files be tagged before the user first searches for them. This may be days, weeks, or months after an OID is first logged.

4 Conclusion

Presence-sharing is an emerging platform for location-aware applications such as file tagging, mobile social networks, and missed connection messaging. However, applications must provide direct benefit to users for their presence information. If these incentives do not exist, services are vulnerable to free-loading and risk collapse. Modeling presence-sharing as an Iterated Prisoner's Dilemma shows that privacy risks provide users with a strong incentive to free-load. To realign individuals' incentives with the goals of application designers, we propose using a trusted broker to transform presence-sharing from a Prisoner's Dilemma to a transactional exchange.

5 Acknowledgments

We thank our shepherd, Adrian Friday, and the anonymous reviewers for their valuable feedback.

References

[1] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[2] S. Consolvo, Ian Smith, T. Matthews, A. Lamarca, J. Tabert, and P. Powledge. Location disclosure to social relations: Why, when, and what people want to share. In *CHI '05*, Portland, OR, April 2005.

[3] M. Davis, N. Van House, J. Towle, S. King, S. Ahern, C. Burgener, D. Perkel, M. Finn, V. Viswanathan, and M. Rothenberg. MMM2: Mobile media metadata for media sharing. In *Extended Abstracts of SIGCHI*, Portland, OR, April 2005.

[4] A.K. Dey, D. Salber, and G.D. Abowd. A context-based infrastructure for smart environments. In *Proceedings of the First International Workshop on Managing Interactions in Smart Environments*, Dublin, Ireland, December 1999.

[5] J. R. Douceur. The Sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems*, pages 251–260, Cambridge, MA, March 2002.

[6] N. Eagle and A. Pentland. Reality Mining: Sensing complex social systems. *Journal of Personal and Ubiquitous Computing*, 2005.

[7] N. Eagle and A. Pentland. Social serendipity: Mobilizing social software. *IEEE Pervasive Computing*, pages 28–34, April–June 2005.

[8] Gmail. http://gmail.google.com.

[9] Google Desktop. http://desktop.google.com.

[10] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 public key infrastructure certificate and CRL profile. Internet RFC 2459, 1999 January.

[11] T. Kindberg, M. Spasojevic, R. Fleck, and A. Sellen. The ubiquitous camera: An in-depth study of camera phone use. *IEEE Pervasive Computing*, 4(2):42–50, 2005.

[12] Craig's list: missed connections. http://www.craigslist.org/mis/.

[13] R.J.T. Morris and B.J. Truskowski. The evolution of storage systems. *IBM Systems Journal*, 42(2):205–217, 2003.

[14] Nokia Sensor. http://www.nokia.com/sensor.

[15] Spotlight for Mac OS X. http://apple.com.

[16] C.A.N. Soules and G. Ganger. Connections: Using context to enhance file search. In *Proceedings of the 20th ACM Symposium on Operating Systems Principles*, Brighton, UK, October 2005.