Putting People in their Place: An Anonymous and Privacy-Sensitive Approach to Collecting Sensed Data in Location-Based Applications

Karen P. Tang, Pedram Keyani, James Fogarty, Jason I. Hong
Human Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{kptang, pkeyani, jfogarty, jasonh}@cs.cmu.edu

ABSTRACT
The emergence of location-based computing promises new and compelling applications, but raises very real privacy risks. Existing approaches to privacy generally treat people as the entity of interest, often using a fidelity tradeoff to manage the costs and benefits of revealing a person's location. However, these approaches cannot be applied in some applications, as a reduction in precision can render location information useless. This is true of a category of applications that use location data collected from multiple people to infer such information as whether there is a traffic jam on a bridge, whether there are seats available in a nearby coffee shop, when the next bus will arrive, or if a particular conference room is currently empty. We present *hitchhiking,* a new approach that treats *locations* as the primary entity of interest. Hitchhiking removes the fidelity tradeoff by preserving the anonymity of reports without reducing the precision of location disclosures. We can therefore support the full functionality of an interesting class of location-based applications without introducing the privacy concerns that would otherwise arise.

Author Keywords
Hitchhiking, privacy, anonymity, location-based computing.

ACM Classification Keywords
H5.2. Information interfaces and presentation: User Interfaces; H1.2. Models and Principles: User/Machine Systems.

INTRODUCTION AND MOTIVATION

A number of technologies are converging to support the widespread deployment of location-based applications on mobile phones, on handheld and laptop computers, and in vehicles. In the case of vehicles, integrated navigation systems are motivating the inclusion of Global Positioning System (GPS) units. For phones and computers, the most promising technology seems to be software that infers a device's location by detecting nearby phone towers or wireless network (WiFi) access points [16, 23]. Because of these advances, we can now build applications that require only the hardware already included in devices that people currently use. Location-based applications can therefore be deployed entirely in software, at a very low cost.

The pending ubiquity of location-based applications has significant implications for anonymity and privacy. Consider an otherwise anonymous person who starts almost every day in a given location and ends the day in that same location. An application that is able to collect this data can identify the person by checking a database to see who lives at that address. There is also a potential for individuals to abuse location-based applications for more malicious purposes, targeting a specific victim and obtaining information about that victim's location and movement.

Significant prior work has examined anonymity and privacy in location-based applications [2, 4, 8, 11, 12, 13, 23, 24]. While we defer a discussion of that work until the next section, prior work generally makes two assumptions. First, *prior work generally treats a person as the entity of interest*. For example, a person might reveal their location as part of a query about their surroundings or as a part of a social interaction with friends. This has the implication that *prior work often treats location privacy as a fidelity tradeoff*. Revealing a more precise indication of one's identity or location often allows these social applications to provide better service. This conception of the problem has led prior work to focus on techniques for balancing the fidelity of disclosure against the utility of an application.

This paper contributes *hitchhiking*, a new approach to anonymous and privacy-sensitive collection of sensed data in location-based applications. *Hitchhiking applications treat locations as the entity of interest.* Because the knowledge of who is in a location is irrelevant, the fidelity tradeoff is removed. Instead, *hitchhiking ensures the anonymity of people providing information about a location*. We can therefore obtain the full functionality of an interesting class of location-based applications without the privacy concerns that would otherwise arise.

CHI 2006, April 22–27, 2006, Montréal, Québec, Canada.

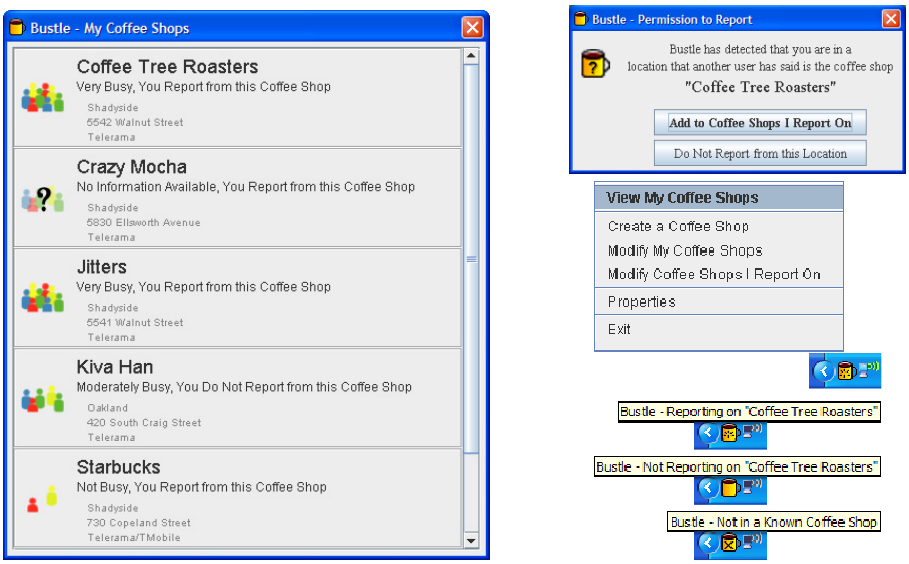

Figure 1. Several screenshots of *Bustle*, a WiFi-based demonstration of the hitchhiking approach. *Bustle* senses laptops on a WiFi network and anonymously reports an estimate of table availability in coffee shops.

Hitchhiking supports applications that combine location information from many people to infer information about locations. Such applications include, but are not limited to, live traffic monitoring, inferring the availability of seats in a nearby coffee shop, estimating the arrival time of a bus, or monitoring the availability of a particular conference room. For these location-centric applications, it is irrelevant who is in a traffic jam or who is riding a bus. For example, Zipdash allows GPS-enabled mobile phone users to check current traffic conditions near their location [26]. But Zipdash requires users agree to continuous fine-grained location disclosure. These continuous location disclosures are used to infer traffic congestion by monitoring the rate at which people are moving, but continuous location disclosure is a significant threat to personal privacy. Hitchhiking provides an anonymous and privacy-sensitive approach to this class of location-centric applications.

These applications hold significant promise, but it is important to make hitchhiking safe. As discussed in the next section, prior approaches to privacy and anonymity are inadequate. Queries about the current traffic conditions at a specific location can be masked using techniques developed in prior work. For example, a person interested in traffic conditions can mask their location with a query that asks "*Tell me the current traffic conditions everywhere in the city*." This query reveals only what city a person is in, and the person's device can locally filter the resulting data to obtain the information that is actually of interest. But this reduction in location precision cannot be applied in an application like Zipdash. Zipdash needs precise location reports to model traffic, as it cannot model traffic congestion using reports of the form "*I am somewhere in the city traveling at 15 miles per hour.*"

The fundamental tenet of hitchhiking is to put people in their place: *reports are always strictly about a location and cannot be tied to a person*. Because a person's anonymity is protected, it is safe to agree to precise location disclosures. The hitchhihking approach is implemented on the client device, limiting the information reported to a server. It can therefore be deployed on existing phone and WiFi networks without the cooperation of a trusted middleware provider and without an intentional reduction in the precision of location reports.

Specifically, this paper makes three contributions. First, it introduces hitchhiking as a novel approach to building an interesting class of useful location-based services in a manner that maintains end-user privacy. Second, it presents a privacy risk analysis of hitchhiking, providing a design rationale for this approach and discussing how the privacy of end-users is protected in multiple ways. The hitchhiking approach and its privacy risk analysis will be useful to anyone who designs and implements location-based applications, as it provides an alternative approach to building a useful class of applications while also protecting end-user privacy. Third, we demonstrate the application of hitchhiking to a set of location-centric services, including estimates of coffee shop space availability, traffic monitoring, bus location tracking, and conference room availability monitoring.

The next section reviews prior work, with a focus on technical approaches to location privacy. We then present *Bustle*, our first implementation of an application based on the hitchhiking approach. *Bustle* uses WiFi to detect laptop computers and infer space availability in coffee shops. We then use *Bustle* as a case study to present the details of hitchhiking. The presentation is organized as a privacy risk analysis, detailing likely threats and presenting strategies for addressing each. We then discuss three additional potential applications: traffic monitoring, bus location tracking, and conference room availability. We finally present a short discussion and conclude.

RELATED WORK

It is important to recognize that location privacy is impacted by social, legal, market, and technical forces [18]. Because our approach is technical, and because space is limited, we focus our discussion of prior work on technical approaches to location privacy. We also focus on methods that are applied before an application obtains a person's location (as opposed to applications that collect complete logs of precise location data and analyze that data using an algorithm that preserves some notion of privacy). Focusing on technical approaches that are applied before data collection provides a specific advantage: information cannot be abused if it has not been collected. Even when social, legal, and market forces are considered, gross violations can still occur. For example, a former America Online employee was recently convicted of stealing and selling 92 million customer email addresses, inflicting an estimated $300,000 of damage [14].

Location-based systems can be based on detecting a variety of radio beacons, such as WiFi access points [1, 3, 7, 10, 16, 23], GSM mobile phone towers [16], and FM radio stations [15]. Different types of beacons have different characteristics, but they are all deployed and maintained by third party providers. Because existing devices can already detect these beacons, applications can be built entirely in software. Their deployment costs are therefore much lower than approaches that require specialized hardware. The Place Lab initiative is pursuing beacon-based location estimates, with an explicit focus on how beacon-based location estimates support privacy [16, 23]. Specifically, location can be computed without connecting to a beacon or otherwise revealing the presence of a device. This is critical to preserving privacy in location-based applications, as local computation gives people control over when to share their location.

Prior work on privacy and beacon-based location estimates has generally focused on two categories of applications. The first category uses a person's location to customize or filter the information delivered to that person, such as location-enhanced web services utilizing the Place Bar [23], the Mobisaic system for location-aware web browsing [25], or location-based reminder systems [6]. The second category treats the person's location as the information of interest, perhaps revealing it to interested members of the person's social network. Significant work has examined when and how people want to release their location to other people [4, 13, 24]. Because the person is the primary entity in these applications, anonymity is difficult or impossible to achieve (anonymity is obviously precluded in applications where a person is sharing their location with their social network). In the case of location-enhanced web browsing, anonymity is often breached when a person is required to login to a website. As discussed in our introduction, our approach treats locations as the primary entity of interest and can therefore preserve end-user anonymity.

Gruteser and Grunwald describe spatial and temporal cloaking to preserve *k*-anonymity [8]. In their approach, people report their location to a trusted middleware server. When an application needs a person's location, it obtains it from this server, which uses its knowledge of the locations of many people to compute an obfuscated result that describes both the location of the desired person and at least $k - 1$ other people. While this approach can be applied to many applications, the reduced precision is likely to undermine the category of applications supported by the hitchhiking approach. Hitchhiking also does not require a trusted middleware server.

Beresford and Stajano present the notion of mix zones, areas in which no application is monitoring a person [2]. Any time a person enters a location of interest, the person begins using a new identifier. Once they leave that area of interest (returning to the mix zone), they never again use that identifier. If a new identifier cannot be linked to a previous identifier, a person cannot be tracked. But a client device cannot independently know whether two identifiers can be linked, because it does not know how many other people are in the mix zone or what paths people typically take through the mix zone. A trusted middleware server is therefore used to inform clients of the expected degree of anonymity associated with a new identifier. Hitchhiking does not require a trusted middleware server, as people do not use identifiers. This is possible because we treat locations, and not people, as the entities of interest.

Policy-based approaches, such as systems based on P3P, allow an application to describe how it will store and use provided information [5, 17]. Given this description, people can make an informed decision about whether to provide the information, though Palen and Dourish note that the opacity of modern technology often makes it difficult for people to make good decisions about privacy [20]. It is often impossible to use technical mechanisms to enforce the human-readable policy that is advertised when data is collected, so these approaches are usually based in the social, legal, and market ramifications of violating the stated policy. But this does not mean that technical solutions have no role in policy-based approaches. For example, Hong and Landay present *Confab*, a client-centered architecture in which personal data is sensed, stored, and processed on end-users' devices as much as possible, with better user interfaces for sharing that information [11]. A system built with *Confab* can also audit its data usage, making it easier for applications to ensure that they are following their stated policies.

Finally, Hong *et al.* present privacy risk models as a method for refining privacy from an abstract concept into concrete issues for a specific application [12]. Our work can be considered an example of a privacy risk model: we have identified the privacy threats encountered in a category of location-based applications and have developed strategies for addressing these threats. An application that uses our approach therefore addresses these privacy threats.

APPLICATION: COFFEE SHOP AVAILABILITY

Figure 1 contains several screenshots of *Bustle*, our first implementation of a hitchhiking application. *Bustle* senses laptops on a WiFi network and anonymously reports an estimate of table availability in coffee shops. *Bustle* is implemented in approximately 3000 lines of Java, using Place Lab for WiFi spotting [16, 23], `jpcap` for network monitoring, and the Java Desktop Integration Components for system tray support. In a typical usage scenario, a person might visit a local coffee shop and begin working on their laptop. Running in a background process, *Bustle* continuously scans for nearby WiFi access points. When it detects an access point in its database, *Bustle* infers that the person is in a coffee shop. It then checks whether this person has previously approved or denied reporting from this coffee shop. If it finds that the person has not set a policy, *Bustle* displays a dialog informing the person that they are in a location that another user has said is a coffee shop, asking whether it is okay to report from this location. After obtaining a one-time approval, *Bustle* monitors the WiFi network to determine how many other computers are present. At regular intervals, it reports this count to a server. The server uses a history of counts at that coffee shop to infer whether the coffee shop is currently busy, sharing this information with interested people.

Bustle's detection of computers on a WiFi network is based on Address Resolution Protocol (ARP) broadcasts. Every computer (regardless of its operating system), sends an ARP broadcast at least once every 10 to 20 minutes (even if the computer is not actively generating network traffic). *Bustle* maintains a list of detected computers, removing a computer if no broadcast is detected for 20 minutes.

We conducted a small feasibility study of sensing coffee shop space availability. It is clear that not everybody uses a laptop in a coffee shop, but it is unclear whether the correlation between laptop usage and the number of people in a coffee shop is sufficient for inferring space availability. We made 20 visits to a laptop-friendly coffee shop in a nearby commercial district. On each visit, we monitored ARP broadcasts for 20 minutes and then counted the number of empty tables. We collected 20 samples over the course of seven days, spacing each pair of samples by at least 90 minutes and aiming for coverage between 9:00 AM and 9:00 PM. The resulting data is shown in Figure 2.

In the coffee shop we sampled, there is a strong correlation between the number of computers on the network and the number of empty tables ($r^2 = .537$, $p < .001$). In every case that no tables were available, eight or more computers were detected on the network. While the strength of this correlation will obviously vary in different coffee shops, this result shows that this approach can be successful in some. The *Bustle* server applies a percentile-based transformation to the reports collected from each coffee shop, automatically learning a threshold for each coffee shop. We use a conservative threshold, so *Bustle* will sometimes report that a coffee shop is crowded when there is still a reasonable chance that a table is available. We are comfortable with this, as we feel the more damaging error is when a person is told that space is available, walks to the coffee shop, and is then unable to find a table.

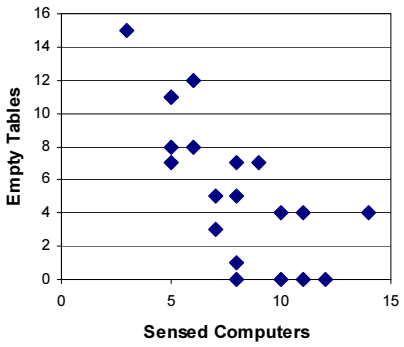

Figure 2. Computers sensed versus empty tables in a local coffee shop ($r^2 = .537$, $p < .001$). In every case that no table was available, 8 or more computers were detected.

ANONYMOUS LOCATION-BASED DATA COLLECTION

Bustle's main contribution was to help us define and refine the hitchhiking approach. To that end, this section presents the details of using the hitchhiking approach to anonymously collect information in *locations of interest*, such as coffee shops, highways, public buses, or conference rooms. It is organized as a privacy risk analysis, introducing a series of threats to anonymity and discussing how to counter each threat. This privacy risk analysis has been developed and iterated upon in parallel with *Bustle*, with specific threats in *Bustle* informing our analysis of hitchhiking and vice versa. We consider a person's anonymity or privacy to have been violated in either of two scenarios:

> **An *identity violation* has occurred if a single report reveals a person's identity**. If a report allows the determination of a person's name, account number, address, or some other identifier, the anonymity of the person providing the report has been compromised.
>
> **A *tracking violation* has occurred if a report can be identified as being provided by the same person who provided an earlier report.** Tracking violations allow the movement of an individual to be tracked over time. This does not necessarily mean their anonymity has been breached, but it is probably a violation of their privacy. Furthermore, a tracking violation can likely be elevated to an identity violation by physically visiting a location frequented by a tracked person.

Figure 3 lists four categories of threats that can result in identity or tracking violations. The hitchhiking approach addresses these threats with the seven requirements in Figure 4. This section focuses on WiFi-based location technology and uses *Bustle* for illustrative purposes, but these threats and their counters also apply to other location technologies and to other location-centric applications.

Location is Computed on the Client

As discussed in related work, local computation of location is important to anonymous location-based applications. If a WiFi-based application is continuously making queries of the form "*I can see access point 00-0C-F1-5C-04-A8, what is my location?*", then it is continuously disclosing the person's location. In Place Lab [16, 23], this is addressed by the local storage of a database mapping WiFi access points to GPS coordinates. An application can therefore infer its location by checking this local database, without sending a query to a server.

In the case of *Bustle* and other hitchhiking applications, the definitions of locations of interest must be stored on the client device. *Bustle* stores a list of coffee shops, together with the WiFi access points that can be detected from each coffee shop. Applications discussed later in this paper store lists of GPS coordinates to define each location of interest. Regardless of the underlying technology, the requirement is that no external communication is required for an application to determine if it is currently in a location about which it could report useful information.

Only the Client Device is Trusted

While it is fairly straightforward to design an application that does not intentionally reveal a person's identity or support tracking, our approach sets the higher standard of assuming that the servers used by an application are completely untrusted. It is therefore necessary to counter active attacks by the server that are intended to induce identity or tracking violations.

By assuming the server is untrusted, we also prevent malicious users from using a server to gain leverage in an attack. For example, a malicious user might target a victim after a face-to-face encounter in a coffee shop. But because *Bustle* does not permit identity or tracking violations, it does not provide the malicious user with any additional information about the intended victim. We cannot prevent the malicious user from sitting in the coffee shop and waiting for the intended victim to return, but even a full disclosure of all data collected by a *Bustle* server would not allow the malicious user to identify the intended victim or determine when the intended victim usually visits the shop.

Trusting only the client device is an important distinction from prior work, as reliance on a trusted server provides a single point of failure. If the server is compromised by a malicious insider or by a security hole, an attacker gains access to location data for everybody who uses a system. Trusting only the client device removes this concern. For the same reason, none of our proposed applications require client storage of a location history, so client device theft does not reveal information about a person's movement.

Each Person Must Approve Reporting from a Location

If a malicious server operator or a malicious user targets an individual, a tracking violation can be induced by defining a location of interest that is likely to only generate reports from that individual. For example, many people who use WiFi-enabled laptops in coffee shops will also have a wireless network in their home. If an attacker obtained the MAC address of the home wireless network of an intended victim, they could create a fake coffee shop with the intended victim's home access point. The attacker could then track when the intended victim is home by noting when reports are generated for the fake coffee shop. As the intended victim is likely to be the only user of this wireless network, they will also be the only person who reports on the fake coffee shop. Therefore, the intended victim is likely home when somebody is reporting on the fake coffee shop, and is likely away when nobody is reporting.

To counter attacks that target a sensitive location, our approach requires that each user approve every location from which they report. In the case of *Bustle*, this is implemented as a dialog that is presented the first time *Bustle* wants to report on a given location. The person can choose to approve the location for future reporting or to permanently deny the release of information about that location, with both choices being reversible.

Physical Constraints Prevent Location Spoofing

Because approval must be obtained in order to report from a location, malicious server operators and malicious users can be expected to attempt to spoof a location, getting an intended victim to approve a location without realizing what they are approving. In the case of *Bustle*, we would expect that the fake coffee shop targeting an intended victim would be given the name and street address of an actual coffee shop that the intended victim regularly visits. Recognizing the name and address of the actual coffee shop, the intended victim might approve reports from the fake coffee shop, enabling a tracking violation.

Our approach uses the physical constraints of real-world location to prevent spoofing. In the case of *Bustle*, a location can be approved only when *Bustle* detects that the

1) Collected location logs can be abused by a server operator or by other people who gain access.
2) A user could be targeted by monitoring their home or another similarly sensitive location.
3) A location approval could be spoofed, tricking a target user into approving a sensitive location.
4) By hiding an identifier in a location definition, a server could track when people visit a location.

Figure 3. Four categories of threats to privacy and anonymity in hitchhiking applications.

1) Location is computed on the client.
2) Only the client device is trusted.
3) Each person must approve reporting from a location.
4) Physical constraints prevent location spoofing.
5) Location identifiers are based in the physical location.
6) Location identifiers are generated by the client.
7) Sensed identifiers are not reported to the server.

Figure 4. The seven requirements of our approach to protecting privacy and anonymity in hitchhiking applications.

person is physically in the approved location. There is no list of coffee shops that a person can browse by name, as a real coffee shop would be indistinguishable from a fake. The approval dialog (see Figure 1) is also carefully worded to make it clear that somebody has claimed that the person's current location is a coffee shop and that *Bustle* is requesting permission to report on their current location. When presented with this dialog while sitting in their living room, it should be clear to the intended victim that they are not currently in a coffee shop.

As we will discuss in the coming sections, this requirement can also be met by using the physical correspondence between GPS coordinates and real-world locations. But an application that uses maps of GPS coordinates for approval must generate that map on the client device, as a map provided by the untrusted server could have been spoofed.

Location Identifiers are Based in the Physical Location

A naïve approach to hitchhiking might assign each location a random or sequential identifier, such as a unique index in the server's database. But this type of arbitrary identifier allows an attack that can induce a tracking violation. Consider a coffee shop with the arbitrary identifier *ID-COFFEE-SHOP*. A malicious server could append a unique suffix every time a user downloads the current list of known coffee shops. So a user *A* would have *ID-COFFEE-SHOP-A* and a user *B* would have *ID-COFFEE-SHOP-B*. Normal operation of the server could be maintained by mapping all reports and queries to the root identifier, so it would appear that everybody was using the same identifier to refer to this coffee shop. But the malicious server operator would know that every report on *ID-COFFEE-SHOP-A* indicates that user *A* is currently in the coffee shop. While we use simple suffixes here, this attack can be masked with randomly generated identifiers.

To address this attack, a location identifier must be based in a physical property of the location. The choice of a physical property will often be based in the location-sensing technology. In *Bustle*, a coffee shop is identified by listing the detected WiFi access points (see Figure 5). The server checks the database of coffee shops to determine what coffee shop is being reported from (using the same matching algorithm used by the client), then updates its records for that coffee shop.

Location Identifiers are Generated by the Client

While the correspondence between physical locations and location identifiers allows clients and servers to exchange information about a location without the use of arbitrary identifiers, a malicious server can still induce a tracking violation by carefully crafting a location identifier. The location identifier must therefore be generated by the client, using information it has sensed about the physical location. Again, the choice of how to implement this requirement will be largely driven by the location technology.

I see access points:	**00:0C:41:66:64:00**
I am connected to a network named:	**Telerama**
I have detected:	**7 computers**

Figure 5. Contents of a *Bustle* report, sent to a server by a person in a coffee shop. Every field is a sensed property of the location, so the report cannot be tied to a person.

In the case of *Bustle*, a malicious server might attempt a tracking violation by inserting fake access points whenever a person downloads the current list of coffee shops. If *Bustle* included the provided access points in later reports, the server would have induced a tracking violation. But, as stated in the last subsection, *Bustle* reports only the access points it has physically detected in the coffee shop. It is able to generate this identifier without assistance from the server, and so the server cannot induce a tracking violation.

The more interesting case arises when using locations defined by a list of GPS coordinates. For example, consider a traffic monitoring application (discussed in detail later in this paper) that reports the speed at which people are traveling on an often congested length of highway. The logical way to define the highway is with a list of GPS coordinates, but a client cannot send this list back to the server when making a report (as the list was provided by the potentially malicious server). The server could hide a unique identifier in the low-order bits of the GPS coordinates or in the structure of the list itself. As we will discuss, we address this attack by reporting the current speed, direction, and GPS coordinate of a vehicle. It is then up to the server to decide on what road the vehicle is traveling. Meeting this requirement is especially difficult for the bus tracking application, leading us to believe that this requirement will generally be the most difficult part of implementing a hitchhiking application.

Sensed Identifiers are Not Reported to the Server

Many applications sense identifiers associated with other people or their devices. For example, the ARP broadcasts that *Bustle* uses to estimate the number of people in a coffee shop contain a unique MAC address associated with the computer that sent the broadcast. These identifiers must not be reported to a server, as this would allow a malicious server to track the people sensed by *Bustle*. This is a very serious invasion of privacy, as every *Bustle* user would effectively be reporting the location of every other person using a computer in the coffee shop.

Bustle addresses this requirement by reporting only a count of the computers detected. This is sufficient for *Bustle*, as every ARP broadcast is seen by every computer on the network. Every computer therefore has an accurate count of how many computers are on the network. In the case that several networks are available in a single coffee shop, we report the name of the network being used, allowing the server to sum reports from different networks.

This requirement does impose limits on certain types of applications. For example, consider if *Bustle* ran on mobile phones and used Bluetooth detection of other phones to estimate the number of people in a coffee shop. Two different phones in the same coffee shop might detect different sets of phones, and we are not aware of any way to give the server enough information to compute the union of these sets without allowing the server to track the detected phones. This an interesting area for future research, but we note that neither a simple cryptographic hash nor a hash that changes over time is adequate (as a malicious server could track any phone with a known MAC address).

Requirement Summary

The seven requirements presented in this section combine to ensure that a malicious server cannot induce identity or tracking violations. Each person approves every location they report from, and the use of physical constraints ensures that a spoof cannot mask what location the person is approving. Because none of the information in a report was initially provided by the server, there is no opportunity for the server to hide an identifier in the report. The server knows the physical properties of each location (such as the GPS coordinates of a highway or the WiFi access points in a coffee shop), so it can infer what location is being reported on. But the server cannot infer who made a report.

The next three sections present some of the most important aspects of hitchhiking in the context of three applications. We provide these applications as a demonstration of the breadth of our approach, to provide more detail on how other approaches can be attacked by a malicious server, and to clarify how our approach counters these attacks.

APPLICATION: TRAFFIC MONITORING

As discussed in our introduction, Zipdash is a service for GPS-enabled mobiles phones that provides live traffic reports, but it requires that users consent to continuous location disclosure [26]. It is very possible for these users to incidentally reveal their home address (the location where most trips start or end) and therefore their identity. This section discusses the use of hitchhiking to build an privacy-preserving application with the same functionality.

We assume that a mobile phone is generating GPS-based location estimates (either via GPS hardware or Place Lab inference [16, 23]). A location of interest, such as a bridge or a length of highway, is defined as a polygon of GPS coordinates. When a person is traveling in an area from which they have approved reporting, their device occasionally sends a current GPS coordinate, a direction of travel, and a travel speed (the last two computed from recent GPS coordinates). The server then infers on what road the person is traveling and uses the speed to update its current model of traffic congestion.

Physical Constraints Prevent Location Spoofing

Because it would obviously be inappropriate to ask people to make disclosure approvals while driving, the physical constraint of "approving your current location" cannot be copied from *Bustle*. Instead, the phone notes each location of interest visited by a person. It then later seeks approval to report on future visits to those locations. A physical constraint is implemented by using a client-generated map during the approval process. The application queries a trusted map source, such as a local database or an online resource like Google Maps. It then maps the location of interest using the list of GPS coordinates that define it. This ensures that the map actually represents the area from which information will be reported (whereas an image provided by a server could be spoofed, showing a harmless map while actually obtaining approval for a sensitive area).

Location Identifiers are Generated by the Client

It is important to note that each report contains only a single GPS coordinate, and that this coordinate was actually sensed by the client. The coordinates defining the highway were provided by the server. If a report said "*I am traveling on the highway defined by this list of GPS coordinates*," it would open the opportunity for a malicious server to induce a tracking violation.

A simple attack hides an identifier in the low-order bits of the coordinates defining the location of interest. But this could be exposed if the points were plotted on a sufficiently high-resolution map (as the coordinates might not exactly align with the highway). A less detectable attack hides the identifier by introducing artificial breaks in the coordinate list. For example, consider if the actual list contains two GPS coordinates that are 100 yards apart. A malicious server could introduce a third coordinate between these two points. This fake coordinate can be placed on the line between the original coordinates, so that it would likely not result in any visible change to a plot of the coordinate list. Placing the fake coordinate in a different location each time a person downloads the list would induce a tracking violation. By reporting only a single GPS coordinate (one that has been sensed by the device), our approach ensures that no such manipulation can induce a tracking violation.

APPLICATION: BUS TRACKING

Various cities are using infrastructure-based approaches to live bus location tracking [19]. These systems typically use GPS or odometer-based dead reckoning, uploading the location of the bus to a transit authority server. While very effective for cities that can afford the instrumentation, the cost can be prohibitive for many other cities.

We propose that bus location can be tracked by the mobile phones of people who are currently riding a bus. When a person's phone decides that they are currently riding a bus, it can anonymously report the current location of that bus to a server that shares the information with interested people. Further, these reports could include a Bluetooth-based estimate of how many other mobile phones are currently on the bus. We will not directly address the problem of inferring whether a person is currently riding a bus, though

we note that the problem is simpler than it might seem because most people ride only a handful of buses. The application's installer could therefore include a form asking what buses the person uses, significantly reducing the number of bus routes that need to be considered. Patterson *et al.* have also shown GPS-based inference of bus use [21].

Most of the hitchhiking requirements for this application can be addressed with the same methods used in *Bustle* and in our traffic monitoring application. For example, a client-generated map can be used to obtain approval for reporting on a bus route. Similarly, the application should report a count of how many phones have been detected on the bus, not the identifier associated with each phone. However, the client generation of the location identifier proves very difficult for this application.

Location Identifiers are Generated by the Client

In both of our previous applications, the location identifier was taken directly from the location sensor (either the visible WiFi access points or the GPS coordinate). But the difficulty with bus tracking is that the bus route is not actually a physical location. It does not broadcast a MAC address and no single GPS coordinate identifies it (there are often several buses that travel along a particular road). Rather, it is a convention, a path typically followed a bus. The anonymity of hitchhiking is built on the physical constraints of location, but a bus route is made up of many locations. Bus tracking therefore stretches the hitchhiking notion of sensing information about a location.

The best solution seems to be the short identifier that transit authorities already associate with each bus route. In our city, for example, we have a *71A*, a *500*, and a *61C*. Riders are familiar with these identifiers, so they are not arbitrary. The client software can also enforce a limit on the length of the identifier (such as 5 characters) to ensure that very little space is available to attempt to hide a malicious identifier. If international characters are accounted for and this identifier is prominently displayed when agreeing to disclose information on a route, it should be obvious if a server has tampered with the identifier. A more descriptive name can also be used the interface, but only the identifier is included in reports. A report would therefore contain the information seen in Figure 6.

APPLICATION: CONFERENCE ROOM AVAILABILITY

While our previous examples have focused on large-scale applications, hitchhiking can also be applied on a smaller scale. For example, consider the problem of finding an available conference room for an impromptu meeting. The typical scenario involves walking from room to room to see if each is actually in use (as people sometimes reserve rooms but do not actually use them). Sensing infrastructure (such as wireless motion detectors) can be installed, but this infrastructure often has few other uses and so it can be difficult to justify the installation and maintenance costs.

I am riding bus:	**71A**
I am at coordinate:	**N 40.44843** **W 79.93399**
I am traveling in the direction:	**East Northeast**
I have detected:	**5 mobile phones**

Figure 6. Contents of a bus tracking report. It reveals the current location of the bus, but cannot be tied to a person.

A hitchhiking approach to this problem can use WiFi-based location estimates in much the same way as *Bustle*. Because there are often offices very close to conference rooms, care needs to be taken to ensure that people working in their office are not reporting that they are using the conference room. This might be as simple as also reporting the signal strength of each access point. The client might also prompt for confirmation before reporting that a person is using a conference room (providing a "never report from this conference room" option for people who trigger many false prompts).

Assuming a system can reliably determine when people are in a conference room, anonymity requirements can be addressed with the same methods used in *Bustle*. Each report provides enough information for the server to infer what conference room a person is reporting from, but does not allow the server to determine who is reporting. Further, no information is released by people not in a conference room. A server could therefore make live room information available without introducing any new privacy concerns.

DISCUSSION

We have presented hitchhiking, a new approach to anonymous and privacy-sensitive collection of sensed data in location-based applications. Hitchhiking supports a general category of applications that collect sensed data from locations of interest. We have used *Bustle* as an example to illustrate potential threats to hitchhiking applications and demonstrated how hitchhiking counters each threat. Several additional examples show that hitchhiking can be applied to a diverse set of problems. Some of our later examples are implemented using the same methods in earlier examples, providing evidence that our approach addresses a general category of applications.

While the focus of this paper is on preserving privacy and anonymity, it is worth addressing the question of whether these types of location-based applications are appropriate. The applications presented in this paper can all be built in other ways, typically by installing custom sensing infrastructure. We believe that hitchhiking warrants consideration exactly because it requires no additional infrastructure. Based entirely in software on devices that people already carry, hitchhiking applications can be deployed at extremely low cost. But applications that ignore privacy concerns (such as the continuous location

disclosure required by Zipdash) can be dangerous. We have therefore presented a general approach to anonymity and privacy in hitchhiking applications. We preserve the full desired functionality of these applications while removing privacy threats that would otherwise arise.

Query Anonymity

While our approach ensures the anonymity of reports from a location, it cannot protect the anonymity of queries about locations. People requesting information about a location may not be in that location, so they can only refer to the location via an identifier or some other server-provided information. If live data is critical, prior work on masking queries can be applied (such as querying for all of the locations in a sufficiently large region to mask which location a person is actually interested in). It might also be appropriate to use a model of typical conditions at a location. A server could use reports to update this model, and clients could occasionally download the most recent model. The model could then be evaluated locally without revealing interest in a specific location.

The exception is that a person can anonymously query for information about a location that they have previously visited. For example, consider that *Bustle* could make a note of what access points were detected when a person visited a particular coffee shop. Next time the person wanted to query information about that coffee shop, it could send a query of the form "*Tell me about the current state of the coffee shop in which I previously detected these access points*." Because this query is based on the access points that *Bustle* actually sensed in that location (not just those that the server claims are located in that coffee shop), it does not contain any server-provided identifier.

Live Reports

The applications presented in this paper all make live reports, but connectivity and live reports are not a critical component of hitchhiking. Applications could store reports locally, uploading them when a connection becomes available. This seems especially appropriate if the issue of query anonymity has led an application designer to use a locally cached model instead of live queries. For example, *Bustle* can detect ARP broadcasts on a network even though the laptop user has not yet authenticated with the WiFi provider. While *Bustle* is unable to send live reports in such a situation (because the WiFi provider requires authentication before allowing Internet access), data could be stored locally until a connection is available. If *Bustle* were based on temporal models of when space is typically available in a given coffee shop, this would be appropriate.

Transport Layer Attacks

Because hitchhiking is based in controlling what information is released to an application server, it cannot protect against malicious network operators or other transport layer attacks. Consider that the provider of a mobile phone network always knows the location of each phone (otherwise the provider would be unable to route an incoming call to the phone). Similarly, a malicious WiFi operator could log the MAC address of every computer that uses an access point. The potential for this type of attack is inherent to current phone and WiFi networks, but our approach allows applications to collect sensed data without introducing any new threats. If these types of attacks need to be addressed, prior work on Onion Routing [22] or temporary WiFi MAC addresses [9] provide a solution.

Denial-of-Service Attacks

The anonymity provided by our approach opens servers to denial-of-service attacks that flood an application with fraudulent reports. The usual approach to this problem would be to give each person an identifier to include with their reports, banning their identifier if they appear to submit fraudulent data. But this obviously undermines anonymity. Instead, application servers might note the IP address used to transmit each report, as it seems unlikely that an IP address would be used to legitimately report on more than a handful of locations in a short period of time (consider that all of *Bustle*'s reports on a given coffee shop will be coming from the external IP address of that shop's WiFi service provider). Databases can also be seeded with false data to detect attacks. *Bustle* servers could include a non-existent MAC address in the list of access points for each coffee shop. Any report that claims to have detected this access point is clearly fraudulent. Such approaches do not completely preclude denial-of-service attacks, but they do make such attacks more difficult.

Timing-Based Attacks

Because the content of a report does not allow for tracking, the final avenue for inducing a tracking violation lies in timing-based attacks. If a *Bustle* client reported from a coffee shop every 5 minutes, then there is a high probability that two reports received 5 minutes apart were generated by the same person. More advanced inference might allow the later recognition of the same person. These attacks can be addressed by synchronizing reports (clients synchronize with an Internet time server and report at the same time, making the timing of their reports indistinguishable).

A more subtle issue arises if there are many locations of interest defined in an area. If a person approves disclosure from two locations that are near each other, it might be possible to track their movement from one to the other. If they have approved many locations along a path, it might be possible to track their movement along the path. In many ways, this is the problem of mix zones discussed by Beresford and Stajano [2]. As such, we can address this problem by limiting the frequency with which a device makes reports (not just the frequency of reports to a particular application). If a device makes only one report (whether that is about a coffee shop, traffic, bus, or conference room) every 5 to 10 minutes, reports will be sufficiently sparse to ensure that timing is not a threat.

CONCLUSION

We have presented hitchhiking, an anonymous and privacy-sensitive approach to a category of location-based applications. The fundamental tenet of hitchhiking is that reports are always strictly about a location and cannot be tied to a person. By presenting a privacy risk analysis of hitchhiking, this paper provides designers of location-based applications and services with an approach to building a useful class of application while also protecting end-user privacy. Implemented entirely in software on the client device, hitchhiking does not require new hardware or a trusted middleware platform. It is therefore possible to deploy applications on existing phone and WiFi networks, without the active cooperation of the network provider. By enabling anonymous and privacy-sensitive data collection, hitchhiking protects users and removes personal privacy as an obstacle to a category of location-based applications.

ACKNOWLEDGMENTS

We thank all of the contributors to Place Lab, `jpcap`, `libpcap`, and the JDesktop Integration Components. We also thank Ian Li for his help with the artwork used in *Bustle*. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010, by an AT&T Labs fellowship, and by the National Science Foundation under grants IIS-0121560 and IIS-032531.

REFERENCES

1. Bahl, P., Balachandran, A., Miu, A., Voelker, G.M., Russell, W. and Wang, Y.-M. (2002) PAWNS: Satisfying the Need for Ubiquitous Connectivity and Location Services. *IEEE Personal Communications Magazine (PCS)*, *9* (1).
2. Beresford, A.R. and Stajano, F. (2003) Location Privacy in Pervasive Computing. *IEEE Pervasive Computing*, *2*(1). 46-55.
3. Cheverst, K., Davies, N., Mitchell, K. and Friday, A. (2000) Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project. *Proceedings of the ACM Conference on Mobile Computing and Networking (MOBICOM 2000)*, 20-31.
4. Consolvo, S., Smith, I., Matthews, T., LaMarca, A., Tabert, J. and Powledge, P. (2005) Location Disclosure to Social Relations: Why, When, & What People Want to Share. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2005)*, 81-90.
5. Cranor, L., Langheinrich, M., Marchiori, M. and Reagle, J. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. *http://www.w3.org/TR/P3P*
6. Dey, A.K. and Abowd, G. (2000) CybreMinder: A Context-Aware System for Supporting Reminders. *Proceedings of the International Symposium on Handheld and Ubiquitous Computing*, 172-186.
7. Griswold, W.G., Shanahan, P., Brown, S.W., Boyer, R.S., Ratto, M., Shapiro, R.B. and Truong, T.M. (2004) ActiveCampus: Experiments in Community-Oriented Ubiquitous Computing. *IEEE Computer*, *37*(10). 71-81.
8. Gruteser, M. and Grunwald, D. (2003) Anonymous Use of Location-Based Services Through Spatial and Temporal Cloaking. *Proceedings of the ACM Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, 31-42.
9. Gruteser, M. and Grunwald, D. (2003) Enhancing Location Privacy in Wireless LAN through Disposable Interface Identifiers: A Quantitative Analysis. *Proceedings of the ACM International Workshop on Wireless Mobile Applications and Services on WLAN (WMASH 2003)*, 46-55.
10. Hightower, J. and Borriello, G. (2001) Location Systems for Ubiquitous Computing. *IEEE Computer*, *34*(8). 57-66.
11. Hong, J.I. and Landay, J. (2004) An Architecture for Privacy-Sensitive Ubiquitous Computing. *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys 2004)*, 177-189.
12. Hong, J.I., Ng, J.D., Lederer, S. and Landay, J. (2004) Privacy Risk Models for Designing Privacy-Sensitive Ubiquitous Computing Systems. *Proceedings of the ACM Conference on Designing Interactive Systems (DIS 2004)*, 91-100.
13. Iachello, G., Smith, I., Consolvo, S., Chen, M. and Abowd, G. (2005) Developing Privacy Guidelines for Social Location Disclosure Applications and Services. *Proceedings of the Symposium on Usable Privacy and Security (SOUPS 2005)*.
14. Kearney, C. Ex-AOL Employee Sentenced to 15 Months in Spam Case. *Washington Post*, August 17, 2005.
15. Krumm, J., Cermak, G. and Horvitz, E. (2003) RightSPOT: A Novel Sense of Location for Smart Personal Object. *Proceedings of the International Conference on Ubiquitous Computing (UbiComp 2003)*, 36-43.
16. LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G. and Schilit, B.N. (2005) Place Lab: Device Positioning Using Radio Beacons in the Wild. *Proceedings of the International Conference on Pervasive Computing (Pervasive 2005)*, 116-133.
17. Langheinrich, M. (2002) A Privacy Awareness System for Ubiquitous Computing Environments. *Proceedings of the International Conference on Ubiquitous Computing (UbiComp 2002)*, 237-245.
18. Lessig, L. (1999) *Code and Other Laws of Cyberspace*. Basic Books, New York, NY.
19. Maclean, S.D. and Dailey, D.J. (2001) MyBus: Helping Bus Riders Make Informed Decisions. *IEEE Intelligent Systems*, *16* (1).
20. Palen, L. and Dourish, P. (2003) Unpacking "Privacy" for a Networked World. *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2003)*, 129-136.
21. Patterson, D.J., Liao, L., Fox, D. and Kautz, H. (2003) Inferring High-Level Behavior from Low-Level Sensors. *Proceedings of the International Conference on Ubiquitous Computing (UbiComp 2003)*, 73-89.
22. Reed, M., Syverson, P. and Goldschlag, D. (1998) Anonymous Connections and Onion Routing. *Proceedings of the IEEE Symposium on Security and Privacy (SP 1997)*, 44-54.
23. Schilit, B.N., LaMarca, A., Borriello, G., Griswold, W.G., McDonald, D., Lazowska, E., Balachandran, A., Hong, J.I. and Iverson, V. (2003) Challenge: Ubiquitous Location-Aware Computing and the Place Lab Initiative. *Proceedings of the ACM International Workshop on Wireless Mobile Applications and Services on WLAN (WMASH 2003)*, 29-35.
24. Smith, I., Consolvo, S., Hightower, J., Iachello, G., LaMarca, A., Scott, J., Sohn, T. and Abowd, G. (2005) Social Disclosure of Place: From Location Technology to Communications Practices. *Proceedings of the International Conference on Pervasive Computing (Pervasive 2005)*, 134-151.
25. Voelker, G.M. and Bershad, B.N. (1994) Mobisaic: An Information System for a Mobile Wireless Computing Environment. *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1994)*, 185-190.
26. Zipdash - Mobile Map and Traffic App. *http://www.zipdash.com*