Providing Location Information in a Ubiquitous Computing Environment

Mike Spreitzer and Marvin Theimer

Xerox Palo Alto Research Center

Abstract

To take full advantage of the promise of ubiquitous computing requires the use of location information, yet people should have control over who may know their whereabouts. We present an architecture that achieves these goals for an interesting set of applications. Personal information is managed by User Agents, and a partially decentralized Location Query Service is used to facilitate location-based operations. This architecture gives users primary control over their location information, at the cost of making more expensive certain queries, such as those wherein location and identity closely interact. We also discuss various extensions to our architecture that offer users additional trade-offs between privacy and efficiency. Finally, we report some measurements of the unextended system in operation, focusing on how well the system is actually able to track people. Our system uses two kinds of location information, which turn out to provide partial and complementary coverage.

1 Introduction

Mobile and ubiquitous computing requires and can exploit a variety of kinds of location information[9, 7, 4]. Just providing a person with access to their normal computing services on a continual basis requires that their location be known to a certain extent. In addition, if information is available about who and what is in the vicinity of a person, then that person's computing environment and applications can behave in a context-sensitive manner. Applications can reflect a user's current circumstances and and can respond to changes that might occur in the user's environment.

While desiring to exploit location information, we consider unrestricted access to a person's location data to be an unacceptable invasion of privacy[5]. One way to address this fundamental tension between location-based functionality and privacy is to try to give each user control over their location information and over who may gain access to it. Unfortunately, guaranteeing that no one can gain unauthorized access to one's location information is, in general, very difficult and expensive.

Another important issue is the accuracy and temporal resolution of location information. The sensing facilities we have available to us are not perfect and hence it is important to determine how well they work in practice. Temporal resolution comes into play because it implicitly defines how small a movement the sensing facilities are able to distinguish, and hence how quickly they are likely to detect a change in someone's location. Providing useful location information to applications thus faces both the problems of limits of the location sensing technologies used as well as protection of users from abuse of those technologies.

A topic not covered in this paper is the spatial resolution provided by a system and the implications that has for the kinds of applications that can be implemented. We did not explore this topic because the only spatial resolution provided by our sensing technologies is "room-level" resolution. This enables applications such as migrating display windows from one's office to a conference room, but does not easily support finer-grained applications, such as "flicking" a window from one's portable notebook computer to that of a neighbor sitting in the next chair.

In order to understand the issues of providing location information to a system we have chosen a suite of location-based applications to focus on and have designed and built a location infrastructure in support of them. The applications we have built or prototyped include the following:

Visitor guidance : Guide a person to a designated location.

Migrating windows : Migrate a user's windows to a designated location.

Note distribution : Send a message to all persons at a given location or set of locations.

*This research was supported in part by the Advanced Research Projects Agency under contract DABT63-91-C-0027.

Ubiquitous Message Delivery (UMD) : A message submitted for delivery is delivered at the soonest "acceptable" time via the most "appropriate" terminal near the recipient. Acceptable delivery time depends on the context of the recipient. For example, the recipient's profile may specify that messages below a certain priority level should not be delivered when the recipient is in a meeting with other people. Similarly, the most appropriate terminal to use will depend on which devices are available at the recipient's current location.

Media Call : A user can request to be connected to one or more other users—wherever they currently are—by the "best" means available. Choices include video, audio, and teletype "talk" connections. As with UMD, users may specify policy constraints that control which kinds of connections may be established under various circumstances.

Scoreboard : This application is an information-oriented "screen saver". When a display is not being used for anything else, it displays information of general interest, but tailored to the interests of the people nearby.

Responsive environment : A "smart building" can optimize its energy usage by exploiting knowledge about which rooms are occupied. It can also control the environmental settings of each room according to the preferences of the people in them[3].

FindNearest : Find the nearest resource or person matching a given specification, such as "color printer" or "Unix wizard".

Locations : Display the current locations of various persons, printers, copiers, etc. A common variant is to show the locations of all nearby persons, printers, etc. (see Figure 1).

Of these applications UMD and the Locations program are deployed and in use in our lab; for the other applications we have initial prototypes running.

In the remainder of this paper we describe the location architecture we have designed and built, the design rationales behind it, various extensions one could add to offer users additional privacy/efficiency trade-offs, and the current status of our implementation. We also report some measurements of the system in operation, focusing on how well the system is actually able to track people. We conclude with a discussion of the insights we have gained from our work.

2 Architecture

2.1 Key Issues

The design of a location infrastructure must concern itself with a variety of fundamental issues. In order to motivate the design of our architecture we start by presenting the key issues that we wanted to address. Examples of how applications use our architecture and more detailed design considerations are presented in later sections, after the description of the architecture itself.

Perhaps the most important assumption we make is that our system will span more than one administrative domain. This being the case, we cannot trust all parts of the system with equal measure. In particular, designs that require one to indiscriminately trust the services and servers of foreign administrative domains seem unacceptable to us. The main consequence is that we cannot simply keep everyone's location information in a federation of centralized databases, which would otherwise be the simplest means of providing a location infrastructure.

A second consequence of multiple administrative domains is that we must assume the possibility of sophisticated attempts at traffic analysis occurring in some or all parts of a system. As a result, "perfect" privacy guarantees are, in general, very hard (and expensive) to provide.

An important observation for our design is that most peoples' privacy and functionality requirements differ according to the context they are in. Many situations in which functionality is most desired are also situations in which strict privacy guarantees are not so important or where greater trust of system components is warranted. For example, coworkers in a company with benevolent management might be perfectly willing to have their whereabouts known to each other while at work, but might insist on exercising far greater control over who may know their movements when off the job. Furthermore, if the building they work in is physically secure, they may also be willing to accept a more centralized implementation of the location infrastructure in exchange for greater functionality or efficiency.

Our canonical example of an untrusted, or partially trusted, environment is a shopping mall. It would be undesirable to allow just anyone (such as junk mail senders) to have access to all one's movements within the mall, yet one might wish to be visible to, or reachable by, a select set of friends and family.

An important consideration to keep in mind is that in many circumstances having one's privacy compromised (e.g. while at the mall) is an inconvenience rather than a real problem. Hence, providing guaranteed privacy all the time at a high price—say, in the form of too little functionality and/or too high a performance cost—will not reflect users' true needs. On the other hand,

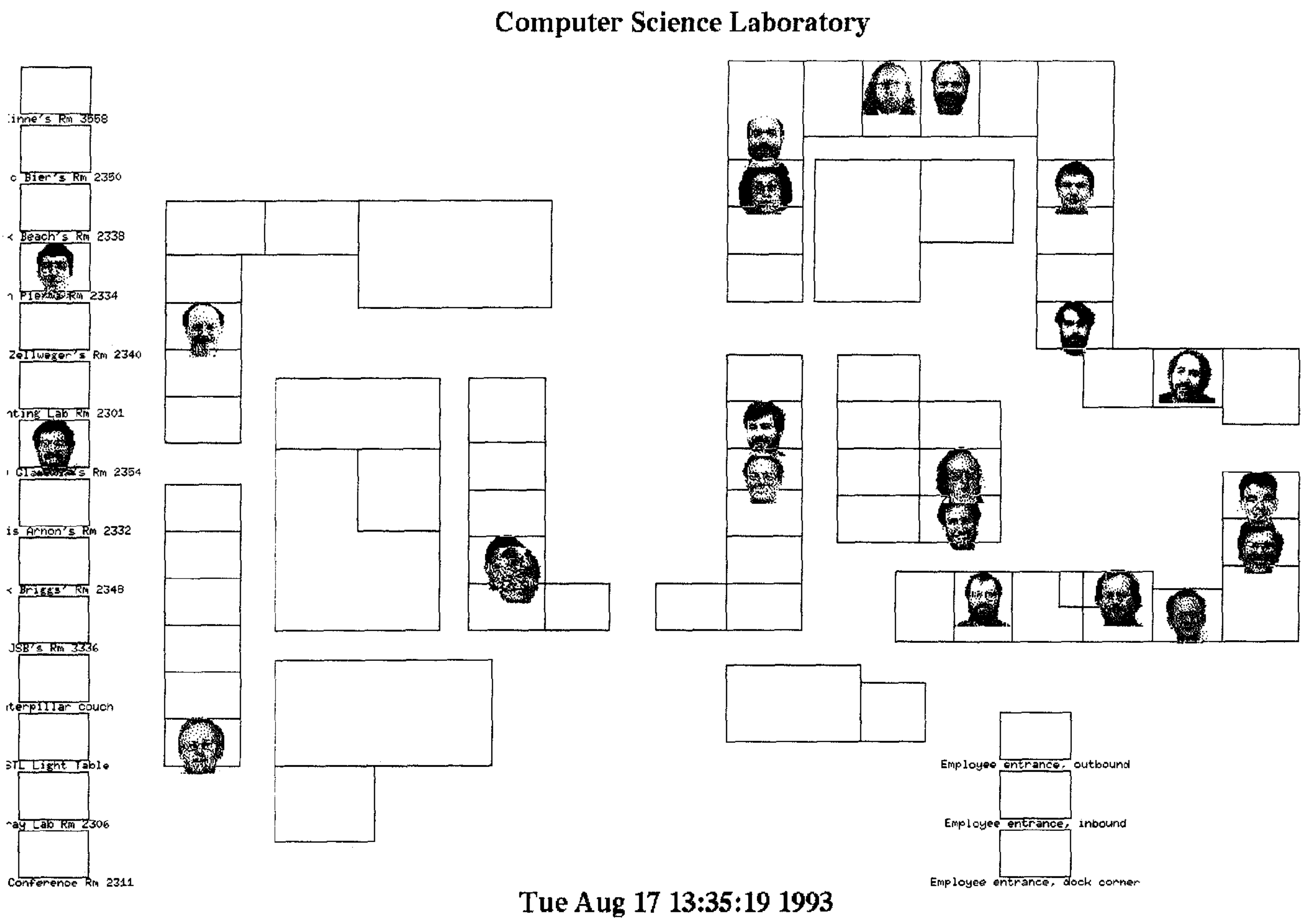


Figure 1: Output of one variant of the Locations program: displays the location of all nearby people willing to be publicly visible.

there are clearly circumstances when very strong privacy guarantees are a requirement. The conclusion we draw from these examples is that we need an architecture that provides user-controllable trade-offs between privacy guarantees and both functionality and efficiency. Much of our design focuses on how to selectively regain the efficiency achievable by centralized designs when users are willing to risk trusting various system components to some degree.

2.2 Description

Figure 2 illustrates the architecture we have designed in response to the issues just discussed. The circles show programs, which run on a network of computers and communicate via RPC. Some of the computers are connected by a wired network; some may be portables that communicate wirelessly. The black arrows show the flow of location information while the gray arrows show the path of a ubiquitous message delivery. Not shown are various application-specific servers and our Name-and-Maintenance Service, which uses familiar techniques to enable lookup of servers by name and keep programs up and running.

There is one User Agent for each user. Access control for personal information is implemented primarily by a user's agent. That is, each User Agent collects and controls all the personal information pertaining to its user and applications can only get personal information from a user's agent—and only if that agent consents. In fact, a user's agent can lie about the user's personal information, because consumers of that information have no other authoritative way of determining that information. A user's agent is under the control of the user: he determines the policies implemented by his agent, chooses which implementation of the agent to run (or

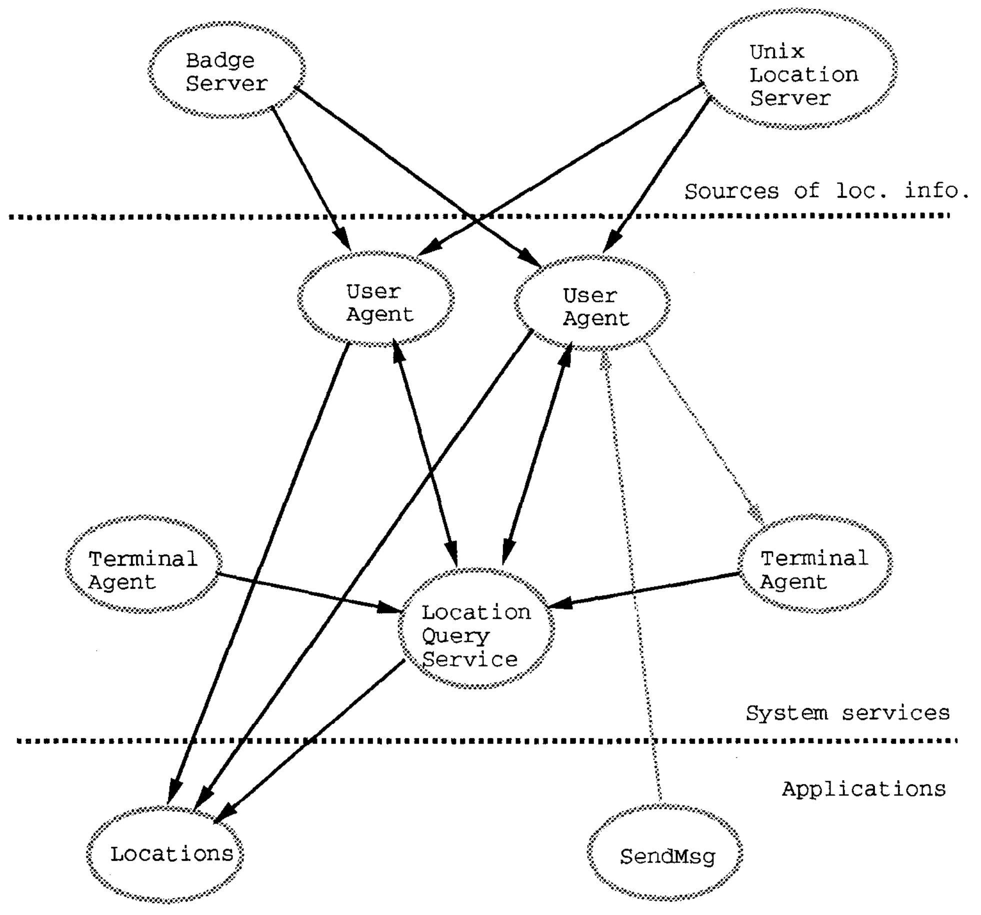


Figure 2: Basic system architecture, with an instance of UMD message delivery illustrated by the gray arrows.

writes his own), and runs it on one or more computers he trusts.

A User Agent consists of several modules, some of which perform system infrastructure functions, and some of which are responsible for implementing the agent's responsibilities for specific applications. User Agents are the locus of control for customizing applications with respect to their environment. That is, knowledge about the user's environment and context and his preferences with respect to various circumstances are maintained in the User Agent and propagated from there to the user's various applications. Thus, the User Agent serves as a general policy coordinator for both the user's privacy concerns as well as his context-sensitive customization concerns.

Each User Agent collects location information from a variety of sources, examples of which might include:

1. infra-red-based active badges[8, 7],
2. wireless nano-cell communications activity[9],
3. global positioning system information[1],
4. device input activity from various computers[6],
5. motion sensors and cameras[3], and
6. explicitly specified information obtained directly from human beings.

Our existing system employs 1, 4, and 6. The badges in our system emit a unique id every 15 seconds and a Badge Server in each region passes data obtained from polling its badge sensors to interested parties. Each User Agent registers for information about the badge id representing its user; the correspondence between a badge id and a user's identity is known only by the user's agent.

In a similar fashion, the Unix Location Server polls the `rusers` daemons on our Unix workstations and passes data on each user to his User Agent. Users may also inform their User Agent of their current location explicitly by running the `AtLocation` program. Each User Agent synthesizes the (possibly inconsistent) location hints it receives from the various sources into one opinion.

A User Agent is a well-known service that external clients can ask for a variety of information about the user it represents, such as the user's current location. The User Agent will either honor or reject any request depending on the policy its user has specified. Subject to the user's policy, the agent also makes its user findable by location using the facilities described later.

Some applications work by simply interacting with User Agents. For example, submitting a message for ubiquitous delivery consists of simply giving the message to the User Agents for the recipients; each agent then takes care of getting the message to its user. Other applications, like Scoreboard, Responsive Environment, and FindNearest, are primarily concerned with a given location, and must perform some kind of a query to find the agents for the people at or near that location. Sometimes User Agents themselves need to perform location-based queries—for example, to find nearby terminals through which to present a message for ubiquitous delivery.

One of the key problems addressed by our architecture is how to keep disclosure of the association between a person's identity and that person's location under the control of that person while supporting our applications with reasonable efficiency. Our architecture provides control through the User Agent: (1) an application starting from a person's identity can only discover that person's location by asking the person's agent, and (2) an application starting from a location must use the Location Query Service (see below) to discover the identities of the people at that location, and the Location Query Service is designed to give User Agents control over the circumstances of the release of that information.

The Location Query Service (LQS) provides a way of executing location queries that offers different trade-offs between efficiency and privacy. It is built around the idea of queries over *located objects*. A located object is represented by a tuple consisting of a location, an RPC handle, and an association-list describing the object's type and other information that the object chooses to make available.[1] Examples of located objects include users (represented by User Agents) and terminals (represented by Terminal Agents). A query is a predicate over a location and an association-list; the result of a query is the set of tuples that satisfy the predicate.

A key feature of the LQS is that located objects can be anonymous. That is, a tuple's association-list may reveal only its type and its RPC handle may employ techniques such as indirection through trustworthy intermediaries to hide the true identity of the real server behind the handle. A client getting a query response listing a tuple with an anonymous RPC handle and no identity in the association-list would have to use the RPC handle to ask the object for its identity.[2] That object (e.g., a User Agent) can respond truthfully, falsely, or not at all, depending on its policy (which might, for example, require authenticating the caller).

Note that a located object could register several tuples for itself, in order to make traffic analysis more difficult.

The LQS is organized by regions, with a centralized server, called the LocationBroker, running in each region. Public objects whose identities and locations are not meant to be kept secret—such as printers and office display terminals—register a full description of themselves in the LocationBroker covering the region they inhabit. A private object—such as a User Agent—who is willing to reveal that *someone* (without revealing who) is at their current location, registers itself in the appropriate LocationBroker in an anonymous fashion.

Each region's LocationBroker also supports standing queries: a client can submit a query and a callback RPC handle, with the LocationBroker notifying the client of the change whenever the answer to the query changes. This is used, for example, by the `Locations` program to monitor a given area.

A final efficiency trade-off that LocationBrokers can be provide is to implement access control on behalf of an object. This amounts to selectively returning a tuple, or portions of its association list, in the results of a query according to a policy specified by the object when it registers itself. An object using a region's LocationBroker thus has the choice of (1) registering minimal information (location, type, and anonymous RPC handle) with the LocationBroker and implementing access control entirely on its own, (2) using (and thus trusting) the access control functionality of the region's LocationBroker, or (3) any combination of the previous two.[3] Note that for regions where most User Agents are willing to entrust their access controls to the region's LocationBroker, that region's LQS has essentially become a centralized design, with all the efficiency benefits and privacy risks that implies.

The last piece of our architecture concerns I/O devices. There is one Terminal Agent for each "terminal", or cluster of I/O devices that operate together (for example, a workstation's keyboard, mouse, screen, and sound card comprise one terminal). As with the User Agent, the Terminal Agent consists of several modules, some infrastructure and some application-specific. The agent provides access through a device-independent interface, and manages the multiple demands on the terminal.

Because terminals have owners, and are dedicated (in some cases) to specified uses, there are also policy decisions to be made by Terminal Agents. Agents for non-mobile terminals register in the LocationBroker, so that

[1] Object type is used indicate which RPC interface to use with the RPC handle.

[2] In a similar way, a client can hide its identity by issuing its queries from an anonymous RPC handle.

[3] Since our interest is in exploring what happens when servers are not trusted, we have not implemented access controls in our LocationBroker.

they can be found by location. A mobile terminal may be dedicated to a particular user and might communicate directly with that user's agent instead.

2.3 Application Examples

To illustrate how applications make use of our system, we describe how two representative applications are implemented: `UMD` and the `Locations` program. Someone wishing to send a message for ubiquitous delivery to a user can invoke the `SendMsg` program to submit a message to the user's User Agent. The User Agent keeps track of which (personal) portable computing devices its user is currently using as well as what "public" terminals and people are near the user's current location. The latter is achieved by registering for callbacks with the LQS for the user's current location. When a message is submitted to the User Agent for delivery, it checks to see if the user's current situation allows delivery of the message (for example, the user's policy profile may specify that only priority messages should be delivered when the user is in the presence of other people) and if a suitable terminal is currently available. If so, it sends the message to the terminal's Terminal Agent; otherwise it waits until the user's circumstances and/or location change and then tries again.

More than one terminal may be available—for example, if the intended recipient is in their office, they might have access to both their workstation and a portable paging device, if they are carrying one. In this case the User Agent picks the most appropriate one, where appropriateness depends on terminal characteristics as well as whether the message to deliver is marked private—and hence shouldn't be delivered to terminals whose display might be publicly visible (as is the case with workstations). Terminal characteristics are exported in the association-list that a Terminal Agent includes when it registers with the LocationBroker (or User Agent if it is dedicated to a particular user).

In a system with many users, the `Locations` program needs to be told which users to display information for. One way is to provide an explicit list of user names. In this case the `Locations` program contacts those users' agents and requests to be kept appraised of any changes in their users' locations (assuming they consent).

Another way to limit things is to have the `Locations` program show all users within a specified physical area. Consider what happens when the area fits entirely within an LQS region. The program issues a callback registration to the region's LocationBroker, asking to be notified of any changes in the area due to User Agents. All User Agents currently registered in the area with the LocationBroker will have their registrations returned in the LocationBroker's initial response to the callback registration. Any User Agents whose users enter the LQS region at some future point in time and register themselves in the area with the LocationBroker will have their registrations returned to the `Locations` program via callback notifications. Similarly, whenever a User Agent leaves the area or changes location within it, a callback will be made to notify the `Locations` program. An area that intersects multiple LQS regions can be handled by performing the above operations in each region.

3 Design Considerations

3.1 Design Principles

As discussed in the previous section, perfect privacy guarantees are difficult to provide. Consequently we have designed a system that allows users to trade off increasing levels of privacy risks for increasing levels of location-based functionality and efficiency. The following three implementation principles guided our work:

- Structure the system so that users have both the ability and the choice to not be visible to the various sensor networks and servers in the system.
- Avoid requiring personal information to be placed in servers (which may be in untrusted administrative domains).
- Use encryption and anonymous handles to limit the kind of information being revealed.

Ideally, sensing systems such as active badges and activity-monitoring OS services would establish secret and authenticated communications channels with their users' agents. Unfortunately the technologies we currently use (Olivetti active badges and the SunOS `rusers` daemons) do not allow us to achieve this goal. Our active badges are simple fixed-signal, infra-red beacons whose emissions must be gathered by a centralized polling server. Querying `rusers` daemons suffers from the same problem and, even worse, from the fact that Unix makes this information available indiscriminately.

Use of these facilities is acceptable in a friendly environment, but would not be if we extended our system to a larger, more heterogeneous setting. Only the ability to remain silent—for example, by not wearing one's active badge—can ensure that corrupted servers and traffic analyzers will not be able to determine the identities of persons entering their domains.

An interesting unsolved problem is the question of knowing which sensor systems are actually present at a given location. For example, most people in our lab were originally unaware that the `rusers` daemon runs on their workstation by default. Similarly, most people do not think about the fact that many of their monetary transactions implicitly reveal their current locations to the merchants and banking agencies that are party to each transaction.

The potential for inadvertently revealing one's identity and location can be reduced by employing anonymity. Examples include the use of multiple, anonymous login ids and facilities such as anonymous electronic cash[2]. Similarly, applications such as Scoreboard only need profile information; they do not need explicit identity information. In our design we rely on anonymity to bridge the gap between wanting to keep location information hidden in a decentralized collection of User Agents and needing to provide some means of performing location queries. User Agents can also use anonymous handles to exercise control over which callers can discover any given piece of information (by choosing how to answer based on the caller's identity).

Queriers, which may themselves be User Agents, can also use anonymity. If both the querier and the responder are initially anonymous and unwilling to reveal themselves without further identification from the other then some additional mechanism is needed to negotiate what to do. We have not explored this topic yet.

3.2 Tradeoffs

Anonymity is not always sufficient to preserve privacy. Simply keeping track of how many people are at each location over time can reveal some strong hints about who has been where. The use of multiple, changing anonymous handles and background noise can obscure this information, but there is a long chain of measures and countermeasures that can be taken. One could even be concerned with detection of minute analog differences between individual devices.

A user must keep in mind is that there are actually several different ways that the privacy of his location can be compromised:

- Application-level operations (such as giving a message to a Terminal Agent for delivery) with untrustworthy parties can reveal location information.
- Location information directly published through the LQS is available to any querier.
- A LocationBroker might not faithfully implement the access controls it offers.
- The intermediaries used to implement anonymity might be corrupt, or not competent enough to foil the attacker.
- Traffic analysis of LQS queries or results might reveal the identity of otherwise anonymous queriers of, or objects in, the LQS.
- The various location sensing systems that gather location information might deliberately give it to other parties.
- The communications between the location sensing systems and the User Agent might not be secret and authenticated.
- The communications between the person's portable computers and his processes running at fixed locations (e.g., a User Agent) might not be secret and authenticated.

Our architecture gives users choices to limit which of the above potential exposures apply. Different potential exposures involve trusting different system services to different degrees. We must allow users to opt out at whatever level their trust is exceeded. Thus, (1) a User Agent might register a single anonymous tuple in the LocationBroker; (2) a User Agent might register multiple anonymous tuples in the LocationBroker; (3) a User Agent might not register in the LocationBroker at all; and (4) a user might disable transmissions from his portable devices (i.e., receive only—or turn them off if he's concerned about noise emissions) and refrain from identifying himself to fixed devices. Thus it is important for the system—including applications—to be tolerant of missing or inaccurate information. Uncertainty of location information (and other personal information) is now fundamentally a part of the system at every level.

It would not make sense (in a real system) for a user to give up a lot of efficiency or functionality protecting against one potential exposure while not protecting against another that applies in the same situation. For example, in a completely untrusted administrative domain, one has to assume that any of the domain's services (LocationBroker, Terminal Agents, location sensing units) could be corrupted. The unfortunate consequence of all this is that the users of a system must stay aware of who controls which aspects of the system they are currently using and must act in an accordingly consistent fashion.

Our architecture is not dependent on the exact nature of the location sensing technologies, nor the communications media, employed. For example, while the experimental system we've built to explore our design extends an inconsistent level of trust—it uses anonymous registrations in the LocationBroker, even though our active badges and Unix workstations reveal location information indiscriminantly—we were willing to accept this because known techniques could be used to provide more secure location sensing facilities.

As one possible replacement, consider using portables that have a GPS receiver and a cellular telephone. The portable could get GPS-resolution information to the User Agent, while exposing only cell-resolution location information to only the phone companies involved (if the user assumes nobody is going to use analog techniques to locate his transmitter more precisely). As another possible replacement, consider putting location beacons in each room, and having an active badge that relays a received location to its User Agent by sending an encrypted message through a chain of intermediaries. In this case the User Agent gets room-resolution location information, trusting the intermediaries to hide the link between location and identity, and revealing that *someone* is in the room to anyone that can see such an agent-bound message and discover its room of origin.

A User Agent trades privacy against functionality when choosing how much information to reveal to which other parties. A completely mistrustful User Agent cannot be found by FindNearest or Note Distribution, cannot customize a Scoreboard or a Responsive Environment, and cannot ubiquitously deliver a message or participate in a Media Call.

In addition to allowing trade-offs between privacy and functionality, our system allows users to make trade-offs between privacy and efficiency. The LQS offers a User Agent three choices in this regard. The first choice is how much information to include in the association list describing the agent. When all the information needed by an information-seeking application is found in the association list, the application need not contact the agent to complete its job; including this information thus increases efficiency, at the privacy cost of perhaps indiscriminately revealing that information.

The second choice offered by the LQS is between registering one or several tuples for a located object. Use of several tuples will cause the User Agent to appear in more query results and hence have to answer more follow-up questions from potentially interested clients.

The third choice offered by the LQS pertains to the use of access controls within the LocationBroker. If most User Agents participating in the LQS of a region are willing to completely trust the region's LocationBroker then certain kinds of queries can be made much more efficient. In particular, queries where the set of User Agents that will appear in the result cannot be well approximated on the basis of location alone will benefit from this optimization. For example, consider querying for the k users in some set of locations whose names are alphabetically nearest one given name (such as might appear in the middle of a scrolling list). Before the final k can be chosen, all agents registered anonymously at those locations must be queried individually for their names.

3.3 Alternatives

An interesting addition to our architecture to consider is the use of multicast. One could imagine having a multicast group instead of a LocationBroker for each LQS region and having clients of the LQS multicast their location queries to all members of a region's multicast group. Interested parties, such as the User Agents of all users currently in a region, would anonymously listen to the region's multicast group to hear location queries. They would answer a query if they matched it and if their current privacy policy allowed it.

The advantage of using multicast is that it only reveals the association between an RPC handle and the region, and that only to the multicast routing infrastructure. In contrast, using the LocationBroker reveals the association between an RPC handle and a specific location. The disadvantage of multicast is increased computation and communication: location queries go to, and must be processed by, all listening objects in the *region* instead of just the LocationBroker, and the cost of multicasting to User Agents located somewhere on the Internet may be substantially more than the cost of communicating just with the LocationBroker. Note also that we require *reliable* multicast for the design just described.

One way to address the inefficiency problems of multicast is to offer it as an option, in addition to the option of using a LocationBroker. Thus, each LQS region could maintain both a multicast group and a LocationBroker, with the LocationBroker listening to its region's multicast group and processing all location queries against the objects that are registered in it. User Agents would thus have a choice between listening to a region's multicast group for greater privacy or registering with a region's LocationBroker for greater efficiency.

Unfortunately, if multicast is unavailable then clients have no choice but to register with the LocationBroker (if they wish to be findable) and accept the increased risk that implies. Note also that a User Agent wishing to listen to a region's multicast group must be able to register in the multicast group from whatever address it (or the last intermediary of its anonymous indirection chain) has. Such functionality is not yet widely available in the Internet.

There are other, radically different, architectures one could design to protect one's privacy, but these do not support all the applications we are interested in. For example, if all we cared about were visitor guidance then a simple scheme whereby each location contains a broadcast location beacon that could be received by nearby portable devices would suffice. Such a scheme would provide strong privacy guarantees as long as no portable device tried to communicate with the rest of the world. Another alternative design could be constructed around "proximity-based" communications and sensing facilities. These could be used to enable communications and presence detection among objects at a particular location without requiring more wide-ranging communications that would be easier to monitor by external parties. Such facilities, combined with portable computer devices, could be used to implement things like Scoreboard, in-room note distribution, and in-room window migration. However, finding out about things beyond one's immediate proximity—as is needed by the FindNearest and Locations applications would not be possible.

4 Status and Experience

Our location infrastructure is built and currently deployed within part of our lab with 12 User Agents, 21 Terminal Agents, and one LocationBroker running. The active badge system includes about 120 infra-red sensors that are deployed in about 70 offices, 10 common areas and lab rooms, and the corridors interconnecting them.[4] This represents about 10% of the entire floor plan of our building. Thus, the people participating in our system are still outside the system a considerable amount of time; both during the work day and outside of it.

As mentioned in the introduction, the two applications currently in use in our system are the Locations program and UMD. The more popular one is the Locations program, which people tend to keep running all the time in a background window on their workstations. Its primary use seems to be as a quick "hint" reference source to know if someone is currently in or where they might currently be. When run with the -map option, the program also provides a convenient map of the lab instead of just an alphabetically sorted list of (name, location) pairs. However, the map output option also takes up more screen space, making it less popular as a permanent background facility.

The UMD application manages to successfully deliver about 63% of all submitted messages to their intended recipients within one minute and about 73% within five minutes.[5] Unfortunately, these statistics are not very informative because they are dominated by the fact that our users are not always near a display terminal and are frequently outside the range of our system altogether. As a consequence, messages may require a considerable time before being delivered, even when the basic location tracking system is functioning perfectly.

Our experience with UMD verified our expectation that recipient context is very important. Originally, we silently popped up a window when a message was delivered to someone. However, because our terminals run screen savers when they aren't in active use, many offered messages didn't get noticed. Unfortunately, when we added an audio beep to announce message delivery, we found that message delivery was perceived as being intrusive if the recipient was with other people. This was especially the case if an unimportant message was delivered to the electronic whiteboard of one of our conference rooms during a meeting. Although UMD provides mechanism for implementing context-sensitivity, it is still unclear what policies are desirable to effect ubiquitous message delivery in both an effective and a socially desirable manner.

In the remainder of this section we describe the data we have gathered concerning how well our system is able to track people.

4.1 Active Badge Tracking System

Our badge system consists of strings of infra-red sensors, mounted in the ceilings of rooms and corridors, that are periodically polled by programs running on workstations. Offices and corridors typically have one sensor installed in them, while common areas and lab rooms have between two and four sensors installed. Our installation includes three separate strings of sensors; each attached to a different workstation. Each of the three poller programs feeds its raw data into the Badge Server, which then forwards the appropriate parts to each User Agent that has registered with it.

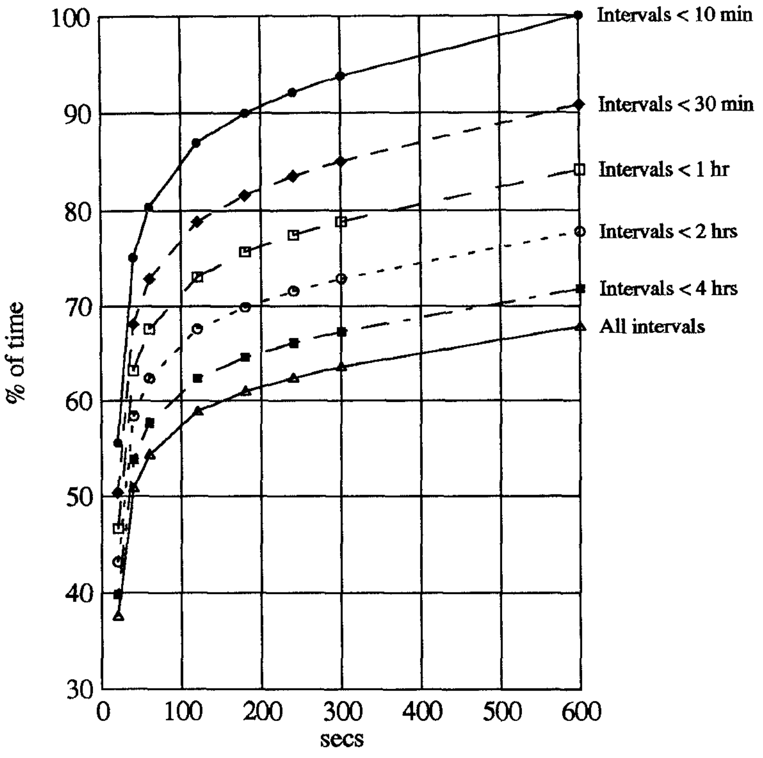
Our badges emit a fixed-id signal every 15 seconds and it takes roughly 2 to 3 seconds for each poller program to interrogate all the sensors on the sensor string it is responsible for; this implies that the minimum temporal resolution of our system is about 15 to 18 seconds. In order to get a handle on how reliably the infra-red sensors manage to detect badge emissions, we have structured the data presented in this section around the notion of a "sighting interval", which is the time between subsequent sightings of the same badge (or a person's input activity in the case of the rusers data presented later on). If badge emissions are reliably detected by the sensor system then the average sighting interval for a person, while they are in the area covered by the sensors, should be around 15 to 20 seconds. Longer sighting intervals will occur when a badge's emissions are missed by the sensor system.

Because people are frequently not within the area that the badge sensor system covers, we have applied two heuristics in this paper to account for absences. To approximate the working day, we only consider badge sighting intervals between the first sighting of a day and the last sighting, with days considered to end at 2AM. While this heuristic does the wrong thing for people who work at 2AM, none of our subjects fall into that category. We have also tried to filter out periods when someone leaves the badge sensor area to go to another part of the building or to leave the building during the "work day". This is done by excluding from consideration intervals longer than an upper bound; we consider several different values for this upper bound because other effects (such as obstructing a badge's emissions) can also produce long intervals.

Figure 3 shows cumulative time graphs of badge sighting intervals by interval length, with various upper

[4] Our infrastructure is a follow-on to an earlier and simpler, but more widely deployed one.

[5] Successful delivery means that the recipient explicitly acknowledged receipt of the message.



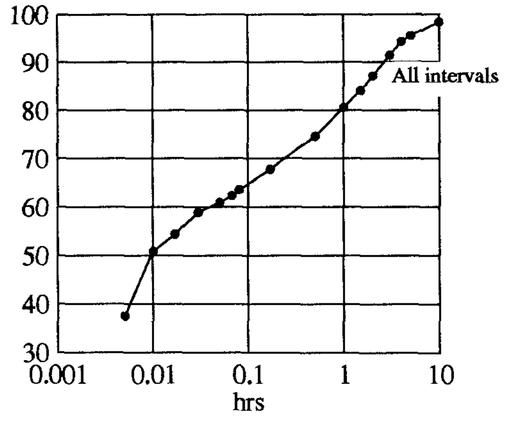


Figure 3: Cumulative graphs of badge sighting intervals by interval length, with various upper bounds on interval length considered.

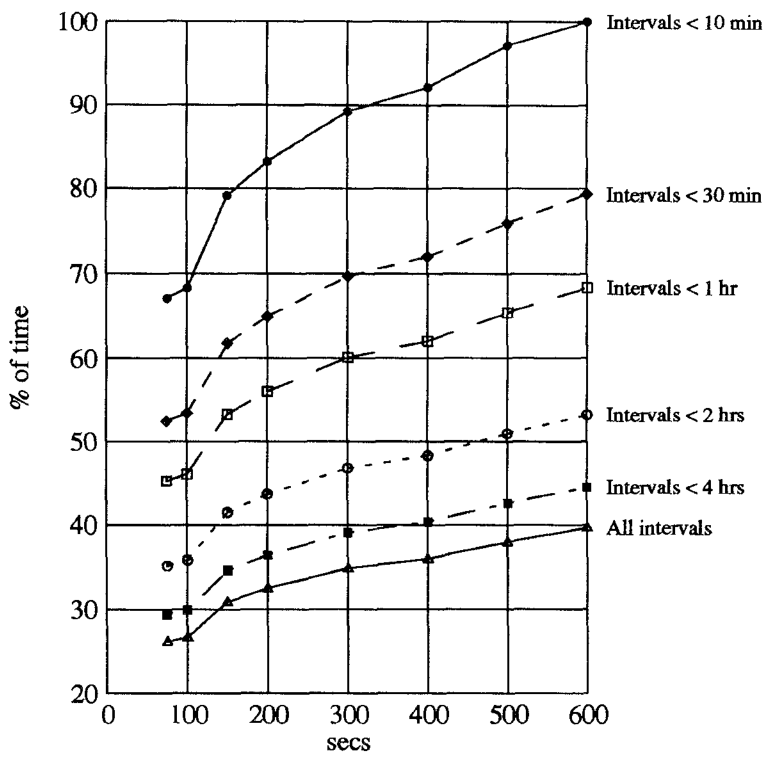
bounds on interval length considered. The curve for all intervals (within a "working day") is shown extended out to 10 hours in the side plot. Each point on a curve represents the total amount of time spent in intervals shorter in length than that point's x axis time value as a fraction of the time spent in all intervals considered for the curve. Note that over the region shown by the main figure, the curves are actually just scaled versions of each other because the sum of the interval values used for any x axis point is the same for each curve. The purpose of showing multiple curves is to give the reader some idea of how the (same) data looks as we apply ever-more-aggressive versions of our heuristic for filtering out intervals during which a person is outside the system.

When all intervals during the "working day" are included then short sighting intervals account for a distressingly small percentage of the time. Accounting for likely absences improves the numbers but still leaves significant periods of time during which a user is sighted only after a lengthy interval.

When the badge sighting data is broken down by person, considerable variation is seen between people. For example, the percentage of time spent in intervals less than 20 seconds long varied from 9% to 63% for the "all intervals" cases. When intervals greater than 1 hour are thrown out then the time spent in intervals of less than 20 seconds varied from 12% to 78%. These variations seem to be due to a variety of factors, such as time spent outside the badge system area during the day, whether or not a person wears their badge all the time, and how "visible" their badge is to sensors. The latter issue is problematic for several reasons:

- Both natural light and some of our ceiling lighting interfere with the sensitivity of our IR sensors.
- Many people prefer wearing their badge on their belt rather than pinned to their chest. Unfortunately a belt-worn badge is frequently obscured by a person's arms or other objects; especially when they are seated.
- Our offices typically have only one sensor in them, yet people tend to face different directions when performing different activities. A common example is working at a computer versus talking with a colleague.

One of the questions we had with using an infrared-based badge system was how often multiple sensors would see a badge at the same time. This can occur in large rooms containing multiple sensors, at corridor intersections, and for glass-walled offices that happen to have a corridor sensor outside them. Our system has multiple-location sightings about 0.2% of the time; with the bulk of them occurring in our meeting rooms and



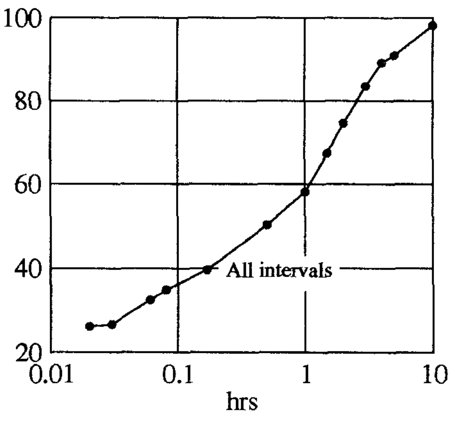


Figure 4: Cumulative graphs of computer input sighting intervals by interval length, with various upper bounds on interval length considered.

lab rooms containing multiple sensors. Note that multiple sightings are not a problem for our architecture since uncertainty is part of the system in any case.

4.2 Computer Input Tracking System

The basic design of the Unix Location Service is the same as that of the Badge Server: the Unix Location Server polls the rusers daemon on each workstation in our lab once every 60 seconds to find out when the most recent input activity occurred and which user login id it occurred for. The results are then forwarded to the appropriate User Agents. Figure 4 describes the data we have gathered for this service.

The rusers data displays similar characteristics to that of the badge system. That is, the time spent in intervals of small size represents only a small fraction of the total time spent in all sighting intervals, with the fraction significantly improving as larger intervals are excluded from the data. The breakdown by person again yields significant differences due to different people's work patterns.

4.3 Overlap Between Tracking Systems

One of the most interesting things we observed about our two tracking systems is that they tend to complement rather than overlap each other. Table 1 lists how often only a person's badge was seen, only a person's computer input activity was seen, both were seen, and neither were seen, as a fraction of the total time that the person is in the system. A person is considered to be "in the system" when they are not absent from the badge data and not absent from the input activity data. As before, we define a person to be absent from sensor data during intervals longer than various bounds. We define the notion of a person being "seen" during a sighting interval as meaning that the length of the interval is less than some cut-off value. The table gives overlap statistics for several different absence bounds and "seen interval" cut-off values, the smallest cut-off value being set at a value slightly larger than the minimum sighting interval of either tracking system. The important thing to note is that the fraction of time during which both badge and input activity are seen is quite small, both in absolute terms and relative to the fractions of time during which only one or the other was seen.

We attribute this phenomenon to primarily two things: (1) people working at home will be seen by their computer input activity and not by the badge system, and (2) people who wear their badge on their belt and are typing at their workstation will tend to obscure their badge's emissions while having clearly visible computer input activity.

Seen interval cut-off size:	75 sec.	150 sec.	5 min.	10 min.
All working-day intervals:				
Only badge seen:	19%	20%	21%	21%
Only input activity seen:	17%	20%	21%	23%
Both seen:	7%	9%	10%	13%
Neither seen:	57%	51%	48%	43%
Only intervals less than 1 hr. considered:				
Only badge seen:	26%	26%	28%	29%
Only input activity seen:	23%	28%	29%	32%
Both seen:	9%	12%	14%	17%
Neither seen:	42%	34%	29%	22%
Only intervals less than 10 min. considered:				
Only badge seen:	32%	33%	35%	36%
Only input activity seen:	31%	36%	38%	43%
Both seen:	11%	15%	17%	21%
Neither seen:	26%	16%	10%	0%

Table 1: Table of badge and computer input activity sighting overlap statistics.

4.4 Tracking Moving Persons

People sitting within their office provide a different sighting profile to the active badge system than do people who are moving around. To get a handle on how well our active badge system could track moving persons—such as visitors—we performed several "walk-about" experiments to see how well the badge system could follow us.

As mentioned earlier, the basic badge sighting interval is 15 seconds; which is enough time to walk past about a half dozen offices and perhaps a corridor or two in our lab. We found that a person who randomly walked about our corridors was seen, on average, every 22 seconds, with a standard deviation of 17 seconds. We also tried the same experiment with the badge emission period changed to 10 seconds and obtained an average sighting interval of 17 seconds, with a standard deviation of 13 seconds.

Note that 17 seconds is still enough time to add noticeable inaccuracy to an application such as `Visitor Guidance`. Decreasing the badge emission interval to 5 seconds would presumably give us an average interval length somewhere between 5 and 10 seconds; but would cut the battery lifetime of our badges from its current value of about 3 months to about 1 month.

We did not have much trouble with people being sighted in offices while walking past, even though roughly half, on average, of the front wall of each office in our lab is open to IR. Only about 11% of the sightings from hall-walking experiments were in offices. While we recognize that the exact placement of a badge sensor within a room can greatly affect this result, we mention it because our sensors were placed in a fashion to optimize office coverage, without much concern about the hall "cross-talk".

We infer from this that message delivery "chase" effects while people move about should probably not disturb the denizens of every office a person walks by. Anecdotal evidence has corroborated this — only one person has reported seeing an attempt at ubiquitous message delivery in his office for someone not present.

5 Conclusions

We have designed and built an infrastructure for providing location information and various applications that use that information. The architecture we advocate is a user-centric one in which the personal information for each user—including location information—is managed and controlled by that user's User Agent. A Location Query Service consisting of a LocationBroker per region is provided to facilitate queries by location.

The principle assumptions behind our design were two-fold:

- The design should scale to use in multiple administrative domains.
- We did not rule out the existence of untrustworthy servers and sophisticated traffic analysis attacks in some domains.

The consequences of these assumptions were that we could not use strictly centralized designs that rely on trusted location databases and we had to accept the fact that strict privacy guarantees are in general difficult and expensive to provide.

The hybrid decentralized architecture we designed gives each user a range of privacy options that they may dynamically choose from. At one extreme is the ability to simply "opt out" of the system, exchanging any participation in (and benefit from) the system for a fairly strong guarantee of privacy. At the other extreme is the ability to convert any trusted region into an efficient centralized design by simply having everyone register themselves in that region's LocationBroker with the appropriate access control specifications. In between, are two levels of anonymity that users can choose to assume, depending on the trust they place in a region's servers and the level of risk aversion they wish to employ.

The price we paid for the decentralized nature of our architecture is increased communications and processing overhead. The worst case occurs for applications like the `FindNearest` and `Locations` programs, which cannot narrow the set of people they are interested in until *after* they have received query responses from many potential candidates. We believe that in practice the additional overhead will rarely be a problem: applications that are continually interested in changing location information can use the callback facilities to obtain incremental updates and "one-shot" applications, like `FindNearest`, are typically not run so frequently as to overwhelm the system's resources. Our personal experience, to date, has borne this out.

Two important qualitative implications of our architecture are the following:

- Uncertainty is a *fundamental* aspect of our system that is visible at the applications level.
- Only certain kinds of location sensing technology can be deployed if users are to be able to hide from the system at will.

Uncertainty has strong implications for a variety of our applications. For context-sensitive applications, such as `UMD` and `Media Call`, it means that they must always assume the possibility that invisible people are at any given location unless explicitly told otherwise. For applications such as `FindNearest` and `Locations`, uncertainty means that they must be viewed as "hint" services. Despite this, we have still found location information to be quite useful; with the `Locations` program being the most popular application in our running system.

Our requirement that users be able to completely hide from the system has strong implications for which sensing technologies may be deployed. Users can hide from our active badge system by simply taking off their badge. If cameras were deployed throughout our lab then it would be almost impossible to allow some people to hide from the system while still being able to track others. In general, any technology that can track some unremovable, unhidable aspect of people must be avoided if we wish to allow people to remain hidden from the system.

Our quantitative experience with providing location information has primarily covered the efficacy of the location sensing systems we deployed. We found that both our infra-red-based badge system and our `rusers` computer input monitoring service gave only partial coverage, even when time spent outside the system was taken into account. While we suspect that substantially greater numbers of badge sensors—probably several per office—could significantly improve our badge tracking performance, this would also substantially increase the cost of deployment of the system.

Interestingly, our two sensing systems tended to complement each other rather than being redundant; thus we obtained a real benefit from employing more than one tracking system. An informative extension to our system would be the introduction of radio-based nanocells[10], which would suffer from a different set of problems than infra-red. It is unclear whether it would be more economical to improve tracking performance by beefing up one of our sensing systems or by trying to deploy additional different ones. We are currently deploying portable notebook computers using nano-cell radio communication, personal communication devices using more advanced infra-red communication, and additional public display devices. These should improve both the coverage and variety of our location information, as well as giving us a greater number of devices through which our applications can interact.

In addition to the question of what accuracy of location information is attainable, there are a variety of open issues that remain to be addressed by future work. Perhaps most important of these is the question of how accurate location information needs to be, given the fundamental uncertainty introduced by peoples' desire for privacy. Our current coverage is sufficient to enable useful, though imperfect, versions of all our applications to be implemented. Until we have further actual usage experience with our system it will be difficult to tell how much users actually value various levels of privacy versus functionality and how important either accuracy or efficiency considerations will turn out to be.

We are also curious to see what kinds of privacy policies users actually deploy. Our architecture is designed to provide users with a great deal of flexibility and control, but it is not at all clear how much users will actually take advantage of all the options offered them. We suspect that in the long run most users will settle into a small number of usage "modes" that reflect common situations, such as maximum trust and functionality (e.g. being at home), a fair amount of trust and lots of functionality (e.g. most of the time at work), less trust and less functionality (e.g. being at a shopping mall), and no trust with no functionality (e.g. when privately negotiating with someone).

A related policy question to examine is that of how many intrusions users are willing to tolerate in order to improve application performance. For example, applications such as `UMD` and `Media Call` can initiate a preliminary dialogue with a user to verify the social context the user is in. Similarly, an application such as note distribution can employ external feedback mechanisms to verify its successful execution. Active participation by users allows an application to overcome inaccurate and incomplete location information by having users modify their behavior. The result is a system that may provide greater privacy safeguards, but is also more intrusive and less automated than it might be.

In addition to these policy issues there are at least two other infrastructure questions that deserve mention for future work. We listed the addition of multicast to our system as an interesting extension to consider. However, whereas the properties of local-area multicast are fairly well understood, it is unclear what the behavior and scaling properties of reliable Internet multicast are. Consequently, it is unclear what would happen if a large-scale deployment of a multicast-based location infrastructure ever happened.

The second infrastructure problem we mention is that of how two anonymous parties can agree to conditionally reveal information to each other. Although special cases of this problem are easy to solve and may represent the common usage case, it is unclear if there is a general solution that will be satisfactory in all cases.

References

[1] N. Ackroyd and R. Lorimer. *Global Navigation: A GPS User's Guide.* Lloyd's of London Press, 1990.

[2] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *CACM*, 28(10):1030–1044, October 1985.

[3] S. Elrod, G. Hall, R. Costanza, M. Dixon, and J. desRivieres. The responsive environment: Using ubiquitous computing for office comfort and energy management. Technical Report CSL-93-5, Xerox Palo Alto Research Center, 1993.

[4] B.N. Schilit, M.M. Theimer, and B.B. Welch. Customizing mobile application. In *Proceedings USENIX Symposium on Mobile & Location-Independent Computing*, pages 129–138. USENIX Association, August 1993.

[5] M. Spreitzer and M.M. Theimer. Scalable, secure, mobile computing with location information. *CACM*, 36(7):27, July 1993.

[6] ruser manual entry of the SUNOS UNIX manual.

[7] R. Want and A. Hopper. Active badges and personal interactive computing objects. *Transactions on Consumer Electronics*, 38(1), February 1992.

[8] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *Transactions on Information Systems*, 10(1), January 1992.

[9] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–104, September 1992.

[10] M. Weiser. Some computer science problems in ubiquitous computing. *CACM*, 36(7):74–83, July 1993.