Location-Aware Scheduling With Minimal Infrastructure*

John Heidemann
USC/ISI

Dhaval Shah
Noika

Abstract

Mobile computers often benefit from software which adapts to their location. For example, a computer might be backed up when at the office, or the default printer might always be a nearby one. In many existing systems, location-triggered actions are only possible for specific applications or with special infrastructure. This paper describes *lcron*, a system which supports *user-configurable* actions triggered on *change in location* or other events common to mobile computers. Key features of *lcron* are its use of existing clues for location information and mapping low-level location information into user-sensible terms. *Lcron* uses a number of existing sources of location such as network connection and base station ID, allowing it to work without special hardware or GPS receivers. We map sources of low-level information such as IP address and latitude/longitude into user-meaningful *logical* locations. We describe the design, implementation and our experiences with this system.

1 Introduction

Laptop computers are widely used today at home, at the office, and on the road. With such computers, location-aware programs could simplify a number of common tasks. Reminders can be sent to a user when a location is reached ("feed the cat when you get home"). Location-aware queueing is easy ("print this job when I'm at work"). System defaults can be updated when a location is reached ("print to the printer in room 232 when I'm at work"). Replicated or cached data can be refreshed when convenient ("back up my hard disk when I have a good network connection").

One of the reasons location-aware software has seen relatively little use is the perception that it requires special hardware or software. Pioneering work with the Xerox PARCTAB [11] and the ORL Active Badge system [14] took place in the context of office-wide wireless infrared networks which provide location information. These and other systems [17, 2, 4] proposed new frameworks for location-reactive software.

Lack of general-purpose location-aware software is surprising given the wide availability of portable computers and the increasing availability of wireless networking. Although some specialized applications such as mail and printing include some location awareness (perhaps queueing messages or print jobs until a network or server becomes available), these servers are usually custom designed. As a result they may have duplicative or limited detection mechanisms and rarely support user-controlled detection and actions.

This paper examines a system which uses widely available hardware sources of location and existing software interfaces to provide user-configurable, location-triggered actions. We use wired (typically Ethernet) attachment points, but we also can use wireless base-station identifiers or GPS if available. We support easy user configuration by mapping low-level sources of location into user-meaningful tags and by expanding existing mechanisms to specify events in time to also specify events in space. Our model for general events is based on the *cron* and *at* programs widely used in Unix and recently available in Microsoft Windows with packages such as the Norton Program Scheduler and Microsoft System Agents. We believe that there is substantial benefit to adding a notion of location to user-configurable scheduling.

The main contribution of this paper are experiences with what kinds of high-level location information are relevant to mobile users, and how we can generate this information from widely available, low-level sources. In the examples above, concepts such as "work", "home" and "fast network" may not directly correspond to sources of physical geographic location such as GPS. The triggered-event model we describe is actually more general, supporting actions triggered on arbitrary events such as current power status. Finally, we describe our experiences using this tool.

*The authors can be contacted at 4676 Admiralty Way, Marina del Rey, CA, 90292-6695, or by electronic mail to {johnh,dhavalsh}@isi.edu. John Heidemann's work was partially funded under the VINT project (DARPA grant ABT63-96-C-0054) and the LSAM project (DARPA grant J-FBI-95-185).

Although we see location awareness as the primary application for our work, location is actually one instance of more general *event awareness*. In addition to location, laptop users often wish to respond to other system events such as power constraints ("power is low, so shut down the wireless network and don't scan for viruses now"). Other events may arise from the interaction of systems ("remind me to ask 'what's new' when I next meet a particular colleague"). Our system generalizes to support actions triggered from arbitrary events such as these.

In a broader sense, we see this paper as one step in understanding the question raised by the Dataman project at Rutgers: "what if location were a first-class operating-system concept, much as time is now?" [6].

2 Using Location Information

To understand the requirements of *lcron*, this section considers what tasks might be simplified if they were triggered based on location and other events. From these tasks we generalize what kinds of location information are important to users.

In the introduction we identified a number of user tasks that would benefit from location information. We believe that many of the tasks which would benefit from location awareness require a much more refined definition of location than simple physical location as provided by GPS. Table 1 breaks these examples into several classes. Although some tasks depend on physical location, many tasks depend on hardware or some external computing capability. Other tasks should occur when another user (or other moving object) happens to be present. Finally, across each of these categories, a task could depend on a specific or a generic resource.

Some examples make these dimensions clearer. For a generic physical location, one might want to be reminded to drop off a shirt when one is near the next dry cleaners, where any cleaners is acceptable. Later one might want to be reminded to pick that shirt up when near that particular cleaners.

In today's mobile computing environment, location aware tasks are often hardware or device specific. Printing is one example. Network access to replicate or back up data or send mail are common examples. Again, these examples may be specific (backup my computer when connected by a high-speed network to our company's backup server) or generic (send mail when next connected to the Internet).

Some may argue that ubiquitous wireless networking may eliminate this class of jobs. Why queue mail if you're always connected to the network? Although we believe that connectivity will reduce these needs, cost and especially power constraints may limit use of otherwise available networking.

Lcron is most useful for systems which connect to several different networks, but it is also useful for connection and disconnection to a single network. Detection of network reconnection can be used to trigger events such as sending queued mail or synchronization of a disconnected file system [7].

Finally, although we have focused on location-aware job scheduling, other events map well into the *lcron* model. For example, computers often do periodic housekeeping (defragmenting a disk, checking for viruses, indexing mail); on a possibly power-constrained laptop these tasks are best scheduled when not running on battery power (when "at" the "AC-powered" virtual location). We have recently added power-triggered events and are currently investigating how useful *lcron* is for general events.

3 System Location

The first step in location awareness is identifying sources of location information. This section considers three potential sources of location information: the global positioning satellite (GPS) network, physical network conditions, and reports of base-stations in wireless networks.

GPS receivers provide an obvious source of location information. With recent price reductions (at the time of writing US$100–200 receivers are not uncommon), GPS receivers seem increasingly attractive. GPS receivers work in most open areas, but they have limited use in buildings or other areas of limited reception. Unfortunately, power consumption and antenna space may be inconvenient for some portable computer users. Also, GPS accuracy is both a help and a hindrance. On one hand, GPS locations are too detailed for direct use. How many people know the latitude and longitude of their office? On the other hand, current accuracy of GPS receivers is not sufficient to place a user in the room of a building without additional processing.

For computers which are frequently networked, the wired network infrastructure itself can provide location information. Portable computers may be assigned different IP addresses and routers (or different mobile IP addresses [9]) depending on where they are. By monitoring the network attachment we can determine where the computer is. The opposite of GPS accuracy, networks attachments only vaguely specify physical location (somewhere on a given IP subnet[1]). However, if location-dependent jobs often require the network (for

[1] A reviewer suggested using ethernet bridging messages to narrow this to a specific Ethernet segment.

constraint	uniqueness	example
physical location	specific	remind me to get shirts at the cleaners
physical location	generic	what will the weather be here, tomorrow
device	specific	print this document to the printer with my letterhead
device	generic	send this message when connected to a network
other users	specific	remind me when I see a particular person
other users	generic	send the next meetings agenda to everyone in this room
other events	—	scan for viruses when not running on battery power

Table 1: Several classes of location-aware tasks.

example, computer backup), network attachment may be more relevant than GPS-measured physical location.

Wireless networking too can provide location information. Most wireless protocols support multiple base stations or cells. If these systems know their physical location and can report it to the *lcron*, then both network connectivity and physical location can be determined.

One important implementation issue moderates location detection. First, although one would like to detect location changes exactly, in some circumstances polling is required. Sometimes change-in-location is impossible to detect at the device level, for example with GPS where the data is continuous. In addition, there may be no operating system or driver support to trigger *cron* when a location is changed. (For example, laptops with built-in Ethernet will not generate PCCard eject events, and most Ethernet drivers do not trigger user-programs on loss of carrier.) In these cases we must poll location periodically, possibly using battery unnecessarily. Fortunately it is often possible to trigger *cron* only when location changes. For example, PCCard insertion events often correspond with network changes.

A more difficult issue is that, although these approaches work well at detecting arrival at a new location, the actions that can occur on departure from a location are more limited. Departure can be identified by lack of a network connection, but this time is too late for actions that require the network. For example, the policy of "fetch my mail just before I leave work" cannot be directly implemented. An analgous problem exists when caching data stored on removable disks, suggesting two solutions. We can either require users to inform the operating system of an impending change, or we could use computer-controlled eject mechanisms for PCcards. Since no such hardware exists currently, software approaches are required.

4 Mapping From System to User Location

Although we have identified a number of ways a system will directly measure location, these approaches are often too low-level for typical users. We believe that one key to useful location awareness is a mapping from low- to user-level information, much as the domain name system maps from 32-bit IP addresses to human-friendly hostnames.

As examples of this mapping, consider GPS and network location. GPS receivers report latitude and longitude. Because of measurement accuracy, a user must specify "within 100m of this lat/lon", instead of "at this lat/lon". For many, lat/lon are as difficult to manage as IP addresses, so another level of mapping should allow "at ISI" or "near USC" rather than "within 100m of 33.97988N, 118.43994W". Similar examples apply to network location. Few users remember what network segment or router they use, but "on the network I use at USC" is obvious.

Although high-level mappings simplify location description, it is also important to match location description to the task mix. "At ISI" could mean "when connected to the high-speed ISI network" to do backups, "when connected to any ISI network" to print a document, or "when physically near ISI" to send a reminder message. A flexible mapping mechanism can make these distinctions visible. We believe a larger user-base is required to understand how important these distinctions are.

5 *Lcron* Implementation

This section summarizes several aspects of *lcron* implementation, including how location is specified and how changes are detected.

5.1 Base *cron*

Lcron is implemented as a modification of an existing *cron* implementation and several helper programs. We based *lcron* on Geoff Kuenning's implementation *xcron* [8]. Although other freely available *cron* implementations are more widely used, *xcron* had several features important to us for mobile use. *Xcron* is aware that

computers may be turned off or suspended; it optionally runs jobs scheduled during down-time when the system next starts. *Xcron* also includes integrated support for delayed one-shot jobs ("*at* jobs"), simplifying support for location-aware *at* services. Finally, *xcron* supports both the traditional crontab format (table driven with one column per field) and a newer format without strict columns. The newer format uses context to interpret the time specification, so "1:00" means to run at 1 a.m. daily.

5.2 Specifying location

Easy user configuration of event-triggered actions is a goal of *lcron*. We accomplish this by adding an "event" field to the existing crontab file format. A user's crontab file lists the commands that are to be executed on user's behalf at specified times on specified dates. The new field specifies that events should be executed when a particular location is reached or a particular event occurs. Actions can be triggered either periodically when at a location, or only when that location is first reached.

A sample crontab with location-triggered jobs appears in Figure 1. The optional location field begins with the "@" sign. In this example, we fetch mail 3 times a day when at home but every 20 minutes when at work and when we first connect to the work network; these entries have both time and location specifications. Other events are triggered only when we first arrive at location. The "m" flag and lack of a time specification indicates this behavior. An example is setting the default printer to vary depending on where we are. Finally, when we connect to the work network and are powered we run a program to back up the portable computer. (We have implemented and regularly use all but the ability to "and" together multiple events.)

We have also modified *xcron*'s *at* program to provide a similar event specification. Users can schedule jobs "at 8pm @home".

5.3 Location sources and detection

Our primary source of location information is network connectivity. We examined GPS receivers as an additional source of information, but most location-aware applications we wanted to schedule depended on network connectivity, so GPS receiver cost, size, and power requirements limited its use. We also map battery power into *lcron* as an example non-location events.

To sense network location we measure what networks are currently configured and map either the gateway host or the network IP address and mask to a user-sensible location (as described in the next section). Currently we sense network attachments with the "netstat -rn" command. Hosts with dynamically assigned IP addresses (with DHCP, for example) may not have stable addresses, so we primarily use gateway addresses to identify location.

We can detect location changes both by *polling* and *triggered notification*. Triggered notification avoids the delay and constant (if low) overhead of polling. We use triggering as our primary means of detecting discrete locations. Over time we have used two ways to detect change in our system (a laptop running the RedHat distribution of Linux). Originally we detected change to network connectivity with a one-line addition to the PCCard- and PPP-configuration scripts that are used with our system. This line informs *cron* of location change after a new network is started. More recently (since RedHat 6.0) we have used the operating system's built-in ability to signal jobs when the network changes.

We also implement polling as a secondary mechanism to detect location changes. To implement polling, a non-location-specific *cron* job periodically checks the system's location, noticing and acting on any changes. The polling interval can be selected to trade off responsiveness and overhead; by default we poll every 10 minutes. Polling is required for events which are continuous in value, such as GPS data and battery power. Polling is disabled if continuous events are not needed and triggering is possible.

We recently added support to detect power transitions (between AC and battery power) as an event. Ric Faith and Avery Pennarun's apm daemon triggers a program when power mode changes. We wrote a small stub that maps these events in to *lcron* events. Support for different classes of events like this motivates support for the ability to "and" events together (when powered and network connected).

5.4 Mapping from system to user

In addition to specifying which actions are triggered in response to particular events, users should also be able to specify how location is defined. As discussed in Section 4, system-measured location information is not always appropriate for direct user consumption. *Lcron* therefore employs a mapping function from system to user locations.

We have experimented with two ways to implement this mapping. Originally mapping was done with a short Perl program which filters raw locations into user-sensible ones. We chose to implement mapping as a program rather than through a table to increase flexibility. Networks might map gateway names into locations, while GPS data might be mapped based on physical proximity or more detailed topological understanding.

Since currently these capabilities are not widely used, we have replaced it with a table driven mapping (from

```
m            @net:home set_default_printer home_lp
- 7,13,20:00 @net:home fetchmail
m            @net:work set_default_printer ps11d_d
- :0-59/20   @net:work fetchmail
m            @net:work fetchmail
m       @power:battery enable_hard_disk_spindown
m       @power:ac      disable_hard_disk_spindown
m  @net:work&&power:ac backup_disk
```

Figure 1: A sample crontab specification with location-specific commands. (Unlike standard Unix *cron*, *lcron*'s first field is list of flags, defaulted fields may be omitted, and the @ field specifies location constraints.)

```
128.9.160.7      isi
128.9.128.3      isi-netwave
128.125.187.254  usc
128.9.97.33      home
128.9.32.13      ppp
128.9.176.100    ppp
```

Figure 2: A mapping table from gateways to user-level locations.

gateway to logical name) that is easier to configure. Figure 2 shows a sample mapping and illustrates that in some cases multiple low-level locations may map to the same logical location (ppp, in this case). Since mapping is user-dependent we plan on a simple tool which associates the current system location (based on one or more low-level criteria) with a user location.

6 Experiences using *lcron*

Lcron has been used by a few researchers at ISI since January 1998. Our users have different levels of network connectivity. At the low-end, one user's laptop attaches to a single network. At the other end, one user regularly uses four networks in three different physical locations (ISI, USC, and home). This section briefly describes our experiences developing and using *lcron*.

6.1 User environment

In Section 2 we described and classified a number of location-dependent tasks. We have considered *lcron* for four classes of day-to-day tasks:

- File replication and e-mail transfer
- Configuration of system environment
- Alarm service
- User-interface teleporting [10]

We currently use *lcron* for first three tasks on a daily basis. We have also experimented with user-interface teleporting (automatically bringing up a copy of running applications on a nearby display).

None of these applications are new: file-replication [7, 5], automatic system configuration [3, 16, 12], location-specific alarms [16, 12], and teleporting [10] have been experimented with before. The advantage of *lcron* is that now these applications can be easily deployed by users without the need for any extensive supporting infrastructure.

A common and effective use of *lcron* is for file replication and e-mail transfer. Since e-mail is often timely, it is helpful to immediately send or fetch queued messages upon connection. In addition, periodic e-mail retrieval is easily configurable with *lcron*. User-customization is helpful here; the importance of timely e-mail delivery can be weighed against battery constraints (when connected by a wireless network) and tolls (when connected from home via a metered service). This level of configuration would be difficult or impossible with the simpler retrieval models of most typical e-mail packages.

File replication poses similar problems. Automatic file backup is important to a safe environment. *Lcron* triggers this backup periodically when connected to a fast network.

For both e-mail and file backup we had previously employed customized systems of polling. To avoid draining laptop power we previously polled for the laptop from a server machine. Although functional, this system was not easy to change and required configuration on multiple machines. Replacing this system with *lcron* simplified configuration and concentrated it on the laptop under user control. As a result of these simplifications we automated cases that before were not considered important enough to justify the effort (for example, occational but automatic retreival of mail from secondary mailboxes), or were not feasible (immediately fetching mail on connection).

Examples of location-dependent system configuration are selecting a default printer and telephone dialing. We

wrote a small program which changes the definition of a particular printer entry when the laptop arrives at a new location. Users can thus set their default printer to the printer called "nearest" to print to a location-dependent nearby printer. Similarly different locations have different context for telephone dialing (area codes, handling of extensions, and telephone interface). Our telephone auto-dialer supports these options; *lcron* allows us to automatically select between them as we do for printers. Other location-dependent tasks similar to these can be completed automatically using *lcron*.

A third example application we experimented with is a location-based notification service. Users can record "reminder messages" which are sent as e-mail when a given location or time is reached. In other cases delayed actions have proven useful (for example, downloading a file when next connected to the net).

We also experimented with teleporting in *lcron*. First done in the VNC system [10], teleporting is the idea that a user's existing applications should move transparently to the nearest, most capable display. We have experimented both with VNC teleporting and a weaker form where new sessions of a set of applications are automatically begun on a nearby display when a network connection is made. Both have been useful, although automatic teleporting is not always desirable (for example, when the laptop is only briefly connected to send and receive mail).

From our experiments with *lcron* the primary benefit is the automation of day-to-day tasks that were difficult or not warranted beforehand. In some cases of e-mail retrieval *lcron* replaced manual requests. In others it replaced older, less capable and more complicated retrieval mechanisms based on application-specific polling. Finally, the simplicity of *lcron* configuration supported additional uses. Prior arrangements required configuration on multiple machines to avoid polling from the power-constrained laptop; *lcron* simplified configuration by allowing all configuration to take place on the laptop.

6.2 Development environment

Lcron was developed under SunOS and several versions of RedHat Linux. Based on *xcron*, it inherits that system's portability. Our primary platform has been loptops running various versions of RedHat Linux. Other than small OS changes to collect location formation are specific to RedHat Linux, the core of *lcron* is portable to any Unix platform.

In early versions of RedHat Linux, small changes to the base operating system's network configuration scripts are required to allow *lcron* to avoid polling. We made 4 lines of modification to RedHat Linux's PCCard and PPP scripts (no other OS changes were required). These are not necessary with more recent versions (since RedHat 6.0).

We found one hardware limitation: we detect network configuration based on PCCard insertion and removal. One laptop included a built-in Ethernet adaptor that lacked these events. Polling the network for carrier can work around this problem; another approach would be to modify the network driver to allow an application to detect changes in the Ethernet carrier.

Our experiences with *lcron* development suggest that it should be easily portable to other versions of Unix. Periodic polling (perhaps at 5 minute intervals) can be used to detect network change, but we expect that direct detection with small changes will often be a possible optimization.

7 Related work

Lcron builds on three areas of related work. First, groups such as the Dataman project, Xerox PARC, and and the Olivetti and Oracle Research Lab have looked at general ways location awareness changes system behavior. Second, a number of groups have looked at how to modify actions to consider location information. Finally, several special purpose systems have developed custom approaches to watch for location changes.

Our work was inspired by the Dataman project's question of "how would system software change if location were a first-class operating-system concept" [6]. The Dataman project has looked at how mobility affects network transport (mobile IP) and multicast (geographic messaging). We apply their proposition in a very real sense by considering the use of location in existing approaches to task scheduling.

Xerox PARC's work in ubiquitous computing [16] through systems such as the PARCTAB [15] pioneered location-aware computing. Their work in context-aware computing described systems similar in function to *lcron* [12, 11]. *Lcron* builds upon this work by describing how multiple, commonly available of location information (such as network connections and GPS information) can be mapped to user-relevant locations.

Schilit, Adams, and Want classify context-aware computing along two axes (see Figure 2 from [11]) based on whether the task at hand is performing information retrieval or command execution and whether it is done manually or automatically. By this classification *lcron* is automatic (not manual) and supports commands (not information), thus falling into the "context-triggered actions" quadrant. However, we have shown how context-triggered actions in *lcron* allows us to implement contextual commands (for example, printing to the

	manual	**automatic**
information	proximate selection/contextual info	auto contextual reconfig
command	contextual commands	contextual-triggered actions

Table 2: Schilit, Adams, and Want's classification of context-aware computing (from [11]).

nearest printer) and automatic contextual reconfiguration (for example, by automatically replicating data), thus providing some support for some kinds of context-aware applications which are not directly supported. We believe that the final quadrant (manual information retrieval) is best solved with other work such as location-aware web-browsers [1, 13].

Location dependent information such as nearest restaurant, availability of space in the nearest parking lot or local weather information can best be retrieved on-demand by location-aware web-browsers. On the contrary *lcron* mainly targets those location dependent tasks whose timely execution on location change is important for proper functioning of the system. However *lcron* could also be used for automatic information retrieval by scheduling an information retrieval task to be executed at different locations. *Lcron* thus gives users explicit control over the nature and contents of location-dependent information to be retrieved in contrast to the limited set of information available to the users of location-aware web browser.

8 Future work

There are several directions for future work. Our primary sources of location come from network attachments; additional experience with other sources would be helpful. Exposing the cells of a cellular network might be attractive since that information is already available. Wider use of GPS information would also be useful.

Improvements in mapping arbitrary events into *lcron* would also be helpful. We are currently developing and experimenting with mapping changes in power consumption to events; more experience here would be helpful.

Currently *lcron* focuses on one variable: what happens as the system changes state by moving around or losing power. Some location-aware applications have more sophisticated requirements. For example, reminders could be triggered when two people are co-located assumes multiple moving objects, or actions triggered when at a location *and* AC-powered. Work in active badges suggests that these kinds of interactions can be provided by periodically broadcasting presence information to the local area. Refinement of this idea in *lcron* remains future work.

Lcron currently uses a table-driven approach to specify event triggers. This tabular approach is easy for users but can limit specification flexibility. For example, it's easy to take actions at particular times if connected, but not easy to schedule a job at a connection and then at regular intervals after that time. Whether the limitations of this fairly rigid structure is a problem in many situations remains to be seen. We work around this problem with hourly polling through a helper program. Possibly a better solution is to specify events triggers through a programming language (as we map from system to user locations) and to use a front-end to construct program statements for simple cases.

9 Conclusions

We have described *lcron*, a system supporting user-configurable actions in response to location- and event-triggers. It has two key features: first, it uses existing sources of information, such as wired and wireless networking information, to determine location. Second, its approach at mapping these low-level sources of location and other events to user-sensible terms. Our experiences with *lcron* suggest that it achieves its goal of simplifying user-triggered actions in response to location changes and other events, and that this simplification allows much broader automation of location-triggered actions than alternatives.

Acknowledgments

The authors thank Geoff Kuenning development of *xcron*, support for integrating *lcron* and discussions of *lcron*-related issues. We would also like to thank Ramesh Govindan, Joseph Bannister and the anonymous reviewers for careful readings and suggestions to improve this paper.

Software Availability

We plan to make the software developed as part of this paper publicly available. Please contact the authors for current status.

References

[1] Arup Acharya, B. R. Badrinath, Tomasz Imielinski, and Julio C. Navas. A WWW-based location-dependent information service for mobile clients. At `http://www.cs.rutgers.edu/~navas/dataman/papers/loc_dep_mosaic/Overview%.html`, July 1995.

[2] H.P.W. Beadle, G.Q. Maguire, Jr., and M.T. Smith. Using location and environment awareness in mobile communications. In *Proceedings of the EEE/IEEE International Conference on Information, Communications, and Signal Processing*, pages 1781–1785, Singapore, September 1997. IEEE.

[3] Mic Bowman, Larry L. Peterson, and Andrey Yeatts. Univers: An attribute-based name server. *Software—Practice and Experience*, 20(4):403–424, April 1990.

[4] P. J. Brown, J. D. Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications Magazine*, 4(5):58–64, October 1997.

[5] John S. Heidemann, Thomas W. Page, Jr., Richard G. Guy, and Gerald J. Popek. Primarily disconnected operation: Experiences with Ficus. In *Proceedings of the Second Workshop on Management of Replicated Data*, pages 2–5. University of California, Los Angeles, IEEE, November 1992.

[6] Tomasz Imielinski and B. R. Badrinath. Wireless mobile computing: Challenges in data management. *Communications of the ACM*, 37(10):18–28, October 1994.

[7] James J. Kistler and Mahadev Satyanarayanan. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems*, 10(1):3–25, 1992.

[8] Geoff Kuenning. Experiences with an extended cron daemon. Unpublished manuscript, November 1997.

[9] C. Perkins. IP mobility support. RFC 2002, Internet Request For Comments, October 1996.

[10] Tristan Richardson, Frazer Bennett, Glenford Mapp, and Andy Hopper. Teleporting in an X window system environment. *IEEE Personal Communications Magazine*, 1(3):6–12, July 1994.

[11] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90, Santa Cruz, CA, USA, December 1994. IEEE.

[12] Bill Schilit, Marvin Theimer, and Brent Welch. Customizing mobile applications. In *Proceedings of the USENIX Symposium on Mobile and Location-Independent Computing*, pages 129–138, Cambridge, MA, USA, August 1993. USENIX.

[13] Geoffrey M. Voelker and Brian N. Bershad. Mobisaic: An information system for a mobile wireless computing environment. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 185–190, Santa Cruz, CA, USA, December 1995. IEEE.

[14] Roy Want, Andy Hopper, Veronica Falcao, and Jonathon Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[15] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. An overview of the PARCTAB ubiquitous computing experiment. *IEEE Personal Communications Magazine*, 2(6):28–43, December 1995.

[16] Mark Weiser. Some computer science problems in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, July 1993.

[17] Girish Welling and B. R. Badrinath. A framework for environment aware applications. In *Proceedings of the 17th International Conference on Distributed Computing Systems*, pages 384–391, Baltimore, Maryland, May 1997. IEEE.