Experiences with User-configurable, Location-aware Scheduling*

USC TR 98-675

John Heidemann   Dhaval Shah

April 23, 1998

Abstract

Mobile computers often benefit from software which adapts to its 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. We describe the design, implementation and our experiences with this system, focusing on use of existing clues for computer location (such as network connection) and mapping of low-level location into user-sensible terms.

1 Introduction

Today many computer users find a single laptop computer more convenient than two computers at home and at work. For laptop users, docking stations provide a convenient way to adapt their computers to heavy or wired computing (and hence location-specific) hardware. Surprisingly, though, today there has been very little experience with software which reacts to its location.

Specialized applications such as mail and printing often include some location awareness, perhaps queueing messages or print jobs until a network or server becomes available. These servers are usually custom designed, though, and may have duplicative or limited detection mechanisms and rarely support user-controlled detection and actions.

Some researchers have examined at systems to relax these constraints. The Xerox PARC TAB supported a customizable context-aware system [9], as does the ORL Active Badge system [12]. Although influential in exploring these ideas, these systems have seen relatively little widespread deployment, possibly because of the overhead of building and maintaining the required infrastructure. A system which uses existing infrastructure for location detection would be much easier to deploy.

This paper examines lcron, a system which addresses each of these points. Lcron allows software configuration in response to location changes and other kinds of events. It supports easy user configuration based on the existing cron model widely used in Unix and recently available in Microsoft Windows with packages such as the Norton Program Scheduler and Microsoft System Agents. In this sense, lcron is an example of an answer to the question raised by the Dataman project at Rutgers: “what if location were a first-class operating-system concept, much as time is now?” [4]. Finally it takes advantage of intrinsic sources of location information in existing wired and wireless networks. We believe that there is substantial benefit to adding a notion of location to user-configurable scheduling.

Location- and event-aware scheduling simplifies 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”).

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 constrains (“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.

The main contribution of this paper is a description of how to determine the kinds of location relevant to mobile users from widely available sources. In the examples above, “work”, “home” and “fast network” may not directly correspond to sources of 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.

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 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 [5].

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 are currently investigating how general an event model is useful for lcron.

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 [7]) depending on where they are. By monitoring the network attachment we can determine where the computer is. The opposite of GPS accuracy, networks attachment only vaguely specify physical location (somewhere a given 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.

However, if location-dependent jobs often require the network (for 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 system, 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. There may be no driver support to trigger cron when a location is changed. 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 correspond with network changes.

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 [6]. 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 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 cron tab file format. A user’s cron tab 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 cron tab with location-specific jobs appears in Figure 1. The optional location field begins with the “@” sign. In this example, we fetch mail 3 times a day at home but every 20 minutes when at work and when we first connect to the work network. We set the default printer to vary depending on where we are; the “m” flag and lack of a time specification indicate that this action is triggered once each time that location is reached. Finally, when we connect to the work network and are powered we run a program to back up the portable computer.

Our implementation allows location specifications to be regular expressions, matching several possible places. We plan to support multiple locations (as in the “when powered at work” example) to allow logical “ands”; we have not yet implemented this facility.

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 have also planned mapping battery power into non-location events but have not yet done so.

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.

We can detect location changes both by polling and triggered notification. To implement polling, a non-location-specific cron job periodically checks the system’s location, noticing and acting 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.

Triggered notification avoids the delay and constant (if low) overhead of polling. We use triggering as our primary means of detecting discrete locations. We detect 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 initiated.

5.4 Mapping from system to user

In addition to what actions are triggered based on events, how location is defined should be user-configurable. As discussed in Section 4, system-measured kinds of location aren’t always appropriate for direct user consumption. Lcron therefore employs a mapping function from system to user locations.

We implement this mapping as a short Perl program which filters raw locations into user-sensible ones. We chose to implement mapping as 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 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. This tool would modify the mapping program, adding stylized code. Since this tool is not yet implemented we currently add mappings by hand. Figure 2 shows an example of current mapping code. Although this example maps only gateway information to user-sensible location, it could also aggregate multiple gateways to a single location or map one gateway to several logical locations (as in the first example). Similar mapping scripts are possible for other kinds of events.
#!/usr/local/bin/perl
open(STDIN, "netstat -rn") || die "netstat failure.\n";
my(@f, %heads);
while (<>) {
    @f = split;
    # parse headers to be portable across different netstats
    if (@f[0] =~ /dest/i) {
        foreach (0..$#f) {
            $heads{lc($f[$_])} = $f[$_];
        }
    }
    next;
}
# The next lines map gateway to multiple logical locations.
print "net:isi\net:isi-wired\n" if ($f[heads('gateway')] eq '128.9.160.7');
print "net:isi\net:isi-wireless\n" if ($f[heads('gateway')] eq '128.9.128.3');
# The remaining map one-to-one.
print "net:usc\n" if ($f[heads('gateway')] eq '128.125.187.254');
print "net:home\n" if ($f[heads('gateway')] eq '128.9.97.33');
print "net:ppp\n" if ($f[heads('gateway')] eq '128.9.32.13');
}

Figure 2: A sample location mapping program. After initial processing, the network gateway field is mapped to a user-sensible 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 is either connected or disconnected 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
- A weak form of user-interface teleporting

We currently use lcron for first three tasks on a daily bases. We have experimented with user-interface teleporting (automatically bringing up a copy of running applications on a nearby display) but some further development is still required.

None of these applications are new: file-replication [5, 3], automatic system configuration [2, 14, 10], location-specific alarms [14, 10], and teleporting [8] have been experimented with before. The advantage of lcron is that now these applications can be casually deployed by users without substantial existing 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.

In both cases of 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 feasible, 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 case that before were not considered important enough to deploy.

Examples of location-dependent system configuration are selecting a default printer and telephone dialing. We wrote a small program which changes the definition 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 replayed when a given location or time is reached. Although we have found this service very useful for time-dependent messages, location-dependent message are currently very limited by our lack of location-awareness in the at program. We are currently working to remove this limitation.

We also experimented with teleporting in lcron. Teleporting is the idea that a user’s existing applications should move transparently to the nearest, most capable display [8]. Although we have not deployed a full teleporting system (such as has been used at ORL), we have experimented with a weaker form where new sessions of a set of applications are automatically begun on a nearby display when a network connection is made. Although maintaining application state would be desirable, there is considerable value in simply having new instances of the correct applications set up.

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 Linux. Based on xrcron, it inherits that system’s portability. Small changes to the base operating system’s network configuration scripts allow lcron to avoid polling. We made 4 lines of modification to RedHat Linux’s PCCard and PPP scripts.

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 block until change in 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 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” [4]. 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 [14] through systems such as the PARC TAB [13] pioneered location-aware computing. Their work in context-aware computing described systems similar in function to lcron [10, 9]. 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 [9]) 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 allow us to implement automatic and 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, 11].

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 basis 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 a 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 location-aware web-browser users.

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 triggered when two people are co-located assumes multiple moving objects. 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 example 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. A key feature of lcron is that it uses network information common to the existing wired and wireless infrastructure to approximate location and that low-level sources of location and events are mapped to user-sensible terms through a configurable process. Our experiences with lcron suggest that it achieves its aim of making these actions easy to use, and that easy user-configuration allows much broader automation of location-triggered actions than more static system.

Acknowledgments

The authors thank Geoff Kuenning for use of the xcron source code, discussions of lcron-related issues, and a careful review of this paper. We would also like to thank Ramesh Govindan and Joseph Bannister for careful readings of the 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


