The ParcTab Ubiquitous Computing Experiment

Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold,
Karin Petersen, David Goldberg, John R. Ellis and Mark Weiser

The PARCTAB system integrates a palm-sized mobile computer into an office network. This project serves as a preliminary testbed for Ubiquitous Computing, a philosophy originating at Xerox PARC that aims to enrich our computing environment by emphasizing context sensitivity, casual interaction and the spatial arrangement of computers. This paper describes the Ubiquitous Computing philosophy, the PARCTAB system, user-interface issues for small devices, and our experience developing and testing a variety of mobile applications.

1 INTRODUCTION

For the past 30 years the operating speed and component density of digital electronics has steadily increased, while the price of components has steadily decreased. Today, designers of consumer goods are incorporating digital electronics into more and more of their products. If these trends continue, as we expect they will, many everyday items will soon include some form of computer.

Although computers are becoming ever more common in appliances such as VCRs, microwave ovens, and personal digital assistants, they remain largely isolated from one another and from more powerful desktop and laptop machines. We believe that in the future many computers will provide more valuable services in combination than they can in isolation. Ideally, many kinds of specialized machines will work together via networks to let users access and control information, computation and their physical and electronic environments.

In the Computer Science Laboratory (CSL) at Xerox PARC we have established a number of research projects to explore this vision, which we call Ubiquitous Computing. This paper presents the results of the PARCTAB project, an experiment intended to clarify the design and application issues involved in constructing a mobile computing system within an office building. The PARCTAB system provides a useful testbed for some of the ideas of the Ubiquitous Computing philosophy, which is described briefly in the next section. The system is based on palm-sized wireless PARCTAB computers (known generically as "tabs") and an infrared communication system that links them to each other and to desktop computers through a local area network (LAN). Although technological and funding limitations forced us to make numerous compromises in designing the PARCTAB hardware, nevertheless the system, as described in Section 3, meets most of our design goals. Likewise the small size

[1] This work was supported by Xerox and ARPA under contract DABT63-91-C-0027. Portions of systems described here may be patented or patent pending.

and low resolution of the PARCTAB displays requires an innovative user interface design to allow efficient text entry and option selection. Our solutions are presented in Section 4.

A community of about 41 people at Xerox PARC take part in the system's operation and in PARCTAB application development, which are covered in some detail in Sections 5 and 6. To date, we have developed and tested more than two dozen PARCTAB applications that allow users to access information on the network, to communicate through paging and e-mail, to collaborate on shared drawings and texts, and even to monitor and control office appliances. Descriptions of the various PARCTAB applications as well as data on users' experiences with them are given in Sections 7 and 8, respectively.

By designing, constructing, and evaluating a fully operational mobile computing system and developing applications that exploit its unique capabilities, we have gained some insight into the practical benefits and real-world problems of such systems. In the paper's final section, we collect these lessons and present some of the many intriguing ideas that the PARCTAB project has spawned for future work in Ubiquitous Computing.

2 UBIQUITOUS COMPUTING

As inexpensive computers add limited intelligence to a wider variety of everyday products, a new model of computing becomes possible.

2.1 The Ubiquitous Computing Philosophy

This new technology aims for the flexibility of a far simpler and more ubiquitous technology: printed text. Depending on the need, print can be large or small, trivial or profound, verbose or concise. But though print surrounds us in myriad forms, it does not dominate our thoughts the way computers do today. We do not need to log on to road signs to use them or turn away from our colleagues to jot notes on a pad of paper. Similarly, ubiquitous computers would demand less of our concentration than present commercial computer interfaces that require users to sit still and focus their attention. Yet through casual interaction they would provide us with more information and all the advantages of an intelligently orchestrated and highly connected computer system.

Creating such an intuitive and distributed system requires two key ingredients: communication and context. Communication allows system components to share information about their status, the user and the environment—that is, the context in which they are operating. Specifically, context information might include such elements as:

- The name of the user's current location;
- The identities of the user and of other people nearby;
- The identities and status of the nearby printers, workstations, Liveboards, coffee machines, etc.;
- Physical parameters such as time, temperature, light level and weather conditions.

The combination of mobile computing and context communications can be a powerful one [40; 30; 28; 32; 33; 31]. Consider, for example, an employee who wants to show a set of figures to his manager. As he approaches her office, a quick glance at his tab confirms that the boss is in and alone. In the midst of their conversation, the employee uses the tab

to locate the data file on the network server and to request a printout. The system sends his request by default to the closest printer and notifies him when the job is finished. Many more examples of the Ubiquitous Computing philosophy are presented in Mark Weiser's article "The Computer of the 21st Century" [39].

2.2 A Ubiquitous Computing Infrastructure

Attaining the goals of Ubiquitous Computing will require a highly sophisticated infrastructure. In the ideal system, a real-time tracking mechanism will derive the locations and operational status of many system components and will use that context to deliver messages more intelligently. Users will be able to choose from among a variety of devices to gain mobile, high-bandwidth access to data and computational resources anywhere on the network. These devices will be intuitive, attractive and responsive. They will automatically adapt their behavior to suit the current user and context.

Although one can speculate about the design of a future system, unfortunately the components needed to build such an infrastructure have yet to be invented. Current processors and microcontrollers are slow and power-hungry compared to their likely descendants 10 years from now. We reasoned that we could bridge this technology gap by constructing an operational system that resembles an optimal design. Despite the inevitable compromise of some engineering characteristics, we could then use the system to assess the advantages and disadvantages of Ubiquitous Computing as if we had glimpsed into the future.

It is impossible to predict the range of device forms and capabilities that will be available a decade from now. We therefore based our device research on size, a factor that is likely to continue to divide computers into functional categories. A useful metaphor that highlights our approach is to consider the traditional English units of length: the inch, foot and yard. These units evolved because they represent three significantly different scales of use from a human perspective.

- Devices on the inch scale, in general, can be easily attached to clothing or carried in a pocket or hand.
- Foot-sized devices can also be carried, though probably not all the time. We expect that office workers will use foot-sized computers similar to the way that they use notebooks today. Some notebooks are personal and are carried to a particular place for a particular purpose. Other notepads are scattered throughout the work environment and can be used by anyone for any purpose.
- In the future office there will be computers with yard-sized screens. These will probably be stationary devices analogous to whiteboards today.

2.3 Ubiquitous Computing Experiments at PARC

Researchers at PARC have built computer systems at the three scales described above [41]:

inch	PARCTAB, a palm-sized computer;
foot	PARCPAD, an electronic notepad;
yard	Liveboard, an electronic whiteboard.

These experimental devices use different mechanisms for communication and computation within the building's distributed system. The Liveboard is not mobile and connects directly to an Ethernet. Our mobile devices extend battery life by using low-power communication technologies: infrared (IR) signalling for the PARCTAB and near-field radio [8] for the PARCPAD. We have also investigated how operating system design can reduce power consumption [43] and this is well suited to mobile computers. The PARCPAD and Liveboard are described elsewhere by Kantarjiev [17; 12] and Elrod [9].

Our goals for the PARCTAB project were:

- To design a mobile hardware device, the PARCTAB, that enables personal communication;
- To design an architecture that supports mobile computing;
- To construct context-sensitive applications that exploit this architecture;
- To test the entire system in an office community of about 41 people acting as both users and developers of mobile applications.

3 PARCTAB SYSTEM DESIGN

We set several design goals for the PARCTAB hardware. It had to be physically attractive to users, compatible with the network, and capable of modifying its behavior in response to the current context. We believed that in order to fulfill these goals the PARCTAB had to be small, light and aesthetically pleasing enough that users would accept it as an everyday accessory. It needed reliable wireless connectivity with our existing networks and a tracking mechanism capable of detecting its location down to the resolution of a room. It had to run on batteries for at least one day without recharging.

We also believed that the PARCTAB's user interface had to let people make casual use of the device, even if they had only one free hand. The screen had to be able to display graphics as well as text. We wanted users to be able to make marks and selections using electronic ink, so the screen needed touch sensitivity with a resolution at least equal that of the display. Furthermore, the cost of the hardware and the network infrastructure had to be within reasonable limits so that we could deploy the system for lab-wide use.

Cost was not the only limitation on our design options. Some factors were also limited by available technology, such as the device's communication bandwidth, display resolution, processor performance and battery capacity.

3.1 PARCTAB Mobile Hardware

We carefully weighed the limitations and requirements above when making the engineering decisions that shaped the final appearance (Figure 1) and functionality of the PARCTAB hardware. One primary trade-off balanced weight, processor performance, and communications bandwidth against battery life. We also had to strike a compromise between screen resolution and the device's size, cost and processor speed.

3.1.1 Packaging

We believed an ergonomic package would be essential if people were to carry and use the tab regularly. We thus enclosed the PARCTAB in a production-quality custom plastic case with a removable belt clip. The tab is about half the size of current commercial personal digital assistants (PDAs), at 10.5cm x 7.8cm x 2.4cm (4.1in x 3.0in x 0.95in). It weighs 215g (7.5oz). We designed the tab so that users could choose either one-handed use with buttons or two-handed use with a stylus. Because the package is symmetric, the tab can be used in either hand—an important feature for left-handers who wish to use the stylus. To convert from right- to left-handed use, the user executes a setup command that rotates the display and touch-screen coordinates by 180 degrees.

3.1.2 Display and Control Characteristics

We found that commercially available touch-sensitive displays provided adequate resolution for our needs. We chose a 6.2cm x 4.5cm (2.4in x 1.8in) LCD display with a resolution of 128 x 64 monochrome pixels.

The PARCTAB is most easily operated with two hands: one to hold the tab, the other to use a passive stylus or a finger to touch the screen. But since office workers often seem to have their hands full, we designed the tab so that three mechanical buttons fall beneath the fingers of the same hand that holds the tab (see Figure 1), allowing one-handed use. The device also includes a piezo-electric speaker so that applications can generate audio feedback.

Figure 1: The PARCTAB mobile hardware

3.1.3 Power Management

Power is the overriding concern that drives most of the design decisions of most small electronic devices, and the PARCTAB is no exception. With more power, there could be faster communication over longer distances, higher-resolution displays, and faster processors. But existing battery technology places stringent limits on the power available for such small components.

We found the prismatic (rectangular) Nicad cell to be the most suitable battery technology given our size, weight and performance goals. Four cells were sufficient to provide a rechargeable power source for the tab while meeting all the packaging requirements. We designed the core of the device around a 12MHz, 8-bit microcontroller (87C524), an Intel 8051 derivative, for two reasons. First, its on-board EPROM, RAM and I/O ports ensured a compact design. But equally important, this processor can be programmed to enter a low-power mode. The PARCTAB takes advantage of this mode when idle in order to extend battery life. The display, touch screen, additional RAM and the communication electronics can also be powered down by the microcontroller.

During normal operation a tab consumes 27mA at 5V. In low-power mode it consumes less than 30μA. We considered nominal use to be 10 minutes per hour, eight hours per working day. In operation, however, we found that the one-day use requirement was easily met. In fact, using a battery storage-capacity of 360mAh, the typical tab need only be charged once per week. A smaller battery may suffice, in which case we estimate that the PARCTAB could be reduced to one-third of its current weight and volume if it were produced today by a commercial electronics company instead of a research lab. We anticipate that within a few years the functions of the PARCTAB probably could be put into a watch.

3.2 PARCTAB Communication

Limited space and power constrained our choice of a wireless communication technology to just two options: radio and infrared (IR). We chose 850nm IR to exploit the small, inexpensive IR components that were commercially available. These offered low power consumption at the modest communication speeds of 9600 and 19200 baud. Because IR signals are contained by the walls of a room, this technology also made it easier to design a cellular system. Moreover, IR communication is unregulated. A radio link would have required more space, higher power equipment and potentially government operating licenses.

We decided that a cellular system [5] would best handle the competition for bandwidth that inevitably would arise in a building-wide system supporting many users. By creating small, room-sized communication cells (nanocells), we could minimize the communication distance from the hub to the mobile user, reducing power needs concomitantly. Since the radiated signal would be blocked by walls, messages would be more secure than if they were broadcast widely. Users are also less likely to interfere with one another's signals in a cellular system, although some situations—such as heavy tab use during a break in a large meeting—can still place large loads on the IR transceivers. Finally, small cells enable the system to pin down a user's location to the resolution of a room.

The tab infrared network [1; 27] thus consists of nanocells defined by the walls of a room surrounding an IR transceiver. Large open rooms and hallways may also support nanocells if transceivers are carefully placed out of communication range of each other. Transceivers connect to a LAN through the RS-232 ports of nearby workstations.

3.2.1 Transceiver Design

A transceiver serves as a communication hub for any PARCTABs located in its particular cell. Typically its communication radius is about 20 feet—less if limited by the walls of an office. The transceiver hardware performs numerous functions in addition to transmission and reception, including:

- Coding and decoding infrared packets;
- Buffering data;
- Executing link-level protocol checks (e.g., format or checksum);
- Providing a serial interface to a workstation's RS-232 port;
- Indicating visually its communication status.

We designed the transceiver conservatively to ensure reliable communication. For transmission, two dozen IR emitters are placed at 15 degree intervals on a circular printed circuit board. For reception, two detectors provide a total viewing angle of 360 degrees (Figure 2). The transceiver is designed to be attached to a ceiling, preferably in the middle of a room as this usually gives an unobscured communication path over the required area. But since transceivers and PARCTABs can sense infrared light reflected from surfaces, it is not necessary that there be a line of sight between the two for them to communicate. Thus a single transceiver usually covers a room completely.

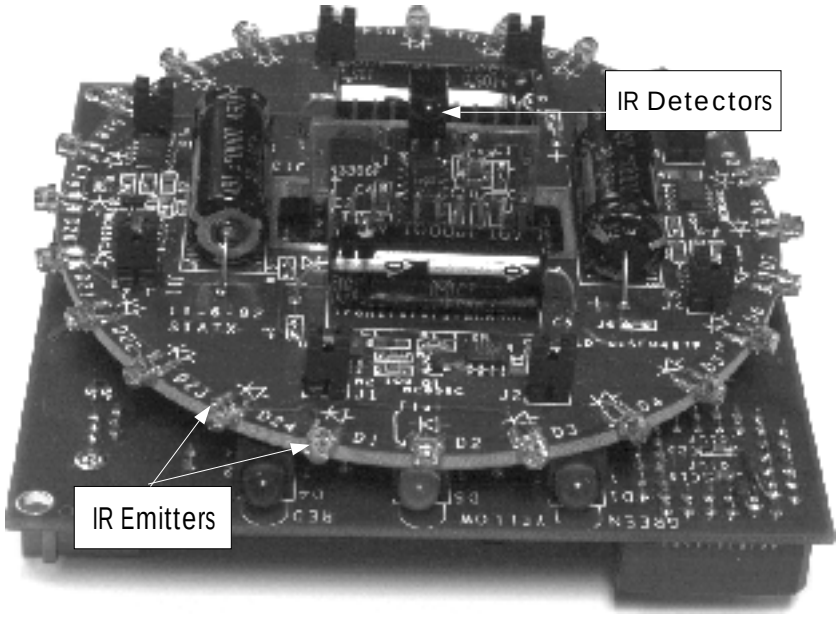


Figure 2: The PARCTAB transceiver

3.2.2 Local Area Network Interface

We found the approach of extending an existing LAN to provide wireless nanocellular communication very attractive for a number of reasons. The additional cost is small because the LAN wiring already exists. Most offices in our building are equipped with at least one workstation that has a spare RS-232 port. We thus had to string only a small amount of additional phone cable to connect ceiling-mounted transceivers to our UNIX workstations and, through them, the ethernet. And since well established communication mechanisms already exist between workstations in commercial distributed systems, we did not have to reinvent that infrastructure. Transceivers could be attached to networks of other platforms, such as the PC or Macintosh, in much the same way.

3.2.3 Transmission Control

The decision to use infrared communication prompts a further design issue: how to enable many PARCTABs to share the medium? Conventional IR detectors have difficulty tuning narrow frequency ranges, ruling out the possibility of using frequency-division multiplexing to divide the bandwidth into several subchannels. We thus chose a simple digital packet-contention scheme that shares the medium using time-division multiplexing.

In this scheme, all data is bundled into packets formed by the baseband modulation of an IR carrier into a sequence of pulses. The pulses are uniform— all have a duration of 4μS—but the gaps between them are not. The variable duration of the silence between pulses encodes the data bits. The durations of the gap encoding a logic 1, logic 0, packet-start synchronization, and data-byte synchronization are all unique and may be decoded using a simple algorithm. By defining data as the absence of a signal, this technique minimizes power consumption, since the infrared carrier is switched off for most of a transmission.

The link-layer packets are divided into several fields, as shown in Figure 3 below. The packet type field is always sent at 9600 baud, and a subfield of the packet type defines the speed at which the rest of the packet will be transmitted. This permits variable speed transmission and allows future high-speed systems to remain backward-compatible. The present system transmits packets at 9600 and 19200 baud.

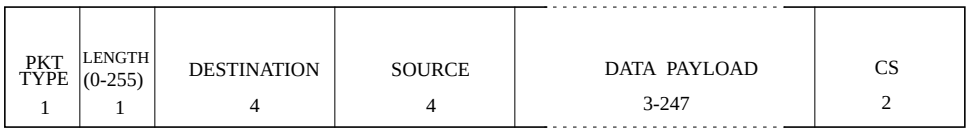


Figure 3: Format of the data fields for a link-layer IR packet (lengths in bytes).

The second field contains the length of the packet. Packets vary in length from 14 bytes for most uplink packets to a maximum of 256 bytes for a downlink packet. Next follow unique 4-byte addresses of the destination and source devices, up to 247 bytes of payload data and finally a 2-byte checksum.

We assumed that communications traffic inside a cell would normally be low since applications are driven by user-generated events, such as button clicks. We thus expected a screen update to be followed by a relatively long silence while the user made the next se-

lection. Because we also assumed that small packets generated under lightly loaded conditions would be delivered promptly, we chose to use a symmetric non-persistent carrier-sense multiple-access (CSMA) protocol to provide access to the IR channel. This protocol simply uses carrier sense and a random-exponential backoff whenever the channel is busy. It does not wait for a packet currently occupying the channel to complete before entering a new backoff period [34].

3.2.4 Reliability and Interference

The PARCTAB system cannot detect packet collisions because any IR transmission creates such a powerful signal that it saturates the local receiver, making it impossible to detect a packet sent simultaneously by another device. Mobile hardware can avoid losing link-layer packets by setting a bit in the packet type field that requests an acknowledgment. When a transceiver sees the request bit set, it immediately transmits a reply back to the sender. In a multiple-access network this type of acknowledgment is quite reliable, since the fact that the request was received implies that there was no contention and therefore the acknowledgment should also not encounter contention [36]. A PARCTAB sets the request bit for some types of tab packets—user events, for example—and then, if no acknowledgment arrives, resends the packet a fixed number of times until finally generating an audible alarm to the user. In principle, downlink packets sent from a transceiver to a PARCTAB could also use this mechanism. Instead, as described in Section 7, we ensure downlink reliability at a higher level of protocol.

When a PARCTAB is in view of two rooms—when in a hallway, for instance, with doors opening into two cells—both cell transceivers might acknowledge event packets simultaneously, corrupting the acknowledgment signal at the PARCTAB. To avoid this problem transceivers that are close enough to interfere with each other are given different network addresses and only acknowledge packets addressed to them, although they still transfer all the packets that they receive to the LAN. Whenever a PARCTAB enters a new cell the system notices events that it produces (e.g., beacons or button clicks) and instructs the tab to use a new transceiver address.

4 USER-INTERFACE DESIGN FOR PALM-SIZED COMPUTERS

As we developed applications for the PARCTAB, it became clear that a traditional user interface designed for the 640 x 480-pixel color display of a typical PC or workstation would not work well on the PARCTAB's 128 x 64-pixel monochrome display [26; 42]. Indeed, the PARCTAB's tiny screen, offering less than half the area of most PDA displays, forced us to devise innovative ways to select, display and enter information in a very limited space. As advancing technology produces higher resolution displays that can pack more information onto a small screen, some of the problems we faced will undoubtedly disappear. But text and symbols can shrink only so much before they become too small to read. Also, as displays increase in resolution, new devices will probably get commensurately smaller. Many of the user-interface solutions we describe below will thus remain relevant.

4.1 Buttons vs. Touch Screen

Since the PARCTAB is well suited for casual, spur-of-the-moment use, we did not want to compel users to free both hands to operate the device. The user interface thus had to allow

users to control applications with the device's three buttons, its touch screen or a combination of both. This requirement complicated the interface design because a user selecting an item on the screen with the buttons alone must then be presented with an intermediate screen allowing her to invoke an operation on that item. Consequently, application developers must decide whether to require two-handed use (of both stylus and buttons) or whether to increase the number of screens in their program so that all the functions can be accessed via the buttons.

We found one convention that seems to solve this problem best, and developers incorporated it into several tab applications. It works as follows: on clicking the middle push-button, a menu of commands pops-up. The top and bottom buttons then move the cursor up and down, while a second click of the middle button selects the command on which the cursor currently rests. On screens that display scrolls or lists of text, the top and bottom buttons scroll the list up or down, respectively. If menus are designed intelligently, then users must usually just click the middle button twice to execute the most common action. Two-handed users can press an on-screen button to pop up the menu and can then point with the stylus to select an item directly.

We have also settled on a preferred interface style for using the push-buttons and the stylus to navigate a tree data-structure. The operator uses the stylus to navigate down through the hierarchy one screen at a time and clicks the middle button to navigate upward. This method works efficiently because users descending the hierarchy must at each level make a choice, a task performed simplest with the stylus. Ascending the tree, on the other hand, requires a user to repeat the same operation over and over, a task well suited to repeated push-button action.

4.2 Spurious Event Prevention

Because the PARCTAB applications often run elsewhere on the network there can be modest delays between a button click or screen touch and the update of the screen 5. The delay between event and response can occasionally cause errant behavior in the user interface. Consider the case in which a menu contains a button icon that selects another screen with its own button icon in a similar position. A user tapping the first button with the stylus might create multiple pen-events, either by unintentionally bouncing the pen on the touch surface or by impatiently tapping the button twice. The initial event will trigger a transition to the next screen, but the latter events could then cause additional, unwanted selections. We solved this problem by adding a field called an epoch to the event packet structure. Every time an application transmits a screen change, it also increments the epoch number in the PARCTAB. Any events that were in the application input queue with a previous epoch number can now be discarded, thus preventing any spurious transitions.

4.3 Text Display

We anticipated that it might be difficult to read text on the PARCTAB because its small display can show only eight lines of 21 (6 x 8-pixel) characters. In practice, this proved not to be a problem, as our popular e-mail application exemplifies. Word-wrap and hyphenation algorithms can often fit three or four words across the screen. The 8-line display is also small enough to update quickly despite the limited communication bandwidth.

Users scroll through text either by clicking the top or bottom push-buttons or by touching the upper or lower half of the display. The experience is similar to reading a newspaper column through a small window that can be moved up or down by the flick of a pen. Although this is relatively efficient, it is nevertheless often useful to filter text information before it is displayed. For example, the PARCTAB e-mail application replaces lengthy message headers with a compressed form that includes only the sender and an abbreviated form of the subject field.

4.4 Text Entry

We experimented with two methods of text entry: graphic, onscreen keyboards and Unistrokes, a novel approach to handwriting recognition. Unistrokes [14] is similar to Graffiti, a system marketed subsequently by Palm Computing.

4.4.1 Keyboard Entry

An onscreen keyboard requires both an array of graphic keys arranged in typewriter format and an area to display text as it is entered. We have experimented with several layouts. The first presents key icons across lines 2 through 8 of the screen and displays the characters that have been "typed" on line 1, which scrolls left and right as necessary to accommodate messages longer than 21 characters. A delete-last-character function bound to the PARCTAB's top push-button allows easy correction of mistakes. One of the other push-buttons serves as a carriage return that terminates an entry. We found that users could enter about two characters per second using this keyboard layout. Experiments with smaller keyboards show that they lower typing accuracy.

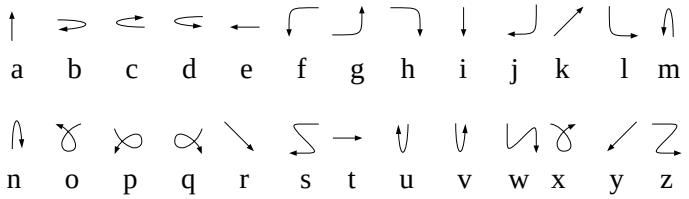


Figure 4: The Unistroke alphabet

4.4.2 Unistrokes

Techniques for handwriting recognition have improved in recent years, and are used on some PDAs for text entry. But they are still far from ideal since they respond differently to the unique writing characteristics of each operator. We have experimented on the PARCTAB with Unistrokes, which depart from the traditional approach in that they require the user to learn a new alphabet—one designed specifically to make handwriting easier to recognize.

For each letter in the English alphabet there is a corresponding Unistroke character which can be drawn in a single pen stroke. The direction of the stroke is significant (Figure 4). To minimize the effort required to learn to write in Unistrokes, all Unistroke characters are either identical to English letters (e.g., L, S and Z) or are based on a characteristic feature of

the corresponding English letter (e.g., the cross of T). We found that most people can learn the Unistroke alphabet in under an hour.

Because Unistroke characters are directional and better differentiated than English letters, they require less processing to recognize reliably. Because the characters are single strokes, users can draw each Unistroke character right on top of the previous one, using the entire screen. Thus the strokes themselves need not appear on the writing surface, but instead the PARCTAB neatly displays the corresponding English characters. Practiced Unistrokers found the simplicity and speed of text entry very attractive.

4.5 Option Selection

The PARCTAB's small screen makes it difficult to present users with a long list of options. We tried a number of different methods.

4.5.1 Text and Icon menus

The PARCTAB screen size places rather severe limits on the number of text or icon options in a menu. Vertically, eight lines of text fit onscreen. Spreading three text buttons per line across the display increases the number of selections to 24. Arranging 16 x 16-pixel icons in an uncluttered format yields about 15 options per screen (see Figure 1). Larger lists require alternative approaches.

4.5.2 Scrolling Lists

Some applications, including Tabmail and Arbitron (Figure 5), present choices in a scrolling list with each option represented by a single line of text. The item on which the cursor rests is highlighted; options that are unavailable because they do not make sense in the current context are crossed out. As users press the top and bottom push-buttons to move the cursor up and down, the list scrolls as necessary to expose more options.

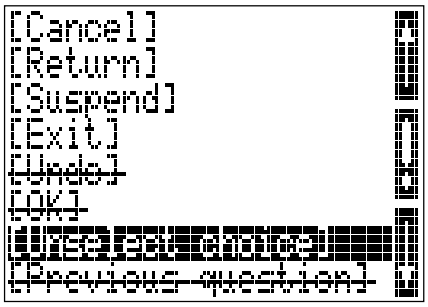


Figure 5: A screen from the PARCTAB Arbitron application

We considered using the "proportional" scroll bars common in PC user interfaces to allow fast touch-screen navigation of large ordered lists on the PARCTAB. This scheme takes the horizontal or vertical position of a screen touch as a percentage of the total screen dimension, then moves the cursor to a similar position in the long list of options. Unfortunately, we found that the resolution of the touch-screen restricts accurate selection to lists smaller

than the maximum number of pen positions that can be resolved. The PARCTAB can resolve 128 horizontal positions per line.

We also chose not to use this type of interface element because it demands continuous display feedback for each movement of the pen. Typically the feedback would be generated remotely, rather than by the tab itself, because the application generating the feedback is running elsewhere on the network. The contention that would result between pen events and continuous display updates would make poor use of the communication bandwidth.

4.5.3 Elision and Incremental Searches

We used the PARCTAB to evaluate the efficiency of two somewhat more sophisticated methods for selecting one item (such as a name or word) from a large ordered list (such as a directory or dictionary): elision and incremental searching. Elision is based on k-ary search techniques. The system divides the list into 15 portions of roughly equal size and displays the first item in each section, followed by an ellipsis (Figure 6). The display ends with the last item in the list.

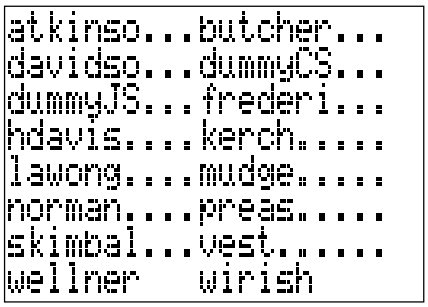


Figure 6: A screen from the PARCTAB locator application

The user selects the target item if it is displayed. Otherwise, selecting any ellipsis redraws the screen to show an expansion of the selected region of the list into 13 smaller portions as before. (The very first and last items in the complete list are always displayed so that users can navigate back to other regions.) The user continues "zooming in" on a particular region until the target item appears.

Elision is reasonably efficient. Because the PARCTAB screen can display 16 abbreviated words with ellipses between them, users need make at most $\log_{16}$ N selections to reach any item, where N is the size of the list. To select one item among one million, for example, requires no more than six selections. The mean word length in the American Heritage online dictionary, containing 84433 words, is 8.9 characters. A user typing a word from this dictionary on a graphic keyboard must thus make 8.9 selections, on average. Elision, by comparison, can bring up any word in this dictionary with just four selections.

Incremental search techniques, implemented in the PARCTAB dictionary application, can do nearly as well. Here the user types the first few letters of the item. With each letter entered, the application narrows the list of possible matches and displays the closest eight. We found that this method identified the desired word after 4.3 characters on average—thus 5.3 selections, since one more tap is needed to choose the correct match from the eight choices.

PARCTAB applications have made successful use of both elision and incremental searches. We observed advantages and disadvantages for each. Elision is the more general method, since it performs well even when the ordered list has no special properties. It also usually requires fewer selections–especially if it is refined so that the system adjusts the size of the subsections to fall between guide words that have been frequently selected. Many PARCTAB users object to elision, however, because it demands hard thinking to pick the appropriate ellipsis.

5 PARCTAB SYSTEM ARCHITECTURE

A multilayer system architecture integrates the PARCTAB hardware into the PARC office network so that network applications can easily control and respond to mobile devices based on the devices' current context. Although the PARCTABs themselves behave more like terminals than independent computers, they do execute local functions in response to remote procedure calls. PARCTABs also generate events that are then forwarded by transceivers and the infrared gateways that manages them to processes called tab agents, which run on network machines. The agents keep track of the mobile tabs and link them to workstation-based applications. PARCTAB applications are generally event-driven, much like X11 or Macintosh programs. Figure 7 illustrates relationships among PARCTABs, transceivers, gateway and agent processes, and applications.

Developers can link into their applications a code library that hides the details of PARCTAB tracking, message routing, and error recovery. Of course, any application can obtain a tab's current location as needed so that the program can modify its behavior appropriately. We developed the PARCTAB system in the Unix programming environment (SunOS 4.3.1) running on SparcStation 2 connected by an ethernet. Communication between Unix processes is achieved using Sun RPC.

5.1 PARCTAB Processing Capabilities

One perennial issue in distributed systems design is the question of partitioning: how much computation should be performed by the mobile devices, and how much by larger computers fixed to the network. One alternative is to execute much of an application's interface locally in the mobile client, similar to the way America Online, Compuserve and Prodigy put most of their user interface onto users' PCs. At the extreme defined by PDAs, a tab might even run whole applications and communicate only occasionally with the network.

Although this approach might reduce the load on the IR channel, it requires a fast processor and much memory. But using today's technology, the power requirements of state-of-the-art CPUs cannot be satisfied by conventional batteries of reasonable size and weight. With a 12MHz processor and 128Kb of memory, the PARCTAB is roughly equivalent in computational power to a PC from the early 1980s. To date, we have thus used tabs primarily as input/output devices that rely on workstation-based applications for most computation. In this model the mobile computer becomes a display device similar to a more conventional graphics terminal. Recently, however, we have experimented with a few applications that execute solely in the tab: taking notes using Unistrokes, for example, and browsing files downloaded from the network.

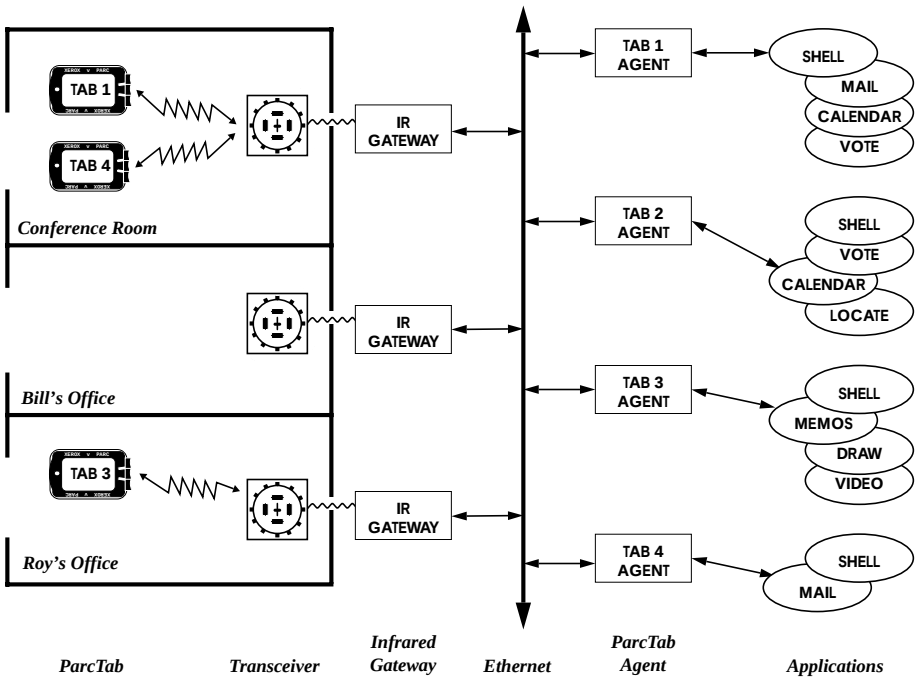


Figure 7: The PARCTAB system architecture

5.1.1 Tab Remote Procedure Call Mechanism

A simple communication mechanism called a tab remote procedure call (T-RPC) allows applications to control various PARCTAB resources, such as the display, touch screen, local memory and tone generator, while remaining oblivious to a tab's location and any underlying communication errors. This mechanism has been incorporated into a library of procedures available to application designers. When an application makes a call into the library, the library assembles a request packet in a format defined by a request/reply protocol.

PAYLOAD TYPE	SEQUENCE NUMBER	FUNCTION			MORE FUNCTIONS	END
		CODE	LENGTH	PARAMETERS		
1	1	1	1	0 - 242		1

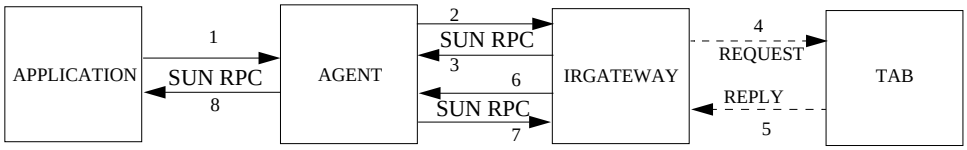
Figure 8: Format of IR packet data payload as used by the request/reply protocol (lengths in bytes)

The request/reply protocol is contained in the data payload of the link-layer packet (Figure 8). The tab supports a set of about 30 function codes, several of which can be combined into a single packet. For efficiency multiple function-requests can be batched into a single packet under program control. A few examples of PARCTAB functions are: display_text, display_bits, generate_tones, set_epoch and wake_up.

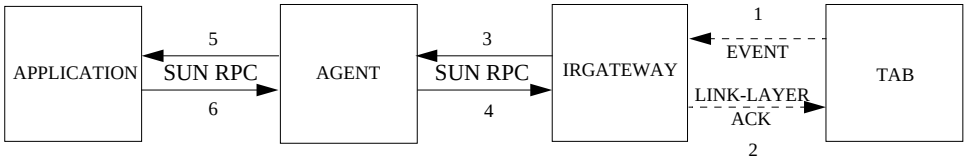
An application delivers the request packet to a tab's agent process, which forwards iet in turn to the tab. The application then waits for a reply. When the PARCTAB finishes executing the request, it returns a reply packet to the application containing an indication of its success and any appropriate results.

Sometimes a request or reply packet will be lost, or the system will be temporarily unable to determine the location of a tab. In that case, the agent will automatically time-out the reply and will retry the request at intervals defined by an exponential back-off algorithm. The back-off algorithm takes into account whether the tab is detected by the network or not, and whether the tab is free or busy executing another T-RPC request.

Only when a request is matched up with a corresponding reply will the the application continue. The agent increments the sequence number for each new request to ensure that retried packets do not inadvertently execute a request twice. The agent likewise discards duplicate replies that result from retries or detection by multiple transceivers. Figure 9 shows the complete path taken by a T-RPC call made from an application to a tab and back again.



T-RPC (Application to Tab Communication)



Event Notification (Tab to Application Communication)

Figure 9: The path taken by a T-RPC call made from an application to a tab.

5.1.2 PARCTAB Events

When a PARCTAB user presses a button or touches the screen, the device transmits an event signal. The PARCTAB may also generate certain events autonomously, such as a low-battery alert and a beacon. The beacon is a signal transmitted every 30 seconds, even when the device is idling in low-power mode, that allows the system to continue to monitor a PARCTAB's location when it is not active. A similar system has been used to locate people using Active Badges [38; 11; 15]. The power cost of waking up a tab every 30 seconds to emit one packet is not high and, in fact, we also designed the tab to listen for a moment after sending a beacon. If a wake-up request is received in this period the PARCTAB will power-up completely. The system can thus deliver priority messages to the device even when it is not in use.

The packet format used to signal PARCTAB events is similar to that used in the request/reply

mechanism. The payload type field distinguishes events, requests and replies. In event packets, the function code is replaced by the appropriate event code.

5.2 Infrared Gateway

The IR-gateway process controls one or more infrared transceivers connected to the serial ports of a workstation. The gateway receives IR packets forwarded by transceivers and delivers them to tab agents. In the reverse direction, the IR-gateway receives packets from an agent over a local-area network, encodes them for IR transmission and delivers them to the appropriate serial port. The transceiver then broadcasts the packets over the IR medium to any tabs within its cell. These packets are coded according to the request/reply protocol described in Section 5.1.1.

The IR-gateway uses a name service to determine which agent should receive each packet. The gateway looks up the packet's source addresses (i.e., the tab's unique address) in the name-service directory to obtain the network address of the corresponding agent. Each gateway process maintains a long-lived cache of agent network address so that it rarely needs to use the name service.

The gateway also appends a return address and a location identifier to every packet it sends to an agent. The location identifier is a short textual description (e.g., "35-2232") of the location of the transceiver that received the packet. Context-sensitive applications can use the identifier in combination with centralized location databases and services to customize their behavior.

In addition to its main functions, the IR-gateway performs configuration, error-reporting, and error-recovery functions. Gateway processes also handle the flow control that matches low-speed infrared communications with the high-speed local area network.

5.3 Tab Agent

For each PARCTAB there is exactly one agent process, which acts like a switchboard to connect applications with tabs via IR-gateways. An agent performs four functions:

- It receives requests from applications to deliver packets to the mobile PARCTAB that it serves;
- In the reverse direction, it forwards messages (along with location identifiers) from its tab to the current application;
- It provides an authoritative source of tab location information for context-savvy applications;
- Finally, it manages application communication channels.

Since the agent is an intermediary on all messages, it has the most complete information on the location of its tab. Even if the PARCTAB moves to a new cell, its agent will soon receive a beacon signal and update the tab's location accordingly. Whenever the tab's location or status changes, the agent notifies a centralized location service [29] of the tab's last known location and its status: "interactive" if it is being used, "idle" if it is transmitting beacons but no other events, and "missing" if the tab is out of sight.

An agent also manages which application is allowed access to its tab at a particular moment. Because the PARCTAB screen is so small, each application takes over the entire display. Although the tab may run many network applications over time, only one "current application" can receive events from the tab and send it messages at a given moment. In our system, a tab's agent interacts with a special application called the "shell" (see Section 5.4) to decide which application is current.

PARCTAB users can currently choose between two shells: the standard shell described in the next section and an alternative described in Section 6.3.

5.4 Shell and Application Control

The shell is a distinguished application that provides a user interface for launching or resuming other tab applications.

A tab agent launches a shell when the agent is initialized, and if the shell exits, the agent automatically restarts it. When it is current, the shell displays an application menu like that shown in Figure 10 and waits for the user to select an application. If the user chooses to launch a program, then the shell creates a new Unix process, registers it with the tab's agent, and finally instructs the agent to switch to the new application. Whenever a user suspends or exits a PARCTAB application, the agent makes the shell the current application.

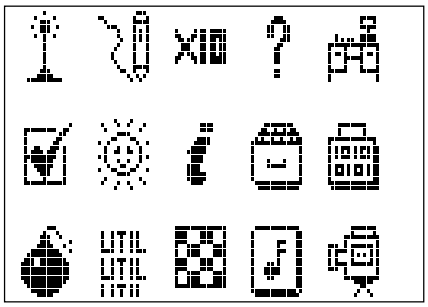


Figure 10: The top-level screen presented by the default Shell

The shell and other applications communicate with an agent through the AppControl interface. This interface offers four procedures: register, suspend, resume, and quit. When an application invokes the 'suspend' or 'quit' command, the agent switches control back to the shell. When a user chooses to resume a suspended application or to switch to a newly registered process, the shell calls the 'resume' procedure. If an application locks up in some way, a PARCTAB user can transmit a special "agent escape" event that forces the agent to suspend the current application and switch back to the shell.

The shell interface is based on user-customized screens. A screen contains active regions (called buttons) and graphic labels, both of which may be represented by text and bitmaps. Buttons invoke built-in actions: jumping to another screen, starting or resuming an application, playing a tune over the PARCTAB speaker, etc.

When the shell is started it loads a user's tabrc initialization file, or a standard configuration file if that is not present. The contents of the tabrc file define the buttons, bitmaps, text and active areas that the shell draws on the PARCTAB's top-level screen. The shell also

looks for a user's tabrc-personal file and uses that to extend the menus described by the tabrc file.

The grammar for the files consists of two parts, as shown below. The first section defines the screen structure displayed on the tab. The second section contains a list of actions, such as running a certain program, that the shell performs when it starts up. In this format, the star ("*****") indicates items that can occur zero or more times; unstarred items occur exactly once.

Tabrc	→	Part*****
Part	→	`(Initialize` Action*****`)`
	→	`(Screens` Screen*****`)`
Screen	→	`(`*label*: Widget*****`)`
Widget	→	`(Text` *text x y invert*`)`
		`(TextButton` *text x y* Action`)`
		`(Bitmap` *bitmap-file x y*`)`
		`(BitmapButton` *bitmap-file x y* Action`)`
Action	→	`(Screen` *label*`)`
		`(Beep` *duration octave note ...*`)`
		`(Program` *program-args*`)`
		`(Load` *tabrc-file*`)`

5.5 Example of System Operation

To explain how the PARCTAB system operates in practice, consider the following example. A user holding a PARCTAB in Roy's office presses a button. The tab transmits a button event packet and requests an acknowledgment.

A transceiver nearby picks up the signal, transmits an acknowledgment back to the tab, and then forwards the event packet over the serial connection. The IR-gateway process listening to the serial line receives the packet, extracts its source address and looks up the network address for the agent associated with the tab that sent the packet. The gateway stamps the packet with the transceiver's location identifier and its own network address, then sends it off to the agent.

When the agent receives the message, it first verifies that this is not a duplicate of a previous packet. It then forwards the data to whichever application is current. The application decodes the event and triggers a procedure call defined by the application developer.

If, for example, the application wants to update the PARCTAB display, then it calls a tab library function and the transmission process reverses. First, the library procedure packs the application's display data into a T-RPC request packet and sends the request to the appropriate agent. The procedure also blocks the application until the call is completed. Next the agent forwards the packet to whichever IR-gateway sent it a message last.

The IR-gateway encodes the request packet for transmission and sends it over the serial link to a transceiver, which broadcasts the data over the IR medium. When the PARCTAB to which the request is addressed receives the packet, it decodes and executes the functions and then transmits a reply back to the IR-gateway indicating its success. The gateway duly forwards the reply to the correct agent as described above.

6 DEVELOPING SYSTEM AND APPLICATION COMPONENTS

Members of the experimental community have built PARCTAB applications using three different approaches: Modula-3 libraries, Tcl/Tk and the MacTabbit system. Each offered different levels of access to the PARCTAB and its capabilities.

6.1 Modula-3

Modula-3 was a natural choice to build the first PARCTAB applications because it is also the language for the PARCTAB's system software [22]. It had many characteristics that recommend it for both tasks, along with a number of shortcomings.

6.1.1 Modula-3 and System Development

Modula-3 is a relatively new language; it has a number of features that we believe are valuable in building large systems. These include garbage collection, light-weight threads, type safety, and support for modules and object-oriented programming. PARC's earlier successes using Cedar (an ancestor of Modula-3) for systems work influenced our decision. In addition, we hoped that the combination of type safety and object-orientation would result in higher quality, more reusable code.

Modula-3's threads were important for our design because they simplified the architecture of the IR-gateway and agent. Both are long running servers that interact with many clients at the same time. Each client has its own dedicated thread: if one client doesn't return promptly from a remote procedure call, others are not adversely affected. Building a non-blocking server without threads would require either changing the remote procedure call (RPC) mechanism to make it asynchronous or abandoning RPC in favor of some lower-level communication mechanism.

6.1.2 Modula-3 and Application Writers

Modula-3 also facilitated the development of reusable libraries for tab application writers. For example, we developed an object-based widget library to handle the user interface. The object-oriented approach meant that each addition could build on previous work.

To simplify development work, we also built a PARCTAB simulator in Modula-3. This program uses an X-window to mirror the PARCTAB display and mouse events to simulate the PARCTAB pen and buttons. In many cases developers prefer the simulator to the mobile hardware for program testing.

Although Modula-3 as a language met our needs well, the implementation we used had a number of deficiencies. Modula-3 is still a young language, and so the programming environment lacked certain tools, especially for debugging. In particular, there was no support for debugging multiple threads: tracking down the deadlocks and race conditions that come with multi-threaded programs was particularly challenging. Modula-3 also produced very large runtime images which occasionally taxed even our 64MB workstations.

To compensate for this shortcoming we built support mechanisms into the tab system software. Each process can write selected information to a log file, and system components have network-accessible interfaces for debugging and control. Programmers can use these interfaces to examine and set parameters, and to restart components. The IR-gateway, for example, has extensive commands for checking the status of the transceiver hardware.

6.2 Code Libraries

We implemented a class-based hierarchy of composable widgets, loosely modeled on the Trestle window toolkit [22], to provide routine components such as iconic and text buttons, scrollbars, bitmaps, text labels, scrollable text areas, and dialog boxes. The PARCTAB's very small screen generally precludes overlapping of widgets, so our widgets do not need to do the clipping required by a conventional window system. This greatly simplified the implementation.

We also built the TabGroup programming interface to support concurrent use of multiple tabs by a single application. A group of tabs could act as a shared whiteboard or notepad, for example, displaying what was drawn on one tab to all the others in the group. With TabGroup, a program can wait for all pending output to be delivered to all its tabs, synchronize on input or other events, and detect tabs that have stopped responding. Using a single process to control a group of tabs with standard interfaces provided by the tab programming library is often easier than running a separate process for each tab and having the processes communicate by application-specific RPC.

6.3 The Tshell and Tcl

Originally the only software available to support developers was the widget library. Developers used it much as they might use a language specific windowing toolkit like Xt [16] to write X-windows applications. As a result, they had to focus on low-level properties of the window system rather than on what they wanted to accomplish. Furthermore, for designers implementing simple user interfaces the turn around time of writing an application in a language like Modula-3 or C was too long. It became apparent that we had to provide fast prototyping capabilities and support the implementation of simple user interfaces at a higher level.

We created the Tshell [25], a PARCTAB-shell extended with a Tcl interpreter and a subset of Tk [24], thus providing both a scripting language that supports remote communication and a windowing toolkit. The choice of Tcl/Tk over other extension languages was based mostly on three reasons:

1. Tcl/Tk is widely used.

2. Tk provides a complete set of building blocks for creating graphical user interfaces. We could quickly select and implement a subset of widgets useful for the PARCTAB's small display size.

3. Tcl/Tk can be embedded into applications so that Tcl interpreters in different applications can exchange commands.

The design of Tab-Tk, the port of Tk to the Tab, focused on maintaining the natural look and feel of the Tk widgets while exploiting the small area of the Tab display as much as possible. We made several key observations and decisions during the port:

- The PARCTAB screen is too small to display multiple windows at the same time. Screen management therefore employs the same "one window at a time" philosophy as other tab applications.

- Because the Tab's screen area is limited, it makes extensive use of menus. They must be intuitive to use and have good response times.

- PARCTAB size and limited processing capabilities call for simplicity. The current implementation of the Tk toolkit for the Tab therefore provides a core widget set of buttons, labels, menus, text, entries, frames and toplevel-windows. We left out such features as the packer and canvas, a full-fledged drawing widget.

Tcl/Tk provides a high level language to rapidly prototype the graphical user interfaces for PARCTAB applications and a communication platform that allows programs to exchange commands with Tcl interpreters in other applications. In a matter of three months, members of our community created a wide range of new applications, including a context-based reminder system, a remote controller for a presentation manager, a pan/tilt camera controller, a remote editor for leaving notes on a workstation.

6.4 The MacTabbit system

Our colleagues at the Rank Xerox Research Centre (RXRC - formally called "EuroPARC") used a different approach to develop applications for the PARCTAB . The Apple Macintosh is the computer of choice at RXRC, and tab users there wanted to access Macintosh applications. MacTabbit does this by arranging for the PARCTAB to control a small portion of the Mac screen. It echoes updates in this region to the tab and sends pen and button events on the tab to the Macintosh.

Using graphical application builders on the Mac such a Hypercard, users can quickly prototype specialized Tab interfaces on the Mac Screen. When the interface works correctly, it takes but a few seconds to move it on to a tab. Furthermore, once the connection has been made to Hypercard, a user may select from a variety of Hypercard-based applications.

MacTabbit has provided an excellent prototyping environment for people unfamiliar with the conventional tab programming environment, and it has drawn in developers who would not normally have become involved. System performance was also good given the small tab screen. An extension of the MacTabbit mechanism caches commonly used image fragments in the tab, thus reducing bandwidth requirements and further improving performance. RXRC has used the MacTabbit mechanism to prototype many tab applications such as Forget-me-not (see Section 7.1), an automatic diary and reminder system, and a media-space controller (see Section 7.2.2).

7 A CLASSIFICATION OF PARCTAB APPLICATIONS

Three characteristics differentiate a tab and the kinds of applications that it supports from traditional personal computers:

1. Portability: very small form factor, low-weight

2. Communication: low-latency interaction between users and system

3. Context-sensitive operation

Mobile Application Categories
Information Access
Communication
Computer Supported Collaboration
Remote Control
Local data/applications

Table 1: Mobile Application Categories

Our system represents context by a combination of factors: location, the presence of other mobile devices, and the presence of people. Context also includes time, nearby non-mobile machines and the state of the network file system. Traditional computer systems have had access to much of this information, but they have typically not made much use of it. Context can be used to adapt the user interface, criteria for extracting and presenting data, system configuration, and even the effects of commands. Although context may be used to present the options most likely to be chosen, a well-designed system would also allow a user access to the full range of choices on request.

Some of the applications we describe are available on small commercial PDAs whose size is comparable to that of a tab, but no PDA has the network infrastructure to support the full range of applications supported by the PARCTAB. The combination of a wireless network and the use of context make this system unique. A summary of the application categories we have experimented with is given in Table 1 and described in some detail in the following sections.

7.1 Information Access

Access to information stored in our computer networks has become central to the way we conduct our work. The PARCTAB IR network has provided a mechanism to make information access independent of location. (Note that although all stored information is accessible from any networked workstation, people tend not to use someone else's machine.)

Each PARCTAB is linked to our local area network and so can retrieve any information available through it or through remote networks connected to it. For example, the commonly used weather program displays the current weather forecast (obtained from the Internet) and the local temperature and wind-speed (obtained from a weather station on the local network). PARCTAB users also have at their fingertips a dictionary, a thesaurus, a Unix file browser and a connection to the World Wide Web. The WWW protocol is a popular way to access information stored all over the Internet. Some care must be taken, however, to adapt the information retrieved to the small PARCTAB screen.

PARCTAB applications have also been integrated with existing desk-top applications. The PARCTAB calendar manager, for example, works with Sun's calendar manager ("cm"), already in use. An update to a user's calendar either on a workstation or on a PARCTAB will enable the data to be viewed on both systems.

The tab location-based file browser shows how context can be used to filter information. Instead of presenting the complete file system hierarchy, it shows only files whose informa-

tion is relevant to the particular room it is in. Such a mechanism can be used to provide a guided tour for a visitor or to provide information that is relevant to a location, such as the booking procedure associated with a conference room.

More complex uses of context can be seen in tools built at RXRC such as Forget-me-not [20; 23; 21; 19; 18]. This application provides a tab user with an automatic biography of their life by remembering for each day details such as: where the person went in the office, whom they met, the documents they edited or printed, and any phone calls that were made or received. The motivation behind this work is to provide an aid to our fallible human memories, a so called memory-prosthesis. The application operates by providing an iconic interface that allows a user to search and filter the biography for a particular event. For example, suppose a forgetful user were trying to find the name of a document that she was editing when Mike came into the room a short while after the seminar last week. The filter would be set up to show documents in use when Mike was around, on the day of the seminar. As we seem to waste a great deal of our lives searching for things we have either misplaced or information we have forgotten, Forget-me-not has the potential to help us work more effectively.

7.2 Communication

Electronic mail has long been a popular communication tool for computer users. Mobile access further enhances e-mail by increasing its availability.

Group meetings often account for a large amount of our work time, and so electronic mail has been an important application for the PARCTAB. Access to e-mail during meetings seems to have satisfied a genuine need.

The PARCTAB e-mail application could be extended to use context to generate filters for displaying messages or notifying users of incoming mail. For example, all messages might be delivered while a user is alone, but only urgent ones would be delivered during a conference. In related work [13] a query language has been used to filter incoming mail.

7.2.1 Locator and Pager Operation

The PARCTAB system inherently provides a locator system, assuming that the person who needs to be found is carrying a PARCTAB. In an office, people can use context to decide whether to disturb a colleague, once they have been located [37]. For example, a person is more likely to welcome interruptions alone in their office than while in a meeting. With the PARCTAB system, a person may be paged unconditionally, or the importance of the page can be assessed in association with the recipient's context, so that the message will be either delivered or delayed until the context is more favorable.

7.2.2 Media Applications

Another RXRC application is the "Communicator", a context-sensitive media-space controller. A description of the original media-space concept is given by Buxton [4] — a video-conferencing mechanism based on an analog-switch controlled by workstations, allowing users to establish video connections to various places in an appropriately wired building. The tab has been used to enhance this facility through an application that will suggest the easiest way to communicate with the person you wish to contact, and then help establish the connection. Knowledge of where the recipient is situated is known to the system because

they are carrying a tab, the calling party only needs to know their name. If a media-space terminal is not available, the application might suggest the best alternative: a phone number, let you know they are actually next door, or offer to send an e-mail note from the tab screen. More recent work at the University of Toronto has taken this work further and combined Ubiquitous Computing with video in a reactive environment [3].

An application that pushes the PARCTAB's communication abilities to their limits is media windowing. An otherwise unused IR channel can transmit one low-resolution frame of slow-scan video in about 1.5 seconds. These images are very grainy because of the coarse resolution of the PARCTAB screen and the limited bandwidth of the link. Nevertheless people are remarkably good at recognizing faces and scenes, and the images are still useful. Future systems with improved screens and higher bandwidth links could provide applications for remote monitoring and mobile communication using sound and video.

7.3 Computer Supported Collaboration

People often gather with a common goal or interest, perhaps at a lecture, or else to arrive at a common decision. Because the PARCTAB is small, it can easily be used in these collaborative situations.

7.3.1 Group Pointing and Annotation

A PARCTAB used as a pointing device operates much like a mouse. However, a PARCTAB can connect to different computers depending on its location.

Many PARCTABs can also connect to the same computer. Consider, for example, the case in which a lecture is presented using a large electronic display such as a Liveboard (see 2.3). Each tab in the audience can control a different pointer on the display. We have built a remote display pointer using the PARCTAB screen as both a relative and absolute positioning tool: the user controls the location and motion of the pointer by moving a finger over the PARCTAB's touch surface[1].

7.3.2 Voting

The PARCTAB can also be used when members of a group wish to arrive at a consensus, perhaps anonymously. Even if anonymity is not important, simultaneous voting can collect data that is unbiased by the voting process. If people vote in sequence, earlier viewpoints inevitably bias later ones.

We have built a voting application called Arbitron for the PARCTAB system. It has proved particularly interesting in the context of presentations. Audience members with PARCTABs vote on the quality and pace of the material being covered by a presenter. The votes are collected anonymously and displayed on the Liveboard. The board is visible to both the audience and the presenter; thus everyone knows whether their colleagues are as bored or entranced as they are. Without the PARCTAB listeners would have to interrupt the presentation to ask the speaker to speed up, slow down, or move to another point.

[1] A tab-based remote pointing and annotation tool was demonstrated as part of the Xerox exhibit at Expo '92 in Seville

7.3.3 Multi-tab Virtual Paper

Tabdraw is a multi-tab application that allows the tab screen to be used as if it were a piece of scrap paper. Each PARCTAB participating in the application owns a piece of virtual paper and can draw on it. The participants also have the option of seeing the drawings of their colleagues by superimposing them on their own work. This scheme ensures that users "own" the line segments they draw; no one else can erase them. As a result, many users can work together in a coordinated fashion without impairing fair participation.

The shared drawing is generally defined by the room that people are in. A group in one room will automatically obtain a separate drawing surface from that in another room. Alternatively, a group might arrange to share a drawing regardless of location.

7.4 Remote Control

Television and stereo system remote-controls have popularized the notion of control at a distance. In fact so many pieces of consumer electronics have such controllers that one can now buy universal remote controls that control many devices at the same time. A PARCTAB can also act as a universal controller. Furthermore, it can command applications that traditionally take their input from a keyboard or a mouse.

Since a tab can display arbitrary data, the controls available to a user can be changed depending on context. (Commercial universal remote controllers, in contrast, tend to need a large array of buttons.) Enabling the remote control application in an office may trigger a tab to provide a control panel that adjusts lighting and temperature, whereas in a conference room the interface might be biased toward presentation tools.

7.4.1 Program Controllers

During our experiments with group drawing and pointing tools it became clear that a PARCTAB has some interesting control possibilities as a drawing interface for a drawing program. It can make additional commands available without cluttering the main screen, and it can also provide a more powerful set of commands than was available in the original program by providing a single button that controls a sequence of low-level drawing primitives. If a program is already intended for remote use and has a network interface, controlling it with a PARCTAB is very easy.

7.4.2 X10 Remote Control

Another Ubiquitous Computing project at Xerox PARC, the Responsive Environment Project [10], has been exploring how environmental control can save energy during the day-to-day operation of a building. The project had created servers that control power outlets through a commercial system called X10 [2].

Because the servers controlling appliances in part of the building being studied by the Responsive Environment project were already connected to the local area network, it was a simple matter to build PARCTAB applications to control them.

7.5 Local Operation

The PARCTAB is near one extreme of a spectrum of possible devices ranging from the remote terminal (devoid of function without its connection to the network) to the standalone

computer (capable of many operations without any communication links). The latest revision of the tab hardware has 128K of on-board memory, so that data and programs can be downloaded through the IR link and executed in a stand-alone mode. Operating the tab in this way frees a user from the IR network, but of course severely limits the tab's functionality.

The storage capacity of a mobile device will probably always be small compared to the expectations of its user. Consequently applications must take care to download only the most relevant information. For example, if a user has unread electronic mail at the end of a work day, the system might transfer the messages to the PARCTAB so that they could be read in transit or at home. (Currently, all downloading of information and programs occurs under the user's control.)

8 EXPERIENCES WITH THE PARCTAB SYSTEM

The PARCTAB system has been in use since March 1993 and now serves a small community of users. We have made a number of useful observations during this period and have begun to understand its successes and failures.

8.1 The Experimental Network at PARC

PARC was a convenient test site for the PARCTAB system because installation was very easy. Before the project began every office already contained a workstation connected by an ethernet. The hallways and common areas also had access to nearby workstations. It was easy to install a communication cell in an office by using velcro to attach a transceiver to the ceiling and then to run phone cable down a wall into a junction box. The junction box usually rests on the floor under a desk and has a power cable, and connects to the RS232 port of the workstation. Typically, the installation takes about 15 minutes.

Some users also installed cells in their homes. They already had ISDN lines, which connect a home ethernet to the office network, and so a transceiver connected to a workstation at home was effectively tied to the PARCTAB infrastructure.

The first PARCTAB system released in March '93 consisted of 20 users and 25 cells. The experience gained in this time enabled a second release in April '94. The latter system was somewhat larger with a community of about 41 users and 50 cells. It included many improvements that enhanced the performance of the communication channel and the tabs' perceived reliability.

For example, the original system relied on a central name-and-maintain service (see Section 5.2) to route packets to tabs; when the service was unavailable the PARCTAB system could not function. The new release has a distributed name service that uses a network multicast mechanism to determine the address of system components.

We discovered in the first release there were problems caused by high utilization of the infrared network. High loads cause three problems: infrared packets are more likely to be corrupted; transmit buffers in the transceiver overflow, causing packets to be dropped; and the corrupted and dropped packets caused more retransmissions, increasing the load. The high load exposed bugs in the system design and implementation such as race conditions and badly-tuned retransmission policies.

To improve user's confidence in the system, we had to increase its reliability and availability. This involved not only fixing bugs but also mundane improvements such as a low-

battery indicator for tabs. System components also needed mechanisms for self monitoring. All the PARCTAB system processes now have control panels designed to provide information in the event of a failure. We have also put new mechanisms in place to monitor and maintain the IR-gateway and the agent processes.

8.2 Infrared Interference

The PARCTAB could not be used effectively in several rooms in our building because of IR noise due to fluorescent lamps controlled by electronic ballasts. This is a waste of a unique form of communication bandwidth. Unfortunately, electronic ballasts are slowly replacing the older magnetic ballasts because they are more energy efficient. We found a considerable variation in interference levels from lamps made by different manufacturers. Some produce acceptable levels of IR, and it would be useful if lamp manufacturers were required to adhere to a maximum limit for IR emissions.

Positioning of a room transceiver is also important. Installers should avoid direct sunlight, that can change position throughout a day (and during the year), and proximity to fluorescent lamps and to obstructions on the ceiling. Transceivers in adjacent cells should be positioned carefully so that their signals do not pass through doorway or interior windows and cause interference.

8.3 Usage Data Measured from the PARCTAB System

Part of the benefit of building a real system has been the opportunity to study how a versatile personal information-terminal might be used in advance of a commercial system. We studied the 1994 release of the tab system for three months to determine its use characteristics. The participants all consented to automatic logging of system events.

We began recording two weeks after system deployment so that users could familiarize themselves with the PARCTAB. To limit the data to a manageable quantity, we logged only the following events: Interactive, Switch, Idle, and Missing[2]. Interactive occurs when a user powers up a tab, Switch occurs when a user switches to a new application, Idle is generated when a tab has not been used for 4 minutes, and Missing is a timeout event generated by the system when the infrared network cannot detect a particular tab. Each event was recorded along with a timestamp and cell location. In addition, there were two questionnaires given out to our users, one at the outset of the tab use study and one at the close. This provided contextual information, and information to interpret the logging data.

8.3.1 Which Applications were Popular?

The switch events can be used to determine the relative popularity of the various PARCTAB applications. Figure 11 shows the percentage of invocations accounted for by each application. Four were distinctly more popular than the rest: e-mail, weather, file browser, and the loader. Possible implications of these results are discussed in Section 9.

8.3.2 How Long were Applications in Use?

Another way of looking at application popularity is to consider how long each application was in use (see Figure 12). It should be noted that the total application interaction time is

[2] During the 3 month study some system processes died and were restarted causing some events not to be logged. This results in minor, but conservative, inaccuracies in the reported statistics.

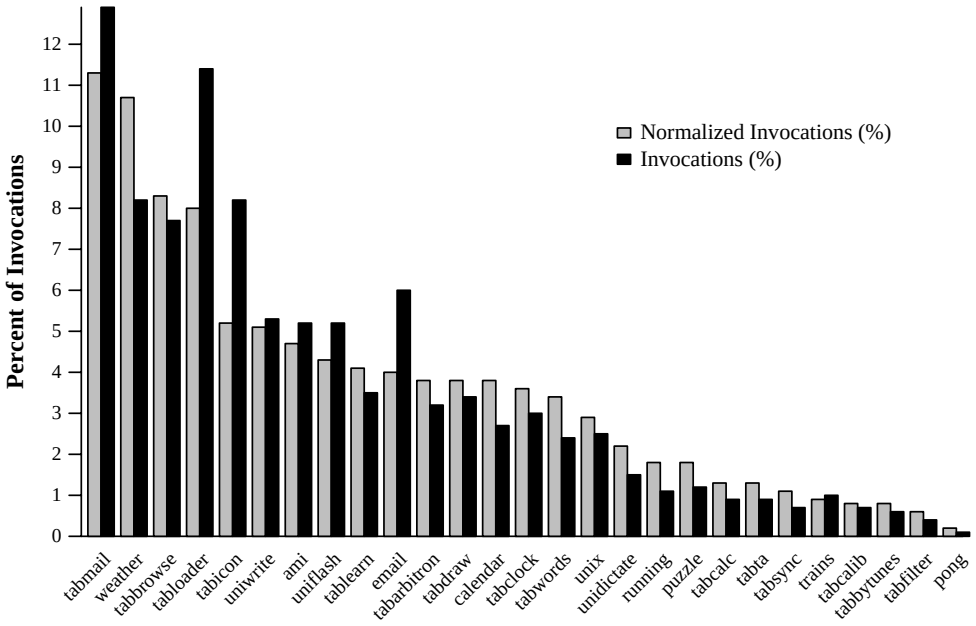


Figure 11: Histogram showing the number of invocations for each application (not including the shell or tshell) expressed as a percentage of the total invocations of these applications during the test period. Normalized results only count one invocation per day per user to remove distortions that might arise when users experiment with an application several times during a brief period. Applications that might normally be invoked several times a day suffer under this measure.

4871 minutes over 3 months (13 weeks) for 41 users. This amounts to only 119 minutes/user or about 1.8 minutes/user/day (65 days, excluding weekends). From our logs the total number of application switches for all tabs throughout the study was 2996 and therefore the average interaction time was about 97 seconds.

The application popularity ranking is somewhat different from Figure 11. The e-mailer, unistroke test and learn program, unistroke notetaker, file browser, and the loader are the most long-lived applications. The weather program falls to 8th place (perhaps because it only imparts a small amount of information at any one time). Meanwhile the note-taker moves up to 3rd place – not surprising, as taking notes is by its nature a time-consuming activity. It is interesting to observe that reading e-mail, browsing system files, and loading data turn out to be the most used in both measurements.

This use pattern differed from the participants own expectations of use. Although they expected to read e-mail, (four of the participants did not use e-mail on the tab at all, due to incompatible mail systems), over half commented that they expected to use the tab primarily as a calendar. It is also worth noting that according to user reports the e-mail program was used to read e-mail much more than to send e-mail using Unistrokes. The Unistroke test and learn programs appear in the ranking even though they are typically not activated very often;

users may spend a block of time running them when first acquiring the skill.

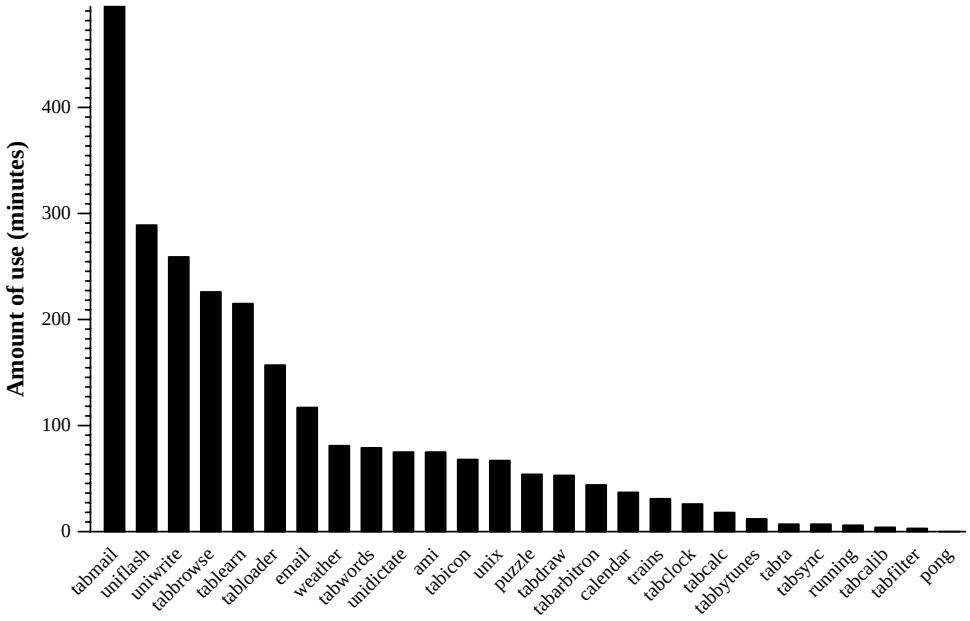


Figure 12: Histogram showing the total interaction time by users for each application in the tab system during the 3 month test period (not-including the shell, 1273 minutes, and the tshell, 1081 minutes).

Graph 13 shows the percentage of application interactions that last less than a given time. We have removed interactions of less than 10 seconds because users often turn a tab on and then off immediately to confirm that it is working normally. From this graph we can see that 50% of interactions last less than 100 seconds (1.7 mins), 75% less than 230 seconds (3.8 mins) and 90% less than 500 seconds (8.3 mins). This supports our notion of the tab as a device for "casual" interactions.

Figure 14 shows what fraction of users had their tabs turned on for various total periods of time. The study group can be roughly divided into three user types. 7% (3 people) used the tab for 360-480 minutes during the test (6.4 minutes/day). 15% (6 people) used it for 144-360 minutes (3.9 minutes/day) and 78% (32 people) used it for less than 144 minutes in total (1.1 minutes/day). The average use time for the majority was very small, implying their interactions were generally very brief.

8.3.3 Who Used the PARCTAB, How Long and Where?

Figure 15 shows interaction time for each user, subdivided according to location: in their own office (black); in a common area such as a conference room, tea area or seminar room (grey); or in a hall or another person's office (white). Only 3 people used a tab primarily (for more than 50% of their total interaction time) in somebody else's office. Approximately 61% (25 people) of our community used the tab primarily in their own rooms, and 27% (11

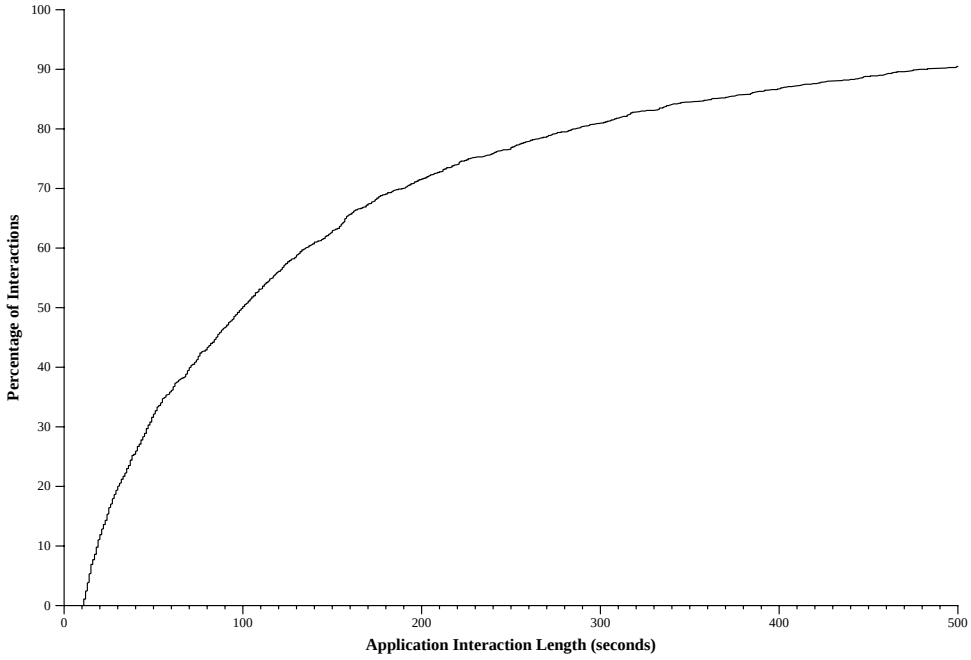


Figure 13: Graph showing the percentage of application interactions that were under a given time during the test period.

people) used it primarily in a common area. Interestingly enough, for each pattern of use the preference was quite clear.

By pooling the results of Figure 15 we can determine that people used tabs in their own offices 57% of the time, in a common area 32% of the time , and in another office 11% of the time (see Figure 16). 7% of own-office interactions are in the presence of other tabs. 90% of common area interactions and 85% of other-office interactions are also in this category.

The multiple-user applications, group drawing and remote pointing, were not available for the duration of the use study. Group applications like this would have generated a much higher network-load in the common areas, but are likely uses of a ubiquitous mobile device.

Figure 15 shows that there is not a typical use pattern among the study group. Our questionnaires showed that there were as many different expectations of the tab system as there were participants in the study. For example, researchers developing applications on the tab that expected to use the tab a great deal did not necessarily have the largest interactions times, even though they had to use the tab for their daily work. In contrast, some researchers who did not expect to use the tab found that visitor demonstrations of the device added significantly to their total usage time.

These results are important for overall system design because multiple tabs interacting in the same area have a strong impact on the available bandwidth. The PARCTAB system needs to be able to handle a usage pattern in which at least 42% of all interactions occur with multiple tabs present.

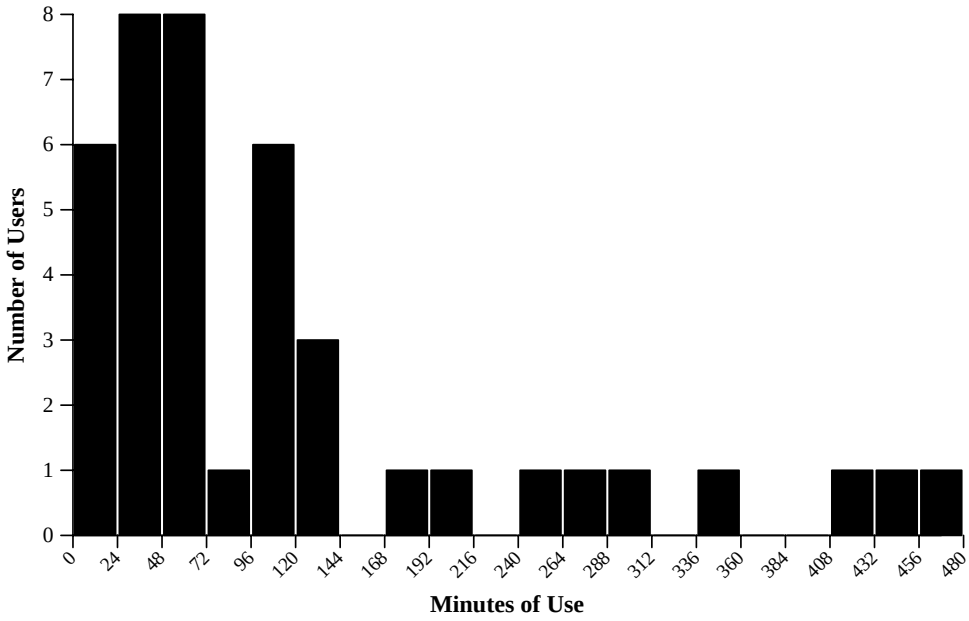


Figure 14: Histogram showing the number of users against their total interaction time divided into 20 equal divisions.

8.4 Discussion

Although the previous graphs give an indication of the way the tab was used it is important to acknowledge the limitations of this study in representing the use of the tab as a consumer item. First, the the user group was too small for statistically significant results. Second, the system was still under development and the applications were not fully supported. Furthermore, participants in the study were not customers but rather laboratory staff using the tab as a prototype. It was up to them to invent ways to use the tab, develop new applications and create ways to incorporate the tab into established work patterns. As a result, we must qualify the numbers with anecdotal evidence and further discussion of the ways people used the tab. Some of these remarks are listed below:

Rich Gold: does not see any value in using a tab in his own office because a powerful workstation is at hand.

John Ellis: prefers to use the tab in his own office to read his e-mail so that he does not have to rearrange the windows on his workstation screen.

Dan Swinehart: found the tab system had a long response time, but found that the tab system was faster than Mosaic for finding the definition of a word, .

Helen Davis: has used the email application and Unistrokes to take notes during seminars and then mailed them to herself.

A number of people found the PARCTAB too heavy or awkward to wear.

Two women tab users (Karin Petersen and Nancy Freige) remarked that the design of the belt clip was oriented towards a particular clothing style. For example, not all outfits include belts, and furthermore not all belts work well with clip on devices. Doug Terry also found

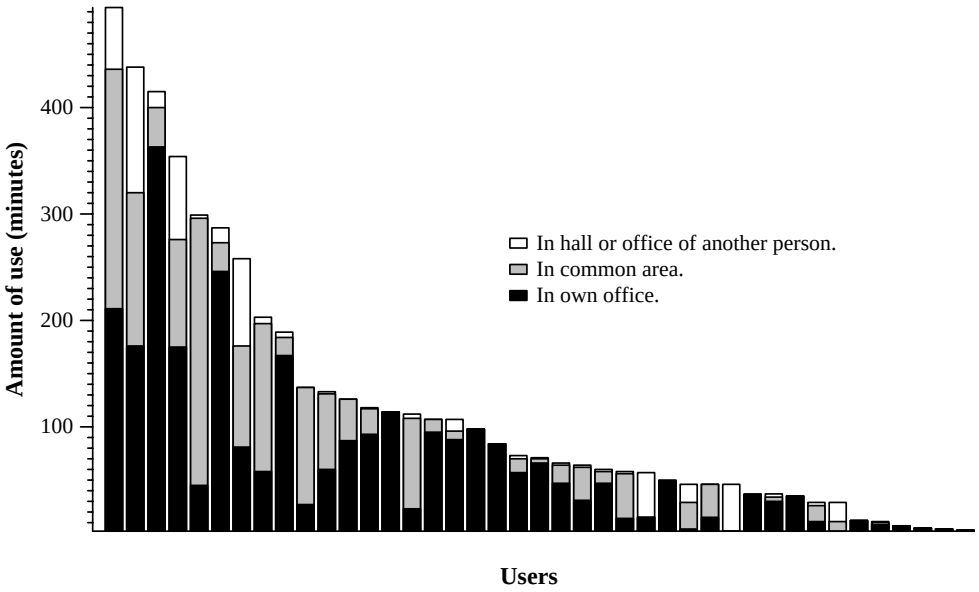


Figure 15: Histogram showing the total interaction time for each user in seconds split between three location types: a user's own office, a common area, a hall or another person's office.

the tab clip inadequate for his use. Instead he used a small zippered nylon and (infrared transparent) fishnet pouch to hold a tab so that it could be attached to his belt and continue to report his location.

A researcher who preferred to remain anonymous commented on the difficulties of building new applications in Modula-3: 'I don't want to say anything against Modula-3 but if I have to learn a new language at the same time as trying to program a new [computer] I may not get much done.'

The ease of reading text on the small screen surprised most of the participants in the use study. At the beginning of the study we found almost 1/2 of the participants had commented that because of the low resolution of the screen they did not intend to read longer files.

As the list above indicates, it is difficult to suggest a 'typical' use of the PARCTAB The PARCTAB system was an experiment that many people volunteered to participate in. It was shaped by their own ideas, needs and contributions. A direct consequence of building a system that can be used by a community is that it is possible to gain understanding of the real problems (see Section 9), issues to be addressed, and activities that need to be supported.

8.5 Research at other Sites

To gain more general experience we gave the tab system (including tabs, transceivers, and software) to a number of other research departments. The largest of these sites was the Rank Xerox Research Centre (Cambridge, England) with 12 transceivers and 10 PARCTABs. Flinders University (Adelaide, Australia) University of Washington, University of Toronto and Olivetti Research Ltd (Cambridge, England) also received small numbers of PARCTAB

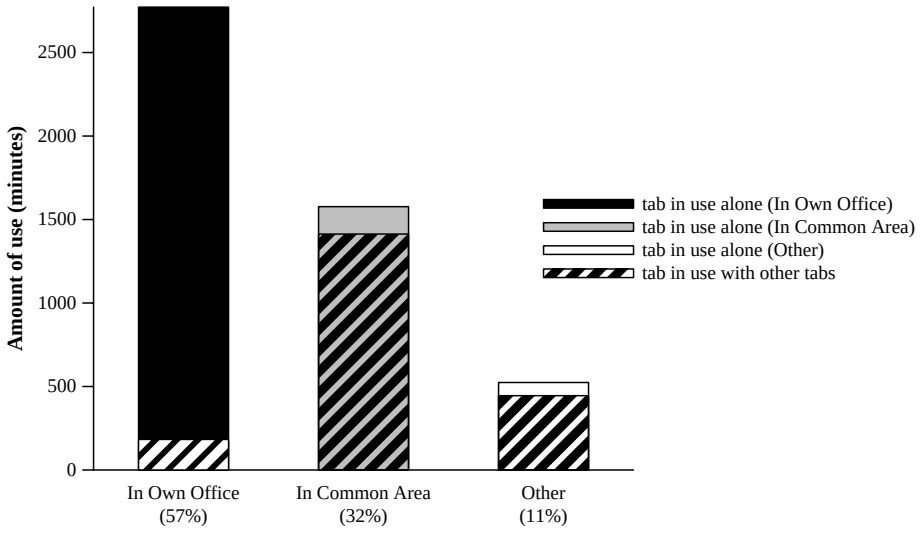


Figure 16: Histogram showing the total interaction time by all users for each of the three general areas: a user's own office, a common area, a hall or another person's office.

system components for their own research. RXRC produced a number of applications (see Section 7), and the University of Toronto now uses tabs to control the equipment in its "telepresence" room.

9 CONCLUSION

The PARCTAB system enables a unique set of applications that have used communication and context to enhance their operation. By designing a system and deploying it, we were able to gain some insight into the benefits and problems faced by mobile systems. The following sections draw some conclusions.

9.1 Design Perspective

The PARCTAB architecture depends on small-cell wireless communication. It thus combines portability with information about context. A downside of this approach was that the PARCTAB was not very useful out of contact with the network. Some of our users were dissatisfied that the tab had only very limited use when disconnected from the network. Perhaps the real value of a PDA comes from both connected and disconnected operation. One without the other leaves them dissatisfied.

Our system design was based on a distributed architecture containing many components. Although each component was relatively simple the complete system presented a level of complexity that made it difficult to debug. We learned to remove as many points of failure as possible to allow users to understand what was going on.

9.2 Bandwidth Limitations

One of our early design assumptions was that a 19.2k baud link was adequate for building the PARCTAB system. If users do not often share cells or do not, on average, operate their PARCTABs at the same time, the system can usually respond within 1 or 2 seconds. In meetings, however, these assumptions seldom hold true. Users tend to operate tabs at the beginning of meetings, at short breaks and perhaps when they are bored, resulting in synchronized use and poor performance.

We now recognize that such systems have to be engineered to deal with the maximum congestion that can result from the maximum number of mobile units in a room. Figures based on average usage patterns do not justify cutting corners.

9.3 Characteristics of User Generated Traffic

Another early design assumption was that applications would have repeating usage patterns of the form 1) event 2) screen update 3) delay, with the delay caused by the time it takes a user to read the screen. However the Unistroke interface changed this pattern. A Unistroke writer can make several strokes per second. In combination with other Unistroke traffic, this can generate a load greater than the IR network was designed to handle. As a result, we have begun work on improving the partitioning of applications between the PARCTAB and the rest of the system. The Unistroke recognizer has recently been ported to the PARCTAB firmware, allowing us to send packets of characters rather than a sequence of stylus positions. This approach uses significantly less bandwidth in both directions and will be included in a future PARCTAB release. Display keyboards could work the same way.

The largest impediment for people using Unistrokes was the slow response-time of the system when displaying a character after each stroke of the stylus. Many of the participants who had learnt Unistrokes, claimed to be able to write faster than the system could keep up. All of those who learnt Unistrokes felt that it was a superior form of text input.

9.4 Factors Affecting Acceptance

Whether or not a tab is adopted in the workplace turns out to depend on many factors: among them size, appearance, convenience, peer pressure, application types, and critical mass of applications. People, in general, have well established work habits that are a barrier to learning a new system. Applications that solve a real problem are however compelling, and a diversity of application type makes the tab a solution to many problems.

It has become clear that changing the nature of a single characteristic can tip the balance between acceptance and rejection of the device e.g., the design of a suitable belt/clothes clip. Small changes in design can have large effects and this makes it difficult to make predictions. Building a system intended for use is the only way to really find out.

We have discovered how difficult it can be to persuade people to make changes to their daily routine in order use a device like the PARCTAB. Furthermore, an individual's style of dress has a significant impact on whether a tab can be easily attached and worn like a pager. One user's tab fell off a belt in a parking lot, damaging the device, and making the user less willing to carry it.

Many people expressed an interest in a system that could be used both inside and outside the building, and if this had been the case, they might have adopted it in more readily. It is clear that a conventional radio broadcast scheme would allow greater mobility, but at the

expense of bandwidth and the lack of context. A more comprehensive system might use a combination of nano-cellular communications for in-building use and a packet-radio scheme for outside use.

There were two important aspects of tab use in the CSL study that were demonstrated by the logging data. First, the brief period that applications were used (50% were under 100 seconds), and second, the generally infrequent usage-pattern.

Given that the typical behavior is of short user-interaction-times, we might be able to better support a user's needs by supplying more casual interfaces that summarize data on the tab top-level screen (e.g., time, weather, amount of mail to read etc), enabling a user to retrieve information at a glance. Perhaps icons that change state to represent the activity of their underlying applications would address this issue, replacing the desktop metaphor currently in use by a wrist-watch metaphor.

The total interaction-time combined for all tabs was not very large. This is as much a reflection on the context of use as any inherent difficulties with the tab. The researchers and support staff participating in this experiment work in a computer-saturated environment. They are never far from a workstation, and apart from attending meetings, their work practices typically do not rely on being mobile (see Figure 16, percentage of time spent in an office). This suggests that further work for integrating the tab into the office environment needs to be considered, for example, using the tab as another computer monitor. But it also suggests that in a manufacturing environment, or a hospital, tabs might support established mobile work-practices.

It should also be noted that the tab system is a prototype and is not supported to the same extent as an established product (e.g., no user manuals). In this case study, the users are participating in the development and therefore it is more appropriate to think of them as participants rather than users.

In the near future, a device capable of performing the PARCTAB's functions could be made about one third the thickness and one third the weight of the current version (3-4 mm thick and perhaps 70 grams). This may further encourage its use.

9.5 Application Development

We set out from the start to encourage the user community to become involved in writing applications. The original Modula-3 programming environment, although a state-of-the-art approach to building systems, was unfamiliar to many of the users. In some cases learning it was too much trouble for producing a relatively simple application. In addition, the compiler created large binaries (often greater than 3MB for each application), imposing a significant load on machine resources when many applications were active. Making it possible to write applications in Tcl/Tk and Hypercard was significant in broadening the interest of application developers.

9.6 Importance of User Interface

An innovative part of building the PARCTAB system has been the design of user interfaces that are suited to a small screen e.g., elision and Unistrokes. The latter is a powerful technique that can be used with pen-based computers of any size.

The design of the PARCTAB packaging was clearly successful. In particular, our users liked a design that was adapted to either right or left handed people. It was also clear that

three physical buttons usually provided an unambiguous mode of use. Although it was tempting to design the user interface with more buttons, enforced simplicity has turned out to be a bonus.

9.7 Popular Applications

Our system provided many programs that could be used in the work environment. It is interesting to consider the four most commonly invoked. In first place was the electronic mail reader, providing access to e-mail that is normally only available at a workstation. Perhaps this is not surprising given that the study was carried out at a computer-science research laboratory. However, electronic mail is becoming more popular in the business community and this result might be significant in predicting a future market.

The weather program scored second highest. It is possible this shows an inherent fascination with weather, or the program may just be good demo-ware. We hope that this indicates a deeper interest in information that is up-to-date and easily accessed. In that case, a mobile interface to the World Wide Web or other information services might prove compelling.

In third place was the file browser, providing access to text and command files stored in the Unix Network Filing System. Since the entire study group works almost entirely with electronic documents which are available on-line, this is a likely result. Finally, in fourth place was the tab loader, which allows users to store information in the tab's local memory and use it outside the infrared network. It is not surprising it has also been popular.

Although the unistroke notetaker was not invoked very often, it accounted for a significant chunk of total tab usage. It is possible that note-taking could become a heavily-used application, especially if local processing of unistrokes yields the expected improvements in performance.

Of the remaining applications there is one result that appears to be out of place. The PARCTAB calendar/diary appeared mid-way through both the popularity and runtime results. In the initial questionnaire all but two of the users had stated that they intended to use the calendar manager regularly. Although there was some difficulty with the compatibility of electronic calendars in use, 80% of the participants could use the appropriate calendar manager on the tab. Given that office environments have schedules that involve many meetings and numerous visitors, this result seems low. We have found, however, that users often have traditional solutions to this problem in place (e.g., pocket-book diaries). New solutions that are as good, or only marginally better (such as tab access to an on-line calendar) are not easily adopted.

9.8 System Benefits

One important contribution of the PARCTAB system has been the experimental infrastructure that allows users to prototype new application ideas. The system has been something of a catalyst in generating new ideas in the area of Ubiquitous Computing and has inspired novel applications. Because the infrastructure is easily assembled and can be exported to other test sites, we have also had the benefit of stimulating other research.

9.9 Future Work

Many system issues still need to be explored, for example, how to resolve conflicts during disconnected operation when related information has changed in both the mobile and the fixed part of the system [7; 35; 6]. Another area that needs exploring is how to partition system functionality across a wireless link with the aim of reducing communication latency. An extension of the existing work that would allow us to make better use of system context, is the design of a mechanism for the precise location of objects in a building. Ubiquitous computing could take advantage of precise location information: knowing which screen a user is currently looking at, for example, is invaluable when deciding how to present urgent information. Finally, the whole area of miniature user-interface research deserves further study and has the potential for many more innovations.

Ubiquitous computing has been the main inspiration for the PARCTAB project. The use of this system has allowed us to study context-sensitive applications. These prototype applications have demonstrated the potential for innovation in this area. In the future we expect to continue to carry out research with the PARCTAB, and also other hardware and software that will help define the future of ubiquitous computing. Our experience with the PARCTAB systems look very promising and brings us a step closer to realizing that future.

ACKNOWLEDGMENTS

We wish to thank the many summer interns that have contributed to this project and made it fun to work on: Michael Tso, Nina Bhatti, Angie Hinrichs, David Maltz, Maria Okasaki, and George Fitzmaurice. We also wish to thank: Jennifer Collins and Sonos Models for facilitating the PARCTAB packaging; Bill Buxton (UT) for his advice concerning UI design; Terri Watson, Berry Kercheval and Ron Frederick for developing novel applications; Natalie Jeremijenko for collecting and processing results from the tab usage experiment; Olivetti Research Ltd (ORL) and Andy Hopper for collaborating with us while developing the communication hardware; Brian Bershad (UW), Craig Mudge(Flinders) and Mike Flynn for their keen advice and collaboration. Finally, we wish to thank and acknowledge Mik Lamming for his original contributions and support during the lifetime of the project.

References

[1] Norman Adams, Rich Gold, Bill N. Schilit, Michael Tso, and Roy Want. An infrared network for mobile computers. In *Proceedings USENIX Symposium on Mobile & Location-independent Computing*, pages 41–52. USENIX Association, August 1993.

[2] Jeff Bachiochi. X-10 interfacing with plix. *Circuit Cellular INK*, pages 74–79, Oct/Nov. 1992.

[3] Bill Buxton. Living in augmented reality: Ubiquitous media and reactive environments. *To appear in CACM*, 1995.

[4] William Buxton and Tom Moran. *EuroPARC's Integrated interactive intermedia facility (iiif): early experiences*. North-Holland, 1990.

[5] George Calhoun. *Digital Cellular Radio*. Artech House Inc, 1988.

[6] A. Demers D. Terry, K. Petersen, M. Spreitzer, , M.M. Theimer, and B. Welch. Session guarantees for weakly-consistent replicated data. In *Proc. 3rd International Conference on Parallel and Distributed Information Systems*, pages 140–149, September 1994.

[7] A. Demers, K. Petersen, M. Spreitzer, D. Terry, M.M. Theimer, and B. Welch. The bayou architecture: Support for data sharing among mobile users. In *Proceedings Workshop on Mobile Computing Systems and Applications*. IEEE, December 1994.

[8] Alan Demers, Scott Elrod, Christopher Kantarjiev, and Edward Richley. A nanocellular local area network using near-field rf coupling. In *Proceedings of Virginia Tech's Fourth Symposium on Wireless Personal Communications*, pages 10.1–10.16, June 1994.

[9] Scott Elrod, Richard Bruce, Rich Gold, David Goldberg, Frank Halasz, William Janssen, David Lee, Kim McCall, Elin Pedersen, Ken Pier, John Tang, and Brent Welch. Liveboard: A large interactive display supporting group meetings, presentations and remote collaboration. In *Proc. of the Conference on Computer Human Interaction (CHI)*, pages 599–607, May 1992.

[10] Scott Elrod, Gene Hall, Rick Costanza, Michael Dixon, and Jim des Rivieres. Responsive office environments. *CACM*, 36(7):84–85, July 1993. In Special Issue, Computer-Augmented Environments.

[11] N. Fishman and M.S. Mazer. Experience in deploying an active badge system. In *Proc. of IEEE Globecom Workshop on Networking of Personal Communications Applications*, December 1992.

[12] Jim Fulton and Chris Kent Kantarjiev. An update on low bandwidth X (LBX). Technical Report CSL-93-2, Xerox Palo Alto Research Center, February 1993.

[13] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *CACM*, 35(12):61–70, Dec 1992.

[14] David Goldberg and Cate Richardson. Touch typing with a stylus. In *Proc. Conference on Human Factors in Computing Systems (INTERCHI)*, pages 80–87. ACM/SigCHI, Apr 1993.

[15] Andy Harter and Andy Hopper. A distributed location system for the active office. *IEEE Network*, pages 62–70, January/February 1994.

[16] Oliver Jones. *Introduction to the X Window System*. Prentice Hall, 1989.

[17] Christopher Kent Kantarjiev, A. Demers, R.T. Krivacic R. Frederick, and M. Weiser. Experiences with X in a wireless environment. In *Proceedings USENIX Symposium on Mobile & Location-independent Computing*, pages 117–128. USENIX Association, August 1993.

[18] M. Lamming. Towards future personalised information environments. In *FRIEND21 Symposium on Next Generation Human Interfaces*, Tokyo Japan, 1994. Also available as RXRC TR 94-104, 61 Regent St., Cambridge, UK.

[19] M. Lamming, P. Brown, K. Carter, M. Eldridge, M. Flynn, G. Louie, P. Robinson, and A. Sellen. The design of a human memory prosthesis. *Computer Journal*, 37(3):153–163, 1994.

[20] M. Lamming and M. Flynn. Forget-me-not: intimate computing in support of human memory. In *FRIEND21 Symposium on Next Generation Human Interfaces*, Tokyo Japan, 1994. Also available as RXRC TR 94-103, 61 Regent St., Cambridge, UK.

[21] Robert Langreth. Total recall. *Popular Science*, pages 46–82, February 1995.

[22] Greg Nelson. *System Programming with Modula-3*. Series in Innovative Technology. Prentice Hall, 1991.

[23] William Newman and Mik Lamming. *Interactive System Design*. Addison-Wesley, 1995.

[24] John K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[25] Karin Petersen. Tcl/tk for a personal digital assistant. In *Proceedings of the USENIX Symposium on Very High Level Languages (VHLL)*, pages 41–56, Santa Fe, New Mexico, October 26-28 1994. USENIX Association.

[26] Ken Pier and James A. Landay. Issues for location-independent interfaces. In *Xerox Parc Blue&White P92-00159*, December 1992.

[27] Bill N. Schilit, Norman Adams, Rich Gold, Michael Tso, and Roy Want. The PARCTAB mobile computing system. In *Proceedings Fourth Workshop on Workstation Operating Systems (WWOS-IV)*, pages 34–39. IEEE, October 1993.

[28] Bill N. Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Proceedings Workshop on Mobile Computing Systems and Applications*. IEEE, December 1994.

[29] Bill N. Schilit and Marvin M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, pages 22–32, September/October 1994.

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

[31] Mike Spreitzer and Marvin Theimer. Providing location information in a ubiquitous computing environment. In *Proceedings of the Fourteenth ACM Symposium on Operating System Principles*, pages 270–283, Asheville, NC, December 1993. SIGOPS, ACM.

[32] Mike Spreitzer and Marvin Theimer. Scalable, secure, mobile computing with location information. *CACM*, 36(7):27, July 1993. In Special Issue, Computer-Augmented Environments.

[33] Mike Spreitzer and Marvin Theimer. Architectural considerations for scalable, secure, mobile computing with location information. In *Proc. 14th Intl. Conf. on Distributed Computing Systems*, pages 29–38. IEEE, June 1994.

[34] Andrew Tanenbaum. *Computer Networks*. Prentice Hall, 1981.

[35] M.M. Theimer, A. Demers, K. Petersen, M. Spreitzer, D. Terry, and B. Welch. Dealing with tentative data values in disconnected work groups. In *Proceedings Workshop on Mobile Computing Systems and Applications*. IEEE, December 1994.

[36] M. Tokoro and K. Tamaru. Acknowledging ethernet. *Compcon*, pages 320–325, October 1977.

[37] Roy Want and Andy Hopper. Active badges and personal interactive computing objects. *IEEE Transactions on Consumer Electronics*, 38(1):10–20, Feb 1992.

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

[39] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, September 1991.

[40] Mark Weiser. Hot topic: Ubiquitous computing. *IEEE Computer*, pages 71–72, October 1993.

[41] Mark Weiser. Some computer science issues in ubiquitous computing. *CACM*, 36(7):74–83, July 1993. In Special Issue, Computer-Augmented Environments.

[42] Mark Weiser. The world is not a desktop. *Interactions*, pages 7–8, January 1994.

[43] Mark Weiser, Alan Demers, Brent Welch, and Scott Shenkar. Scheduling for reduced CPU energy. In *Operating System Design and Implementation (OSDI)*, Monterey, CA, 1994.

Contents

List of Figures