SOME COMPUTER SCIENCE ISSUES IN

UBIQUITOUS
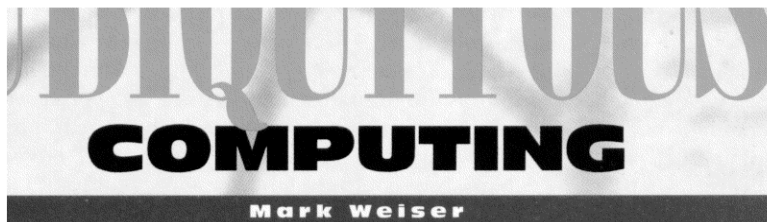
COMPUTING

Mark Weiser

U biquitous computing enhances computer use by making many computers available throughout the physical environment, while making them effectively invisible to the user. This article explains what is new and different about the computer science involved in ubiquitous computing. First, it provides a brief overview of ubiquitous computing, then elaborates through a series of examples drawn from various subdisciplines of computer science: hardware components (e.g., chips), network protocols, interaction substrates (e.g., software for screens and pens), applications, privacy, and computational methods. Ubiquitous computing offers a framework for new and exciting research across the spectrum of computer science.

Since we started this work at Xerox Palo Alto Research Center (PARC) in 1988 a few places have begun work on this possible next-generation computing environment in which each person is continually interacting with hundreds of nearby wirelessly interconnected computers. The goal is to achieve the most effective kind of technology, that which is essentially invisible to the user. To bring computers to this point while retaining their power will require radically new kinds of computers of all sizes and shapes to be available to each person. I call this future world "Ubiquitous Computing" (Ubicomp) [27]. The research method for ubiquitous computing is standard experimental computer science: the construction of working prototypes of the necessary infrastructure in sufficient quantity to debug the viability of the systems in everyday use; ourselves and a few colleagues serving as guinea pigs. This is

an important step toward ensuring that our infrastructure research is robust and scalable in the face of the details of the real world.

The idea of ubiquitous computing first arose from contemplating the place of today's computer in actual activities of everyday life. In particular, anthropological studies of work life [14, 22] teach us that people primarily work in a world of shared situations and unexamined technological skills. The computer today is isolated from the overall situation, however, and fails to get out of the way of the work. In other words, rather than being a tool through which we work, and thus disappearing from our awareness, the computer too often remains the focus of attention. And this is true throughout the domain of personal computing as currently implemented and discussed for the future, whether one thinks of personal computers, palmtops, or dynabooks. The characterization of the future computer as the "intimate computer" [12], or "rather like a human assistant" [25] makes this attention to the machine itself particularly apparent.

Getting the computer out of the way is not easy. This is not a graphical user interface (GUI) problem, but is a property of the whole context of usage of the machine and the attributes of its physical properties: the keyboard, the weight and desktop position of screens, and so on. The problem is not one of "interface." For the same reason of context, this is not a multimedia problem, resulting from any particular deficiency in the ability to display certain kinds of real-time data or integrate them into applications. (Indeed, multimedia tries to grab attention, the opposite of the ubiquitous computing ideal of invisibility.) The challenge is to create a new kind of relationship of people to computers, one in which the computer would have to take the lead in becoming vastly better at getting out of the way, allowing people to just go about their lives.

In 1988, when I started PARC's work on ubiquitous computing, virtual reality (VR) came the closest to enacting the principles we believed important. In its ultimate envisionment, VR causes the computer to become effectively invisible by taking over the human sensory and affector systems [19]. VR is extremely useful in scientific visualization and entertainment, and will be very significant for those niches. But as a tool for productively changing everyone's relationship to computation, it has two crucial flaws: first, at the present time (1992), and probably for decades, it cannot produce a simulation of significant verisimilitude at reasonable cost (today, at any cost). This means that users will not be fooled and the computer will not be out of the way. Second, and most important, it has the goal of fooling the user—of leaving the everyday physical world behind. This is at odds with the goal of better integrating the computer into human activities, since humans are of and in the everyday world.

Ubiquitous computing is exploring quite different ground from personal digital assistants, or the idea that computers should be autonomous agents that take on our goals. The difference can be characterized as follows: Suppose you want to lift a heavy object. You can call in your strong assistant to lift it for you, or you yourself can be made effortlessly, unconsciously, stronger and just lift it. There are times when both are good. Much of the past and current effort for better computers has been aimed at the former; ubiquitous computing aims at the latter.

The approach I took was to attempt the definition and construction of new computing artifacts for use in everyday life. I took my inspiration from the everyday objects found in offices and homes, in particular those objects whose purpose is to capture or convey information. The most ubiquitous current informational technology embodied in artifacts is the use of written symbols, primarily words, but including also pictographs, clocks, and other sorts of symbolic communication. Rather than attempting to reproduce these objects inside the virtual computer world, leading to another "desktop model" [2], I wanted to put the new kind of computer out in this world of concrete information conveyers. Since these written artifacts occur in many different sizes and shapes, with many different qualities, I wanted the computer embodiments to be of many sizes and shapes, including tiny inexpensive ones that could bring computing to everyone.

The physical world comes in all sizes and shapes. For practical reasons our ubiquitous computing work begins with just three different sizes of devices: enough to give some scope, not enough to deter progress. The first size is the wall-sized interactive surface, analogous to the office whiteboard or the home magnet-covered refrigerator or bulletin board. The second size is the notepad, envisioned not as a personal computer but as analogous to scrap paper to be grabbed and used easily, with many being used by a person at one time. The cluttered office desk or messy front-hall table are real-life examples. Finally, the third size is the tiny computer, analogous to tiny individual notes or Post-it notes, and also similar to the tiny little displays of words found on book spines, light switches, and hallways. Again, I saw this not as a personal computer, but as a pervasive part of everyday life, with many active at all times. I called these three sizes of computers boards, pads, and tabs, and adopted the slogan that, for each person in an office, there should be hundreds of tabs, tens of pads, and one or two boards. Specifications for some prototypes of these three sizes in use at PARC are shown in Figure 1.

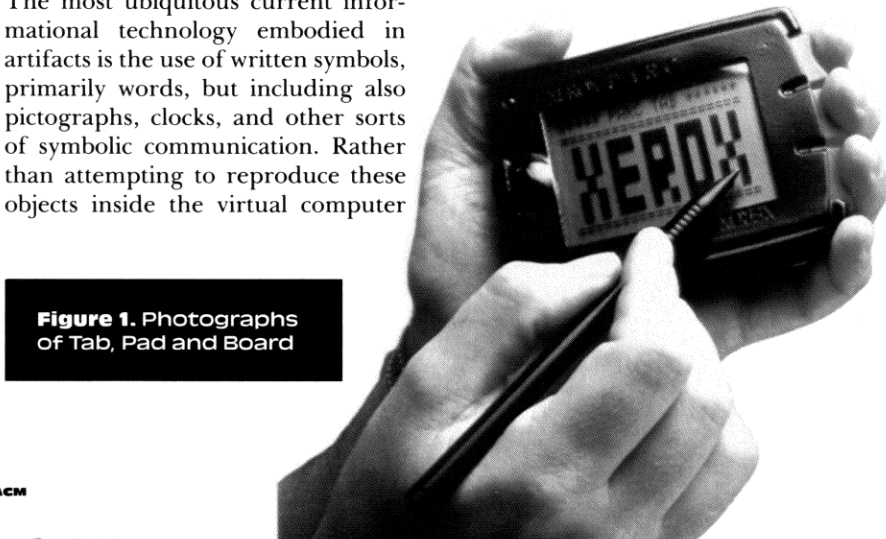This then is Phase I of ubiquitous computing: to construct, deploy, and

Figure 1. Photographs of Tab, Pad and Board

learn from a computing environment consisting of tabs, pads, and boards. This is only Phase I, because it is unlikely to achieve optimal invisibility. (Later phases are yet to be determined.) But it is a start down the radical direction, for computer science, away from emphasis on the machine and back to the person and his or her life in the world of work, play, and home.

Hardware Prototypes

New hardware systems design for ubiquitous computing has been oriented toward experimental platforms for systems and applications of invisibility. New chips have been less important than combinations of existing components that create experimental opportunities. The first ubiquitous computing technology to be deployed was the Liveboard [6], which is now a Xerox product. Two other important pieces of prototype hardware supporting our research at PARC are the Tab and the Pad.

Tab

The ParcTab is a tiny information doorway. For user interaction it has a pressure-sensitive screen on top of the display, three buttons positioned underneath the natural finger positions, and the ability to sense its position within a building. The display and touchpad it uses are standard commercial units.

The key hardware design problems affecting the pad are physical size and power consumption. With several dozens of these devices sitting around the office, in briefcases, in pockets, one cannot change their batteries every week. The PARC design uses the 8051 chip to control detailed interactions, and includes software that keeps power usage down. The major outboard components are a small analog/digital converter for the pressure-sensitive screen, and analog sense circuitry for the IR receiver. Interestingly, although we have been approached by several chip manufacturers about our possible need for custom chips for the Tab, the Tab is not short of places to put chips. The display size leaves plenty of room, and the display thickness dominates total size. Off-the-shelf components are more than adequate for exploring this design space, even with our severe size, weight, and power constraints.

A key part of our design philosophy is to put devices in everyday use, not just demonstrate them. We can only use techniques suitable for quantity 100 replication. This excludes certain techniques that could make a huge difference, such as the integration of components onto the display surface itself. This technology is being explored at PARC, as well as other research organizations. While it is very promising, it is not yet ready for replication.

The Tab architecture incorporates a careful balance of display size, bandwidth, processing, and memory. For instance, the small display means that even the tiny processor is capable of providing a four-frames-per-second video rate, and the IR bandwidth is capable of delivering this. The bandwidth is also such that the processor can actually time the pulse widths in software timing loops. Our current design has insufficient storage, and we are increasing the amount of nonvolatile RAM in future tabs from 8K to 128K. The tab's goal of casual use similar to that of Post-it notes puts it into a design space generally unexplored in the commercial or research sector.

Pad

The pad is really a family of notebook-sized devices. Our initial pad, the ScratchPad, plugged into a Sun SBus card and provided an X-Window-system-compatible writing and display surface. This same design was used inside our first wall-sized displays, the liveboards, as well. Our later untethered pad devices, the XPad and MPad, continued the system design principles of X-compatibility, ease of construction, and flexibility in software and hardware expansion.

As I write this article, at the end of 1992, commercial portable pen devices have been on the market for two years, although most of the early companies have now gone out of business. Why should a pioneering research lab build its own such device? Each year we ask ourselves the same question, and so far three things always drive us to continue to
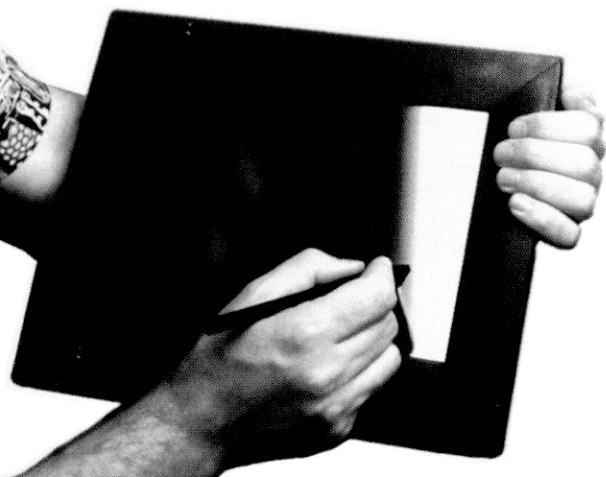
design our own pad hardware:

First, we need the correct balance of features—this is the essence of systems design. The commercial devices all aim at particular niches, balancing their design to that niche. For research we need a rather different balance, particularly for ubiquitous computing. For instance, can the device communicate simultaneously along multiple channels? Does the operating system support multiprocessing? What about the potential for high-speed tethering? Is there a high-quality pen? Is there a high-speed expansion port sufficient for video in and out? Is sound in/out and ISDN connectivity available? Optional keyboard? Any one commercial device tends to satisfy some of these needs, ignore others, and choose a balance of the ones it does satisfy, optimizing its niche, rather than ubiquitous computing-style scrap computing. The balance for us emphasizes communication, system memory, multimedia, and expansion ports.

Second, apart from balance of features are the requirements for particular features. Key among these are a pen emphasis, connection to research environments such as Unix, and communication emphasis. A high-speed (>64KB/sec) wireless capability is not built into any commercial devices, and they do not generally have a sufficiently high-speed port to add such a radio. Commercial devices generally come with DOS or Penpoint, and while we have developed in both, they are not our favorite research vehicles because they lack full access and customizability.

The third factor driving our own pad designs is ease of expansion and modification. We need full hardware specifications, complete operating system source code, and the ability to remove and replace both hardware and software components. Naturally these goals are opposed to best price in a niche market, which orients the documentation to the end user, and keeps prices down by integrated rather than modular design.

We have built and used three generations of Pad designs. Six scratchpads were built, three XPads, and 13 MPads, the latest. The MPad uses an FPGA for almost all random logic,

giving extreme flexibility. For instance, changing the power control functions and adding high-quality sound were relatively simple FPGA changes. The MPad has both IR (tab compatible) and radio communication built-in and includes sufficient uncommitted space for adding new circuit boards later. It can be used with a tether that provides it with recharging and operating power and an Ethernet connection. The operating system is a standalone version of the public-domain Portable Common Runtime developed at PARC [28].

The Computer Science of Ubicomp

To construct and deploy tabs, pads, and boards at PARC, we found ourselves having to readdress some of the well-worked areas of existing computer science. The fruitfulness of ubiquitous computing for new computer science problems justified our belief in the ubiquitous computing framework.

The following subsections "ascend" the levels of organization of a computer system, from hardware to application. One or two examples of computer science work required by ubiquitous computing are described for each level. Ubicomp is not yet a coherent body of work, but consists of a few scattered communities. The point of this article is to help others understand some of the new research challenges in ubiquitous computing, and inspire them to work on them. This is more akin to a tutorial than a survey, and necessarily selective. The areas included are hardware components (e.g., chips), network protocols, interaction substrates (e.g., software for screens and pens), applications, privacy, and computational methods.

Issues of Hardware Components

In addition to the new systems of tabs, pads, and boards, ubiquitous computing necessitates some new kinds of devices. Examples of three new kinds of hardware devices are very low-power computing, low-power high-bits/cubic-meter communication, and pen devices.

Low Power

In general the need for high performance has dominated the need for low-power consumption in processor design. However, recognizing the new requirements of ubiquitous computing, a number of people have begun work in using additional chip area to reduce power rather than to increase performance [16]. One key approach is to reduce the clocking frequency of their chips by increasing pipelining or parallelism. Then, by running the chips at reduced voltage, the effect is a net reduction in power, because power falls off as the square of the voltage, while only about twice the area is needed to run at half the clock speed.

$$\text{Power} = C_L * V_{dd}^2 * f$$
where C_L is the gate capacitance,
V_{dd} the supply voltage,
and f the clocking frequency.

This method of reducing power leads to two new areas of chip design: circuits that will run at low power, and architectures that sacrifice area for power over performance. The second requires some additional comment, because one might suppose one would simply design the fastest possible chip, and then run it at reduced clock and voltage. However, as Lyon illustrates, circuits in chips designed for high speed generally fail to work at low voltages. Furthermore, attention to special circuits may permit operation over a much wider range of voltage operation, or achieve power savings via other special techniques, such as adiabatic switching [16].

Wireless

A wireless network capable of accommodating hundreds of high-speed devices for every person is well beyond the commercial wireless systems planned for the next 10 years [20], which are aimed at one low-speed (64kb/sec or voice) device per person. Most wireless work uses a figure of merit of bits/sec × range, and seeks to increase this product. We believe that a better figure of merit is bits/sec/meter^3. This figure of merit causes the optimization of total bandwidth throughout a 3D space, leading to design points of very tiny cellular systems.

Because we felt the commercial world was ignoring the proper figure

To construct and deploy tabs, pads, and boards at PARC, we found ourselves
having to readdress some of the well-worked areas of existing computer science.

of merit, we initiated our own small radio program. In 1989 we built spread-spectrum transceivers at 900MHz, but found them difficult to build and adjust, and prone to noise and multipath interference. In 1990 we built direct frequency-shift-keyed transceivers also at 900MHz, using very low power to be license-free. While much simpler, these transceivers had unexpectedly and unpredictably long range, causing mutual interference and multipath problems. In 1991 we designed and built our current radios, which use the near-field of the electromagnetic spectrum. The near-field has an effective fall-off of r^6 in power, instead of the more usual r^2, where r is the distance from the transmitter. At the proper levels this band does not require an FCC license, permits reuse of the same frequency over and over again in a building, has virtually no multipath or blocking effects, and permits transceivers that use extremely low power and low parts count. We have deployed a number of near-field radios within PARC.

Pens

A third new hardware component is the pen for very large displays. We needed pens that would work over a large area (at least 60in × 40in), not require a tether, and work with back projection. These requirements are generated from the particular needs of large displays in ubiquitous computing—casual use, no training, naturalness, simultaneous multiple use. No existing pens or touchpads could come close to these requirements. Therefore members of the Electronics and Imaging lab at PARC devised a new infrared pen. A camera-like device behind the screen senses the pen position, and information about the pen state (e.g., buttons) is modulated along the IR beam. The pens need not touch the screen, but can operate from several feet away. Considerable DSP and analog design work underlies making these pens effective components of the ubiqui-

tous computing system [6].

Network Protocols

Ubicomp changes the emphasis in networking in at least four areas: wireless media access, wide-bandwidth range, real-time capabilities for multimedia over standard networks, and packet routing.

Wireless Media Access

A "media access" protocol provides access to a physical medium. Common media access methods in wired domains are collision detection and token-passing. These do not work unchanged in a wireless domain because not every device is assured of being able to hear every other device (this is called the "hidden terminal" problem). Furthermore, earlier wireless work used assumptions of complete autonomy, or a statically configured network, while ubiquitous computing requires a cellular topology, with mobile devices frequently coming on- and off-line. We have adapted a media access protocol called MACA, first described by Karn [11], with some of our own modifications for fairness and efficiency.

The key idea of MACA is for the two stations desiring to communicate to first do a short handshake of Request-To-Send-N-bytes followed by Clear-To-Send-N-bytes. This exchange allows all other stations to hear that there is going to be traffic, and for how long they should remain quiet. Collisions, which are detected by timeouts, occur only during the short Request-To-Send packet.

Adapting MACA for ubiquitous computing use required considerable attention to fairness and real-time requirements. MACA requires stations whose packets collide to back off a random time and try again. If all stations but one back-off, that one can dominate the bandwidth. By requiring all stations to adapt the back-off parameter of their neighbors, we create a much fairer allocation of bandwidth.

Some applications need guaranteed bandwidth for voice or video. We added a new packet type, NCTS(n) (Not Clear To Send), to suppress all other transmissions for (n) bytes. This packet is sufficient for a basestation to do effective bandwidth allocation among its mobile units. The solution is robust, in the sense that if the basestation stops allocating bandwidth the system reverts to normal contention.

When a number of mobile units share a single basestation, that basestation may be a bottleneck for communication. For fairness, a basestation with $N > 1$ nonempty output queues needs to contend for bandwidth as though it were N stations. We therefore make the basestation contend just enough more aggressively that it is N times more likely to win a contention for media access.

Other Network Issues

Two other areas of networking research at PARC with ubiquitous computing implications are Gb networks and real-time protocols. Gb-per-second speeds are important because of the increasing number of medium-speed devices anticipated by ubiquitous computing, and the growing importance of real-time (multimedia) data. One hundred 256kb/sec portables per office implies a Gb per group of 40 offices, with all of PARC needing an aggregate of some five Gb/sec. This has led us to do research into local-area ATM switches, in association with other Gb networking projects [15].

Real-time protocols are a new area of focus in packet-switched networks. Although real-time delivery has always been important in telephony, a few hundred milliseconds never mattered in typical packet-switched applications such as telnet and file transfer. With the ubiquitous use of packet-switching, even for telephony using ATM, the need for real-time capable protocols has become urgent if the packet networks are going to support multimedia applications.

Again, in association with other members of the research community, PARC is exploring new protocols for enabling multimedia on the packet-switched Internet [4].

The Internet routing protocol, IP, has been in use for over 10 years. However, neither this protocol nor its OSI equivalent, CLNP, provides sufficient infrastructure for highly mobile devices. Both interpret fields in the network names of devices in order to route packets to the device. For instance, the "13" in IP name 13.2.0.45 is interpreted to mean net 13, and network routers anywhere in the world are expected to know how to get a packet to net 13, and all devices whose name starts with 13 are expected to be on that network. This assumption fails as soon as a user of a net 13 mobile device takes her device on a visit to net 36 (Stanford). Changing the device name dynamically depending on location is no solution: higher-level protocols such as TCP assume that underlying names will not change during the life of a connection, and a name change must be accompanied by informing the entire network of the change so that existing services can find the device.

A number of solutions have been proposed to this problem, among them Virtual IP from Sony [24], and Mobile IP from Columbia University [10]. These solutions permit existing IP networks to interoperate transparently with roaming hosts. The key idea of all approaches is to add a second layer of IP address, the "real" address indicating location, to the existing fixed-device address. Special routing nodes that forward packets to the correct real address, and keep track of where this address is, are required for all approaches.*

Interaction Substrates
Ubicomp has led us to explore new substrates for interaction. Four such substrates are mentioned here, spanning the space from virtual keyboards to protocols for window systems.

Tabs have a very small interaction area—too small for a keyboard, too small even for standard handprint-

*The Internet community has a working group considering standards for this area (contact deering@xerox.com for more information).

ing recognition. Handprinting has the further problem of requiring looking at what is written. Improvements in voice recognition are no panacea, because when other people are present, voice will often be inappropriate. As one possible solution, we developed a method of touch-printing that uses only a tiny area and does not require looking. A drawback of our method is it requires a new printing alphabet to be memorized, and reaches only half the speed of a fast typist [8].

Liveboards have a high interaction area, 400 times that of the tab. Using conventional pull-down or pop-up menus might require walking across the room to the appropriate button, a serious problem. We have developed methods of location-independent interaction by which even complex interactions can be popped up at any location. [13].

The X-Window system, although designed for network use, makes it difficult for windows to move once instantiated at a given X server. This is because the server retains considerable state about individual windows, and does not provide convenient ways to move that state. For instance, context and window IDs are determined solely by the server, and cannot be transferred to a new server, so applications that depend on knowing their value (almost all) will break if a window changes servers. However, in the ubiquitous computing world a user may move frequently from device to device, and want to bring windows along.

Christian Jacobi at PARC has implemented a new X toolkit that facilitates window migration. Applications need not be aware that they have moved from one screen to another; or if desired, the user can be so informed with an upcall. We have written a number of applications on top of this toolkit, all of which can be "whistled up" over the network to follow the user from screen to screen. The author, for instance, frequently keeps a single program development and editing environment open for days at a time, migrating its windows back and forth from home to work and back each day.

A final window system problem is bandwidth. The bandwidth available

to devices in ubiquitous computing can vary from Kb/sec to Gb/sec, and with window migration a single application may have to dynamically adjust to bandwidth over time. The X-Window system protocol was primarily developed for Ethernet speeds, and most of the applications written in it were similarly tested at 10Mb/sec. To solve the problem of efficient X-Window use at lower bandwidth, the X consortium is sponsoring a "Low Bandwidth X" (LBX) working group to investigate new methods of lowering bandwidth. [7].

Applications
Applications are of course the whole point of ubiquitous computing. Two examples of applications are locating people and shared drawing.

Ubicomp permits the location of people and objects in an environment. This was first pioneered by Olivetti Research Labs in Cambridge, England, in their Active Badge system [26]. In ubiquitous computing we continued to extend this work, using it for video annotation and updating dynamic maps. For instance, Figure 2 shows a portion of CSL early one morning, and the individual faces are the locations of people. This map is updated every few seconds, permitting quick locating of people, as well as quickly noticing a meeting one might want to go to (or where one can find a fresh pot of coffee).

Xerox PARC, EuroPARC, and the Olivetti Research Center have built several different kinds of location servers. Generally these have two parts: a central database of information about location that can be quickly queried and dumped, and a group of servers that collect information about location and update the database. Information about location can be deduced from logins, or collected directly from an active badge system. The location database may be organized to dynamically notify clients, or simply to facilitate frequent polling.

Some example uses of location information are automatic phone forwarding, locating an individual for a meeting, and watching general activity in a building to feel in touch with

its cycles of activity (important for telecommuting).

Xerox PARC has investigated a number of shared meeting tools over the past decade, starting with the CoLab work [21], and continuing with videodraw and commune [23]. Two new tools were developed for investigating problems in ubiquitous computing. The first is Tivoli [18], the second is Slate,—each tool is based on different implementation paradigms. First their similarities: both emphasize pen-based drawing on a surface; both accept scanned input and can print the results; both can have several users simultaneously operating independently on different or the same pages; both support multiple pages. Tivoli has a sophisticated notion of a stroke as spline, and has a number of features making use of processing the contents and relationships among

Table 1. Some hardware prototypes in use at Xerox PARC

| ParcTab	MPad	Liveboard
Dimensions: 10.2- × 7.8- × 2.4cm Weight: 7.2 oz Screen: 6.2- × 4.2cm, 128- × 64 monochrome Touch input: passive pressure sensing Sound: Piezo speaker Wireless interfaces: IR at 850nm, DEC/Olivetti active badge compatible, 19.2k baud PWM baseband modulation, CSMA Processor: Intel 8051-type, 8k (v1) 128k (v2) nvram Ports: I²C external bus, recharge port Battery: 12 hours continuous or est. 2 weeks normal use, rechargeable	Dimensions: 22.2- × 28- × 3.8cm Weight: 5lbs 4oz Screen: 640- × 480 LCD Display (3 levels of grey) Pen: tethered electromagnetic sensing Sound: Built-in microphone, Speaker, Piezo Beeper Wireless Interfaces: 250Kbps Radio, 19.2Kbps IR Processor: Motorola 68302, 4MB of DRAM, 1/2MB of VRAM, 1/4MB of EPR External Ports: Stylus/microphone, PCMCIA, 1MB Serial, RS232, I²C bus, Keyboard Internal Ports: Second audio channel, ISDN, Expansion Port Battery: rechargeable, 3 hours	Dimensions: 83in, 52in and 30in Weight: 560lb (250kg) Screen: very bright 45- × 65in, 1024- × 768 monochrome pixels, 640- × 480 pixels color, also NTSC video Pen: IR wireless Stereo sound Networking, processor, and ports determined by choice of embedded workstation, either PC or Sun 12 amps at 115 volts

Computer Science Laboratory

coffee

Employee entrance, outbound

Employee entrance, inbound
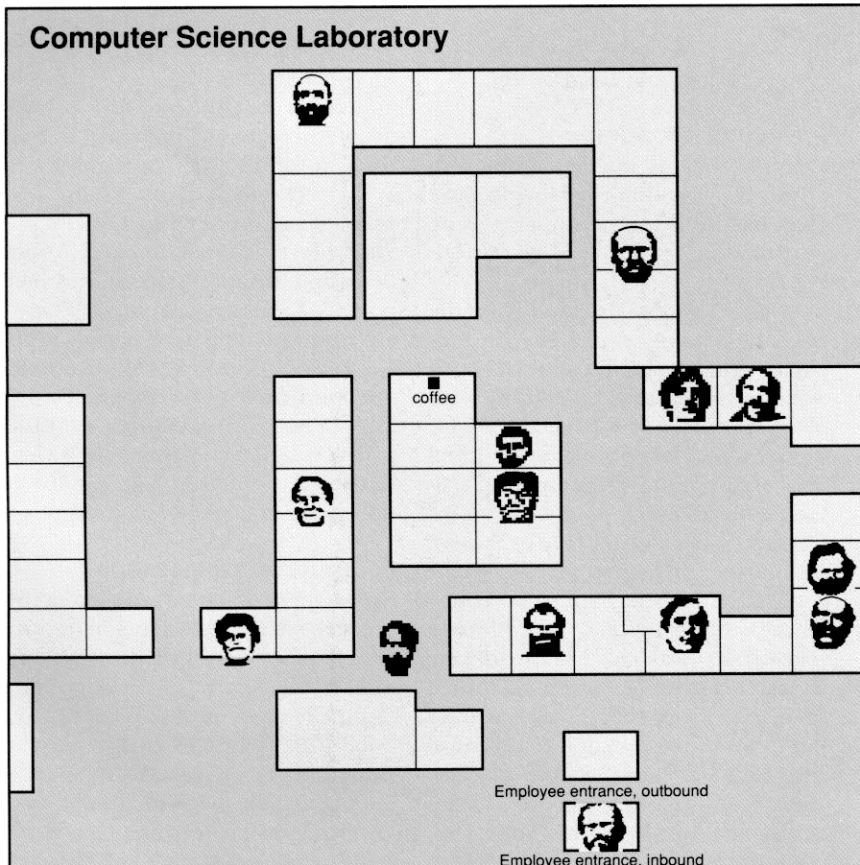
Figure 2. Display of CSL activity from personal locators

strokes. Tivoli also uses gestures as input control to select, move, and change the properties of objects on the screen. When several people use Tivoli, each must be running a separate copy, and connect to the others. On the other hand, Slate is completely pixel-based, simply drawing ink on the screen. Slate manages all the shared windows for all participants, as long as they are running an X-Window server, so its aggregate resource use can be much lower than Tivoli, and it is easier to set up with large numbers of participants. In practice we have used slate from a Sun to support shared drawing with users on Macs and PCs. Both Slate and Tivoli have received regular use at PARC.

Shared drawing tools are a topic of research at many places. For instance, Bellcore has a toolkit for building shared tools [9], and Jacobsen at LBL uses multicast packets to reduce bandwidth during shared tool use. There are some commercial products [3], but these are usually not multipage and so not really suitable for creating documents or interacting over the course of an entire meeting. The optimal shared drawing tool has not been built. For its user interface, there remain issues such as multiple cursors or one, gestures or not, and using an ink or a character recognition model of pen input. For its substrate, is it better to have a single application with multiple windows, or many applications independently connected? Is packet-multicast a good substrate to use? What would it take to support shared drawing among 50 people; 5,000 people? The answers are likely to be both technological and social.

Three new kinds of applications of ubiquitous computing are beginning to be explored at Xerox PARC. One is to take advantage of true invisibility, literally hiding machines in the walls. An example is the Responsive Environment project led by Scott Elrod. This aims to make a building's heat, light, and power more responsive to individually customized needs, saving energy and making a more comfortable environment.

A second new approach is to use so-called "virtual communities" via the technology of MUDs. A MUD, or

"Multi-User Dungeon," is a program that accepts network connections from multiple simultaneous users and provides access to a shared database of 'rooms', 'exits', and other objects. MUDs have existed for about 10 years, being used almost exclusively for recreational purposes. However, the simple technology of MUDs should also be useful in other, nonrecreational applications, providing a casual environment integrating virtual and real worlds [5].

A third new approach is the use of collaboration to specify information filtering. Described in the December 1992 issue of *Communications of the ACM*, this work by Doug Terry extends previous notions of information filters by permitting filters to reference other filters, or to depend on the values of multiple messages. For instance, one can select all messages that have been replied to by Smith (these messages do not even mention Smith, of course), or all messages that three other people found interesting. Implementing this required inventing the idea of a "continuous query," which can effectively sample a changing database at all points in time. Called "Tapestry," this system provides new ways for people to invisibly collaborate.

Privacy of Location
Cellular systems inherently need to know the location of devices and their use in order to properly route information. For instance, the traveling pattern of a frequent cellular phone user can be deduced from the roaming data of cellular service providers. This problem could be much worse in ubiquitous computing with its more extensive use of cellular wireless. So a key problem with ubiquitous computing is preserving privacy of location. One solution, a central database of location information, means the privacy controls can be centralized and perhaps done well—on the other hand one break-in there reveals all, and centrality is unlikely to scale worldwide. A second source of insecurity is the transmission of the location information to a central site. This site is the obvious place to try to snoop packets, or even to use traffic analysis on source addresses.

Our initial designs were all central,

initially with unrestricted access, gradually moving toward individual users' controlling who can access information about them. Our preferred design avoids a central repository, instead storing information about each person at that person's PC or workstation. Programs that need to know a person's location must query the PC, and proceed through whatever security measures the user has chosen to install. EuroPARC uses a system of this sort.

Accumulating information about individuals over long periods is one of the more useful things to do, but quickly raises hackles. A key problem for location is how to provide occasional location information for clients who need it, while somehow preventing the reliable accumulation of long-term trends about an individual. So far at PARC we have experimented only with short-term accumulation of information to produce automatic daily diaries of activity [17].

It is important to realize there can never be a purely technological solution to privacy, and social issues must be considered in their own right. In the computer science lab we are trying to construct systems that are privacy-enabled, giving power to the individual. But only society can cause the right system to be used. To help prevent future oppressive employers or governments from taking this power away, we are also encouraging the wide dissemination of information about location systems and their potential for harm. We have cooperated with a number of articles in the *San Jose Mercury News,* the *Washington Post,* and the *New York Times* on this topic. The result, we hope, is technological enablement combined with an informed populace that cannot be tricked in the name of technology.

Computational Methods
An example of a new problem in theoretical computer science emerging from ubiquitous computing is optimal cache sharing. This problem originally arose in discussions of optimal disk cache design for portable computer architectures. Bandwidth to the portable machine may be quite low, while its processing power is relatively high. This introduces as a

possible design point the compression of pages in a RAM cache, rather than writing them all the way back over a slow link. The question arises of the optimal strategy for partitioning memory between compressed and uncompressed pages.

This problem can be generalized as follows [1]:

The Cache Sharing Problem. A problem instance is given by a sequence of page requests. Pages are of two types, U and C (for uncompressed and compressed), and each page is either IN or OUT. A request is served by changing the requested page to IN if it is currently OUT. Initially all pages are OUT. The cost to change a type-U (type-C) page from OUT to IN is C_U (respectively, C_C). When a requested page is OUT, we say that the algorithm missed. Removing a page from memory is free.

Lower Bound Theorem: No deterministic, on-line algorithm for cache sharing can be c-competitive for

$$c < MAX \quad (1 + C_U/(C_U + C_C), \quad 1 + C_C/(C_U + C_C))$$

This lower bound for c ranges from 1.5 to 2, and no on-line algorithm can approach closer to the optimum than this factor. Bern et al. [1] also constructed an algorithm that achieves this factor, therefore providing an upper bound as well. They further propose a set of more general symbolic programming tools for solving competitive algorithms of this sort.

Concluding Remarks

As we start to put tabs, pads, and boards into use, the first phase of ubiquitous computing should enter its most productive period. With this substrate in place we can make much more progress both in evaluating our technologies and in choosing our next steps. A key part of this evaluation is using the analyses of psychologists, anthropologists, application writers, artists, marketers, and customers. We believe they will find some features work well; we know they will find some features do not work. Thus we will begin again the cycle of cross-disciplinary fertilization and learning. Ubicomp is likely

to provide a framework for interesting and productive work for many more years or decades, but we have much to learn about the details.

References

1. Bern, M., Greene, D., Raghunathan. On-line algorithms for cache sharing. *25th ACM Symposium on Theory of Computing* (San Diego, Calif., 1993).
2. Buxton, W. Smoke and mirrors. *Byte 15*, 7 (July 1990) 205–210.
3. Chatterjee, S. Sun enters computer conferencing market. *Sunworld 5*, 10 (Oct. 1992), Integrated Media, San Francisco, 32–34.
4. Clark, D.D., Shenker, S., Zhang, L. Supporting real-time applications in an integrated services packet network:Architecture and mechanism. *SIGCOMM '92 Conference Proceedings.* Communications Architectures and Protocols. (Baltimore, Md, Aug. 17–20, 1992). *Computer Commun. Rev. 22*, 4 (Oct. 1992) New York, N.Y. pp. 14–26.
5. Curtis, P. MUDDING: Social phenomena in text-based virtual realities. *DIAC—Directions and Implications of Advanced Computing Symposium Proceedings.* Computer Professionals for Social Responsibility (Palo Alto, Calif. May, 1992).
6. Elrod, B., Gold, Goldberg, Halasz, Janssen, Lee, McCall, Pedersen, Pier, Tang and Welch. Liveboard: A large interactive display supporting group meetings, presentations and remote collaboration. *CHI '92 Conference Proceedings*, May 1992. ACM, New York, N.Y. pp. 599–607.
7. Fulton, J. and Kantarjiev, C. An update on low bandwidth X (LBX). In *Proceedings of the Seventh Annual X Technical Conference* (Boston, Mass Jan. 1993), To be published in The X Resource by O'Reilly and Associates.
8. Goldberg, D., Richardson, C. Touch typing with a stylus. To be published in *INTERCHI '93*.
9. Hill, R.D., Brinck, T., Patterson, J.F., Rohall, S.L. and Wilner, W.T. The RENDEZVOUS language and architecture: Tools for constructing multi-

user interactive systems. *Commun. ACM, 36*, 1 (Jan. 1993). To be published.
10. Ioannidis, J., Maguire, G.Q., Jr. The design and implementation of a mobile internetworking architecture. *Usenix Conference Proceedings, Usenix '93* (Jan. 1993). To be published.
11. Karn, P. MACA—A new channel access method for packet radio. In *Proceedings of the ARRL 9th Computer Networking Conference* (London Ontario, Canada, Sept. 22, 1990). ISBN 0-87259-337-1.
12. Kay, A. Computers, networks, and education. *Sci. Am.* (Sept. 1991), 138–148.
13. Kurtenbach, G. and Buxton, W. The limits of expert performance using hierarchic marking menus. To be published in *INTERCHI '93*.
14. Lave, J. Situated Learning: Legitimate Peripheral Participation. Cambridge University Press, Cambridge, New York, N.Y. 1991.
15. Lyles, B.J., Swinehart, D.C. The emerging gigabit environment and the role of local ATM. *IEEE Commun.* (Apr. 1992), 52–57.
16. Lyon, R.F. Cost, power, and parallelism in speech signal processing. *Custom Integrated Circuits Conference, IEEE* (San Diego, May 9–12, 1993).
17. Newman, W.M., Eldridge, M.A., Lamming, M.G. PEPYS: Generating autobiographies by automatic tracking. EuroPARC Tech. Rep. Cambridge, England.
18. Pedersen, E., McCall, K., Moran, T.P., Halasz, F., Tivoli, F.G.: An electronic whiteboard for informal workgroup meetings. To be published in *INTERCHI'93*.
19. Rheingold, H. *Virtual Reality.* Summit Books. New York, N.Y., 1991.
20. Rush, C.M. How WARC '92 will affect mobile services. *IEEE Commun.* (Oct. 1992), 90–96.
21. Stefik, M., Foster, G., Bobrow, D.G., Kahn, K., Lanning, S. and Suchman, L. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Commun. ACM 30*, 1 (Jan. 1987), 32–47.
22. Suchman, L.A. Plans and situated actions: The problem of human-machine communication. Xerox PARC Tech. Rep. ISL-6, Feb. 1985.

23. Tang, J.C. and Minneman, S.L. VideoDraw: A video interface for collaborative drawing. *ACM Trans. Off. Inf. Syst. 9*, 2 (Apr. 1991), 170–184.

24. Teraoka, F., Tokote, Y., Tokoro, M. A network architecture providing host migration transparency. In *Proceedings of SIGCOMM'91* (Sept. 1991), pp. 209–220.

25. Tesler, L.G. Networked computing in the 1990's. *Sci. Am.* (Sept. 1991), 86–93.

26. Want, R., Hopper, A., Falcao, V. and Gibbons, J. The active badge location system. *ACM Trans Inf. Syst. 10*, 1 (Jan. 1992), 91–102.

27. Weiser, M. The computer for the twenty-first century. *Sci. Am.* (Sept. 1991), 94–104.

28. Weiser, M., Demers, A. and Hauser, C. The portable common runtime approach to interoperability. In *Proceedings of the ACM Symposium on Operating Systems Principles* (Dec. 1989).

About the Author:
MARK WEISER is principal scientist and manager of the Computer Science laboratory at Xerox PARC. Current research interests include new theories of automatic memory reclamation (garbage collection), visualization of operating system internals, ubiquitous computing and embodied virtuality. **Author's Present Address:** Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304; email: weiser. parc@xerox.com

SCOTT ELROD, GENE HALL, RICK COSTANZA, MICHAEL DIXON, AND JIM DES RIVIERES

Responsive Office Environments

A team of facilities staff and computer science researchers at Xerox PARC is exploring applications of ubiquitous computing to energy management and environmental control [1]. By interconnecting PARC's rich computational infrastructure with a computerized building management system (BMS) that controls heating, air conditioning, lighting and desktop appliances, we plan to explore new strategies for energy conservation and office comfort control.

Over the past 20 years, the designers of heating, ventilation, and air conditioning (HVAC) systems have gradually shifted toward the use of digital computers [2], replacing direct manual control and simple analog feedback loops such as thermostats. Digital control makes possible more flexible, precise, and complex control strategies that in turn can provide significant energy savings. For example, most computer-controlled buildings offer automatic temperature setbacks to reduce energy consumption after working hours and on weekends. Many systems also control lighting to save additional energy.

Current progress in low-cost distributed computing, communications, and sensing technologies will enable future building management systems to be much more responsive to individual preferences and activities. The following are examples of the types of enhancements that are possible:

• Small, location-sensing mobile computers such as PARCTabs provide an ideal interface to allow office occupants and maintenance staff to set parameters (such as preferred ranges for temperature and light level) and receive feedback about current conditions.
• Occupancy sensors are already used at PARC to avoid heating or cooling conference rooms when they are not in use. We are now experimenting with user-selectable strategies for switching off lights, computer displays, and other appliances and for setting back the air conditioning when offices are unoccupied.
• Other sensors can be integrated into the system to provide additional automatic functionality, such as adjusting artificial lights and/or closing blinds to compensate for changing natural light levels, lowering light levels while a workstation is in use, and adjusting ventilation depending on whether doors are opened or closed.
• Active badges and on-line calendars may be used to identify periods in which an office's occupant will be away for an extended period of time (e.g., because the occupant is at a meeting, at lunch, or away from the building).

To quantify the potential for energy savings by such strategies, we conducted a detailed one-day energy and occupancy audit of PARC's Computer Science Laboratory. This study revealed that offices are typically vacant for 50% of the average 9-hour workday. Initial calculations suggest that occupancy-based control of lights and computer monitors alone would save $45,000 per year at PARC, and similar control of the air conditioning system could save as much as $90,000.

Encouraged by these estimates, we have constructed a flexible hardware and software testbed to experiment with new energy control strategies. The testbed consists of 13 offices and some adjacent public areas. Within each office we have installed temperature, light level, occupancy, and active badge sensors together with computer-controlled ventilation, heating, electric outlets, and overhead lighting. These devices are connected to a conventional computerized building management system, which in turn is connected to the PARC computer network through a gateway. Using a conventional building management system as the backbone allows us to ensure that the system's basic functionality is highly reliable, while still allowing new control strategies and data analysis to be run from workstations.

As of this writing (March) the complete system is just coming on-line and we are beginning to collect the baseline data needed to evaluate future energy savings. At present, the system continuously logs over 800 variables from the test area and from other parts of the building. Over the next few months we plan to experiment with a variety of control strategies and user interfaces (UIs) and measure their effects.

In the development of this system we encountered a number of design issues common to many ubiquitous computing systems:

• *reliability and invisibility*: Computer systems entrusted with such basic aspects of comfort as temperature and light levels must obviously be highly reliable.