Some Computer Science Issues in Ubiquitous Computing

Mark Weiser
March 23, 1993

[to appear in CACM, July 1993]

Ubiquitous computing is the method of enhancing computer use by making many computers available throughout the physical environment, but making them effectively invisible to the user. Since we started this work at Xerox PARC in 1988, a number of researchers around the world have begun to work in the ubiquitous computing framework. This paper explains what is new and different about the computer science in ubiquitous computing. It starts with a brief overview of ubiquitous computing, and 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.

A few places in the world have begun work on a possible next generation computing environment in which each person is continually interacting with hundreds of nearby wirelessly interconnected computers. The point 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" (short form: "Ubicomp") [Weiser 1991]. 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, using ourselves and a few colleagues as guinea pigs. This is an important step towards insuring 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 [Suchman 1985, Lave 1991] teach us that people primarily work in a world of shared situations and unexamined technological skills. However the computer today is isolated and isolating from the overall situation, and fails to get out of the way of the work. In other words, rather than being a tool through which we work, and so which disappears 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 PC’s, palmtops, or dynabooks. The characterization of the future computer as the "intimate computer" [Kay 1991], or "rather like a human assistant" [Tesler 1991] 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 affordances 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 was not a multimedia problem, resulting from any particular deficiency in the ability to display certains kinds of realtime 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 so people could 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 [Rheingold 91]. 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 importantly, 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 can be yourself 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" [Buxton 90], instead I wanted to put the new kind of computer also out in this world of concrete information conveyers. And because these written artifacts occur in many different sizes and shapes, with many different affordances, so I wanted the computer embodiments to be of many sizes and shapes, including tiny inexpensive ones that could bring computing to everyone.

The physical affordances in the world come 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 in use by a person at once. 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 PostIts, and also like the tiny little displays of words found on book spines, lightswitches, 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, respectively, 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 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 attention on the machine and back on 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 towards 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 [Elrod 92], 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 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 in the pad are 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 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, which excludes certain things that could make a huge difference, such as the integration of components onto the display surface itself. This technology, being explored at PARC, ISI, and TI, while very promising, is not yet ready for replication.

The Tab architecture is carefully balanced among display size, bandwidth, processing, and memory. For instance, the small display means that even the tiny processor is capable of four frame/sec video to it, 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 non-volatile RAM in future tabs from 8k to 128k. The tab’s goal of postit-note-like casual use 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, 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 be building 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 right balance of features; this is the essence of systems design. The commercial devices all aim at particular niches, and so balance their design to that niche. For research we need a rather different balance, all the more so for ubiquitous computing. For instance, can the device communicate simultaneously along multiple channels? Does the O.S 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 available? Optional keyboard? Any one commercial device tends to satisfy some of these, ignore others, and choose a balance of the ones it does satisfy that optimize its niche, rather than ubiquitous computing-style scrap computing. The balance for us emphasizes communication, ram, multi-media, and expansion ports.

Second, apart from balance are the requirements for particular features. Key among these are a pen emphasis, connection to research environments like Unix, and communication emphasis. A high-speed (>64kbps) wireless capability is built into no commercial devices, nor do they generally have a sufficiently high speed port to which such a radio can be added. Commercial devices generally come with DOS or Penpoint, and while we have developed in both, they are not our favorite research vehicles because of lack of full access and customizability.

The third thing driving our own pad designs is ease of expansion and modification. We need full hardware specs, complete O.S. source code, and the ability to rip-out 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 which keeps price down by integrated rather than modular design.

We have now gone through three generations of Pad designs. Six scratchpads were built, three XPads, and thirteen 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 built-in both IR (tab compatible) and radio communication, 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 [Weiser 89].

FIGURE 1 - some hardware prototypes in use at PARC

FIGURE 2 - Photographs of each of tabs, pads, boards (at end of paper).

The CS of Ubicomp

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

In what follows I walk up the levels of organization of a computer system, from hardware to application. For each level I describe one or two examples of computer science work required by ubiquitous computing. Ubicomp is not yet a coherent body of work, but consists of a few scattered communities. The point of this paper 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 I discuss below 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 needs 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 [Lyon 93]. 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.

Power = CL * Vdd2 * f

- where CL is the gate capacitance, Vdd 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 that 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 [Lyon 93].

Wireless

A wireless network capable of accommodating hundreds of high speed devices for every person is well beyond the commercial wireless systems planned even ten years out [Rush 92], which are aimed at one low speed (64kbps or voice) device per person. Most wireless work uses a figure of merit of bits/sec x range, and seeks to increase this product. We believe that a better figure of merit is bits/sec/meter3. This figure of merit causes the optimization of total bandwidth throughout a three-dimensional space, leading to design points of very tiny cellular systems.

Because we felt the commercial world was ignoring the proper figure 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 r6 in power, instead of the more usual r2, 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 60"x40"), 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, multiple people at once. 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 ubiquitous computing system [Elrod 92].

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.

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 Phil Karn [Karn 90], 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 RTS packet.

Adapting MACA for ubiquitous computing use required considerable attention to fairness and real-time requirements. MACA (like the original ethernet) requires stations whose packets collide to backoff a random time and try again. If all stations but one backoff, that one can dominate the bandwidth. By requiring all stations to adapt the backoff 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 then 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.

Two other areas of networking research at PARC with ubiquitous computing implications are gigabit networks and real-time protocols. Gigabit-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 256kbps portables per office implies a gigabit per group of forty offices, with all of PARC needing an aggregate of some five gigabits/sec. This has led us to do research into local-area ATM switches, in association with other gigabit networking projects [Lyles 92].

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 like 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 multi-media applications. Again in association with other members of the research community, PARC is exploring new protocols for enabling multimedia on the packet-switched internet [Clark 92].

The internet routing protocol, IP, has been in use for over ten years. However, neither it 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 like TCP assume that underlying names won’t 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 [Teraoka 91], and Mobile IP from Columbia University [Ioannidis 93]. 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 right real address, and keep track of where this address is, are required for all approaches. The internet community has a working group looking at standards for this area (contact Deering@xerox.com for more information).

Interaction Substrates

Ubicomp has led us into looking at new substrates for interaction. I mention four here that span the space from virtual keyboards to protocols for window systems.

Pads have a tiny interaction area -- too small for a keyboard, too small even for standard handprinting recognition. Handprinting has the further problem that it requires 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. As drawbacks, our method requires a new printing alphabet to be memorized, and reaches only half the speed of a fast typist [Goldberg 93].

Liveboards have a huge interaction area, 400 times that of the tab. Using conventional pulldown or popup 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. [Kurtenbach 93].

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 that applications that depend upon knowing their value (almost all) will break if a window changes servers. However, in the ubiquitous computing world a user may be moving frequently from device to device, and wanting 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 they like, they 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 kilobits/sec to gigabits/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 10Mbps. 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. [Fulton 93].

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 work at Olivetti Research Labs in Cambridge, England, in their Active Badge system [Want 92]. In ubiquitous computing we continue to extend this work, using it for video annotation, and updating dynamic maps. For instance, the picture below (figure 3) 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).

Figure 3. Display of CSL activity from personal locators.

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).

PARC has investigated a number of shared meeting tools over the past decade, starting with the CoLab work [Stefik 87], and continuing with videodraw and commune [Tang 91]. Two new tools were developed for investigating problems in ubiquitous computing. The first is Tivoli [Pedersen 93], the second Slate, each based upon different implementation paradigms. First their similarities: they both emphasize pen-based drawing on a surface, they both accept scanned input and can print the results, they both can have several users at once operating independently on different or the same pages, they 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 strokes. Tivoli also uses gestures as input control to select, move, and change the properties of objects on the screen. When multiple 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 setup 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 at many places. For instance, Bellcore has a toolkit for building shared tools [Hill 93], and Jacobsen at LBL uses multicast packets to reduce bandwidth during shared tool use. There are some commercial products [Chatterjee 92], but these are usually not multi-page and so not really suitable for creating documents or interacting over the course of a whole 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 both technological and social.

Three new kinds of applications of ubiquitous computing are beginning to be explored at 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 ten years, being used almost exclusively for recreational purposes. However, the simple technology of MUDs should also be useful in other, non-recreational applications, providing a casual environment integrating virtual and real worlds [Curtis 92].

A third new approach is the use of collaboration to specify information filtering. Described in the December 1992 issue of Communcations of the ACM, this work by Doug Terry extends previous notions of information filters by permitting filters to reference other filters, or to depend upon 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 that the privacy controls can be centralized and so 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 towards controls by individual users on who can access information about them. Our preferred design avoids a central repository, but instead stores 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 run whatever gauntlet of security the user has chosen to install there. EuroPARC uses a system of this sort.

Accumulating information about individuals over long periods is both one of the more useful things to do, and also most quickly raises hackles. A key problem for location is how to provide occasional location information for clients that 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 [Newman 90].

It is important to realize that there can never be a purely technological solution to privacy, that social issues must be considered in their own right. In the computer science lab we are trying to construct systems that are privacy enabled, that can give 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 dissimenation 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, introducing 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 [Bern 93]:

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 CU (respectively, CC). 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

- $c < MAX (1+CU/(CU+CC), 1+CC/(CU+CC))$

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 also construct 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, phase I 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 things right; we know they will find some things wrong. Thus we will begin again the cycle of cross-disciplinary fertilization and learning. Ubicomp seems 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.

Acknowledgements: This work was funded by Xerox PARC. Portions of this work were sponsored by DARPA. Ubiquitous computing is only a small part of the work going on at PARC; we are grateful for PARC's rich, cooperative, and fertile environment in support of the document company. Bern 93. Bern, M., Greene, D., Raghunathan. On-line algorithms for cache sharing. 25th ACM Symposium on Theory of Computing, San Diego, 1993.

Buxton 90. Buxton, W. (1990). Smoke and Mirrors. Byte,15(7), July 1990. 205-210.

Chatterjee 92. Chatterjee, Shalini. Sun enters computer conferencing market. Sunworld. pp. 32-34. Vol 5, no. 10. October 1992. Integrated Media, San Francisco.

Clark 92. Clark, David D., Shenker, Scott, Zhang, Lixia. Supporting real-time applications in an integrated services packet network:architecture and mechanism. SIGCOMM '92 Conference Proceedings. Communicatins architectures and protcosl. August 17-20, 1992. Baltimore, Maryland. Computer Communication Review. Vol. 22, no. 4, October 1992.published by Accosication for Comptuing Machinery, New York, NY. pp. 14-26

Curtis 92. Curtis, Pavel. MUDDING: social phenomena in text-based virtual realities. DIAC - Directions and Implications of Advanced Computing. May, 1992 Symposium Proceedings. Computer Professionals for Social Responsibility. Palo Alto, CA.

Elrod 92. Elrod, Bruce, Gold, Goldberg, Halasz, Janssen, Lee, McCall, Pedersen, Pier, Tang, and Welch. Liveboard: a large interactive display supporting group meetings, presentations and remote collaboration. pp. 599-607. CHI '92 Conference proceedings. May 1992. ACM, New York, NY.

Fulton 93. Fulton, Jim and Kantarjiev, Chris. An Update on Low Bandwidth X (LBX). Proceedings of the Seventh Annual X Technical Conference, January, 1993, Boston, MA. Published in The X Resource by O'Reilly and Associates. (to appear)

Goldberg 93. Goldberg, David, Richardson, Cate. Touch Typing with a Stylus. to appear, INTERCHI '93.

Hill 93. Hill, R.D. , T. Brinck, J.F. Patterson, S.L. Rohall, and W.T. Wilner. The RENDEZVOUS Language and Architecture: Tools for Constructing Multi-User Interactive Systems. Communications of the ACM, Vol. 36, No. 1 (Jan. 1993). (to appear)

Ioannidis 93. Ioannidis, John, Maguire, Gerald Q., Jr. The Design and Implementation of a Mobile Internetworking Architecture. Usenix Conference Proceedings, Usenix '93. January 1993. to appear.

Karn 90. Karn, P. MACA - A New Channel Access Method for Packet Radio. Proceedings of the ARRL

9th Computer Networking Conference, London Ontario, Canada, September 22, 1990. ISBN 0-87259-337-1.

Kay 91. Kay, Alan. Computers, Networks, and Education. Scientific American, September 1991. pp. 138-148.

Kurtenbach 93. Kurtenbach, Gordon, Buxton, William. The Limits of Expert Performance Using Hierarchic Marking Menus. to appear, INTERCHI ’93

Lave 91. Lave, Jean. Situated learning: legitimate peripheral participation. Cambridge University Press. Cambridge. New York, NY. 1991.

Lyles 92. Lyles, B. J., Swinehart, D. C. The emerging gigabit environment and the role of local ATM. IEEE Communications Magazine. April 1992. p. 52-57.

Lyon 93. Lyon, Richard F. Cost, Power, and Parallelism in Speech Signal Processing. Custom Integrated Circuits Conference, IEEE, 1993. May 9-12, 1993. San Diego.

Newman 90. Newman, William M., Eldridge, Margery A., Lamming, Machael G. PEPYS: Generating Autobiographies by Automatic Tracking. EuroPARC technical report. Cambridge England.

Pedersen 93. Pedersen, Elin, McCall, Kim, Moran, Thomas P., Halasz, Frank G. Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. to appear, INTERCHI ’93

Rheingold 91. Rheingold, Howard. Virtual Reality. Summit Books. New York, NY. 1991.

Rush 92. Rush, Charles M. How WARC ’92 Will Affect Mobile Services. IEEE Communications Magazine. October 1992. pp. 90-96.

Stefik 87. 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. CACM 30, 1. January 1987. pp. 32-47.

Suchman 85. Suchman, Lucy A. Plans and Situated Actions: The problem of human-machine communication. Xerox PARC Technical Report ISL-6. February 1985

Tang 91. Tang, John C., Minneman, Scott L. VideoDraw: A video interface for collaborative drawing. ACM Trans. on Office Information Systems. Vol 9, no 2. April 1991. pp. 170-184.

Teraoka 91. Teraoka, Fumio, Tokote, Yasuhiko, Tokoro, Mario. A network architecture providing host migration transparency. Proceedings of SIGCOMM’91. ACM pp. 209-220. September 1991.

Tesler 91. Tesler, Lawrence G. Networked Computing in the 1990’s. Scientific American, September 1991. pp. 86-93.

Want 92a. Want, Roy, Hopper, Andy, Falcao, Veronica, and Gibbons, Jonathan. The active badge location system. ACM T. on Information Systems. Vol 10, No. 1 January 1992. pp. 91-102.

Weiser 89. Weiser, Mark, Demers, Alan, Hauser, C. The Portable Common Runtime Approach to Interoperability. Proceedings of the ACM Symposium on Operating Systems Principles, December 1989.

Weiser 91. Weiser, Mark. The Computer for the Twenty-First Century. Scientific American. September 1991. pp. 94-104.