PIP-1: A Personal Information Portal with wireless access to an information infrastructure

John Ioannidis
Gerald Q. Maguire Jr.

450 Computer Science
Columbia University
New York, NY 10027

ji@cs.columbia.edu, maguire@cs.columbia.edu

Technical Report No. CUCS-055-90

DRAFT

Abstract

We discuss our ideas for the Personal Information Portal, its hardware and software platform, our visions for its use and its impact on the student and the professional community.

1 Introduction

The initial description of a personal machine for keeping notes was Vannevar Bush's "memex". He envisioned a machine for "browsing and making notes in an extensive on-line text and graphics system" (Atlantic Monthy, 1945). His dream machine was even more ambitious that the other machines which are described in this document – as he envisioned a complete hypertext system.

What we are seeking is much closer to a realization of the Dynabook idea developed by Alan Kay in the late 1960's. He envisoned that the portable notebook sized device would be the "personal dynamic medium" and would provide access to text, drawings (including animation), and music.

We expect that there will be a number of spinoffs from this effort, just as the effort to develop a Dynabook resulted in quite a number of other developments at Xerox Palo Alto Research Center (PARC):

- Smalltalk-72, Smalltalk-76, Smalltalk-78, and Smalltalk-80,
- programming environment for the Alto,
- window oriented environments, and
- object oriented programming.

Along the way a system called the NoteTaker was developed at Xerox PARC (in 1977)[1]. This was to have been a hand-held device for taking notes, but ended up being as big as a suitcase. Roughly 10 of these devices were built using Intel 8086 microprocessors with 256KB of memory. The Smalltalk-76 kernel was 32KB of code (on the Alto). In order to avoid having to port all this code a major effort to dekernelize Smalltalk was made. Nearly all of the bitmapped graphics code was moved out of the kernel leaving only one primitive (BitBlt) in the kernel. The result was that the NoteTaker kernel was approximately 6KB of 8086 code.

Many of the concepts developed for the Xerox Alto made their way into the Xerox D-machines, the Apple Lisa and Macintosh, and most recently into the NeXT Machine.

Most of these early efforts to produce a useable dynabook were stunted because of unavailable technology. In those days, processors were not powerful enough, memory was scarce and expensive, and hardware in general was bulky. Almost a decade later, we witness a proliferation of portable machines, some more portable than others, that have significant processing and storage capabilities. Some are actually light enough to be carried around at all times, others require more physical effort, but are much more powerful. To mention two extreme examples, the Atari Portfolio is a palm-held IBM-PC-compatible computer that will run MS-DOS programs, weighs well under a pound, and can be carried in a jacket pocket. On the other extreme, the Toshiba-SPARC 'luggable' weighs about 20 pounds, and is functionally equivalent to a Sun Microsystems Sparcstation-1. Between these two extremes there is a whole range of portable micros. Some have LCD displays and some have gas-plasma, some only have text screens and some have high-resolution graphics (some even in color), etc.

The one thing all the current 'portables' have in common is that they all have local secondary storage and emphasis is on local processing power. Communications and reliance on networked resources is a secondary concern. Almost all portable micros have the ability to upload and download files via a serial connection, with some of the more high-end models able to use ethernet or other high-speed interface. Even so, the user has to take special steps to transfer files to and from his[2] machine. Even when faster interfaces are available, the user of a portable machine is always aware of where his files are and where the processing is taking place. Contrast this with the user of a modern workstation, where there is usually

[1]For more information see "The Evolution of the Smalltalk Virtual Machine" by Daniel H. H. Ingalls, in "Smalltalk-80: Bits of History, Words of Advice" by Glenn Krasner

[2]the use of the third person masculine pronoun is merely a grammatical convenience and does not connote gender

no local storage, user files and executable programs reside on a distributed file system, and where all the machines run more or less the same software.

In developing the Personal Information Portal (**PIP**), we seek to combine the portability and convenience of a portable machine while retaining the processing power and reduced administrative load of a workstation. The major change from previous systems is that instead of having the device store all data within itself this proposal allows the device to utilize a wireless connection to an infrastructure of servers that can provide *storage, computation, and communication*.

In keeping with Kay's ideas, we have specified a visual interface (the size of an A4[3] sheet of paper) comparable to current copier quality (greater than 240 bilevel dots per inch, and evolving to greyscale and color), audio quality comparable to CD players (16 bits per sample, with stereo), and user input via either a stylus (for high resolution input) or touch (for low resolution input, such as pointing). Future systems should have voice input, while current systems will have a keyboard. To keep it portable while providing access to networked resources, we shall need a wireless networking interface with bandwidth within an order of magnitude of current Ethernet speeds.

The initial 'market' for these devices would be campus environments (which includes colleges, hospitals, business parks, etc.) as there is a need to build and support the server and communication infrastructure. Perhaps, after an initial period of approximately 10 years the required infrastructure will allow the operation of these devices at some distance from the campus, and/or allow the interoperability of such devices in diverse campuses.

A key idea is that this is *not* a PC, a workstation, or even a terminal – it is, instead, a means for a new set of users to participate in an online community founded on the electronic exchange of information. Donald Langenberg, Chancellor of the University of Illinois at Chicago, has coined the term "inforum" (from "information forum") to refer to this computing milieu, whose infrastructure must include facilities for accessing, storing, manipulating, and communicating information. It is clearly the case that there will be an explosion in the use of these portable devices with the introduction of better communication. We have already seen this explosion in mobile telephones with the introduction of digital cellular telephone networks.

We are trying to build this infrastructure: the **PIP** provides access, the servers provide storage, together they provide manipulation, and the wireless communication network provides the glue that binds it all together. It is important to emphasize that this new group of users will *not* be computer specialists and that this will truly be a *personal* machine; the **PIP** should be the main, if not the sole, piece of computer equipment that a modern professional would ever use to access, process and disseminate information. That is to say, we seek to replace the collection of personal computers, workstations, microfiche readers, facsimile machines, notebooks, agendas, and even books and journals, with a **personal** machine, to be used to access **information**, that is, to be the **portal** through which a modern user communicates and cooperates with his colleagues.

[3] 210x297 millimeters; roughly 8.25x11.7 inches

2 Hardware Specifications

The early attempts at building a dynabook-like device, or any **PIP**-like device for that matter, had mostly hardware problems – packing a sufficient amount of processing power in a device small enough to be carried around. In recent years, we have seen an explosive growth in hardware power, speed and miniaturization, which finally allows us to build machines powerful enough to be used as **PIP**s.

The following form the technological milieu necessary for the conception of the **PIP**:

- Cheap processing power, miniaturization and memory volume. (very powerful processors, surface-mount/multi-chip/WSI components, soon-to-be 64Mbit DRAMs)
- High-resolution display technology that is becoming more affordable and less massive.
- Secondary storage capacity, especially read-only mass storage (CD-ROMS, LaserDisks etc), which makes reference works available in electronic form for a reasonable cost.
- Network capacity (fiber-optic LANs, internetworking everywhere, 1Gbps long-haul networks). In addition, high-speed wireless network interfaces started appearing recently.
- Software advances in both distributed systems software (distributed OSs, file systems etc).

With these technological advances in mind, let us describe a possible configuration for a **PIP**. For each item, we shall be stating what the **PIP** should have, and what the current commercially available technology can provide.

2.1 Physical Limitations

Size: The **PIP** should be small enough to be carried around. A reasonable upper bound on the size is a standard sheet of paper (8.5 by 11 inches, or 210 by 297 millimeters). That is also a lower bound if we want to have a flat screen and paper-like interface. With other display solutions, this need not be a lower bound. Current portable are roughly that size today, with 'palmtops' (like the Atari Portfolio) and high-end calculators (like the HP-75) being even smaller. Portables are nowhere as powerful as we would want the **PIP** to be, but they are within an order of magnitude of that. Also, the logic board from a modern workstation (like the SUN Sparcstation-SLC) measures six by nine inches. Clearly, it should be possible to package such a board in an A4-sized box.

Weight: The **PIP** should be carried comfortably by an average person. Again, a 3-ring binder with a reasonable amount of paper in it provides a good indication of what an acceptable weight might be – less than two kilograms (four pounds). Most of the weight of current portable machines is the display, the disk drives, and the power supply. As we shall describe below, we want the **PIP** to be a completely diskless machine, so that should significantly reduce the weight and power consumption (hence power supply size).

Handling: The **PIP** should be easily carried and not interfere with its user's activities. It should be able to withstand small mechanical shocks that are to be expected during its everyday usage. For example, Hewlett-Packard calculators have been famous throughout the years for being able to withstand 6-foot drops, being run over by cars etc., While the **PIP** will be considerably bigger than a pocket calculator, it should be rugged enough to be able to take some abuse. A machine with no moving parts can be made reasonably rugged without increasing its weight too much.

2.2 User Interface Devices

Display: The display should have sufficient resolution to display multifont text and graphics. A 'megapixel' display with bi-level pixels (black and white) is the minimum acceptable configuration; anything less than that is not enough. A 1280x1024x8 would actually be better, and it is within the range of current technology. An A4-size screen with laser-printer resolution (210x297mm, at 300dpi (11.8 dots per millimeter, or 84 micron pixels!)) would be the goal. With two bits per pixel (four gray levels: white, 33%, 67%, black), that translates to 2480x3508 pixels or about 8.7Mpixels or about 2.2 megs of dual-ported video memory! Special display hardware (BITBLT) for scrolling, windowing, and copying cached bitmaps of glyphs to the screen would reduce the load on the main processor.

There are various alternatives to the display technology. LCD screens are getting denser and larger, but they are also heavy and fragile. An LCD-screen with a touch-sensitive overlay can be used if we insist on a paper-like interface. Currently commercially available LCD screens go up to 1152x900 (Toshiba Sparc). CRT based 300dpi screens are available (e.g., MegaScan Technology's 2560x2048 display).

Another alternative is a "Private-Eye"-like setup. An LED array is mounted inside a small module attached to a head-strap. By means of a vibrating mirror and a lens, one gets the same effect as a scanned line. Currently available is a device with a 280-LED array, capable of 720x280 resolution. The picture is crisp, and moves with one's head. By positioning the module at the "bifocal" position of the eye, one gets the illusion that there is a 20x25cm screen around where the keyboard is. The device does not interfere with binocular vision, and one gets used to it after a very short period of time. There is no fundamental reason why such a setup should not be capable of much higher resolution, say 2048x2560. Two of these could be worn for binocular vision, and a stereo screen effect. With the addition of a head tracker, the user could have a view which changed depending on the direction they turned their head, thus making a very large virtual display screen available. In order to prevent motion sickness either the entire view must be computed in advance and rapidly copied into the refrest memory or the processor must be fast enough to compute the view as the user's head turns[4]. Another advantage of such a solution is that it increases the ruggedness of the design.

Input devices: A keyboard is currently the easiest and fastest way to enter text. An average touch-typist can enter fifty or so words per minute, and an expert can do about

[4]Based on statements of Henry Fuchs at a DARPA workshop.

a hundred. Using different keyboard layouts (e.g., Dvorak), the rate can be even higher. However, there are several problems with keyboards:

- There is a limit on how small a keyboard can be made. If it is too small, and if the keys do not have the right kind of tactile feedback, it becomes almost impossible to use.
- Non-techical people, especially upper management, are reluctant to use a keyboard.
- It's hard to use a keyboard standing up, or with only one hand.
- The keyboard if separate is yet another item to carry around and if attached may often be unnecessary weight and bulk.
- A keyboard is not always the 'natural' way of doing things.

The last point has already been addressed by using a stylus as an input for graphical input device. Why? Better coordination than with mouse. 'clicking' can be handled by having a 'clickpad' operated with the other hand. Handwriting recognition not essential. Gesture recognition is useful (e.g., circling a block of text and moving it to another place in the document being edited), and there are research results that show this to be feasible. Non-computer-literate people are afraid to use a keyboard and to learn keychords and keysequences to accomplish editing tasks.

Stylus only works if the screen is held roughly horizontal.

If the screen is a private-eye, or a projection TV, a mouse can still be used. If we want to get fancy, a dataglove can be worn and the movements of the hand can be followed. Now, if we are using a binocular Private Eye, a glove is mandatory. Imagine having windows floating around in front of you, 3-D windows where graphs and pictures are being drawn. A molecule can be rendered in one such 3-d window, and then have particular atoms moved around, polymer bonds breaken or rearranged, etc. Maybe this is too much for a "Personal Information Portal" but if the technology gets developed for higher-end machines, it will eventually trickle down to the portable world.

Additional User Interface (UI) devices: Sound

A CODEC chip can provide telephone-quality sound for voicemail and other sound-related applications. The output can also be a dual 16-bit D/A converter to provide CD-quality sound. The nearly ubiquitous presence of Sony Walkmans (or clones) indicates just how large a demand there would be for sound. A number of workstations now include audio outputs (e.g., NeXT, SUN).

2.3 Processing Capbilities

Memory:

The Amdahl/Case rule states that a 'balanced' machine should have 1MB of memory per 1 MIPS of CPU speed per 1Mbit/sec of I/O bandwidth. So let us figure how much memory we need. A modern workstation (e.g., a Sun Sparcstation-1 running SunOS 4.1) stops swapping when it has around 24M, when it's running a kernel (of course), the standard system daemons, a window system (X11), gnuemacs, and a few compile jobs. As a rule of thumb, the size of the average program doubles every year and a half. This does not mean that the memory needed doubles, since nowadays most programs share libraries, so a large amount of code is simply reused. An educated guess is that 32Megabytes of will be a sufficient amount of memory for the next few years. This is could be achieved with Wafer-Scale Intergration modules (e.g., Anamartic, Inc., makes 40MByte WSI modules for the Tandem "Cyclone" machines).

Non-volatile memory:

There exists a small set of programs that every machine will be using; e.g., the kernel, the editor, directory utilities, the window system, etc. Rather than load all these every time the **PIP** reboots, we can store them in non-volatile memory. Flash memory is a prime candidate for non-volatile storage in this case. Part of it will be used to store the 'working set' of programs that will run on the **PIP**. The rest can be used as backup memory where the state of the system can be checkpointed to, so that it can be warm-booted after a power cycle. With a typical duty cycle of 10^5 write cycles, one can write to flash memory once an hour and it will still last eleven years. The contents of RAM can be written to flash memory just before powering down. The read-only part of the kernel and other programs can be run directly from the flash memory, thereby reducing the main memory requirements. This requires OS support (however, it is not a significant problem – simply introduce the concept of a flash-memory type file system).

Processor:

Deciding how much processing power to put in the unit is a non-trivial issue. We intend the unit to be more than a windowing terminal, but how much more use will we put it to? If we were building an ordinary workstation, we would use the Amdahl/Case rule stated above. But we are not building an ordinary workstation that will run a 'typical' mix of jobs (assuming there exists such a thing). Most of the CPU will be used by the display processor (to render PostScript, for example) and the networking code for transferring files. Therefore the CPU should be optimized for fixed-point arithmetic and bulk data transfers.

The perceived mode of accessing networked resources is a single-level store (à-la MULTICS), with a segmented and paged address space, where each 'object' occupies one segment. These segments can be demand-paged, or they can be brought it in by anticipatory-paging in advance of demand. For this to be a good model, we shall need a large address space per object (32 bits as a minimum) and a large number of segments (32- or 48-bit segment numbers). The operating system, with the help of naming/binding services, will be responsible for mapping named objects to virtual memory segments.

The number of running processes in the **PIP** will be relatively small; all that needs to run is the kernel and its daemons, the window system, an editor and a previewer, etc. It would be desirable to have the processor save its context in hardware, so that context switching can be very fast.

To sum up, we would need a 32-MIPS CPU, with a 32+32 or 48+32 bit address space

(32 or 48 bits for the segment identifier, and a 32-bit address space per segment), and 16 or 32 contexts in hardware. The CPU should be geared towards fast data transfers and scalar operations. Floating point is almost useless, so a lot of CPU power can be saved that way.

3 Firmware Requirements

The **PIP** will need some special firmware to make it a useable device. This added firmware will be responsible for:

- Power management.
- Device restarting.
- Booting from a variety of sources.
- Specialized hardware support for system and user software.

In a personal workstation, it is often the case than only a small subset of its circuitry is being used. For example, while the user is staring at the screen thinking what he should type next, there is no reason to have the processor running; rather, the processor can enter a 'standby' mode, and be restarted when an interrupt from the keyboard arrives. Handheld calculators and computers use this feature a lot. Also, the user should be able to power down the machine from software.

The **PIP** should be able to boot from a variety of sources; from the network interface, the serial interface, from plug-in ROM cards, or from its non-volatile memory. These options are necessary to increase the operability of the **PIP**. Normally, one would boot (or rather, restart) from the non-volatile memory. If that gets corrupted, the user should have the ability to boot from the network interface if he is within range of a network tap, or the serial interface (using a modem, for example) or even a plug-in ROM card with specialized software.

The firmware should also support Operating System functions like checkpoint/restart. The OS should be able to checkpoint itself and save its contents in non-volatile memory, or transfer them to a network server. The firmware should be able to give the OS the necessary support, e.g., the ability to save and restore the contents of the MMU, probe, save and restore the state of peripheral devices, etc.[5].

Portions, if not the entirety, of the system memory should be either battery-backed or saved to some non-volatile medium and then restored.

Another area that can benefit from firmware support is the display. That could be a PostScript display coprocessor, e.g., a chipset with hardware stack and either PostScript in hardware or a RISC instruction set geared to running a PostScript interpeter efficiently. Note that this is not strictly speaking necessary; if the processor is fast enough, it can be

[5]For a description on how to checkpoint a kernel, see e.g., Ioannidis and Maguire, "Checkpointing and Restarting a Unix-like Kernel", Technical Report CUCS-xxx-90

responsible for running the display system. Since the **PIP** is a personal machine, the user will be waiting anyway for the output to be displayed, so it does not matter whether it is the CPU or a co-processor that is doing the display processing. We discuss the reasons behind PostScript as a display primitive in the following section.

4 Operating System

In selecting an operating system for the **PIP**, we have to consider the class of programs that will run on it, and what facilities they will need. Let us consider some alternatives:

1. No operating system. Just run programs on the bare metal
2. A small, single-tasking OS like MS-DOS.
3. A real multi-tasking OS like one of the many Unix variants.
4. A custom-made OS specifically tailored to run notebook-type applications

Now, since we do not know what the partition of tasks should be between the infrastructure and the **PIP** (remember that one of the premises of the **PIP** design is that it depends on the existence of an infrastructure), we want a real operating system. This rules out the first option. That is, we do not want to hardwire all the applications the way it is done with calculators.

We want our users to be able to do more than one things at any time, hence we want a multitasking environment. Even an environment like the Macintosh Multifinder is not acceptable, since it does not allow true multitasking. In short, we want to provide a true multitasking operating system.

Like any other modern operating system, our OS shoule provide segmented virtual memory, efficient and secure inter-process communication, a network file system (or some other general scheme for accessign networked resources). In addition, we want a modular OS where we can remove unnecessary portions of it without affecting its functionality. Candidates for our OS include CMU's Mach (or the OSF kernel). The Plan-9 operating system, under development at AT&T Bell Labs, is being developed for a similar setup (a large number of personal workstations dependent on an infrastructure), even though the workstations are not meant to be portable and mobile.

5 Networking Capabilities

The **PIP** relies heavily on the existence of a storage, computation and communication infrastructure. It goes without saying that a significant amount of effort must be put in providing the right networking capabilities. The network interface should be fast enough so that network resources can be accessed in as little time as possible.

- On the physical/circuitry layer:
  - Attempt to increase the bit rate, if necessary by running multiple 'cells' simultaneously. (investigate what can be done without upsetting the FCC).
  - Collect data on how 'strong' the signal is in the cell, so that we can tell if, for example, we are in a null and by moving a foot to the right we can get a better signal. Also see (3) below.
  - Even if it only transmits on one channel/cell/whatever at a time, include the ability to listen to more than one (can this be done without doubling, tripling etc. the number of correlators?) We need this so that
    1. we can actually snoop on more than one cell
    2. be able to detect the presence of a second cell
    3. with the data collected above, be able to decide when it's time to switch cells (and kick-in our handoff-protocol negotiations).
  - needless to say, make all these data available on chip registers!
  - We should be able to dynamically change the spreading code without reinitializing the whole circuit (like we have to do with the arlans).
- On the data-link layer.
  - No attempt should be made to provide a reliable data-link level. even if they do, we should be able to turn it off. If a packet gets lost, let the network or the transport layers deal with it.
  - Packet addressing is probably best done by altering the spreading code and not by including an src/dst address header in each packet. This way:
    * When we are interested in unicast (NOT point-to-point) connections, i.e., when we are not in a 'coherent traffic' situation, we are not interfering with each other's traffic. Obviously, if the 'central' machine is to handle this kind of traffic, it should have a circuit dedicated to each spreading code.
    * Multicast/broadcast communication is implemented by selecting the same spreading code to listen to on many/all machines. In fact, unicast is the multicast case with only two stations in the multicast group.
  - This seems to violate our arguments against centralized service, but this is not the case:
    * Most of the time, all machines are on the same ('broadcast') spreading code, using the IP (or other) addresses in the headers to decide whether the packet is for them or not.
    * When large transfers have to take place, the participants can temporarily switch to a different spreading code so that they won't interfere with the normal operation of the network.

  - This scheme would work best if the mobile stations have the ability to be on two or three spreading codes simultaneously, (one for the regular, broadcast traffic, one (or more) for any heavy-traffic transfers, and one listening for the presence of adjacent cells and whether it's time to switch to a new cell, as described above.
  - The base-station would, of course, have to have the ability of listen to and send on more spreading codes, so that it can always be on the broadcast channel, plus participate in more than one heavy-duty transfers.
  - Each base station cell could have a 'beacon' on one of four preallocated spreading codes, so that mobile stations trying to decide whether they are near a stronger cell. (why four? well, assuming a planar topology and sufficient distances, this is graph coloring.)

- On the network layer.
  - Not much here, except that we would have to define additional control protocols (cooperating with IP) to decide when/how to do cell handoffs.
  - IP already provides the notion of a multi-homed host (a machine with more than one IP addresses), so dealing with more than one 'virtual network interfaces' (the various spreading-code groups that a machine can be in) should not pose a problem. Either that or use the notion of multicast host groups.

While we are on the subject of handoffs, we probably **must** handle these on the network layer and do something about dynamically changing the IP addresses. The reason for this it is that Internet routing is based on the (network , host) structure of IP addresses and routing is done only based on the network part (except on connected networks when you can have per-host routes). This would be a problem only when you move off of a campus (for example: taking a **PIP** from Columbia to Penn), and protocols will have to be developed to handle cases like this.

6 Research Activities

An effort such as the Personal Information Protal can spawn a number of research activities.

We need to come up with higher-density, lower-power devices in order to pack more processing and memory power in smaller space. An effort like the **PIP** can benefit chiefly from advances in low-power and non-volatile memory. At the circuit level, issues like 3-dimensional VLSI chips, wafer-scale integration, multi-chip carriers, direct bonding of chips onto the circuit boards, and so on. The goal is to reduce the number of interconnects and the power consumed.

Display technology is crucial; most of the information is conveyed to the user via the display. Liquid-crystal technology can be pushed at least an order of magnitude. The problem with LCDs though, is that a linear increase in each dimension results in a quadratic increase in the number of pixels. A scanning-mirror display, on the other hand, only needs a linear increase in the density of the scan-line, with a corresponding linear increase in the accuracy of the scanning mirror.

In terms of systems software, Issues to explore include: prefetching of executables and data files. Read-ahead, write-behind, logging/journaling for data integrity. Interaction of memory with filesystem. Access to replicated resources (this is more of a naming/binding issue). Systematically deciding what functionality belongs to the infrastructure and what to the portable. For example, in deciding what kind of display software to use, should we have a high-level command set like NeWS, or a set of low-level primitives like X? The answer to that is not simple, but preliminary results show that the deciding factor should be the minimization of network traffic. Also, in a very large paged/segmented address space (32+32 bits or 48+32 bits), it makes more sense to go for a single-level store a-la Multics. Again, we need a naming/binding mechanism to assign names of network resources ("objects") to particular segments in our machine.

7 Conclusions

We have described what we consider to be the future in personal computing; that it is both feasible and desirable for information and computation services to be accessed through a highly sophisticated Personal Information Portal. We showed that most of the technology for such a device, be it hardware technology, systems software technology, or user applications, is either already here or arriving soon.

Nevertheless, additional effort is needed to push the limits of current state of the art to make the **PIP** not only an intellectual curiosity but an mass-produced everyday appliance. The research effort to push this current state is very diverse. It is beyond the means of any individual research group, with the possible exception of large, multi-national companies. However, it can form the framework for a research initiative in all these areas.