CITI Technical Report 91–5

Taking a LITTLE WORK Along

Peter Honeyman
honey@citi.umich.edu

ABSTRACT

The continuing micro-miniaturization of components has moved high-powered, microprocessor-based machines from the desktop, to the laptop, to notebook-sized, and now to palmtop computers. These machines are distinguished in their hardware technology, but supporting software has not kept pace: the predominant operating system on such machines is MS-DOS, absent integrated support for distributed computing. With the LITTLE WORK project, I propose to close this gap in the information technology environment.

The LITTLE WORK prototype will be a notebook computer well-endowed with memory and local disk. It will run the Mach operating system and an AFS cache manager, operating predominantly in a dataless mode. The network interface will be the serial port, attached to a fixed or cellular phone attached to a high-speed modem.

To economize on limited network bandwidth and substantial cellular phone charges, AFS will be engineered to support compressed headers and to operate in a disconnected mode. Other technical challenges abound, such as congestion avoidance and control for AFS, application-level support for network reconfiguration, dynamic IP address assignment, X windows over VGA, operating system support for battery power management, *etc*.

The underlying thesis of the LITTLE WORK project is that mobile computers are capable of supporting the kind of distributed computing environments common in academia and industry. The LITTLE WORK prototype will make a powerful statement in what is achievable today. Furthermore, it positions CITI and its partners to take advantage of further enhancements in computing technology: faster notebook computers, better screens, denser memory and disks, digital cellular communications, *etc.*

August 28, 1991

Taking a LITTLE WORK Along

Peter Honeyman
honey@citi.umich.edu

COMPUTING ENVIRONMENTS TODAY

Most computer scientists work with two vastly different computing environments: the computers on their desks at work and the remote setups for access from home. The work computer may be a fully-functioning workstation, say a 10+ million instructions per second (MIPS) computer running the UNIX† operating system, an IBM PC (or clone), an IBM PS/2, or a fast Macintosh II.

The work machine is a full-fledged member of a larger distributed computing environment, with support for distributed filing, printing, and electronic mail, as well as transparent access to highly-capable compute servers. These distributed services are provided through a suite of network protocols, typically layered on top of the Internet Protocol (IP) suite. Data communication is usually provided by Ethernet, offering 10 million bits per second (Mbps) of bandwidth.

In contrast, access to the computing environment from home sets a much lower standard, relying on a modem over a voice-grade line. This limits communication to something between 1.2 and 14.4 Kbps, with 2.4 Kbps most common these days. The conventional wisdom holds that even 14.4 Kbps is inadequate for full-functioning network access to the office computing environment. So the scientist is relegated to a home environment based on a dumb terminal, or a terminal emulator running on a workstation of some sort, with serial-line access to the ‘‘real’’ computing environment. The vast disparity between the home and office environments makes location transparency an impossible dream.

MY HOME COMPUTING ENVIRONMENT

Since January 1990, I have had nearly identical home and work computing environments. The principal components of these computers are an IBM RT/115, a three MIPS computer with 12M of memory and a 70M disk, running Berkeley UNIX; the X Window System [1]; and Transarc’s AFS (formerly, Carnegie-Mellon University’s Andrew File System) [2]. The computers are ‘‘dataless,’’ *i.e.*, the only files stored permanently on the local disk are those necessary for bootstrapping when the machine is turned on and some administrative applications.

From work, network access is provided by an Ethernet connection. From home, the network is a serial line IP (SLIP) connection over a pair of modems running the V.32 protocol at 9.6 Kbps. These modems use an ordinary phone line, which costs me about $13 per month.

Local Disk Caching

AFS plays a critical role in supporting remote access to the file system: without local disk caching, access to files would be prohibitively slow, around 850 bytes per second after accounting for protocol overhead. For illustration, a typical UNIX file is 11K bytes [3], which translates into a dozen seconds or so over SLIP. If file accesses were forced to wade through the communications network on every read or write, the delays that ensued would be unbearable. It would be inconceivable to rely on NFS [4] over a slow link. Thanks to AFS, though, most read and execute accesses are satisfied from the local disk cache, vastly reducing the pain of the slow SLIP line.

In addition, I have eased into patterns of use that avoid local cache misses and other network expenses:

- Instead of opening XTERM[1] windows on remote machines that ‘‘call back’’ to my home machine, I open XTERM windows locally and TELNET[2] to the remote machines.

† UNIX is a Trademark of AT&T Bell Laboratories.

1 XTERM is the character terminal emulator provided with the MIT X distribution.

2 TELNET is a virtual terminal protocol in the IP

This keeps X events (mouse selections, window exposures) local.

To appreciate the importance of this, consider a single character typed through a TELNET session. This produces one byte of TCP data sent to the remote host and one byte returned as the echo.[3] With a remote XTERM, though, the keystroke turns into a 32 byte message sent to the XTERM client, followed by 60 bytes of image data returned to the X server for the character echo. Remote XTERM is vastly more expensive in network resources than the TELNET connection!

- I don't compile kernels on the home computer, since this would store several megabytes of object code. Instead, I TELNET to the work machine and build there.
- I run a window-based editor called `sam` [5], which has an option that is very useful for my home environment. `Sam` runs in two processes: a file manager, and a display manager. These pieces do not have to run on the same host. Furthermore, `sam` was designed with a slow connection between the two pieces in mind. So I run the display manager on my home computer, and the file manager on my work computer.

  `Sam` is really great: the display manager requests only the portion of the file that it wants to show, and caches it to boot! Once the display cache is warmed up, moving around in the file is very fast. And because the file manager runs remotely, saving the file is equally fast. (This paper was written at home with `sam`.)

TCP header compression [6] goes a long way toward making TELNET over SLIP bearable; without header compression, each keystroke becomes a pair of 41 byte messages: 20 bytes each for TCP and IP, and one byte of TCP data. With header compression, the headers shrink to three or four bytes.

Macintosh Support

My home computing environment also includes a Macintosh, handy for some applications not available on UNIX (EXCEL, POWER POINT, and WORD). The Macintosh is also a playground for my children.

To integrate the Macintosh into the larger computing environment, I rely on some software developed at the University. Wes Craig, in the Research Systems group, added the AppleTalk [7] address family to the Berkeley UNIX kernel, supporting network and transport layers. Wes and Mark Smith, also in Research Systems, developed AppleTalk Filing Protocol (AFP) and AppleTalk Printer Access Protocol (PAP) daemons, which communicate through sockets in the AppleTalk domain.

I run these tools on my home UNIX box, so from my home Macintosh, I am able to ''mount'' an AppleShare volume stored in the UNIX file system. Of course, the volume I mount is in my home directory, which is itself in AFS. This gives me access to the same files from home and from work, from UNIX and from the Macintosh. Once again, thanks to AFS caching, access times for reading are very fast, comparable to a local disk.

To support printing, the PAP daemon lets me spool print requests from the Macintosh to the UNIX machine and print them on any PostScript printer accessible to the Berkeley UNIX line printer software. Ordinarily, the printer I use is an IBM Personal PagePrinter (with PostScript) attached to the home RT. However, the PAP daemon is configured to advertise several remote printers, and I can select one of these with the Macintosh Chooser.

The UNIX AppleTalk stack does not yet route across SLIP, so I am not connected to the larger campus AppleTalk network. Nonetheless, access to distributed filing and printing services through the AFP and PAP daemons satisfies most of my remote service needs, and provides tight integration of two widely disparate network protocol stacks.

IP Routing

My home IP routing scheme, depicted in Figure 1, is some jerry-rigged to some extent. Campus network services are provided by the University's UMNET organization. It owns the IP routers, is responsible for maintaining the routes and the IP address space, and manages the gateways to the Internet.

To secure the campus network, UMNET configures its routers to ignore route updates from routers beyond its control. At my request,

suite.

[3] There is additional overhead for the TCP and IP headers, as well as an occasional acknowledgement packet, but these costs are about equal for the XTERM and TELNET cases.

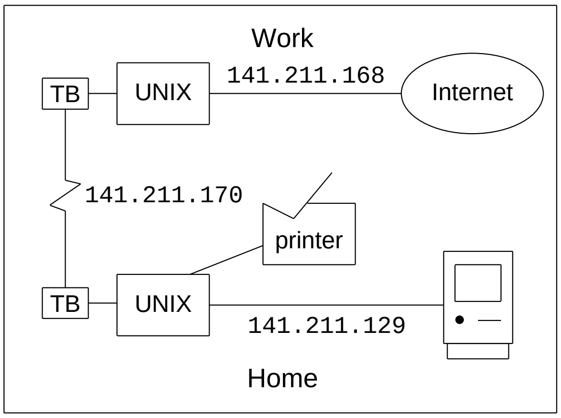

Figure 1

UMNET configured their routers to show a static route from my office computer to a SLIP subnet (141.211.170). I then added hosts on this subnet: one for the office machine, and one for the UNIX box at home. I also installed a home Ethernet so that I could get to distributed services from my home Macintosh; this also required coordination with UMNET.

My reliance on static addresses and routes is not entirely satisfactory. When I have some time, I plan to explore the CSNET software that negotiates things like IP address assignment. Someday I will run the Point-to-Point Protocol (PPP) [8], which supports dynamic address assignment.

THE LITTLE WORK PROJECT

I am occasionally confronted with week-long business trips. To stay in touch with my students and colleagues, I lug along a portable Macintosh and a 2.4 Kbps modem. Each evening, I dial into my home computer to keep up with my electronic mail. Even though I use fairly modern software that supports a multiple window environment, it pales when compared to the style of computing to which I am accustomed. I am stunned by the realization that this is the standard home computing environment for most of my colleagues.

Outstanding as my home computer has proven to be, it suffers a major inconvenience: all those wires. I want to hang up the phone, turn off the juice, grab the box, and hit the road! For this, the system must be self-contained and portable, and the network connection must be wireless.

In a few years, high-powered, networked, portable computers will be commonplace. But why wait? Whether traveling across town, across the state, or across the country, it should be possible to take a LITTLE WORK along. I don't say this standing on a visionary's soapbox, referencing Vannevar Bush [9] or Alan Kay [10]. I want the LITTLE WORK machine now, or at least this year. And I don't want it to do much beyond what I do now, principally system building, document preparation, and electronic correspondence.

The LITTLE WORK machine should make no compromises except where required by space, mass, and power considerations, *e.g.*, while a 19 inch screen (or more) is *de rigueur* on the desktop, it is impractical for a portable machine. Typical off-the-shelf components in today's marketplace in fixed and portable computers are shown in the following table.

	Fixed	Portable
MIPS	20 – 50	2 – 10
Display	1000×1000	640×480
Disk	100 – 500 MB	20 – 100 MB
Memory	8 – 32 MB	1 – 16 MB
OS	UNIX	MS-DOS
Comm	10 Mbps	< 2 Mbps

With these constraints in mind, I will next describe related work, and then outline the components of the prototype LITTLE WORK machine.

Student Electronic Notebook

The Student Electronic Notebook project [11, 12] is a joint Columbia University/IBM project. The goal is to produce a portable device that can serve students' needs for textbooks, class notes, handouts, *etc.*, as well as offering access to the University's information technology infrastructure. In many respects, the SEN project is similar to LITTLE WORK, with the major differences in communications technology and file system.

The SEN aims for a high-bandwidth network connection and employs spread-spectrum technology in the 902 to 928 Mhz band. This offers data rates between 230 Kbps and 2 Mbps. However, the wireless network has a restricted range, about 150 feet. This can be extended to about six miles with line-of-sight between a pair of antennae.

The SEN is diskless, so all filing and paging is network-based. Early reports suggest that the data rates supported by SEN are adequate. The file system protocol being developed for SEN does not support caching [13].

Coda

Coda [14] is a distributed file system that addresses data unavailability owing to server failure or network partition. Coda uses AFS, Mach, and Camelot as a base. The principal approach used by Coda is server replication, but more interesting, from our perspective, is its use of an optimistic strategy to accommodate disconnected operation.

A portable client out of touch with the network is viewed as an instance of *voluntary* network partition. Here, Coda allows the client to operate on its cached files without requiring server approval. LITTLE WORK is precisely the kind of platform for which Coda was devised.

BEHEMOTH

The BEHEMOTH[4] project is rather difficult to describe, but fun. At its heart is a custom-built, eight-foot recumbent bicycle pulling a four-foot trailer, also hand-crafted. Included in the 350 pounds of gear are 80+ watts of solar panels charging 30 amp-hours of lead-acid batteries. These drive a Macintosh Portable, a 16 Mhz PC/AT clone, a Toshiba T1000, a Sun IPC SPARCstation, and dozens of embedded microprocessors.

For communications, BEHEMOTH relies principally on a 12 inch diameter, 7 inch tall, 14 GHz radome that tracks a communications satellite in geosynchronous orbit. This provides a data rate in the low hundreds of bps. While this is certainly inadequate for dataless computing, BEHEMOTH offers the ultimate in mobility!

For more information on BEHEMOTH, send mail to `wordy@bikelab.sun.com.`

THE LITTLE WORK PROTOTYPE

The remainder of this paper describes joint work by the author and his colleagues: Jim Rees, senior systems programmer at the Center for Information Technology Integration (CITI); Larry Huston, a graduate student in the Electrical Engineering and Computer Science department and a member of staff at CITI; and Dave Bachmann, a doctoral candidate in EECS and a CITI staff member.

The prototype LITTLE WORK machine will be built from conventional, off-the-shelf hardware. The software base is also pretty ordinary, consisting of the sort of components people use to build today's distributed systems.

Computer

The RT/115 weighs a ton and consumes 7.5 amps of AC power; it is far from a portable machine. Its replacement will be a battery-powered laptop or notebook computer.

An attractive mobile system is the IBM PS/2 Model L40, based on a 20 Mhz 80386SX, about four MIPS. It has a 640×480 VGA screen, just (barely) enough resolution to run X, and can hold up to 18 Mbytes of RAM, more than we need. The disk, 60 Mbytes, is big enough (again, barely!) to accommodate a healthy swap space (necessary for X!) and a 20M cache.

We are looking at other portable platforms, but this is a place where some compromises will have to be made. For example, while the SONY NEWS machine has a dynamite screen, it requires AC power and uses the MIPS R3000 CPU, with which we have little experience at CITI. Recent SPARC-based portable systems are also strong players, but are not part of our plan for the near-term.

Communications

Mobile network technologies range from wireless communications to satellite uplinks. Wireless communication offers high bandwidth, up to a few megabytes per second, but suffers from low mobility, with about 150 foot range. On the other hand, a satellite uplink provides world-wide mobility, but can sustain only a few hundred bps. Nestled between these is the option of connecting a modem to the serial port and communicating over a cellular phone; this will be LITTLE WORK's data lifeline.

Word has it that V.32 does not survive over cellular, but we plan to experiment with it anyway. When we've had enough, we will run PEP, Telebit's half-duplex protocol, over a cellular line. PEP simulates a full-duplex line by changing the direction of data flow every so often. In data-streaming mode, PEP can sustain upwards of 10 Kbps, comparable to current throughput to my home computer. However, without a fair amount of engineering, PEP's ‘‘ping-pong’’-ing can severely hamper good throughput.

PEP wants packets ‘‘against the flow’’ (principally acknowledgement packets) to be short and

[4] Big Electronic Human-Energized Machine... Only Too Heavy

infrequent. The forward traffic wants to be big enough to fill a window of ‘‘long’’ packets. After the window is transmitted, PEP requires that the line turn around briefly. In the turnaround interval, one or two ‘‘micro-packets’’ can be sent back without diminishing the forward data flow. Thus, to maximize PEP throughput, we must squeeze the acknowledgement packets into a micro-packet and send them infrequently. Accordingly, Larry Huston is developing compression techniques for Rx [15], the RPC package used by AFS.

There is a (rumored) product called Outback, which has a battery operated cellular phone and a PEP modem, in a package comparable in size and weight to a notebook computer. While this latter fact is unfortunate — we would prefer something on a card that plugs into LITTLE WORK — it represents today’s state of the art and achieves our mobility objectives. In the future, we hope that digital cellular offers some interesting options.

Naturally, there will be many occasions when fixed physical networks are available, *e.g.*, at home (dial-up) or on campus (Ethernet or possibly wireless), and they will be employed in preference to the cellular connection whenever possible.

File System

A distributed file system (DFS) is essential for a machine with limited permanent storage. In addition, a DFS allows system administration to be centralized, so that the contents of disks on DFS clients don’t require backup. As mentioned earlier, anticipated limitations in network bandwidth mandate local disk caching for good overall performance. This rules out NFS, but AFS fits the bill quite well.

Coda is an obvious candidate, but is not yet publicly available. We have a lot of experience with AFS, both as users and as developers, so we plan to stick with that. We intend to add support for disconnected operation to AFS, using lessons learned from the Coda project as a guide.

A serious problem with AFS is the inability of Rx to handle congested networks — placing a 9.6 Kbps link into an otherwise high-speed network is a sure prescription for congestion. This is another area where we are developing the means to make Rx better match the PEP environment: Dave Bachmann is enhancing Rx, principally by following Van Jacobson’s prescription for congestion avoidance and control [16] and the recent recommendations for establishing the packet frame length [17].

Operating System

It’s a good bet that we’ll be running an Intel 80386 CPU, possibly among others. The ’386 can run any of a number of operating systems: AIX, OS/2, BSD, Mach, *etc*. Most of the work we’ve done so far carries over easily to either BSD or Mach, but BSD doesn’t support AFS or Coda, while Mach supports both, so Mach is the clear choice here.

LITTLE WORK PILOT PROJECT

Over the next six months, we will acquire the necessary components and build a prototype LITTLE WORK machine. Jim Rees, Larry Huston, and I are hard at work developing the software for the prototype. Here is a list of tasks we have planned for the project and their status at this writing.

- Mach/AFS/X on stock 386

  We have Mach 2.5 and AFS 3.0 running on a Zenith ’386 clone, and are working on upgrading to AFS 3.1. Work on the X server will begin soon.

- Mach/AFS/X on notebook

  Two IBM L/40 notebooks are on order. As soon as they arrive, we will move our software to them.

- PEP modifications to Rx

  Telebit has provided us with several Trailblazer modems, a Netblazer router, and access to their technical staff for consultation. We have built a PEP workbench for testing our Rx modifications. Our focus is on Rx header compression; so far, we have managed to squeeze 56 byte Rx/UDP/IP headers down to seven bytes. We have begin working on an Rx acknowledgement strategy appropriate for PEP communications.

- Congestion control for Rx

  Dave Bachmann has modified the retransmission algorithms in Rx according to Jacobson’s recommendations, and has incorporated Karn’s algorithm [18]. We have installed support for MTU discovery in our kernels, and are looking at slow-start strategies.

- Disconnected operation for AFS

  We plan to begin working on this in a few months.

- Dynamic IP address assignment, PPP

  This task is on the back-burner, while we focus on building the LITTLE WORK prototype.

- Managing network interfaces

  To economize on cellular phone charges and to support network reconfiguration, we plan to develop some applications that give the user fine-grained control over connection establishment and frequency. We plan to build in Tk [19] for its ease of programming and snappy look-and-feel. This work will begin after we get a better feel for the degree of control needed to manage the network.

In summary, we have a vision firmly in our grasp and are making progress toward making it possible to take a LITTLE WORK with us wherever we go.

ACKNOWLEDGEMENTS

I thank Telebit for donating data communications equipment to CITI. This work was partially supported by IBM.

REFERENCES

1. R.W. Scheifler and J. Gettys, ''The X Window System,'' *ACM Transactions on Graphics* **5**(2), pp. 79–109 (April, 1987).
2. J.H. Howard, ''An Overview of the Andrew File System,'' pp. 23–26 in *Winter 1988 USENIX Conf. Proc.*, Dallas (February, 1988).
3. J. Ousterhout, H.L. DaCosta, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson, ''A Trace-Driven Analysis of the Unix 4.2 BSD File System,'' *Proc. of the 10th ACM Symp. on Operating System Principles*, Orcas Island (December, 1985).
4. Sun Microsystems, Inc., ''NFS: Network File System Protocol Specification,'' RFC 1094, Network Information Center, SRI International, Menlo Park, CA (March 1989).
5. Rob Pike, ''The Text Editor `sam`,'' *Software — Practice and Experience* **17**(11), pp. 813–845 (1982).
6. V. Jacobson, ''Compressing TCP/IP Headers for Low-Speed Serial Links,'' *Internet Request for Comments*, Menlo Park, CA(1145), Network Information Center, SRI International (February 1990).
7. G.S. Sidhu, R.F. Andrews, and A.B. Oppenheimer, *Inside AppleTalk,* Addison-Wesley, Reading (1989).
8. D. Perkins, ''The Point-to-Point Protocol for the Transmission of Multi-Protocol Datagrams Over Point-to-Point Links,'' RFC 1171, Network Information Center, SRI International, Menlo Park, CA (July 1990).
9. Vannevar Bush, ''As We May Think,'' *The Atlantic Monthly*, pp. 101–108 (July, 1945).
10. Alan Kay and Adele Goldberg, ''Personal Dynamic Media,'' *IEEE Computer* **10**(3), pp. 31–41 (March, 1977).
11. John Ioannidis and Gerald Q. Maguire Jr., ''PIP-1: A Personal Information Portal with Wireless Access to an Information Infrastructure,'' Tech. Report CUCS-055-90, Columbia University (1990).
12. John Ioannidis, Gerald Q. Maguire Jr., Israel Ben-Shaul, Marios Levedopoulos, and Micky Liu, ''Porting AIX onto the Student Electronic Notebook,'' Tech. Report CUCS-042-90, Columbia University (December, 1990).
13. John Ioannidis and Gerald Q. Maguire Jr., ''The Coherent Trivial File Transfer Protocol,'' Tech. Report CUCS-043-90, Columbia University (1990).
14. Mahadev Satyanarayanan, James J. Kistler, Puneet Kumar, Maria E. Okasaki, Ellen H. Siegel, and David C. Steere, ''Coda: A Highly Available File System for a Distributed Workstation Environment,'' *IEEE Transactions on Computers* **4**(39), pp. 447–459 (April, 1990).
15. ''Rx: Extended Remote Procedure Call,'' *Proc. of the Nationwide File System Workshop*, Pittsburgh, Information Technology Center, Carnegie-Mellon University (August, 1988).
16. V. Jacobson, ''Congestion Avoidance and Control,'' *Proc. ACM SIGCOMM '88*, Stanford, CA, pp. 314–329 (August 1988).
17. J.C. Mogul and S.E. Deering, ''Path MTU Discovery,'' RFC 1191, Network Information Center, SRI International, Menlo Park, CA (November 1990).
18. P. Karn and C. Partridge, ''Improving

Round-trip Time Estimates in Reliable Transport Protocols,'' *Proc. ACM SIGCOMM '87*, Stowe, Vermont, pp. 2–7 (1987).

19. John K. Ousterhout, ''An X11 Toolkit Based on th Tcl Language,'' pp. 105–116 in *Winter 1991 USENIX Conf. Proc.*, Dallas (January, 1991).