You and I are Past Our Dancing Days

Position Paper for the 6th European SIGOPS Workshop, 'Matching Operating Systems to Application Needs'

Sape J. Mullender
University of Twente
Faculty of Computer Science
Enschede, Netherlands

April, 1994

Operating systems have grown in size and functionality. Today's many flavours of Unix provide a multi-user environment with protection, address spaces, and attempts to allocate resources fairly to users competing for them. They provide processes and threads, mechanisms for synchronization and memory sharing, blocking and non-blocking system calls, and a complex file system. Since it was first introduced, Unix has grown more then a factor twenty in size.

Several operating systems now consist of a microkernel, surrounded by user-space services [Accetta et al., 1986; Mullender et al., 1990; Rozier et al., 1988]. Together they provide the functionality of the operating system. This operating system structure provides an opportunity to make operating systems even larger.

The trend for operating systems to grow more and more baroque was signalled more than a decade ago [Feldman, 1980], but has continued unabated until, today, we have OSF/1, the most baroque Unix system ever. And we have Windows/NT as a demonstration that MS-DOS also needed to be replaced by something much bigger and a little better.

In this position paper, I am asking what community we serve with our operating systems research. Should we continue doing this, or can we make ourselves more useful to society and industry by using our experience in operating systems in new environments.

I argue that there is very little need for bigger and better operating systems; that, in fact, most CPUs will never run an operating system at all; and that our experience in operating systems will be better applied to designing new generations of distributed and ubiquitous applications.

Why will this happen? Largely because most processors will not be general-purpose, user-programmable workstations that need an operating system. Instead,

most processors will be — and are, in fact, already — incorporated in special-purpose devices built by the consumer-electronics industry. But, unlike today's devices, they will be connected in global networks of information-service markets and they will have to participate in communication protocols and execute distributed algorithms.

Processors dressed up as a computer already form a minority. Processors are built into all sorts of devices, from Airbuses to washing machines.[1] Some of these will participate in what we would call distributed systems. Personal digital assistants — pocket computers — will become popular since they will provide access to the system while away from the office. Smart cards will carry out the authentication functions needed for access to a distributed system. TV-sets will become viewing stations not only for televised broadcasts, but also for watching movies (video on demand) and for browsing through multimedia databases.

All these systems do not need an operating system of the traditional sort, but something much smaller, simpler and more efficient.

General-purpose computers still need an operating system, but most of them only serve as a private computing resource for their owner, so they do not need a multi-user operating system.

On shared networks of private machines, operating systems are not essential for realizing security. Distributed systems built out of personal computers can be as secure when the individual machines use a run-time system such as MS-DOS as when they run a multi-user operating system such as Unix. File servers, although they are shared by a user community, only run trusted code and, therefore, do not need a multi-user operating system either.

Multi-user systems need an operating system to arbitrate the sharing of the system's resources among its users. Users compete for processor cycles, memory, and I/O devices. The operating system must enforce a fair allocation.

On private (but multiprogrammed) machines, there is also competition for scarce resources, but this can be resolved by cooperation rather than competition. Derek McAuley came up with a useful analogy: Applications sharing a window system can, in principle, take over the screen or the colour map; they normally don't, because, if they did, nobody would run them. Collaborative resource allocation does not require a separate operating system. Anderson et al. [1991] has alread shown that scheduling can actually be done more effectively when it is not done just by the operating system.

Coping with continuous media is creating new styles of resource allocation. Continuous media systems share conflicting needs with real-time systems and time-sharing systems. They require real-time performance, but do not have an *a priori* bounded load. Multimedia applications must adapt to what resources are available. The parameters used to make this adjustement are called Quality-of-Service (QoS) parameters. Applications will need to work out the allocation of resources for an acceptable QoS collaboratively — and again, there is no vital function for the operating system here.

[1] I've been told that even electric razors will soon have a processor in them. I did not find out what these processors will do.

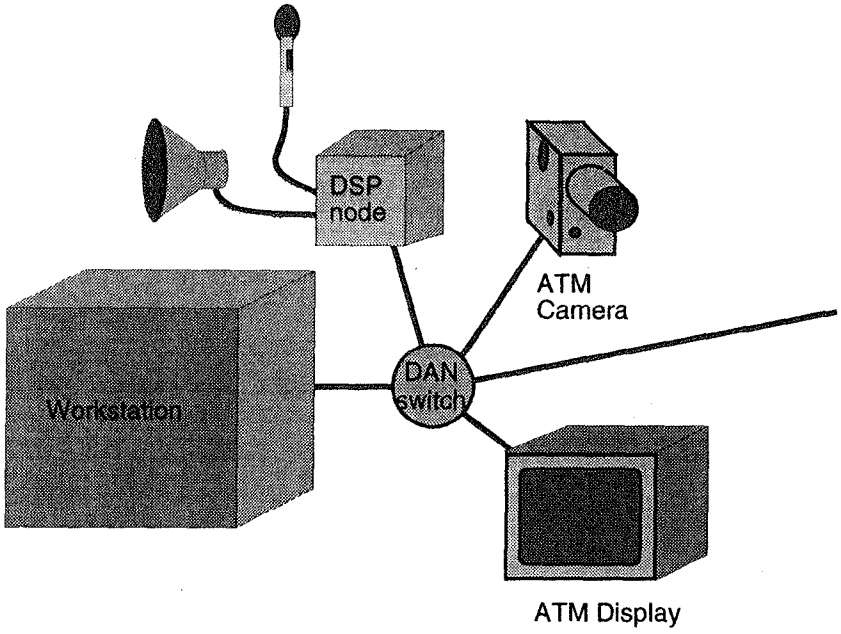

Figure 1: Architecture of the multimedia workstation

The support of multimedia, pocket computers, wireless networks, and smart cards is beginning to bring about new systems architectures. In our Pegasus[2] Project, multimedia devices are directly connected to the network [Mullender, Leslie and McAuley, 1994]. In the DANOS project of the University of Cambridge, even the computer is built out of networked components [Hayter and McAuley, 1991]. Some projects are building high-performance file servers by using 'networked disks' [Hartman and Ousterhout, 1993].

Figure 1 shows an important aspect of the Pegasus architecture — the target end-system architecture. The figure shows a conventional workstation and its network interface connected to an ATM switch. However, also connected to the switch, we see a camera device, a display device, an audio device, and then the rest of the ATM network. The switch is under control of the workstation; that is, all connections through the switch are managed by the workstation, so that the workstation is also in control of the multimedia devices.

In this architecture, when video flows from a camera in one system to a display in another — as is the case in video-phone and video-conferencing applications — no processors need to process any video data. This goes for the audio data too, of course. Hence the processors in the workstations, at both the camera and display, only need

[2]The Pegasus Project is a project of the Universities of Twente and Cambridge, supported by the European Communities' ESPRIT Programme through BRA project 6586. It is partially supported by the Cambridge Olivetti Research Laboratory and a grant from Digital Equipment Corporation.

to manage the connections and devices.

The Pegasus architecture seems to be ideally suited for implementing multimedia services extending into the consumer market. One can imagine replacing the TV set by an ATM display (with a built-in DSP node) and the remote control by a (wireless) ATM pocket computer that controls the display and the local switch.

The 'remote control' can communicate with remote services as well as local devices and set up video-on-demand sessions, but also video-phone conversations or video conferences.

Such consumer devices will be programmable by their users to a very limited extent only. They don't need an operating system; a mere run-time system will do fine. Users can buy extra applications for some of the devices; e.g., by plugging in a flash-RAM card. Applications need not be protected from each other, however; applications managing sensitive data can make sure no other applications are present on the device and, after they are done, erase or encrypt the sensitive data.

Naturally, operating systems, such as Unix, will still be useful as vehicles for program development, even when the programs being developed are for operating-system-less devices. But the developers should realize that their target environment is not necessarily a Unix-like system or even a system with an operating system. Research *using* operating systems is a fine thing, but research *on* operating systems may only benefit a very small community.

References

Accetta et al. [1986]

M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian and M. Young, Mach: A New Kernel Foundation for UNIX Development, *Proceedings of the Summer Usenix Conference*, Atlanta, GA, July 1986.

Anderson et al. [1991]

T. E. Anderson, B. N. Bershad, E. D. Lazowska and H. M. Levy, Scheduler Activations: Effective Kernel Support for the User-Level Thread Management, *Proceedings of the 13th Symposium on Operating Systems Principles*, Pacific Grove, CA, October 1991, In *ACM Operating Systems Review* **25**(5).

Feldman [1980]

S. I. Feldman, *An Architecture History of the UNIX System*, 1980.

Hartman and Ousterhout [1993]

J. H. Hartman and J. K. Ousterhout, The Zebra Striped Network File System, *Proceedings of the 14th Symposium on Operating Systems Principles*, Asheville, NC, December 1993, 29–43, In *ACM Operating Systems Review* **27**(5).

Hayter and McAuley [1991]

M. Hayter and D. McAuley, The Desk-Area Network, *ACM Operating System Review* **25**(4), October 1991, 14–21.

Mullender, Leslie and McAuley [1994]

S. J. Mullender, I. M. Leslie and D. McAuley, Operating-System Support for Distributed Multimedia, *Proceedings of the Summer Usenix Conference*, Boston, MA, June 1994.

Mullender et al. [1990]
S. J. Mullender, G. van Rossum, A. S. Tanenbaum, R. van Renesse and J. M. van Staveren, Amoeba — A Distributed Operating System for the 1990s, *IEEE Computer* **23(5)**, May 1990.

Rozier et al. [1988]
M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Hermann, C. Kaiser, S. Langlois, P. Léonard and W. Neuhauser, *CHORUS Distributed Operating Systems*, Chorus Systèmes, Report CS/TR-88-7.6, Paris, November 1988.