Distributed Operating Systems

An Overview of Current Research

Dave Taylor — tay...@hplabs.hp.com

External draft of: Monday, June 13th 1988

Introduction

In the last few decades, we've seen computers move from large, monolithic machines that allowed a single user complete access to the entire machine to large machines that allowed multiple users to have access to the machine simultaneously. Then, a decade or so ago, the next step was taken; the machines became smaller and again returned to single user computers, this time being called 'personal computers'. The final step, so far, is to tie these personal computers to a central resource system for shared disks, printers, CPU cycles, and so on; distributed computing.

Since the late 1970's distributed computing, and more specifically distributed operating systems research, has yielded an impressive amount of excellent work, moving to the forefront of computer science research areas in the university environment.

This paper, presents a succinct overview of the major distributed operating systems research going on at universities throughout the world, including the current status of the project, the operating environment, and a brief description of each system.

Appendix one is a bibliography of introductory papers not only for the projects discussed herein but also other related projects and papers of interest, and appendix two lists non-university research in the distributed operating systems area.

If you have any questions on this paper, any corrections, or any further information, please feel free to contact me at the electronic mail address listed above.

Format of this Paper

The format used herein is;

- Name of the Project
- Where the Research is Going On

- Primary Contact People
- Current Project Status
- Operating Environment
- Brief Description of the Project
- References

The references section contains references to specific citations in the appendix, and those in bold face were used as primary references for the information in this paper.

Amoeba

Name:	The Amoeba Project
Where:	Vrije Universiteit, Amsterdam
Contact:	Dr. S. J. Mullender Centre for Mathematics and Computer Science Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
Status:	Active
Environment:	Digital Equipment Corporation VAX 11/750's, Motorola 68000-based workstations (unknown vendor) and a Protean network ring.

Description: ‘‘Fifth generation computers must be fast, reliable and flexible. One way to achieve these goals is to build them out of a small number of basic modules that can be assembled together to realise machines of various sizes. The use of multiple modules can not only make the machines fast, but also achieve a substantial amount of fault tolerance.’’

Amoeba focuses on not only the use and management of the large processor-set, but also on the communications and *protection* aspects as well.

Overall, Amoeba is more of an ‘object oriented’ aproach to distributed operating systems, with the designers rejecting the traditional approach of a multilayer set of discrete processes (eg. the ISO seven layer model). Nonetheless, Amoeba is based on message-passing modules, a transaction approach to file passing (versus the more common stream).

They spurn the ISO seven layer model in favor of their own simplified, four layer model:

- The Semantic Layer: for example, what commands do specific types of processor modules understand? This is the only layer visible to users.
- The Reliable Transport Layer: resonsible for requests and replies between clients and servers — presumably this is where the transaction protocol is used.
- The Port Layer: service locations and transmission of datagrams (unreliable packet delivery) to servers. Also enforces the protection mechanism.
- The Physical Layer: deals with the electrical, mechanical, and related aspects of the network hardware.

References: **[Mullend86]**, [Mullend82]

Andrew

Name: The Andrew System

Where: Carnegie Mellon University, Pittsburgh, Pennsylvania

Contact: Dr. Alfred Spector
Information Technology Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213
email: `spec...@andrew.cmu.edu`

phone: (412) 268-6731

Status: Active

Environment: IBM RT-PCs and some Sun workstations

Description: The Andrew project is designed to be a prototype computing and communications system for universities. The main areas that are targetted by the development team are:

- Computer Aided Instruction
- Creation and use of new tools
- Communications
- Information Access

As with a number of other distributed operating systems, Andrew is a ‘‘marriage between personal computers and time-sharing. It incorporates the flexibility and visually rich user-machine interface made possible by the former, with the ease of communication and information-sharing characteric of the latter.’’

Note: no support for diskless machines in the Andrew system. Reasons: less robust system (if network is down so is machine); cost of complete server similar to cost of individual disks; individuals unlikely to purchase machines only functional on local CMU network; paging over network precludes privacy and security; difficult to support a varied, heterogeneous set of computers typically found at a university.

Andrew is based on Berkeley 4.2 BSD, partially due to one of the premises of the project: that a significant percentage of the user population will be involved in ongoing software development. [1]

The Andrew system is based on a virtual single file system called VICE (that is, a file system with global naming and a single hierarchical organization) and a workstation based application support system called VIRTUE. Typical Andrew workstations have VIRTUE running on top of BSD 4.2, using the campus IBM Token Ring Network to communication with the VICE file system. (Andrew also supports smaller personal computers with minimal functionality (yet still more sophisticated than having the PC emulate a terminal and dial-up))

The VICE file space is actually broken into two different parts; local and shared space. The

1. This is an interesting point of difference between academia and industry — in the industry customers are more interested in ‘solutions’ than in something that they’ll need to learn how to program to use.

local space is accessible to the user (typically on their own machine's disk) but inaccessible to the rest of the Andrew community, while the shared space can actually exist anywhere on the network and be accessed by anyone with the appropriate permissions.

References: **[Morris86]**, **[Nichols87]**, [Satyan85]

Argus

Name:	The Argus Project
Where:	Massachusetts Institute of Technology
Contact:	Barbara Liskov, Massachusetts Institute of Technology Laboratory for Computer Science Cambridge, MA 02139 email: `lis...@lcs.mit.edu`
Status:	Active
Environment:	unknown
Description:	‘‘Argus is a programming language and system developed to support the implementation and execution of distributed programs. It provides mechanisms that help programmers cope with the special problems that arise in distributed programs, such as network partitions and crashes of remote nodes.’’
References:	**[Liskov87]**

Cambridge Distributed System — CDS

Name:	The Cambridge Distributed Computing System
Where:	The University of Cambridge, England
Contact:	Dr. Roger Needham or Dr. Andrew Herbert Computer Laboratory The University of Cambridge Cambridge, England
Status:	Presumed to have been completed
Environment:	Unknown, but probably based on Xerox machines and some sort of personal computing devices.
Description:	The Cambridge Distributed System is of great interest for a number of reasons, including its being based on the Cambridge Digital Communications Ring, a 'slot ring' over twisted pair wires. Another item of interest is that the system is built of a virtual processor bank, and when a user connects to the system (via a terminal concentrator through a resource management system) they are assigned a certain number of actual CPUs that remain theirs throughout the entire session. What's interesting about this approach is that it neatly solves a couple of traditional problems in distributed computing; namely process migration and utilisation of multiple processors by a single task. It also allows a network that has *n* possible users to have significantly less than *n* processors available, that actual amount based on the peak demand need on the system.
References:	**[Needham82]**

DASH

Name: The DASH Project

Where: The University of California at Berkeley

Contact: Dr. David Anderson or Dr. Dominico Ferrari,
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA, 94720.
email: `ander...@arpa.berkeley.edu` or `fera...@arpa.berkeley.edu`

Status: Active

Environment: Sun 3 workstations.

Description: DASH is designed to be a Very Large Distributed System (eg. one that is numerically, geographically, and administratively distributed, offering access to non-local resources and is transparent (no syntactic changes for local versus remote access, and relatively minor performance degradation))

‘‘The following are some of the principles for VLDS design that we have arrived at . . . The DASH prototype incorporates all of these principles:

- separate the levels of network communication, execution environment, execution abstraction, and kernel structure, and provide an open framework where possible.
- Use a hybrid naming system using a tree-structured symbolic naming for global permanent entities, and capabilities to communications streams for other entities.
- When possible, put communication functions such as security and inferface scheduling at a *host-to-host* rather than *process-to-process* level, and consolidate these functions in a *sub-transport* layer.
- Provide flexible support for stream-oriented communication.
- Provide a service abstraction that allows for replication, local caching and fault-tolerance, but does not directly supply them.
- Support real-time computation and communication at every level.

References: **[Anders87]**, [Anders87/2]

DEMOS/MP

Name: The DEMOS/MP Distributed Operating System

Where: The University of Wisconsin

Contact: Dr. Barton Miller
Computer Sciences Department
The University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706
email: `mil...@cs.wisc.edu`

Status: Presumed Active

Environment: A collection of Z-8000 processor-based workstations (unknown vendor) on a LAN

Description: ''The DEMOS operating system began on a Cray 1 computer and has since moved through several computing environments. Its current home is a collection of Z8000 processors connected by a network. This distributed version of DEMOS is known as DEMOS/MP. DEMOS has successfully moved between substantially different architectures, while providing a consistent programming environment to the user.''

The main goals of the DEMOS/MP project are:

- Provide a clean message interface
- Provide a well structured system that can be easily modified (DEMOS/MP is the basis for a number of different research projects at Wisconsin, including distributed program measurement, reliable computing and process migration)
- To keep a high degree of network transparency while experimenting to see what mechanisms could be easily adapted to a distributed environment.

Programs are constructed of 'computational elements' (called *processes*) and 'communications paths' that join the elements (called *links*). To make DEMOS distributed, the approach was to leave the computational elements intact and modify the links to support distribution of the processing.

Processes are free to migrate without letting the initiating client know; migrated processes leave a 'link process address' that is a pointer to the new machine that the process is running on (which can be a link process address, ad infinitum).

The DEMOS/MP system is based on a special purpose lightweight protocol based on the original DEMOS model of Inter Process Communication (IPC). Due to this basis, the system supports remote demand paging (including having multiple machines sharing a single page device), and also allows diskless Z8000s to be connected to the network.

The DEMOS file system is broken up into four separate file system processes (*not specified*).

References: **[Miller87]**, [Powell77]

EDEN

Name: The Eden distributed system

Where: The University of Washington

Contact: Dr. Andrew Black
Department of Computer Science FR-35
University of Washington
Seattle, WA 98195
email: `black@cs.washington.edu`
phone: (206) 543-9281

Status: Presumed complete

Environment: Digital Equipment Corporation VAX machines, and Sun Workstations

Description: Eden represents a merging of three different approaches to operating system design, namely:

- Eden is a complete distributed operating system.
- Eden is an object-oriented system (a descendent of the Hydra system)
- Eden is also a system based on a single Remote Procedure Call (RPC) mechanism.

‘‘It is important to observe that Eden is not a set of facilities provided on top of an existing operating system in an attempt to graft distribution onto some other model of computation. This is true despite the fact that the current prototype of Eden is implemented using the facilities of Unix [Berkeley 4.2]. Eden itself provides the user with a complete environment for program development and execution.’’

‘‘Eden is an integrated system with a single uniform system-wide [eg. global] namespace spanning multiple machines.’’ Within the Eden system, each process or set of processes (called an *object*) has the following attributes:

- Objects are referenced by *capabilities*
- *Invocation* is how objects request and obtain services from other objects
- Objects are *mobile* (the processes can migrate freely)
- Objects are *active* at all times
- Objects always have a *concrete Edentype* which is in essence a description of the [finite] state machine that represents the behaviour of that particular object.
- All objects have a *data part*, including long and short term data.
- Objects can *checkpoint* autonomously (that is, they can choose to write their current state to the file system).

Eden was designed and coded in the Eden Programming Language, a language based on Concurrent Euclid[2]. This provides direct support for the low level abstractions of Eden (capabilities and invocation), as well as supporting lightweight processing within individual Eden processes.

2. An extension of the Pascal language that adds processes, modules, and monitors.

This project is assumed to be completed. It is the direct precursor of the University of Washington HCS Heterogeneous Computing System, described elsewhere in this paper.

References: **[Black85/2]**

HCS

Name: The Heterogeneous Computer Systems Project

Where: The University of Washington

Contact: Dr. David Notkin
Department of Computer Science, FR-35
University of Washington
Seattle WA, 98195
email: `not...@cs.washington.edu`
phone: (206) 545-3798

Status: Active

Environment: 15 different hardware/software combinations, including DEC VAXen (including VAXstations IIs), Sun workstations, Xerox D-Machines, Tektronix 4404/4405 computers and IBM RT-PCs. Operating Systems include VMS, Unix, and Xerox OS.

Description: HCS is designed to alleviate the following common problems in heterogeneous academic computing environments: *inconvenience* (eg. multiple, duplicate systems and peripherals or isolation from the entire campus computing facility); *expense* (eg. the cost of extra machines, servers, peripherals, etc); *diminished effectiveness* (too much time spent porting between different campus machines and on different operating systems to be productive).

Consequently, HCS is designed for many system types and different operating systems. Based on TCP/IP, it has *remote procedure calls* (RPC) and *naming* (to create a global name space for the entire heterogeneous environment) as the two key technologies.

The approach is to choose key network services and to redo them for the networked environment. The services HCS support are: remote computation; mail; and filing.

To accomplish this they have four cornerstones:

- RPC and naming give network access to the services fundamental to cooperation and sharing
- The system is designed to accomodate multiple standards
- Tradeoff: *not* transparent access to existing software (that is, unlike NFS, RFA, etc where the program will run in the distributed environment unchanged, HCS requires relinking and possibly modification to the source).
- Tradeoff: HCS is designed to support a system network rather than a language-based network.

Designed to be modular, portable, and non-OS dependent.

References: **[Notkin88]**, [Black85]

ISIS

Name:	The ISIS System
Where:	Cornell University
Contact:	Dr. Kenneth Birman Department of Computer Science Cornell University Ithaca, New York 14853 email: `ken@cs.cornell.edu` phone: (607) 255-9199
Status:	Active: new release announced June 7th, 1988
Environment:	Hewlett-Packard, Sun, Digital Equipment Corporation and GOULD computers (specific models unknown).
Description:	‘‘The ISIS system transforms abstract type specifications into fault-tolerant distributed implementations while insulating users from the mechanisms used to achieve fault-tolerance . . . the fault-tolerant implementation is achieved by *concurrently updating* replicated data. The system itself is based on a small set of communication primitives.’’ ‘‘The performance of distributed fault-tolerant services running on this initial version of ISIS is found to be nearly as good as that of non-distributed, fault-intolerant ones.’’ ‘‘No kernel changes are needed to support ISIS; you just roll it in and should be able to use it immediately. The current implementation of ISIS performs well in networks of up to about 100-200 sites.’’ ‘‘You will find ISIS useful if you are interested in developing relatively sophisticated distributed programs under Unix (eventually, other systems too). These include programs that distributed computations over multiple processes, need fault-tolerance, coordinate activities underway at several places in a network, recover automatically from software and hardware crashes, and/or dynamically reconfigure while maintaining some sort of distributed correctness constraint at all times. ISIS is also useful in building certain types of distributed real time systems.’’ The ISIS group created a fault-tolerant, shadowed, version of Sun’s NFS, called RNFS, which has worst case 25%-50% degredation of performance, but offers transparent file replication, etc.
References:	**[Birman85], [Birman88]**

LOCUS

Name:	The LOCUS Distributed Operating System
Where:	The University of California at Los Angeles
Contact:	Dr. Gerald Popek Department of Computer Science The University of California at Los Angeles Los Angeles, CA email: `po...@maui.cs.ucla.edu` phone: (213) 825-6507
Status:	Transfered to commercial venture: LOCUS Computing Corporation, Santa Monica, California.
Environment:	International Business Machine PCs, 11/70's, and Digital Equipment Corporation VAX 11/750's.
Description:	''LOCUS is a distributed operating system which supports transparent access to data through a network wide filesystem, permits automatic replication of storage, suppports transparent distributed process execution, supplies a number of high reliability functions such as nested transactions, and is upward compatible with Unix. Partioned operation of subnets and their dynamic merge is also supported.'' (further description is deemed unnecessary due to the status of the project)
References:	**[Walker83]**

Mach

Name: The Mach Project

Where: Carnegie-Mellon University, Pittsburgh, Pennsylvania

Contact: Dr. Rick Rashid
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213-3890
email: rashid@spice.cs.cmu.edu
phone: (412) 268-2617

Status: Active

Environment: Mach runs on a considerable number of different machines, including the Digital Equipment Corporation's VAX series (including the 11/780, 8600 and microVAXen), Sun series 3 workstations, the IBM RT-PC, and the Encore MultiMax.

Description: ‘‘Mach is a multiprocessor operating system kernel ... In addition to binary compatability with Berkeley 4.3 Unix, Mach also provides a number of new facilities not available in 4.3:

- Support for tightly coupled and loosely coupled general purpose multiprocessors.
- An internal adb-like kernel debugger.
- Support for transparent remote file access between autonomous systems.
- Support for large, sparse virtual address spaces, copy-on-write virtual copy operations, and memory mapped files.
- Provisions for user-provided memory objects and pagers.
- Multiple threads of control within a single address space.
- A capability-based interprocess communication facility integrated with virtual memory management to allow transfer of large amounts of data (up to the size of a process address space) via copy-on-write techniques.
- Transparent network interprocess communications with preservation of capability protection across network boundaries.’’

More than that, however, Dr. Rashid's vision is to reorganize Mach to free it from any upward dependencies on the Berkeley 4.3 Unix kernel (which it conceptually fits under) and have available a portable ‘microkernel’ that can be fit under any operating system to offer easy RPC and IPC access, as well as a shared file system, in an arbitrary, heterogeneous environment.

References: **[Rashid87]**

The Newcastle Connection

Name: The Newcastle Connection Protocol

Where: The University of Newcastle upon Tyne, England

Contact: Dr. C.R. Snow or Dr. H. Whitfield
Computing Laboratory
University of Newcastle upon Tyne
Claremont Road
Newcastle upon Tyne NE1 7RU
England

Status: Presumed active.

Environment: unknown

Description:

‘‘... [it] demonstrates that the Newcastle Connection technique can be used to connect together operating systems with differing structures and philosophies. ’’

‘‘In the field of distributed computing, an interesting recent development has been the Unix United system, implemented using the Newcastle Connection. This mechanism . . . connects together a set of Unix systems to forma coherent distributed system.’’

References: partial: **[Snow86]**

SIGMA

Name: The SIGMA Project

Where: Japan: The Japanese Information-Technology Promotion Agency under the Ministry of International Trade and Industry.

Contact: unknown.

Status: Active

Environment: unknown — part of the SIGMA project is to specify a future working environment for distributed workstations, a copy of which can be found as appendix three.

Description: The SIGMA Project is tasked with the role of consolidating Japan's software development resources. The key points noted are:

- The development of a central database for the storage, cataloging, advertising, and retrieval of software tools, and
- The structuring of a network capable of providing wide access to the database (including connections by companies, universities, and research institutes).

SIGMA is, so far, based very heavily on existing standards, with a fundamental basis of System V from AT&T because the SIGMA team found ‘‘System V more reliable and safer’’ than the Berkeley BSD distributions.

The team seems to want to avoid choosing a technology until it is very clearly the accepted standard for the industry. For example the specification does not indicate which network file system they are interested in supporting; either NFS from Sun or RFS from AT&T V.3.

For further insights, consider the SIGMA workstation feature list in the appendix: note especially the specification of a number of windows to be supported, but no indication of a specific window system having been choosen.

References: **[Schrie87]**

Sprite

Name:	The Sprite Project
Where:	The University of California at Berkeley
Contact:	Dr. John Ousterhout Computer Science Division Department of Electrical Engineering and Computer Sciences University of California at Berkeley Berkeley, CA, 94720. email: `ous...@arpa.berkeley.edu` phone: (415) 642-0865 Alternatively, the group is accessible via ARPANET at the electronic mail address: `spriters@arpa.berkeley.edu`
Status:	Active
Environment:	Sun 2 and Sun 3 series workstations
Description:	Sprite is a distributed operating system that is optimized for a small, fast local LAN, and will offer, via the file system (eg. file based Inter-Process Communications (IPC)), the resources and transparent peripheral access advantages of a mainframe while retaining the performance advantages of an individual workstation. There were three key issues for the designers; the network, physical memory, and multiprocessors. The main goal of the system is to support the Berkeley SPUR multiprocessor workstation in a distributed environment, with the target software environment being LISP.
References:	**[Ouster88]**, [Ouster87]

V

Name: The V Distributed System

Where: Stanford University, Stanford California.

Contact: Dr. David Cheriton
Computer Science, Building 460, Room 422
Stanford, CA, 94305-6110
email: cheri...@cs.stanford.edu
phone: (415) 723-1054

Status: Active

Environment: DEC microVAX II workstations, Sun 3/75's, and access to the DEC Firefly multiprocessor workstation prototype.

Description: V is designed to be a testbed for distributed systems research — built out of four logical parts: the distributed Unix kernel; the service modules; the runtime support libraries; and the added user-level commands. Due to the modular design of the system, porting particular applications to work within V is often as easy as simply relinking the binary with the new runtime libaries.

Basis of the V design is the hypothesis: ''Operating Systems developed that could manage a *cluster* of these workstations and server machines, providing the resources and information sharing facilities of a conventional single-machine system but running on this new, more powerful and more economical hardware base.'' The tenets include:

- High performance communication is the *most* critical facility for distributed systems.
- protocols, not software, define the system,
- Design distributed opearting systems as *software backplanes* — small operating system kernel implements just the basic protocols and services, with the rest in process level/user space.

V Uses high-speed Inter-Process Communications (IPC) as a base.

Tektronix is currently using V as a basis for their distributed instrumentation.

References: **[Cheriton88]**, [Cheriton87], [Lantz85]

Appendix One : References

[Anders87] Anderson, David, et. al., *The DASH Project: Design Issues for Very Large Distributed Systems* login - The Newsletter of the Usenix Association, Vol 12, No 2, March/April 1987. (**DASH**)

[Anders87/2] Anderson, David, et. al., *The DASH Project: Issues in the Design of Very Large Distributed Systems*, UCB/Computer Science Report No. 87/338, January 1987. (**DASH**)

[Birman85] Birman, Kenneth, *Replication and Fault Tolerance in the ISIS System*, Proceedings of the Tenth ACM Symposium on Operating Systems Principles, December, 1985, pp 79-86. (**ISIS**)

[Birman88] Birman, Kenneth, *Availability of ISIS Distributed Programming Environment*, netnews posting to 'comp.os.research' dated June 7th, 1988 from `ken@gvax.cs.cornell.edu` (**ISIS**)

[Black85] Black, Andrew, et. al., *An Approach to Accomodating Heterogeneity*, Technical Report 85-10-04, University of Washington, Seattle, WA. (**HSC**)

[Black85/2] Black, Andrew, *Supporting Distributed Applications: Experience with Eden*, Proceedings of the Tenth ACM Sypmosium on Operating Systems Principles, December 1985. (**Eden**)

[Cheriton87] Cheriton, David, *UIO: A Uniform I/O Interface for Distributed Systems*, ACM Transactions on Computing Systems, Vol 5, No 1, February 1987. (**V**)

[Cheriton88] Cheriton, David, *The V Distributed System*, Communications of the ACM, Vol 31, No 3, March 1988, pp 314-333. (**V**)

[Lantz85] Lantz, Keith, et. al., *An Empirical Study of Distributed Application Performance*, IEEE Transactions on Software Engineering, Vol 11, No 10, October 1985, pp 1162-1174. (**V**)

[Liskov87] Liskov, Barbara, *Distributed Programming In Argus*, Programming Methodology Group Memo 58 (to be published in the Communications of the ACM), October 1987. (**Argus**)

[Miller87] Miller, Barton, et. al., *DEMOS/MP: The Development of a Distributed Operating System*, Software Practice and Experience, Vol 17, No 4, April 1987, pp 277-290. (**DEMOS/MP**)

[Morris86] Morris, James H., et. al., *Andrew: A Distributed Personal Computing Environment*, Communications of the ACM, Vol 29, No 3, March 1986, pp 184-201. (**Andrew**)

[Mullend82] Mullender, S. J., et. al., *Protection and Resource Control in Distributed Operating Systems*, (to appear in *Computer Networks*), Vrije Universiteit, Amsterdam, August 1982. (**Amoeba**)

[Mullend86] Mullender, S. J., et. al., *The Design of a Capability-Based Distributed Operating System*, The Computer Journal, Vol 29, No 4, 1986, pp 289-299. (**Amoeba**)

[Needham82] Needham, Roger, and Herbert, Andrew, *The Cambridge Distributed Computing System*, Addison-Wesley, 1982. (**CDS**)

[Nichols87] Nichols, David A., *Using Idle Workstations in a Shared Computing Environment*, Proceedings of the Eleventh ACM Syposium on Operating Systems Principles, November 8-11, 1987, pp 5-12. (**Andrew**)

[Notkin88] Notkin, David, et. al., *Interconnecting Heterogeneous Computer Systems*, Communications of the ACM, Vol 31, No 3, March 1988, pp 258-273. (**HCS**)

[Ouster87] Ousterhout, John, el. al., *An Overview of the Sprite Project*, login - The Newsletter of the Usenix Association, Vol 12, No 1, January/February 1987, pp 13-17. (**Sprite**)

[Ouster88] Ousterhout, John, et. al., *The Sprite Network Operating System*, IEEE Computer, February 1988, pp 23-36. (**Sprite**)

[Powell77] Powell, M. L., *The DEMOS File System*, Proceedings of the Sixth ACM Symposium of Operating Systems Principles, November 1977, pp 33-42. (**DEMOS/MP**)

[Rashid87] Rashid, Rick, *From RIG to Accent to MACH: The Evolution of a Network Operating System*, CMU Computer Science Department Research Report, August 1987. (**Mach**)

[Satyan85] Satyanarayanan, M, et. al., *The ITC Distributed File System: Principles and Design*, Proceedings of the Tenth ACM Symposium on Operating Systems Principles, December 1981. (**Andrew**)

[Schrie87] Schriebman, Jeff, *Blueprints for the Future*, Unix Review, February 1987, pp 37-43. (**SIGMA**)

[Snow86] Snow, C. R., et. al., *An Experiment with the Newcastle Connection Protocol*, Software Practice and Experience, Vol 16, No 11, November 1986. (**The Newcastle Connection**)

[Walker83] Walker, Bruce, et. al., *The LOCUS Distributed Operating System*, Proceedings of the Ninth ACM Symposium on Operating System Principles, October 1983, pp 49-70. (**LOCUS**)

Appendix Two : Other Research

There are a significant number of other research projects going on in the area of distributed operating systems, however they are at specific research institutes or corporations rather than universities.

Among the more interesting projects are:

DUNIX	This is a multi-level distributed Unix kernel being done at Bell Communications Research in New Jersey. See: Litman, Ami, *The DUNIX Distributed Operating System*, ACM Operating Systems Review, Vol 22, No 1, January 1988, pg 42.
Apollo Domains	This distributed system is proprietary to Apollo Computer, and is the basis of their successful distributed workstation package.
The R* System	This research is being carried out at IBM's Thomas J. Watson Research Center in New York.
Grapevine	This is one of the many areas of distributed operating systems research done at XEROX Palo Alto Research Center (PARC), though most of the work seems to have reached a state of stasis and is no longer being pursued.
VAXClusters	This distributed operating system is built within Digital Equipment Corporations' VMS system, as a proprietary protocol for clustering machines in the VAX architecture family.
DUX	This proprietary distributed operating system is from Hewlett-Packard, Fort Collins, and is also the basis for the successful diskless implementation available on the 9000/300 series of machines.
Meglos	This system from AT&T Bell Laboratories in Holmdel, New Jersey, provides a user-level, message-based programming environment for interconnected processors. See: Gaglianello, Robert, et. al., *Communications In Meglos*, Software Practice and Experience, Vol 16, No 10, October 1986.

Appendix Three

The SIGMA Workstation of the 1990’s

l l.

Price: $18,980

CPU: 32 bit + floating point processor

Performance: 1 MIPS or greater

Memory: 4 Megabytes of RAM or greater

Disk: 80 Megabytes or more

Streamer MT: 40 Megabytes or more

Floppy: 5" 1.6 Megabyte (*format not specified*)

Serial: RS-232C (4 or more)

Display: 1024x768 color or black&white bitmapped supporting 4 or more windows

Pointing Device: 2 or more button mouse

Source: Unix Review — see [Schrie87].