The Amoeba Distributed Operating System

Andrew S. Tanenbaum
Vrije Universiteit
De Boelelaan 1081a
Amsterdam, The Netherlands
FAX: +31 20 427705
Email: ast@cs.vu.nl

1. INTRODUCTION

At the beginning of the 1990s, it is becoming clear that the computing environment will continue to undergo fundamental changes, driven primarily by technological progress. Roughly speaking, we can divide the history of modern computing into eras:

- 1970s: Timesharing (1 computer with many users)
- 1980s: Personal computing (1 computer per user)
- 1990s: Parallel computing (many computers per user)

In the 1970s and earlier, computers were huge, expensive, and located in computer centers. Most organizations had a single large machine.

In the 1980s, prices came down to the point where each user could have his own personal computer or workstation. These machines were often networked together, so that users could do remote logins on other people's computers or share files in various (often ad hoc) ways.

In the 1990s, we expect further price reductions to make it feasible to put many CPUs on a single board or even on a single chip. It will then become possible to build systems with tens or even hundreds of computers per user, possibly spread over a wide area. Such systems are usually called *parallel* or *distributed computer systems*.

This development raises the question of what kind of software will be needed for these new systems. To answer this question, a group under the direction of Prof. Andrew S. Tanenbaum at the *Vrije Universiteit* (VU) in Amsterdam (The Netherlands) has been doing research since 1980 in the area of distributed computer systems. This research, partly done in cooperation with the *Centrum voor Wiskunde en Informatica* (CWI), has resulted in the development of a new distributed operating system, called *Amoeba*, designed for an environment consisting of a large number of computers.

When Amoeba had reached the point where it was beyond the laboratory stage and was ready for commercial use, VU and CWI decided to join forces with ACE, a leading Dutch R&D company specializing in advanced software, including UNIX, CASE tools, compilers, and networking. While the research labs will continue to push the software technology forward, ACE will handle the marketing and most importantly, provide commercial support of Amoeba to customers requiring it.

For more detailed information corporate, government, and nonprofit organizations should contact:

Contracting and Licensing:
Hester v. Marwijk kooy
Email: hester@ace.nl

Technical information:
Peter Keuning
Email: peterk@ace.nl

ACE - Associated Computer Experts BV
van Eeghenstraat 100
1071 GL Amsterdam
The Netherlands
Tel: +31 20 6646416
Fax: +31 20 750389
Telex: 11702 (ace nl)

Universities should contact Andy Tanenbaum at the address given above. A special low price is available to universities, but no support is provided.

2. DESIGN GOALS

The basic design goals of Amoeba are:

- Distribution—Connecting together many machines
- Parallelism—Allowing individual jobs to use multiple CPUs easily
- Transparency—Having the collection of computers act like a single system
- Performance—Achieving all of the above in an efficient manner

Amoeba is a *distributed* system, in which multiple machines can be connected together. These machines need not all be of the same kind. The machines can be spread around a building on a LAN, and multiple LANs can be connected together over wide-area networks (WANs).

Amoeba is also a *parallel* system. This means that a single job or program can use multiple processors to gain speed. For example, a branch and bound problem such as the Traveling Salesman Problem can use tens or even hundreds of CPUs, if available, all working together to solve the problem more quickly. Large ‘‘back end’’ multiprocessors, for example, can be harnessed this way as big ‘‘compute engines’’.

Another key goal is *transparency*. The user need not know the number or the location of the CPUs, nor the place where the files are stored. Similarly, issues like file replication should be handled largely automatically, without manual intervention by the users.

Put in different terms, a user does not log into a specific machine, but into the system as a whole. Once logged in, the user does not have to give special *remote login* commands to take advantage of multiple processors or do special *remote mount* operations to access distant files. To the user, the whole system looks like a single conventional timesharing system.

None of the above would be acceptable if the resulting performance were poor, so a major effort has been made to provide good performance. In particular, the basic communication mechanism has been optimized to allow messages to be sent and replies received with a minimum of delay, and to allow large blocks of data to be shipped from

machine to machine at large bandwidth. These building blocks serve as the basis for implementing high performance subsystems and applications on Amoeba.

3. SYSTEM ARCHITECTURE

Since the hardware of the 1990s will no doubt be different from what was common in previous decades, it is worth while first describing the kind of hardware configuration for which Amoeba was designed. A typical Amoeba system will consist of four types of machines. First, each user has a workstation for running the user interface, X window System. This workstation can be a typical engineering workstation, or a specialized X terminal. It is entirely dedicated to running the user interface, and does not have to do other computing.

Second, there exists a pool of processors that are dynamically allocated to users as required. These processors can be part of a multiprocessor or multicomputer, be a collection of single-board computers (e.g., on a VME bus), or be a group of workstations allocated for this purpose. Usually, each pool processor has several megabytes of private memory, that is, pool processors need not have any shared memory (but it is not forbidden). Communication is performed by sending packets over the LAN. All the heavy computing happens in the processor pool.

Third, there are specialized servers, such as file servers and directory servers that run all the time. They may run on processor pool processors, or on dedicated hardware, as desired.

Finally, there are gateway machines that allow multiple Amoeba systems that are far apart to be connected together in such a way that to the user, the whole appears to be a single integrated system, rather than a collection of different systems.

All these components must be connected by a fast LAN. At present only Ethernet is supported, but ports to other LANs are negotiable.

4. THE AMOEBA MICROKERNEL

4.1. Microkernel + Server Architecture

Amoeba was designed with what is currently termed a *microkernel* architecture. This means that every machine in an Amoeba system runs a small, identical piece of software called the *kernel*. The kernel supports the basic process, communication, and object primitives. It also handles raw device I/O and memory management. Everything else is built on top of these fundamentals, usually by *server* processes running in user space.

Thus the system is structured as a collection of independent processes. Some of these are user processes, running application programs. Such processes are called *clients*. Others are *server* processes, such as the Bullet file server or the directory server. The basic function of the microkernel is to provide an environment in which clients and servers can run and communicate with one another.

This modular design makes it easier to understand, maintain, and modify the system. For example, since the file server is an isolated server, rather than being an integral part of the operating system, it is possible for users to implement new file servers for specialized purposes (e.g. NFS, database). In conventional systems, such as UNIX, adding additional user-defined file systems is totally infeasible.

4.2. Threads

In UNIX and many other traditional operating systems, a process consists of an address space and a single thread of control. In Amoeba, each process has its own address space, but it may contain multiple ‘‘threads of control’’ (*threads*). Each thread has its own program counter and its own stack, but shares code and global data with all the other threads in its process.

Having multiple threads inside each process is convenient for many purposes and fits into the model of distributed and parallel computing very well. For example, a file server may have multiple threads, each thread initially waiting for a request to come in. When a request comes in, it is accepted by some thread, which then begins processing it. If that thread subsequently blocks waiting for disk I/O, other threads can continue. Despite their independent control, however, all the threads can access a common block cache, using semaphores to provide interthread synchronization. This design makes programming servers and parallel applications much easier.

4.3. Remote Procedure Call

Threads often need to communicate with one another. Threads within a single address space can just communicate via the shared memory, but threads located in different processes need a different mechanism. The basic Amoeba communication mechanism is the *remote procedure call* (RPC). Communication consists of a client thread sending a message to a server thread, then blocking until the server thread sends back a return message, at which time the client is unblocked. Amoeba has a special language called *Amoeba Interface Language* (AIL) for automatically generating stub routines to marshal parameters and hide the details of the communication from the users.

Because RPC is the basis of all communication between different machine in a distributed system, it has been optimized to be very fast. Measurements show that the time for a short RPC between two user-level processes on different machines (Sun 3/60s) connected by an Ethernet is 1.1 msec for the request plus the reply. Furthermore, the bandwidth for large transfers exceeds 800 kbytes/sec.

4.4. Memory Management

Not only are user processes structured as collections of threads communicating by RPC, but the kernel itself is as well. In particular, threads in the kernel provide access to memory management services. The Amoeba memory model is simple and efficient. A process’ address space consists of one or more segments mapped onto user specified virtual addresses. When a process is executing, all its segments are in memory. There is no swapping or paging at present, although this may change in the future. The primary advantage of this scheme is simplicity and high performance. The primary disadvantage is that it is not possible to run programs larger than physical memory.

4.5. Input/Output

I/O is also handled by kernel threads. To read raw blocks from a disk, for example, a user process having the appropriate authorization, does RPCs with a disk I/O thread in the kernel. The caller is not aware that the server is actually a kernel thread, since the interface to kernel threads and user threads is identical. Generally speaking, only file servers and similar system-like processes communicate with kernel I/O threads.

5. SOFTWARE OUTSIDE THE KERNEL

The job of the Amoeba microkernel is to support threads, RPC, memory management and I/O. Everything else is built on top of these primitives.

5.1. Objects and Capabilities

Just as there are two unifying concepts in the kernel, threads and RPC, there are also two unifying concepts in the user-level software: objects and capabilities. An *object* is conceptually an abstract data type, that is, a data structure on which certain operations are defined. For example, a directory is an object to which certain operations can be applied, such as ‘‘enter file’’ and ‘‘look up name’’.

Amoeba primarily supports software objects, but hardware objects can also exist. Each object is managed by a server process to which RPCs can be sent. Each RPC specifies the object to be used, the operation to be performed, and any parameters to be passed.

When an object is created, the server doing the creation constructs a 128-bit value called a *capability* and returns it to the caller. Subsequent operations on the object require the user to send its capability to the server to both specify the object and prove the user has permission to manipulate the object. Capabilities are protected cryptographically to prevent tampering. All objects in the entire system are named and protected using this one simple, transparent scheme.

5.2. Bullet File Server

The standard Amoeba file server has been designed for high performance and is called the *bullet server*. It stores files contiguously on disk, and caches whole files contiguously in core. Except for very large files, when a user program needs a file, it will request that the bullet server send it the entire file in a single RPC. It is recommended to use a dedicated machine with at least 16 Mbyte of RAM for the bullet file server.

5.3. Directory Server

The bullet server does not provide naming services. It just reads and writes files, specified by capabilities. A directory server maps ASCII strings onto capabilities. Directories contain (ASCII string, capability) pairs; these capabilities will be for files, directories, and other objects. Since directories may contain capabilities for other directories, conventional hierarchical file systems can be easily built, but more general structures are also possible.

A directory entry does not need to contain only a single capability. It may contain a set of capabilities, to allow a file name to map onto a set of replicated files. When the user looks up a name in a directory, the entire set of capabilities is returned, to provided high availability. These replicas may be on different file servers, potentially far apart (the directory server has no idea about what kind of objects it has capabilities for, let alone where they are located). Various operations are provided for managing replicated files in a consistent way.

5.4. Compilers

Amoeba comes standard with compilers for ANSI standard C, Pascal, Modula 2, and a parallel programming language called Orca. Each of these comes with appropriate libraries.

5.5. Utilities

Amoeba provides a large number of utilities modeled after the programs that come with UNIX. Furthermore, a number of new programs are provided such as *amake*, a highly parallel configuration manager.

5.6. UNIX Emulation

To aid in porting UNIX programs to the radically different Amoeba environment, the emulation library Ajax offers near-complete POSIX P1003.1 compatibility.

5.7. TCP/IP

Although the basic communication mechanism in Amoeba is the Amoeba RPC protocol, a user-level server is provided to allow TCP/IP communication, through RPCs to the TCP/IP server. In this way, remote machines can be accessed through Internet.

5.8. X Windows

Amoeba's user interface is the industry standard X Window System (X11R4). For X servers running on workstations, a special version of X is available that uses the Amoeba RPC for high-performance communication. When hard-wired X terminals are used, these can be interfaced using the TCP/IP server.

5.9. Connection to UNIX

A special UNIX driver is provided with Amoeba that can be linked into a UNIX kernel, allowing UNIX programs to communicate with Amoeba programs. It is also possible, as stated before, to use TCP/IP for this communication, but the feature described here is much faster and less complex. Utilities are provided to transfer files between UNIX and the bullet file server.

6. NONTECHNICAL ASPECTS OF AMOEBA

6.1. Source code availability

All Amoeba distributions contain the entire source code. There are currently no binary distributions available, but computer vendors and OEMs who license Amoeba may provide binary distributions to their customers.

6.2. Amoeba is unencumbered by AT&T licensing

Amoeba is a completely new operating system. Although it provides a partial UNIX emulation, it contains no AT&T code whatsoever. This means that no additional licensing is needed for Amoeba, and that the price for Amoeba is not increased because of the use of any kind of protected software.

6.3. Documentation

Amoeba comes with nearly 1000 pages of documentation, in several volumes:

- A collection of published scientific papers describing the initial basic ideas (the ''yellow book'')
- A users' guide (how to work with Amoeba; man pages for the utility programs)

- A programmers' guide (writing clients/servers; man pages for library routines)
- A system administrators' guide (how to install and maintain an Amoeba system)

6.4. Machines on which Amoeba runs

Amoeba currently runs on the following architectures:

- Sun 3/50 and 3/60 workstations
- 68020 VME-bus boards (Tadpole TP20 and TP22; Plessey; Motorola)
- 68030 VME-bus boards (Force CPU-30)
- VAXstation 2000, VAXstation II, MicroVAX 3500

In addition, ports are currently underway to the following machines. These ports will become available sometime in 1991.

- Intel i386 (IBM AT bus)
- SPARC workstation (Sun 4)
- MIPS-based workstations (DEC 3100, SG IRIS)

Amoeba is normally distributed in ‘‘tar’’ format on either 1/4" QIC-24 streamer cartridge tape or on 1600 bpi 9-track magnetic tape. The distribution is about 20 Mbytes of source, documents and binaries. For each architecture a different subtree will be generated, and each binary tree will need another 80 Mbytes. The large size is due to the X libraries compiled into the binaries. The tape contains the Amoeba sources, documentation sources, native Amoeba binaries, and binaries of the utilities. The X sources are not included. However, the changes needed for the X11R4 sources are provided.

It should be noted that since Amoeba is a distributed system, it is necessary to have multiple machines to run it on. At the very least, there should be a dedicated file server machine, with at least 12M RAM, a processor pool with at least 1 pool processor (although more is highly desirable), and at least one workstation or X terminal for users. The pool processors can be single board computers or workstations, possibly without disks, monitors, and keyboards, since none of these are used. All machines must have an Ethernet connection.

For embedded applications, where the file server is not necessary and only the kernel is being used, it is possible to run the Amoeba kernel on a single CPU. Some installations are running Amoeba in kernel-only mode, in effect using it as a distributed high-performance kernel for real-time applications.

At the moment, Amoeba is initially compiled under UNIX; future releases will be able to boot directly from a cartridge.

7. ACE - ASSOCIATED COMPUTER EXPERTS BV

Since support for Amoeba is being provided by ACE, it is worth giving a brief introduction to this company, its experience, and its products.

7.1. Technical Background of ACE

ACE is a Dutch research and development company that provides technology, products and services in the areas of: CASE Technology, Operating Systems Technology, Compiler Technology and Networking Technology.

ACE technology and expertise include:

- Unix products: the X/OPEN XPG compliant ACE UNIX and more recently a multi-processor implementation for computer systems with a heterogeneous multi-processor architecture
- Compiler products: the integrated ACE EXPERT Compiler family for C, Pascal, Fortran-77, Modula-2 and Cobol for architectures such as Motorola MC68K, DEC VAX, Intel I386 and Norsk Data ND5000
- Standardization of Operating Systems and Programming Languages through participation in the International Standards Organization (ISO) for C, Pascal, Modula-2, Prolog and ‘‘Language Independent Data Types’’, the American IEEE Posix commission for UNIX, the X/OPEN Cobol definition and the European standardization body CEN/CENELEC for ‘‘European Workshop on Open Systems’’ (EWOS)
- Verification of Operating Systems and Programming Languages through X/OPEN XPG testing, validation of Pascal, Modula-2, Fortran-77 and Cobol with industry-standard validation suites, and validation of ANSI-C through SuperTest (a product developed by ACE in cooperation with Canadian HCR in Toronto)
- Benchmarking of UNIX systems with yearly publication of the ‘‘Benchmarking UNIX Systems’’ report, which lists the result of industry-standard benchmarks for a variety of over 100 systems
- Computer Aided Software Engineering: due to the heavy requirements for systems software, such as reliability, adherence to industry standards and efficiency, in combination with the complicated nature of this technology, CASE functionality was implemented to integrate Configuration Management, Software Engineering and Quality Assurance in one CASE environment.

From a marketing point of view ACE products and technology are mostly distributed via OEM-partners and VARs that integrate ACE technology with their own products, and distributors that resell the product for their particular market.

7.2. Project Background of ACE

In order to strengthen the technical, financial and political backing of its research and development projects, ACE established cooperation with other partners in Europe.

7.2.1. CERN

For a considerable number of years ACE has been delivering technology to the European Organization for Nuclear Research (CERN) in Geneva. ACE software is used for the development of software systems for Operational Control and Processing of Experimental Data during physics experiments.

7.2.2. ESPRIT

At this moment ACE participates in three ESPRIT projects. These projects are performed under contract with the Commission of the European Communities (CEC) and involve cooperation with several companies with different backgrounds, objectives (either research or product oriented), culture, language, etc. Setting up a Software Engineering Environment and a Project Control Environment for distributed environments like this is part of ACE expertise today.

The three ESPRIT projects, for which in all cases ACE is responsible for technical project management, are:

- ESPRIT-2 SPIRIT-1: 8.7 MECU budget, 60 man-years, 4 year duration. ACE is partner in a consortium of in total seven partners, topic is a High Performance Graphical Workstation based on a multi-processor heterogeneous computer architecture.
- ESPRIT-2 SPIRIT-2: 4.4 MECU budget, 41 man-years, 2 year duration. ACE is partner in a consortium of in total nine partners, topic is a High Performance Graphical Workstation based on a multi-processor heterogeneous computer architecture.
- ESPRIT-2 COMPARE: 14 MECU budget, 100 man-years, 5 year duration. ACE is coordinating partner of a consortium of in total seven partners, topic is Compiler Generation for Parallel Machines.

7.2.3. ACE and the European Space Industry

At this moment ACE is involved in two projects for the European Space Industry:

- For the Remote Visual Access (RVA) project ACE will supply the other contractors with expertise on the issue of Amoeba, and distributed environments.
- Eurosim is a generic simulation environment. These simulations include Hardware In the Loop Testing, HILT. The Simulation Host System will be a real time system, and the Simulation Development Environment, SDE, will be based on a graphical workstation running UNIX.

As the use of computer systems for simulation, operational control, processing of experimental data and image processing increases, ACE is expanding its role in this industry.

The objectives of the ESPRIT program of the CEC and the projects of the European Space Agency have in common that both initiatives aim for development of competitive European technology, products and services.

8. CADESE

Today’s products (both application and system software) consist of very large trees of binaries and source files. During development of ‘‘next year’s release’’, the end user still works with ‘‘this year’s release’’. In order to ensure the distribution and use of consistent releases a number of points must be considered:

- The completeness of a product with all its related objects such as directories, data files and executables should be guaranteed, even when commercially determined packaging is enforced.
- Software once distributed should really work: errors detected early by tests are preferable to bug reports filed by the end user.

- Bugs do occur, and they should be responded to. It should be simple to submit a problem report containing the relevant information and to forward it to the proper address, to check that problem reports are solved in a stated time, and to find in which version a problem report was solved.

CADESE provides a system that serves these needs, and that ensures a consistent and uniform distribution.

8.1. The Amoeba environment

The Amoeba distribution consists of a large number of files, that change over time as development is continuously taking place. Furthermore, Amoeba has been and is being ported to a number of different hardware platforms, and each such port results in some minor changes. The overall result is that version management is a complex issue.

RCS was originally used for storing the Amoeba sources, but no other functionality than version control was provided.
As the Amoeba version control system (RCS) and CADESE are incompatible and cannot be integrated, it was decided that the Amoeba source tree should be converted to CADESE. This will make the extra functionality of the CADESE version control system available and allow the use of other CADESE subsystems.

8.2. The CADESE environment

CADESE is a Configuration Control / Configuration Management system, which makes it possible to reliably control changes due to development and porting.

8.2.1. Version control

In CADESE all sources, documentation and test programs are kept together and receive a version stamp. With this version stamp it is possible to restore all files of any release as they were used for generating the system, its documentation and the test suite.
The system is automated and all versions are kept on-line in a compressed format. Simple commands make it possible to manipulate versions of a source tree. These commands provide developers with the ability to manipulate complete products and packages, as opposed to a version control system using only SCCS.

In order to create responsive and consistent support to end users and system managers, Amoeba requires professional configuration management. CADESE guarantees high quality maintenance by ACE to customers in the course of the project and after product delivery.

8.2.2. Distribution control

In CADESE the generating of the final product is driven by **descr** files. Each final target to be generated, a file or a directory, is described by a **descr** rule, a line in the **descr** file. Such a rule describes the generation process to be selected and the parameters to be passed to that generation process. The advantage of this approach is that it is easy to change the parameters for a distribution run, as they are all kept in one place. This makes CADESE especially suitable for generation of systems consisting of a large number of targets on different platforms, such as Amoeba.

8.2.3. Validation

To test a generated distribution, CADESE provides a system to store and run tests. After generating the system, it is necessary to make sure that all targets have been generated and that the generated executables really work. This is done by running the different test suites for the targets, which makes an automated verification of the distribution possible, at least to the extent that testing a kernel is possible in practice.

8.3. SPR Administration

Another package in CADESE is the SPR (Software Performance Report) environment. A user who finds a problem in a distribution can send an SPR by e-mail to the maintenance site. All SPRs are received there and dispatched by a reviewer. The reviewer reads each SPR and checks whether the problem is understood, already known, should be fixed etc., and then dispatches the SPR to a maintainer. All SPRs receive a unique identifier, which can be used for reference in e.g. the logging of source code changes. SPRs are never thrown away, for instance all solved SPRs are kept on the system to provide the maintainers of the sources with as much information as possible.

The reporting part of this package, *spr*, can be delivered to end users, to allow easy and consistent problem reporting by e-mail. SPR can also generate hardcopies, to allow submitting an SPR by e.g. fax if an e-mail connection is not available.
Using SPR with the Amoeba distribution will provide a well-guided and versatile support mechanism to Amoeba licensees.

9. LICENSING AND SUPPORT

Currently there is great interest in Amoeba within a large number of industrial areas. Amoeba is a general-purpose distributed operating system designed to be extremely fast on local area networks. Because Amoeba uses no AT&T or Berkeley (UNIX) code there is no need for the associated licenses and/or royalties.

For source licensees of Amoeba, ACE can supply a Configuration Control / Configuration Management system, called CADESE, at a discount. Non-Amoeba source licensees can, of course, also purchase CADESE, but at the normal price.

As stated earlier, one package of CADESE is the Software Performance Report environment (SPRENV). The reporting part of this package, **spr**, makes it possible to use e-mail for problem reports and/or bug-fixes. This SPR package can also be supplied separately, without the obligation of using the total CADESE environment. This possibility is extremely useful for non-source licensees.

9.1. Support available

Amoeba 4.0 will be delivered ‘‘as is’’. The reason for this is that at the moment ACE does not fully control the supplied version of Amoeba. Therefore ACE will only provide limited support for this release.
Early 1991 a new major version of Amoeba will be released (5.0), which will be fully supported by ACE. Starting with this release ACE will also offer the possibility for training and additional educational sessions concerning Amoeba.

Source licensees of Amoeba will be able to adapt the operating system to their specific requirements. ACE is available to assist in these modifications. As these changes to the OS may have implications for the maintenance contract, ACE would like to be informed of or involved in such changes. In all cases, ACE reserves the right to include these changes in future releases.

9.2. Pricing

An Amoeba pricelist for regular companies is enclosed with this document. The price will depend on the number of LANs on which Amoeba will run. Depending on the kind of license that is required a suitable amount of documentation sets will be provided. Additional sets can, of course, always be ordered against a very low fee.

The price for universities will be 500 ECU (European Currency Units) or the equivalent in other convertible currencies. Please check with your bank or us before ordering. As of Dec. 1990, 500 ECU is about US $700. This price includes a tape with binaries and sources, and one set of printed documentation.

9.3. Availability

A preliminary release is currently scheduled for February 1991. A definitive release will be available later in Spring 1991. Customers licensing the preliminary release will be able to upgrade to the definitive release for a small charge.

10. SUMMARY

Amoeba is a modern operating system that is designed for an environment consisting of multiple computers. Its major properties are summarized as follows:

- Suitable for distributed computing
- Suitable for parallel computing
- Microkernel + server architecture
- Multiple threads in an address space
- Communication based on high-performance RPC
- Object based
- Various servers available
- ANSI C, Pascal, Modula-2, Orca compilers provided
- Partial UNIX emulation provided
- Many utilities and libraries provided
- TCP/IP
- X Window System
- Full source code provided
- over 1000 pages of documentation
- No AT&T license required