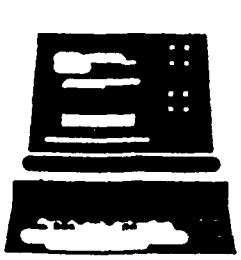


# USAISEC

*US Army Information Systems Engineering Command*
*Fort Huachuca, AZ 85613-5300*

(4)

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES
(AIRMICS)



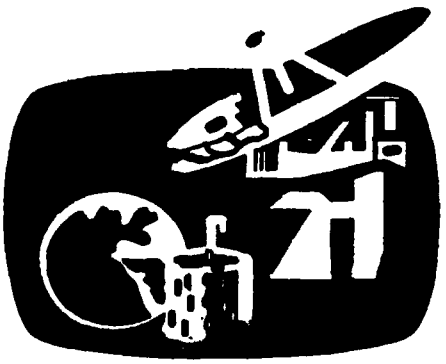
# TECHNOLOGY ASSESSMENT OF DISTRIBUTED SYSTEMS

(ASQBG-C-89-015)

February, 1989



AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800



This research was performed as an in-house project at the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the U.S. Army Information Systems Engineering Command (USAISEC). This effort was performed under the AIRMICS Technology Insertion Program to support the U.S. Army Information Systems Command (USAISC) in the development of a report entitled "Long Range Planning Guidance - Objective Configuration." An initial meeting was held in early December in Atlanta to coordinate the task. Twenty-six topics were selected for consideration, with AIRMICS agreeing to conduct technology assessments on fifteen of the topics. Planning Research Corporation (PRC) was assigned responsibility for conducting the remaining assessments and consolidating all the assessments for use in the planning document. In a two-week period, AIRMICS completed the assessments and provided the results to ISC-DCSPLANS and ISEC-SID. This research report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

**THIS REPORT HAS BEEN REVIEWED AND IS APPROVED**


s/ John W. Gowens
John W. Gowens
Division Chief
CNSD

s/ John R. Mitchell
John R. Mitchell
Director
AIRMICS

# Distributed Systems

## I. Historical Review

A computer network is not a distributed system. In a simple (nondistributed) network, a user has the capability of logging into any of the computers in the network and using the resources which reside there. However, a given resource is tied to a physical location; for example, a data file in the directory 'data', is physically stored on a particular hard disk, and every other file in the directory 'data' will be stored on the same hard disk. Thus it is possible to fill the hard disk associated with the 'data' directory while another hard disk that is tied to a different directory remains well below capacity. Thus, the logical structure of the network -- the user directories -- is directly mapped onto the physical structure of the network. The idea behind a distributed system is to completely separate the logical system topology from the physical. Continuing the previous example, in a distributed system the user would enter the 'data' directory for his files, however, the data files could be stored on any of the hard disks in the network.

There are multiple advantages to configuring the available resources as a distributed system as opposed to a simple network. Performance is improved with a distributed system since the storage space, processors, and communications devices are equally loaded. By taking advantage of all of the resources available, it is possible to significantly increase system response times and the availability of resources. In addition, the system is more flexible. For example, if each node in a distributed system contains a single modem, it is possible to assign high priority data to one of the modems. This is not possible in a simple network because the node that has the priority modem will not be able to send out nonpriority traffic. Reliability is also improved. If a resource is not available due to failure or maintenance, an equivalent resource will be used to service a user's request without requiring the user to have any knowledge of the failure. Finally,

the distributed system is easily expandable. The addition of a resource at any node will improve the functionality of the entire system.

The success of a distributed system, also known as a loosely coupled system, is tied to the following technology areas: computer networking, the Distributed Operating System (DOS), programming languages which will take advantage of the distributed environment, design tools to aid in the building of distributed systems and the software for those systems, and application packages to fully exploit the additional versatility of a distributed system.

The function of the DOS is to manage the available resources, for example, it would handle the resource allocation, interprocess communications, error control and recovery, and naming. Naming refers to the methodology that the operating system uses to find the location of a particular resource. In the past, operating systems have been designed to support at a minimum a single user, single processor machine such as an IBM AT running a Disk Operating System, and at a maximum a multiuser, multiprocessor computer. In the multiprocessor computer, or tightly coupled system, the communication between processors is usually effected through a shared memory space, and the processors are located in close proximity to one another. There is nothing to preclude incorporating a tightly coupled system as a node in distributed system; however, the multiprocessor is not itself a distributed system.

In the case of the Sequent Balance computer, a hybrid Unix operating system called DYNIX is used to support the multiple processors (Thakkar et al., 1987). A multiprocessor operating system such as DYNIX would not be useful as a DOS. If a Sequent Balance computer was integrated into a distributed system, the DOS would have to reside on top of DYNIX, or directly on the computer's hardware. In the latter case, the DOS could either replace the existing operating system or supplement it.

Similarly, most programming languages have been developed to, at most, support a multiprocessor computer. A distributed programming language must not only provide mechanisms for partitioning processes and handle reentering code, but also must coordinate interprocess synchronization and communication. Synchronization and interprocess communication are more difficult in a distributed system than in a tightly coupled computer due to the lack of a shared memory and higher communications cost relative to a tightly coupled computer. Thus, as in the case of operating systems, the design methodology used to develop tightly coupled computer languages is not useful in a distributed system.

Ideally, a distributed system could be designed to operate on a global level as well as on the local level. One metric that may be used to determine the feasibility of configuring a distributed system on a given network level is granularity. Given a set of processes which can be executed in parallel, the granularity of the system refers to the optimum process size relative to system performance. Fine granularity implies that the average process is fairly small; for example, the size of an instruction as opposed to a subroutine. The granularity of a system can be defined in terms of the internodal communications rate and the speed of the processing elements in the network. The internodal communications rate can be defined in terms of the network media, communication protocols, and the communication rates of individual devices.

On the local level, Local Area Network (LAN) technology has improved significantly in the past few years both from a technology standpoint and a standards standpoint. Three LAN standards have been defined by the IEEE 802.3, 802.4, and 802.5 standards; CSMA/CD (Ethernet), token bus, and token ring, respectively. These protocols define networks that are capable of supporting coarse-grained distributed systems.

Wide area network communications protocols are not as advanced in that the minimum data rate required for a global distributed system is not readily available to users.

In the past, dedicated lines have been set up between computer nodes which usually run at rates ranging from 1200 bits per second (bps) on a fairly slow modem to 64 Kbits per second (Kbps) on the Defense Data Network (DDN). More recently T1 services have been provided by the telephone companies which allow for internodal communications rates of 1.5 Mbits per second. These rates are insufficient considering that only very coarse granularity can be achieved in distributed systems where the nodes are fairly close together and the data rate is 10 Mbits per second (Mbps).

The complex dependencies on the hardware and software that make up a distributed system require that a developer have access to design and analysis tools. Such tools will allow for specific aspects of an operating system to be tested with a set of applications on a target hardware architecture. This will then allow for adjustments in the operating system, applications software, or hardware configuration in order to meet the design objectives. These tools will be developed by advancing the approaches used in developing modeling tools for network and telecommunications system design.

## II. Currently Available

Currently, there are approximately ten Distributed Operating Systems designed for LANs which are beyond the experimental stage; they can be transported to other sites and are fairly well documented. However, since most of these systems come from university projects, they are not supported to the degree that would be expected if they were available commercially. Most DOS use a construct known as an object in order to permit transparent network-wide access to all resources. As an example, the object in the Clouds DOS (Dasgupta et al., 1988) contains user data and code segments for user programs and databases; system data and code segments to handle the synchronization of processes, recovery from faults, and data consistency; and permanent and volatile heaps

to support memory allocation. In this manner, an object could represent a file, a subroutine, or a device driver.

The Cloud's objects are "inactive" in that a "thread" process enters an object, executes the code, and, collecting an appropriate set of parameters, continues to another object as a result of a procedure call. After the called object has been executed, the thread returns to the original object.

Another approach, using "active" objects, consists of associating each object with an object manager. In this paradigm, a message is directed to an object from a client process with the appropriate parameters, the manager then performs the desired operation using the object that was cited, and returns a message to the client process. The Cronus DOS operates in the latter manner (Schantz et al., 1986).

Both approaches have their advantages and disadvantages with respect to communications overhead, support of concurrent processes, and system maintenance and control. In both paradigms an object is defined as a part of a global address space which encapsulates all of the storage space in the network, thus allowing any object to be called from any node.

There are also some significant barriers which impede a DOS from functioning effectively in a heterogeneous environment which have to do with standardizing communications protocols for support of distributed systems. At this time, the Open Systems Interconnect (OSI) protocols do not support distributed communications, while IBM's System Network Architecture (SNA) has defined session level as well as higher level protocols for distributed processing under the Advanced Program-to-Program Communications protocol.

The lack of universal standards precludes using a straightforward approach for fielding a heterogeneous DOS. However, fielding such a DOS is possible if gateways are

developed between the DOS and each resident operating system. The gateways translate the system communications requests into a form that is recognized by the resident operating systems. This method was used successfully in a Cronus test bed.

Conventional languages can be used for developing the basis of an object in all of the available DOS. However, the complete object requires the appropriate system code and data. In other words, the language must be appropriately augmented so that the DOS can support the program. This is necessary since both the language and the operating system are involved in synchronization and communication procedures.

Some of the languages that have been augmented are Modula II, Ada, and Pascal. For example, Modula II is used as the basis for the Honeywell Object Oriented Programming System (HOPS) (Badarinath et al., 1988). HOPS uses the active object paradigm or an object management system and a Distributed Runtime Support (DRS) kernel. The kernel is simply a program that runs in conjunction with the resident Unix operating system to form the equivalent of a distributed operating system. Included as part of DRS is a preprocessor for programs written in the HOPS language which separates out the information required by the system to manage the remaining Modula II code. The Modula II code is then compiled using a conventional compiler. Since the augmented languages are, by definition, dependent on the operating system, a program developed in one distributed system is not portable to another.

Available tool support can be broken down into two areas: distributed software development and distributed system design. In the distributed software development, there are a few commercially available packages. One of the oldest of these types of systems is the Distributed Computing Design System (DCDS) developed by TRW for the Army Strategic Defense Command (Baker, 1987). DCDS provides a structured methodology for designing software for a distributed system. In DCDS the user is guided through a set of procedures which enable complete specification of the system requirements.

specification of the software requirements, definition of the processing environment and data distributions, and perform unit and system level test and verification. At this time the state-of-the-art multicomputer design tools are not out of the experimental stages.

## III. Near Term (1995)

The principle factor blocking the development of a distributed system from Non-Developmental Item (NDI) software and hardware is the lack of a commercially available, well-supported operating system. Within the next five years, a distributed UNIX operating system will be available for users which have already installed a homogeneous LAN of UNIX machines. This operating system will most likely be built to support some special-purpose applications packages that would take advantage of the distributed storage and data access capabilities available with a distributed system. In this form the network will not be entirely transparent -- the user will have to maintain some knowledge of the architecture and the location of resources. The degree to which the distributed aspects of the system can be utilized will be limited by the lack of applications software.

In the next five years, some progress will be made towards defining a set of specifications for Broadband ISDN (BISDN). This issue will be addressed by the next study session (1989 to 1992) of the International Telegraph and Telephone Consultative Committee (CCITT). The committee is expected to recommend a user-network interface of approximately 150 Mbps (Handel, 1989). However, the higher rate will relieve only one of the problems associated with fielding a wide area distributed system. Some other problems associated with system performance, naming, and network transparency on a global scale are expected to be addressed on a applied research level.

Distributed system development tools will be available to some degree. The design packages will offer the capabilities currently being integrated into experimental packages. One example of an experimental tool is the Parallel Architecture Research and Evaluation

Tool (PARET) being developed at AT&T (Nichols & Edmark, 1988). PARET uses directed flow graphs to represent the flow between processing elements, communication elements, and switching elements. This flexibility allows the user to model user programs, the system functions, interconnection routines, and the interaction between the three levels. With this particular tool, the graphics interface is well developed enabling a user to run simulations interactively in addition to collecting post simulation statistics.

The effort to allow for finer granularity within a distributed system will be impeded by the relatively long message processing times for internodal traffic. The long processing times will be primarily due to the software handling the protocols and the rate at which the traffic can be switched. The latter condition will be alleviated by the integration of electro-optic devices into networks. The former condition will improve as the operating systems improve and distributed systems communication protocols become better defined.

## IV. Long Term (2010)

In the long term, there will be no discernible separation between the operating system of the individual machines and the distributed operating system. The standardized distributed operating system will be a variation of UNIX. As a direct result of this development, the physical orientation of the system will be completely transparent to the user, and software packages will be fully transportable. Software venders will then be able to begin offering packages designed to take advantage of all the capabilities offered by distributed systems.

A number of factors will enable local area distributed systems to achieve fairly fine levels of granularity. These factors include improved compiler technology, improved software development methodology, faster processor technology, and the integration of all-optical processors into networks. The next set of advances for distributed systems will

involve the development of methods for interconnecting local distributed systems using the Broadband Integrated Services Digital Network (ISDN).

In the next ten to twenty years, tools for the modeling and analysis of distributed systems will be available which will enable the designer to optimize operating systems, applications packages, and hardware configurations relative to one another against a set of design requirements. The tools will utilize a library of databases and modules which the user will be able to access via an expert system. The user will be able to quickly model the system under consideration by answering a set of questions which will become more detailed as the design process progresses. The analysis tools will also be comprised of an expert system, which will aid the user in identifying potential bottlenecks.

## V. References


Badarinath, N. et al., (1988). HOPS User's Manual. Honeywell, Inc.

Baker, L. (1987). "Distributed computing design system." Technical Overview. TRW.

Dasgupta, P., LeBlanc, R., & Appelbe, W. (1988, June). "The Clouds distributed operating system". Proceedings of the Eighth International Conference on Distributed Computing Systems (pp. 2-9). San Jose, CA.

Handel, R. (1989, Jan). "Evolution of ISDN towards broadband ISDN". IEEE Network, 3, 7-13.

Nichols, K. & Edmark, J. (1988, May). "Modeling multicomputer systems with PARET". Computer, 21, 39-48.

Schantz, R., Thomas, R. & Bono, G. (1986, May). "The architecture of the Cronus distributed operating system". Proceedings of the Sixth International Conference on Distributed Computing Systems (pp. 250-259). Cambridge, MA.

Thakker, S., Gifford, P. & Fielland, G. (1987, May). "Balence: A shared memory multiprocessor". Proceedings of the Second International Conference on Supercomputing (pp. 56-63). Santa Clara, CA.