LBL-14771 c.2

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

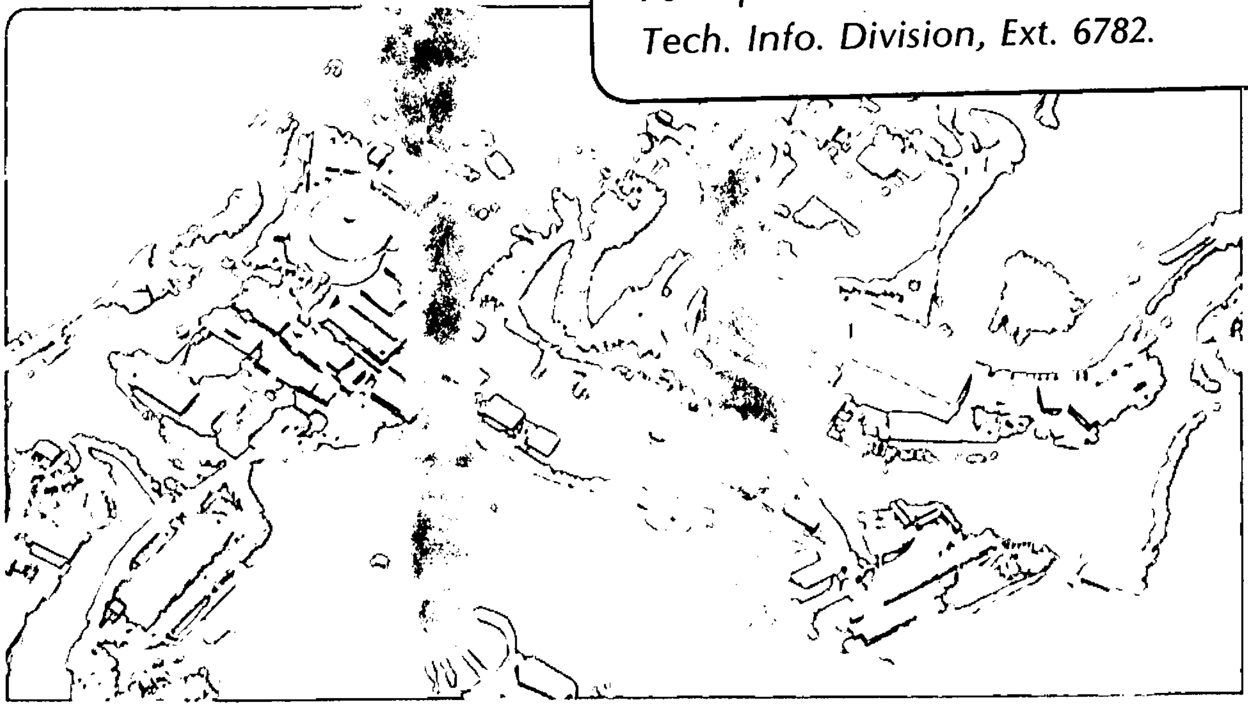
Physics, Computer Science &
Mathematics Division

Submitted for the IEEE Fall 1982 Computer Conference, Washington, DC, September 21-23, 1982; and to be published in the Proceedings

PORTABILITY ISSUES IN THE DESIGN OF DISTRIBUTED APPLICATIONS. A PORTABLE, ELECTRONIC MAIL SYSTEM

Joseph S. Sventek

July 1982



LBL-14771 c.2

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

PORTABILITY ISSUES IN THE DESIGN OF DISTRIBUTED APPLICATIONS

A PORTABLE, ELECTRONIC MAIL SYSTEM

Joseph S. Sventek

Computer Science and Mathematics Department
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

Abstract

The effects of virtual machine concepts on the design and implementation of distributed applications are explored. After outlining the critical features of such an approach, a test case of an electronic mail system is described. The results of the experiment are summarized and future directions of this work are outlined.

1. Introduction

Despite the rapid increase in the number of real-world networks, there has not been a corresponding increase in the number of distributed applications designed to exploit these networks, other than the canonical file transfer and virtual terminal protocols. This situation is especially prevalent when the hosts connected by a particular network are heterogeneous - i.e. consist of different (hardware, operating system) pairs. This situation will become more exacerbated as the number of commercially available networks increases.

A distributed application is nothing more than an implementation of a protocol layer on top of the process-to-process layer provided by the network, and can be visualized as occupying the application layer of the ISO open systems interconnection model [1]. It is imperative that the distributed pieces of the application apply identical semantics to identical situations encountered during operation of the application to realize the robustness and freedom from dead-lock required of any protocol. Guaranteeing such an identical interpretation is generally a difficult and error-prone process, especially in heterogeneous environments.

The typical user-utility (man-machine) interaction can be viewed as a two-processor, distributed application over a fairly simple network (interactive terminal line). The use of virtual machine concepts has proven extremely useful in the implementation of these "distributed" applications in heterogeneous environments [2]. The extension of these concepts to situations where a set of processes replace the human client should prove equally useful when designing more traditional distributed applications.

The remainder of the paper is concerned with the extension of virtual machine concepts to the design and implementation of distributed applications. In section 2, the use of virtual machine concepts to aid distributed application design is described. The design and implementation of a distributed, electronic mail system is outlined in section 3. The conclusions drawn from this work are summarized in section 4, together with future directions to be pursued.

2. Virtual Machine Concepts

Virtual machine concepts have proven useful in many areas of software engineering. The major points of such an approach are (a complete account may be found in reference 2):

* Determine the universe of target (hardware, operating system) pairs upon which the software is to operate.

* Determine the resources which are manipulated by the proposed software. All access to these resources should be via entry points in the virtual machine.

* Select a programming language available on all target systems.

* Implement the virtual machine on each system as a subroutine library callable from the selected programming language.

* Write the application software, invoking virtual machine "primitives" whenever access to one of the resources determined above is required. Porting each software module to all systems requires only that the code be compiled and linked with the virtual machine library on each system.

Two primary benefits accrue from the use of this approach:

* The same source code is portable to all of the systems. This guarantees that the same functionality and interface is presented to users on each of the systems.

* By isolating the access to critical resources in the virtual machine primitives, the most efficient access possible can be guaranteed for any software utilizing the resource through the virtual machine.

One particular virtual machine which has received wide popular acceptance is the Software Tools Virtual Machine and the resulting

Virtual Operating System based upon it [2]. Philosophically driven by the book Software Tools [3], the software is widely portable to a large universe of systems, and consists of 60-70 utilities providing a variety of text-processing functionality.

Features of this system of particular interest in this discussion are:

* The universe of target systems are those supporting an ANSI-66 compliant FORTRAN compiler [4].

* The programming language used is RATFOR (rational FORTRAN), to permit use of essential flow-control constructs while utilizing FORTRAN.

* Four categories of resources were included in the virtual machine:

    i. File and device input/output, as well as general manipulation of files (creation, deletion, renaming);
    ii. Directory (catalogue) manipulation providing read access to directories (write access is implicit in the creation, deletion and renaming of files);
    iii. Process control - creation, annihilation, suspension, resumption, and query of status;
    iv. Interactions with the user's environment (determine home directory, user's name, etc.).

In addition, the system based upon these features was designed to incorporate many of the popular features of the UNIX [5] system software, including I/O redirection, data flow constructs in the command interpreter syntax, and the on-line availability of all documentation.

An obvious deficiency of this virtual machine for use in multi-processing and distributed applications is the lack of interprocess communication facilities (IPC) between unrelated processes. Implementation of the process control primitives in a particular system may entail use of an IPC mechanism, but this was left as a system-specific implementation issue. The definition of a standard set of IPC primitives is complicated due to a variety of reasons:

* Many of the original target systems provide for little or no interprocess communication between unrelated processes.

* Implementation of an IPC mechanism in user code for such systems is difficult, due to the requirements for synchronization and message buffering.

* Implementation of an IPC mechanism by modifying the kernel violates the modus operandi of most organizations.

* The lack of commonality among existing IPC mechanisms prevents a simple determination of common practice upon which to base the virtual IPC calls.

Research into the design of pseudo-portable kernel modifications implementing a standard IPC mechanism is being pursued, and is described in a preliminary report [6]. The lack of a large body of common practice upon which to base this IPC mechanism has relegated such work to non-production status. The approach taken in this paper requires that a set of optimal, virtual machine IPC primitives be determined for each distributed application. It is hoped that as the number of these sample points increases, a body of common functionality can be extracted and iterated into the research described above.

3. The Software Tools Distributed Mail System

The first test of the concepts described above was the design and implementation of a distributed, electronic mail system. In addition to extending the virtual machine concepts to distributed environments, several other goals affected the design of the software:

* The software must satisfy existing and proposed DARPA mail standards due to heavy collaboration with other DARPA contractors;

* The adaptability of the software is of paramount importance; in addition to permitting the inclusion of future standards, it is critical to permit the software to use any existing inter-machine protocol schemes for message transport.

* Mail systems can provide a high level of inter-machine communication across heterogeneous networks. Such a "protocol layer" should be exploitable to provide even higher level services. It is therefore imperative that the service level be consistent on each host connected on such an internet.

* The system should provide a useful substrate upon which to base future research in the area of mail system design. Topics of possible interest include the general problem of gateway design and security issues involved in mail transport and storage.

The major premise of this paper is that the adaptability issues can be addressed by extending the virtual machine concepts of reference 2. The resulting portable modules provide the following benefits:

* The software is adaptable to changes necessitated by changing needs and/or standards. New features can be added to the source once, with all hosts on the internet benefiting from the change. The ease with which these simultaneous updates can occur permits the use of the code as a "software breadboard" upon which new features can be implemented and tested in an heterogeneous network environment before they are included in a standard. Such experimentation is highly desirable, and not available in the normal implementation scenarios.

* Since all systems run identical implementations, the service level is consistent throughout the internet. This expedites the implementation of higher-level applications using the mail system as a network protocol layer.

The virtual machine approach should also prove valuable in permitting the adaptation of the system to a variety of network protocol substrates. By encapsulating the required network functionality in a well-defined virtual machine, adapting the mail delivery to a new network should be quite straightforward. For any general distributed application to have a cost-effective lifetime, such open-ended design with respect to network inter-connection is critical.

A prototype system has been developed which satisfies the DARPA standards for message format [7], user addressing [8], and mail transport protocol [9]. Studies of existing implementations led to the design of the system as two separate layers [9,10]:

* the mail delivery modules, which are concerned with interhost message traffic, local delivery, and return of undeliverable mail.

* the user-interface modules, which permit mail composition and the perusal, filing, and general manipulation of received mail.

The linkage between the layers consists of a two-part interface: 1) a protocol by which the composition utilities queue a formatted message to the delivery system for delivery, and 2) the format of the message file created by the delivery system. In addition, the mail system was designed around the concept of a mail server (daemon) to ensure that the mail system cannot monopolize the resources of the host computer. The user-interface utilities were patterned after existing systems (SNDMSG/MSG [11]) due to their wide availability on the ARPANET and their general acceptance by that user community.

The initial implementation using the virtual machine concepts is completed and in production use. The delivery system currently employs three different network substrates (DECNET, UUCP and LBL's Hyperchannel protocol) for interhost message traffic.

In addition to providing several state-of-the-art mail system facilities (user aliasing at several different precedence levels, mail forwarding, delivery of mail to mail agents (processes), etc.), several high-level services have been provided using the mail system as a protocol layer, including:

* A simple news (or teleconference) facility. Through the use of aliases, the teleconferences may be subscribed to by users on several hosts in a network environment. Read and/or write access lists can be used for each teleconference topic; in addition, help information is provided to permit interested parties to rendezvous with the topic maintainer.

* Remote command execution, initially to provide a directory assistance feature for users of the system. Since the directory assistance problem was simply a special case of running a remote process on behalf of a user and mailing the output back to the requestor, the general capability was implemented; this permits additional features to be incorporated in the future.

A gateway between the mail system and traditional UUCP mail service has been designed and implemented, and is being studied as a model for such modules. The incompatibility between the DARPA addressing scheme and that used in UUCP is a special problem in this situation, and work continues toward the development of algorithms to solve this class of problems. In the interim, the gateway is operational, at the expense of forcing the users to provide hybrid addresses to the system.

4. Conclusions and Future Plans

The adaptability of the source code has been instrumental in the implementation and maintenance of the system. The portable source code is operational on several operating systems (RSX-11M, VMS, UNIX). The standards upon which the system is based have changed several times in the last year; the portability of the source code permits a drastic reduction in the effort to conform to the revised standards.

Of even greater importance, the extended virtual machine concepts have permitted additional network support in a straight-forward manner. Templates exist for sender and receiver processes to move the messages over a particular network, requiring only that five network-specific primitive functions be implemented. The effort required to support a new network substrate is approximately one staff-day.

The identical service level on each host provided by the portable delivery modules permitted the implementation of the higher-level services using the delivery system as a protocol layer. The general availability of such capabilities among heterogeneous hosts on other networks are not known to the author, and may represent the single most important feature provided by this design methodology.

The system is currently serving as a testbed for the development and testing of other aspects of mail systems. Of special interest is security of the mail during transport and after delivery. Several methods for securing the messages are being studied, and implementation will occur as required. As local machines are connected to additional networks, more gateway modules will require implementation. The availability of these extra points in the design space should permit a consolidation of common features, and the development of a design standard for future implementations should result.

The use of the virtual machine concepts in other, more time-critical distributed applications are underway. It is hoped that some common IPC needs can be determined, such that an essentially complete set of IPC primitives can be defined; the best method of providing these capabilities on the systems of interest can then be pursued.

5. References

[1] Zimmerman, H., "OSI Reference Model - the ISO Model of Architecture for Open Systems Interconnection," IEEE Transactions on Communications, Vol. COM-28, April 1980.

[2] Hall, D.E., D.K. Scherrer and J.S. Sventek, "A Virtual Operating System," Communications of the ACM, Vol. 23, No. 9, September 1980.

[3] Kernighan, B. and P. Plauger, "Software Tools," Addison-Wesley Publishing Company, ISBN 0-201-03669-X, 1976.

[4] "American National Standard FORTRAN," ANS X3.9-1966, American National Standards Institute, New York, 1966.

[5] Ritchie, D.M. and K. Thompson, "The UNIX Time-Sharing System," The Bell System Technical Journal, Vol. 57, No. 6, Part 2, July-August 1978, pp. 1905-1930.

[6] Sventek, J.S., "The Virtual Aether: An Inter-process Communication Mechanism for Distributed Processing," Lawrence Berkeley Laboratory internal report LBL-12658, Berkeley, CA, 94720, May 1981.

[7] Crocker, D., J. Vittal, K. Pogran and D. Henderson, "Standard

for the Format of ARPA Network Text Messages," Network Information Center report NIC 41952, SRI International, Menlo Park, CA, November 1977.

[8] Postel, J., "Computer Mail Meeting Notes," ARPANET Request for Comments 805, USC Information Sciences Institute, Marina del Rey, CA, February 1982.

[9] Postel, J., "Simple Mail Transfer Protocol," ARPANET Request for Comments 788, USC Information Sciences Institute, Marina del Rey, CA, November 1981.

[10] National Bureau of Standards, "Features of a Message Transfer Protocol," Report No. ICST/CBOS-81-5, U.S. Department of Commerce, November 1981.

[11] Vittal, J., "MSG Manual," USC Information Sciences Institute, Marina del Rey, CA, July 1975.

This work was supported by the U.S. Department of Energy under contract number DE-AC03-76SF00098.