Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

S.J. Mullender

Distributed systems management in wide area networks

Department of Computer Science Report CS-R8419 November

Distributed Systems Management in Wide Area Networks

Sape J. Mullender

Centre for Mathematics and Computer Science
Amsterdam

While quite a few distributed operating systems for local-area networks exist, hardly any work has been done to date on distributed operating systems for wide-area networks.

In Europe, a number of public networks are now operational, with gateways between some of them. However, the use of these networks is still mostly restricted to *"remote login"* and, in some cases, simple *file transfer* operations.

To study these problems and to find structural solutions for efficient and simple use of national and international networks the working group *"Distributed Systems Management"* was founded within COST 11. Recently, this working group has submitted a research proposal to COST 11 to realise an infrastructure for the implementation of distributed services in a wide-area network in a European collaborative effort. The model, underlying the reserach is the *service model*, used in many local-area network distributed operating systems.

The research project is described, and the proposed infrastructure is discussed in some detail.

1980 Mathematics Subject Classification: 68A05, 68B20.
CR Categories: C.2.2, C.2.4, D.4.4, D.4.7. 69C22, 69C24, 69D54, 69D57
Keywords & Phrases: service model, distributed systems, wide-area networks, COST-11
Note: This report has appeared in the Proceedings of the *NGI-SION Informatica Symposium* 1984, pp 415 — 424.

1. Introduction

Many distributed operating systems exist, based on local area networks, but, in spite of a growing need, the possibilities to use the potential of national and international networks efficiently are virtually non-existent.

Some networks are now operational in Europe, with gateways connecting them here and there. In principle information could be exchanged on these networks. Currently, however, these networks are almost solely used for *remote login, electronic mail, teleconferencing* and *file transfer*. Most of these applications have been developed on an *ad hoc* basis, each application with its own protection mechanisms, network protocols, etc. Using an international network for any other applications often requires nested log-in on a number of hosts on the path through various networks to the destination host.

The COST 11 [Martin-Löf83, Kalin83] *"Distributed Systems Management"* group has been started to study these problems and find an infrastructure for simplifying management of distributed processing activities. In january of this year, the COST 11 DSM group finished a research proposal for a four year programme of collaborative research on some of the issues of distributed system's management in Wide Area Networks. The work will be carried out by research institutes all over Europe. In The

Report CS-R8419
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Netherlands, participants are the *Centrum voor Wiskunde en Informatica,* the Computer Science Department of the *Vrije Universiteit* and the Network Group of the *Technische Hogeschool Twente.* COST 11 is asked to finance the collaboration costs, such as travel and subsistence cost, network connections and communication costs. This paper discusses the proposed research.

2. REQUIREMENTS, PROBLEMS AND ISSUES

The principles underlying the management of information processing systems apply whether the systems are local or distributed. In the present context a three part definition of management is used:

i. management is planning and organising the provisions of resources and identifies (a) where resources may be located, (b) their availability and (c) their cost;

ii. management is the control of the use of, and access to, resources according to allocation, optimisation and authorisation rules;

iii. management is the task of ensuring that resources remain accessible and that they function correctly; and, when this objective cannot be achieved, of ensuring that suitable signals are available which identify the failure.

This definition is wide ranging, covering management both within and external to individual network hosts. To narrow down the area addressed by distributed systems management it is important to differentiate between local management activities of the various host operating systems within the network and those activities concerned with the distributed activities of the system. The open systems' environment offers a set of services provided by the host operating system. The way in which those services may be implemented is outside the scope of open systems interconnection. Standards for Distributed Systems Management are concerned with a non-local use of these services. However, the interaction between local and distributed aspects of management are a significant R & D matter.

Management is realised through the actions of managers. It is important that the managers of host systems (i.e., people) have the freedom to effect the management policies appropriate to their systems. Distributed systems management must provide the framework for general mechanisms in which a variety of management policies can operate.

Given the functions of management and the understanding of the constraints imposed by wide area networks, the tasks to be addressed are:

1. to identify models for distributed systems as a context within which management activities can be considered;
2. to identify (and, if possible, develop) the set of management tools which assist in the planning and organisation of distributed processing;
3. to identify the mechanisms through which managers can apply their policies to control and use resources, through appropriate optimisation and authorisation strategies;
4. to identify protocols for the control of resources, protocols which maintain resource availability and protocols which signal system failure.

For network users and managers to have their requirements satisfied a number of mechanisms and services need to be provided. Perhaps the greatest barrier to offering such mechanisms and services in a Wide Area Network is the lack of an agreed model for the organisation of distributed computing and for a set of communication standards for the exchange of control and supervisory information.

Even with such a model and a set of management communication protocols there are still detailed problems to be resolved concerning the details of the management mechanisms and services which are needed. The more important ones concern mechanisms for service location and authorisation, and the lack of quantitative information which can offer guidance to management in controlling the resources for which it is responsible. Of lesser, but still significant, importance are the services which assist users to obtain the most effective use of the facilities offered by the network of distributed computers.

At present, there do not appear to be either the data to help resolve these issues nor any general models of distributed systems through which these issues can be investigated. Managers need realistic data from efficiently managed distributed systems to feed into their models in order to help them with their planning. Yet, at the same time, managers are unable to establish whether their systems are operating efficiently for lack of adequate diagnostics and tools to help them analyse distributed system performance.

Managers are unlikely to accept other than the most stringent safeguards in the application of authorisation rules. The global access which is (theoretically) possible in a Wide Area Network and the fact that management units have autonomy means that users and the system they use have to carry out an authentication excercise at the start of any instance of a communication session. Subsequent dialogue must be policed by the computers within the Wide Area Network (even by the Wide Area Network itself) to maintain the integrity of the session. The most effective authentication mechanism and the way that mechanism is made apparent to both users and computer or network manager still requires detailed study.

3. The Service Model for Distributed Processing

Today most people use computers interactively; that is, they type commands at their terminals, the system will process their commands and return a reply. If the user is satisfied with the answer, he may type a new command; if not, the user may retry the command, or phrase it differently. Inside a computer's operating system the same thing goes on, albeit at another level: the user's command interpreter makes system calls, requesting programs to be run, files to be read, tapes to be rewound, etc., and the system replies with data, or simply with an acknowledgement. At a lower level still, programs make extensive use of subroutine calls; the call can be seen as a request to execute the body of the subroutine, and the subroutine return as its reply. Obviously, thinking in terms of requests and replies, possibly nested recursively, is an excellent way of structuring problems into small portions.

Conceptually, distributed systems are among the most difficult to oversee, so a structured approach to building distributed systems is essential. The natural choice of a model is that of using requests and replies. In this section we shall examine this model in some detail and discuss its consequences on distributed systems design.

The maker of a request shall be named the **client**, the processor of the request shall be the **service**. A client can be a person at a terminal, an operating system, a process, a processor, etc. A service is an abstraction of the requests that can be made and the replies that can be expected, comparable to *abstract data types* [Liskov74]. A service is always embodied by one or more **servers**, the processes, processors, or devices that carry out the requests as defined by the service.

A **request** consists of information travelling from client to server, a **reply** is information sent by the server back to the client. In a distributed system client and server do not generally reside on the same physical machine; requests and replies must therefore be sent through a computer network from one host to another. Depending on the type and speed of the network, requests and replies can be sent as packets, messages, or over connections.

So far, the service model closely resembles a remote procedure call mechanism, the request representing the call, and the reply the return. It is more than that, however:

unlike a subroutine, a service can fail. The processor where the service is implemented may stop working, a bug in the service program may cause the server to crash obscurely once every thirteen weeks, or the network may break down. Making a request is like calling a subroutine that *almost always* returns. In a program, a non-returning subroutine causes the program to fail; in a distributed system, a non-replying service need not crash the client. The client can retry a number of times, expecting the service to be repaired, or to contact another instance of the service, or it can resort to another service to achieve its goals in a different way.

The property of distributed systems of potentially surviving server crashes is what makes distributed systems so interesting from the point of reliability and error recovery. But it is necessary to design the software to expect errors, and to react to them appropriately. A client must *expect* a server to crash every now and then. When a reply does arrive the client may always assume the server has done its work correctly, but if no reply comes, the client does not know if its request has been carried out; the client must try to find out, and, if necessary, repeat the request.

In the same philosophy, services must be designed in such a way that recovery from crashes can be simple and straightforward. Most request can be so defined that carrying them out once, or more than once does not yield different results. If a server crashes, such requests can safely be repeated.

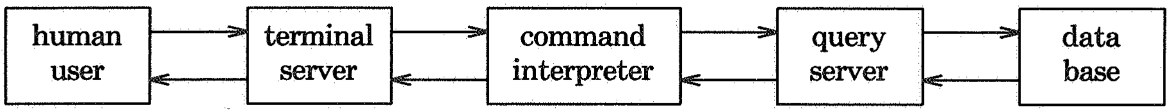

FIGURE 1. An example of a service hierarchy

Naturally, a server can itself be client to another service. In fact, the possible hierarchy of services is the strength of the model. As an example, a possible hierarchy of services is shown in Fig. FIGURE 1, where a human being is shown as a client of a terminal server, which in turn, is a client of the command interpreter, etc. A crash, for instance, of the database server, will be detected by the query server, which must then try to recover from it. The query server can retry the request, it might rephrase a query to get the answer from another database server, and as a last resort, it can report failure to its client, the command interpreter. In this way the human client at the top of the hierarchy gets to cope only with irrecoverable errors and crashes in the system.

4. SERVICES in TRADITIONAL OPERATING SYSTEMS

Traditional operating systems provide service to its clients in a much more restricted way than conceived in the service model of the previous section. Usually, the only services available to programs are those provided by the operating system in the form of system calls. This restricts the service hierarchy to two levels: user to program, and program to operating system. Some operating systems, such as UNIX,† have a well designed user-program interface: to the client-user a number of alternative services is available, and programs can sometimes be combined to provide powerful "programs of programs," but even among the best of operating systems, the possibilities are limited, and, at the system call level, no choice of service is left to the programmer at all. Most operating systems, for instance, have a built-in file system, and, whether one likes it or not, it is the only available file system.

† UNIX is a Trademark of Bell Laboratories.

Traditional operating systems run on one centralised processor; if it, the operating system, or one of its components (file system, terminal controller, etc.) crashes, the whole system crashes. Naturally, in these systems it is not necessary to pay much attention to recovery from crashes: if the system crashes, nothing can be done anyway. Sometimes, operating systems run on more than one processor, but even then, the processors are so closely coupled that a crash in one brings down the others also. If we want to use existing systems as a basis for reliable crash-proof distributed systems, we must add mechanisms for error recovery, and increase the possibilities of allowing clients the choice of many services.

Many computer centres now have a connection to one of the wide area networks, so communication is possible between one computer and another. The services available over the network are very limited, however. Often there is a network-wide electronic mail service, and sometimes there are file transfer capabilities. Occasionally we find another special-purpose application that can be accessed through the network, such as teleconferencing systems, but only in very few cases does the operating system allow processes on one host under one operating system to communicate with another process in another host under another operating system.

5. Integrating the Service Model with Existing Software

In the short term, computer networks will be mainly used for mail exchange, file exchange, and remote file and data base access. In a primitive form this is already provided on many existing systems. Accessing these services requires intimate knowledge, however, of both the system where requests originate and the system where the service is implemented.

In the long term people will have to use the network more intensively and for many more types of access. If computer networks still work in the same fashion, the expert knowledge required to use these services will increase dramatically. It is therefore essential that a uniform way of accessing remote services is inserted between the operating system and the (remote) client. This model is shown in Fig. Figure 2.

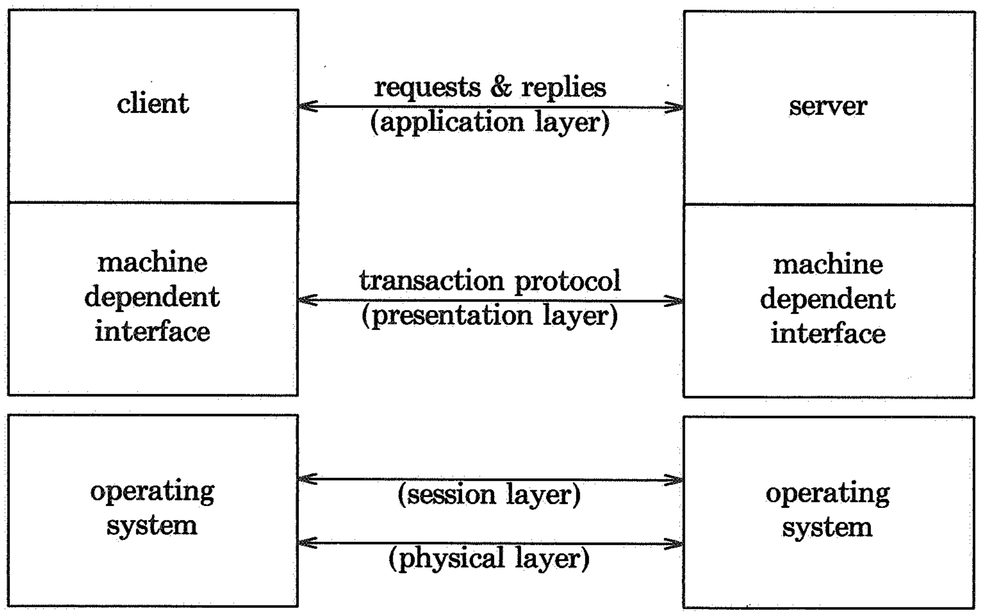

Figure 2. The client server implementation model for existing computer systems. The double line represents the separation between application software and the operating system.

An essential property of this model is that it allows existing software to be integrated into a distributed environment. If new software is henceforth written directly in the language of clients and servers, requests and replies, reliability and error recovery, a gradual changeover to practical distributed systems on a large scale is possible. Several tries have been made in the past to build a coherent distributed system on top of existing operating systems [Thomas73, Millstein77, Mamrak82]. We also mention [Hall80] which is an attempt to build a uniform user interface on a collection of operating systems, an approach very similar to the uniform client-server model.

6. The Service Model as the Operating System of the Future

As processes shall rely on fewer services of the local operating system, but rather on services replacing traditional file i/o, terminal management, etc., the underlying operating systems will become increasingly simple. This is fortunate, because today operating systems are among the most complicated programs written. Few, if any, operating systems are free of bugs, and we believe the main cause lies in their complexity.

We must set as our goal to reduce the complexity of the operating systems, by removing every function from it that can also be realised outside the operating system. Eventually, the only task the operating system has, is to provide programs with an environment for execution, and interprocess communication primitives. Remaining functions will then be memory management, exception handling, and the implementation of system calls for interprocess communication and process control (allocate memory, ignore or catch certain exceptions, timers, exit, etc.). It is likely even that processors will become so cheap that it is no longer worth while to implement multiprogramming, but assign one process per host. From the viewpoint of protection and scheduling this can be a great simplification of the operating system.

Process creation can be done through the services of the **process server**, a service that finds a suitable processor for the process to be run, downloads the code into it and starts the processor. Accounting can be done by an accounting service, the **Bank Server**, to be described later. Different file systems can co-exist to give the users maximum choice of service, interface, reliability and speed. The nature of the storage system for a database is completely different from the one needed by, for instance, a compiler that makes a temporary file, and different again from the storage needed by a text editor. This is indeed why today database systems often run on separate machines; the file system provided by the "regular" operating system is unsuitable for database applications [Stonebraker81, Tanenbaum82].

Existing software can be ported to the new generation distributed operating system by building an emulation layer that translates archaic system calls into the new service requests.

7. Management of Services in Wide Area Networks

An important difference between distributed systems in local area networks and those in wide area networks is that local systems are usually under control of one administration, while wide area networks are usually under control of many different administration. In wide area networks the lower layer communication protocols are usually provided by the PTTs so the choice of using datagram service or virtual circuits is not available.

Each administration in a wide area network potentially has its own procedures for accounting, resource control, and access permissions. For wide area distributed systems, it is important that one accounting and resource control mechanism is available that can be used to realise all the different policies.

Basically, there are two methods of access control. One is to use *access control lists* (ACLs), the other is to use *capabilities*. Both methods are well known: In the ACL method, a server grants a client access to an object after checking if the client is on the object's list of authorised users. In the capability method, a server grants a client access to an object if the client can present a capability for the object. The first model is characterised by a list of authorised clients, stored with each object; the second by a list of capabilities for objects to which access is allowed, stored by each client.

The ACL method requires a way for a server to establish the identity of its clients. It may not be possible that a client impersonates another to obtain access to an object (or service) that would have been refused otherwise. The capability method requires a method of distributing capabilities to clients in such a way that clients cannot forge them, construct them, or obtain access to an object by trail and error. For both problems adequate solutions exist [Mullender82, Donnelley80, Evans74, Needham78]. Both methods should be supported by the interprocess communication mechanisms to allow different administrations to use different access control policies.

8. Management Supporting Services

A number of services are conceived to support usage and management of Wide Area Network services. Among these are *Name Service* which map local, private names for objects onto globally unique object names, *error reporting* to help managers detect malfunctioning network components, *performance monitoring* for managers to detect bottlenecks in the system, *Bank Service* for accounting and resource control, *help service* for the assistance of users who get stuck, and *command interpreters* to help users to communicate with Wide Area Network services in a natural and meaningful way. We shall discuss some of these services below.

Name Servers or **Directory Servers** provide a mapping between clients' private name spaces and globally unique object or service names. A further mapping maps global object and service names onto the network address of the object or service. This two-level mapping allows a clean separation of functionality: when a client renames an object, only the first mapping is affected; if an object migrates to another host, only the second mapping is affected.

Directory Service thus consists of two more-or-less independent services: a service in the user domain, for conveniently naming private objects, and a service in the operating system domain, for locating objects, given their globally unique name. This separation allows the existence of several independent directory services in the user domain, offering different capabilities. Directory services could offer *"yellow pages service"* which responds to queries of the nature: "Tell me the names and give me a description of file servers that implement atomic update and concurrency control mechanisms."

The global-name to network-address mapping is the subject of considerable research. This map has to be carried out efficiently, and it has to be carried out securely. It is obviously unacceptable if requests, containing sensitive information for a particular trusted service, end up on the wrong host. Service location and object location is closely related to issues of authentication, protection, and encryption, and the DSM group intends to investigate the problem in this context.

An example of a versatile accounting mechanism that can be used for resource control, access control, and, of course, accounting is the **Bank Server**, described below.

The **Bank Service** consists of one or more Bank Server processes that maintain *accounts* for each user in the system. An account may contain *"virtual money"* in one or more *"virtual currencies."* One of the currencies could correspond to *"real money"*. Other currencies can represent disk quota, cpu seconds, phototypesetter pages, etc. A service can ask the Bank Service to make a new currency for it, specify the amount of money to be *coined*, and hand out the money to its (potential) clients, possibly in return for virtual or real money in another virtual or real currency.

To make the Bank Server secure, it uses a capability mechanism; the user that creates an account receives a capability for it. Only by presenting the capability a client can take money out of an account. Keeping the capability of an account secret is the key to preventing other users from stealing one's money.

A typical interaction between a client and a server goes something like this: first the client, presenting a capability for his account, requests the bank service to prepare a cheque for some amount; the Bank Service debits the client account, makes a unique unforgeable bit pattern representing the amount, and returns that to the client as a cheque for the amount. The client then sends his request to the server along with the cheque, and the server clears the cheque with the Bank Service before carrying out the request; the Bank Service credits the server account with the amount, and erases the bit pattern in the cheque from its list of outstanding cheques, preventing a cheque from being cashed twice.

Doing business like this requires two extra transaction with the Bank Service for every transaction that has to be carried out, but, fortunately, it can easily be optimised. The client can send an amount to the server that covers many transactions in one blow, the server can cash the cheque once, and maintain a local credit account for each client for which it works. The amount sent at one time by a client to a server must not exceed the amount of trust the client has in the server.

A mechanism as just described can be made very secure. A property that could make the accounting system desirable in an international environment is that untraceable payments can easily be implemented [Chaum82]. The Bank Service is not in a position to analyse a user's spending patterns in this way.

Network users need meaningful messages when interacting with the services provided by the system. The purpose is to make the users' interaction with the network and its facilities as effective as possible. Users are unlikely to use the range of facilities which a network can offer unless a user-friendly environment is available. Observation that a network's facilities are easy to use and that *"Help"* is readily available will act as a catalyst to promote others to use them.

A **Help Service** can operate in four ways:

i. giving users assistance when there are faults, e.g., how long it will take before a service is resumed, or whether the fault led to any loss of information;

ii. giving guidance on how to access services, e.g., by providing on-line documentation, structured walk-throughs for novices, and, in the last resort, human contact points for further information;

iii. offering a focal point for user feedback, e.g., customer complaints and requests;

iv. providing users with status information, e.g., on service or network availability, maintenance schedules, and advising on (advertising) new services.

9. Programme for Research and Development

Having considered the evidence for distributed systems management and presented lists of options for tools and services, it is now appropriate to draw some conclusions.

One of the motivating forces for the Distributed Systems Management study carried out by the COST 11 bis DSM-Group was the realisation that standardisation bodies were having to grapple with issues which are still lively research topics. The analysis presented in the report show that they remain research topics. Although some studies have been carried out and some systems have been built to demonstrate principles, they have not been withing the scope of open systems interconnection nor of the open use of wide area networks. Therefore, if standardisation work is to receive any support from this type of activity, it has to come from relevant practical exploration of the issues and of the proposed methods for carrying out distributed systems management. A further reason why standardisation bodies are having difficulties in this area is the interdisciplinary nature of the problem. Although distributed information processing has been facilitated by the development of adequate data communications, many of the key issues relate to standards for the user interface (OSCRL) and to matters which cannot be the subject of standards such as the services and tools that are provided to support management.

The research programme consists of two related activities.

1. an investigation of tools which can enable managers to perform their functions more effectively;

2. an investigation of the services which are needed to provide a distributed system management infrastructure.

This work will need to research the types of model which should be developed and the general applicability of those models. At present, it appears that, whereas a single, more general model would be a desirable commodity in the long run, in the short term there is not yet the information available upon which to build more general models. A more pragmatic approach to model building to analyse particular and well identified scenarios is therefore recommended.

In meeting the requirements of the second activity a list of topics for study, development and implementation can be drawn up. The following list proposes a priority order:

1) The distributed system management kernel.
2) Name Servers
3) Authentication mechanisms
4) Journalling and performance monitoring
5) Help Services
6) Error reporting and diagnostics
7) Bank Service (accounting)
8) Command interpreters

It is noted that the technical solution to providing many of these management services is by the simple expedient of passing messages – in well specified format – for storage within or retrieval from an information base. In the short term the message facilities provided by Computer Based Message Services (e.g., GILT [Wallerath83] perhaps) could suffice. The accounting requirements are not dissimilar from those put forward in the proposal to COST 11 for the OSIS project. Also the provision of "Help," the concern for adequate user interfaces and the legal implications of distributed information processing are of significance to those concerned with Human Factors [Eason83]. Thus, the study of Distributed Systems Management has synergystic relevance to other COST 11 activities.

References

[Chaum82]
Chaum, D., "Blind Signatures for Untraceable Payments," *Proc. Crypto*, 1982, Plenum Publishing Co. N.Y..

[Donnelley80]
Donnelley, J. E. and Fletcher, J. G., "Resource Access Control in a Network Operating System," *ACM Pacific '80 Conf.*, Nov. 1980.

[Eason83]
Eason, K.D. and Jensen, W., "Human Factors in Teleinformatics," *Proc. of the European Teleinformatics Conf.*, pp.3-5, October 1983.

[Evans74]
Evans, A., Kantrowitz, W., and Weiss, E., "A User Authentication Scheme Not Requiring Secrecy in the Computer," *Comm. ACM*, vol. 17, no. 8, pp.437-442, August 1974.

[Hall80]
Hall, D. E., Scherrer, D. K., and Sventek, J. S., "A Virtual Operating System," *Comm. ACM*, vol. 23, no. 9, pp.495-502, Sept. 1980.

[Kalin83]
Kalin, T., "Technical and Organisational Overview of COST 11 Bis Projects and Working Groups," *Proc. of the European Teleinformatics Conf.*, pp.3-5, October 1983.

[Liskov74]
Liskov, B. and Zilles, S., "Programming With Abstract Data Types," *SIGPLAN Notices*, vol. 9, pp.50-59, April 1974.

[Mamrak82]
Mamrak, S. A., Maurath, P., Gomez, J., Janardan, S., and Nicholas, C., "Guest Layering Distributed Processing Support on Local Operating Systems," *Proc. 3rd Int. Conf. on Distr. Comp. Syst.*, october 1982.

[Martin-Löf83]
Martin-Löf, J., "The COST Framework and its Activities in Teleinformatics," *Proc. of the European Teleinformatics Conf.*, pp.3-5, October 1983.

[Millstein77]
Millstein, R. E., "The National Software Works: A Distributed Processing System," *Proc. ACM Annual Conf.*, pp.44-52, 1977.

[Mullender82]
Mullender, S.J. and Tanenbaum, A.S., "Protection and Resource Control in Distributed Operating Systems", IR-79 (to appear in Computer Networks), Vrije Universiteit, Amsterdam, August 1982.

[Needham78]
Needham, R. M. and Schroeder, M. D., "Using Encryption for Authentication in Large Networks of Computers," *Comm. ACM*, vol. 21, no. 12, pp.993-999, December 1978.

[Stonebraker81]
Stonebraker, M., "Operating System Support for Database Management," *Comm. ACM*, vol. 24, no. 7, pp.412-418, July 1981.

[Tanenbaum82]
Tanenbaum, A. S. and Mullender, S. J., "Operating System Requirements for Distributed Data Base Systems," pp. 105-114 in Distributed Data Bases, ed. H. J. Schneider, North-Holland Publishing Co. (1982).

[Thomas73]
Thomas, R. H., "A Resource Sharing Executive for the ARPANET," *Proc. NCC*, 1973.

[Wallerath83]
Wallerath, P., "The GILT Abstract Model of a Computer Based Message System," *Proc. of the European Teleinformatics Conf.*, pp.3-5, October 1983.