Towards a Worldwide Distributed File System
The OSF DCE File System as an example

Norbert Leser
DCE Evaluation Team
Open Software Foundation™

September 27, 1990

Accessing data resources at remote systems is a fundamental service in a distributed environment. FTAM and similar technologies are designed to allow transfer of unstructured and structured data from and to remote locations. In contrast, NFS™ - a de facto standard - provides for sharing files in a heterogeneous system. Restrictions in this NFS protocol due to its stateless server architecture will be addressed.

This paper will elaborate on how an improved file server architecture can address requirements and characteristics inherent in a distributed computing environment. Issues are: standardized user interface and local access and sharing semantics (IEEE POSIX 1003.1), ease of administration, authentication and authorization, availability, uniform naming, worldwide scaling, and interoperability with existing NFS sites.

Further, it discusses how OSF's DCE File System accommodates these features, and how a sophisticated naming model is integrated. The file server and the caching design offloads the server and keeps the network traffic low for performance and scalability. A token technique enables both correctness (single site semantics) and speed. A file server file system allows for replication, movement and backup of logical file system units (filesets). A logging mechanism provides for quick recovery after server crashes. The vfs switch and other significant interfaces for implementations in heterogeneous environments will be explained.

BIOGRAPHY

Norbert Leser is a member of OSF's Distributed Computing Environment evaluation team. He joined OSF in June 1988. His engineering experience in the high-technology industry spans 18 years. Most recently he worked for Siemens, responsible for Internet networking and distributed computing systems. In positions with other firms, he led software-development projects for several types of commercial applications, including data processing and control, agricultural management and planning, and computer-aided design. His over ten years of experience in the UNIX® environment has focused on communications software and hardware.

Towards a Worldwide Distributed File System
The OSF DCE File System as an example

1. General architectural Issues

Upon entering the 90s, distributed computing becomes increasingly important for information processing. Distributed systems certainly include a wide range of technologies such as security and naming. Providing a model for sharing data resources in this environment is important.

Computer systems and communications hardware have been rapidly changing and evolving over the last decade. The cost of CPU power has declined at a very fast rate which has lead to the success of highly sophisticated graphic workstations and the evolvement from traditional mainframes into powerful multiprocessor systems with the ability to process and manage extremely large data storages. The last decade has been characterized by rapidly growing sites connected in Local Area Networks (i.e. Ethernet) and by establishing cost effective but fairly slow distance connections. Changing policies by communication providers, technologies such as X.25 and ISDN and the availability of high speed long distance connections will provide for reduced costs of communication bandwidth.

In designing services in distributed systems, these rapidly changing underlying technologies have to be taken into account to achieve optimal usage of resources. Data sharing services such as NFS, FTAM, or FTP have been successful because of their alignment with previously available system technologies. The conceptual models of file transfer protocols such as FTAM for OSI based interconnections and FTP for the Internet network meet the requirements for peer-to-peer long haul connections, but these technologies explicitly do not address interactive and synchronous data sharing. NFS, on the other hand, accommodates the requirement of expanding the local file system view (UNIX) into a heterogeneous local area network environment to a certain degree[1]. In this paper, I will discuss, how the DCE File System (DFS)[2][3] meets the requirements in today's and future environments.

A distributed file system approaches a general purpose service for accessing and controlling data resources transparently, independent of the location where they are stored. Thus, specialized services such as distributed databases, transaction processing, and fault tolerance systems need to be addressed separately to meet their specific requirements. For a general purpose distributed file system it is necessary to achieve a consensus on the data model and the file system interface to provide the needed generality. This is still one area which needs to be addressed in ongoing and future research projects and standards activities with the intention of providing a conceptual model for access to various types of structured data.

For the near future, the commitment and compliance to the IEEE/POSIX file system interface seems to be the right way to go. Although IEEE (i.e. the POSIX 1003.1 standard for operation system interfaces[4]) is explicitly designed for operations in stand-alone systems and does not address matters occurring in distributed systems, the specified primitives seem to be well suited for a general purpose distributed file system as well. The UNIX style file semantics specified in POSIX are becoming increasingly more accepted. It specifies the access of unstructured data (byte-stream files) which is appropriate for a wide range of applications.

Another general architectural consideration is faced with outgrowing local, communal boundaries in distributed systems. International operating organizations (such as OSF) have already established information system environments which require access and operations on shared data resources worldwide. The scale of such systems must not be limited. Distributed systems in production use already exceed 50,000 nodes. The dynamic reconfiguration of such systems (increasingly required by portable personal computers and mobile systems) must not be restricted and the access to data resources must be the same from any point in the world.

Problems such as carrying data from one location to another have been solved in distributed file system implementations in the past, but todays architectures have to deal with problems beyond physical interconnection and communication, with services which accommodate the inherent complexity of distributed systems. The global and uniform naming of data objects, protecting the integrity and privacy of resources and operations on them, providing efficient and reliable message transfer and remote operations, achieving accuracy, and synchronization of access times require the integration of various technologies in a coherent distributed computing environment in order to provide the desired functionality of a distributed file system.

2. Functional Requirements

To accommodate the goals mentioned above, the architectural design of a distributed file system must meet certain requirements which will be discussed in the following chapters.

2.1 Transparency and Reliability

Wherever possible, it is important to provide a file system service which guarantees the same behavior and exposes the same interface no matter where the files or directories are located. Standardized operations such as *open*, *close*, *read*, *write*, and *fcntl* for access and control and for locking must be fully supported.

Exceptions and derivations from these semantics which have been developed for file systems on traditional stand-alone systems are necessary to support the distributed environment appropriately. Some are obvious, such as error handling and extended security semantics to protect the resources. More sophisticated access control mechanisms (beyond permission bits) need to be introduced and UNIX-specifics such as *setuid* cannot be supported in the same way. Other extensions have to be made because of architectural requirements in a distributed system which must distinguish between the physical location and the logical view of the file system resources.

Synchronization of concurrent operations on data objects must be provided for in a distributed file system. File and record locking with same behavior must be implemented on distributed file servers. Write operations on files and directories must be synchronized and concurrent read and write accesses must be supported.

A distributed file system can only be considered for production use if it can guarantee the delivery of data. Applications must be able to rely on this behavior. If files have been deleted or appended, the correct execution of these operations or, in case of failure, a reliable notification of requesters is necessary. Concurrent invocations from independent clients or the recovery from server crashes must be at least as reliable as operations on local file systems.

The RPC based protocol in DFS assumes stateful behavior on file servers. Stateful in contrast to stateless requires the server to keep track of the context of client connections and requests. The server is responsible for cleanups and recoveries after connections have been broken or the system crashes. The RPC, in turn, supports this stateful behavior by offering semantics such as context handling and cancellation (alerts).

A *token* technique has been chosen to ensure the correctness and synchronization of file system operations. Several types of tokens (open, status, data, lock) are available, which are used for granting access rights to clients for a particular file and even segments of files. The file server obtains these tokens, passes these to requesting clients and keeps track of their status. Before granting write tokens, outstanding read tokens will be revoked to synchronize and guarantee consistency. This token technique is a key element in the DFS protocol which enables servers to keep their load low and reduce the network traffic by guaranteeing correct semantics. Based on these tokens, mandatory and advisory locking for files and records of files is provided for.

2.2 Availability

Clients of a distributed file service cannot predict or control the server activity. Servers can be shut down, crashed, or the network link can be broken. In most cases the users (clients) in distributed environments don't have access (physical, permission) to the appropriate server systems, which is far more true in nation- or worldwide distributed systems. However, it is fatal if the architecture of the distributed file system does not prevent excessive down times and does not have provisions for backup and replication servers. Furthermore, the access to resources has to be the same from everywhere. No matter if one is logged in at his workstation in the office or from any other connected node in the world, the same data should be available with the same view of the file space.

In DFS, global and high availability has been achieved by introducing the DCE Local File System (LFS), a file system tailored to support file servers, which performs specific tasks not available on traditional file systems such as Berkeley or SysV. In DFS, clients and file servers have distinct roles. Clients are basically responsible for keeping the network traffic and server load low by caching as much sensible information as possible, whereby servers are dedicated to support the synchronization and availability of their resources. These server functions are by nature much more complex and enhanced than on stand-alone file systems.

DCE Local File System introduces two significant enhancements for high availability:

- **Logging**

  A logging technique is used for write ahead logging of system metadata, which provides for consistency of the file system structure. DFS uses logging to reduce the time for recovery from system crashes. Long running file system salvage programs (i.e. *fsck*) are obviated as long as no media failures occur. Logging and the associated recovery techniques will allow the system either to complete or to completely undo interrupted operations before resuming normal operations after rebooting the system.

  The logging system is designed with a small overhead, so as not to delay regular file access. Log operations are short time transactions, exceeding the performance of traditional file systems such as the Berkeley Fast File System.

- **Filesets**

Filesets are the most significant element of the DCE Local File System. They build a logical abstraction of the physical file and file system repositories. The purpose is to hide the complexity of the underlying physical file system structure (aggregates) which represent a partition of disk storage. Filesets are mountable subtrees of a directory hierarchy. The size of filesets is only limited by the capacity of physical media. Common usage of filesets are user home-directory trees, source code trees, or binary directories. Again, these filesets and their logical positioning in the global file hierarchy is completely independent of the physical location which might change over time for certain reasons (balancing server load, disk capacity).

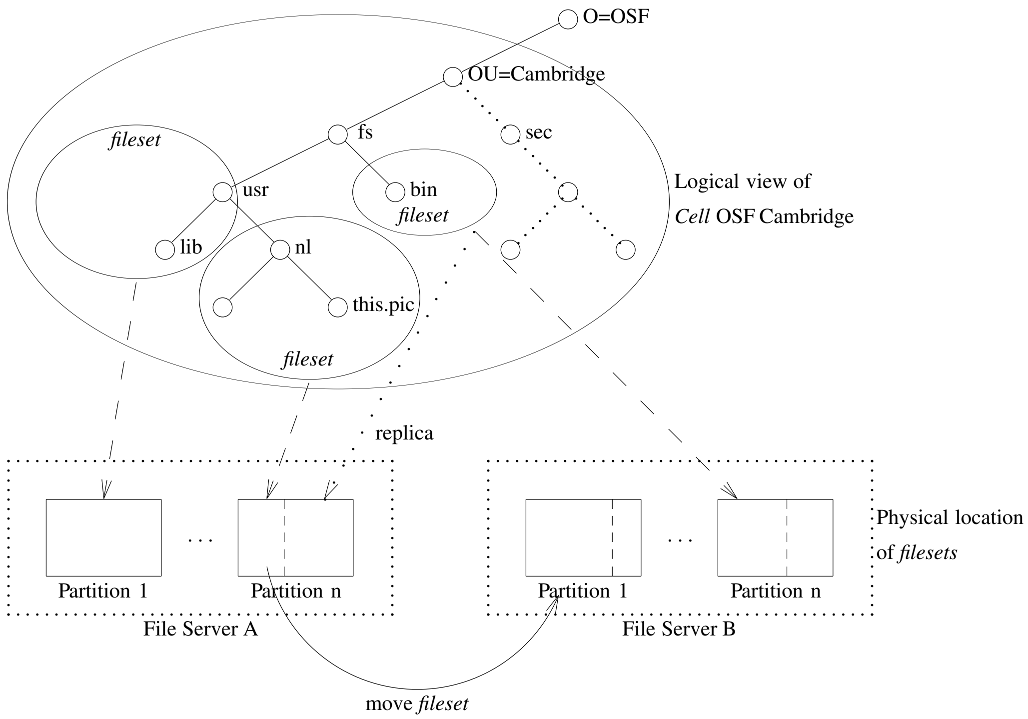

Figure 1. DFS Filesets

The logical organization of the file system into filesets is supported by operations on filesets (move, replication, and backup) to improve availability. Administrators shield the complexity of the underlying system structure for users by issuing these operations in the background without visible interruption. This is achieved by creating a snapshot (clone) of the original fileset. Instead of always copying the entire fileset, which would require long off-line times to prevent inconsistency, clones consist of pointers to the blocks of the original and only those data which have been updated in the meantime will be written to the clone (copy-on-write duplicate). This allows for fileset movement, backup, and lazy read-only replication.

2.3 Uniform and global Name Space

A distributed file system which spans the globe, must have provisions for a uniform view of the entire accessible shared file space. This requires integration with an appropriate name or directory service.

The approach taken by DFS is the organization into *cells*. Cells are elementary entities for performing desired operations and services inside certain boundaries, designed to conform to social, political and organizational divisions. Cells are interconnected to the entire distributed system via global directory services such as X.500 and are to be considered as open end systems in the distributed systems environment.

The cell architecture is derived from assumptions on how humans and their organizations interact with information systems. Different policies between individuals, companies, countries, and other organizational boundaries have been accommodated by introducing cells in the distributed computing environment. Basic assumptions are:

- Intra-organizational communication is more frequent than communication between units of information exchange.
- Different security policies apply to different organizations. The level of trust beyond these boundaries is usually lower than inside them.
- Organizations and subdivisions want to have administrative autonomy.
- Although exceptions may apply, internal communication has shorter distance and can usually use higher speed connections.

These assumptions lead to the cell model as being well suited for organizing the distributed environment. It allows efficient communication inside the boundaries by aligning with specific requirements as well as performing cost effective communication with the entire connected world. Compliance with Open Systems Interconnection has a key role in providing these services. Dependencies are basically caused by transport providers and underlying technologies.

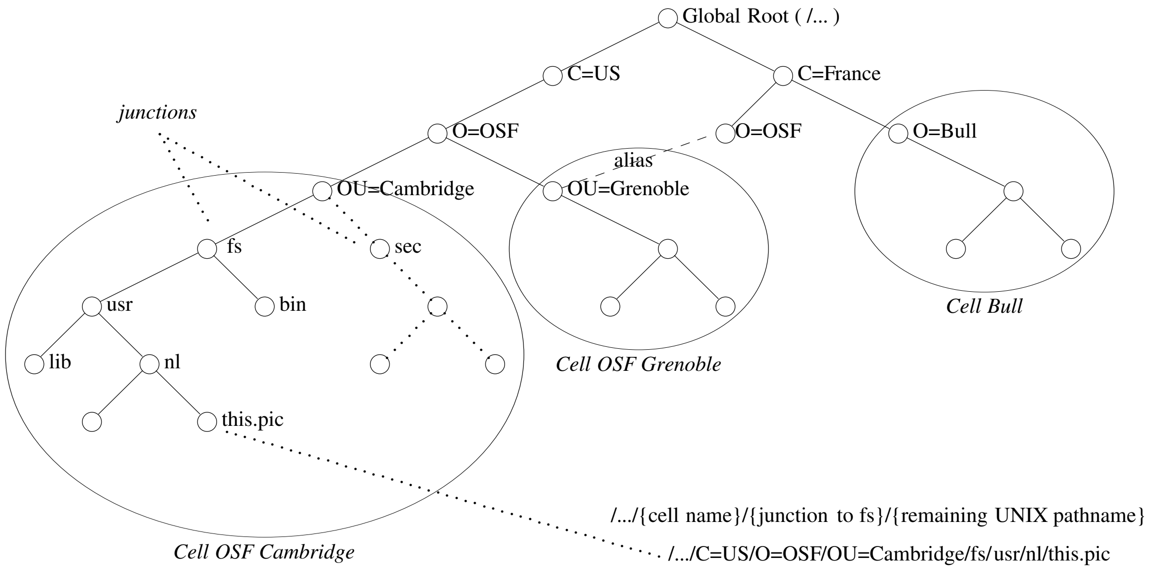

Figure 2. Uniform Shared File Space

The internal view of the cell name space is uniform but even different needs by various categories of resources is supported. Categories such as user registration and certification, service and process administration, and file system have distinct policies and must be implemented differently. DFS, for instance, provides compliance to the POSIX directory and file semantics and subdivides the file space into filesets as logical units. These different name services are integrated via *junctions* to the globally uniform name space.

2.4 Ease of Administration and Security

Distributing the file system has implications on the administration of the distributed systems and on the levels on security provided for this environment. It becomes most significant to integrate reliable and mature management and security technologies if worldwide access to the connected community has to be taken into account.

Both coherency and autonomy are features which must be solved in the design of the entire system. Coherency is necessary to provide the same tools, the same access rights and the same file tree structure from everywhere. Autonomy, on the other hand, is an absolute must, not only because of different levels of trust between distinct groups, but also because of various administrative domains inside and between organizations for purposes such as flexibility and accounting.

The approach taken by DFS is the integration of the file service with other proven technologies such as secure RPC, naming and security, and the design of the file system itself, which emphasizes the role of an improved file server, rather than providing complex and irreducible tools which are supposed to compensate for insufficient design.

The file server file system in DCE File System plays a key role in achieving these goals. The bundling of clusters of files into filesets and the associated operations such as movement, replication, and backups streamline the complexity of administering the shared file space. Administrators can make the distinction between logical and physical file system entities and can manage the fileset integration via designated location servers. They can balance the load simply by moving filesets and can recover from system crashes with a minimum of disruption of users.

The support of Access Control Lists, compliant to the POSIX recommendations, is another feature of the DCE Local File System which supersedes the traditional file system permission bits. Accommodated with a mutual authentication, supported by the underlying protocol (Kerberos authenticated RPC), unauthorized access to resources can be detected and prevented and the right resources for service requests are insured.

The certification and administration units are cells. Cells are domains with autonomy from the entire distributed computing environment. They are end systems in the open system. The creation of cells is a very simple administrative operation, achieved just by advertising a new cell to the global name or directory service. Intra- and inter-cell communication can be set up on a different level of protection with restricted access rights for in-bound and out-bound connections.

2.5 High Degree of Interconnectivity

Distributed file systems which span the world have to deal with heterogeneity on all levels. It is not predictable which machine, operating system, or even file system will be connected at the other end of the line. A distributed file system must not make any assumptions on homogeneity and a strategy must be provided for to migrate from existent environments with specific applications to an environment which fulfills the requirements for cooperative and distributed systems.

DFS has a highly modular design and file and file system operations are invoked via a standard virtual file system switch. Although the DCE Local File System plays a significant role in providing the desired behavior of the distributed file system, it is conceivable that other file systems such as Berkeley Fast File System, System V file system, or OS/2 file system can export their resources to the shared file space, by tolerating the given restrictions. DFS supports the BSD Fast File System, however extended features such as ACLs or operations on filesets (e.g. movement and replication), are only supported on the DCE Local File System.

NFS sites can mount the DFS file space and can issue operations on these files with the same NFS semantic. Unless security enhancements (mutual authentication) are implemented in NFS, the data resources exported to NFS sites must be publicly available, so as not to subvert and weaken the protection of DFS resources.

3. DCE File System (DFS)

To accommodate the requirements for DFS as a file service which is fully integrated to the distributed computing environment, some design decisions, distinct from those made for other remote file systems such as NFS, have been made.

First of all, the protocol requires a stateful behavior of the file server. The stateful file server keeps track of the context of client-server connections, which is essential to synchronize concurrent access to shared resources and to guarantee reliable operations with specified semantics. The token technique, locking, and detection of race conditions require the management of state information at the server.

Another significant consideration was to emphasize the role of the file server which goes far beyond features implemented on stand-alone systems. DFS implements a highly sophisticated file server which support functions such as synchronization, operations on logical volumes (filesets), replication, and protection of resources (authentication and authorization). Extensions in the vfs switch implementation have been introduced for these purposes.

A third design goal for providing a scalable system with high performance is reducing the network traffic and keeping the server load low. The token based synchronization, transfers of segments of requested data in reasonably sized bulks (*chunks*) and a very aggressively caching algorithm on DFS clients are introduced to achieve these goals.

3.1 Components

The four main components of DCE File System are the Local File System (LFS), the vfs+ (extended virtual file system switch), the DFS Protocol Exporter, and the client Cache Manager.

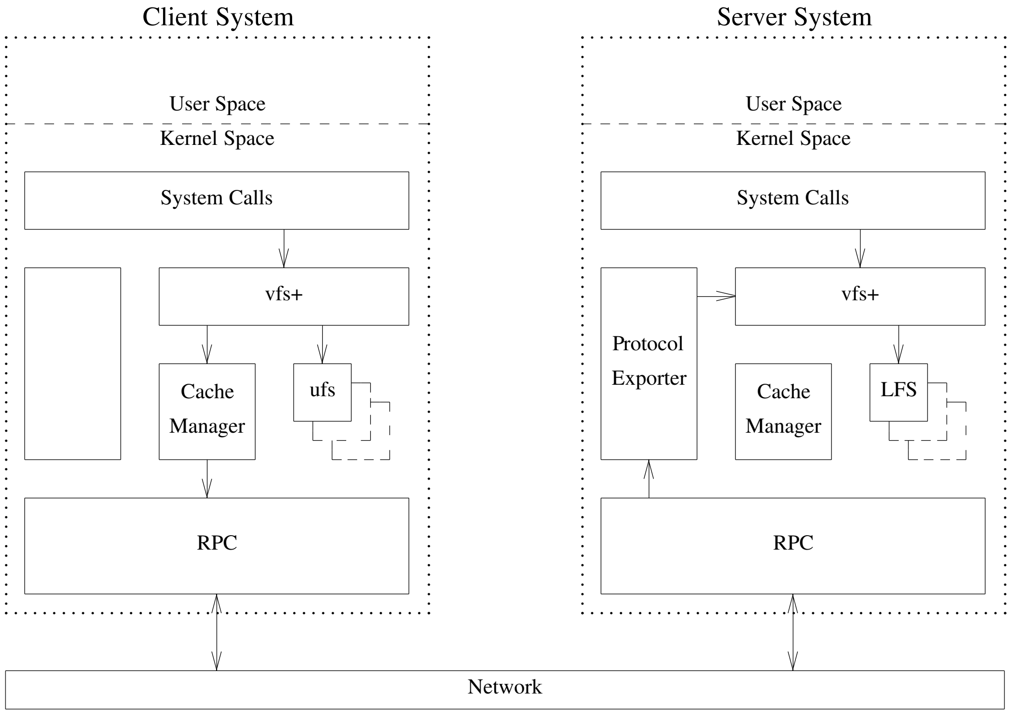

Figure 3. DCE File System Components

DCE Local File System is a special physical file system tailored for the purpose as file server file system. LFS supports components such as Filesets, Access Control Lists, and Logging. It is implemented as physical file system under vfs.

Vfs+ is the virtual file system switch (introduced by Sun Microsystems™) with extension for supporting synchronization, fileset operations, and authorization.

DFS Protocol Exporter is the implementation of the file server, responsible for issuing and synchronization of file operations, context handling and recovery.

Cache Manager introduces aggressive cache algorithms which allow for caching frequently used information for binding and naming, status and data operations.

3.2 The DFS Protocol

The DFS "wire" protocol specifies the message format for data and control information, exchanged between file system clients and servers. According to the design goals of DFS, the basic principles of the DFS protocol are:

- The server is responsible for the synchronization of operations. The order of operations is determined by the server and the server obtains, keeps track of and revokes tokens.
- Operations on directories are maintained at servers to avoid heavy network traffic for directory updates, which usually require many calls. Directories are cached at the server.
- Clients initiate calls via RPC and they keep track and verify update operations.

The operations supported by this DFS protocol are:

— Data and status calls (fetch and store)
— Directory calls (read and modify)
— Token calls (get, release, and renew)
— Authorization calls (fetch and store ACLs)
— Statistics call (get)

The DFS protocol is sufficiently implemented with OSF's DCE Remote Procedure Call. Beyond generic semantics which are supported by almost every available RPC implementation[5], DCE RPC provides for enhanced features which are necessary and very efficient for DFS. Important RPC features are the support for

- non-idempotent operations (at-most-once semantic),
- bulk operations,
- mutual authentication,
- context handles,
- cancellations.

In contrast to many other distributed services, in distributed file systems the transfer of large bulks of data are frequent operations. Thus, the support of operations on pipe data types is a very significant feature of OSF's RPC.

3.3 Integration with other Technologies

DFS can only meet the requirements for a system with industrial strength by its integration with other fundamental services offered by OSF's DCE. Technologies such as the DCE Directory Service, RPC, and the Security Service have already been mentioned. A further example, the DCE Time Services shows that a distributed file system which operates on worldwide scale must be integrated with other technologies, tailored for the purposes in the distributed environment.

The DCE Time Services synchronizes the time among the participating systems. File system applications such as *make*, *tar* or *sccs* require synchronized timestamps to ascertain the order of events. The underlying algorithm of the DCE Time Service handles the concept of inaccuracy which accounts for the inherent uncertainties in adjustments between clocks in a distributed environment. These truly distributed compound timestamps with inaccuracy values (128 bit) are stored in the DCE Local File System index data for files (anodes).

It ought to be emphasized that a distributed file system as any other service suited for an open distributed environment must be integrated with other proven but emerging technologies by providing well specified interfaces and protocols and the right level of modularity.

4. Summary

As outlined in this paper, DCE File System, embedded in other proven technologies for distributed computing, is indeed a powerful and essential application which allows the connected community to gain access to shared data resources worldwide. DCE File System takes into account the inherent properties of distributed systems, as a step ahead of traditional technologies which were designed and implemented as extensions to stand-alone capabilities.

DCE File System is derived from AFS® and the previous Andrew File System, designed and implemented at the Carnegie Mellon University in Pittsburgh (USA) and Transarc Corporation. It has already gained acceptance at several sites with established cells. Most recently the Defense Advanced Research Projects Agency (DARPA) in the United States funded a project to deploy and study an initial Wide Area File System, employing AFS. The results of this project will be used to predict the behavior of such a system for millions of potential users.

Conceivable areas for further consideration are:

- Extended semantics for file operations beyond the current standards (i.e. IEEE POSIX 1003.1) which accommodate requirements in distributed systems need to be developed and standardized.
- Emerging security technologies such as public key encryption and standardized access control services must be incorporated.
- Research and development of improved replication technologies (strong consistency and multiple write replicas) needs to be addressed.

REFERENCES

1. Russel Sandberg, *The Sun Network Filesystem: Design, Implementation and Experience*, Sun Microsystems, Inc. (1986)

2. Open Software Foundation, *OSF Distributed Computing Environment Rationale*, (May 14, 1990)

3. Michael L. Kazar, *DEcorum File System Architectural Overview*, USENIX Summer Conference (June 11-15, 1990)

4. IEEE, *Portable Operating System Interface for Computer Environments*, ANSI/IEEE Std 1003.1-1988

5. Andrew D. Birell and Bruce Jay Nelson (Xerox Palo Alto Research Center), *Implementing Remote Procedure Calls*, ACM Transactions on Computer Systems, Vol. 2, No. 1, (February 1984)

Towards a Worldwide Distributed File System
The OSF DCE File System as an example
September 27, 1990

CONTENTS

1. General architectural Issues 2
2. Functional Requirements 3
    2.1 Transparency and Reliability 3
    2.2 Availability 4
    2.3 Uniform and global Name Space 6
    2.4 Ease of Administration and Security 7
    2.5 High Degree of Interconnectivity 8
3. DCE File System (DFS) 8
    3.1 Components 9
    3.2 The DFS Protocol 10
    3.3 Integration with other Technologies 10
4. Summary 11

LIST OF FIGURES

Figure 1. DFS Filesets 5
Figure 2. Uniform Shared File Space 7
Figure 3. DCE File System Components 9