Distributed File System Requirements

UNIX International
Distributed Computing Work Group
Revision: 1.0 (91/12/10)

Published by

UNIX International
Waterview Corporate Center
20 Waterview Boulevard
Parsippany, NJ 07054

For further information, contact:
Vice President of Marketing

Phone: (1) 201-263-8400
Fax: (1) 201-263-8401

International Offices:

UNIX International Asian/Pacific Office Shinei Bldg. 1F Kameido Koto-ku, Tokyo 136 Japan	UNIX International Australian Office Suite 1 80 Berry Street North Sydney, NSW 2060 Australia	UNIX International European Office 25, Avenue de Beaulieu 1160 Brussels Belgium	UNIX International Pacific Basin Office Cintech II 75 Science Park Drive Singapore Science Park Singapore 0511 Singapore
Phone: (81) 3-3636-1122 Fax: (81) 3-3636-1121	Phone: (61) 2-922-5341 Fax: (61) 2-929-0635	Phone: (32) 2-672-3700 Fax: (32) 2-672-4415	Phone: (65) 776-0313 Fax: (65) 776-0421

NOTICE:
UNIX International is making this documentation available as a reference point for the industry. While UNIX International believes that these requirements are well defined in this first release of the document, minor changes may be made prior to conforming products meeting these requirements being made available.

Table of Contents

1. EXECUTIVE SUMMARY

This document summarizes the recommendations of the UNIX International Working Group on Distributed Computing for Distributed File Systems contained in SVR4.0.

The DCWG studied the current Distributed File System requirements from the following perspectives:

a) **Size of network** - It was determined that the DFS must operate efficiently in a Wide Area Network environment with as many as 10,000 nodes. This means that the DFS must be scalable.

b) **Interoperability** - Implementation of the DFS must be possible across a wide range of platforms, both UNIX Systems and non-UNIX Systems. Of particular importance is operation in PC/MAC/Desktop environments.

c) **Security** - It was determined that the DFS must provide authentication, authorization, assurance and auditing to inhibit violation of the DFS.

d) **Administration** - It was determined that the DFS must provide easy to use administration.

These requirements are refined for internal consistency and described as a set of desired Distributed File System features. The resulting features are described below, Figure 1, page 1 with priorities that reflect the level of motivation.

Title	Priority
Caching on NFS	Critical
NFS over connection-oriented transport	Critical
Better install, manage, admin tools	Critical
Admin of systems in a coordinated name space	Critical
Replication on NFS	Desirable
Automatic Failover on NFS replicas	Desirable
Location Broker for Replicated copies of NFS files	Desirable
Unified Name Space	Desirable
Pluggable authentication interface for NFS	Desirable
Short-Term Fix for idempotency	Desirable
Administration of Replicates	Desirable

Figure 1 - Complete List of DFS Requirements

2. INTRODUCTION

2.1. Background

The DCWG was created by the UNIX International Steering Committee. The Steering Committee provided the DCWG group with a charter to "Establish requirements for Distributed File Systems." With this charter, the UNIX International members assigned leading industry experts in Distributed File Systems as representatives to the DCWG.

2.2. Participants

Companies whose representative actively participated in the DCWG activities and discussions are:

Aggregate Computing	CDC	Intel
KMart	Locus	NCR
NEC	Retix	SPARC Intl
Sun Microsystems	Texas Instruments	Unisys
Veritas Software		

Figure 2 - List of Attendees

The DCWG is chaired by Joan Wrabetz of Aggregate Computing.

2.3. Contents

This paper contains the following sections:

Section 1 Provides an Executive Summary of the document.

Section 2 Contains an introduction to the document.

Section 3 Summarizes the principle decisions of the DCWG on DFS.

Section 4 Presents the basic requirements and recommendations.

Section 5 Describes features and provides requirements and recommendations for the features.

3. PRINCIPLE DECISIONS

As Distributed Computing becomes more and more a part of everyday computing, new functionality will be required of the Distributed File System. The DCWG has identified that NFS required simple extensions such as caching, replication and support for large distributed environments. The DCWG concluded that such extensions could be added to NFS with no or minor additions to the DFS protocol. Other functionality which will require protocol changes will be addressed in a future white paper.

3.1. Extend NFS

NFS is currently shipped with SVR4. NFS has installed over 1.5 million copies with over 300,000 additional copies being purchased each year. Since NFS will be the most widely used DFS for the foreseeable future, it is the DCWG's belief that NFS will continue to be a major DFS in the market place. Because of this, the DCWG concluded that continued extensions to NFS must be made to meet the changing needs of the users of NFS.

The group concluded, that if NFS were given the following extensions, NFS would continue to provide excellent functionality for files requiring loose consistency between multiple clients.

1. Increase the number of nodes that NFS server will support by providing file caching on the client and support for replicating the file on multiple servers.

2. Provide support for additional security authentication techniques besides DES, e.g., Kerberos.

3. Provide easier to use administration and installation of NFS, e.g., automounter enable by default, name service with default maps, etc.

4. Make NFS transport independent so it can be run over transports like OSI, Internet, TCP/IP etc.

5. Make NFS more robust by enhancing NFS to support larger files, handles, etc.

These enhancements to NFS may result in minor additions to the protocol. These enhancements must be backward compatible with current NFS. However, currently installed versions of NFS may not be able to fully utilize all of the new capabilities of the new DFS.

3.2. Support Large Distributed Environments

3.2.1. Locality of the Distributed File System

Distributed File Systems are used in a number of environments. Some of these environments are composed of 10-10,000 machines dispersed geographically all over the globe. Others may consist of 10-1,000 machines all existing in a single office complex. Because of the diversity of these environments different connection mechanisms will be used. The cost of reliable data exchange and the cost of obtaining acceptable response times for file accesses will result in the following needs that the Distributed File System must fulfill:

1. Ability to keep a local copy of a file via replication and caching.

2. Ability to continue operation using a replica should a failure occur.

3. Efficient mechanism for establishing sub-domains within the total net.

4. Intra-domain and inter-domain authentication and secure file access and data exchange.

5. Inter-domain naming and mapping capabilities.

6. Administration of distributed computing environments regardless of size or dispersion.

3.2.2. Size of Distributed Computing Network

The Distributed File System will be used in networks of 10-10,000 nodes. Large environments require additional functionality. The DCWG believes that a DFS must provide the following functionality for operation in large environments.

1. High availability.

2. Increased data locality.

3. Support 10,000 nodes.

4. Ability to perform most maintenance activity without stopping the system's use of the file.

5. Ability to administer the entire system from a single site.

3.2.3. Heterogeneous Systems

SVR4 will be connected to non-UNIX Systems. It is necessary that SVR4 interoperate with these systems. The following capabilities will be needed.

1. A limited support of remote devices.

2. Support of large file handles.

4. BASIC REQUIREMENTS AND RECOMMENDATIONS

There is a set of basic requirements common to all features. This section states those basic requirements.

1. A guide for the user should be provided on how to administer and install each of the capabilities recommended in this document.

2. A standard set of metrics must be available for tuning of the Distributed File System.

3. These features should be compatible with other UNIX System V requirements developed by other working groups.

4. These features should not break any current user applications and existing applications should continue to run unchanged. However, some interfaces may require extensions or modifications which cause minor incompatibilities with existing practice.

5. Metrics must be provided via the PMWG's defined interface to tune the cache size, access time, network, and server workload to desired levels.

6. Simple easy-to-use configuration and administration tools must be provided to install and manage these features.

7. Servers/clients which support these new features must provide backward compatibility and support for clients/servers using older versions of the DFS. However some features may only have limited functionality when used in conjunction with an older version of NFS.

8. It must be possible to upgrade the network with these new features in such a manner that not all of the network has to be upgraded at the same time.

5. DESCRIPTION OF FEATURES

5.1. NFS with Caching and Replication

Caching is used to increase the number of clients which a server can provide acceptable performance. Client resident file caching is much like the caching used by hardware multiprocessors to reduce the memory and bus loads. Like hardware caches the type of caching used depends on the manner in which the file is used.

One of the basic requirements is to increase the number of nodes that NFS supports. This can be done by simply providing NFS with a large enough cache to encompass the working set of the files being accessed and using a good caching algorithm, see Figure 3, page 8.

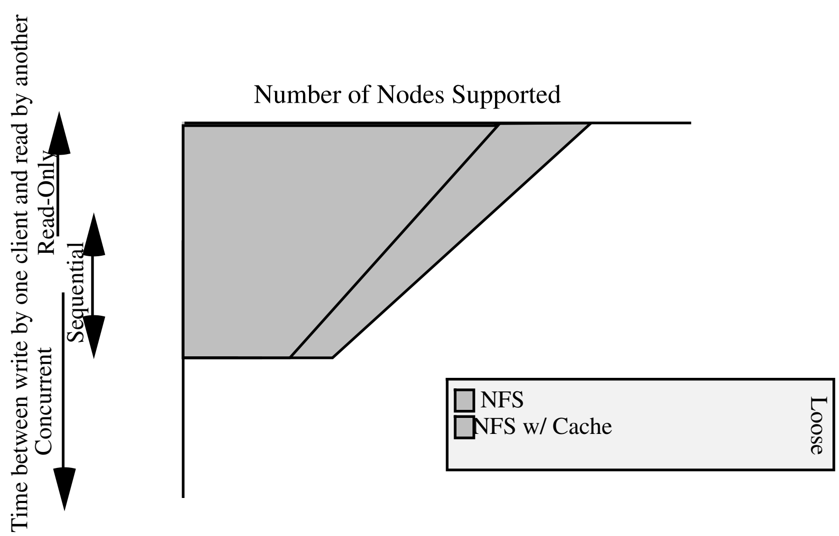


Figure 3 - NFS with Caching

The DCWG believes that caching will increase the number of nodes supported by any single server. However, in most environments a small percentage of nodes will be responsible for most of the load. Only nodes which are responsible for the network/server load should have to bear the memory and disk cost necessary to support caching. Thus, it is desirable that caching not be mandatory on all nodes which wish to access a file, only on nodes where caching will provide a benefit.

Replication is used for a number of reasons. The key reasons why the DCWG believe that replication is an important part of any Distributed File System are:

1. Increase number of nodes beyond that provided by caching
2. Increase response time for clients in a WAN by providing a local copy
3. Increase availability by providing an alternate copy of the file

NFS is currently used to access replicated files. However, NFS provides no automatic consistency mechanism between replicas nor does it provides automatic failover to an alternate copy.

If a failover mechanism was provided on NFS, NFS could be used as a replicatable Distributed File System for read-only files. If automatic consistency between replicas were added to NFS, NFS would provide an acceptable DFS supporting loose consistency replication. This would allow NFS to support more nodes than the current implementation of NFS or NFS with caching, see Figure 4, page 9.

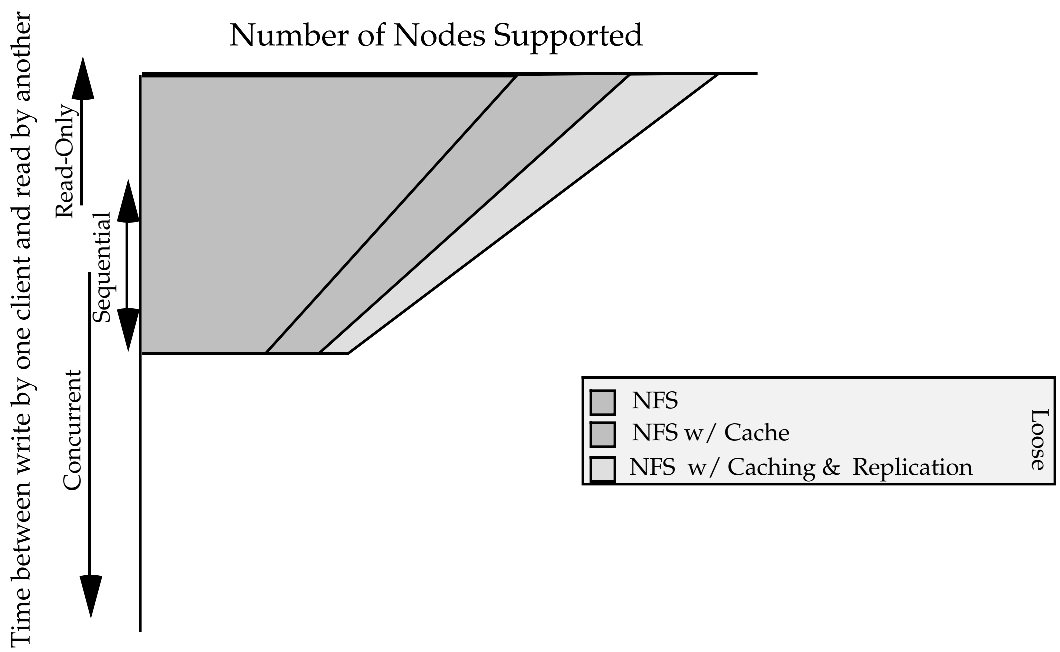


Figure 4 - NFS with Caching and Replication

5.1.1. NFS with Caching

5.1.1.1. Priority Critical

5.1.1.2. Motivation

The primary motivation for this type of caching is read/write caching on read-only files, private files or files used by a sequential set of users. Caching reduces the access time, network traffic and server load in heavily loaded read-mostly environments allowing servers to support more nodes.

A second motivation is that, for caching of read-only files, private files or files used by a sequential set of users, more nodes per server can be supported for environments that currently use NFS.

Additionally, read/write caching on read-only files, private files or files used by a sequential set of users can be implemented without significantly increasing the resources in the client. This makes it usable in PC and Desktop environments where resources are limited.

5.1.1.3. Description

The client will cache frequently used files on a local storage medium. The management of the cache will reside on the client side. Since this feature is intended to be used for caching of read-only files, private files or files used by a sequential set of users, there is no need for better consistency checks than what is currently supplied by NFS.

Read-only files, private files or files used by a sequential set of users are changed in a controlled manner. The problem of guaranteeing that all applications accessing the file will see correctly updated data is handled by administrative procedures and need not be part of the file system. The administrative procedure may be as simple as running applications which use the same file in a sequential manner. File consistency is maintained in this environment because the time between the write of the last user and read of the new user is longer than the time set for NFS's periodic consistency update.

5.1.1.4. Current Practice

Today when the network/server becomes overloaded with NFS requests, it is necessary to buy a more powerful server, reconfigure the net into subnets, or upgrade the bandwidth of the net. All of these solutions increase the cost of the system and limit the size of distributed environments. NFS does currently provide a buffer cache in local memory. However, a buffer cache does not provide as optimum a hit ratio as a large working set cache. Because of the size of a typical working set cache, typically a local disk is used.

5.1.1.5. Impact

The impact is strictly on the client. There should be no need for protocol changes. Since this capability is intended as operating with on currently installed NFS, no protocol change is also an assumed requirement.

5.1.1.6. Requirements and Recommendations

1. The option of whether to cache should be up to the client not the server.

2. If local persistent caching is provided via some permanent storage media the cached data should be available even after a reboot. Since it is desired that the reboot time be minimum, the cached data should be reevaluated at the time of access rather than during reboot.

3. Servers/clients which support caching must provide backward compatibility and support for clients/servers using older versions of NFS.

4. Metrics must be provided via the PMWG's defined interface to tune the cache size, access time, network traffic and the server workload to desired levels.

5. Simple easy to use configuration and administration tools must be provided to install, and manage the cache and associated system.

6 It must be possible to gracefully shut down the client or the server and still maintain data integrity.

7. The user/administrator should be able to choose one of the following caching environments on a per client basis.

   a. Scalability (reduce server/network load)
   b. Minimize client disk usage (for cache)
   c. Minimize response time

This will cause automatic selection of cache policies affecting the following.

a. Block Size
b. Read Ahead policy
c. Write Back policy

8. On a realistic benchmark with response time equivalent to NFS, caching with loose consistency must be able to support 2x-3x more nodes per server than NFS.

9. The caching write policy must provide consistency at least consistent with that currently provided by NFS today.

10. The frequency of checking the consistency between the client and the server should be selectable by the client.

11. The user needs a mechanism to instigate a consistency check.

5.1.2. Replication on NFS

5.1.2.1. Priority Desired

5.1.2.2. Motivation

There are three key motivations for replication:

a. Increase the number of nodes in a network beyond what caching provides.
b. Provide an alternate copy to increase the level of availability of specific files
c. Allow faster access in WAN environments via maintaining a localized copy

5.1.2.3. Description

A file will have replicated copies of itself on a network. These replicas are updated only at fixed periodic times. This allows for a large number of replicas to exist and substantially reduces the server load that must be dedicated to maintaining consistency. Such replication works well in environments that have read-only files, files used in a sequential manner, and files updated in a background mode by a single writer. This type of replication is also ideal for use in providing local copies to reduce the response time in WANs

5.1.2.4. Current Practice

NFS is used to replicate files today, but it does not support automatic consistency or failover today. AFS provide loose replication with automatic replica consistency, caching, and failover. TCF provides tight replication with caching (only incore) and rollover and it also provides simultaneous replication under program control. Some databases provide a form of simultaneous replication providing high availability via mirroring a file on two different servers.

5.1.2.5. Impact

Providing replication would require changes to the VFS interface, naming, caching, as well as to the DFS protocol.

5.1.2.6. Requirements and Recommendations

1. The implementation should be pluggable to an extension of the current VFS interface.

2. Must allow read/write access to a replicated file from an installed NFS 2.0 systems.

3. Sufficient metrics must be available to adjust the location and number of replicas needed for good response time to client requests.

4. Simple easy to use configuration and administration tools must be provided to install, and manage the replicated file.

5. The list of available replications must be available all of the time.

6. Should have the ability to perform dynamic load balancing of the clients across a set of servers by use of some pluggable load balancing package. The load balancing package should not be a fixed part of replication mechanism.

7. Information must be available for decision making of such things as dynamic load balancing.

8. It should be possible to specify the maximum and minimum number of replicas the system will support.

9. It should be possible to specify context specific locations that replicas should exist e.g., machine architecture.

10. It is desirable to be able to partition large files up across multiple servers and to replicate only specific partitions of the file.

11. Memory mapped files must be supported.

12. It should be possible to dynamically tune the replication policy under programmatic and administrative controls.

13. It should be possible to add a new replication method without rebooting.

14. Must be able to coexist with NFS caching. Should be able to coexist with any type of caching.

15. On a realistic benchmark with response time equivalent to current NFS, replication should be able to support 2x-3x more nodes than NFS.

16. Provisions should be made that third party developers can provide add on products that will perform migration of a replica, load balancing on replica creation, automatic dynamic load balancing of clients among replicas, advisories for tuning the use of replicas, and other decision making mechanisms and capabilities.

17. Ability to write data to any replica from any client.

18. It is desirable that at least some minimal load balancing mechanism exist to perform load balancing of accesses among replicas.

19. All clients must automatically failover to an alternate copy of the replicated file if no updates have occurred since the last time all replicas were consistent. Automatic failover should occur, even if some of the replicas have not completed the last consistency update, to a replica which has completed the consistency update.

20. Handle a dozen or more copies without severe overhead. Handle hundreds of copies gracefully (at least with loose consistency semantics).

21. Run using existing file systems.

22. The DFS should be extensible so future versions may allow access to a newly created replica prior to completion of the creation of the entire replica.

23. The file and all of its replicas should be accessible via the same name.

5.1.3. Location Broker for Replicated Copies of a File

5.1.3.1. Priority Desired

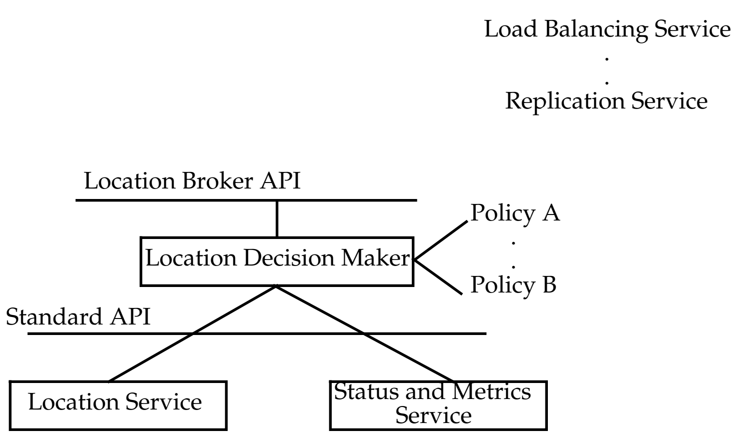
5.1.3.2. Motivation

The primary motivation for a location broker is to have the capability of choosing a specific replica that fits the needs of a requester.

5.1.3.3. Description

To perform failover and updating of any file, it is necessary to know where replicated copies of a file reside. This service will keep track of the location of all replicated copies of a file and which clients are currently assigned to access which copies. Any client, server, administrator, etc in the system may query this service. The location broker will return the location of the replica which best matches the list of requirements provided by the client, server, administrator, etc.

The location broker will use a standard location service to obtain a list of the locations of all replicas. The broker will also reference system metrics and status information about the replicas. The broker will match the list of requirements that the requester desires to meet against the metric and status information. The broker will return to the requester the location of the replica which best matches the list of requirements.



5.1.3.4. Current Practice

Currently the location of replicated copies of a file are kept manually by the administrator.

5.1.3.5. Impact

This will require new protocol extensions to NFS.

5.1.3.6. Requirements and Recommendations

1. The location broker must be highly available.

2. Must be able to provide the location of the latest version available in the network and also of the current version being used by the client.

3. The broker should be able to find a replica on a per-file basis give the following list of parameters: the closest, the least busy, the latest, same version as currently being used.

4. The location service should be able to obtain system metrics.

5. The broker should be aware of which replica the requester is currently using.

6. The administrator must be able to establish broker advisories and broker policies.

7. For PC environments, it is desired that the broker reside on the server. This saves disk space and memory. However, in other environments it maybe desirable for the location broker to reside on the client.

8, The policies of determining what makes a best match will change from environment to environment, so the policies that the location broker uses must be configurable.

5.1.4. Automate Failover to an Alternate Replicated Copy of a File

5.1.4.1. Priority   Desired

5.1.4.2. Motivation

The motivation for this feature is to make failover to a replica transparent to the application should a file become inaccessible.

5.1.4.3. Description

A copy of the file or file system is kept on multiple servers. Should the replicated file become inaccessible, all file requests will be redirected to an alternate copy of the file.

5.1.4.4. Current Practice

Currently, when a file becomes inaccessible, all clients using the file on the server will hang. This results in loss of productivity and availability. The current practice is to reboot. However, no matter how fast the administrator, the administrator is rarely fast enough to minimize the loss of productivity to an acceptable level for most environments. Currently several DFSs provide automatic failover, Locus, Coda, DCE.

5.1.4.5. Impact

This assumes that there is some mechanism to locate an alternate replicated copy of the file. The simplest mechanism may be that the client keeps a static list locally. A more dynamic mechanism may involve a location broker which provides a prioritized list of alternate replicates. This may require changes to the DFS protocol.

5.1.4.6. Requirements and Recommendations

1. Failover to an alternate replica must be transparent to the client application.

2. Failover will be to an equivalent copy.

3. It should be possible to failover to a replica on a per-file basis.

4. The administrator must be able to establish failover advisories and failover policies.

5. If a location broker exists, failover may make use of the location broker to determine the correct copy, version, location to failover to.

5.1.4.7. Other Issues

Replication is widely used by read-only, and sequentially used files. NFS is typically used as a DFS for read-only and sequentially used files. Just by providing a simple failover service for NFS, we can satisfy most customers needs for replication failover. The service could be nothing more than allowing the administrator to configure a specific location to failover to if the primary becomes inaccessible. This should be able to be accomplished quickly and in a timely manner.

5.2. NFS Enhancements

5.2.1. Pluggable Authentication Interface for the DFS

There has been a been a great deal of interest in security in the last few years. Over time, new and better technology will be developed that provides an increased level of security. It is not the DCWG's intent that the requirements described in this section be the ultimate answer to security in the Distributed File Systems. Rather the DCWG recommends that the security in Distributed File Systems be constructed in such a manner that is it easily upgradable to whatever new technology becomes available without having to replace the entire Distributed File System.

The DCWG also recognizes that the Distributed File System is only one component in the entire system which needs security and that the real issue which needs to be addressed is 'how the Distributed File System's security integrate with the rest of the system's security capabilities'. However, such an issue is beyond the scope of the DCWG and should be addressed by a set of security experts. So the DCWG has established a set of recommendations from the viewpoint of the Distributed File System as input into such a group.

The DCWG believes that security for a Distributed File System should address the areas listed below:

1. Authentication and Identification that anyone/service/client requesting access to a file.

2. Verification that anyone/service/client requesting access to a file is authorized to access the file.

3. Assurance that anyone/service/client requesting access to a file is not being spied on, copies of his data are being made by looking at the transmission line, or someone has stolen his data and substituted something else.

4. Auditing and Reporting of sufficient events in such a manner that should someone violate security that we can detect the violation and than take steps to prevent further violation.

5.2.1.1. Priority Desirable

5.2.1.2. Motivation

To dynamically make the DFS easily extensible for new authentication techniques by simply plugging in a new authentication method.

5.2.1.3. Description

When authentication is required, the DFS will call the common authentication interface specifying the desired authentication technique.

5.2.1.4. Current Practice

This is difficult as different environments require different authentication mechanisms.

5.2.1.5. Impact

This will have no impact on the use or protocol of the DFS.

5.2.1.6. Requirements and Recommendations

1. Must be able to provide authentication for mounts that occur at boot time.
2. The authentication interface must provide DES and Kerberos as a minimum set of authentication techniques.
3. The DFS must use this standard authentication interface.
4. The server must be able to export to clients different authentication services for the same file system.
5. Different users may use different authentication services where multiple users exist on the same client.
6. Authentication should be transparent to the user

5.2.2. Short-Term Fix for Idempotency

5.2.2.1. Priority Desirable

5.2.2.2. Motivation

By providing idempotency NFS will have better reliability and functionality. Idempotency is necessary for data consistency. The motivation for this short-term fix to idempotency will be superseded by providing single site semantics.

5.2.2.3. Description

This is a fix to areas of NFS which have idempotency problems.

5.2.2.4. Current Practice

Currently the only way to provide idempotency is to provide state in the Distributed File system.

5.2.2.5. Impact

Requires protocol revision and will result in making NFS stateful.

5.2.2.6. Requirements and Recommendations

1. Provide an exclusive create

2. Provide an exclusive remove.

3. Provide a set of rename procedures.

4. Provide an access procedure which will allow determination of access prior to attempting the I/O.

5. Provide an exclusive link.

5.2.3. NFS over Connection-Oriented Transport

5.2.3.1. Priority Critical

5.2.3.2. Motivation

The primary motivation is to provide NFS over TCP/IP and gain the benefit of TCP/IP's more sophisticated congestion control for Wide Area Networks (WANs) environments that require NFS.

A secondary motivation is to provide NFS better reliability, robustness, traffic, support for semantics, and performance for WAN environments, e.g., Internet and OSI, by putting NFS on a connection oriented protocol.

Provide a migration strategy for providing NFS on an OSI transport. There is a large market which will only use connection-oriented protocol. NFS should provide for such an environment.

5.2.3.3. Description

It is desirable that NFS have better performance over WANs. Better performance over WANs could be achieved by using a connection-oriented protocol like TCP/IP. Using a connection-oriented transport, would result in a reduced number of redundant retransmissions.

5.2.3.4. Current Practice

NFS is not currently considered adequate as a Wide Area File System (WAFS). The reason for this is that connection-oriented protocols have been enhanced over the last few years to have good congestion control. Currently NFS works over UDP. UDP lacks transport support for such things as congestion control, connection establishment, negotiation and transport support for reliable transfer. AFS, which claims to be a WAFS, uses a connection-oriented transport.

5.2.3.5. Impact

Because of the need to interoperate with older versions of NFS that use connection-less oriented transports, it will be necessary to support both connection and connection-less oriented transports.

Using a connection-oriented transport complicates the recovery when one side of the NFS connection goes down.

The degree to which the connection-oriented transport improves performance is dependent on the degree to which the connection-oriented transport has been optimized to give superior performance over a WAN.

5.2.3.6. Requirements and Recommendations

1. NFS must be enhanced to support a connection-oriented networking protocol (including TCP/IP and OSI).

2. Congestion, recovery and reestablishment of connection issues must be addressed.

3. Use of SVR4's transport independent selection mechanism.

5.2.3.7. Key Issues

Because of the cost of a WAN network we need to be able to support the following:

1. Protocol changes to reduce network latency, # of packets transmitted, and the round trip overhead.

2. Support large transfers efficiently. Connection-oriented transports may favor data transfers larger than the 8KB currently supported on UDP based NFS. Chunking techniques using large data transfers may also be beneficial for local caching.

3. Support read ahead so we can move multiple blocks more efficiently.

4. Connection management strategy for connection-oriented protocols may be critical for OSI. OSI networks charge for duration of connection rather than data transfers, so efficient use and connection management will be critical to a cost effective OSI transport for the DFS.

5. Data Compression will also reduce the amount of network traffic.

We need to develop similar techniques for NFS over UDP.

5.2.4. Unified File Naming

One of the key goals of a DFS is to provide users operating on a wide variety of systems access to a common, shared file system. Key to the realization of this goal is the selection of a file naming convention that will define a unified name space.

File naming in traditional uniprocessor systems provide a convenient means for users to identify files without worrying about the exact location of the file in terms of device addresses. In general, file naming provides a mapping of a textual name to some lower level numerical identifier. This is in turn mapped to the disk blocks comprising the file. In a DFS, a new parameter is introduced, the location of the file within the network. Thus, a file name within a DFS can potentially be used to help hide the location of the file within the network. In a DFS, the mapping will identify where in the network the file is located, and if the file is replicated, it will indicate where one or more of the replicas is located.

Two possible attributes of a DFS file naming mechanism are location transparency and location independence. Location transparency means that the file name provides no information about the physical location of the file within the network. Location independence means that the file name does not need to be modified in the event the file is moved one or more times during the lifetime of the file.

5.2.4.1. Priority Desirable

5.2.4.2. Motivation

One of the fundamental characteristics of a true distributed system is the high degree of transparency provided in the operations made available to users. It is through this transparency that one of the key advantages of a distributed system, scalability, is achieved. Transparency within a DFS is realized through the ability to present to the users of the distributed system access to a single, shared file system. This is best achieved by providing a common, shared name space allowing users to reference files with the same pathnames regardless of the location of the user within the distributed system.

5.2.4.3. Description

The DFS should provide a common, shared name space to all hosts in the distributed system. This name space will allow users located anywhere in the distributed system to use the same name to reference a file. This naming scheme should provide both location transparency and location independence.

In providing access to a common, shared name space, authentication and administration responsibilities will need to be considered. In particular, the system will need to provide the same level of security in controlling access to files as is provided on uniprocessor systems. A file naming service will need to manage the following:

Common user ID space
Authentications
Administration information

5.2.4.4. Current Practice

There are three basic naming schemes used in contemporary distributed systems:

1) Combination of host name and local name: a symbolic identifier for the host on which a file is located is combined with the standard UNIX System-like pathname used to identify a file within a host (e.g., host:local-name). This is used in Ibis. This naming strategy is neither location transparent nor location independent.

2) Mount remote directories to local name spaces: an operation is performed by a privileged user (e.g., root) to mount a directory on a remote system to a local directory. The user references the files in the remote directory by naming the local directory in a pathname. This is the current practice within NFS. When combined with utilities such as the automounter, the mount operation can be transparent to most users. A characteristic of this technique is that the name space used to reference the shared files need not be identical. It is the client system's responsibility to establish the naming.

3) Single global name structure spanning all files in the system: this is the strategy used in the Andrew File System (AFS). AFS separates the name space into a private, local name space and a global, shared name space. The latter name space provides location transparency and location independence. The global, shared name space consists of a collection of components called volumes. Location independence is limited to the granularity of a volume. That is, one can relocate entire volumes and leave a name unchanged

5.2.4.5. Impact

Protocol changes will be required in order to provide the global, shared name space. Less transparent access using the current mount technique should be supported to provide backward compatibility.

5.2.4.6. Requirements and Recommendations

1. The naming strategy must provide a common, global name space within an administrative domain.

2. Scalable and extensible.

3. Naming for variable grain objects/files/file systems.

4. The naming scheme must support mechanisms within the DFS to provide replication and high availability.

5. Retain System V file system semantics.

6. The naming strategy must provide location transparency.

7. The naming strategy must provide location independence. Providing location independence at the file level would be ideal, but providing a somewhat higher granularity (e.g., an AFS volume) is acceptable.

8. The domain should be able to accommodate the following:
    - Local Files (/tmp)
    - Context dependent Files
        - Hidden Directories
        - Machine architecture dependent files
        - Binaries
    - Shared only Files (not private to anyone)

9. Allows customizable private name space.

10. Process context information available for private file space

11. Administrative procedures should be similar to those used on single systems.

5.3. Administration of the DFS

The following section describes various enhancements to the NFS's administration that would give the DFS a wider appeal to the different Distributed Computing environments and make the DFS easier to install, administer, and manage.

Some capabilities and requirements in this section may be better addressed by UNIX International's System Management Work Group. However, the DCWG felt that administration is part of any component of the distributed environment, so the DFS document should include key administration requirements as perceived by DFS experts. Hopefully, the SMWG will use the contents of this section as a starting point for their requirements of how to administrate a DFS.

5.3.1. Better Installation, Management and Administration Tools

5.3.1.1. Priority Critical

5.3.1.2. Motivation

In order to reduce the level of support vendors must provide customers, better installation, management and administration tools are required.

The DFS should be so simple to install that the DFS can be simply plugged in and play for simple configurations. This requires that the local management and administration to be done by typical clerical staff.

End users buy things because of the ease of use and beauty of the package more than because the product is better. We must realize that computer systems are becoming commodity items and packaging sells.

5.3.1.3. Description

Client systems are typically installed and administered by personnel with only clerical level knowledge of how the computer operates. It is therefore necessary that installation and administration of the client systems do not require knowledge or skills beyond normal clerical personnel.

It must be possible for clerical staff to simply plug in a client system and expect it to play without reading a manual.

5.3.1.4. Current Practice

Current installation procedures for NFS are too complex for most end users as it requires fluency with technical jargon that most have never heard of before.

5.3.1.5. Impact

This is primarily a packaging and administrative tool set issue and will have no impact on the DFS.

5.3.1.6. Requirements and Recommendations

1. The installation of the DFS required services must be easy enough to be done by a typical clerical staff.

2. Drive the site specific policies from the server space rather than from the client space, e.g., automounting replication, etc.

3. Make administration possible from any single point.

4. Provide a single global consistent file space by default for the DFS and provide tools to administrate the DFS's global file space from any single point.

5. Need the ability and facilities to manage the authentication and authoring in large environments from any single point.

6. Provide the ability to deal with authentication and authorization from non-UNIX Systems.

7. Provide a standard interface to the metrics to tune the DFS systems.

8. Improve the metrics available to tune the DFS systems.

9. Increase the number of system resources that NFS can dynamically modify in response to changes in its environment or usage. Allow administrators the capability to set range limits of these resources, e.g., let NFS figure out how many deamons it needs.

10. SVR4.0 must come packaged for a distributed environment with the DFS and naming enabled. System vendors who wish may disable the DFS and naming on the specific binary products they supply.

11. Permit servers to perform a forced unmount.

12. The DFS administration must be integrated with the overall System Management framework as defined by the SMWG.

13. Must be user-friendly.

14. Must provide a Graphical interface for use of the tools.

15. Provide a consistency interface integrated with the other system management tools that exist with the administrative framework.

5.3.2. Administration of large scale systems sharing a coordinated naming environment

5.3.2.1. Priority Critical

5.3.2.2. Motivation

In order to reduce the administrative effort in a distributed environment, the common name space shared by a variety of systems should be able to be administered from a single location and not require active administrative actions on the part of the individual client systems.

5.3.2.3. Description

The management support within the DFS must provide for effective administration of very large scale systems sharing a coordinated naming environment. For example, it must be very easy to create a single rooted name space across a large number of DFS systems without manually Logging onto and locally configuring individual clients. That is, global name space changes must not require local impacts.

5.3.2.4. Current Practice

Limited support is provided in the DFS for a shared name space with the automounter, but this only provides shared access to file systems. Global naming issues are addressed in the client's space, and thus the names used to reference the shared files are determined by the individual clients and not part of a common name space. AFS supports a single shared name space that allows users to distinguish private and shareable files. Management tasks can be performed remotely without any apparent interruption of service to client systems.

5.3.2.5. Impact

May require NFS, remote mount daemon, or other protocol changes.

5.3.2.6. Requirements and Recommendations

1. Clients should be able to access a shared name space without requiring extensive modifications to their configuration.

2. Global naming modifications should not require local configuration changes.

3. Administrative procedures must be executable from a single location and not require modification of the configuration of any client system.

4. Users must still have the ability to perform arbitrary mounts (i.e., backward compatibility).

5.3.3. Administration of Replicated Files

5.3.3.1. Priority Desirable

5.3.3.2. Motivation

The motivation for the administration of replicated files is to make file replication usable and available in as many environments as possible.

5.3.3.3. Description

Replicated files require a special set of administrative tools. These tools must be easy to use.

5.3.3.4. Current Practice

Currently administration of replicated files is done manually.

5.3.3.5. Impact

There will be no impact on the DFS.

5.3.3.6. Requirements and Recommendations

1. There must be an out-of-band mechanism to replace a file or file system with an updated version in a timely manner.

2. Metrics must be provided via the PMWG's defined interface so that it is possible to tune the number of clients assigned to a replica, reduce the access time, reduce the network traffic and reduce the server workload to desired levels.

3. Administration must be well integrated with administration of the automounter as well as other DFS services.

4. Administration should provide an easy-to-use replication tool utilizing a graphical interface.

5. Administrative tools must exist for backup.

6. Administrative tools must exist to manage replicas, updates, and installation.

7. Administration tools should exist to set which metrics to monitor and how to format these metrics into advisories and reports which can be printed for the administrator or piped to some programmatic management tool.

8. Administrative tools should exist to establish failover rules.

9. Addition and deletion of replicas should be easily performed.

10. All administrative actions must be available in a single logical group. The intent is to avoid an administrator from having to perform actions on several disjointed screens, files, etc to accomplish a simple administrative action. This could be done by providing a simple graphical representation and manipulation tools for the replicated files.