CITI Technical Report 92–4

Intermediate File Servers in a Distributed File System Environment

James Howe
James.Howe@umich.edu

ABSTRACT

A component of the Institutional File System (IFS), the intermediate file server addresses scaling and interoperability issues on the University of Michigan campus. The IFS is based on AFS, a distributed file system from Transarc. Intermediate servers provide protocol translation that enables non-AFS clients to easily access the distributed file system. For example, Macintosh users can manipulate UNIX files that reside on a server by moving folders and icons, just as they would manipulate local Macintosh files. Intermediate servers also offer other benefits, such as multi-level caching and data preloading, that increase the capacity of the network by reducing the load on central servers.

June 30, 1992

Center for Information Technology Integration
University of Michigan
519 West William Street
Ann Arbor, MI 48103-4943

Intermediate File Servers in a Distributed File System Environment

James W. Howe

June 30, 1992

The University of Michigan is a large educational and research institution with over 35,000 students and approximately 24,000 faculty and staff on the Ann Arbor campus. As with any large organization, the ability to share and convey information quickly is crucial to its success. Until recently in large organizations, data was often centralized on mainframe computers.

However, that computing environment has changed. Information is now often dispersed among mainframes, workstations, and network file servers, increasing the computing power available to end users, but making sharing and maintaining information difficult.

As a result of the shift away from centralized computing to a distributed environment, a new requirement has emerged at U-M for simple, secure data sharing across campus. To address this requirement, CITI staff members working on the Institutional File System (IFS) project [1] have developed and are refining a mainframe-based, heterogeneous, campus-wide file system. This paper first presents some background information on the U-M computing environment and the underlying file system of the IFS and then focuses on the intermediate file server, a component of the IFS.

The U-M Environment

The University of Michigan has a large investment in mainframes, workstations, and personal computers. The University currently has approximately 18,000 personal computers and workstations, three IBM mainframe data centers, and widespread networking capabilities.

The data centers use two IBM ES/9000-720s, and an ES/9000-580 computer. Workstations available on campus include Apple Macintoshes, IBM PS/2s and PCs, PC clones, Apollos, Suns, Digital DECstations, and IBM RS/6000s among others.

The workstations on campus have traditionally been connected to various file servers running Suns Network File System (NFS), Apples AppleTalk Filing Protocol (AFP), Novells Netware, and Banyans VINES. The workstations themselves run a variety of operating systems including DOS, MacOS, OS/2 and UNIX. The IFS enables users across campus and across platforms to easily access data and applications in this diverse environment.

The File System Base of IFS

Because of the nature of the University campus, the IFS needed to address problems related to scale, such as degraded server performance and complicated administration and operation. Developing a file system from scratch was not feasible, so the IFS team based their work on previous efforts. The early designers of the IFS chose Transarc's AFS [2] for the IFS file system base. (AFS was first developed at Carnegie Mellon University as the Andrew File System.)

Although NFS was considered as the basis for the IFS, it was not considered viable for several reasons. The AFS protocol is better suited to long-haul networks, NFS validity checks degrade performance, and the AFS local disk caching model provides high performance. By pushing CPU-intensive tasks to workstations, AFS allows higher client-to-server ratios than NFS.

Transarc's AFS

AFS provides the amenities of a shared file system to workstation-based users. AFS offers a homogeneous, location-independent file name space to all clients, which are generally UNIX-based.

AFS uses the client/server model to distribute the workload between the server and the client, and organizes the file system into volumes, which are units of management that ease the operations effort for support staff. In AFS, client workstations play an active role in the operation of the file system. Each client manages a local disk cache that fills as files are requested from central file servers. As long as a file is not changed by another user, AFS satisfies additional client requests for the same file locally, with no network access required.

Studies of local UNIX file systems show that the ratio of reads-to-writes is approximately 4 reads to 1 write. Caching files on the local disk substantially reduces network load and improves file access times. After a client cache is filled, over 80% of the file requests made by the user can be satisfied by retrieving the file from the cache [3].

Data consistency is maintained through a callback mechanism. When the client receives a file from the server it receives a promise, in the form of a callback, that the server will notify the AFS client if the file is modified by another client. The use of callbacks obviates validity checks and their associated performance problems [4].

Whenever a file is modified and subsequently closed on a client, the updates are sent to the central server, ensuring that the central server has the most recent version of the file. The callback mechanism ensures that files contained in client caches are consistent with the files maintained by the server.

AFS provides a secure environment through the use of user authentication and file protection. User authentication in AFS is based on Kerberos [5] from the Massachusetts Institute of Technology (MIT). Kerberos uses the concept of mutual authentication using a shared secret [6]. Users can be sure that they are talking to valid AFS servers, and the servers can be sure that they are giving files to valid users.

The file protection mechanism in AFS, Access Control Lists (ACLs), are assigned to directories rather than files. Available ACLs include Read, Write, Lookup, Insert, Delete, Lock, and Administer. Permissions may be granted to groups and individuals.

Currently, the IFS uses AFS 3 for its distributed file system base. All references to AFS in this document refer to AFS 3. The next step is to move to DFS, the distributed file system component of the Distributed Computing Environment (DCE) offered by the Open System Foundation (OSF). DFS is described briefly in the "Futures" section of this paper.

Intermediate Servers

In a standard AFS environment, clients communicate directly with one or more central file servers. As the number of clients accessing a particular server grows, performance of the server declines. To address the scaling issues that occur in very large networks, the

IFS uses an innovation developed at U-M, intermediate file servers.

Instead of file servers directly servicing many clients, servers service a number of intermediate servers, which in turn service many clients (Figure 1). To the server, an intermediate file server looks like an ordinary client machine. To the client, the intermediate server looks like a central server.

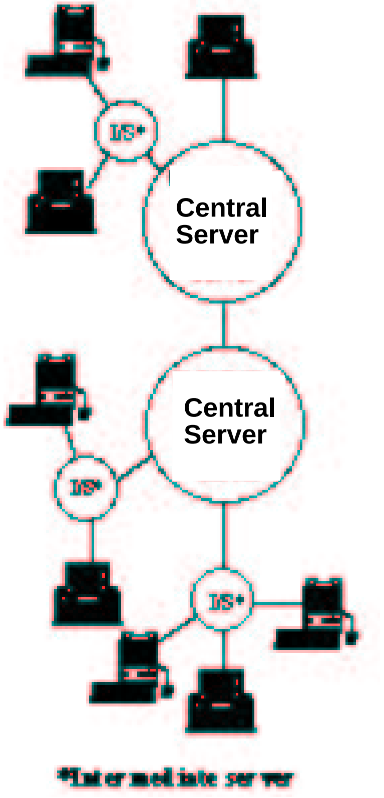

Figure 1. IFS Intermediate Configuration

The intermediate server provides several benefits over the standard AFS environment, including:

- Multi-level caching
- Data preloading
- Protocol translation

Multi-Level Caching

One of the benefits of using an intermediate server is a mid-level file cache. Intermediate servers are typically machines with large disk capacities (1 gigabyte minimum at U-M) that are well suited to the task of caching files. As the client requests files, those files are stored in the intermediate cache, as well as the client workstation cache (if a native AFS client).

The workstation cache will likely be much smaller than the intermediate cache. For some clients, particularly those accessing AFS through NFS and AFP, the cache on the workstation will be minimal to non-existent. As a result, the intermediate may have files in its cache that were once in the workstation cache, if one existed, but are no longer stored there because of space restrictions. The intermediate will also have files in its cache that have been accessed by the other workstations serviced by the intermediate.

As the number of clients increases, more intermediates can be added to the network. Adding this level of indirection dramatically increases the number of clients the central file server can service efficiently. CITI staff are currently analyzing the performance characteristics of a distributed file system using intermediate servers and comparing that performance with an identical system that does not utilize an intermediate server [7].

Data Preloading

As mentioned earlier, an intermediate server provides a large cache that is shared by the clients serviced by the intermediate. Under normal conditions, this cache is populated with files requested by the clients over a period of time. A single client could request files that will be used by members of a work group.

At UM, the Population Studies Center routinely extracts and works with specific subsets of census data. These extracted files often include gigabytes of data, which can be installed in the cache during off hours when there is less network traffic and the servers are less burdened. Once in the cache, workers can reference this data without further burdening the main file server or the backbone network.

Protocol Translation

Another benefit of using an intermediate file server is the ability of the intermediate to provide translation between AFS and other file system protocols via a protocol translator. Protocol translators convert file server requests from one file server type to another. This conversion allows other systems which normally wouldn't be able to access AFS files to be included in the distributed file system.

For the IFS, this means translating file server requests from Sun Microsystems Network File System (NFS), AppleTalk Filing Protocol (AFP), and others to AFS file requests. For example, a Macintosh can talk to an intermediate server using AFP requests. The intermediate translates these requests into appropriate AFS requests, receives and caches the file returned by the server, and returns the file to the requesting client using the AFP (Figure 2).

Protocol translators enable non-UNIX clients to access files that reside in the large scale, distributed file system. Because the campus includes several thousand machines that do not run UNIX but still have a need to share files with other users, this ability is critical.

Another advantage comes from reduced file maintenance requirements. Because the users' data and application files are stored on the central server, system administrators need only be concerned with a minimal set of system files located on each workstation. System administrators can perform backups of user data, for example, without having to go to each workstation. Because the files reside on the central server, they can be backed up easily from the central server.

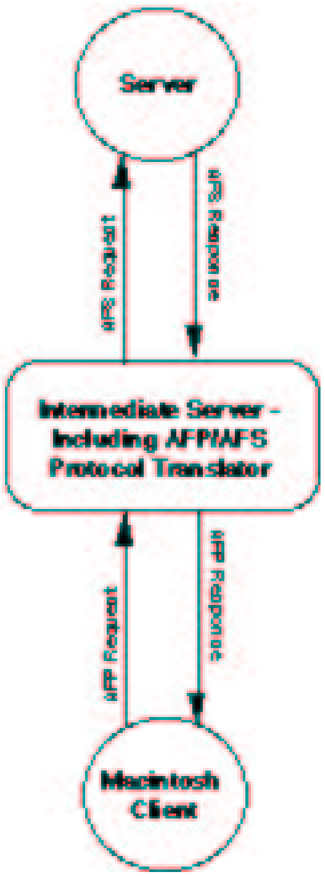

Figure 2. AFP/AFS Protocol Translator

It should be noted that an intermediate file server is not required to provide protocol translators. Protocol translators could be run on the central server. However, running translators on intermediates provides advantages over running the translators on central servers.

One advantage is that the server is not burdened with protocol conversions. In an environment where a central server may be expected to interact with NFS, AFP, and other clients, individual intermediate servers are assigned the task of talking only one protocol. For example, one or more intermediates may be assigned the task of providing services to all AFP clients. Other intermediates may provide services to NFS clients. Users of AFP are not impacted by NFS protocol conversion performance and vice versa.

Another advantage of running the translator on an intermediate is that the intermediate server can provide a cache to machines that may not have the ability to cache files. A Macintosh, for example, cannot run the AFS Cache Manager. As a result, every request for a file must go to the server. If the server is an intermediate, the file can be retrieved from the cache instead of going all the way to the central server. Because the client and intermediate are likely to be physically closer than the client and server, the time to retrieve a file across the network will normally be faster than if the client had to retrieve the file directly from the central server. Additionally, the central server will not be bothered by continual requests from noncaching machines, reducing the load on the central file server.

Finally, the responsibility for writing file changes back to the central server resides with the protocol translator. The client need not be concerned with the central server. Because the intermediate maintains the file cache, writebacks can occur in the background without affecting client resources.

Building Translators

Many issues must be resolved and tradeoffs examined when implementing a protocol translator for an intermediate server. Some of these issues include:

- User authentication
- Permission mapping (ACLs)
- File naming
- Semantic content of files
- Navigation
- Availability of AFS commands
- Migration

Authentication

One of the most important features of a large scale file system is its privacy. User data must be kept secure at all times. In the standard AFS implementation, Kerberos handles authentication issues. The issue of authentication becomes problematic, however, for clients that can't or don't support Kerberos.

Permission Mapping

Another issue related to security is file permission mapping. AFS uses access control lists (ACLs) attached to directories, and allows a user to specify the following permissions: Read, Write, Lookup, Insert, Delete, Lock, and Administer. The permissions Read, Write, and Lock apply to files in the directory. Lookup, Insert, Delete, and Administer apply to the directory itself.

In most cases, the file server protocol being exported, for example AFP, does not have permissions that map to ACLs on a one-to-one basis. When you write a protocol translator, you must decide how to handle ACL mappings.

File Naming

A third issue concerns the naming of files. Certain characteristics of filenames may differ between the client and AFS. Some of the problems encountered in file names include maximum length, allowable characters, and directory separators. When differences occur between what the client allows for a file name and what the server allows, some adaptations must be made to correct for the differences.

Semantic Content of Files

A fourth issue concerns the semantic content of a file. DOS text files, for example, use a carriage-return/line-feed combination to end each line, whereas UNIX does not. This difference makes editing the same file on the two different systems problematic. Also, AFS stores files as a byte stream with no support for record orientation. This can cause problems if the client operating system assumes some form of record-oriented support. For example, some applications may assume that the ability to lock records exists. How is this handled if the file system doesn't support records?

Navigation

A fifth issue concerns file system navigation. How does the user access various files? The navigation methods should match those of the native client interface. A Macintosh user should be able to manipulate folders, and an NFS user should be able to use file and path names.

Availability of AFS Commands

A sixth issue concerns the availability of AFS commands to the end user. AFS provides several commands that allow a user to view various characteristics of the file system. For example, commands exist that indicate the status of the available file servers. Another command returns file space usage information.

It may be possible to integrate some of this information into the client environment without the need for special commands (or even the user's knowledge), but most commands have specific meaning only for AFS files and have no meaning in other cases. The question becomes how many of the AFS-specific commands to make available, and in what manner.

Migration

Finally, there is the issue of migration. How do users who are currently not using the distributed file system migrate to the new file system? How are their databases and individual files moved to the new system without disrupting their work?

Tradeoffs

When confronted with the issues described above it is necessary to examine the tradeoffs to achieve a high degree of compatibility between AFS and the client protocol. The biggest area for tradeoffs concerns modifications of the client. Modifications fall into one of three types:

- No modifications to the client are necessary.
- Client requires modifications that aren't visible to the user.
- Client requires user-visible modifications.

Client modifications can range from adding and requiring the use of new programs to actually changing the operating system itself. The goal is to create a seamless appearance for the user.

The following sections describe three protocol translators that are currently in use. These translators support AFS, AFP, and NFS.

AFS/AFS

At first glance, an AFS protocol translator may seem redundant. However, an AFS to AFS translator provides an intermediate level that may be useful in improving the throughput of the file system. The AFS/AFS translator simply takes AFS requests from a client and either processes them itself from its own cache, or passes the requests on to the central file server for processing. From the user's point of view, he/she is talking to an AFS server; from the server's point of view, it is talking to another client.

Authentication is handled identically to the standard AFS environment. All file naming and navigation commands are identical and all AFS commands are available.

AFP/AFS

The AFP to AFS protocol translator is currently implemented in the kernel of an intermediate server. The protocol translator looks like an AFP server to a Macintosh client. No changes were made to the AFP protocol or to existing software running on the client. However, some additional software was installed on the client.

Accessing an AppleShare volume stored in AFS is no different than accessing a normal AppleShare volume. The user opens the Chooser dialog box and selects the file server he/she wishes to access. After the user selects a file server, he/she is prompted for authentication information. After authentication, the user is presented with a list of volumes available from the file server. Selecting one or more volumes results in those volumes appearing on the desktop. Users manipulate these volumes as they would any other AppleShare volume.

An AppleShare volume[1] exported by an AFS server is simply a portion of the AFS tree. Usually a user has a choice between mounting the entire AFS tree and mounting just the user's home directory. A user can create a configuration file in his/her AFS home directory, called AppleVolumes, which will let the user define the volumes from which he/she wishes to select. The configuration file maps from a volume name to the corresponding portion of the AFS file tree.

Authentication

Authentication occurs through a new Macintosh program module that works in conjunction with the Chooser dialog. The new module implements the Kerberos authentication mechanism. To make use of this module, the user simply stores a copy of the program in the folder called the AppleShare folder, found in the system folder. After that, whenever the user selects an AFS file server from the Chooser, he/she is given the option of using either the standard AFP authentication method or the Kerberos authentication method.

To connect to the AFS server, the user must select Kerberos authentication. From that point on, authentication looks identical to the user. A dialog asks for the user's ID and password. Assuming the ID and password are acceptable to Kerberos, the user is presented with a list of available AFS volumes.

The toughest part of authentication is providing a mechanism in which the intermediate is allowed to operate as if it were the user. Because the intermediate makes file requests of the central server, the intermediate must possess the credentials necessary to access files desired by the user. In the Macintosh environment, the requests are approved through a moderately complicated series of conversations between the client, intermediate, and authentication servers.

The intermediate runs an additional service that mediates a Kerberos authentication conversation. (Kerberos avoids passing passwords over campus data networks.) The conversation is managed so the intermediate obtains usable credentials, while still providing the same level of security as if the intermediate were not involved [8].

Permission Mapping (ACLs)

Macintosh users of AFS files cannot modify ACL settings using standard Macintosh software. Users of AFS receive a Desk Accessory (DA) that enables the user to manipulate permissions for folders (directories) contained on an AFS volume.

File Naming

Most allowable characters in Mac OS are supported by AFS and vice-versa. The exceptions are ":" and "?". File name length presents a problem. The Macintosh Finder imposes a maximum name length of 31 characters. File names in AFS that are longer than 31 characters are displayed on the Macintosh with characters truncated starting at the 32nd character. Most Finder operations and applications are unable to process these files. Long file names that map to the same truncated name will display with the same name, but with different icons. The Macintosh file system provides for a file tree through the use of folders. In general, users have no need to think of folder separator characters when manipulating folders via the Finder.

[1] An AppleShare volume represents a collection of files and folders. In AFS, the internal management of data is done in units called volumes.

Occasionally, however, users do want to refer to a complete path name when accessing a file. On the Macintosh, the character used to delimit folder names is a " :". The AFS/AFP translator simply translates this character into the AFS equivalent (" /"). Unfortunately, AFS files that contain a " :"in the filename itself cannot be processed by most Finder operations or application programs.

Navigation

After an AFS volume is mounted, an icon representing that volume appears on the user's desktop. Double-clicking on the volume causes the volume to open. Inside the volume, files appear exactly as the user would expect. The user manipulates the directory hierarchy by opening and closing folders. Applications are started by double-clicking on the application, or a data file that is associated with the application. In other words, the user manipulates AFS files in exactly the same manner as regular Macintosh files.

Availability of AFS Commands

In the Macintosh environment, only authentication and ACL manipulation commands are currently available. The user does not use the AFS command itself, but a Macintosh program that fits the Macintosh paradigm and presents an equivalent capability. Instead of issuing a UNIX style command, the user makes choices and fills in values on a dialog box.

NFS/AFS

The NFS/AFS translator runs in the kernel of the intermediate file server. The IFS project developed an NFS/AFS translator because no other translator was available. An NFS translator was subsequently developed by Transarc with minor performance and authentication differences [9]. The end-client requires no changes; the intermediate appears to the user as a standard NFS server. Additional programs are installed in the intermediate kernel, which manage authentication and file access control. Access to the server is via the NFS command, mount.

Note: We are referring to a UNIX-based NFS environment.

Authentication

Authentication in the NFS environment requires the addition of a program called ilog. Users who wish to access AFS files via NFS, must first authenticate using the ilog command. The command functions in a manner similar to the Kerberos klog command. The intermediate runs an additional service that takes end-client identifiers and performs authentication mapping. Performing authentication in this manner creates a shared secret between the client and the NFS translator. This mechanism allows the translator to act on behalf of the client without the need for the translator to have a copy of the user's password.

Permission Mapping (ACLs)

Permissions are handled via the AFS fs command, just as they would be in a standard AFS environment.

File Naming

File names are limited to the same character set used by AFS. File name length and the directory separator character are also the same. If the user mounts the root of the AFS tree (/afs) from the root (/afs), file name space semantics are identical. If a file resides in /afs/umich.edu/j/w/jwh/foo, an NFS user can access the file using the same file name.

Navigation

Navigation appears identical to that of a standard AFS environment. Users see file and path names.

Availability of AFS Commands

Most AFS commands are not directly supported on the client in the NFS environment.

Experiences

CITI staff are currently evaluating the performance of intermediate AFS servers. The exact performance improvement gained by running an intermediate server when the client is running standard AFS remains to be determined.

The NFS environment is usable; however because most UNIX workstations are able to run standard AFS clients, very few workstations are using the NFS translator.

The Macintosh AFP translator is used extensively. Some performance issues exist, particularly the time it takes to open folders, but overall the implementation is quite satisfactory. We have also encountered a few problems with some applications that use byte range locking. CITI staff are working to both improve the performance and solve the locking problem.

Futures

The major task for the future is integrating PC networking environments such as Banyan VINES, Novell Netware, and Microsoft LAN Manager into the distributed file system. The goal with these implementations, as with the others outlined above, is a seamless appearance to the user. The user should barely know that anything has changed. Integrating PC networks to provide this seamless capability may present more of a challenge than has been experienced to date.

Another major task for the future is the migration from the AFS platform to DFS, the distributed file system component of OSF/DCE. Transarc is developing DFS, and it is essentially the next generation of AFS.

Because DFS is derived from AFS, it shares many of AFS' characteristics. One difference concerns ACLs. In AFS 3.x, ACLs are associated with directories only. In DFS, ACLs are also associated with files. Another difference is the ability of DFS files to have property lists. The availability of property lists may make supporting foreign file systems easier. How to best make use of these new features will be an area of investigation.

In addition to the future tasks outlined above, other potential tasks include:

- OSF Distributed Management Environment/ SNMP management issues.
- Caching algorithm modifications for different working environments.
- Performance evaluation and enhancements.
- Improving fault tolerance.
- Using an intermediate as a network router, to allow the placement of intermediates at remote sites without the purchase of an additional box.

The result of this work should be a distributed file system that meets the performance, security, and functional requirements needed to support the computing environment at the University of Michigan, with applicability in the commercial world as well.

References

1. T. Hanss, "University of Michigan Institutional File System," */AIXTRA: The AIX Technical Review*, pp. 25-32 (January 1992).
2. J.H. Howard "*An Overview of the Andrew File System*," pp. 23-26 in *Winter 1988 USENIX Conference Proceedings*. Dallas, TX. (February 1988).
3. M. Satyanarayanan, et. al. "The ITC Distributed File System: Principles and Design," pp 35-50 in *Proceedings of the Tenth ACM Symposium on Operating Systems Principles* (December 1985).
4. J.H. Howard, et. al. "Scale and Performance in a Distributed File System" in *ACM Transactions on Computer Systems Vol. 6 No. 1*, (February 1988).
5. J. Steiner et. al. "Kerberos: An Authentication Service for Open Network System" in *Winter 1988 USENIX Conference Proceedings*. Dallas, TX. (February 1988).

6. R.M. Needham and M.D. Schroeder, “Using Encryption for Authentication in Large Networks of Computers” in *Communications of the ACM, Vol. 21, No. 12* (December 1978).

7. D. Muntz and P. Honeyman, “Multi-level Caching in Distributed File Systems,” CITI Technical Report 91-3 (August 1991).

8. J. Rees and B. Doster, “Third-Party Authentication in the Institutional File System,” CITI Technical Report 92-1 (February 1992).

9. L. Huston, “Comparison of the Transarc NFS/AFS Translator and the IFS NFS/AFS Translator,” in preparation.