Network Working Group
Internet Draft
Expires: January 27, 1995

Keith Moore
Reed Wade
Stan Green
University of Tennessee
July 27, 1994

# An Architecture for Bulk File Distribution

draft-moore-bfd-arch-01.txt

## Status of this Memo



## Abstract

This memo describes a system for the automated replication of data files and their descriptions to various file servers across the Internet. The system maintains a distributed database which contains the locations of each file distributed by the system, and will provide a list of locations for any file upon request. The system provides assurances of integrity, and authenticity, of the replicated files. It is intended for use with the World Wide Web, Gopher, and similar applications, to provide higher availability, improved response, and better use of network resources.

## 1. Introduction

There are a number of problems associated with the current Internet information infrastructure, which result in poor service to its users. These problems include:

+ lack of scability. Many files are available at only a single file server. Any popular file (e.g. Mosaic home page, weather map) will cause a file server to be swamped.

+ lack of fault tolerance. If a file server is unavailable, there is no mechanism to find alternate servers for that file.

+ inefficient use of network resources. The primary location of a file may be halfway across the globe, or on the other side of a low-bandwidth link. Even when alternate locations exist, there is currently no mechanism to find a "nearby" location of a particular

file.

+ no assurances of authenticity or integrity. Files are currently replicated from one server to another using a variety of ad hoc mechanisms. Various translations may occur during this process, and errors (or even deliberate modifications) may be introduced. There is currently no mechanism for ensuring the integrity of replicated files, nor any assurance that a copy of a file which is available on a server has not been modified by someone other than the author.

## 2. Proposal

In order to address these problems, we propose the following architecture. It is intended to provide replication of files across multiple servers, scalable access to the files distributed by the system, and the assurance of integrity and (optionally) authenticity for each file. In addition it provides the ability to reliably cache such files as well as the potential to take advantage of network proximity for improved utiliziation.

Each file is given a unique name called a Location Independent File Name (LIFN), which refers to that particular sequence of octets. Once a LIFN has been assigned to a file, the binding between the LIFN and that sequence of octets may not be changed. The space of LIFNs is sub-divided among several "publishers" (or "naming authorities"), who are responsible for ensuring the uniqueness of LIFNs within their portion of LIFN-space, and also provide a LIFN-to-location mapping service for those LIFNs.

The LIFN-to-location mapping service is provided by a network of "location servers" collectively known as the "location database". These servers accept requests for locations of LIFNs, as well as updates containing new locations or requests to delete old LIFN-to-location mappings. Such update requests require authentication; only those file servers which are authorized by the publisher may store locations in the database.

Access to files themselves is provided by more-or-less conventional file servers, using any protocol which provides binary transparent file access. Such protocols would include HTTP, Gopher, FTP, and others, as long as certain restrictions are observed.

Files are replicated among file servers using a "replication daemon". A copy of the replication daemon runs on each file server. It accepts descriptions of newly published files, and decides (based on site-provided criteria) which files should be acquired by the file server. It then queries the location database to find a location for each file desired, and retrieves the file from one of the locations listed. Finally, it updates the location database to inform it of the new location for that file. The replication daemon may also act as a file

reaper, deciding when to delete files, and informing the location database when such files will no longer be available.

Associated with each file is a description. Included in the description is so-called "bibliographic information", such as title, author, content-type, etc., but also an MD5 or similar fingerprint of the file. The relevant portions of the description are cryptographically signed by the publisher. To perform an integrity check, a file server or user can retrieve the description for any file (using whois++ or a similar protocol), compute the MD5 fingerprint for that file, and compare this with the one listed in the description. To check authenticity, it can also verify the credentials of the file's description.

A file is "published" in the system by creating a description to go along with the file, signing the description with the publisher's private key, making the file available via one or more "master" file servers, and listing those locations in the location database. The description may also be sent (perhaps via ordinary email) to interested parties. Such parties may include slave file servers (which can use them to decide which new files to acquire), resource discovery servers (which can then provide search services based on the file descriptions and/or the files themselves), and ordinary users.

The location database consists of one or more servers for each publisher. These are listed in either a well-known master directory, or a reserved portion of the DNS name space, so that a client can easily find out which server to query for a particular LIFN. The query itself uses a datagram-based protocol, which is designed to impose the least possible overhead for both client and server. Updates to the location server use a similar protocol; however, these protocols also require authentication to prevent unauthorized (or untrusted) servers from listing alternate locations for a file. Location updates are posted to a single location server and propagated to the other peer servers via a batch version of the update protocol (using virtual circuits rather than datagrams).

It is not necessary to keep all location servers for a publisher in sync. The location query service does not guarantee that it returns all instances of a given file. If the list of locations provided by one server is insufficient, the client is free to consult the other servers in the hope of finding a better one. Similarly, if one or more of the locations thus provided is "stale" (that is, points to a file that no longer exists), the client may also look for the file at its other listed locations. (Note that while a file server may delete a file whose location is listed in the location database, it may not re-use a filename for a different file or change that file in any way.).

To minimize the liklihood of stale file locations, file servers are encouraged to inform the location database in advance of actually deleting a file. The response to a location query includes a "time to live" field which is used by clients or proxy servers to maintain a

cache of LIFN-to-location mappings. After being informed that a file is going to disappear, the location servers will adjust the "time to live" field in future responses to queries for that file, to reflect the time when the file is expected to disappear. Until that time, the "time to live" field is the value supplied by the file server when it posted the location. Specifying a time to live of "N" in a location update is tantamount to an agreement that the file server will not delete the file without informing the location server "N" seconds in advance of doing so.

A special file replication protocol is used between file servers. It provides mutual authentication to prevent spoofing, and pipelining to transfer large numbers of files effeciently, even over high-delay links. It may also accomodate compression on a per-file basis (for low-bandwidth links), and checkpointing to allow for recovery when transfers of large files are interrupted.

## 3. Evaluation

The system should scale in several ways. User demand for any particular file can be distributed over multiple file servers. The location database is also distributed, both because each publisher maintains its own servers, and also because several servers can be provided for any publisher. In addition, the current location query protocol provides for a cacheable "redirect" response that allows the LIFN space for a particular publisher to be divided across several secondary servers, without imposing any additional structure on the LIFNs themselves. Because there is no need for synchronization, location updates can also be distributed across several servers, and effeciently transmitted among location server peers. This avoids the overhead with multi-phase commit protocols which would be needed to ensure consistency.

Integrity and authenticity are provided by the MD5 fingerprint and the cryptographic signature in the file description. A possible weak point in the current system is the assumption that DNS will be used to identify location servers for a particular publisher, since DNS is not itself secure. It should be pointed out that since only trusted locations will be listed by the location service, a user may not wish to perform integrity or authenticity checks for every file accessed. However, the capability is there for when it is needed.

The system allows a client to consult a local cache or proxy server before attempting to access a file which may already be available locally. Since the binding between a LIFN and the file is fixed, if a client has a LIFN for a file, and the cache has a file which goes with it, the client has a reasonable assurance that the cached copy is correct (assuming it trusts the cache). If the time-to-live field in a location response is nonzero, LIFN-to-location bindings can also be reliably cached for that amount of time.

Finally, this system allows the potential that a client can select a "nearby" location from among several locations for a file, or from among several available location servers. A means by which this may be accomplished has been proposed and is under investigation.

## 4. Open Problems

As stated above, there is a need to provide a means by which a client may choose from among several service locations to take advantage of network proximity.

If existing file servers are to adopt this plan, there needs to be a transition scheme. As stated above, locations (i.e. file names) of files provided by the location databases may not be reused, even for updates to the file. This is in contrast to the present-day use of file locations (URLs) which are expected to be stable references to the *current* version of a file. If the replication daemon is to replace the ordinary mirroring software that is presently in use, it must also provide "stable" locations for the same files, which may be updated in place and accessed by more traditional means. This could be accomplished by having each description include a "suggested-filename" field. The file server would concoct a local filename from this field; any new file from the same publisher with the same suggested-filename would replace the old copy of the file stored in that location (but not the copy of the file stored in the location listed in the location database).

There is a need to describe many types of relationships in the file description. The details of such descriptions are yet to be defined.

## 5. Implementation status

A prototype version of this system is being constructed by the authors. A distributed location database and client library have been constructed and interfaced to Mosaic; the resulting client demonstrated the ability to (crudely) select from among multiple locations of a file, and to recover from the failure of both file servers and location database servers. The replication daemon and its associated protocols are currently under development.

Experience from the use of the prototype will be used to construct a second version of the system, which the authors intend to make widely available.