Network Working Group
Request for Comments: 2055
Category: Informational

B. Callaghan
Sun Microsystems, Inc.

October 1996

WebNFS Server Specification

Status of this Memo

Abstract

This document describes the specifications for a server of WebNFS clients. WebNFS extends the semantics of versions 2 and 3 of the NFS protocols to allow clients to obtain filehandles more easily, without recourse to the portmap or MOUNT protocols. In removing the need for these protocols, WebNFS clients see benefits in faster response to requests, easy transit of firewalls and better server scalability This description is provided to facilitate compatible implementations of WebNFS servers.

Table of Contents

1. Introduction

The NFS protocol provides access to shared filesystems across networks. It is intended to be machine, operating system, network architecture, and transport independent. The protocol currently exists in two versions: version 2 [RFC1094] and version 3 [RFC1813], both built on Sun RPC [RFC1831] and its associated eXternal Data Representation (XDR) [RFC1832]. This document assumes some familiarity with the NFS protocol and underlying RPC protocols.

WebNFS servers implement semantic extensions to both versions of the NFS protocol to support a lightweight binding mechanism for conventional or web browser clients that need to communicate with NFS servers across the Internet. a WebNFS server supports the public filehandle and multi-component lookup features described herein, as well as meeting some additional requirements.

For a description of WebNFS client requirements, read RFC 2054.

2. TCP vs UDP

The NFS protocol is most well known for its use of UDP which performs acceptably on local area networks. However, on wide area networks with error prone, high-latency connections and bandwidth contention, TCP is well respected for its congestion control and superior error handling. A growing number of NFS implementations now support the NFS protocol over TCP connections.

A WebNFS client will first attempt to connect to its server with a TCP connection. If the server refuses the connection, the client will attempt to use UDP. All WebNFS servers should support client use of TCP and must support UDP.

3. Well-known Port

While Internet protocols are generally identified by registered port number assignments, RPC based protocols register a 32 bit program number and a dynamically assigned port with the portmap service which is registered on the well-known port 111. Since the NFS protocol is RPC-based, NFS servers register their port assignment with the portmap service.

NFS servers are constrained by a requirement to re-register at the same port after a server crash and recovery so that clients can recover simply by retransmitting an RPC request until a response is received. This is simpler than the alternative of having the client repeatedly check with the portmap service for a new port assignment. NFS servers typically achieve this port invariance by registering a

constant port assignment, 2049, for both UDP and TCP.

To avoid the overhead of contacting the server's portmap service, and to facilitate transit through packet filtering firewalls, WebNFS clients optimistically assume that WebNFS servers register on port 2049. Most NFS servers use this port assignment already, so this client optimism is well justified.

A WebNFS server must register on UDP port 2049 and TCP port 2049 if TCP is supported.

4. Server Port Monitoring

Some NFS servers accept requests only from reserved UDP or TCP ports, i.e. port numbers below 1024. These "privileged" ports are available only to Unix processes with superuser permissions. Requests that do not originate from the range of reserved ports are rejected. This an optimistic way of preventing direct access to the server from user processes that may attempt to spoof AUTH_UNIX RPC credentials.

Since WebNFS clients are not required to use reserved ports, a WebNFS server must not check the originating port for requests to filesystems made available to WebNFS clients.

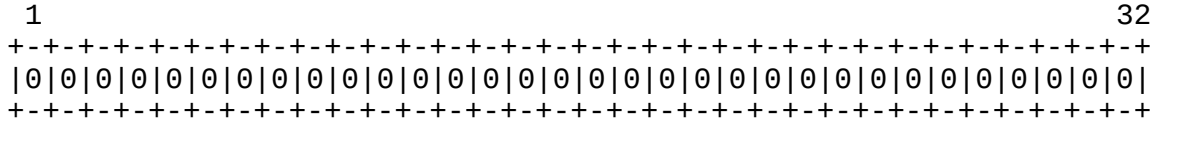
5. Public Filehandle

The public filehandle is an NFS file handle with a reserved value and special semantics that allow an initial filehandle to be obtained. A WebNFS client can use the public filehandle as an initial filehandle without using the MOUNT protocol. Since NFS version 2 and version 3 have different filehandle formats, the public filehandle is defined differently for each.

The public filehandle is a zero filehandle. For NFS version 2 this is a filehandle with 32 zero octets. A version 3 public filehandle has zero length.

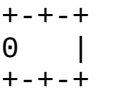
5.1 Version 2 Public Filehandle

A version 2 filehandle is defined in RFC1094 as an opaque value occupying 32 octets. A version 2 public filehandle has a zero in each octet, i.e. all zeros.



5.2 Version 3 Public Filehandle

A version 3 filehandle is defined in RFC1813 as a variable length opaque value occupying up to 64 octets. The length of the filehandle is indicated by an integer value contained in a 4 octet value which describes the number of valid octets that follow. A version 3 public filehandle has a length of zero.



6. Multi-component Lookup

Normally the NFS LOOKUP request (versions 2 or 3) takes a directory file handle along with the name of a directory member, and returns the filehandle of the directory member. If a client needs to evaluate a pathname that contains a sequence of components, then beginning with the directory file handle of the first component it must issue a series of LOOKUP requests one component at a time. For instance, evaluation of the Unix path "a/b/c" will generate separate LOOKUP requests for each component of the pathname "a", "b", and "c".

A LOOKUP request that uses the public file handle can provide a pathname containing multiple components. The server is expected to evaluate the entire pathname and return a filehandle for the final component. The pathname syntax is assumed to be understood by the server, but the client must not make assumptions of the pathname syntax.

A Unix server, for instance, uses a slash "/" character to separate components in a Unix pathname.

For example, rather than evaluate the path "a/b/c" as:

```
     LOOKUP  FH=0x0  "a"  --->
                          <---  FH=0x1
     LOOKUP  FH=0x1  "b"  --->
                          <---  FH=0x2
     LOOKUP  FH=0x2  "c"  --->
                          <---  FH=0x3
```

Relative to the public filehandle these three LOOKUP requests can be replaced by a single multi-component lookup:

```
     LOOKUP  FH=0x0  "a/b/c"  --->
                              <---  FH=0x3
```

Multi-component lookup is supported only for LOOKUP requests relative to the public filehandle.

6.1 Canonical Path vs. Native Path

If the pathname in a multi-component LOOKUP request begins with a printable ASCII character, then it must be a canonical path. A canonical path is a hierarchically-related, slash-separated sequence of components, <directory>/<directory>/.../<name>.

Occurrences of the "/" character within a component will be escaped using the escape code %2f. Non-printable ascii characters (with values in the range 00-1F and 7f hexadecimal) will also be escaped using the "%" character to introduce a two digit hexadecimal code. Occurrences of the "%" character that do not introduce an encoded character will themselves be encoded with %25.

If the first character of a canonical path is a slash, then the canonical path must be evaluated relative to the server's root directory. If the first character is not a slash, then the path must be evaluated relative to the directory with which the public filehandle is associated.

Not all WebNFS servers can support arbitrary use of absolute paths. Clearly, the server cannot return a filehandle if the path identifies a file or directory that is not exported by the server. In addition, some servers will not return a filehandle if the path names a file or directory in an exported filesystem different from the one that is associated with the public filehandle.

If the first character of the path is 0x80 (non-ascii) then the following character is the first in a native path. A native path conforms to the natural pathname syntax of the server. For example:

Lookup for Canonical Path:

```
LOOKUP FH=0x0 "/a/b/c"
```

Lookup for Native Path:

```
LOOKUP FH=0x0  0x80 "a:b:c"
```

Other introductory characters in the range 0x81 - 0xff may be added in future specifications. If the server receives any character in this range that it does not understand then it must return an error to the client: NFSERR_IO for NFS V2, NFS3ERR_IO for NFS V3.

6.2 Symbolic Links

Servers that support symbolic links may encounter pathname components that are symbolic links. The server is expected to evaluate these symbolic links as a part of the normal pathname evaluation process. This is a different semantic from that of conventional component-at-a-time pathname evaluation by NFS clients, where the client is expected to do the evaluation.

However, if the final component is a symbolic link, the server must return its filehandle and let the client evaluate it.

6.3 Export Spanning Pathnames

The server may evaluate a pathname, either through a multi-component LOOKUP or as a symbolic link embedded in a pathname, that references a file or directory outside of the exported hierarchy.

Clearly, if the destination of the path is not in an exported filesystem, then the server must return an error to the client.

Many NFS server implementations rely on the MOUNT protocol for checking access to exported filesystems and NFS server does no access checking. The NFS server assumes that the filehandle does double duty: identifying a file as well as being a security token. Since WebNFS clients do not normally use the MOUNT protocol, a server that relies on MOUNT checking cannot automatically grant access to another exported filesystem at the destination of a spanning path. These servers must return an error.

For example: the server exports two filesystems. One is associated with the public filehandle.

```
/export/this   (public filehandle)

/export/that
```

The server receives a LOOKUP request with the public filehandle that identifies a file or directory in the other exported filesystem:

```
   LOOKUP 0x0  "../that/file"
or
   LOOKUP 0x0  "/export/that/file"
```

Even though the pathname destination is in an exported filesystem, the server cannot return a filehandle without an assurance that the client’s use of this filehandle will be authorized.

Servers that check client access to an export on every NFS request have more flexibility. These servers can return filehandles for paths that span exports since the client's use of the filehandle for the destination filesystem will be checked by the NFS server.

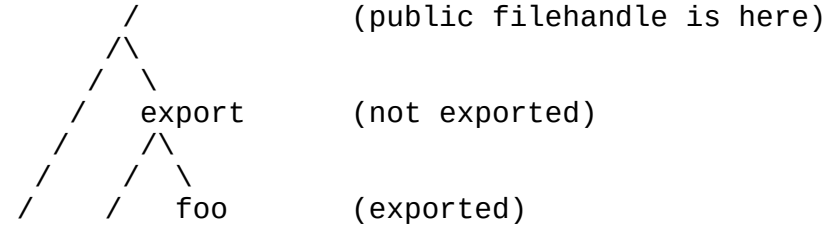
7. Location of Public Filehandle

A server administrator can associate the public filehandle with any file or directory. For instance, a WebNFS server administrator could attach the public filehandle to the root of an anonymous FTP archive, so that anonymous FTP pathnames could be used to identify files in the FTP hierarchy, e.g.

```
# share -o ro,public  /export/ftp
```

On servers that support spanning paths, the public filehandle need not necessarily be attached to an exported directory, though a successful LOOKUP relative to the public filehandle must identify a file or directory that is exported.

For instance, if an NFS server exports a directory "/export/foo" and the public filehandle is attached to the server's root directory, then a LOOKUP of "export/foo" relative to the public filehandle will return a valid file handle but a LOOKUP of "export" will return an access error since the server's "/export" directory is not exported.




```
LOOKUP 0x0  "export"      (returns an error)

LOOKUP 0x0  "export/foo"  (returns an filehandle)
```

8. Index Files

Most HTTP servers support the concept of an index file. If a browser references a directory that contains an index file, then the server will return the contents of the index file rather than a directory listing. For instance if a browser requests "a/b/c" then the server might return the contents of "a/b/c/index.html".

A WebNFS server may choose to emulate this feature for the benefit of clients using the NFS protocol to browse a Web hierarchy. On receiving a multi-component lookup for a canonical path that names a directory, the server can check that directory for the presence of an index file. If the file exists then the server may choose to return the filehandle of the index file instead of the directory. Index files are commonly called "index.html" though the name is usually configurable.

9. Bibliography

[RFC1831] Srinivasan, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1831, August 1995. http://www.internic.net/rfc/rfc1831.txt

[RFC1832] Srinivasan, R., "XDR: External Data Representation Standard," RFC 1832, August 1995. http://www.internic.net/rfc/rfc1832.txt

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, August 1995. http://www.internic.net/rfc/rfc1833.txt

[RFC1094] Sun Microsystems, Inc., "Network Filesystem Specification", RFC 1094, March 1989. NFS version 2 protocol specification. http://www.internic.net/rfc/rfc1094.txt

[RFC1813] Sun Microsystems, Inc., "NFS Version 3 Protocol Specification", RFC 1813, June 1995. NFS version 3 protocol specification. http://www.internic.net/rfc/rfc1813.txt

[RFC2054] Callaghan, B., "WebNFS Client Specification", RFC 2054, October 1996. http://www.internic.net/rfc/rfc2054.txt

[Sandberg] Sandberg, R., D. Goldberg, S. Kleiman, D. Walsh, B. Lyon, "Design and Implementation of the Sun Network Filesystem," USENIX Conference Proceedings, USENIX Association, Berkeley, CA, Summer 1985. The basic paper describing the SunOS implementation of the NFS version 2 protocol, and discusses the goals, protocol specification and trade-offs.

[X/OpenNFS] X/Open Company, Ltd., X/Open CAE Specification: Protocols for X/Open Internetworking: XNFS, X/Open Company, Ltd., Apex Plaza, Forbury Road, Reading Berkshire, RG1 1AX, United Kingdom, 1991. This is an indispensable reference for NFS version 2 protocol and accompanying protocols, including the Lock Manager and the Portmapper.

[X/OpenPCNFS] X/Open Company, Ltd., X/Open CAE Specification: Protocols for X/Open Internetworking: (PC)NFS, Developer's Specification, X/Open Company, Ltd., Apex Plaza, Forbury Road, Reading Berkshire, RG1 1AX, United Kingdom, 1991. This is an indispensable reference for NFS version 2 protocol and accompanying protocols, including the Lock Manager and the Portmapper.

10. Security Considerations

Since the WebNFS server features are based on NFS protocol versions 2 and 3, the RPC security considerations described in RFC 1094, RFC 1813, and Appendix A of RFC 1831 apply here also.

Clients and servers may separately negotiate secure connection schemes for authentication, data integrity, and privacy.

Implementors must note carefully the implications of export spanning pathnames as described in section 6.3.

11. Acknowledgements

This specification was extensively reviewed by the NFS group at SunSoft and brainstormed by Michael Eisler.

12. Author's Address

Address comments related to this document to:

nfs@eng.sun.com

Brent Callaghan
Sun Microsystems, Inc.
2550 Garcia Avenue
Mailstop Mpk17-201
Mountain View, CA 94043-1100

Phone: 1-415-786-5067
Fax: 1-415-786-5896
EMail: brent.callaghan@eng.sun.com