Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 29, 2009

Alex RN, Ed.
Bhargo Sunil, Ed.
Dhawal Bhagwat
Dipankar Roy
Rishikesh Barooah
NetApp
June 27, 2009

NFS operation over IPv4 and IPv6
draft-alexrn-nfsv4-ipv6-00.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on December 29, 2009.

Copyright Notice

Abstract

This Internet-Draft provides the description of problem set faced by NFS and its various side band protocols when implemented over IPv6 in various deployment scenarios. Solution to the various problems are also given in the draft and are sought for approval in the respective NFS and side band protocol versions.

Foreword

This "forward" section is an unnumbered section that is not included in the table of contents. It is primarily used for the IESG to make comments about the document. It can also be used for comments about the status of the document and sometimes is used for the RFC2119 requirements language statement.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Terminology

ALG: Application Level Gateway.

NFS-ALG: NFS Application Level Gateway.

Host: Used to refer to the client or the server where the specific(s) of client or the server does not matter.

IPv4: Internet Protocol Version 4.

IPv6: Internet Protocol Version 6.

NAT: Network Address Translator.

NFS: Used to refer to Network File System irrespective of the version.

NFSv2: Network File System Protocol Version 2.

NFSv3: Network File System Protocol version 3.

NFSv4: Network File System Protocol version 4.

NFSv4.1: Network File System Protocol version 4.1.

NLM: Network Lock Manager Protocol.

NSM: Network Status Monitor Protocol.

Operation: Refers to the NFS operation when its mode of request or response is inconsequential.

2. Introduction

NFS being a application layer protocol can operate over several network layer protocols. This draft addresses problems associated with NFS operation over IPv4 and IPv6. NFS follows the client server approach and does not restrict the network layer capability of the hosts.

2.1. Various Deployment Scenarios

The various deployment scenarios are as follows:

(a) Client in IPv4 only network and Server being in IPv4 only network.
(b) Client in IPv6 only network and Server being in IPv6 only network.
(c) Client in IPv4 only network and Server being in IPv6 only network.
(d) Client in IPv6 only network and Server being in IPv4 only network.
(e) Client in IPv4 and IPv6 capable network and Server being in IPv4 and IPv6 capable network

In the above mentioned scenarios, types a) b) are called symmetric stack mode of operation. Type c) and d) can be called as asymmetric stack mode . Type a)-d) can be interchangeably be called as single stack mode. Type e) can be interchangeably be called as dual stack mode.

In the case of asymmetric stack operation mode the client and server MUST use protocol translation at the network level and the application level. This is achieved using a combination of network level and application level protocol address translations done using NAT-PT and NFS-ALG.

The problems which are discussed below are primarily related to state sharing across different protocol address families. The states are maintained through NFSv4(or later versions) and NLM/NSM.

3. Multi homing support

IPv6 supports various types of scoped addresses which can be ambiguous at the scope boundary as referenced in [RFC4007] . A link local address is one whose validity is only within a link and hence is link scoped. Global addresses on the other hand are valid across links.

In IPv6 case we can have link local address in a link scope boundary like say Host H1 having 2 interfaces connected to Clients C1,C2 on two links L1 and L2 as below.

Typical multi homed host

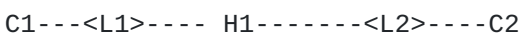

Figure 1

The link local address is ambiguous because of the private nature Link L1 and L2 could have the same address (similar to private addresses of 10.x , 192.x in IPv4. In IPv4 this is an exception but for IPv6 this is default, due to auto configuration). In the case of outbound packet transmission the link scope could be ambiguous and MUST be explicitly specified or the implementation MAY support a default scoped addresses.

NFS as a service operating in the link scope boundary MAY need to explicitly specify its scope of communication using a scope identifier [RFC4007], depending on different client scenarios supported. NFS over IPv6 using link local address MAY be supported. In particular, the NFS server or client may or may not decide to support the scope field in an IPv6 link local address.

NFS servers on scope boundaries doing an outbound communication, the scope SHOULD be derived from the inbound context when the server receives an NFSv4.0 SETCLIENTID operation or an RPCBIND GETVERSADDR procedure which contain universal addresses section 2.2 [RFC3530] . Any outbound communication initiated like NFSv4 callbacks or NSM notifies SHOULD use this information.

For scenarios like NFS servers running over IPv6 with only auto-configured (link local only)addresses, some clients could boot off the NFS servers. In such situations link local support SHOULD be supported.The NFS server MAY provide for a default scope to operate unambiguously.

4. RPCBIND

NFS servers supporting IPv6 MUST support RPCBINDv3 as defined in [RFC1833], over IPv6. Additionally, RPCBINDv4 SHOULD be supported, as noted later in this section.

RPCBINDv3/4 protocols 'use a transport-independent format for the transport address'. Using RPCBINDv3/4, a client can clearly communicate to the server which transport (IPv4/v6, TCP/UDP) it is interested in for contacting a service.The server can communicate clearly to the client, the various transports on which a service is available. RPCBINDv2 (aka PORTMAP) provides limited support in this area.

RPCBINDv4 SHOULD be supported because it introduces useful procedures like, RPCBPROC_GETVERSADDR - to query the server for the address of a specific version of an RPC service and RPCBPROC_GETADDRLIST - to query the server for a list of all addresses / transports on which an RPC service is available.

Supporting RPCBINDv3/4 over IPv4 is OPTIONAL, if supported it may ease basic troubleshooting of IPv6 servers from IPv4 hosts. It may also make it easier for applications to identify a transport of interest, that the server supports, by making RPCBPROC_GETADDRLIST calls over IPv4.

The netid and address formats in the RPCBINDv3/4 procedures, MUST be as per those defined for netid and universal addresses, in netid_ID draft [netid_ID]. The implementation SHOULD NOT use IPv4 embedded IPv6 addresses defined in Section 2.5.5 [RFC4291], for the RPCBINDv3/4 procedures.

Where specific information regarding a service is needed from a server, a client SHOULD prefer the RPCBINDv4 procedure RPCBPROC_GETVERSADDR over RPCBPROC_GETADDR. On the other hand, in scenarios where a list of details is required, a client SHOULD use the procedures RPCBPROC_DUMP and RPCBPROC_GETADDRLIST, as applicable.

An NFS implementation SHOULD NOT propagate the link local addresses encoded as universal addresses from one link scope domain to another. This could result in the following problem

a) Server has 2 different link local addresses, A1 and A2 on links L1 and L2 .

b) Client is in link L1.

c) Server publishes RPCBIND universal address of link local address A1 and A2.

d) Client decided to connect using A2.

Step d) violates the usage of a link local private address as A2 could be assigned to a valid host in the link scope of L1.

5. NLM and NSM

A dual stack NSM server implementation with persistent recording of source IP address SHOULD record at least one IPv4 and one IPv6 address, for sending out NOTIFY messages.

This will be required in the following scenario :

a) Client connects to the server using IPv6 address.

b) Client connects to the server using IPv4 address.

c) The server switches from dual stack to single stack mode of operation.

d) Server restarts.

Step c) can happen due to a network or interface disruption. Step c) can also happen after step d) which results in loss of ability of a host to contact the other host on a particular address family after restarting.

The server in this case SHOULD associate two client addresses with the client identifier "caller_name" field as referenced in section 6.1.4 [RFC1833] .

After d) if the server does not record at least one address of each address family, the server after restarting will not be able to send NOTIFY to the client on the address family that faced disruption. This will result in the client incorrectly assuming that the server is holding locks when it is not, if the client expects the NOTIFY to come on the same interface and same version of the IP protocol on which the locks were requested. The client when it receives the NOTIFY from the server holding locks, through one of the address families SHOULD try to reclaim the locks it had established on the other address family as well, if it can find out that both the addresses or address families refer to the same server.

Another example is, the client after establishing the locks on the server on both the address families, restarts. When the client comes back up and if the server can not be contacted on one of the address families, it SHOULD use the "caller_name" in the name field of the NOTIFY so that server when it receives the NOTIFY should be able to remove the locks the client had held on other address family as well.

The server SHOULD identify the client based on the "caller_name" field in the NLM request and it SHOULD remain constant irrespective of the address family that the client uses to contact the server. This would help the server in identifying which client's lock to release in the event the client sends the notify from a different address family than the one on which it had sent the NLM operations. This would also help in the case, when the mode of operation is asymmetric stack and the intermediate NAT-PT changes source address while connecting to the server.

Callback port information verification from server to client for IPv6 cases MUST be done through RPCBIND. The scenarios are asynchronous lock requests and call back port verification during granting of waiting locks.

6. Client Identification

In the case of NFS4.1 the short hand clientid is very similar to NFSv4.0 clientid. Since states are tied to clientid, state sharing across and within sessions are immune to individual connection failures. The failures from individual connections of an address family can be failed over to another address family if available.

As per current NFSv4.0 standards RFC3530 [RFC3530] server identifies the client using the short identifier called as Client Identifier. Client MUST send a different client string in SETCLIENTID to a different destination addresses(s)/family of address(s). Even if the same server is servicing on a different network address/address family the server MUST return a different clientid to the client. This is to prevent confusion on the client side as there is no way of determining whether the server to which the client is connecting again is the same or not.

The usage of different client strings for different network addresses families might result in a case that the request(s) from the same client be deemed to conflict and will result in revocation of delegation. This can happen in the given client scenario:

a) Client establishes a connection to server on address X with address family IPv4 and then opens the file Z in write mode.

b) Server grants the client will get a write delegation.

c)Client breaks the connection with server address X and/or tries to establish another connection with server address Y with address family B and then tries to open the file Z.

In step c) as client is trying to connect to a different server address/address famiily it would send the SETCLIENTID with different client string than in step a). As the clientid given in step c) will be different than the clientid granted in step a) the server will end up revoking the delegation granted in step b). This is because the clientid is calculated based on client string which would be different in step a) and step c). Step c) can happen even if the client side faced a disruption on one of its address families and then connected on a different address family to the server. Example would be client connected using IPv4 in step a) and then client IPv4 stack or interfaces faced disruption after step b). Client then uses the IPv6 to connect to server in step c). In either case after step c) the client would not be holding the delegation.

To handle this case the server SHOULD NOT use the client string alone in the generation of the short hand clientid but should combine the

client string with its own server identifier to calculate the clientid. The server identifier should be generated in a unique way on similar lines as that of the client identifier. Specifically the server identifier should be such that no two servers should use the same server identifier. An example of well generated server identifier can be the one that includes the following :

a) MAC address

b) Machine serial number

The client SHOULD always send the SETCLIENTID as the first request on the connection. Even if the client is retransmitting the request on a new connection it SHOULD send the SETCLIENTID as the first request. The client SHOULD use the same client string across different IP address families as no two different servers would end up giving the same clientid as both the components, client string and server identifier, used in the calculation are unique. If the clientid returned by server is the same as one of the existing clientid, the client SHOULD conclude that both the connections are to the same server. To prevent the server from expunging the client due to non renewal, the client should send a RENEW even if it does not have a lease after a SETCLIENTID to the server.

In the above example the client string in step c) would have been the same as step a) and therefore the server would not revoke the delegation granted in step b).

7. Dual to single stack mode transition

Dual stack implementations of NFS over IPv4 and IPv6 should ensure that the shutdown of one stack implementation leaves the host in a usable state. This is important for shared state like locks accessible through both IPv4 and IPv6 paths. A shutdown of one path would result in a permanent, partial, or temporary un-reachability to the client. Anticipating reconnects, after the partial reachability condition are resolved, the states should be left intact. Administrative support for forceful lock management (removal /cleanup etc) is recommended.

Shutting down of one stack IPv4/IPv6 or loss of all interfaces of one particular address family IPv4/IPv6 should not result in the NFSv4.0 client states to be removed after the lease period expires. This is required so that server does not grant the locks to any other client reachable from another address family. If the locks are removed the subsequent clients over IPv4, which are granted locks might find the the file to be in unusable state.

a) If the IPv6 client A is connected to the server and has locks and open share state.

b) The partial reachability condition happens for IPv6.

c) The IPv4 client B tries to access the files on which the client A had states.

If after step b) the server had removed the state then the client B might find the file to be in unusable state and so the state for client A should be maintained unless the disruption due to step b) is permanent, in which case the administrator needs to take some steps to prevent the unusable file state.

For NFSv4 one of the ways to implement the above recommendation is that the server should mark the client and the states associated with it as temporarily unusable but should not remove the state associated with the clients in such case. After the complete reachability is restored the server should go into partial restart case for only the clients that had their state marked as temporarily unusable and thus should allow such clients to regain their state. The server should identify the clientid/states that are marked as temporarily unusable and should send the NFSERR_STALE_CLIENTID/NFSERR_STALE_STATEID which will start the state recovery procedure on the client side. The server can remove the client state if the clients have not recovered the state in the grace period after the complete reachability condition has been restored.

Supposedly the partial reachability condition affected only the clients accessing the server over IPv6, after the reachability is restored then the grace period should be started for only the clients coming over IPv6. If the clients are coming over IPv4 contending for the same lock for which is being held by IPv6 client the IPv4 client should be denied.

For the case of NLM in such cases the NOTIFY should be sent to clients that were affected due to partial reachability condition and that had held the locks prior to that condition coming into effect.

8. NFSv4 Callback Information

In the case of asymmetric stack mode, if the client is in IPv6 network and tries to contact the server in IPv4 domain it should be using an IPv6 address transparently translated by an ALG. The intermediate network element (NFS-ALG) in the asymmetric stack mode should convert the embedded addresses in the NFS operations so that the other host can understand the network address.

The NFSv4 server implementation should verify the netid information in the callbacks corresponds to respective address families. The netid used for IPv6 address is tcp6 and for IPv4 addresses is tcp.

9. Reply cache tuples for NFSv4

Currently, most of the reply cache implementations use some form of combination of the elements client address, client port, server address, protocol, RPC XID as the values on which to decide on the hit in the cache. But if the environment is such that the client is working only in one address family and the server in another family (no dual stack) and the translation is handled by an intermediate element, then it might be possible that when the client retransmits the request the source pair (address/port) that the server sees might change from the previous connection values and the replay cache might be rendered ineffective. It might also be that the client has lost all of its interfaces of a particular address family and might switch to another address family. It might also be the case that the client uses a new source pair for the retransmission which will have the similar effect.

As listed in Client Identification (Section 6) the client can obviate the need to send different client strings to different server addresses. The NFSv4.0 client should always send the SETCLIENTID procedure as the first request to the server. If a request is to be retransmitted on a different connection, the first procedure sent out should be a SETCLIENTID with no change in the callback address or client string or verifier. This will help the server to associate the new connection with the clientid.

The replay cache implementations thus can switch to just using the client string and RPC XID as the basis for determining a replay cache hit, if the server is using the address/address family to determine the cache hit.

10. External Interfaces

NFS servers considering usage of information like MTU control or other lower level information from the IPv6 stack MAY consider, the updated standard sockets APIs in the form of advanced sockets implementation as specified in RFC 3542 [RFC3542] and RFC 3493 [RFC3493].

In the case of client and server operating in different address families with no dual stack support DNS-ALG and NAT-PT MUST be used.

11. Other optimizations

11.1. Address Persistence

In all cases where the IP addresses are stored in stable storage it is always preferable to store addresses with the address family information. The scenarios could be that of recording addresses for NSM notifies and client addresses persistently stored for reply cache. One way of implicit typing is with the IPv4 mapped IPv6 addresses for storage efficiency. It is RECOMMENDED that IPv6 support capability information is preserved. This information can be used during upgrades and downgrades of NFS server instances which have may or may not have turned on support for NFS over IPv6.

11.2. IP addresses as keys

There are a number of situations where decisions are based on the IP addresses like access control, which determines which IP address gets access and which does not. Other situations like distributing resources based on the client IP addresses within an implementation system, the resources could be the amount of cached replies.

When considering using IP addresses as keys into these scenarios, the variability of the bits in the IP addresses SHOULD be considered. In IPv6 for the same interface the different address differ mostly in the non subnet part, in IPv4 that is not the case. The lower order bits are generated from the MAC addresses and are mostly static.

11.3. NFSv4 Id Mapping

The "dns_domain" in "user@dns_domain" as referred in section 5.8 [RFC3530], used to map owners, groups, users and user groups in the string principals to internal representations at the client and server SHOULD be the same for the same client accessing an NFSv4 server simultaneously over IPv4 and IPv6.

12. Consideration for running NFS along with NAT/ALG

For NFS ALGs, the only information at the protocol level is the embedded universal addresses, which SHOULD be translated for RPCBIND and SETCLIENTID calls to work transparently. If names are used as identifiers then DNS-ALG should be used for translating names. The ALG SHOULD NOT multiplex multiple client side connections to single server side connection.

12.1. Asymmetric Reachability

If the server is in IPv6/IPv4 address space and the client is in IPv4/IPv6 address space connected through intermediate translation network element, then a separate callback port can be fixed.

13. Acknowledgments

The authors would like to acknowledge Mike Eisler for reviews of the various early versions of the draft.

14. IANA Considerations

This memo includes no request to IANA.

15. Security Considerations

All considerations from RFC 3530 Section 16 [RFC3530]

16. References

16.1. Normative References

[RFC1813] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, June 1995.

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, August 1995.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://xml.resource.org/public/rfc/html/rfc2119.html>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, March 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4506] Eisler, M., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006.

[RFC5378] Bradner, S. and J. Contreras, "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, November 2008.

[RFC5531] Thurlow, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 5531, May 2009.

[netid_ID] Eisler, M., "IANA Considerations for RPC Net Identifiers and Universal Address Formats", draft-ietf-nfsv4-rpc-netid-04 (work in progress), December 2008.

16.2. Informative References

[RFC1094] Nowicki, B., "NFS: Network File System Protocol specification", RFC 1094, March 1989.

[RFC2624] Shepler, S., "NFS Version 4 Design Considerations", RFC 2624, June 1999.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, May 2003.

[RFC3593] Tesink, K., "Textual Conventions for MIB Modules Using Performance History Based on 15 Minute Intervals", RFC 3593, September 2003.

[RFC4620] Crawford, M. and B. Haberman, "IPv6 Node Information Queries", RFC 4620, August 2006.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

Authors' Addresses

Alex RN (editor)
NetApp
3rd Floor, Fair Winds Block, EGL Software Park,
Bangalore, Karnataka 560071
IN

Phone: +91-80-41843352
Email: rnalex@netapp.com

Bhargo Sunil (editor)
NetApp
3rd Floor, Fair Winds Block, EGL Software Park,
Bangalore, Karnataka 560071
IN

Phone: +91-80-41843963
Email: bhargo@netapp.com

Dhawal Bhagwat
NetApp
3rd Floor, Fair Winds Block, EGL Software Park,
Bangalore, Karnataka 560071
IN

Phone: +91-80-41843134
Email: dhawal@netapp.com

Dipankar Roy
NetApp

Phone: +91-80-41843303
Email: dipankar@netapp.com

Rishikesh Barooah
NetApp

Email: rbarooah@netapp.com