Network File System (NFS) Upper Layer Binding To RPC-Over-RDMA Version One
draft-ietf-nfsv4-rfc5667bis-08

Abstract

   This document specifies Upper Layer Bindings of Network File System
   (NFS) protocol versions to RPC-over-RDMA Version One, enabling the
   use of Direct Data Placement. This document obsoletes RFC 5667.

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 [RFC2119].

Status of This Memo

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

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   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."

   This Internet-Draft will expire on October 6, 2017.

Copyright Notice

Table of Contents

1. Introduction

The RPC-over-RDMA Version One transport may employ direct data
placement to convey data payloads associated with RPC transactions
[I-D.ietf-nfsv4-rfc5666bis]. To enable successful interoperation,
RPC client and server implementations using RPC-over-RDMA Version One
must agree which XDR data items and RPC procedures are eligible to
use direct data placement (DDP).

An Upper Layer Binding specifies this agreement for one RPC Program.
Other operational details, such as RPC binding assignments, pairing
Write chunks with result data items, and reply size estimation, are
also specified by this Binding.

This document contains material required of Upper Layer Bindings, as
specified in [I-D.ietf-nfsv4-rfc5666bis], for the following NFS
protocol versions:

o NFS Version 2 [RFC1094]

o NFS Version 3 [RFC1813]

o NFS Version 4.0 [RFC7530]

o NFS Version 4.1 [RFC5661]

o NFS Version 4.2 [RFC7862]

Upper Layer Bindings are also provided for auxiliary protocols used
with NFS versions 2 and 3.

This document assumes the reader is already familiar with concepts
and terminology defined in [I-D.ietf-nfsv4-rfc5666bis] and the
documents it references.

2. Reply Size Estimation

During the construction of each RPC Call message, a requester is
responsible for allocating appropriate resources for receiving the
corresponding Reply message. If the requester expects the RPC Reply
message will be larger than its inline threshold, it provides Write
and/or Reply chunks wherein the responder can place results and the
reply's Payload stream.

A reply resource overrun occurs if the RPC Reply Payload stream does
not fit into the provided Reply chunk, or no Reply chunk was provided
and the Payload stream does not fit inline. This prevents the
responder from returning the Upper Layer reply to the requester.

Therefore reliable reply size estimation is necessary to ensure
successful interoperation.

In most cases, the NFS protocol's XDR definition provides enough
information to enable an NFS client to predict the maximum size of
the expected Reply message. If there are variable-size data items in
the result, the maximum size of the RPC Reply message can be
estimated as follows:

o The client requests only a specific portion of an object (for
   example, using the "count" and "offset" fields in an NFS READ).

o The client limits the number of results (e.g. using the "count"
   field of an NFS READDIR request).

o The client has already cached the size of the whole object it is
   about to request (say, via a previous NFS GETATTR request).

o The client and server have negotiated a maximum size for all calls
   and responses (using a CREATE_SESSION operation, for instance).

2.1. Short Reply Chunk Retry

In a few cases, either the size of one or more returned data items or
the number of returned data items cannot be known in advance of
forming an RPC Call.

If an NFS server finds that the NFS client provided inadequate
receive resources to return the whole reply, it returns an RPC level
error or a transport error, such as ERR_CHUNK.

In response to these errors, an NFS client can choose to:

o Terminate the RPC transaction immediately with an error, or

o Allocate a larger Reply chunk and send the same request as a new
   RPC transaction (to avoid hitting in a Duplicate Reply Cache).
   The NFS client should avoid retrying the request indefinitely
   because a responder may return ERR_CHUNK for a variety of reasons.

Subsequent sections of this document discuss exactly which operations
might have ultimate difficulty with Reply size estimation. These
operations are eligible for "short Reply chunk retry." Unless
explicitly mentioned as applicable, short Reply chunk retry should
not be used.

   NFS server implementations can avoid connection loss by first
   confirming that target RDMA segments are large enough to receive
   results before initiating explicit RDMA operations.

3. Upper Layer Binding for NFS Versions 2 and 3

   The Upper Layer Binding specification in this section applies to NFS
   Version 2 [RFC1094] and NFS Version 3 [RFC1813]. For brevity, in
   this document a "Legacy NFS client" refers to an NFS client using the
   NFS version 2 or NFS version 3 RPC Programs (100003) to communicate
   with an NFS server. Likewise, a "Legacy NFS server" is an NFS server
   communicating with clients using NFS version 2 or NFS version 3.

   The following XDR data items in NFS versions 2 and 3 are DDP-
   eligible:

   o The opaque file data argument in the NFS WRITE procedure

   o The pathname argument in the NFS SYMLINK procedure

   o The opaque file data result in the NFS READ procedure

   o The pathname result in the NFS READLINK procedure

   All other argument or result data items in NFS versions 2 and 3 are
   not DDP-eligible.

   A transport error does not give an indication of whether the server
   has processed the arguments of the RPC Call, or whether the server
   has accessed or modified client memory associated with that RPC.

3.1. Reply Size Estimation

   A Legacy NFS client determines the maximum reply size for each
   operation using the criteria outlined in Section 2. There are no
   operations in NFS version 2 or 3 that benefit from short Reply chunk
   retry.

3.2. RPC Binding Considerations

   Legacy NFS servers traditionally listen for clients on UDP and TCP
   port 2049. Additionally, they register these ports with a local
   portmapper [RFC1833] service.

   A Legacy NFS server supporting RPC-over-RDMA Version One on such a
   network and registering itself with the RPC portmapper MAY choose an
   arbitrary port, or MAY use the alternative well-known port number for
   its RPC-over-RDMA service (see Section 8). The chosen port MAY be

registered with the RPC portmapper under the netids assigned in
[I-D.ietf-nfsv4-rfc5666bis].

4. Upper Layer Bindings for NFS Version 2 and 3 Auxiliary Protocols

NFS versions 2 and 3 are typically deployed with several other
protocols, sometimes referred to as "NFS auxiliary protocols." These
are distinct RPC Programs that define procedures which are not part
of the NFS version 2 or version 3 RPC Programs. The Upper Layer
Bindings in this section apply to:

o Versions 2 and 3 of the MOUNT protocol [RFC1813]

o Versions 1, 3, and 4 of the NLM protocol [RFC1813]

o Version 1 of the NSM protocol, described in Chapter 11 of [XNFS]

o Version 1 of the NFSACL protocol, which does not have a public
   definition. NFSACL is treated in this document as a de facto
   standard, as there are several interoperating implementations.

4.1. MOUNT, NLM, and NSM Protocols

Historically, NFS/RDMA implementations have chosen to convey the
MOUNT, NLM, and NSM protocols via TCP. To enable interoperation of
these protocols when NFS/RDMA is in use, a legacy NFS server MUST
provide TCP-based MOUNT, NLM, and NSM services.

4.2. NFSACL Protocol

Legacy clients and servers that support the NFSACL RPC Program
typically convey NFSACL procedures on the same connection as NFS RPC
Programs. This obviates the need for separate rpcbind queries to
discover server support for this RPC Program.

ACLs are typically small, but even large ACLs must be encoded and
decoded to some degree. Thus no data item in this Upper Layer
Protocol is DDP-eligible.

For procedures whose replies do not include an ACL object, the size
of a reply is determined directly from the NFSACL RPC Program's XDR
definition.

There is no protocol-specified size limit for NFS version 3 ACLs, and
there is no mechanism in either the NFSACL or NFS RPC Programs for a
Legacy client to ascertain the largest ACL a Legacy server can
return. Legacy client implementations should choose a maximum size
for ACLs based on their own internal limits.

Because an NFSACL client cannot know in advance how large a returned
ACL will be, it can use short Reply chunk retry when an NFSACL GETACL
operation encounters a transport error.

5. Upper Layer Binding For NFS Version 4

The Upper Layer Binding specification in this section applies to RPC
Programs defined in NFS Version 4.0 [RFC7530], NFS Version 4.1
[RFC5661], and NFS Version 4.2 [RFC7862].

5.1. DDP-Eligibility

Only the following XDR data items in the COMPOUND procedure of all
NFS version 4 minor versions are DDP-eligible:

o The opaque data field in the WRITE4args structure

o The linkdata field of the NF4LNK arm in the createtype4 union

o The opaque data field in the READ4resok structure

o The linkdata field in the READLINK4resok structure

5.2. Reply Size Estimation

Within NFS version 4, there are certain variable-length result data
items whose maximum size cannot be estimated by clients reliably
because there is no protocol-specified size limit on these arrays.
These include:

o The attrlist4 field

o Fields containing ACLs such as fattr4_acl, fattr4_dacl,
   fattr4_sacl

o Fields in the fs_locations4 and fs_locations_info4 data structures

o Fields opaque to the NFS version 4 protocol which pertain to pNFS
   layout metadata, such as loc_body, loh_body, da_addr_body,
   lou_body, lrf_body, fattr_layout_types and fs_layout_types,

5.2.1. Reply Size Estimation for Minor Version 0

The NFS version 4.0 protocol itself does not impose any bound on the
size of NFS calls or responses.

Some of the data items enumerated in Section 5.2 (in particular, the
items related to ACLs and fs_locations) make it difficult to predict

the maximum size of NFS version 4.0 replies that interrogate
variable-length fattr4 attributes. Client implementations might rely
on their own internal architectural limits to constrain the reply
size, but such limits are not always guaranteed to be reliable.

When an especially large fattr4 result is expected, a Reply chunk
might be required. An NFS version 4.0 client can use short Reply
chunk retry when an NFS COMPOUND containing a GETATTR operation
encounters a transport error.

The use of NFS COMPOUND operations raises the possibility of requests
that combine a non-idempotent operation (e.g. RENAME) with a GETATTR
operation that requests one or more variable-length results. This
combination should be avoided by ensuring that any GETATTR operation
that requests a result of unpredictable length is sent in an NFS
COMPOUND by itself.

5.2.2. Reply Size Estimation for Minor Version 1 and Newer

In NFS version 4.1 and newer minor versions, the csa_fore_chan_attrs
argument of the CREATE_SESSION operation contains a
ca_maxresponsesize field. The value in this field can be taken as
the absolute maximum size of replies generated by an NFS version 4.1
server.

This value can be used in cases where it is not possible to estimate
a reply size upper bound precisely. In practice, objects such as
ACLs, named attributes, layout bodies, and security labels are much
smaller than this maximum.

5.3. RPC Binding Considerations

NFS version 4 servers are required to listen on TCP port 2049, and
they are not required to register with an rpcbind service [RFC7530].

Therefore, an NFS version 4 server supporting RPC-over-RDMA Version
One MUST use the alternative well-known port number for its RPC-over-
RDMA service (see Section 8). Clients SHOULD connect to this well-
known port without consulting the RPC portmapper (as for NFS version
4 on TCP transports).

5.4. NFS COMPOUND Requests

5.4.1. Multiple DDP-eligible Data Items

An NFS version 4 COMPOUND procedure can contain more than one
operation that carries a DDP-eligible data item. An NFS version 4
client provides XDR Position values in each Read chunk to

disambiguate which chunk is associated with which argument data item.
However NFS version 4 server and client implementations must agree in
advance on how to pair Write chunks with returned result data items.

In the following list, a "READ operation" refers to any NFS Version 4
operation which has a DDP-eligible result data item. The mechanism
specified in Section 4.3.2 of [I-D.ietf-nfsv4-rfc5666bis]) is applied
to this class of operations:

o If an NFS version 4 client wishes all DDP-eligible items in an NFS
   reply to be conveyed inline, it leaves the Write list empty.

o The first chunk in the Write list MUST be used by the first READ
   operation in an NFS version 4 COMPOUND procedure. The next Write
   chunk is used by the next READ operation, and so on.

o If an NFS version 4 client has provided a matching non-empty Write
   chunk, then the corresponding READ operation MUST return its DDP-
   eligible data item using that chunk.

o If an NFS version 4 client has provided an empty matching Write
   chunk, then the corresponding READ operation MUST return all of
   its result data items inline.

o If a READ operation returns a union arm which does not contain a
   DDP-eligible result, and the NFS version 4 client has provided a
   matching non-empty Write chunk, an NFS version 4 server MUST
   return an empty Write chunk in that Write list position.

o If there are more READ operations than Write chunks, then
   remaining NFS Read operations in an NFS version 4 COMPOUND that
   have no matching Write chunk MUST return their results inline.

5.4.2. NFS Version 4 COMPOUND Example

The following example shows a Write list with three Write chunks, A,
B, and C. The NFS version 4 server consumes the provided Write
chunks by writing the results of the designated operations in the
compound request (READ and READLINK) back to each chunk.

Write list:

```
    A --> B --> C
```

NFS version 4 COMPOUND request:

```
    PUTFH LOOKUP READ PUTFH LOOKUP READLINK PUTFH LOOKUP READ
                  |                 |                       |
                  v                 v                       v
                  A                 B                       C
```

If the NFS version 4 client does not want to have the READLINK result
returned via RDMA, it provides an empty Write chunk for buffer B to
indicate that the READLINK result must be returned inline.

5.5. NFS Callback Requests

The NFS version 4 family of protocols support server-initiated
callbacks to notify NFS version 4 clients of events such as recalled
delegations.

5.5.1. NFS Version 4.0 Callback

NFS version 4.0 implementations typically employ a separate TCP
connection to handle callback operations, even when the forward
channel uses an RPC-over-RDMA Version One transport.

No operation in the NFS version 4.0 callback RPC Program conveys a
significant data payload. Therefore, no XDR data items in this RPC
Program is DDP-eligible.

A CB_RECALL reply is small and fixed in size. The CB_GETATTR reply
contains a variable-length fattr4 data item. See Section 5.2.1 for a
discussion of reply size prediction for this data item.

An NFS version 4.0 client advertises netids and ad hoc port addresses
for contacting its NFS version 4.0 callback service using the
SETCLIENTID operation.

5.5.2. NFS Version 4.1 Callback

In NFS version 4.1 and newer minor versions, callback operations may
appear on the same connection as is used for NFS version 4 forward
channel client requests. NFS version 4 clients and servers MUST use
the approach described in [I-D.ietf-nfsv4-rpcrdma-bidirection] when
backchannel operations are conveyed on RPC-over-RDMA Version One
transports.

The csa_back_chan_attrs argument of the CREATE_SESSION operation
contains a ca_maxresponsesize field. The value in this field can be
taken as the absolute maximum size of backchannel replies generated
by a replying NFS version 4 client.

There are no DDP-eligible data items in callback procedures defined
in NFS version 4.1 or NFS version 4.2. However, some callback
operations, such as messages that convey device ID information, can
be large, in which case a Long Call or Reply might be required.

When an NFS version 4.1 client can support Long Calls in its
backchannel, it reports a backchannel ca_maxrequestsize that is
larger than the connection's inline thresholds. Otherwise an NFS
version 4 server MUST use only Short messages to convey backchannel
operations.

5.6. Session-Related Considerations

The presence of an NFS session (defined in [RFC5661]) has no effect
on the operation of RPC-over-RDMA Version One. None of the
operations introduced to support NFS sessions (e.g. the SEQUENCE
operation) contain DDP-eligible data items. There is no need to
match the number of session slots with the number of available RPC-
over-RDMA credits.

However, there are a few new cases where an RPC transaction can fail.
For example, a requester might receive, in response to an RPC
request, an RDMA_ERROR message with an rdma_err value of ERR_CHUNK,
or an RDMA_MSG containing an RPC_GARBAGEARGS reply. These situations
are no different from existing RPC errors which an NFS session
implementation is already prepared to handle for other transports.
And as with other transports during such a failure, there might be no
SEQUENCE result available to the requester to distinguish whether
failure occurred before or after the requested operations were
executed on the responder.

When a transport error occurs (e.g. RDMA_ERROR), the requester
proceeds as usual to match the incoming XID value to a waiting RPC
Call. The RPC transaction is terminated, and the result status is
reported to the Upper Layer Protocol. The requester's session
implementation then determines the session ID and slot for the failed
request, and performs slot recovery to make that slot usable again.
If this is not done, that slot could be rendered permanently
unavailable.

5.7. Transport Considerations

5.7.1. Congestion Avoidance

Section 3.1 of [RFC7530] states:

> Where an NFSv4 implementation supports operation over the IP
> network protocol, the supported transport layer between NFS and IP
> MUST be an IETF standardized transport protocol that is specified
> to avoid network congestion; such transports include TCP and the
> Stream Control Transmission Protocol (SCTP).

Section 2.9.1 of [RFC5661] also states:

> Even if NFSv4.1 is used over a non-IP network protocol, it is
> RECOMMENDED that the transport support congestion control.

> It is permissible for a connectionless transport to be used under
> NFSv4.1; however, reliable and in-order delivery of data combined
> with congestion control by the connectionless transport is
> REQUIRED. As a consequence, UDP by itself MUST NOT be used as an
> NFSv4.1 transport.

RPC-over-RDMA Version One is constructed on a platform of RDMA
Reliable Connections [I-D.ietf-nfsv4-rfc5666bis] [RFC5041]. RDMA
Reliable Connections are reliable, connection-oriented transports
that guarantee in-order delivery, meeting all above requirements for
NFS version 4 transports.

5.7.2. Retransmission and Keep-alive

NFS version 4 client implementations often rely on a transport-layer
keep-alive mechanism to detect when an NFS version 4 server has
become unresponsive. When an NFS server is no longer responsive,
client-side keep-alive terminates the connection, which in turn
triggers reconnection and RPC retransmission.

Some RDMA transports (such as Reliable Connections on InfiniBand)
have no keep-alive mechanism. Without a disconnect or new RPC
traffic, such connections can remain alive long after an NFS server
has become unresponsive. Once an NFS client has consumed all
available RPC-over-RDMA credits on that transport connection, it will
forever await a reply before sending another RPC request.

NFS version 4 clients SHOULD reserve one RPC-over-RDMA credit to use
for periodic server or connection health assessment. This credit can
be used to drive an RPC request on an otherwise idle connection,

   triggering either a quick affirmative server response or immediate
   connection termination.

   In addition to network partition and request loss scenarios, RPC-
   over-RDMA transport connections can be terminated when a Transport
   header is malformed, Reply messages are larger than receive
   resources, or when too many RPC-over-RDMA messages are sent at once.
   In such cases:

   o If there is a transport error indicated (ie, RDMA_ERROR) before
      the disconnect or instead of a disconnect, the requester MUST
      respond to that error as prescribed by the specification of the
      RPC transport. Then the NFS version 4 rules for handling
      retransmission apply.

   o If there is a transport disconnect and the responder has provided
      no other response for a request, then only the NFS version 4 rules
      for handling retransmission apply.

6. Extending NFS Upper Layer Bindings

   RPC Programs such as NFS are required to have an Upper Layer Binding
   specification to interoperate on RPC-over-RDMA Version One transports
   [I-D.ietf-nfsv4-rfc5666bis]. Via standards action, the Upper Layer
   Binding specified in this document can be extended to cover versions
   of the NFS version 4 protocol specified after NFS version 4 minor
   version 2, or separately published extensions to an existing NFS
   version 4 minor version, as described in [I-D.ietf-nfsv4-versioning].

7. Security Considerations

   RPC-over-RDMA Version One supports all RPC security models, including
   RPCSEC_GSS security and transport-level security [RFC2203]. The
   choice of what Direct Data Placement mechanism to convey RPC argument
   and results does not affect this, since it changes only the method of
   data transfer. Specifically, the requirements of
   [I-D.ietf-nfsv4-rfc5666bis] ensure that this choice does not
   introduce new vulnerabilities.

   Because this document defines only the binding of the NFS protocols
   atop [I-D.ietf-nfsv4-rfc5666bis], all relevant security
   considerations are therefore to be described at that layer.

8. IANA Considerations

   The use of direct data placement in NFS introduces a need for an
   additional port number assignment for networks that share traditional

UDP and TCP port spaces with RDMA services. The iWARP protocol is
such an example [RFC5041] [RFC5040].

For this purpose, a set of transport protocol port number assignments
is specified by this document. IANA has assigned the following ports
for NFS/RDMA in the IANA port registry, according to the guidelines
described in [RFC6335].


```
nfsrdma 20049/tcp Network File System (NFS) over RDMA
nfsrdma 20049/udp Network File System (NFS) over RDMA
nfsrdma 20049/sctp Network File System (NFS) over RDMA
```


This document should be listed as the reference for the nfsrdma port
assignments. This document does not alter these assignments.

9. References

9.1. Normative References

[I-D.ietf-nfsv4-rfc5666bis]
           Lever, C., Simpson, W., and T. Talpey, "Remote Direct
           Memory Access Transport for Remote Procedure Call, Version
           One", draft-ietf-nfsv4-rfc5666bis-11 (work in progress),
           March 2017.

[I-D.ietf-nfsv4-rpcrdma-bidirection]
           Lever, C., "Bi-directional Remote Procedure Call On RPC-
           over-RDMA Transports", draft-ietf-nfsv4-rpcrdma-
           bidirection-08 (work in progress), March 2017.

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2",
           RFC 1833, DOI 10.17487/RFC1833, August 1995,
           <http://www.rfc-editor.org/info/rfc1833>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119,
           DOI 10.17487/RFC2119, March 1997,
           <http://www.rfc-editor.org/info/rfc2119>.

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol
           Specification", RFC 2203, DOI 10.17487/RFC2203, September
           1997, <http://www.rfc-editor.org/info/rfc2203>.

[RFC5661]   Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed.,
            "Network File System (NFS) Version 4 Minor Version 1
            Protocol", RFC 5661, DOI 10.17487/RFC5661, January 2010,
            <http://www.rfc-editor.org/info/rfc5661>.

[RFC6335]   Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S.
            Cheshire, "Internet Assigned Numbers Authority (IANA)
            Procedures for the Management of the Service Name and
            Transport Protocol Port Number Registry", BCP 165,
            RFC 6335, DOI 10.17487/RFC6335, August 2011,
            <http://www.rfc-editor.org/info/rfc6335>.

[RFC7530]   Haynes, T., Ed. and D. Noveck, Ed., "Network File System
            (NFS) Version 4 Protocol", RFC 7530, DOI 10.17487/RFC7530,
            March 2015, <http://www.rfc-editor.org/info/rfc7530>.

[RFC7862]   Haynes, T., "Network File System (NFS) Version 4 Minor
            Version 2 Protocol", RFC 7862, DOI 10.17487/RFC7862,
            November 2016, <http://www.rfc-editor.org/info/rfc7862>.

9.2. Informative References

[I-D.ietf-nfsv4-versioning]
            Noveck, D., "Rules for NFSv4 Extensions and Minor
            Versions", draft-ietf-nfsv4-versioning-09 (work in
            progress), December 2016.

[RFC1094]   Nowicki, B., "NFS: Network File System Protocol
            specification", RFC 1094, DOI 10.17487/RFC1094, March
            1989, <http://www.rfc-editor.org/info/rfc1094>.

[RFC1813]   Callaghan, B., Pawlowski, B., and P. Staubach, "NFS
            Version 3 Protocol Specification", RFC 1813,
            DOI 10.17487/RFC1813, June 1995,
            <http://www.rfc-editor.org/info/rfc1813>.

[RFC5040]   Recio, R., Metzler, B., Culley, P., Hilland, J., and D.
            Garcia, "A Remote Direct Memory Access Protocol
            Specification", RFC 5040, DOI 10.17487/RFC5040, October
            2007, <http://www.rfc-editor.org/info/rfc5040>.

[RFC5041]   Shah, H., Pinkerton, J., Recio, R., and P. Culley, "Direct
            Data Placement over Reliable Transports", RFC 5041,
            DOI 10.17487/RFC5041, October 2007,
            <http://www.rfc-editor.org/info/rfc5041>.

   [RFC5666]   Talpey, T. and B. Callaghan, "Remote Direct Memory Access
               Transport for Remote Procedure Call", RFC 5666,
               DOI 10.17487/RFC5666, January 2010,
               <http://www.rfc-editor.org/info/rfc5666>.

   [RFC5667]   Talpey, T. and B. Callaghan, "Network File System (NFS)
               Direct Data Placement", RFC 5667, DOI 10.17487/RFC5667,
               January 2010, <http://www.rfc-editor.org/info/rfc5667>.

   [XNFS]   The Open Group, "Protocols for Interworking: XNFS, Version
               3W", February 1998.

Appendix A. Changes Since RFC 5667

   Corrections and updates made necessary by new language in
   [I-D.ietf-nfsv4-rfc5666bis] have been introduced. For example,
   references to deprecated features of RPC-over-RDMA Version One, such
   as RDMA_MSGP, and the use of the Read list for handling RPC replies,
   have been removed. The term "mapping" has been replaced with the
   term "binding" or "Upper Layer Binding" throughout the document.
   Material that duplicates what is in [I-D.ietf-nfsv4-rfc5666bis] has
   been deleted.

   Material required by [I-D.ietf-nfsv4-rfc5666bis] for Upper Layer
   Bindings that was not present in [RFC5667] has been added. A
   complete discussion of reply size estimation has been introduced for
   all protocols covered by the Upper Layer Bindings in this document.

   Technical corrections have been made. For example, the mention of
   12KB and 36KB inline thresholds have been removed. The reference to
   a non-existant NFS version 4 SYMLINK operation has been replaced.

   The discussion of NFS version 4 COMPOUND handling has been completed.
   Some changes were made to the algorithm for matching DDP-eligible
   results to Write chunks.

   Requirements to ignore extra Read or Write chunks have been removed
   from the NFS version 2 and 3 Upper Layer Binding, as they conflict
   with [I-D.ietf-nfsv4-rfc5666bis].

   A section discussing NFS version 4 retransmission and connection loss
   has been added.

   The following additional improvements have been made, relative to
   [RFC5667]:

   o An explicit discussion of NFS version 4.0 and NFS version 4.1
      backchannel operation has replaced the previous treatment of
      callback operations.

   o A binding for NFS version 4.2 has been added.

   o A section suggesting a mechanism for periodically assessing
      connection health has been introduced.

   o Ambiguous or erroneous uses of RFC2119 terms have been corrected.

   o References to obsolete RFCs have been updated.

   o An IANA Considerations Section has been added, which specifies the
      port assignments for NFS/RDMA. This replaces the example
      assignment that appeared in [RFC5666].

   o Code excerpts have been removed, and figures have been modernized.

Appendix B. Acknowledgments

   The author gratefully acknowledges the work of Brent Callaghan and
   Tom Talpey on the original NFS Direct Data Placement specification
   [RFC5667]. The author also wishes to thank Bill Baker and Greg
   Marsden for their support of this work.

   Dave Noveck provided excellent review, constructive suggestions, and
   consistent navigational guidance throughout the process of drafting
   this document. Dave also contributed the text of Section 5.6 and
   Section 6, and insisted on precise discussion of reply size
   estimation.

   Thanks to Karen Deitke for her sharp observations about idempotency,
   and the clarity of the discussion of NFS COMPOUNDs and NFS sessions.

   Special thanks go to Transport Area Director Spencer Dawkins, nfsv4
   Working Group Chair Spencer Shepler, and nfsv4 Working Group
   Secretary Thomas Haynes for their support.

Author's Address

   Charles Lever (editor)
   Oracle Corporation
   1015 Granger Avenue
   Ann Arbor, MI 48104
   USA

   Phone: +1 248 816 6463
   Email: chuck.lever@oracle.com