

Network File System Version 4
Internet-Draft
Intended status: Standards Track
Expires: January 26, 2018

D. Noveck
July 25, 2017

Transport-generic Network File System (NFS) Upper Layer Bindings To RPC-
Over-RDMA
[draft-dnoveck-nfsv4-nfsulb-01](#)

Abstract

This document specifies Upper Layer Bindings to allow use of RPC-over-RDMA by protocols related to the Network File System (NFS). Such bindings are required when using RPC-over-RDMA, in order to enable use of Direct Data Placement and for a number of other reasons. These bindings are structured to be applicable to all known version of the RPC-over-RDMA transport, including optional extensions. All versions of NFS are addressed.

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 January 26, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

Internet-Draft

Transport-generic NFS ULBs

July 2017

This document is subject to [BCP 78](http://trustee.ietf.org/license-info) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	3
2.	Conveying NFS Operations On RPC-Over-RDMA	3
2.1.	Direct Placement of Request Data	4
2.2.	Direct Placement of Response Data	5
2.3.	Scatter-gather when Using DDP	5
2.4.	DDP-eligibility Violations	5
2.5.	Long Calls and Replies	6
3.	Preparatory Material for Multiple Bindings	7
3.1.	Reply Size Estimation	7
3.2.	Retry to Deal with Reply Size Mis-estimation	8
4.	Upper Layer Binding for NFS Versions 2 And 3	9
4.1.	Auxiliary Protocols	9
4.1.1.	MOUNT, NLM, And NSM Protocols	10
4.1.2.	NFSACL Protocol	10
5.	Upper Layer Binding for NFS Version 4	10
5.1.	DDP-Eligibility	11
5.1.1.	READ_PLUS Replies	11
5.2.	NFS Version 4 Reply Size Estimation	12
5.2.1.	Reply Size Estimation for Minor Version 0	12

5.2.2.	Reply Size Estimation for Minor Version 1 And Newer .	12
5.3.	NFS Version 4 COMPOUND Requests	13
5.3.1.	NFS Version 4 COMPOUND Example	13
5.4.	NFS Version 4 Callback	14
5.4.1.	NFS Version 4.0 Callback	14

5.4.2.	NFS Version 4.1 Callback	14
5.5.	Session-Related Considerations	15
5.6.	Connection Keep-Alive	16
6.	Extending NFS Upper Layer Bindings	16
7.	IANA Considerations	17
8.	Security Considerations	17
9.	References	17
9.1.	Normative References	18
9.2.	Informative References	18
Appendix A.	Acknowledgments	19
	Author's Address	20

[1.](#) Introduction

An RPC-over-RDMA transport, such as the ones defined in [[RFC8166](#)] and [[rpcrdmav2](#)], may employ direct data placement to convey data payloads associated with RPC transactions. To enable successful interoperation, RPC client and server implementations must agree as to which XDR data items in what particular RPC procedures are eligible for direct data placement (DDP). Specifying those data items is a major component of a protocol's Upper Layer Binding.

In addition, Upper Layer Bindings are required to include additional information to assure that adequate resources are allocated to receive RPC replies, and for a number of other reasons.

This document contains material required of Upper Layer Bindings, as specified in [[RFC8166](#)], for the NFS protocol versions listed below. In addition, bindings are provided, when necessary, for auxiliary protocols used together with NFS versions 2 and 3.

- 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](#)]

2. Conveying NFS Operations On RPC-Over-RDMA

This document is written to apply to multiple versions of the RPC-over-RDMA transport, only the first of which is currently specified by a Proposed Standard (in [[RFC8166](#)]). However, it is expected that other versions will be created, and this document has been structured to support future versions by focusing on the functions to be

provided by the transport and the transport limitations which the Upper Layer Protocols need to accommodate, allowing the transport specification and the specifications for associated extensions to define how those functions will be provided and the details of the transport limitations.

In the subsections that follow, we will describe the generic function to be provided or limitation to be accommodated and follow it with material that describes, in general, how that issue is dealt with in Version One. For more detail about Version One, [[RFC8166](#)] should be consulted. How these issues are to be dealt with in future versions is left to the specification documents for those versions and associated documents defining optional extensions.

For example:

- o ULBs within this document define which data items are eligible for Direct Data Placement while transport versions might differ as how this is to be effected. See Sections [2.1](#) and [2.2](#) for more detail. In both cases, [Section 2.3](#) discusses issues connected with the use of discontinuous areas for Direct Data Placement.
- o [Section 2.4](#) defines the concept of a DDP-eligibility violation and requires that such violations be reported while transport versions might differ as to the manner in which the reporting is to be done.
- o [Section 2.5](#) discusses issues arising from limits on the size of messages conveyed using RDMA SENDs. Different transport version

may have different size limits, while, in the case of replies the ULBs are responsible for specifying how limits on reply sizes are to be determined.

[2.1.](#) Direct Placement of Request Data

When DDP-eligible XDR data items appear in a request the requester needs to take special actions in order to provide for the direct placement of those items in the responder's memory. The specific actions to be taken are defined by the transport version being used.

In Version One, the Read list in each RPC-over-RDMA transport header represents a set of memory regions containing a single item of DDP-eligible NFS argument data. Large data items, such as the data payload of an NFS version 3 WRITE procedure, can be referenced by the Read list. The NFS server pulls such payloads from the client and places them directly into its own memory.

[2.2.](#) Direct Placement of Response Data

When a request is such that it is possible for DDP-eligible data items to appear in the corresponding reply, the requester needs to take special actions in order to provide for the direct placement of those items in the requester's memory, if such placement is desired. The specific actions to be taken are defined by the transport version being used as is the means to indicate that such direct placement is not to be done

In Version One, the Write list in each RPC-over-RDMA transport header represents a set of memory regions that can receive DDP-eligible NFS result data. Large data items, such as the payload of an NFS version 3 READ procedure, can be referenced by the Write list. The NFS server pushes such payloads to the client, placing them directly into the client's memory, using target addresses provided by the client when sending the request.

Each Write chunk corresponds to a specific XDR data item in an NFS reply. This document describes how NFS client and server implementations determine the correspondence between Write chunks and XDR results.

[2.3.](#) Scatter-gather when Using DDP

In order to accommodate the storage of multiple data blocks within individual cache buffers, the RPC-over-RDMA transport allows the addresses to which a DDP-eligible data item to be discontinuous. How these addresses are indicated depends on the transport version.

Within Version One, a chunk typically corresponds to exactly one XDR data item. Each Read chunk is represented as a list of segments at the same XDR Position. Each Write chunk is represented as an array of segments. An NFS client thus has the flexibility to advertise a set of discontinuous memory regions in which to convey a single DDP-eligible XDR data item.

[2.4.](#) DDP-eligibility Violations

When the transport header uses the means defined to directly place on an XDR item is applied to an XDR item not define in the ULB as DDP-eligible, a DDP-eligibility violation is recognized. The means by which such violations are to be reported is defined by the particular transport version being used.

To report a DDP-eligibility violation within Version One, an NFS server returns one of:

Noveck	Expires January 26, 2018	[Page 5]
--------	--------------------------	----------

Internet-Draft	Transport-generic NFS ULBs	July 2017
----------------	----------------------------	-----------

- o An RPC-over-RDMA message of type RDMA_ERROR, with the rdma_xid field set to the XID of the matching NFS Call, and the rdma_error field set to ERR_CHUNK
- o An RPC message (via an RDMA_MSG message) with the xid field set to the XID of the matching NFS Call, the mtype field set to REPLY, the stat field set to MSG_ACCEPTED, and the accept_stat field set to GARBAGE_ARGS.

[2.5.](#) Long Calls and Replies

Because of the use of pre-posted receive buffers whose size is fixed, all RPC-over-RDMA transport versions have limits on the size of messages which can be conveyed without use of explicit RDMA operations, although different transport versions may have different

limits. In particular, when the transport version allows messages to be continued across multiple RDMA SENDs, the limit can be substantially greater than the receive buffer size. Also note that the size of the messages allowed may be reduced because of space taken up by the transport header fields.

Each transport version is responsible for defining the message size limits and the means by which the transfer of messages that exceed these limits is to be provided for. These means may be different in the cases of long calls and replies.

When using Version One, if an NFS request is too large to be conveyed within the NFS server's responder inline threshold, even after any DDP-eligible data items have been removed, an NFS client must send the request in the form of a Long Call. The entire NFS request is sent in a special Read chunk called a Position Zero Read chunk.

Also when using Version One, if an NFS client determines that the maximum size of an NFS reply could be too large to be conveyed within its own inline threshold, it provides a Reply chunk in the RPC-over-RDMA transport header conveying the NFS request. The server places the entire NFS reply in the Reply chunk.

There exist cases in which an NFS client needs to provide both a Position Zero Read chunk and a Reply chunk for the same RPC. One common source of such situations is when the RPC authentication flavor being used requires that DDP-eligible data items never be removed from RPC messages.

[3.](#) Preparatory Material for Multiple Bindings

Although each of the NFS versions and each of the auxiliary protocols discussed in [Section 4.1](#) has its own ULB, there is important preparatory material in the subsections below that applies to multiple ULBs. In particular:

- o The material in [Section 3.1](#) applies to all of the ULBs discussed

in this document.

- o The material in [Section 3.2](#) applies to NFSv2, NFSv3, NFSv4.0, the MOUNT protocol, and the NFSACL protocol.

[3.1.](#) Reply Size Estimation

During the construction of each RPC Call message, a client is responsible for allocating appropriate resources for receiving the matching Reply message. The resources required depends on the maximum reply size expected, whether DDP-eligible can be removed from the reply and the transport version being used. The ULB is responsible for defining how the maximum reply size is to be determined while the specification of the transport version being used is responsible for defining how this maximum affects the resources to be allocated. Because the responder may not be able to send the required response when these resources have not been allocated, reliable reply size estimation is necessary to allow successful interoperation.

In many cases the Upper Layer Protocol's XDR definition provides enough information to enable the client to make a reliable prediction of the maximum size of the expected Reply message. However, If there are variable-size data items in the result, the maximum size of the RPC Reply message can be reliably estimated in many cases:

- 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 has already cached the size of the whole object it is about to request (e.g., via a previous NFS GETATTR request).
- o The client specifies a reply size limit for the particular reply, as it does by setting the count field of READDIR request.

It is sometimes not possible to determine the maximum Reply message size based solely on the above criteria. Client implementers can choose to provide the largest possible Reply buffer in those cases, based on, for instance, the largest possible NFS READ or WRITE payload (which is negotiated at mount time).

There exist cases in which a client cannot be sure a priori

determination is fully reliable. Handling of such cases is discussed in [Section 3.2](#).

[3.2](#). Retry to Deal with Reply Size Mis-estimation

For some of the protocols discussed in this document, it is possible for a compliant responder to send a valid reply whose length exceeds the client's a priori estimate. In such cases, the client needs to expect an error indication that indicates the existence of the oversize reply. When this happens, the client can either terminate that RPC transaction, or retry it with a larger reply size estimate.

In the case of the NFSv4.0, the use of NFS COMPOUND operations raises the possibility of non-idempotent requests that combine a non-idempotent operation with an operation whose maximum reply size cannot be determined with certainty. This makes retrying the operation problematic. It should be noted that many operations normally considered non-idempotent (e.g. WRITE, SETATTR) are actually idempotent. Truly non-idempotent operations are quite unusual in COMPOUNDS that include operations with uncertain reply sizes.

Depending on the transport version used, the client's choices may be restricted as follows:

- o The client may be required to treat the error as permanent, with retry not allowed.
- o The client may be allowed to reissue the request with a larger reply estimate, unless it is a non-idempotent request. In this case, non-idempotent requests may not be retried and will result in errors being reported to the issuer in this case.
- o The client may be allowed to reissue the request with a larger reply estimate, in essentially all cases. In this case, the client has sufficient information to avoid re-executing a non-idempotent request and may, if it chooses, retry all requests with a larger reply size.

In the case of Version One, the absence of a distinct error code to signal a reply chunk of inadequate size means that retry in this situation is not available.

[4.](#) Upper Layer Binding for NFS Versions 2 And 3

This Upper Layer Binding specification applies to NFS Version 2 [[RFC1094](#)] and NFS Version 3 [[RFC1813](#)]. For brevity, in this section a "legacy NFS client" refers to an NFS client using NFS version 2 or NFS version 3 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 legacy NFS client determines the maximum reply size for each operation using the basic criteria outlined in [Section 3.1](#). Such clients deal with reply sizes beyond the maximum as escribed in [Section 2.5](#).

[4.1.](#) Auxiliary Protocols

NFS versions 2 and 3 are typically deployed with several other protocols, sometimes referred to as "NFS auxiliary protocols." These are separate RPC programs that define procedures which are not part of the NFS version 2 or version 3 RPC programs. These include:

- o The MOUNT and NLM protocols, introduced in an appendix of [[RFC1813](#)]
- o The NSM protocol, described in Chapter 11 of [[NSM](#)]
- o The NFSACL protocol, which does not have a public definition (NFSACL here is treated as a de facto standard as there are several interoperating implementations).

RPC-over-RDMA treats these programs as distinct Upper Layer Protocols [[RFC8166](#)]. To enable the use of these ULPs on an RPC-over-RDMA

transport, an Upper Layer Binding specification is provided here for each.

[4.1.1.](#) MOUNT, NLM, And NSM Protocols

Typically MOUNT, NLM, and NSM are conveyed via TCP, even in deployments where NFS operations on RPC-over-RDMA. When a legacy server supports these programs on RPC-over-RDMA, it advertises the port address via the usual rpcbind service [[RFC1833](#)].

No operation in these protocols conveys a significant data payload, and the size of RPC messages in these protocols is uniformly small. Therefore, no XDR data items in these protocols are DDP-eligible. The largest variable-length XDR data item is an xdr_netobj. In most implementations this data item is not larger than 1024 bytes, making this size a reasonable basis for reply size estimation. However, since this limit is not specified as part of the protocol, the techniques described in [Section 3.1](#) should be used to deal with situations where these sizes are exceeded.

[4.1.2.](#) NFSACL Protocol

Legacy clients and servers that support the NFSACL RPC program typically convey NFSACL procedures on the same connection as the NFS RPC program. 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 program's XDR definition.

There is no protocol-wide size limit for NFS version 3 ACLs, and there is no mechanism in either the NFSACL or NFS programs for a legacy client to ascertain the largest ACL a legacy server can store. Legacy client implementations should choose a maximum size for ACLs based on their own internal limits. A recommended lower bound for this maximum is 32,768 bytes, though a larger Reply chunk (up to the negotiated rsize setting) can be provided. Since no limit is

specified as part of the protocol, the techniques described in [Section 3.1](#) should be used to deal with situations where these recommended bounds are exceeded.

[5.](#) Upper Layer Binding for NFS Version 4

This Upper Layer Binding specification applies to all protocols defined in NFS Version 4.0 [[RFC7530](#)], NFS Version 4.1 [[RFC5661](#)], and NFS Version 4.2 [[RFC7862](#)].

Noveck

Expires January 26, 2018

[Page 10]

Internet-Draft

Transport-generic NFS ULBs

July 2017

[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
- o In minor version 2 and newer, the rpc_data field of the read_plus_content union (further restrictions on the use of this data item follow below).

[5.1.1.](#) READ_PLUS Replies

The NFS version 4.2 READ_PLUS operation returns a complex data type [[RFC7862](#)]. The rpr_contents field in the result of this operation is an array of read_plus_content unions, one arm of which contains an opaque byte stream (d_data).

The size of d_data is limited to the value of the rpa_count field, but the protocol does not bound the number of elements which can be returned in the rpr_contents array. In order to make the size of READ_PLUS replies predictable by NFS version 4.2 clients, the following restrictions are placed on the use of the READ_PLUS operation on RPC-over-RDMA transports:

- o An NFS version 4.2 client MUST NOT provide more than one Write

chunk for any READ_PLUS operation. When providing a Write chunk for a READ_PLUS operation, an NFS version 4.2 client MUST provide a Write chunk that is either empty (which forces all result data items for this operation to be returned inline) or large enough to receive rpa_count bytes in a single element of the rpr_contents array.

- o If the Write chunk provided for a READ_PLUS operation by an NFS version 4.2 client is not empty, an NFS version 4.2 server MUST use that chunk for the first element of the rpr_contents array that has an rpc_data arm.
- o An NFS version 4.2 server MUST NOT return more than two elements in the rpr_contents array of any READ_PLUS operation. It returns as much of the requested byte range as it can fit within these two elements. If the NFS version 4.2 server has not asserted rpr_eof

in the reply, the NFS version 4.2 client SHOULD send additional READ_PLUS requests for any remaining bytes.

[5.2.](#) NFS Version 4 Reply Size Estimation

An NFS version 4 client provides a Reply chunk when the maximum possible reply size is larger than the client's responder inline threshold.

There are certain NFS version 4 data items whose size cannot be estimated by clients reliably, however, because there is no protocol-specified size limit on these structures. 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 Opaque fields 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 items enumerated above in [Section 5.2](#) make it difficult to predict the maximum size of GETATTR replies that interrogate variable-length attributes. As discussed in [Section 3.1](#), client implementations can rely on their own internal architectural limits to bound the reply size. However, since such limits are not guaranteed to be reliable, use of the techniques discussed in [Section 3.2](#) may sometimes be necessary.

It is best to avoid issuing single COMPOUNDS that contain both non-idempotent operations and operations where the maximum reply size cannot be reliably predicted.

[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 a replying NFS version 4 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

Noveck	Expires January 26, 2018	[Page 12]
--------	--------------------------	-----------

Internet-Draft	Transport-generic NFS ULBs	July 2017
----------------	----------------------------	-----------

ACLs, named attributes, layout bodies, and security labels are much smaller than this maximum.

[5.3](#). NFS Version 4 COMPOUND Requests

The NFS version 4 COMPOUND procedure allows the transmission of more than one DDP-eligible data item per Call and Reply message. 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.

The mechanism specified in [Section 4.3.2 of \[RFC8166\]](#) is applied here, with additional restrictions that appear below. In the following list, an "NFS Read" operation refers to any NFS Version 4 operation which has a DDP-eligible result data item (i.e., either a `READ`, `READ_PLUS`, or `READLINK` operation).

- 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 an 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.

[5.3.1.](#) 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



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.

[5.4.](#) NFS Version 4 Callback

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

[5.4.1.](#) 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 a RPC-over-RDMA 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.

[5.4.2.](#) 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 mechanism described in [[RFC8167](#)] when backchannel operations are conveyed on RPC-over-RDMA 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 reports a backchannel `ca_maxrequestsize` that is larger than the connection's inline thresholds, the NFS version 4 client can support Long Calls. Otherwise an NFS version 4 server MUST use Short messages to convey backchannel operations.

[5.5](#). Session-Related Considerations

Typically, the presence of an NFS session [[RFC5661](#)] has no effect on the operation of RPC-over-RDMA. None of the operations introduced to support NFS sessions 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 some rare error conditions which require special handling when an NFS session is operating on an RPC-over-RDMA transport. 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. Within RPC-over-RDMA Version One, this class of error can be generated for two different reasons:

- o There was an XDR error detected parsing the RPC-over-RDMA headers.
- o There was an error sending the response, because, for example, a necessary reply chunk was not provided or the one provided is of insufficient length.

These two situations, which arise due to incorrect implementations or underestimation of reply size, have different implications with regard to Exactly-Once Semantics. An XDR error in decoding the request precludes the execution of the request on the responder, but failure to send a reply indicates that some or all of the operations were executed.

In both instances, the client SHOULD NOT retry the operation without addressing reply resource inadequacy. Such a retry can result in the same sort of error seen previously. Instead, it is best to consider the operation as completed unsuccessfully and report an error to the consumer who requested the RPC.

In addition, within the error response, the requester does not have the result of the execution of the SEQUENCE operation, which identifies the session, slot, and sequence id for the request which has failed. The xid associated with the request, obtained from the `rdma_xid` field of the `RDMA_ERROR` or `RDMA_MSG` message, must be used to determine the session and slot for the request which failed, and the slot must be properly retired. If this is not done, the slot could be rendered permanently unavailable.

[5.6.](#) Connection 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.

[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 transports [[RFC8166](#)]. 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 [[RFC8178](#)].

7. IANA Considerations

NFS use of direct data placement introduces a need for an additional NFS port number assignment for networks that share traditional UDP and TCP port spaces with RDMA services. The iWARP [[RFC5041](#)] [[RFC5040](#)] protocol is such an example (InfiniBand is not).

NFS servers for versions 2 and 3 [[RFC1094](#)] [[RFC1813](#)] traditionally listen for clients on UDP and TCP port 2049, and additionally, they register these with the portmapper and/or rpcbind [[RFC1833](#)] service. However, [[RFC7530](#)] requires NFS version 4 servers to listen on TCP port 2049, and they are not required to register.

An NFS version 2 or version 3 server supporting RPC-over-RDMA 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. The chosen port MAY be registered with the RPC portmapper under the netid assigned by the requirement in [[RFC8166](#)].

An NFS version 4 server supporting RPC-over-RDMA on such a network MUST use the alternative well-known port number for its RPC-over-RDMA service. Clients SHOULD connect to this well-known port without consulting the RPC portmapper (as for NFS version 4 on TCP transports).

The port number assigned to an NFS service over an RPC-over-RDMA transport is available from the IANA port registry [[RFC3232](#)].

8. Security Considerations

RPC-over-RDMA supports all RPC security models, including RPCSEC_GSS security and transport-level security [[RFC2203](#)]. The choice of RDMA Read and RDMA Write to convey RPC argument and results does not affect this, since it changes only the method of data transfer. Specifically, the requirements of [[RFC8166](#)] ensure that this choice does not introduce new vulnerabilities.

Because this document defines only the binding of the NFS protocols atop [[RFC8166](#)], all relevant security considerations are therefore to

be described at that layer.

9. References

Noveck

Expires January 26, 2018

[Page 17]

Internet-Draft

Transport-generic NFS ULBs

July 2017

9.1. Normative References

- [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>>.
- [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>>.
- [RFC8166] Lever, C., Ed., Simpson, W., and T. Talpey, "Remote Direct Memory Access Transport for Remote Procedure Call Version 1", [RFC 8166](#), DOI 10.17487/RFC8166, June 2017, <<http://www.rfc-editor.org/info/rfc8166>>.

- [RFC8167] Lever, C., "Bidirectional Remote Procedure Call on RPC-over-RDMA Transports", [RFC 8167](#), DOI 10.17487/RFC8167, June 2017, <<http://www.rfc-editor.org/info/rfc8167>>.
- [RFC8178] Noveck, D., "Rules for NFSv4 Extensions and Minor Versions", [RFC 8178](#), DOI 10.17487/RFC8178, July 2017, <<http://www.rfc-editor.org/info/rfc8178>>.

9.2. Informative References

- [I-D.ietf-nfsv4-rfc5667bis]
Lever, C., "Remote Direct Memory Access Transport for Remote Procedure Call, Version One", [draft-ietf-nfsv4-rfc5667bis-11](#) (work in progress), May 2017.

Noveck	Expires January 26, 2018	[Page 18]
--------	--------------------------	-----------

Internet-Draft	Transport-generic NFS ULBs	July 2017
----------------	----------------------------	-----------

- [NSM] The Open Group, "Protocols for Interworking: XNFS, Version 3W", February 1998.
- [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>>.
- [RFC3232] Reynolds, J., Ed., "Assigned Numbers: [RFC 1700](#) is Replaced by an On-line Database", [RFC 3232](#), DOI 10.17487/RFC3232, January 2002, <<http://www.rfc-editor.org/info/rfc3232>>.
- [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>>.

[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>>.

[rpcrdmav2]

Lever, C., Ed. and D. Noveck, "RPC-over-RDMA Version Two", July 2017, <<http://www.ietf.org/id/draft-cel-nfsv4-rpcrdma-version-two-05.txt>>.

Work in progress.

[Appendix A](#). Acknowledgments

The author gratefully acknowledges the work of Brent Callaghan and Tom Talpey on the original NFS Direct Data Placement specification [[RFC5667](#)].

A large part of the material in this document is taken from [[I-D.ietf-nfsv4-rfc5667bis](#)] written by Chuck Lever. The author wishes to acknowledge the debt he owes to Chuck for his work in providing an updated Upper Layer Binding for the NFS-related protocols.

Noveck

Expires January 26, 2018

[Page 19]

Internet-Draft

Transport-generic NFS ULBs

July 2017

The author also wishes to thank Bill Baker and Greg Marsden for their support of the work to revive RPC-over-RDMA.

Author's Address

David Noveck
1601 Trapelo Road
waltham, MA 02451
United States of America

Phone: +1 781 572 8038
Email: davenoveck@gmail.com

