Network File System (NFS) Upper-Layer Binding To RPC-Over-RDMA Version 2

Abstract

This document specifies Upper-Layer Bindings of Network File System
(NFS) protocol versions to RPC-over-RDMA version 2.

Discussion of this draft takes place on the NFSv4 working group
mailing list (nfsv4@ietf.org), which is archived at https://
mailarchive.ietf.org/arch/browse/nfsv4/. Working Group information
can be found at https://datatracker.ietf.org/wg/nfsv4/about/.

This note is to be removed before publishing as an RFC.

The source for this draft is maintained in GitHub. Suggested changes
can be submitted as pull requests at https://github.com/chucklever/
i-d-nfs-ulb-v2. Instructions are on that page as well.

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 https://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 9 April 2021.

Copyright Notice

Table of Contents

1. **Introduction**

   The RPC-over-RDMA version 2 transport may employ direct data
   placement to convey data payloads associated with RPC transactions,
   as described in [I-D.ietf-nfsv4-rpcrdma-version-two]. RPC client and
   server implementations using RPC-over-RDMA version 2 must agree
   which XDR data items and RPC procedures are eligible to use direct
   data placement (DDP) to ensure successful interoperation.

   An Upper-Layer Binding specifies this agreement for one or more
   versions of 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 such a Binding.

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

   *NFS version 2 [RFC1094]

   *NFS version 3 [RFC1813]

   *NFS version 4.0 [RFC7530]

   *NFS version 4.1 [RFC8881]

   *NFS version 4.2 [RFC7862]

The current document also provides Upper-Layer Bindings for auxiliary protocols used with NFS versions 2 and 3 (see Section 4).

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

2. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 version 2 or version 3 of the NFS RPC program (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:

   *The opaque file data argument in the NFS WRITE procedure

   *The pathname argument in the NFS SYMLINK procedure

   *The opaque file data result in the NFS READ procedure

*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.

Whether or not an NFS operation is considered non-idempotent, a
transport error might not indicate 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

During the construction of each RPC Call message, a Requester is
responsible for allocating appropriate transport resources to
receive the corresponding Reply message. These resources must be
capable of holding the entire Reply, therefore the Requester needs
to estimate the maximum possible size of the expected Reply message.

   *In many cases, the expected Reply can fit in one or a few RDMA
    Send messages. The Requester need not provision any RDMA
    resources, relying instead on message continuation to handle the
    entire Reply message.

   *In cases where the Requester deems direct data placement to be
    the most efficient transfer mechanism, it provisions Write chunks
    wherein the Responder can place results. In these cases, the
    Requester must reliably estimate the maximum size of each result
    that is to be placed in a Write chunk.

   *When the Requester expects an especially large Reply message, it
    can provision a combination of a Reply chunk and Write chunks for
    result data items. In such cases, the Requester must reliably
    estimate the maximum size of each result that is to be placed in
    a Write chunk and the maximum size of the remainder to be placed
    in the Reply chunk.

A legacy NFS client needs to make every effort to avoid
retransmission of non-idempotent NFS requests due to underestimated
Reply resources. Thanks to the mechanism of message continuation in
RPC-over-RDMA version 2, the need for such retransmission is greatly
reduced.

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 service [RFC1833].

A Legacy NFS server supporting RPC-over-RDMA version 2 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 using the netids assigned in [Section 12](#) of [[I-D.ietf-nfsv4-rpcrdma-version-two](#)].

3.3. Transport Considerations

Legacy NFS client implementations often rely on a transport-layer keep-alive mechanism to detect when a legacy 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 retransmission of outstanding RPC transactions.

3.3.1. Keep-Alive

Some RDMA transports (such as the Reliable Connected QP type 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 or unreachable. Once an NFS client has consumed all available RPC-over-RDMA version 2 credits on that transport connection, it awaits a reply indefinitely before sending another RPC request.

Legacy NFS clients **SHOULD** reserve one RPC-over-RDMA version 2 credit to use for periodic server or connection health assessment. Either peer can use this credit to drive an RPC request on an otherwise idle connection, triggering either an affirmative server response or a connection termination.

3.3.2. Replay Detection

Legacy NFS servers typically employ request replay detection to reduce the risk of data and file namespace corruption that could result when an NFS client retransmits a non-idempotent NFS request. A legacy NFS server can send a cached response when a replay is detected, rather than executing the request again. Replay detection is not perfect, but it is usually adequate.

For legacy NFS servers, replay detection commonly utilizes heuristic indicators such as the IP address of the NFS client, the source port of the connection, the transaction ID of the request, and the contents of the request's RPC and upper-layer protocol headers. In short, replay detection is typically based on a connection tuple and the request's XID. A legacy NFS client is careful to re-use the same source port, if practical, when reconnecting so that legacy NFS servers are better able to detect retransmissions.

However, a legacy NFS client operating over an RDMA transport has no control over connection source ports. It is almost certain that an RPC request that is retransmitted on a new connection can never be

detected as a replay if the legacy NFS server includes the
connection source port in its replay detection heuristics.

Therefore a legacy NFS server using an RDMA transport should never
use a legacy NFS client connection's source port as part of its NFS
request replay detection mechanism.

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

Storage administrators typically deploy NFS versions 2 and 3 with
several other protocols, sometimes referred to as the "NFS auxiliary
protocols." These are distinct RPC programs that define procedures
that are not part of the NFS RPC program (100003). The Upper-Layer
Bindings in this section apply to:

  *Versions 2 and 3 of the MOUNT RPC program (100005) [RFC1813]

  *Versions 1, 3, and 4 of the NLM RPC program (100021) [RFC1813]

  *Version 1 of the NSM RPC program (100024), described in Chapter
   11 of [XNFS]

  *Versions 2 and 3 of the NFSACL RPC program (100227). The NFSACL
   program does not have a public definition. In this document it is
   treated 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. A legacy NFS server
implementation **MUST** provide support for these protocols via TCP to
enable interoperation of these protocols when NFS/RDMA is in use.

4.2. **NFSACL Protocol**

Often legacy clients and servers that support the NFSACL RPC program
convey NFSACL procedures on the same transport connection and port
as the NFS RPC program (100003). Utilizing the same port obviates
the need for separate a rpcbind query to discover server support for
this RPC program.

ACLs are typically small, but even large ACLs must be encoded and
decoded to some degree before being made available to users. 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. However, legacy client implementations should choose a
maximum size for ACLs based on internal limits, and can rely on

message continuation to handle the a priori unknown size of large
ACL objects in Replies.

5. Upper-Layer Binding For NFS Version 4

The Upper-Layer Binding specification in this section applies to
versions of the NFS RPC program defined in NFS version 4.0 [RFC7530]
NFS version 4.1 [RFC8881] 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:

  *The opaque data field in the WRITE4args structure

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

  *The opaque data field in the READ4resok structure

  *The linkdata field in the READLINK4resok structure

5.1.1. The NFSv4.2 READ_PLUS operation

NFS version 4.2 introduces an enhanced READ operation called
READ_PLUS [RFC7862]. READ_PLUS enables an NFS server to perform data
reduction of READ results so that the returned READ data is more
compact.

In a READ_PLUS result, returned file content appears as a list of
one or more of the following items:

  *Regular data content: the same as the result of a traditional
   READ operation.

  *Unallocated space in a file: where no data has yet been written
   or previously-written data has been removed via a hole-punch
   operation.

  *A counted pattern.

Upon receipt of a READ_PLUS result, an NFSv4.2 client expands the
returned list into the preferred local representation of the
original file content.

Before receiving that result, an NFSv4.2 client typically does not
know how the file's content is organized on the NFS server. Thus it
is not possible to predict the size or structure of a READ_PLUS
Reply in advance. The use of direct data placement is therefore
challenging.

A READ_PLUS content list containing more than one segment of regular file data could be conveyed using multiple Write chunks, but only if the client knows in advance where those chunks appear in the Reply Payload stream. Moreover, the usual benefits of hardware-assisted data placement are entirely waived if the client-side transport must parse the result of each read I/O.

Therefore this Upper Layer Binding does not make any element of an NFSv4.2 READ_PLUS Reply DDP-eligible. Further, this Upper Layer Binding recommends that implementations avoid the use of the READ_PLUS operation on NFS/RDMA mount points.

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 result arrays. These include:

   *The attrlist4 field

   *Fields containing ACLs such as fattr4_acl, fattr4_dacl, and
    fattr4_sacl

   *Fields in the fs_locations4 and fs_locations_info4 data
    structures

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

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 upon internal architectural limits to constrain the reply size, but such limits are not always guaranteed to be reliable.

When an NFS version 4.0 client expects an especially sizeable fattr4 result, it can rely on message continuation or provision a Reply chunk to enable that server to return that result via explicit RDMA.

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.

An NFS version 4 client can use this value 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 are not required to register with an rpcbind service [RFC7530]. Therefore, an NFS version 4 server supporting RPC-over-RDMA version 2 **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 lists, a "READ operation" refers to any NFS version 4 operation that has a DDP-eligible result data item. An NFS version 4 client applies the mechanism specified in Section 4.3.2 of [I-D.ietf-nfsv4-rpcrdma-version-two] to this class of operations as follows:

  *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.

An NFS version 4 server acts as follows:

  *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.

*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.

*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.

*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.

*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. Chunk List Complexity

By default, the RPC-over-RDMA version 2 protocol places limits on
the number of chunks or segments that may appear in Read or Write
lists (see [Section 5.2](#) of [[I-D.ietf-nfsv4-rpcrdma-version-two](#)]).

These implementation limits are especially important when Kerberos
integrity or privacy is in use [[RFC7861](#)]. GSS services increase the
size of credential material in RPC headers, potentially requiring
the use of a Long message, which increases the complexity of chunk
lists independent of the particular NFS version 4 COMPOUND being
conveyed.

In the absence of an explicit transport property exchange that
alters these limits, NFS version 4 clients **SHOULD** follow the
prescriptions listed below when constructing RPC-over-RDMA version 2
messages. NFS version 4 servers **MUST** accept and process all such
requests.

  *The Read list can contain either a Position-Zero Read chunk, one
   Read chunk with a non-zero Position, or both.

  *The Write list can contain no more than one Write chunk.

NFS version 4 clients wishing to send more complex chunk lists can
provide configuration interfaces to bound the complexity of NFS
version 4 COMPOUNDs, limit the number of elements in scatter-gather
operations, and avoid other sources of chunk overruns at the
receiving peer.

If an NFS version 4 server receives an RPC request via RPC-over-RDMA
version 2 that it cannot process due to chunk list complexity

limits, it **SHOULD** return one of the following responses to the
client:

  *A problem is detected by the transport layer while parsing the
   transport header in an RPC Call message. The server responds with
   an RDMA2_ERROR message with the err field set to ERR_CHUNK.

  *A problem is detected during XDR decoding of the RPC Call message
   while the RPC layer reassembles the call's XDR stream. The server
   responds with an RPC reply with its "reply_stat" field set to
   MSG_ACCEPTED and its "accept_stat" field set to GARBAGE_ARGS.

After receiving one of these errors, an NFS version 4 client **SHOULD
NOT** retransmit the failing request, as the result would be the same
error. It **SHOULD** terminate the RPC transaction associated with the
XID in the reply without further processing, and report an error to
the RPC consumer.

5.4.3. 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

An NFS version 4.0 client uses the SETCLIENTID operation to advertise the IP address, port, and netid of its NFS version 4.0 callback service. When an NFS version 4.0 server provides a backchannel service to an NFS version 4.0 client that uses RPC-over-RDMA version 2 for its forward channel, the server **MUST** advertise the backchannel service using either the "tcp" or "tcp6" netid.

Because the backchannel does not operate on RPC-over-RDMA, no XDR data item in the NFS version 4.0 callback RPC program is DDP-eligible.

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 that is in use for NFS version 4 forward channel client requests. NFS version 4 clients and servers **MUST** use the mechanisms described in [Section 4.5](#) of [[I-D.ietf-nfsv4-rpcrdma-version-two](#)] to convey backchannel operations on an RPC-over-RDMA version 2 transport.

The csa_back_chan_attrs argument of the CREATE_SESSION operation contains a ca_maxresponsesize field. The value in this field is 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 sizeable. A sender can use Message Continuation or a Long message in this situation.

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 version 4 session (as defined in [[RFC8881](#)]) does not effect the operation of RPC-over-RDMA version 2. 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 version 2 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 RDMA2_ERROR message with a rdma_err value of ERR_CHUNK. These situations are not different from existing RPC errors, which an NFS session implementation can already handle for other transport types. Moreover, there might be no SEQUENCE result available to the Requester to distinguish whether failure occurred before or after the Responder executed the requested operations.

When a transport error occurs (e.g., an RDMA2_ERROR type message is received), the Requester proceeds, as usual, to match the incoming XID value to a waiting RPC Call. The Requester terminates the RPC transaction and reports the result status to the RPC consumer. 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. Otherwise, that slot could be rendered permanently unavailable.

When an NFS session is not present (for example, when NFS version 4.0 is in use), a transport error does not indicate 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.

5.7. Transport Considerations

5.7.1. Congestion Avoidance

Section 3.1 of [RFC7530] states:

> Where an NFS version 4 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 [RFC8881] further states:

> Even if NFS version 4.1 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 NFS version 4.1; 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 NFS version 4.1 transport.

RPC-over-RDMA version 2 utilizes only reliable, connection-oriented transports that guarantee in-order delivery, meeting all the above requirements for NFS version 4.0 and 4.1. See Section 4.2.1 of [I-D.ietf-nfsv4-rpcrdma-version-two] for more details.

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 the Reliable Connected QP type 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 version 2 credits on that transport connection, it indefinitely awaits a reply before sending another RPC request.

NFS version 4 clients **SHOULD** reserve one RPC-over-RDMA version 2 credit to use for periodic server or connection health assessment. Either peer can use this credit 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 version 2 transport connections can be terminated when a Transport header is malformed, Reply messages exceed receive resources, or when too many RPC-over-RDMA messages are sent at once. In such cases:

  *If a transport error occurs (e.g., an RDMA2_ERROR type message is received) 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.

  *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 2 transports [I-D.ietf-nfsv4-rpcrdma-version-two]. 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 to cover separately published extensions to an existing NFS version 4 minor version, as described in [RFC8178].

7. Security Considerations

RPC-over-RDMA version 2 supports all RPC security models, including RPCSEC_GSS security and transport-level security [RFC7861]. 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. Because the current document defines only the binding of the NFS protocols atop RPC-over-RDMA version 2 [I-D.ietf-nfsv4-rpcrdma-version-two], all relevant security considerations are, therefore, 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 [RFC5040] [RFC5041].

For this purpose, the current document specifies a set of transport protocol port number assignments. 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
```

The current document should be added as a reference for the nfsrdma port assignments. The current document does not alter these assignments.

9. References

9.1. Normative References

[I-D.ietf-nfsv4-rpcrdma-version-two]
          Lever, C. and D. Noveck, "RPC-over-RDMA Version 2
          Protocol", Work in Progress, Internet-Draft, draft-ietf-
          nfsv4-rpcrdma-version-two-03, 10 August 2020, <https://
          tools.ietf.org/html/draft-ietf-nfsv4-rpcrdma-version-
          two-03>.

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version
          2", RFC 1833, DOI 10.17487/RFC1833, August 1995,
          <https://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, <https://www.rfc-editor.org/info/
             rfc2119>.

[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, <https://
             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, <https://www.rfc-editor.org/info/
             rfc7530>.

[RFC7861]   Adamson, A. and N. Williams, "Remote Procedure Call (RPC)
             Security Version 3", RFC 7861, DOI 10.17487/RFC7861,
             November 2016, <https://www.rfc-editor.org/info/rfc7861>.

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

[RFC8174]   Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
             2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
             May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8881]   Noveck, D., Ed. and C. Lever, "Network File System (NFS)
             Version 4 Minor Version 1 Protocol", RFC 8881, DOI
             10.17487/RFC8881, August 2020, <https://www.rfc-
             editor.org/info/rfc8881>.

9.2. Informative References

[RFC1094]   Nowicki, B., "NFS: Network File System Protocol
             specification", RFC 1094, DOI 10.17487/RFC1094, March
             1989, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc5041>.

[RFC8178] Noveck, D., "Rules for NFSv4 Extensions and Minor Versions", RFC 8178, DOI 10.17487/RFC8178, July 2017, <https://www.rfc-editor.org/info/rfc8178>.

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

Acknowledgments

Author's Address

Charles Lever
Oracle Corporation
United States of America

Email: chuck.lever@oracle.com