

Network File System Version 4
Internet-Draft
Intended status: Informational
Expires: August 26, 2018

D. Noveck
NetApp
February 22, 2018

Issues Related to RPC-over-RDMA Internode Round Trips
draft-dnoveck-nfsv4-rpcrdma-rtissues-05

Abstract

As currently designed and implemented, the RPC-over-RDMA protocol requires use of multiple internode round trips to process some common operations. For example, NFS WRITE operations require use of three internode round trips. This document looks at this issue and discusses what can and what should be done to address it, both within the context of an extensible version of RPC-over-RDMA and potentially outside that framework.

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 August 26, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://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

Internet-Draft

RPC/RDMA Round-trip Issues

February 2018

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Preliminaries	2
1.1.	Requirements Language	2
1.2.	Introduction	2
2.	Review of the Current Situation	3
2.1.	Potentially Troublesome Requests	3
2.2.	WRITE Request Processing Details	4
2.3.	READ Request Processing Details	5
3.	Near-term Work	6
3.1.	Target Performance	7
3.2.	Message Continuation	8
3.3.	Send-based Data Placement	8
3.4.	Feature Synergy	9
3.5.	Feature Selection and Negotiation	10
4.	Possible Future Development of RPC-over-RDMA	12
5.	Other Possible Approaches	13
6.	Summary	13
7.	Security Considerations	15
8.	IANA Considerations	15
9.	References	15
9.1.	Normative References	15
9.2.	Informative References	15
Appendix A.	Acknowledgements	16
	Author's Address	16

[1.](#) Preliminaries

[1.1.](#) 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](#)].

[1.2.](#) Introduction

When many common operations are performed using RPC-over-RDMA, additional inter-node round-trip latencies are required to take advantage of the performance benefits provided by RDMA Functionality.

While the latencies involved are generally small, they are a reason for concern for two reasons.

- o With the ongoing improvement of persistent memory technologies, such internode latencies, being fixed, can be expected to consume

an increasing portion of the total latency required for processing NFS requests using RPC-over-RDMA.

- o High-performance transfers using NFS may be needed outside of a machine-room environment. As RPC-over-RDMA is used in networks of campus and metropolitan scale, the internode round-trip time of sixteen microseconds per mile becomes an issue.

Given this background, round trips beyond the minimum necessary need to be justified by corresponding benefits. If they are not, work needs to be done to eliminate those excess round trips.

We are going to look at the existing situation with regard to round-trip latency and make some suggestions as to how the issue might be best addressed. We will consider things that could be done in the near future and also explore further possibilities that would require a longer-term approach to be adopted.

[2.](#) Review of the Current Situation

[2.1.](#) Potentially Troublesome Requests

We will be looking at four sorts of situations:

- o An RPC operation involving Direct Data Placement of request data (e.g., an NFSv3 WRITE or corresponding NFSv4 COMPOUND).
- o An RPC operation involving Direct Data Placement of response data (e.g., an NFSv3 READ or corresponding NFSv4 COMPOUND).
- o An RPC operation where the request data is longer than the inline buffer limit.
- o An RPC operation where the response data is longer than the inline buffer limit.

These are all simple examples of situations in which explicit RDMA operations are used, either to effect Direct Data Placement or to respond to message size limits that derive from a limited receive buffer size.

We will survey the resulting latency and overhead issues in an RPC-over-RDMA Version One environment in Sections [2.2](#) and [2.3](#) below.

[2.2](#). WRITE Request Processing Details

We'll start with the case of a request involving direct placement of request data. In this case, an RDMA READ is used to transfer a DDR-eligible data item (e.g. the data to be written) from its location in requester memory, to a location selected by the responder.

Processing proceeds as described below. Although we are focused on internode latency, the time to perform a request also includes such things as interrupt latency, overhead involved in interacting with the RNIC, and the time for the server to execute the requested operation.

- o First, the memory to be accessed remotely is registered. This is a local operation.
- o Once the registration has been done, the initial send of the request can proceed. Since this is in the context of connected operation, there is an internode round trip involved. However, the next step can proceed after the initial transmission is received by the responder. As a result, only the responder-bound side of the transmission contributes to overall operation latency.
- o The responder, after being notified of the receipt of the request, uses RDMA READ to fetch the bulk data. This involves an internode round-trip latency. After the fetch of the data, the responder needs to be notified of the completion of the explicit RDMA operation

- o The responder (after performing the requested operation) sends the response. Again, as this is in the context of connected operation, there is an internode round trip involved. However, the next step can proceed after the initial transmission is received by the requester.
- o The memory registered before the request was issued needs to be deregistered, before the request is considered complete and the sending process restarted. When remote invalidation is not available, the requester, after being notified of the receipt of the response, performs a local operation to deregister the memory in question. Alternatively, the responder will use Send With Invalidate and the responder's RNIC will effect the deregistration before notifying the requester of the response which has been received.

To summarize, if we exclude the actual server execution of the request, the latency consists of two internode round-trip latencies plus two responder-side interrupt latencies plus one requester-side

interrupt latency plus any necessary registration/de-registration overhead. This is in contrast to a request not using explicit RDMA operations in which there is a single inter-node round-trip latency and one interrupt latency on the requester and the responder.

The processing of the other sorts of requests mentioned in [Section 2.1](#) show both similarities and differences:

- o Handling of a long request is similar to the above. The memory associated with a position-zero read chunk is registered, transferred using RDMA READ, and deregistered. As a result, we have the same overhead and latency issues noted in the case of direct data placement, without the corresponding benefits.
- o The case of direct data placement of response data follows a similar pattern. The important difference is that the transfer of the bulk data is performed using RDMA WRITE, rather than RDMA READ. However, because of the way that RDMA WRITE is effected over the wire, the latency consequences are different. See [Section 2.3](#) for a detailed discussion.
- o Handling of a long response is similar to the previous case.

[2.3.](#) READ Request Processing Details

We'll now discuss the case of a request involving direct placement of response data. In this case, an RDMA WRITE is used to transfer a DDR-eligible data item (e.g. the data being read) from its location in responder memory, to a location previously selected by the requester.

Processing proceeds as described below. Although we are focused on internode latency, the time to perform a request also includes such things as interrupt latency, overhead involved in interacting with the RNIC, and the time for the server to execute the requested operation.

- o First, the memory to be accessed remotely is registered. This is a local operation.
- o Once the registration has been done, the initial send of the request can proceed. Since this is in the context of connected operation, there is an internode round trip involved. However, the next step can proceed after the initial transmission is received. As a result, only the responder-bound side of the transmission contributes to overall operation latency.

- o The responder, after being notified of the receipt of the request, proceeds to process the request until the data to be read is available in its own memory, with its location determined and fixed. It then uses RDMA WRITE to transfer the bulk data to the location in requester memory selected previously. This involves an internode latency, but there is no round trip and thus no round-trip latency,
- o The responder continues processing and sends the inline portion of the response. Again, as this is in the context of connected operation, there is an internode round trip involved. However, the next step can proceed immediately. If the RDMA WRITE or the send of the inline portion of the response were to fail, the responder can be notified subsequently.

- o The requester, after being notified of the receipt of the response, can analyze it and can access the data written into its memory. Deregistration of the memory originally registered before the request was issued can be done using remote invalidation or can be done by the requester as a local operation

To summarize, in this case the additional latency that we saw in [Section 2.2](#) does not arise. Except for the additional overhead due to memory registration and invalidation, the situation is the same as for a request not using explicit RDMA operations in which there is a single inter-node round-trip latency and one interrupt latency on the requester and the responder.

[3.](#) Near-term Work

We are going to consider how the latency and overhead issues discussed in [Section 2](#) might be addressed in the context of an extensible version of RPC-over-RDMA, such as that proposed in [\[I-D.cel-nfsv4-rpcrdma-version-two\]](#).

In [Section 3.1](#), we will establish a performance target for the troublesome requests, based on the performance of requests that do not involve long messages or direct data placement.

We will then consider how extensions might be defined to bring latency and overhead for the requests discussed in [Section 2.1](#) into line with those for other requests. There will be two specific classes of requests to address:

- o Those that do not involve direct data placement will be addressed in [Section 3.2](#). In this case, there are no compensating benefits justifying the higher overhead and, in some cases, latency.

- o The more complicated case of requests that do involve direct data placement is discussed in [Section 3.3](#). In this case, direct data placement could serve as a compensating benefit, and the important question to be addressed is whether Direct Data Placement can be effected without the use of explicit RDMA operations.

The optional features to deal with each of the classes of messages discussed above could be implemented separately. However, in the

handling of RPCs with very large amounts of bulk data, the two features are synergistic. This fact makes it desirable to define the features as part of the same extension. See Sections [3.4](#) and [3.5](#) for details.

[3.1](#). Target Performance

As our target, we will look at the latency and overhead associated with other sorts of RPC requests, i.e. those that do not use data placement, and that have request and response messages which do fit within the receive buffer limit.

Processing proceeds as follows:

- o The initial send of the request is done. Since this is in the context of connected operation, there is an internode round-trip involved. However, the next step can proceed after the initial transmission is received. As a result, only the responder-bound side of the transmission contributes to overall operation latency.
- o The responder, after being notified of the receipt of the request, performs the requested operation and sends the reply. As in the case of the request, there is an internode round trip involved. However, the request can be considered complete upon receipt of the requester-bound transmission. The responder-bound acknowledgment does not contribute to request latency.

In this case, there is only a single internode round-trip latency necessary to effect the RPC. Total request latency includes this round-trip latency plus interrupt latency on the requester and responder, plus the time for the responder to actually perform the requested operation.

Thus the delta between the operations discussed in [Section 2](#) and our baseline consists of two portions, one of which applies to all the requests we are concerned with and the second of which only applies to request which involve use of RDMA READ, as discussed in [Section 2.2](#). The latter category consists of:

- o One additional internode round-trip latency.

- o One additional instance of responder-side interrupt latency.

The additional overhead necessary to do memory registration and deregistration applies to all requests using explicit RDMA operations. The costs will vary with implementation characteristics. As a result, in some implementations, it may be desirable to replace use of RDMA Write with send-based alternatives, while in others, use of RDMA Write may be preferable.

[3.2.](#) Message Continuation

Using multiple RPC-over-RDMA transmissions, in sequence, to send a single RPC message avoids the additional latency and overhead associated with the use of explicit RDMA operations to transfer position-zero read chunks. In the case of reply chunks, only overhead is reduced.

Although transfer of a single request or reply in N transmissions will involve $N+1$ internode latencies, overall request latency is not increased by requiring that operations involving multiple nodes be serialized. Generally, these transmissions are pipelined.

As an illustration, let's consider the case of a request involving a response consisting of two RPC-over-RDMA transmissions. Even though each of these transmissions is acknowledged, that acknowledgement does not contribute to request latency. The second transmission can be received by the requester and acted upon without waiting for either acknowledgment.

This situation would require multiple receive-side interrupts but it is unlikely to result in extended interrupt latency. With 1K sends (Version One), the second receive will complete about 200 nanoseconds after the first assuming a 40Gb/s transmission rate. Given likely interrupt latencies, the first interrupt routine would be able to note that the completion of the second receive had already occurred.

[3.3.](#) Send-based Data Placement

In order to effect proper placement of request or reply data within the context of individual RPC-over-RDMA transmissions, receive buffers need to be structured to accommodate this function

To illustrate the considerations that could lead clients and servers to choose particular buffer structures, we will use, as examples, the cases of NFS READs and WRITES of 8K data blocks (or the corresponding NFSv4 COMPOUNDS).

In such cases, the client and server need to have the DDP-eligible bulk data placed in appropriately aligned 8K buffer segments. Rather than being transferred in separate transmissions using explicit RDMA operations, a message can be sent so that bulk data is received into an appropriate buffer segment. In this case, it would be excised from the XDR payload stream, just as it is in the case of existing DDP facilities.

Consider a server expecting write requests which are usually X bytes long or less, exclusive of an 8K bulk data area. In this case the payload stream will most likely be less than X bytes and will fit in a buffer segment devoted to that purpose. The bulk data needs to be placed in the subsequent buffer segment in order to align it properly, i.e. with the appropriate alignment, in the data placement target buffer. In order to place the data appropriately, the sender (in this case, the client) needs to add padding of length $X-Y$ bytes where Y is the length of payload stream for the current request. The case of reads is exactly the same except that the sender adding the padding is the server.

To provide send-based data placement as an RPC-over-RDMA extension, the framework defined in [[I-D.cel-nfsv4-rpcrdma-version-two](#)] could be used. A new "transport characteristic" could be defined which allowed a participant to expose the structure of his receive buffers and to identify the buffer segments capable of being used as data placement targets. In addition, a new optional message header would have to be defined. It would be defined to provide:

- o A way to designate a DDP-eligible data item as corresponding to target buffer segments, rather than memory registered for RDMA.
- o A way to indicate to the responder that it should place DDP-eligible data items in DDP-targetable buffer segments, rather than in memory registered for RDMA.
- o A way to designate a limited portion of an RPC-over-RDMA transmission as constituting the payload stream.

[3.4.](#) Feature Synergy

While message continuation and send-based data placement each address an important class of commonly used messages, their combination allows simpler handling of some important classes of messages:

- o READs and WRITEs transferring larger IOs

- o COMPOUNDS containing multiple IO operations.

- o Operations whose associated payload stream is longer than the typical value.

To accommodate these situations, it would be best to have the definition of the headers to support message continuation interact with data structures to support send-based data placement as follows:

- o The header type used for the initial transmission of a message continued across multiple transmissions would contain placement-directing structures which support both send-based data placement as well as DDP using Explicit RDMA operations.
- o Buffer references for Send-based data placement should be relative to the start of the group of transmissions and should allow transitions between buffer segments in different receive buffers.
- o The header type for messages continuing a group of transmissions should not have DDP-related fields but should rely on the initial transmission of the group for DDP-related functions.
- o The portion of each received transmission devoted to the payload stream should be part of the header for each message within a group of transmissions devoted to a single RPC message. The payload stream for the message as a whole should be the concatenation of the streams for each transmission.

A potential extension supporting these features interacting as described above can be found in [[I-D.dnoveck-nfsv4-rpcrdma-rtrext](#)].

[3.5.](#) Feature Selection and Negotiation

Given that an appropriate extension is likely to support multiple OPTIONAL features, special attention will have to be given to defining how implementations which might not support the same subset of OPTIONAL features can successfully interact. The goal is to allow interacting implementations to get the benefit of features that they both support, while allowing implementation pairs that do not share support for any of the OPTIONAL features to operate just as base Version Two implementations could do in the absence of the potential

extension.

It is helpful if each implementation provides characteristics defining its level of feature support which the peer implementation can test before attempting to use a particular feature. In other similar contexts, the support level concerns the implementation in its role as responder, i.e. whether it is prepared to execute a given request. In the case of the potential extension discussed here, most

characteristics concern an implementation in its role as receiver. One might define characteristics which indicate,

- o The ability of the implementation, in its role as receiver, to process messages continued across multiple RPC-over-RDMA transmissions.
- o The ability of the implementation, in its role as receiver, to process messages containing DDP-eligible data items, placed using a send-based data placement approach.

Use of such characteristics might allow asymmetric implementations. For example, a client might send requests containing DDP-eligible data items using send-based data placement without being able to accept messages containing data items using send-based data placement. That is a likely implementation pattern, given the greater performance benefits of avoiding use of RDMA Read.

Further useful characteristics would apply to the implementation in its role of responder. For instance,

- o The ability of the implementation, in its role as responder, to accept and process requests which REQUIRE that DDP-eligible data items in the response be sent using send-based DDP. The presence of this characteristic would allow a requester to avoid registering memory to be used to accommodate DDP-eligible data items in the response.
- o The ability of the implementation, in its role as responder, to send responses using message continuation, as opposed to using a reply chunk.

Because of the potentially different needs of operations in the forward and backward directions, it may be desirable to separate the receiver-based characteristics according the direction of operation that they apply to.

A further issue relates to the role of explicit RDMA operations in connection with backwards operation. Although, no current protocols require support for DDP or transfer of large messages when operating in the backward direction, the protocol is designed to allow such support to be developed in the future. Since the protocol, with the extension discussed here is likely to have multiple methods of providing these functions, we have a number of possible choices regarding the role of chunk-based methods of providing these functions

- o Support for chunk-based operation remains a REQUIREMENT for responders, and requesters always have the option of using it, regardless of the direction of operation.

Requesters could select alternatives to the use of explicit RDMA operations when these are supported by the responder

- o When operating in the forward direction, support for chunk-based operation remains a REQUIREMENT for responders (i.e. servers), and requesters (i.e. clients).

When operating in the backward direction, support for chunk-based is OPTIONAL for responders (i.e. clients) allowing requesters (i.e. servers) to select use of explicit RDMA operations or alternatives when each of these is supported by the requester.

- o Support for chunk-based operation is treated as OPTIONAL for responders, regardless of the direction of operation.

In this case, requesters would select use of explicit RDMA operations or alternatives when each of these is supported by the responder. For a considerable time, support for explicit RDMA operations would be a practical necessity, even if not a REQUIREMENT, for operation in the forward direction.

4. Possible Future Development of RPC-over-RDMA

Although the reduction of explicit RDMA operation reduces the number of inter-node round trips and eliminates sequences of operations in which multiple round-trip latencies are serialized with server interrupt latencies, the use of connected operations means that round-trip latencies will always be present, since each message is acknowledged.

One avenue that has been considered is use of unreliable-datagram (UD) transmission in environments where the "unreliable" transmission is sufficiently reliable that RPC replay can deal with a very low rate of message loss. For example, UD in Infiniband specifies a low enough rate of frame loss to make this a viable approach, particularly for use in supporting protocols such as NFSv4.1, that contain their own facilities to ensure exactly-once semantics.

With this sort of arrangement, request latency is still the same. However, since the acknowledgements are not serving any substantial function, it is tempting to consider removing them, as they do take up some transmission bandwidth, that might be used otherwise, if the protocol were to reach the goal of effectively using the underlying medium.

The size of such wasted transmission bandwidth depends on the average message size and many implementation considerations regarding how acknowledgments are done. In any case, given expected message sizes, the wasted transmission bandwidth will be very small.

When RPC messages are quite small, acknowledgments may be of concern. However, in that situation, a better response would be transfer multiple RPC messages within a single RPC-over-RDMA transmission.

When multiple RPC messages are combined into a single transmission, the overhead of interfacing with the RNIC, particularly the interrupt handling overhead, is amortized over multiple RPC messages.

Although this technique is quite outside the spirit of existing RPC-over-RDMA implementations, it appears possible to define new header types capable of supporting this sort of transmission, using the extension framework described in [[I-D.cel-nfsv4-rpcrdma-version-two](#)].

5. Other Possible Approaches

It is possible that the additional round-trips associated with writing data to the server might be addressed outside the context of RPC-over-RDMA, by avoiding use of the RDMA paradigm for such transfers.

One possibility that has been discussed is the use of an RDMA-based pNFS mapping type, in which areas in server memory are presented via RDMA-based layouts so that the client could obtain file data using RDMA Read and modify it using RDMA Write. In each case, only a single round-trip would be required to effect each transfer, assuming that the appropriate layouts have been obtained. Although some I-D's have been written presenting the outlines of this approach, none are currently active.

6. Summary

We've examined the issue of round-trip latency and concluded:

- o That the number of round trips per se is not as important as the contribution of any extra round trips to overall request latency.
- o That the latency issue can be addressed using the extension mechanism provided for in [[I-D.cel-nfsv4-rpcrdma-version-two](#)].
- o That in many cases in which latency is not an issue, there may be overhead issues that can be addressed using the same sorts of techniques as those useful in latency reduction, again using the

extension mechanism provided for in
[[I-D.cel-nfsv4-rpcrdma-version-two](#)].

As it seems that the features sketched out could put internode latencies and overhead for a large class of requests back to the baseline value for the RPC paradigm, more detailed definition of the required extension functionality is in order.

We've also looked at round trips at the physical level, in that acknowledgments are sent in circumstances where there is no obvious need for them. With regard to these, we have concluded:

- o That these acknowledgements do not contribute to request latency.
- o That while UD transmission can remove acknowledgements of limited value, the performance benefits are not sufficient to justify the disruption that this would entail.
- o That issues with transmission bandwidth overhead in a small-message environment are better addressed by combining multiple RPC messages in a single RPC-over-RDMA transmission. This is particularly so, because such a step is likely to reduce overhead in such environments as well

As the features described involve the use of alternatives to explicit RDMA operations, in performing direct data placement and in transferring messages that are larger than the receive buffer limit, it is appropriate to understand the role that such operations are expected to have once the extensions discussed in this document are fully specified and implemented.

It is important to note that these extensions are OPTIONAL and are expected to remain so, while support for explicit RDMA operations will remain an integral part of RPC-over-RDMA.

Given this framework, the degree to which explicit RDMA operations will be used will reflect future implementation choices and needs. While we have been focusing on cases in which other options might be more efficient in some cases, it worth looking also at the cases in which explicit RDMA operations are likely to remain preferable:

- o In some environments in which direct data placement to memory of a certain alignment does not meet application requirements and in which data needs to be read into a particular address on the client. Also, large physically contiguous buffers may be required in some environments. In these situations, send-based data placement is not an option.

- o Where large transfers are to be done, there will be limits to the capacity of send-based data placement to provide the required functionality, since the basic pattern using send/receive is to allocate a pool of memory to contain receive buffers in advance of

issuing requests. While this issue can be mitigated by use of message continuation, tying up large numbers of credits for a single request can cause difficult issues as well. As a result, send-based data placement may be restricted to IO's of limited size, although the specific limits will depend on the details of the specific implementation.

[7.](#) Security Considerations

This document does not raise any security issues.

[8.](#) IANA Considerations

This document does not require any actions by IANA.

[9.](#) References

[9.1.](#) Normative References

- [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>>.
- [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, <<https://www.rfc-editor.org/info/rfc8166>>.
- [RFC8267] Lever, C., "Network File System (NFS) Upper-Layer Binding to RPC-over-RDMA Version 1", [RFC 8267](#), DOI 10.17487/RFC8267, October 2017, <<https://www.rfc-editor.org/info/rfc8267>>.

[9.2.](#) Informative References

- [I-D.cel-nfsv4-rpcrdma-version-two]
Lever, C. and D. Noveck, "RPC-over-RDMA Version 2 Protocol", [draft-cel-nfsv4-rpcrdma-version-two-06](#) (work in progress), January 2018.

[I-D.dnoveck-nfsv4-rpcrdma-rtrext]

Noveck, D., "RPC-over-RDMA Extensions to Reduce Internode Round-trips", [draft-dnoveck-nfsv4-rpcrdma-rtrext-03](#) (work in progress), December 2017.

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

[Appendix A](#). Acknowledgements

The author gratefully acknowledges the work of Brent Callaghan and Tom Talpey producing the original RPC-over-RDMA Version One specification [[RFC5666](#)] and also Tom's work in helping to clarify that specification.

The author also wishes to thank Chuck Lever for his work reviving RDMA support for NFS in [[RFC8166](#)] and [[RFC8267](#)], for providing a path for incremental improvement of that support by his work on [[I-D.cel-nfsv4-rpcrdma-version-two](#)], and for helpful discussions regarding RPC-over-RDMA latency issues.

Author's Address

David Noveck
NetApp
1601 Trapelo Road
Waltham, MA 02451
United state of America

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

