NFSv4 C. Lever
Internet-Draft Oracle
Intended status: Experimental May 29, 2015
Expires: November 30, 2015

Size-Limited Bi-directional Remote Procedure Call On Remote Direct Memory Access Transports
draft-ietf-nfsv4-rpcrdma-bidirection-00

Abstract

Recent minor versions of NFSv4 work best when ONC RPC transports can send ONC RPC transactions in both directions. This document describes conventions that enable RPC-over-RDMA version 1 transport endpoints to interoperate when operation in both directions is necessary.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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. Scope Of This Document

This document describes a set of experimental conventions that apply to RPC-over-RDMA version 1, specified in [RFC5666]. When observed, these conventions enable RPC-over-RDMA version 1 endpoints to concurrently handle RPC transactions that flow from client to server and from server to client.

These conventions can be observed when using the existing the RPC-over-RDMA version 1 protocol definition. Therefore this document does not update [RFC5666].

The purpose of this document is to permit interoperable prototype implementations of bi-directional RPC-over-RDMA, enabling the use of NFSv4.1 (including pNFS and later NFSv4 minor versions) on RDMA transports.

Providing an Upper Layer Binding for NFSv4.x callback operations is outside the scope of this document.

1.3. Understanding RPC Direction

The ONC RPC protocol as described in [RFC5531] is fundamentally a message-passing protocol involving one server and perhaps multiple clients. ONC RPC transactions are made up of two types of messages.

A CALL message, or "call", requests work. A call is designated by the value CALL in the message’s msg_type field. An arbitrary unique value is placed in the message’s xid field. A host that originates a call is referred to in this document as a "caller."

A REPLY message, or "reply", reports the results of work requested by a call. A reply is designated by the value REPLY in the message’s msg_type field. The value contained in the message’s xid field is copied from the call whose results are being reported. A host that emits a reply is referred to as a "responder."

RPC-over-RDMA is a connection-oriented RPC transport. When a connection-oriented transport is used, ONC RPC client endpoints are responsible for initiating transport connections, while ONC RPC service endpoints wait passively for incoming connection requests.

We do not consider RPC direction on connectionless RPC transports in this document.

1.3.1. Forward Direction

A traditional ONC RPC client is always a caller. A traditional ONC RPC service is always a responder. This traditional form of ONC RPC message passing is referred to as operation in the "forward direction."

During forward direction operation, the ONC RPC client is responsible for establishing transport connections.

1.3.2. Backward Direction

The ONC RPC standard does not forbid passing messages in the other direction. An ONC RPC service endpoint can act as a caller, in which case an ONC RPC client endpoint acts as a responder. This form of message passing is referred to as operation in the "backward direction."

During backward direction operation, the ONC RPC client is responsible for establishing transport connections, even though ONC RPC calls may come from the ONC RPC server.

ONC RPC clients and services are optimized to perform and scale well while handling traffic in the forward direction, and may not be prepared to handle operation in the backward direction. Not until recently has there been a need to handle backward direction operation.

1.3.3. Bi-direction

A pair of endpoints may choose to use only forward or only backward direction operations on a particular transport. Or, the endpoints may send operations in both directions concurrently on the same transport.

Bi-directional operation occurs when both transport endpoints act as a caller and a responder at the same time. As above, the ONC RPC client is responsible for establishing transport connections.

1.3.4. XID Values

Section 9 of [RFC5531] introduces the ONC RPC transaction identifier, or "xid" for short. The value of an xid is interpreted in the context of the message's msg_type field.

- The xid of a call is arbitrary but is unique among outstanding calls from that caller.

- The xid of a reply always matches that of the initiating call.

A caller matches the xid value in each reply with a call it previously sent.

1.3.4.1. XIDs with Bi-direction

During bi-directional operation, the forward and backward directions use independent xid spaces.

In other words, a forward direction caller MAY use the same xid value at the same time as a backward direction caller on the same transport connection. Though such concurrent requests use the same xid value, they represent distinct ONC RPC transactions.

1.4. Rationale For RPC-over-RDMA Bi-Direction

1.4.1. NFSv4.0 Callback Operation

An NFSv4.0 client employs a traditional ONC RPC client to send NFS requests to an NFSv4.0 server's traditional ONC RPC service [RFC7530]. NFSv4.0 requests flow in the forward direction on a connection established by the client. This connection is referred to as a "forechannel."

NFSv4.0 introduces the use of callback operations, or "callbacks", in Section 10.2 of [RFC7530], for managing file delegation. An NFSv4.0 server sets up a traditional ONC RPC client, and an NFSv4.0 client sets up a traditional ONC RPC service to handle callbacks. Callbacks flow in the forward direction on a connection established by an NFSv4.0 server. This connection is distinct from connections being used as forechannels. This connection is referred to as a "backchannel."

When an RDMA transport is used as a forechannel, an NFSv4.0 client typically provides a TCP callback service. The client's SETCLIENTID operation advertises the callback service endpoint with a "tcp" or "tcp6" netid. The server then connects to this service using a TCP socket.

NFSv4.0 implementations are fully functional without a backchannel in place. In this case, the server does not grant file delegations. This might result in a negative performance effect, but functional correctness is unaffected.

1.4.2. NFSv4.1 Callback Operation

NFSv4.1 supports file delegation in a similar fashion to NFSv4.0, and extends the repertoire of callbacks to manage pNFS layouts, as discussed in Chapter 12 of [RFC5661].

For various reasons, NFSv4.1 requires that all transport connections be initiated by NFSv4.1 clients. Therefore, NFSv4.1 servers send callbacks to clients in the backward direction on connections established by NFSv4.1 clients.

An NFSv4.1 client or server indicates to its peer that a backchannel capability is available on a given transport when sending a CREATE_SESSION or BIND_CONN_TO_SESSION operation.

NFSv4.1 clients may establish distinct transport connections for forechannel and backchannel operation, or they may combine forechannel and backchannel operation on one transport connection using bi-directional operation.

When an RDMA transport is used as a forechannel, an NFSv4.1 client must additionally connect using a transport with backward direction capability to use as a backchannel. Without a backward direction RPC-over-RDMA capability, TCP is the only choice at present for an NFSv4.1 backchannel connection.

Some implementations find it more convenient to use a single combined transport (ie. a transport that is capable of bi-directional operation). This simplifies connection establishment and recovery during network partitions or when one endpoint restarts.

As with NFSv4.0, if a backchannel is not in use, an NFSv4.1 server does not grant delegations. But because of its reliance on callbacks to manage pNFS layout state, pNFS operation is not possible without a backchannel.

1.5. Design Considerations

As of this writing, the only use case for backward direction ONC RPC messages is the NFSv4.1 backchannel. The conventions described in this document take advantage of certain characteristics of NFSv4.1 callbacks, namely:

- NFSv4.1 callbacks typically bear small argument and result payloads

- NFSv4.1 callback payloads are insensitive to alignment relative to system pages

- NFSv4.1 callbacks are infrequent relative to forechannel operations

1.5.1. Backward Compatibility

Existing clients that implement RPC-over-RDMA version 1 should interoperate correctly with servers that implement RPC-over-RDMA with backward direction support, and vice versa.

The approach taken here avoids altering the RPC-over-RDMA version 1 XDR specification. Keeping the XDR the same enables existing RPC-over-RDMA version 1 implementations to interoperate with implementations that support operation in the backward direction.

1.5.2. Performance Impact

Support for operation in the backward direction should never impact the performance or scalability of forward direction operation, where the bulk of ONC RPC transport activity typically occurs.

1.5.3. Server Memory Security

RDMA transfers involve one endpoint exposing a section of its memory to the other endpoint, which then drives RDMA READ and WRITE operations to access or modify the exposed memory. RPC-over-RDMA client endpoints expose their memory, and RPC-over-RDMA server endpoints initiate RDMA data transfer operations.

If RDMA transfers are not used for backward direction operations, there is no need for servers to expose their memory to clients. Further, this avoids the client complexity required to drive RDMA transfers.

1.5.4. Payload Size

Small RPC-over-RDMA messages are conveyed using only RDMA SEND operations. SEND is used to transmit both ONC RPC calls and replies.

To send a large payload, an RPC-over-RDMA client endpoint registers a region of memory known as a chunk, and transmits its coordinates to a server endpoint, who uses an RDMA transfer to move data to or from the client. See Sections 3.1, 3.2, and 3.4 of [RFC5666].

To transmit RPC-over-RDMA messages larger than the receive buffer size (typically 1024 bytes), a chunk must be used. For example, in an RDMA_NOMSG type message, the entire RPC header and Upper Layer payload are contained in chunks. See Section 5.1 of [RFC5666] for details.

If chunks are not allowed to be used for conveying backward direction messages, an RDMA_NOMSG type message cannot be used to convey a backward direction message using the conventions described in this document. Therefore, backward direction messages sent using the conventions in this document can be no larger than a single receive buffer.

Stipulating such a limit on backward direction message size assumes that either Upper Layer Protocol consumers of backward direction messages can advertise this limit to peers, or that ULP consumers can agree by convention on a maximum size of their backchannel payloads.

In addition, using only inline forms of RPC-over-RDMA messages and never populating the RPC-over-RDMA chunk lists means that the RPC header's msg_type field is always at a fixed location in messages flowing in the backward direction, allowing efficient detection of the direction of an RPC-over-RDMA message.

With few exceptions, NFSv4.1 servers can break down callback requests so they fit within this limit. There are potentially large NFSv4.1 callback operations, such as a CB_GETATTR operation where a large ACL must be conveyed. Although we are not aware of any NFSv4.1 implementation that uses CB_GETATTR, this state of affairs is not guaranteed in perpetuity.

2. Conventions For Backward Operation

Performing backward direction ONC RPC operations over an RPC-over-RDMA transport can be accomplished within limits by observing the conventions described in the following subsections. For reference, the XDR description of RPC-over-RDMA version 1 is contained in Section 4.3 of [RFC5666].

2.1. Flow Control

For an RDMA SEND operation to work, the receiving consumer must have posted an RDMA RECV Work Request to provide a receive buffer in which to capture the incoming message. If a receiver hasn't posted enough RECV WRs to catch incoming SEND operations, the RDMA provider is allowed to drop the RDMA connection.

The RPC-over-RDMA version 1 protocol provides built-in send flow control to prevent overrunning the number of pre-posted receive buffers on a connection's receive endpoint. This is fully discussed in Section 3.3 of [RFC5666].

2.1.1. Forward Credits

An RPC-over-RDMA credit is roughly the capability to handle one RPC-over-RDMA transaction. Each forward direction RPC-over-RDMA call requests a number of credits from the responder. Each forward direction reply informs the caller how many credits the responder is prepared to handle in total. The value of the request and grant are carried in each RPC-over-RDMA message's rdma_credit field.

Practically speaking, the critical value is the value of the rdma_credit field in RPC-over-RDMA replies. When a caller is operating correctly, it sends no more outstanding requests at a time than the responder's advertised forward direction credit value.

The credit value is a guaranteed minimum. However, a receiver can post more receive buffers than its credit value. There is no requirement in the RPC-over-RDMA protocol for a receiver to indicate a credit overrun. Operation continues as long as there are enough receive buffers to handle incoming messages.

2.1.2. Backward Credits

Credits work the same way in the backward direction as they do in the forward direction. However, forward direction credits and backward direction credits are accounted separately.

In other words, the forward direction credit value is the same whether or not there are backward direction resources associated with an RPC-over-RDMA transport connection. The backward direction credit value MAY be different than the forward direction credit value. The rdma_credit field in a backward direction RPC-over-RDMA message MUST NOT contain the value zero.

A backward direction caller (an RPC-over-RDMA service endpoint) requests credits from the responder (an RPC-over-RDMA client endpoint). The responder reports how many credits it can grant. This is the number of backward direction calls the responder is prepared to handle at once.

When an RPC-over-RDMA server endpoint is operating correctly, it sends no more outstanding requests at a time than the client endpoint's advertised backward direction credit value.

2.2. Managing Receive Buffers

An RPC-over-RDMA transport endpoint must pre-post receive buffers before it can receive and process incoming RPC-over-RDMA messages. If a sender transmits a message for a receiver which has no prepared receive buffer, the RDMA provider is allowed to drop the RDMA connection.

2.2.1. Client Receive Buffers

Typically an RPC-over-RDMA caller posts only as many receive buffers as there are outstanding RPC calls. A client endpoint without backward direction support might therefore at times have no pre-posted receive buffers.

To receive incoming backward direction calls, an RPC-over-RDMA client endpoint must pre-post enough additional receive buffers to match its advertised backward direction credit value. Each outstanding forward direction RPC requires an additional receive buffer above this minimum.

When an RDMA transport connection is lost, all active receive buffers are flushed and are no longer available to receive incoming messages. When a fresh transport connection is established, a client endpoint must re-post a receive buffer to handle the reply for each retransmitted forward direction call, and a full set of receive buffers to handle backward direction calls.

2.2.2. Server Receive Buffers

A forward direction RPC-over-RDMA service endpoint posts as many receive buffers as it expects incoming forward direction calls. That is, it posts no fewer buffers than the number of RPC-over-RDMA credits it advertises in the rdma_credit field of forward direction RPC replies.

To receive incoming backward direction replies, an RPC-over-RDMA server endpoint must pre-post a receive buffer for each backward direction call it sends.

When the existing transport connection is lost, all active receive buffers are flushed and are no longer available to receive incoming messages. When a fresh transport connection is established, a server endpoint must re-post a receive buffer to handle the reply for each retransmitted backward direction call, and a full set of receive buffers for receiving forward direction calls.

2.2.3. In the Absense of Backward Direction Support

An RPC-over-RDMA transport endpoint might not support backward direction operation. There might be no mechanism in the implementation to do so. Or the Upper Layer Protocol consumer might not yet have configured the transport to handle backward direction traffic.

A loss of the RDMA connection may result if the receiver is not prepared to receive an incoming message. Thus a denial-of-service could result if a sender continues to send backchannel messages after every transport reconnect to an endpoint that is not prepared to receive them.

Generally, for RPC-over-RDMA version 1 transports, the Upper Layer Protocol consumer is responsible for informing its peer when it has no support for the backward direction. Otherwise even a simple backward direction NULL probe from a peer would result in a lost connection.

An NFSv4.1 server should never send backchannel messages to an NFSv4.1 client before the NFSv4.1 client has sent a CREATE_SESSION or a BIND_CONN_TO_SESSION operation. As long as an NFSv4.1 client has prepared appropriate backchannel resources before sending one of these operations, denial-of-service is avoided. Legacy versions of NFS should never send backchannel operations.

Therefore, an Upper Layer Protocol consumer MUST NOT perform backward direction ONC RPC operations unless the peer consumer has indicated it is prepared to handle them. A description of Upper Layer Protocol mechanisms used for this indication is outside the scope of this document.

2.3. Backward Direction Retransmission

In rare cases, an ONC RPC transaction cannot be completed within a certain time. This can be because the transport connection was lost, the call or reply message was dropped, or because the Upper Layer consumer delayed or dropped the ONC RPC request. Typically, the caller sends the transaction again, reusing the same RPC XID. This is known as an "RPC retransmission".

In the forward direction, the caller is the ONC RPC client. The client is always responsible for establishing a transport connection before sending again.

In the backward direction, the caller is the ONC RPC server. Because an ONC RPC server does not establish transport connections with clients, it cannot send a retransmission if there is no transport connection. It must wait for the ONC RPC client to re-establish the transport connection before it can retransmit ONC RPC transactions in the backward direction.

If an ONC RPC client has no work to do, it may be some time before it re-establishes a transport connection. Backward direction callers must be prepared to wait indefinitely before a connection is established before a pending backward direction ONC RPC call can be retransmitted.

2.4. Backward Direction Message Size

RPC-over-RDMA backward direction messages are transmitted and received using the same buffers as messages in the forward direction. Therefore they are constrained to be no larger than receive buffers posted for forward messages. Typical implementations have chosen to use 1024-byte buffers.

It is expected that the Upper Layer Protocol consumer establishes an appropriate payload size limit for backward direction operations, either by advertising that size limit to its peers, or by convention. If that is done, backward direction messages would not exceed the size of receive buffers at either endpoint.

If a sender transmits a backward direction message that is larger than the receiver is prepared for, the RDMA provider drops the message and the RDMA connection.

If a sender transmits an RDMA message that is too small to convey a complete and valid RPC-over-RDMA and RPC message in either direction, the receiver MUST NOT use any value in the fields that were transmitted. Namely, the rdma_credit field MUST be ignored, and the message dropped.

2.5. Sending A Backward Direction Call

To form a backward direction RPC-over-RDMA call message on an RPC-over-RDMA version 1 transport, an ONC RPC service endpoint constructs an RPC-over-RDMA header containing a fresh RPC XID in the rdma_xid field (see Section 1.3.4 for full requirements).

The rdma_vers field MUST contain the value one. The number of requested credits is placed in the rdma_credit field (see Section 2.1).

The rdma_proc field in the RPC-over-RDMA header MUST contain the value RDMA_MSG. All three chunk lists MUST be empty.

The ONC RPC call header MUST follow immediately, starting with the same XID value that is present in the RPC-over-RDMA header. The call header's msg_type field MUST contain the value CALL.

2.6. Sending A Backward Direction Reply

To form a backward direction RPC-over-RDMA reply message on an RPC-over-RDMA version 1 transport, an ONC RPC client endpoint constructs an RPC-over-RDMA header containing a copy of the matching ONC RPC call's RPC XID in the rdma_xid field (see Section 1.3.4 for full requirements).

The rdma_vers field MUST contain the value one. The number of granted credits is placed in the rdma_credit field (see Section 2.1).

The rdma_proc field in the RPC-over-RDMA header MUST contain the value RDMA_MSG. All three chunk lists MUST be empty.

The ONC RPC reply header MUST follow immediately, starting with the same XID value that is present in the RPC-over-RDMA header. The reply header's msg_type field MUST contain the value REPLY.

3. Limits To This Approach

3.1. Payload Size

The major drawback to the approach described in this document is the limit on payload size in backward direction requests.

- o Some NFSv4.1 callback operations can have potentially large arguments or results. For example, CB_GETATTR on a file with a large ACL; or CB_NOTIFY, which can provide a large, complex argument.

- o Any backward direction operation protected by RPCSEC_GSS may have additional header information that makes it difficult to send backward direction operations with large arguments or results.

- o Larger payloads could potentially require the use of RDMA data transfers, which are complex and make it more difficult to detect backward direction requests. The msg_type field in the ONC RPC header would no longer be at a fixed location in backward direction requests.

3.2. Preparedness To Handle Backward Requests

A second drawback is the exposure of the client transport endpoint to backward direction calls before it has posted receive buffers to handle them.

Clients that do not support backward direction operation typically drop messages they do not recognize. However, this does not allow bi-direction-capable servers to quickly identify clients that cannot handle backward direction requests.

The conventions in this document rely on Upper Layer Protocol consumers to decide when backward direction transport operation is appropriate.

3.3. Long Term

To address the limitations described in this section in the long run, a new version of the RPC-over-RDMA protocol would be required. The use of the conventions described in this document to enable backward direction operation is thus a transitional approach that is appropriate only while RPC-over-RDMA version 1 is the predominantly deployed version of the RPC-over-RDMA protocol.

4. Security Considerations

As a consequence of limiting the size of backward direction RPC-over-RDMA messages, the use of RPCSEC_GSS integrity and confidentiality services (see [RFC2203]) in the backward direction may be challenging due to the size of the additional RPC header information required for RPCSEC_GSS.

5. IANA Considerations

This document does not require actions by IANA.

6. Acknowledgements

Tom Talpey was an indispensable resource, in addition to creating the foundation upon which this work is based. Our warmest regards go to him for his help and support.

Dave Noveck provided excellent review, constructive suggestions, and navigational guidance throughout the process of drafting this document.

Dai Ngo was a solid partner and collaborator. Together we constructed and tested independent prototypes of the conventions described in this document.

The author wishes to thank Bill Baker for his unwavering support of this work. In addition, the author gratefully acknowledges the expert contributions of Karen Deitke, Chunli Zhang, Mahesh Siddheshwar, and Tom Tucker.

Special thanks go to the nfsv4 Working Group chair Spencer Shepler and the WG Editor Tom Haynes for their support.

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, September 1997.

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

[RFC5661] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 2010.

[RFC5666] Talpey, T. and B. Callaghan, "Remote Direct Memory Access Transport for Remote Procedure Call", RFC 5666, January 2010.

[RFC7530] Haynes, T. and D. Noveck, "Network File System (NFS) Version 4 Protocol", RFC 7530, March 2015.

Author's Address

Charles Lever
Oracle Corporation
1015 Granger Avenue
Ann Arbor, MI 48104
US

Phone: +1 734 274 2396
Email: chuck.lever@oracle.com