NFSv4 C. Lever
Internet-Draft Oracle
Intended status: Experimental April 24, 2015
Expires: October 26, 2015

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

Abstract

Recent minor versions of NFSv4 work best when ONC RPC transports can send ONC RPC calls 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

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 26, 2015.

Copyright Notice

Copyright (c) 2015 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 (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.

Table of Contents

1. Introduction 2
1.1. Requirements Language 2
1.2. Scope Of This Document 3
1.3. Understanding RPC Direction 3
1.3.1. Forward Direction 3
1.3.2. Backwards Direction 4
1.3.3. Bi-direction 4
1.4. Rationale For RPC-over-RDMA Bi-Direction 4
1.4.1. NFSv4.0 Callback Operation 4
1.4.2. NFSv4.1 Callback Operation 5
1.5. Design Considerations 5
1.5.1. Backwards Compatibility 6
1.5.2. Performance Impact 6
1.5.3. Client Complexity 6
1.5.4. Payload Size 6
2. Conventions For Backwards Operation 7
2.1. Flow Control 7
2.1.1. Forwards Credits 7
2.1.2. Backwards Credits 8
2.2. Managing Receive Buffers 8
2.2.1. Client Receive Buffers 8
2.2.2. Server Receive Buffers 9
2.2.3. In the Absense of Backwards Direction Support 9
2.3. Backwards Direction Message Size 10
2.4. Sending A Backwards Direction Call 10
2.5. Sending A Backwards Direction Reply 11
3. Limits To This Approach 11
3.1. Payload Size 11
3.2. Preparedness To Handle Backwards Requests 11
3.3. Long Term 12
4. Security Considerations 12
5. IANA Considerations 12
6. Acknowledgements 12
7. Normative References 12
Author's Address 13

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 handle RPC calls that flow from client to server and server to client concurrently.

No changes to the RPC-over-RDMA version 1 protocol definition are needed, thus this document does not update [RFC5666].

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

Providing an Upper Layer Binding for NFSv4.x callback operations is not in 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. There are two types of messages.

A CALL message requests work. A CALL message 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. The host that originates a CALL message is referred to as the "caller."

A REPLY message reports the results of requested work. A REPLY message 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 message whose results are being reported. The host that emits a REPLY message is referred to as the "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 await 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 the caller. A traditional ONC RPC service is always the responder. This traditional form of ONC RPC message passing is referred to as operation in the "forward direction."

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

1.3.2. Backwards 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 "backwards direction."

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

Notably, traditional ONC RPC clients and services are usually not prepared for backwards operation. ONC RPC clients and services are heavily optimized to perform and scale well while handling traffic in the forward direction. Not until recently has there been any need to handle operation in the backwards direction.

1.3.3. Bi-direction

Finally, 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.

1.4. 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 a 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, 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 callback operations. These requests flow in a forward direction on a connection established by the server. This connection is referred to as a "backchannel."

When an RDMA transport is used for the 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 is fully functional without a backchannel in place. The server simply does not grant file delegations. Applications might experience a performance impact, but operational correctness is not affected.

1.4.2. NFSv4.1 Callback Operation

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

Further, NFSv4.1 requires that all transport connections are initiated by NFSv4.1 clients. Thus, NFSv4.1 servers send callback operations to clients in the backwards direction on connections that NFSv4.1 clients establish with servers.

NFSv4.1 clients may establish separate transport connections for forechannel and backchannel operation, or they may combine forechannel and backchannel operation on the same transport connection.

An NFSv4.1 client or server can signal its peer that a backchannel capability is available on a given transport by sending a CREATE_SESSION or BIND_CONN_TO_SESSION operation. When an RDMA transport is used for the forechannel, an NFSv4.1 client must additionally connect using a transport with bi-directional RPC capability to use as a backchannel. Without a bi-directional RPC-over-RDMA capability, TCP is the only choice at present for an NFSv4.1 backchannel connection.

Some implementations prefer using a single combined transport. This simplifies connection establishment and recovery during network partitions or when one endpoint restarts.

Like NFSv4.0, if a backchannel is not in use, an NFSv4.1 server does not grant delegations. But because of its reliance on callback operations to manage pNFS layout state, pNFS operation is impossible without a backchannel.

1.5. Design Considerations

1.5.1. Backwards Compatibility

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

We prefer to avoid altering the RPC-over-RDMA version 1 XDR specification. Keeping the XDR the same enables existing RPC-over-RDMA version 1 implemenations to continue to interoperate with implementations that support operation in the backwards direction.

1.5.2. Performance Impact

Support for operation in the backwards 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. Client Complexity

RDMA transfers involve one endpoint exposing a portion of its memory to the other, which then drives RDMA READ and WRITE operations to access or modify the exposed memory. NFS clients expose their memory, and NFS servers initiate RDMA data transfers.

We prefer to avoid having the server expose its memory to clients, and to avoid introducing client complexity to drive RDMA operations.

1.5.4. Payload Size

Perhaps the only extant use case for backwards direction ONC RPC messages is the NFSv4.1 backchannel. Our conventions can leverage certain characteristics of NFSv4.1 callback operations. These operations typically do not bear large argument or result payloads, and are infrequent relative to forechannel operations.

Small RPC-over-RDMA messages are conveyed using only RDMA SEND, without the complexity overhead of managing chunks. If only SEND is permitted for backwards direction message, an RDMA_NOMSG type message, which requires the use of a chunk, cannot be used to convey a backwards direction message.

The price for this simplicity is that no backwards direction message can be larger than the forward direction’s receive buffer size (typically 1024 bytes).

Stipulating such a limit on backwards direction message size assumes either that Upper Layer Protocol consumers of backwards 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 backwards 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 a few 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 Backwards Operation

Performing backwards 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

An RDMA SEND operation fails if the receiver has not pre-posted enough buffers to receive the sent message. A sender might retransmit the SEND operation, or it can choose to drop the connection if message reception fails.

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

2.1.1. Forwards Credits

An RPC-over-RDMA credit is roughly the capability to handle one RPC-over-RDMA call. 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 an caller is operating correctly, it sends no more outstanding requests at a time than the responder's advertised forward direction credit value.

2.1.2. Backwards Credits

Credits work the same way in the backwards direction as they do in the forward direction. However, forward direction credits and backwards 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 backwards direction credit value MAY be different than the forwards direction credit value.

A backwards 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 backwards 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 backwards direction credit value.

If a sender transmits a backward direction message that exceeds the receiver's backwards direction credit limit, the receiver MAY drop the transport connection, or it MAY return an RPC-over-RDMA error to the sender. The rdma_credit field in a backwards direction RPC-over-RDMA message MUST NOT contain the value zero.

2.2. Managing Receive Buffers

A 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 receiver MUST drop the transport connection (?). This is true no matter which direction a message flows.

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 backwards direction support might therefore at times have no pre-posted receive buffers.

To receive incoming backwards direction calls, an RPC-over-RDMA client endpoint must pre-post enough additional receive buffers to match its backwards direction credit advertisement.

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 backwards 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 backwards direction replies, an RPC-over-RDMA server endpoint must pre-post a receive buffer for each backwards 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 backwards direction call, and a full set of receive buffers for receiving forward direction calls.

2.2.3. In the Absense of Backwards Direction Support

An RPC-over-RDMA transport endpoint might not support backwards 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 backwards direction traffic.

Since a receiver may drop the transport connection after receiving a message it was not prepared for, a denial-of-service could result if a sender continues to send backchannel messages after every transport reconnect.

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 backwards direction. Otherwise even a simple backwards direction NULL probe from a peer results in a lost connection.

For NFSv4.1, there is a built-in safety net for this case. 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 backwards direction ONC RPC operations unless the peer consumer has signaled it is prepared to handle them. A description of the Upper Layer Protocol mechanisms used for this signal is not in the scope of this document.

2.3. Backwards Direction Message Size

RPC-over-RDMA backwards 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, either by advertising that size limit to its peers, or by convention. That way, backwards direction messages do not exceed the size of receive buffers at either endpoint.

If a sender transmits a backwards direction message that is larger than the receiver is prepared for, or the message is too small to convey a complete and valid RPC-over-RDMA and RPC message, the receiver MUST drop the transport connection.

2.4. Sending A Backwards Direction Call

To form a backwards 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. 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.

2.5. Sending A Backwards Direction Reply

To form a backwards 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. 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 backwards direction requests.

- 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.

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

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

3.2. Preparedness To Handle Backwards Requests

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

Clients that do not support backwards 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 backwards direction requests.

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

3.3. Long Term

To address these limitations in the long run, we feel a revision of the RPC-over-RDMA version 1 XDR is required, and that using conventions to enable backwards direction operation is therefore a transitional approach which 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 backwards direction RPC-over-RDMA messages, the use of RPCSEC_GSS integrity and confidentiality services (see [RFC2203]) may be challenging in the backwards direction 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

The author of this document gratefully acknowledges the contributions of Tom Talpey, Dai Ngo, Karen Deitke, Chunli Zhang, Dave Noveck, and Bill Baker.

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