Network File System Version 4 C. Lever
Internet-Draft Oracle
Intended status: Standards Track May 2, 2016
Expires: November 3, 2016

Bi-directional Remote Procedure Call On RPC-over-RDMA Transports
draft-ietf-nfsv4-rpcrdma-bidirection-03

Abstract

Recent minor versions of NFSv4 work best when ONC RPC transports can send Remote Procedure Call transactions in both directions on the same connection. This document describes how RPC-over-RDMA transport endpoints convey RPCs in both directions on a single connection.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The purpose of this document is to enable bi-directional RPC operation on RPC-over-RDMA protocol versions that do not have specific protocol facilities for backward direction operation. Backward direction RPC transactions enable the operation of NFSv4.1, and in particular pNFS.

For example, using the protocol described in this document, RPC transactions can be conveyed in both directions on the same RPC-over-RDMA Version One connection without changes to the Version One header XDR description. Therefore this document does not update [I-D.ietf-nfsv4-rfc5666bis].

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

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].

2. Understanding RPC Direction

The ONC RPC protocol as described in [RFC5531] is fundamentally a message-passing protocol between one server and one or more 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 "Requester."

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 returned. A host that emits a Reply is referred to as a "Responder."

Typically, a Call generates a corresponding Reply. A Reply is never sent without a corresponding Call.

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.

RPC direction on connectionless RPC transports is not considered in this document.

2.1. Forward Direction

A traditional ONC RPC client is always a Requester. 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.

2.2. Backward Direction

The ONC RPC specification [RFC5531] does not forbid passing messages in the other direction. An ONC RPC service endpoint can act as a Requester, 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 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.

2.3. Bi-directional Operation

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

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

2.4. 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.

- o The xid of a Call is arbitrary but is unique among outstanding Calls from that Requester.
- o The xid of a Reply always matches that of the initiating Call.

When receiving a Reply, a Requester matches the xid value in the Reply with a Call it previously sent.

2.4.1. XID Generation

During bi-directional operation, forward and backward direction XIDs are typically generated on distinct hosts by possibly different algorithms. There is no co-ordination between forward and backward direction XID generation.

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

3. Rationale For Bi-Directional RPC-over-RDMA

3.1. 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" connection.

An NFSv4 "delegation" is simply a promise made by a server that it will notify a client before another agent is allowed access to a file. With this guarantee, that client can operate as sole accessor of the file. In particular, it can manage the file’s data and metadata caches aggressively.

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

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.

3.2. NFSv4.1 Callback Operation

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

To facilitate operation through NAT routers, all NFSv4.1 transport connections are initiated by NFSv4.1 clients. Therefore NFSv4.1 servers send callbacks to clients in the backward direction on connections established by NFSv4.1 clients.

NFSv4.1 clients and servers indicate to their peers that a backchannel capability is available on a given transport in the arguments and results of NFS CREATE_SESSION or BIND_CONN_TO_SESSION operations.

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.

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

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.

4. Flow Control

For an RDMA Send operation to work, the receiving peer must have posted an RDMA Receive Work Request (WR) to provide a receive buffer in which to land the incoming message. If a receiver hasn’t posted enough Receive WRs to land incoming Send operations, the RDMA provider is allowed to drop the RDMA connection.

RPC-over-RDMA transport protocols provide 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 4.3 of [I-D.ietf-nfsv4-rfc5666bis].

4.1. 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 Requester (ie, an RPC-over-RDMA service endpoint) requests credits from the Responder (ie, an RPC-over-RDMA client endpoint). The Responder reports how many credits it has granted. This is the number of backward direction Calls the Responder is prepared to handle at once.

When message direction is not fully determined by context or by an accompanying RPC message with a call direction field, it is not possible to tell whether the header credit value is a request or grant, or whether the value applies to the forward direction or backward direction. In such cases, the receiver MUST NOT use the header’s credit value.

4.2. 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.

4.2.1. Client Receive Buffers

Typically an RPC-over-RDMA Requester 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.

4.2.2. 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 credits granted 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.

5. Protocol For Backward Operation

Performing backward direction ONC RPC operations over an RPC-over-RDMA transport connection can be accomplished by observing the protocol described in the following subsections. For reference, the XDR description of RPC-over-RDMA Version One is contained in Section 5.1 of [I-D.ietf-nfsv4-rfc5666bis].

5.1. Sending A Backward Direction Call

To form a backward direction RPC-over-RDMA Call message, an ONC RPC service endpoint constructs an RPC-over-RDMA header containing a fresh RPC XID in the rdma_xid field (see Section 2.4 for full requirements).

The rdma_vers field MUST contain the same value in backward and forward direction Call messages on the same connection.

The number of requested backward direction credits is placed in the rdma_credit field (see Section 4).

Whether presented inline or as a separate chunk, the ONC RPC Call header MUST start with the same XID value that is present in the RPC-over-RDMA header, and the header’s msg_type field MUST contain the value CALL.

5.2. Sending A Backward Direction Reply

To form a backward direction RPC-over-RDMA Reply message, 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 2.4 for full requirements).

The rdma_vers field MUST contain the same value in a backward direction Reply message as in the matching Call message.

The number of granted backward direction credits is placed in the rdma_credit field (see Section 4).

Whether presented inline or as a separate chunk, the ONC RPC Reply header MUST start with the same XID value that is present in the RPC-over-RDMA header, and the header’s msg_type field MUST contain the value REPLY.

5.3. Backward Direction Chunks

Chunks MAY be used in the backward direction. They operate the same way as in the forward direction (see [I-D.ietf-nfsv4-rfc5666bis] for details).

An implementation might not support any Upper Layer Protocol that has DDP-eligible data items. The Upper Layer Protocol may also use only small messages, or it may have a native mechanism for restricting the size of backward direction RPC messages, obviating the need to handle Long Messages in the backward direction.

When there is no Upper Layer Protocol requirement for chunks, implementers can choose not to provide support for chunks in the backward direction. This avoids the complexity of adding support for performing RDMA Reads and Writes in the backward direction.

When chunks are not implemented, RPC messages in the backward direction are always sent using RDMA_MSG, and therefore can be no larger than what can be sent inline (that is, without chunks). Sending an inline message larger than the receiver’s inline threshold can result in loss of connection.

If a backward direction requester provides a non-empty chunk list to a responder that does not support chunks, the responder MUST reply with an RDMA_ERROR message with rdma_err field set to ERR_CHUNK.

5.4. 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 Requester sends the transaction again, reusing the same RPC XID. This is known as an "RPC retransmission".

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

In the backward direction, the Requester 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 Requesters must be prepared to wait indefinitely for a connection to be established before a pending backward direction ONC RPC Call can be retransmitted.

6. In the Absence of Backward Direction Support

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

If an endpoint is not prepared to receive an incoming backward direction message, loss of the RDMA connection might result. Thus a denial-of-service could result if a sender continues to send backward direction messages after every transport reconnect to an endpoint that is not prepared to receive them.

When dealing with the possibility that the remote peer has no transport level support for backward direction operation, the Upper Layer Protocol becomes responsible for informing peers when backward direction operation is supported. Otherwise even a simple backward direction NULL probe from a peer could result in a lost connection.

An NFSv4.1 server does not 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 announcing support for backchannel operation, denial-of-service is avoided.

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.

7. Backward Direction Upper Layer Binding

Since backward direction operation occurs on an already-established connection, there is no need to specify RPC bind parameters.

An Upper Layer Protocol that operates on RPC-over-RDMA transports in the backward direction may have DDP-eligible data items. These are specified in an Upper Layer Binding document.

By default, no data items in a ULP are DDP-eligible. If there are no DDP-eligible data items to document, an explicit Upper Layer Binding may not be needed for an Upper Layer Protocol that operates only in the backward direction.

Consult Section 7 of [I-D.ietf-nfsv4-rfc5666bis] for details about what else may be contained in a binding.

8. Security Considerations

Security considerations for operation on RPC-over-RDMA transports are outlined in Section 9 of [I-D.ietf-nfsv4-rfc5666bis].

9. IANA Considerations

This document does not require actions by IANA.

10. 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 changes 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, Steve Wise, and Tom Tucker.

Special thanks go to the nfsv4 Working Group Chair Spencer Shepler and the nfsv4 Working Group Secretary Tom Haynes for their support.

11. Normative References

[I-D.ietf-nfsv4-rfc5666bis]
Lever, C., Simpson, W., and T. Talpey, "Remote Direct Memory Access Transport for Remote Procedure Call", draft-ietf-nfsv4-rfc5666bis-04 (work in progress), March 2016.

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

[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
USA

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