Network File System Version 4   D. Noveck
Internet-Draft   HPE
Intended status: Standards Track   August 18, 2016
Expires: February 19, 2017

RPC-over-RDMA Extension to Manage Transport Characteristics
draft-dnoveck-nfsv4-rpcrdma-xcharext-02

Abstract

This document specifies an extension to RPC-over-RDMA Version Two. The extension enables endpoints of an RPC-over-RDMA connection to exchange information which can be used to optimize message transfer.

Status of This Memo

Copyright Notice

Table of Contents

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

This document specifies an extension to RPC-over-RDMA Version Two. It allows each participating endpoint on a single connection to communicate various characteristics of its implementation, to request changes in characteristics of the other endpoint, and to effect

changes and notify the other endpoint of changes to these characteristics during operation.

The extension described herein specifies OPTIONAL message header types to implement this mechanism. The means by which the implementation support status of these OPTIONAL types is ascertained is described in [rpcrmdav2].

Although this document specifies the new OPTIONAL message header types to implement these functions, the precise means by which the presence of support for these OPTIONAL functions will be ascertained is not described here, as would be done more appropriately by the RFC defining a version of RPC-over-RDMA which supports protocol extension.

This document is currently written to conform to the extension model for RPC-over-RDMA Version Two as described in [rpcrdmav2].

1.3. Role Terminology

A number of different terms are used regarding the roles of the two participants in an RPC-over-RMA connection. Some of these roles last for the duration of a connection while others vary from request to request or from message to message.

The roles of the client and server are fixed for the lifetime of the connection, with the client defined as the endpoint which initiated the connection.

The roles of requester and responder often parallel those of client and server, although this is not always the case. Most requests are made in the forward direction, in which the client is the requester and the server is the responder. However, backward-direction requests are possible, in which case the server is the requester and the client is the responder. As a result, clients and servers may both act as requesters and responders.

The roles of sender and receiver vary from message. With regard to the messages described in this document, both the client and the server can act as sender and receiver. With regard to messages used to transfer RPC requests and replies, the requester sends requests and receives replies while the responder receives requests and sends replies.

2. Transport Characteristics

2.1. Characteristics Model

An initial set of receiver and sender characteristics are specified in this document. An extensible approach is used, allowing new characteristics to be defined in future standards track documents.

Such characteristics are specified using:

- A code identifying the particular transport characteristic being specified.

- A nominally opaque array which contains within it the XDR encoding of the specific characteristic indicated by the associated code.

The following XDR types are used by operations that deal with transport characteristics:

```
<CODE BEGINS>

typedef xcharid         uint32;

struct xcharval {
        xcharid         xchv_which;
        opaque          xchv_data<>;
};

typedef xcharspec       xcharval<>;

typedef uint32          xcharsubset<>;

<CODE ENDS>
```

An xcharid specifies a particular transport characteristic. In order to allow easier XDR extension of the set of characteristics by concatenating XDR files, specific characteristics are defined as const values rather than as elements in an enum.

An xcharval specifies a value of a particular transport characteristic with the particular characteristic identified by xchv_which, while the associated value of that characteristic is contained within xchv_data.

While xchv_data is defined as opaque within the XDR, the contents are interpreted using the XDR typedef associated with the characteristic specified by xchv_which. The receiver of a message containing an

xcharval MUST report an XDR error if the length of xchv_data is such that it extends beyond the bounds of the message transferred.

In cases in which the xcharid specified by xchv_which is understood by the receiver, the receiver also MUST report an XDR error if either of the following occur:

- The nominally opaque data within xchv_data is not valid when interpreted using the characteristic-associated typedef.

- The length of xchv_data is insufficient to contain the data represented by the characteristic-associated typedef.

Note that no error is to be reported if xchv_which is unknown to the receiver. In that case, that xcharval is not processed and processing continues using the next xcharval, if any.

An xcharspec specifies a set of transport characteristics. No particular ordering of the xcharvals within it is imposed.

An xcharsubset identifies a subset of the characteristics in a previously specified xcharspec. Each bit in the mask denotes a particular element in a previously specified xcharspec. If a particular xcharval is at position N in the array, then bit number N mod 32 in word N div 32 specifies whether that particular xcharval is included in the defined subset. Words beyond the last one specified are treated as containing zero.

xcharsubsets are useful in a number of contexts:

- In an initial specification of transport characteristics, they allow the sender to specify what subset of those are subject to later change.

- In responding to a request to modify a set of transport characteristics, allows the responding endpoint to specify the subset of those characteristics that have been performed, have been requested, or have been accepted for later change, with notification of that change to be done asynchronously.

2.2. Transport Characteristics Groups

Transport characteristics are divided into a number of groups

- An initial set of transport characteristics defined in this document. See Section 3 for the complete list.

- Additional transport characteristics defined in future standards track documents as specified in Section 6.1.

- Experimental transport characteristics being explored preparatory to being considered for standards track definition. See the description in Section 6.2.

2.3. Operations Related to Transport Characteristics

There are a number of operations defined in Section 4 which are used to communicate and manage transport characteristics.

Prime among these is ROPT_INITXCH (defined in Section 4.1 which serves as a means by which an endpoints transport characteristics may be presented to its peer, typically upon establishing a connection.

In addition, there are a set of related operations concerned with requesting, effecting and reporting changes in transport characteristics:

- ROPT_REQXCH (defined in Section 4.2 which serves as a way for an endpoint to request that a peer change the value of a set of transport characteristics.

- ROPT_RESPXCH (defined in Section 4.3 is used to report on the disposition of each of the individual transport characteristic changes requested in a previous ROPT_REQXCH.

- ROPT_UPDXCH (defined in Section 4.4 is used to report a change in a transport characteristic. This may be one requested by a previous ROPT_REQXCH, or an unsolicited one, not being requested by a peer.

Unlike many other operation types, the above are not used to effect transfer of RPC requests but are internal one-way information transfers. However, a ROPT_REQXCH and the corresponding ROPT_RESPXCH do constitute an RPC-like remote call. The other operations are not part of a remote call transaction, although one or more asynchronous ROPT_UPDXCH operations may result from a ROPT_REQXCH.

3. Initial Transport Characteristics

Although the set of transport characteristics is subject to later extension, an initial set of transport characteristics is defined below in Table 1.

In that table, the columns contain the following information:

- The column labeled "characteristic" identifies the transport characteristic described by the current row.

- The column labeled "code" specifies the xcharid value used to identify this characteristic.

- The column labeled "XDR type" gives the XDR type of the data used to communicate the value of this characteristic. This data type overlays the nominally opaque field xchv_data in an xcharval.

- The column labeled "default" gives the default value for the characteristic which is to be assumed by those who do not receive, or are unable to interpret, information about the actual value of the characteristic.

- The column labeled "section" indicates the section (within this document) that explains the semantics and use of this transport characteristic.

characteristic	code	XDR type	default	section
Receive Buffer Size	1	uint32	4096	3.1
Requester Remote Invalidation	2	bool	false	3.2
Backward Request Support	3	enum bkreqsup	BKREQSUP_INLINE	3.3

Table 1

Note that there is no explicit indication regarding whether a particular characteristic can change or whether a change in the value may be requested (see Section 4.2). Such matters are not addressed by the protocol definition. A partner implementation can always request a change but peers MAY reject a request to change a characteristic for any reason. Implementations are always free to reject such requests if they cannot or do not wish to effect the requested change.

Either of the following will result in effective rejection requests to change specific characteristics:

- If an endpoint does not wish to accept request to change particular characteristics, it may reject such requests as described in Section 4.3.

- If an endpoint does not support the ROPT_REQXCH operation, the effect would be the same as if every request to change a set of characteristic were rejected.

With regard to unrequested changes in transport characteristics, it is the responsibility of the implementation making the change to do so in a fashion that which does not interfere with the other partner's continued correct operation (see Section 3.1).

3.1. Receive Buffer Size

The Receive Buffer Size specifies the minimum size, in octets, of pre-posted receive buffers. It is the responsibility of the participant sending this value to ensure that its pre-posted receives are at least the size specified, allowing the participant receiving this value to send messages that are of this size.

```
<CODE BEGINS>

const uint32    XCHAR_RBSIZ = 1;
typedef uint32  xchrbsiz;

<CODE ENDS>
```

The sender may use his knowledge of the receiver's buffer size to determine when the message to be sent will fit in the preposted receive buffers that the receiver has set up. In particular,

- Requesters may use the value to determine when it is necessary to provide a Position-Zero read chunk when sending a request.

- Requesters may use the value to determine when it is necessary to provide a Reply chunk when sending a request, based on the maximum possible size of the reply.

- Responders may use the value to determine when it is necessary, given the actual size of the reply, to actually use a Reply chunk provided by the requester.

Because there may be pre-posted receives with buffer sizes that reflect earlier values of the buffer size characteristic, changing this characteristics poses special difficulties:

- When the size is being raised, the partner should not be informed of the change until all pending receives using the older value have been eliminated.

- The size should not be reduced until the partner is aware of the need to reduce the size of future sends to conform to this reduced value. To ensure this, such a change should only occur in response to an explicit request by the other endpoint (See Section 4.2). The participant making the request should use that lower size as the send size limit until the request is rejected (See Section 4.3) or an update to a size larger than the requested value becomes effective and the requested change is no longer pending (See Section 4.4).

3.2. Requester Remote Invalidation

The Requester Remote Invalidation characteristic indicates that the current endpoint, when in the role of a requester, is prepared for the responder to use RDMA Send With Invalidate when replying to an RPC-over-RDMA request containing non-empty chuck lists.

As RPC-over-RDMA is currently used, memory registrations exposed to peers are not established by the server and explicit RDMA operations are not done to satisfy backward direction requests. This makes it unlikely that servers will present non-default values of the XCHAR_REQREMINV characteristic or that clients will take note of that value when presented by servers.

```
<CODE BEGINS>

const uint32    XCHAR_REQREMINV = 2;
typedef bool    xchrreqrem;

<CODE ENDS>
```

When the Requester Remote Invalidate characteristic is set to false, a responder MUST use Send to convey RPC reply messages to the requester. When the Requester Remote Invalidate characteristic is set to true, a responder MAY use Send With Invalidate instead of Send to convey RPC replies to the requester.

The value of the Requester Remote Invalidate characteristic is not likely to change from the value reported by ROPT_INITXCH (see Section 4.2).

3.3. Backward Request Support

The value of this characteristic is used to indicate a client implementation's readiness to accept and process messages that are part of backward-direction RPC requests.

```
<CODE BEGINS>

enum bkreqsup {
        BKREQSUP_NONE    = 0,
        BKREQSUP_INLINE  = 1,
        BKREQSUP_GENL    = 2
};

const uint32    XCHAR_BRS = 3;
typedef bkreqsup xchrbrs;

<CODE ENDS>
```

Multiple levels of support are distinguished:

- The value BKREQSUP_NONE indicates that receipt of backward-direction requests and replies is not supported.

- The value BKREQSUP_INLINE indicates that receipt of backward-direction requests or replies is only supported using inline messages and that use of explicit RDMA operations for backward direction requests or responses is not supported.

- The value BKREQSUP_GENL that receipt of backward-direction requests or replies is supported in the same ways that forward-direction requests or replies typically are.

The support level of servers can be inferred from the backward-direction requests that they issue, assuming that issuing a request implicitly indicates support for receiving the corresponding reply. On this basis, support for receiving inline replies can be assumed when requests without read chunks, write chunks, or Reply chunks are issued, while requests with any of these elements allow the client to assume that general support for backward-direction replies is present on the server.

4. New Operations

The proposed new operation are set forth in Table 2 below. In that table, the columns contain the following information:

- The column labeled "operation" specifies the particular operation.

- The column labeled "code" specifies the value of opttype for this operation.

- The column labeled "XDR type" gives the XDR type of the data structure used to describe the information in this new message

type. This data overlays the nominally opaque field optinfo in an RDMA_OPTIONAL message.

- The column labeled "msg" indicates whether this operation is followed (or not) by an RPC message payload.

- The column labeled "section" indicates the section (within this document) that explains the semantics and use of this optional operation.

operation	code	XDR type	msg	section
Specify Initial Characteristics	1	optinfo_initxch	No	4.1
Request Characteristic Modification	2	optinfo_reqxch	No	4.2
Respond to Modification Request	3	optinfo_respxch	No	4.3
Report Updated Characteristics	4	optinfo_updxch	No	4.4

Table 2

Support for all of the operations above is OPTIONAL. RPC-over-RDMA Version Two implementations that receive an operation that is not supported MUST respond with RDMA_ERROR message with an error code of RDMA_ERR_INVAL_OPTION as specified in [rpcrdmav2]

The only operation support requirements are as follows:

- Implementations which send ROPT_REQXCH messages must support ROPT_RESPXCH and ROPT_UPDXCH messages.

- Implementations which support ROPT_RESPXCH or ROPT_UPDXCH messages must also support ROPT_INITXCH messages.

4.1. ROPT_INITXCH: Specify Initial Characteristics

The ROPT_INITXCH message type allows an RPC-over-RDMA participant, whether client or server, to indicate to its partner relevant transport characteristics that the partner might need to be aware of.

The message definition for this operation is as follows:

```
<CODE BEGINS>

const uint32     ROPT_INITXCH = 1;

struct optinfo_initxch {
        xcharspec       optixch_start;
        xcharsubset     optixch_nochg;
};

<CODE ENDS>
```

All relevant transport characteristics that the sender is aware of should be included in optixch_start. Since support of this request is OPTIONAL, and since each of the characteristics is OPTIONAL as well, the sender cannot assume that the receiver will necessarily take note of these characteristics and so the sender should be prepared for cases in which the partner continues to assume that the default value for a particular characteristic is still in effect.

The subset of transport characteristic specified by optixch_nochg is not expected to change during the lifetime of the connection.

Generally, a participant will send a ROPT_INITXCH message as the first message after a connection is established. Given that fact, the sender should make sure that the message can be received by partners who use the default Receive Buffer Size. The connection's initial receive buffer size is typically 1KB, but it depends on the initial connection state of the RPC-over-RDMA version in use. See [rpcrdmav2] for details.

Those receiving an ROPT_INITXCH may encounter characteristics that they do not support or are unaware of. In such cases, these characteristics are simply ignored without any error response being generated.

4.2. ROPT_REQXCH: Request Modification of Characteristics

The ROPT_REQXCH message type allows an RPC-over-RDMA participant, whether client or server, to request of its partner that relevant transport characteristics be changed.

The partner need not change the characteristics as requested by the sender but if it does support the message type, it will generate a ROPT_RESPXCH message, indicating the disposition of the request.

The message definition for this operation is as follows:

```
<CODE BEGINS>

const uint32     ROPT_REQXCH = 2;

struct optinfo_reqxch {
        xcharspec       optreqxch_want;
};

<CODE ENDS>
```

The xcharspec optreqxch_want is a set of transport characteristics together with the desired values requested by the sender.

4.3. ROPT_RESPXCH: Respond to Request to Modify Transport Characteristics

The ROPT_RESPXCH message type allows an RPC-over-RDMA participant to respond to a request to change characteristics by its partner, indicating how the request was dealt with.

The message definition for this operation is as follows:

```
<CODE BEGINS>

const uint32     ROPT_RESPXCH = 3;

struct optinfo_respxch {
        xcharsubset     optrespxch_done;
        xcharsubset     optrespxch_rej;
        xcharsubset     optrespxch_pend;
};

<CODE ENDS>
```

The rdma_xid field of this message must match that used in the ROPT_REQXCH message to which this message is responding.

The optrespxch_done field indicates which of the requested transport characteristic changes have been immediately effected. For each such characteristic, the receiver is entitled to conclude that the requested change has been made and that future transmissions may be made based on the new value.

The optrespxch_rej field indicates which of the requested transport characteristic changes have been rejected by the sender. This may be because of any of the following reasons:

- The particular characteristic specified is not known or supported by the receiver of the ROPT_REQXCH message.

- The implementation receiving the ROPT_REQXCH message does not support modification of this characteristic.

- The implementation receiving the ROPT_REQXCHG message has rejected the modification for another reason.

The optrespxch_pend field indicates which of the requested transport characteristic modifications remain pending, since they were neither rejected nor effected immediately. The receiver can expect the modification to be effected by a later ROPT_UPDXCH message, although there is no way to determine when this will happen. For each characteristic bit set in this field, one or more ROPT_UPXCH can be expected, the last of which will have optupdxch_pendclr flag set.

The subsets of characteristics specified by optrespxch_done, optrespxch_rej, optrespxch_pend should not overlap and, when ored together, should cover the entire set of characteristics specified by optreqxch_want in the corresponding request.

4.4. ROPT_UPDXCH: Update Transport Characteristics

The ROPT_UPDXCH message type allows an RPC-over-RDMA participant to notify the other participant that a change to the transport characteristics has occurred.

This may be because:

- A change requested by a ROPT_REQXCH message, has, after some delay, been effected.

- The sender has decided, independently, to modify the transport characteristic and is notifying the receiver of this change.

One should pay particular attention to the fact that there is no there no way to tie a message reporting a change to the specific request which asked for the change. In particular, the rdma_xid field in this message is independent of that for any earlier ROPT_REQXCH message.

The message definition for this operation is as follows:

```
<CODE BEGINS>

const uint32     ROPT_UPDXCH = 4;

struct optinfo_updxch {
        xcharval        optupdxch_now;
        bool            optupdxch_pendclr;
};

<CODE ENDS>
```

optupdxch_now defines the new characteristic value to be used.

optupdxch_pendclr, if true, indicates that a previous request to update the characteristic specified by optupdxch_now.xchv_which is no longer to be considered pending. This may be set true even if the characteristic value is not changed from the previous value.

Some instances of ROPT_UPDXCH are the result of a previous a previous ROPT_REQXCH while others are unsolicited. This distinction relates to the setting of optupdxch_pendclr as follows:

- If a characteristic update is unsolicited, then optupdxch_pendclr will always be false.

- If a characteristic update is prompted by a previous ROPT_REQXCH and optupdxch_pendclr is true, then the current message indicates the (asynchronous) completion of that previous change request.

  In this case the disposition of the change request can be determined using optupdxch_now. If the value is that requested by the associated ROPT_REQXCH then the request was successful, while if the value is unchanged from the original value, the change can be considered rejected.

  In cases in which the characteristic has a range of values, intermediate value are possible, indicating a partial satisfaction of the original request.

- If a characteristic update is prompted by a previous ROPT_REQXCH and optupdxch_pendclr is false, then the current message does not indicate completion of a previous change request.

  In such cases, the characteristic value indicates the current value of the characteristic, which the receiver is entitled to rely upon, just as would have been the case if the change had been unsolicited.

Nevertheless, the change request is still active and will remain so until a ROPT_UPDXCH with optupdxch_pendclr is received.

5. XDR

This section contains an XDR [RFC4506] description of the proposed extension.

This description is provided in a way that makes it simple to extract into ready-to-use form. The reader can apply the following shell script to this document to produce a machine-readable XDR description of extension which can be combined with XDR for the base protocol to produce an XDR that combines the base protocol with the optional extensions.

```
<CODE BEGINS>

#!/bin/sh
grep '^ *///' | sed 's?^ /// ??' | sed 's?^ *///$??'

<CODE ENDS>
```

That is, if the above script is stored in a file called "extract.sh" and this document is in a file called "ext.txt" then the reader can do the following to extract an XDR description file for this extension:

```
<CODE BEGINS>

sh extract.sh < ext.txt > charext.x

<CODE ENDS>
```

5.1. Code Component License

Code components extracted from this document must include the following license text. When the extracted XDR code is combined with other complementary XDR code which itself has an identical license, only a single copy of the license text need be preserved.

```
   <CODE BEGINS>

   /// /*
   ///  * Copyright (c) 2010, 2016 IETF Trust and the persons
   ///  * identified as authors of the code.  All rights reserved.
   ///  *
   ///  * The author of the code is: D. Noveck.
   ///  *
   ///  * Redistribution and use in source and binary forms, with
   ///  * or without modification, are permitted provided that the
   ///  * following conditions are met:
   ///  *
   ///  * - Redistributions of source code must retain the above
   ///  *   copyright notice, this list of conditions and the
   ///  *   following disclaimer.
   ///  *
   ///  * - Redistributions in binary form must reproduce the above
   ///  *   copyright notice, this list of conditions and the
   ///  *   following disclaimer in the documentation and/or other
   ///  *   materials provided with the distribution.
   ///  *
   ///  * - Neither the name of Internet Society, IETF or IETF
   ///  *   Trust, nor the names of specific contributors, may be
   ///  *   used to endorse or promote products derived from this
   ///  *   software without specific prior written permission.
   ///  *
   ///  *   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS
   ///  *   AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
   ///  *   WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
   ///  *   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
   ///  *   FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO
   ///  *   EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
   ///  *   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
   ///  *   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
   ///  *   NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
   ///  *   SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   ///  *   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
   ///  *   LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
   ///  *   OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
   ///  *   IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
   ///  *   ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
   ///  */

   <CODE ENDS>
```

5.2. XDR Proper for Extension

```
   <CODE BEGINS>

   ///
   ////*
   /// * Basic transport characteristic types
   /// */
   ///typedef xcharid         uint32;
   ///
   ///struct xcharval {
   ///        xcharid         xchv_which;
   ///        opaque          xchv_data<>;
   ///};
   ///
   ///typedef xcharspec       xcharval<>;
   ///
   ///typedef xcharsubset     uint32<>;
   ///
   ////*
   /// * Transport characteristic codes
   /// */
   ///const uint32    XCHAR_RBSIZ = 1;
   ///const uint32    XCHAR_REQREMINV = 2;
   ///const uint32    XCHAR_BRS = 3;
   ///
   ////*
   /// * Other transport characteristic types
   /// */
   ///enum bkreqsup {
   ///        BKREQSUP_NONE    = 0,
   ///        BKREQSUP_INLINE  = 1,
   ///        BKREQSUP_GENL    = 2
   ///};
   ///
   ////*
   /// * Transport characteristic typedefs
   /// */
   ///typedef uint32   xchrbsiz;
   ///typedef bool     xchrreqrem;
   ///typedef bkreqsup xchrbrs;
   ///
   ////*
   /// * Optional operation codes
   /// */
   ///const uint32     ROPT_INITXCH = 1;
   ///const uint32     ROPT_REQXCH = 2;
```

```
   ///const uint32     ROPT_RESPXCH = 3;
   ///const uint32     ROPT_UPDXCH = 4;
   ///
   ////*
   /// * Optional operation message structures
   /// */
   ///struct optinfo_initxch {
   ///        xcharspec       optixch_start;
   ///        xcharsubset     optixch_nochg;
   ///};
   ///
   ///struct optinfo_reqxch {
   ///        xcharspec       optreqxch_want;
   ///};
   ///
   ///struct optinfo_respxch {
   ///        xcharsubset     optrespxch_done;
   ///        xcharsubset     optrespxch_rej;
   ///        xcharsubset     optrespxch_pend;
   ///};
   ///
   ///struct optinfo_updxch {
   ///        xcharval        optupdxch_now;
   ///        bool            optupdxch_pendclr;
   ///};

   <CODE ENDS>
```

6. Extensibility

6.1. Additional Characteristics

The set of transport characteristics is designed to be extensible. As a result, once new characteristics are defined in standards track documents, the operations defined in this document may reference these new transport characteristics, as well as the ones described in this document.

A standards track document defining a new transport characteristic should include the following information paralleling that provided in this document for the transport characteristics defined herein.

- The xcharid value used to identify this characteristic.

- The XDR typedef specifying the form in which the characteristic value is communicated.

- A description of the transport characteristic that is communicated by the sender of ROPT_INITXCH and ROPT_UPDXCH and requested by the sender of ROP_REQXCH.

- An explanation of how this knowledge could be used by the participant receiving this information.

- Information giving rules governing possible changes of values of this characteristic.

The definition of transport characteristic structures is such as to make it easy to assign unique values. There is no requirement that a continuous set of values be used and implementations should not rely on all such values being small integers. A unique value should be selected when the defining document is first published as an internet draft. When the document becomes a standards track document working group should insure that:

- The xcharids specified in the document do not conflict with those currently assigned or in use by other pending working group documents defining transport characteristics.

- The xcharids specified in the document do not conflict with the range reserved for experimental use, as defined in Section 6.2.

Documents defining new characteristics fall into a number of categories.

- Those defining new characteristics and explaining (only) how they affect use of existing message types.

- Those defining new OPTIONAL message types and new characteristics applicable to the operation of those new message types.

- Those defining new OPTIONAL message types and new characteristics applicable both to new and existing message types.

When additional transport characteristics are proposed, the review of the associated standards track document should deal with possible security issues raised by those new transport characteristics.

6.2. Experimental Characteristics

Given the design of the transport characteristics data structure, it possible to use the operations to implement experimental, possibly unpublished, transport characteristics.

xcharids in the range from 4,294,967,040 to 4,294,967,295 are reserved for experimental use and these values should not be assigned to new characteristics in standards track documents.

When values in this range are used there is no guarantee if successful interoperation among independent implementations.

7. Security Considerations

Like other fields that appear in each RPC-over-RDMA header, characteristic information is sent in the clear on the fabric with no integrity protection, making it vulnerable to man-in-the-middle attacks.

For example, if a man-in-the-middle were to change the value of the Receive buffer size or the Requester Remote Invalidation boolean, it could reduce connection performance or trigger loss of connection. Repeated connection loss can impact performance or even prevent a new connection from being established. Recourse is to deploy on a private network or use link-layer encryption.

8. IANA Considerations

This document does not require any actions by IANA.

9. References

9.1. Normative References

[bidir]   Lever, C., "Size-Limited Bi-directional Remote Procedure Call On Remote Direct Memory Access Transports", April 2016, <http://www.ietf.org/id/draft-ietf-nfsv4-rpcrdma-bidirection-02.txt>.

Work in progress.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4506]   Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, DOI 10.17487/RFC4506, May 2006, <http://www.rfc-editor.org/info/rfc4506>.

[rfc5666bis]
Lever, C., Ed., Simpson, W., and T. Talpey, "Remote Direct Memory Access Transport for Remote Procedure Call", May 2016, <http://www.ietf.org/id/draft-ietf-nfsv4-rfc5666bis-07.txt>.

Work in progress.

[rpcrdmav2]
Lever, C., Ed. and D. Noveck, "RPC-over-RDMA Version Two", June 2016, <http://www.ietf.org/id/draft-cel-nfsv4-rpcrdma-version-two-01.txt>.

Work in progress.

9.2. Informative References

[RFC5662] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 External Data Representation Standard (XDR) Description", RFC 5662, DOI 10.17487/RFC5662, January 2010, <http://www.rfc-editor.org/info/rfc5662>.

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

Appendix A. Acknowledgments

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 resurrecting NFS support for RDMA in [rfc5666bis] and for his helpful review of and suggestions for this document.

The extract.sh shell script and formatting conventions were first described by the authors of the NFSv4.1 XDR specification [RFC5662].

Author's Address

David Noveck
Hewlett Packard Enterprise
165 Dascomb Road
Andover, MA 01810
USA

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