

TCPM working group
Internet Draft
Intended Status: Informational
Expires: 11/7/2015

M. Fox
C. Kassimis
J. Stevens
IBM
May 7, 2015

IBM's Shared Memory Communications over RDMA
draft-fox-tcpm-shared-memory-rdma-07.txt

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on November 7, 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.

Abstract

This document describes the IBM's Shared Memory Communications over RDMA (SMC-R) protocol. This protocol provides RDMA communications to TCP endpoints in a manner that is transparent to socket applications. It further provides for dynamic discovery of partner RDMA capabilities and dynamic setup of RDMA connections, transparent high availability and load balancing when redundant RDMA network paths are available, and it maintains many of the traditional TCP/IP qualities of service such as filtering that enterprise users demand, as well as TCP socket semantics such as urgent data.

Table of Contents

1. Introduction.....	5
1.1. Summary of changes in this draft.....	6
1.2. Protocol overview.....	6
1.2.1. Hardware requirements.....	8
1.3. Definition of common terms.....	8
2. Link Architecture.....	10
2.1. Remote Memory Buffers (RMBs).....	12
2.2. SMC-R Link groups.....	16
2.2.1. Link group types.....	17
2.2.2. Maximum number of links in link group.....	20
2.2.3. Forming and managing link groups.....	21
2.2.4. SMC-R link identifiers.....	22
2.3. SMC-R resilience and load balancing.....	23
3. SMC-R Rendezvous architecture.....	25
3.1. TCP options.....	25
3.2. Connection Layer Control (CLC) messages.....	26
3.3. LLC messages.....	26
3.4. CDC Messages.....	28
3.5. Rendezvous flows.....	28
3.5.1. First contact.....	28
3.5.1.1. TCP Options pre-negotiation.....	28
3.5.1.2. Client Proposal.....	29
3.5.1.3. Server acceptance.....	30
3.5.1.4. Client confirmation.....	31
3.5.1.5. Link (QP) confirmation.....	31
3.5.1.6. Second SMC-R link setup.....	34
3.5.1.6.1. Client processing of "Add Link" LLC message from server.....	34
3.5.1.6.2. Server processing of "Add Link" reply LLC message from the client.....	35

3.5.1.6.3. Exchange of Rkeys on second SMC-R link....	37
3.5.1.6.4. Aborting SMC-R and falling back to IP.....	37
3.5.2. Subsequent contact.....	37
3.5.2.1. SMC-R proposal.....	38
3.5.2.2. SMC-R acceptance.....	39
3.5.2.3. SMC-R confirmation.....	40
3.5.2.4. TCP data flow race with SMC Confirm CLC message	40
3.5.3. First contact variation: creating a parallel link group	
.....	41
3.5.4. Normal SMC-R link termination.....	42
3.5.5. Link group management flows.....	43
3.5.5.1. Adding and deleting links in an SMC-R link group	43
3.5.5.1.1. Server initiated Add Link processing.....	43
3.5.5.1.2. Client initiated Add Link processing.....	44
3.5.5.1.3. Server initiated Delete Link Processing...	44
3.5.5.1.4. Client initiated Delete Link request.....	46
3.5.5.2. Managing multiple Rkeys over multiple SMC-R links	
in a link group.....	48
3.5.5.2.1. Adding a new RMB to an SMC-R link group...	49
3.5.5.2.2. Deleting an RMB from an SMC-R link group..	52
3.5.5.2.3. Adding a new SMC-R link to a link group with	
multiple RMBs.....	53
3.5.5.3. Serialization of LLC exchanges, and collisions.	54
3.5.5.3.1. Collisions with ADD LINK / CONFIRM LINK	
exchange.....	56
3.5.5.3.2. Collisions during DELETE LINK exchange....	57
3.5.5.3.3. Collisions during CONFIRM_RKEY exchange...	57
4. SMC-R memory sharing architecture.....	59
4.1. RMB element allocation considerations.....	59
4.2. RMB and RMBE format.....	59
4.3. RMBE control information.....	59
4.4. Use of RMBEs.....	60
4.4.1. Initializing and accessing RMBEs.....	60
4.4.2. RMB element reuse and conflict resolution.....	61
4.5. SMC-R protocol considerations.....	62
4.5.1. SMC-R protocol optimized window size updates.....	62
4.5.2. Small data sends.....	63
4.5.3. TCP Keepalive processing.....	63
4.6. TCP connection failover between SMC-R links.....	66
4.6.1. Validating data integrity.....	66
4.6.2. Resuming the TCP connection on a new SMCR link.....	67
4.7. RMB data flows.....	67
4.7.1. Scenario 1: Send flow, window size unconstrained...	68
4.7.2. Scenario 2: Send/Receive flow, window unconstrained.	70
4.7.3. Scenario 3: Send Flow, window constrained.....	71
4.7.4. Scenario 4: Large send, flow control, full window size	
writes.....	73

4.7.5. Scenario 5: Send flow, urgent data, window size unconstrained.....	76
4.7.6. Scenario 6: Send flow, urgent data, window size closed	78
4.8. Connection termination.....	80
4.8.1. Normal SMC-R connection termination flows.....	80
4.8.1.1. Abnormal SMC-R connection termination flows....	85
4.8.1.2. Other SMC-R connection termination conditions..	87
5. Security considerations.....	88
5.1. VLAN considerations.....	88
5.2. Firewall considerations.....	88
5.3. Host-based IP Filters.....	89
5.4. Intrusion Detection Services.....	89
5.5. IP Security (IPSec).....	89
5.6. TLS/SSL.....	89
6. IANA considerations.....	89
7. References.....	90
7.1. Normative References.....	90
7.2. Informative References.....	90
8. Acknowledgments.....	90
9. Conventions used in this document.....	90
Appendix A. Formats.....	91
A.1. TCP option.....	91
A.2. CLC messages.....	91
A.2.1. Peer ID format.....	91
A.2.2. SMC Proposal CLC message format.....	93
A.2.3. SMC Accept CLC message format.....	96
A.2.4. SMC Confirm CLC message format.....	99
A.2.5. SMC Decline CLC message format.....	102
A.3. LLC messages.....	103
A.3.1. CONFIRM LINK LLC message format.....	104
A.3.2. ADD LINK LLC message format.....	106
A.3.3. ADD LINK CONTINUATION LLC message format.....	108
A.3.4. DELETE LINK LLC message format.....	111
A.3.5. CONFIRM RKEY LLC message format.....	113
A.3.6. CONFIRM RKEY CONTINUATION LLC message format.....	116
A.3.7. DELETE RKEY LLC message format.....	118
A.3.8. TEST LINK LLC message format.....	120
Appendix B. Socket API considerations.....	126
Appendix C. Rendezvous Error scenarios.....	128
C.1. SMC Decline during CLC negotiation.....	128
C.2. SMC Decline during LLC negotiation.....	128
C.3. The SMC Decline window.....	130
C.4. Out of synch conditions during SMC-R negotiation.....	130
C.5. Timeouts during CLC negotiation.....	131
C.6. Protocol errors during CLC negotiation.....	131
C.7. Timeouts during LLC negotiation.....	132
C.7.1. Recovery actions for LLC timeouts and failures.....	133

C.8. Failure to add second SMC-R link to a link group.....140

1. Introduction

This document specifies IBM's Shared Memory Communications over RDMA (SMC-R) protocol. SMC-R is a protocol for Remote Direct Memory Access (RDMA) communication between TCP socket endpoints. SMC-R runs over networks that support RDMA over Converged Ethernet (RoCE). It is designed to permit existing TCP applications to benefit from RDMA without requiring modifications to the applications or predefinition of RDMA partners.

SMC-R provides dynamic discovery of the RDMA capabilities of TCP peers and automatic setup of RDMA connections that those peers can use. SMC-R also provides transparent high availability and load balancing capabilities that are demanded by enterprise installations but are missing from current RDMA protocols. If redundant RoCE capable hardware such as RDMA NICs (RNICs) and RoCE capable switches is present, SMC-R can load balance over that redundant hardware and can also non-disruptively move TCP traffic from failed paths to surviving paths, all seamlessly to the application and the sockets layer. Because SMC-R preserves socket semantics and the TCP three-way handshake, many TCP qualities of service such as filtering, load balancing, and SSL encryption are preserved, as are TCP features such as urgent data.

Because of the dynamic discovery and setup of SMC-R connectivity between peers, no RDMA connection manager (RDMA-CM) is required. This also means that support for UD queue pairs is also not required.

It is recommended that the SMC-R services be implemented in kernel space, which enables optimizations such as resource sharing between connections across multiple processes and also permits applications using SMC-R to spawn multiple processes (e.g. fork) without losing SMC-R functionality. A user space implementation is compatible with this architecture, but it may not support spawned processes (i.e. fork) which limits sharing and resource optimization to TCP connections that originate from the same process. This might be an appropriate design choice if the use case is a system that hosts a large single process application that creates many TCP connections to a peer host, or in implementations where a kernel space implementation is not possible or introduces excessive overhead for kernel space to user space context switches.

1.1. Summary of changes in this draft

Changed the title to add "IBM's".

1.2. Protocol overview

SMC-R defines the concept of the SMC-R Link, which is a logical point-to-point link using reliably connected queue pairs between TCP/IP stack peers over a RoCE fabric. An SMC-R link is bound to a specific hardware path, meaning a specific RNIC on each peer. SMC-R links are created and maintained by an SMC-R layer, which may reside in kernel or user space depending upon operating system and implementation requirements. The SMC-R layer resides below the sockets layer and directs data traffic for TCP connections between connected peers over the RoCE fabric using RDMA rather than over a TCP connection. The TCP/IP stack with its fragmentation, packetization, etc. requirements is bypassed and the application data is moved between peers using RDMA.

Multiple SMC-R links between the same two TCP/IP stack peers are also supported. A set of SMC-R links called a link group can be logically bonded together to provide redundant connectivity. If there is redundant hardware, for example two RNICs on each peer, separate SMC-R links are created between the peers to exploit that redundant hardware. The link group architecture with redundant links provide load balancing, increased bandwidth as well as seamless failover.

Each SMC-R link group is associated with an area of memory called Remote Memory Buffers (RMBs), which are areas of memory that are available for SMC-R peers to write into using RDMA writes. Multiple TCP connections between peers may be multiplexed over a single SMC-R link, in which case the SMC-R layer manages the partitioning of the RMBs between the TCP connections. This multiplexing reduces the RDMA resources such as queue pairs and RMBs that are required to support multiple connections between peers, and also reduces the processing and delays related to setting up queue pairs, pinning memory, and other RDMA setup tasks when new TCP connections are created. In a kernel space SMC-R implementation in which the RMBs reside in kernel storage, this sharing and optimization works across multiple processes executing on the same host. In a user space SMC-R implementation in which the RMBs reside in user space, this sharing

and optimization is limited to multiple TCP connections created by a single process, as separate RMBs and QPs will be required for each process.

SMC-R also introduces a rendezvous protocol that is used to dynamically discover the RDMA capabilities of TCP connection partners and exchange credentials necessary to exploit that capability if present. TCP connections are set up using the normal TCP 3-way handshake, with the addition of a new TCP option that indicates SMC-R capability. If both partners indicate SMC-R capability then at the completion of the 3-way TCP handshake the SMC-R layers in each peer take control of the TCP connection and use it to exchange additional connection level control (CLC) messages to negotiate SMC-R credentials such as queue pair (QP) information, addressability over the RoCE fabric, RMB buffer sizes, keys and addresses for accessing RMBs over RDMA, etc. If at any time during this negotiation a failure or decline occurs, the TCP connection falls back to using the IP fabric.

If the SMC-R negotiation succeeds and either a new SMC-R link is set up or an existing SMC-R link is chosen for the TCP connection, then the SMC-R layers open the sockets to the applications and the applications use the sockets as normal. The SMC-R layer intercepts the socket reads and writes and moves the TCP connection data over the SMC-R link, "out of band" to the TCP connection which remains open and idle over the IP fabric, except for termination flows and possible keepalive flows. Regular TCP sequence numbering methods are used for the TCP flows that do occur; data flowing over RDMA does not use or affect TCP sequence numbers.

This architecture does not support fallback of active SMC-R connections to IP. Once connection data has completed the switch to RDMA, a TCP connection cannot be switched back to IP and will reset if RDMA becomes unusable.

The SMC-R protocol defines the format of the Remote Memory Buffers that are used to receive TCP connection data written over RDMA, as well as the semantics for managing and writing to these buffers using Connection Data Control (CDC) messages.

Finally, SMC-R defines link level control (LLC) messages that are exchanged over the RoCE fabric between peer SMC-R layers to manage the SMC-R links and link groups. These include messages to test and confirm connectivity over an SMC-R link, add and delete SMC-R links to or from the link group, and exchange RMB addressability information.

1.2.1. Hardware requirements

SMC-R does not require full Converged Enhanced Ethernet switch functionality. SMC-R functions over standard Ethernet fabrics provided endpoint RNICs are provided and IEEE 802.3x Global Pause Frame is supported and enabled in the switch fabric.

While SMC-R as specified in this document is designed to operate over RoCE fabrics, adjustments to the rendezvous methods could enable it to run over other RDMA fabrics such as Infiniband and iWARP.

1.3. Definition of common terms

This section provides definitions of terms that have a specific meaning to the SMC-R protocol and are used throughout this document.

SMC-R link

An SMC-R Link is a logical point to point connection over the RoCE fabric via specific physical adapters (MAC/GID). The Link is formed during the first contact sequence of the TCP/IP 3 way handshake sequence that occurs over the IP fabric. During this handshake an RDMA RC-QP connection is formed between the two peer SMC hosts and is defined as the SMC Link. The SMC Link can then support multiple TCP connections between the two peers. An SMC link is associated with a single LAN (or VLAN) segment and is not routable.

SMC-R link group

An SMC-R Link Group is a group of SMC-R Links typically each over unique RoCE adapters between the same two SMC-R peers. Each link in the link group has equal characteristics such as the same VLAN ID (if VLANs are in use), access to the same RMB(s) and the same TCP server / client

SMC-R peer

The SMC-R Peer is the peer software stack within the peer Operating System with respect the Shared Memory Communications (messaging) protocol.

SMC-R Rendezvous

The SMC-R Rendezvous is the SMC-R peer discovery and handshake sequence that occurs transparently over the IP (Ethernet) fabric during and immediately after the TCP connection 3 way handshake

by exchanging the SMC capabilities and credentials using experimental TCP option and CLC messages.

TCP Client

The TCP socket-based peer that initiates a TCP connection

TCP Server

The TCP socket-based peer that accepts a TCP connection

CLC messages

The SMC-R protocol defines a set of Connection Layer Control Messages that flow over the TCP connection that are used to manage SMC link rendezvous at TCP connection setup time. This mechanism is analogous to SSL setup messages

LLC Commands

The SMC-R protocol defines a set of RoCE Link Layer Control Commands that flow over the RoCE fabric using RDMA sendmsg, that are used to manage SMC Links, SMC Link Groups and SMC Link Group RMB expansion and contraction.

CDC message

The SMC-R protocol defines a Connection Data Control message that flows over the RoCE fabric using RDMA sendmsg that is used to manage the SMC-R connection data. This message provides information about data being transferred over the out of band RDMA connection, such as data cursors, sequence numbers, and data flags (for example urgent data). The receipt of this message also provides an interrupt to inform the receiver that it has received RDMA data.

RMB

A Remote (RDMA) Memory Buffer is a fixed or pinned buffer allocated in each of the peer hosts for a TCP (via SMC-R) connection. The RMB is registered to the RNIC and allows remote access by the remote peer using RDMA semantics. Each host is passed the peer's RMB specific access information (RKey and RMB Element offset) during the SMC-R rendezvous process. The host stores socket application user data directly into the peer's RMB using RDMA over RoCE.

Rtoken

The combination of an RMB's Rkey and RDMA virtual addressing, an Rtoken provides addressability to an RMB to an RDMA peer

RMBE

The Remote Memory Buffer Element is an area of an RMB that is allocated to a specific TCP connection. The RMBE contains data for the TCP connection. The RMBE represents the TCP receive buffer whereby the remote peer writes into the RMBE and the local peer reads from the local RMBE. The alert token resolves to a specific RMBE.

Alert Token

The SMC-R alert token is a four byte value that uniquely identifies the TCP connection over an SMC-R connection. The alert token allows the SMC peer to quickly identify the target TCP connection that now has new work. The format of the token is defined by the owning SMC-R end point and is considered opaque to the remote peer. However the token should not simply be an index to an RMBE element; it should reference a TCP connection and be able to be validated to avoid reading data from stale connections.

RNIC

The RDMA capable Network Interface Card (RNIC) is an Ethernet NIC that supports RDMA semantics and verbs using RoCE.

First Contact

Describes an SMC-R negotiation to set up the first link in a link group

Subsequent Contact

Describes an SMC-R negotiation between peers who are using an already existing SMC-R link group

2. Link Architecture

An SMC-R link is based on reliably connected queue pairs (QPs) that form a "logical point to point link" between the two SMC-R peers over a RoCE fabric. An SMC-R link extends from SMC-R peer to SMC-R peer,

where typically each peer would be a TCP/IP stack would reside on separate hosts.

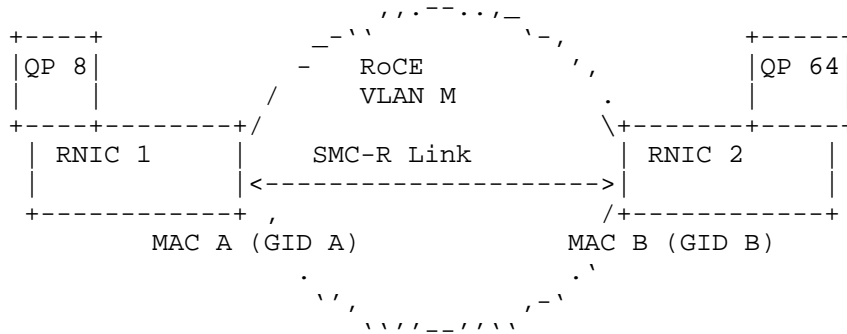


Figure 1 SMC-R Link Overview

Figure 1 illustrates an overview of the basic concepts of SMC-R peer to peer connectivity which is called the SMC-R Link. The SMC-R Link forms a logical point to point connection between two SMC-R peers via RoCE. The SMC Link is defined and identified by the following attributes:

SMC-R Link = RC QPs (source VMAC GID QP + target VMAC GID QP + VLAN ID)

The SMC-R Link can optionally be associated with a VLAN ID. If VLANs are in use for the associated IP (LAN) connection then the VLAN attribute is carried over on the SMC-R link. When VLANs are in use each SMC-R link group is associated with a single and specific VLAN. The RoCE fabric is the same physical Ethernet LAN used for standard TCP/IP over Ethernet communications, with switches as described in 1.2.1.

An SMC-R Link is designed to support multiple TCP connections between the same two peers. An SMC Link is intended to be long lived while the underlying TCP connections can dynamically come and go. The associated RMBs can also be dynamically added and removed from the link as needed. The first TCP connection between the peers establishes the SMC-R link. Subsequent TCP connections then use the previously established link. When the last TCP connection terminates the link can then be terminated, typically after an implementation defined idle time-out period has elapsed. The TCP server is responsible for initiating and terminating the SMC Link.

2.1. Remote Memory Buffers (RMBs)

Figure 2 shows the hosts X and Y and their associated RMBs within each host. With the SMC-R link and the associated RMB keys (Rkeys) and RDMA virtual addresses each SMC-R enabled TCP/IP stack can remotely access its peer's RMBs using RDMA. The RKeys and virtual addresses are exchanged during the rendezvous processing when the link is established. The combination of the Rkey and the virtual address is the Rtoken. Note that the SMC-R Link ends at the QP providing access to the RMB (via the Link + RToken).

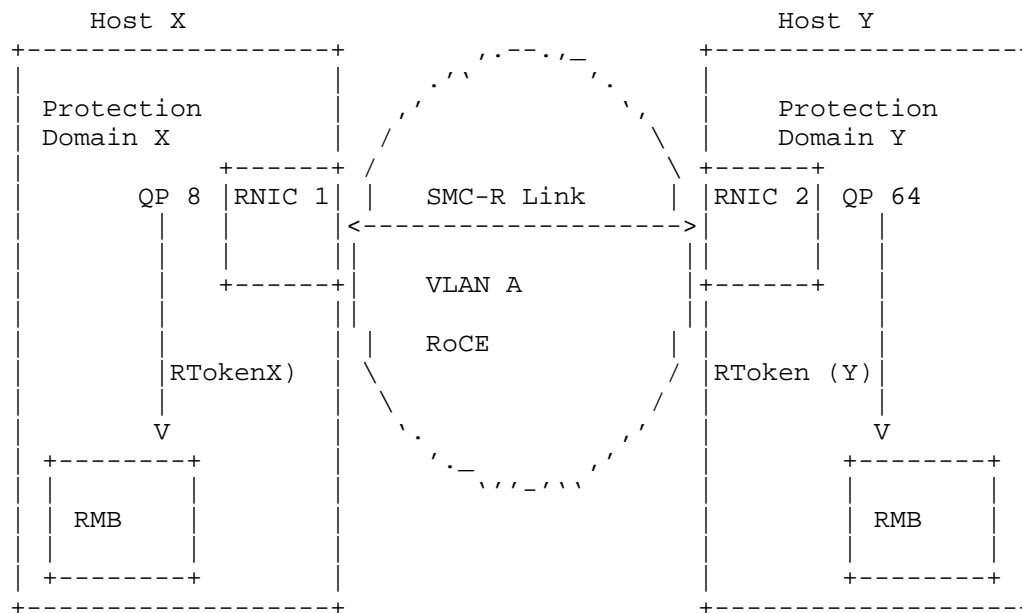


Figure 2 SMC link and RMBs

An SMC-R link can support multiple RMBs which are independently managed by each peer. The number of and the size of RMBs are managed by the peers based on host unique memory management requirements; however the maximum number of RMBs that can be associated to a link group on one peer is 255. The QP has a single protection domain, but each RMB has a unique RToken. All RTokens must be exchanged with the peer.

Each peer manages the RMBs in its local memory for its remote SMC-R peer by sharing access to the RMBs via Rtokens with its peers. The remote peer writes into the RMBs via RDMA and the local peer (RMB owner) then reads from the RMBs.

When two peers decide to use SMC-R for a given TCP connection, they each allocate a local RMB Element for the TCP connection and communicate the location of this local RMB Element during rendezvous processing. To that end, RMB elements are created in pairs, with one RMB element allocated locally on each peer of the SMC-R link.

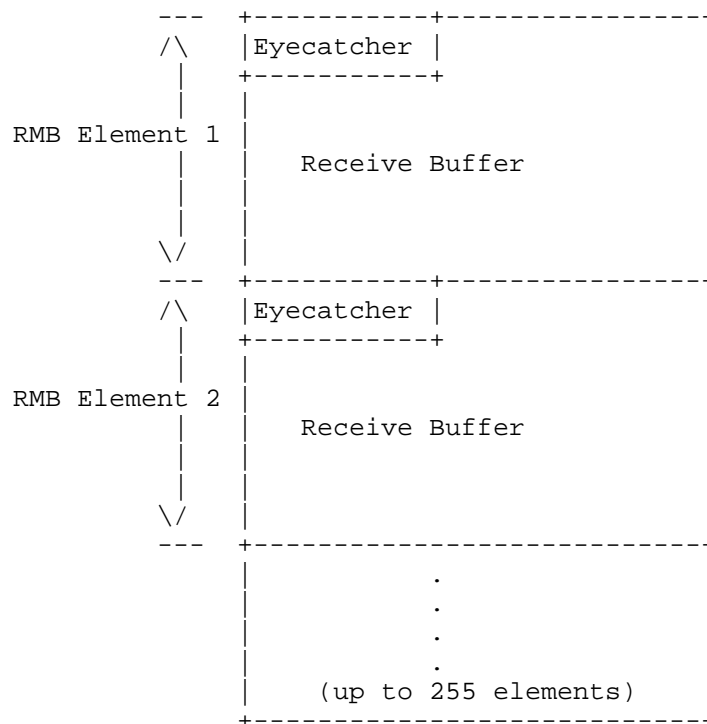


Figure 3 RMB Format

Figure 3 illustrates the basic format of an RMB. The RMB is a virtual memory buffer whose backing real memory is pinned, which can support up to 255 TCP connections to exactly one remote SMC-R peer. Each RMB is therefore associated with the SMC-R links within a link group for the two peers and a specific RoCE Protection Domain. Other than the 2 peers identified by the SMC-R link no other SMC-R peers can have RDMA access to an RMB; this requires a unique Protection Domain for every SMC-R Link. This is critical to ensure integrity of SMC-R communications.

RMBs are subdivided into multiple elements for efficiency, with each RMBE element (RMBE) is associated with a single TCP connection. Therefore multiple TCP connections across an SMC link group can share the same memory for RDMA purposes, reducing the overhead of having to register additional memory with the RNIC for every new TCP connection. The number of elements in an RMB and the size of each RMB Element is entirely governed by the owning peer subject to the SMC-R architecture rules, however, all RMB elements within a given RMB must be the same size. Each peer can decide the level of resource sharing that is desirable across TCP connections based on local constraints such as available system memory, etc. An RMB Element is identified to the remote SMC-R peer via an RMB Element Token which consists of the following:

- o RMB RToken: The combination of the Rkey and virtual address provided by the RNIC that identifies the start of the RMB for RDMA operations.
- o RMB Index: Identifies the RMB element index in the RMB. Used to locate a specific RMB element within an RMB. Valid value range is 1-255.
- o RMB element length: The length of the RMB element's eyecatcher plus the length of receive buffer. This length is equal for all RMB elements in a given RMB. This length can be variable across different RMBs.

Multiple RMBs can be associated to an SMC-R link group and each peer in an SMC-R link group manages allocation of its RMBs. RMB allocation can be asymmetric. For example, server X can allocate 2 RMBs to an SMC-R link group while server Y allocates 5. This provides maximum implementation flexibility to allow hosts optimize RMB management for their own local requirements. The maximum number of RMBs that can be allocated on one peer to a link group is 255. If more RMBs are required, the peer may fall back to IP for subsequent connections or, if the peer is the server, create a parallel link group.

One use case for multiple RMBs is multiple receive buffer sizes. Since every element in an RMB must be the same size, multiple RMBs with different element sizes can be allocated if varying receive buffer sizes are required.

Also since the maximum number of TCP connections whose receive buffers can be allocated to an RMB is 255, multiple RMBs may be required to provide capacity for large numbers of TCP connections between two peers.

Separately from the RMB, the TCP/IP stack that owns each RMB maintains control data for each RMB element within its local control structures. The control data contains flags for maintaining the state of the TCP data (for example, urgent indicator) and most importantly, two cursors which are illustrated in Figure 4:

- o The peer producer cursor: This is a wrapping offset into the RMB element's receive buffer that points to the next byte of data to be written by the remote peer. This cursor is provided by the remote peer in a Connection Data Control (CDC message), which is sent using RDMA sendmsg processing, and tells the local peer how far it can consume data in the RMBE buffer.
- o The peer consumer cursor: This is a wrapping offset into the remote peer's RMB element's receive buffer that points to the next byte of data to be consumed by the remote peer in its own RMBE. The local cannot write into the remote peer's RMBE beyond this point without causing data loss. This cursor is also provided by the peer using a Connection Data Control message.

Each TCP connection peer maintains its cursors for a TCP connection's RMBE in its local control structures. In other words, the peer who writes into a remote peer's RMBE provides its producer cursor to the peer whose RMBE it has written into. The peer who reads from its RMBE provides its consumer cursor to the writing peer. In this manner the reads and writes between peers are kept coordinated.

For example, referring to Figure 4, peer B writes the hashed data into the receive buffer of peer A's RMBE. After that write completes, peer B uses a CDC message to update its producer cursor to peer A, to indicate to peer A how much data is available for peer A to consume. The CDC message that peer B sends to peer A wakes up peer A and notifies it that there is data to be consumed.

Similarly, when peer A consumes data written by peer B, it uses a CDC message to update its consumer cursor to peer B to let peer B know how much data it has consumed, so peer B knows how much space is available for further writes. If peer B were to write enough data to peer A that it would wrap the RMBE receive buffer and exceed the consumer cursor, data loss would result.

Note that this is a simplistic description of the control flows and they are optimized to minimize the number of CDC messages required, as described in 4.7. RMB data flows.

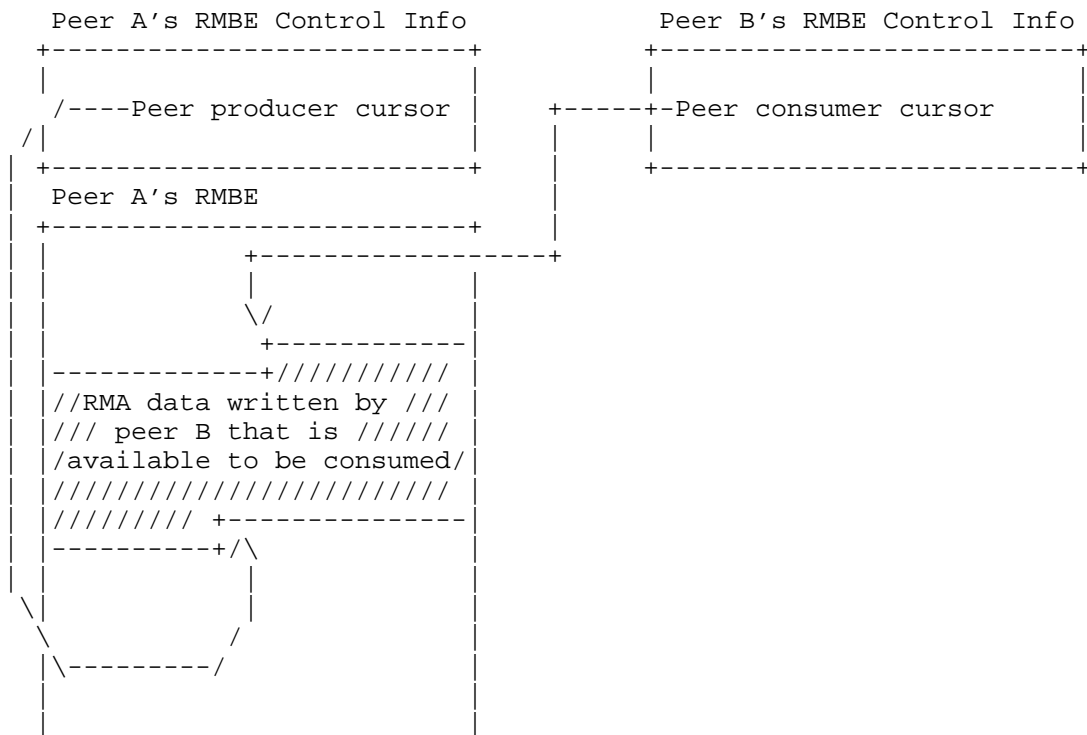


Figure 4 RMBE cursors

Additional flags and indicators are communicated between peers. In all cases, these flags and indicators are updated by the peer using CDC messages with the control information contained in inline data. More details on these additional flags and indicators are described in [4.3. RMBE control information](#).

2.2. SMC-R Link groups

SMC-R links are logically grouped together to form an SMC-R Link Group. The purpose of the Link Group is for supporting multiple links between the same two peers to provide for:

- o Resilience: Provides transparent and dynamic switching of the link used by existing TCP connections during link failures, typically hardware related. TCP traffic using the failing link can be switched to an active link within the link group avoiding disruptions to application workloads.

- o Link utilization: Provides an active/active link usage model allowing TCP traffic to be balanced across the links, which increases bandwidth and avoids hardware imbalances and bottlenecks. Note that both adapter and switch utilization can become potential resource constraint issues

SMC-R Link Group support is required. Resilience is not optional. However, the user can elect to provision a single RNIC (on one or both hosts).

Multiple links that are formed between the same two peers fall into two distinct categories:

1. Equal Links: Links providing equal access to the same RMB(s) at both endpoints whereby all TCP connections associated with the links must have the same VLAN ID and have the same TCP server and TCP client roles or relationship.
2. Unequal Links: Links providing access to unique, unrelated and isolated RMB(s) (i.e. for unique VLANs or unique and isolated application workloads, etc.) or have unique TCP server or client roles.

Links that are logically grouped together forming an SMC Link Group must be equal links.

2.2.1. Link group types

Equal links within a link group also have another "Link Group Type" attribute based on the link's associated underlying physical path. The following SMC-R link types are defined:

1. Single Link: the only active link within a link group
2. Parallel Link: not allowed - SMC Links having the same physical RNIC at both hosts
3. Asymmetric Link: links that have unique RNIC adapters at one host but share a single adapter at the peer host
4. Symmetric Link: links that have unique RNIC adapters at both hosts

These link group types are further explained in the following figures and descriptions.

Figure 2 above shows the single link case. The single link illustrated in Figure 2 also establishes the SMC-R Link Group. Link groups are supposed to have multiple links, but when only one RNIC is available at both hosts then only a single link can be created. This is expected to be a transient case.

Figure 5 shows the symmetric link case. Both hosts have unique and redundant RNIC adapters. This configuration meets the objectives for providing full RoCE redundancy required to provide the level of resilience required for high availability for SMC-R. While this configuration is not required, it is a strongly recommended "best practice" for the exploitation of SMC-R. Single and asymmetric links must be supported but are intended to provide for short term transient conditions, for example during a temporary outage or recycle of a RNIC.

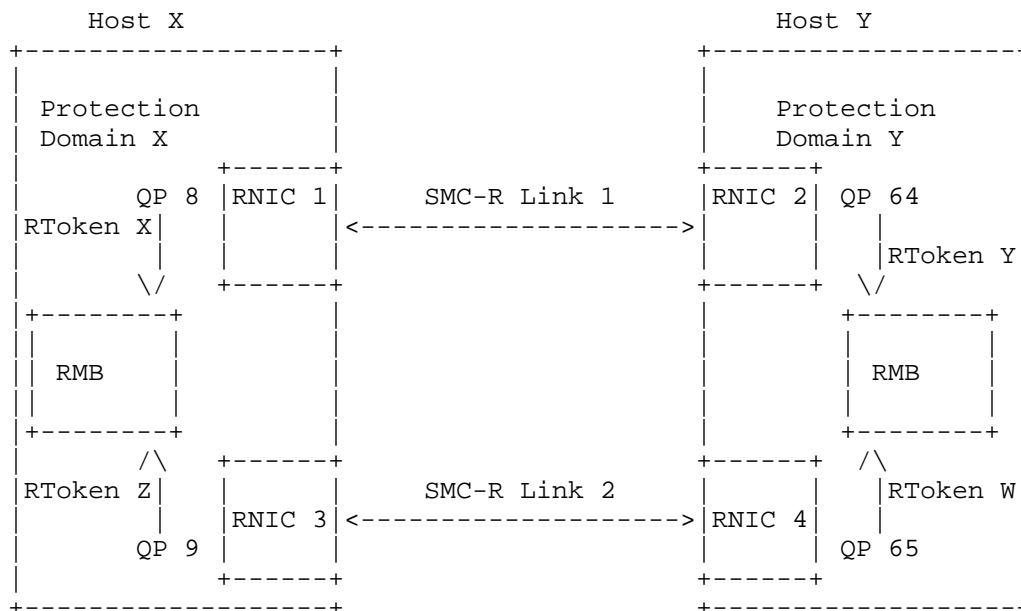


Figure 5 Symmetric SMC-R links

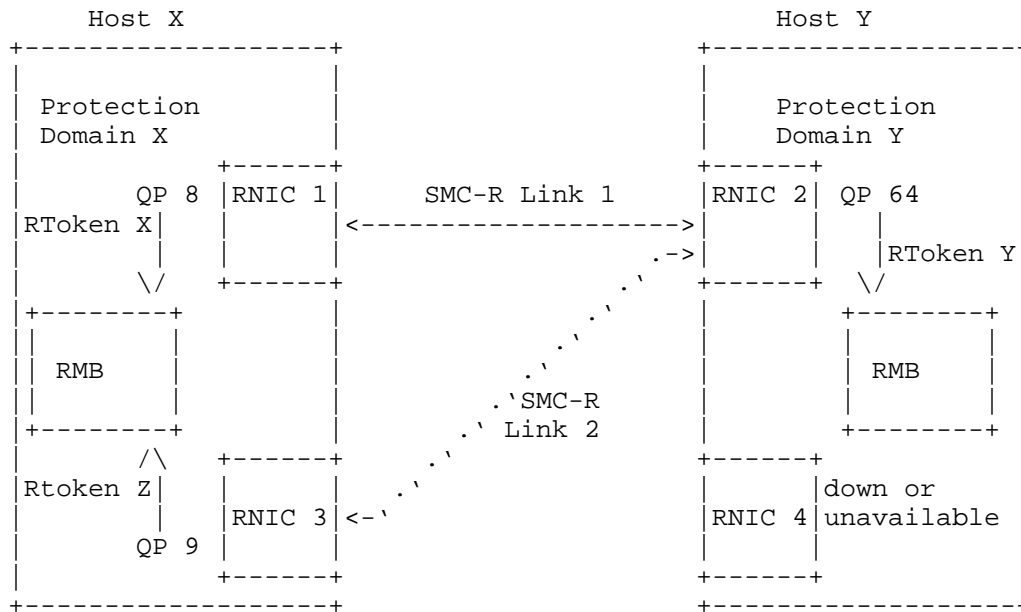


Figure 6 Asymmetric SMC-R links

In the example provided by Figure 6, host X has two RNICs but Host Y only has one RNIC. This configuration allows for the creation of an asymmetric link. While an asymmetric link will provide some resilience (i.e. when RNIC 1 fails) ideally each host should provide two redundant RNICs. This should be a transient case, and when RNIC 4 becomes available, this configuration must transition to a symmetric link configuration. This transition is accomplished by first creating the new symmetric link, then deleting the asymmetric link with reason code "Asymmetric link no longer needed" specified in the DELETE LINK LLC message.

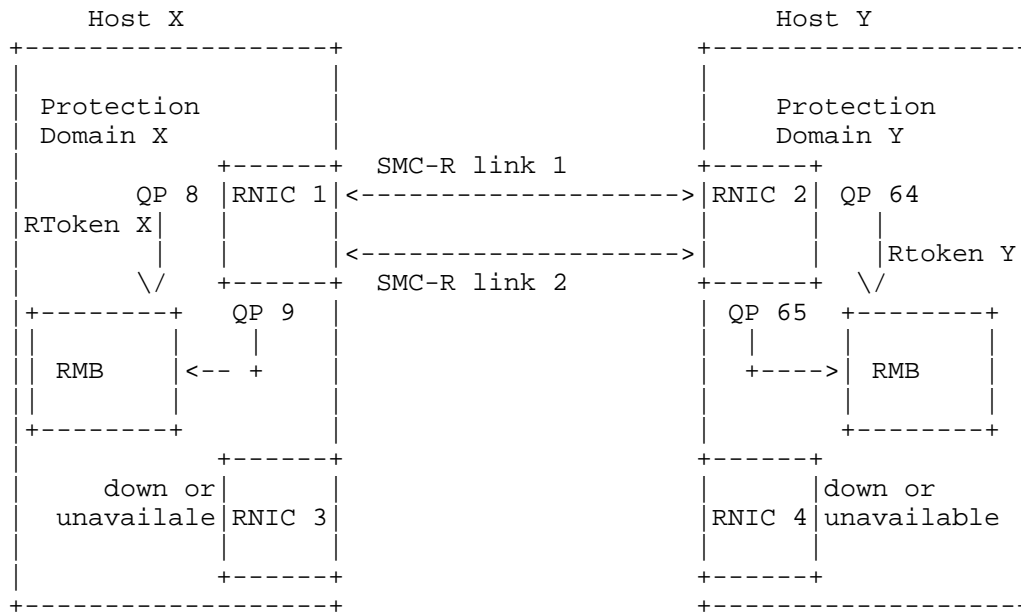


Figure 7 SMC-R parallel links (not supported)

Figure 7 shows parallel links, which are two links in the link group that use the same hardware. This configuration is not permitted. Because SMC-R multiplexes multiple TCP connections over an SMC-R link and both links are using the exact same hardware, there is no additional redundancy or capacity benefit obtained from this configuration. However this configuration does add unnecessary overhead of additional queue pairs, generation of additional Rkeys, etc.

2.2.2. Maximum number of links in link group

The SMC-R protocol defines a maximum of 8 symmetric SMC-R links within a single SMC-R link group. This allows for support for up to 8 unique physical paths between peer hosts. However, in terms of meeting the basic requirements for redundancy support for at least 2 symmetric links must be implemented. Supporting greater than 2 links also simplifies implementation for practical matters relating to dynamically adding and removing links, for example starting a third SMC-R link prior to taking down one of the two existing links. Recall that all links within a link group must have equal access to all associated RMBs.

The SMC-R protocol allows an implementation to implement an implementation specific and appropriate value for maximum symmetric links. The implementation value must not exceed the architecture limit of 8 and the implementation must not be lower than 2, because the SMC-R protocol requires redundancy. This does not mean that two RNICs are physically required to enable SMC-R connectivity, but at least two RNICs for redundancy are strongly recommended.

The SMC-R peers exchange their implementation maximum link values during the link group establishment using the defined maximum link value in the CONFIRM LINK LLC command. Once the initial exchange completes the value is set for the life of the link group. The maximum link value can be provided by both the server and client. The server must supply a value, whereas the client maximum link value is optional. When the client does not supply a value, it indicates that the client accepts the server supplied maximum value. If the client provides a value it can not exceed the server maximum value. If the client passes a lower value then this lower value then becomes the final negotiated maximum number of symmetric links for this link group. Again, the minimum value is 2.

During run time the client must never request that the server add a symmetric link to a link group that would exceed the negotiated maximum link value. Likewise the server must never attempt to add a symmetric link to a link group that would exceed the negotiated maximum value.

In terms of counting the active link count within a link group, the initial link (or the only / last) link is always counted as 1. Then as additional links are added they are either symmetric or asymmetric links.

With regards to enforcing the maximum link rules, asymmetric links are an exception having a unique set of rules:

- o Asymmetric links are always limited to one asymmetric link allowed per link group
- o Asymmetric links must not be counted in the maximum symmetric link count calculation. When tracking the current count or enforcing the negotiated maximum number of links, an asymmetric link is not to be counted

2.2.3. Forming and managing link groups

SMC-R link groups are self-defining. The first SMC-R link in a link group is created using TCP option flows on the TCP three-way

handshake followed by CLC message flows over the TCP connection. Subsequent SMC-R links in the link group are created by sending LLC messages over an SMC-R link that already exists in the link group. Once an SMC-R link group is created, no additional SMC-R links in that group are created using TCP and CLC negotiation. Because subsequent SMC-R links are created exclusively by sending LLC messages over an existing SMC-R link in a link group, the membership of SMC-R links to a link group is self-defining.

This architecture does not define a specific identifier for an SMC-R link group. This identification may be useful for network management and may be assigned in a platform specific manner, or in an extension to this architecture.

In each SMC-R link group, one peer is the server for all TCP connections and the other peer is the client. If there are additional TCP connections between the peers that use SMC-R and have the client and server roles reversed, another SMC-R link group is set up between them with the opposite client-server relationship.

This is required because there are specific responsibilities divided between the client and server in the management of an SMC-R link group.

In this architecture, the decision of whether or not to use an existing SMC-R link group or create a new SMC-R link group for a TCP connection is made exclusively by the server.

Management of the links in an SMC-R link group is also a server responsibility. The server is responsible for adding and deleting links in a link group. The client may request that the server take certain actions but the final responsibility is the server's.

2.2.4. SMC-R link identifiers

This architecture defines multiple identifiers to identify SMC-R links and peers.

- o Link number: This is a one-byte value that identifies an SMC-R link within a link group. Both the server and the client use this number to distinguish an SMC-R link from other links within the same link group. It is only unique within a link group. In order to prevent timing windows that may occur when a server creates a new link while the client is still cleaning up a previously existing link, link numbers cannot be reused until the entire link numbering space has been exhausted.

- o Link User ID: This is an architecturally opaque four byte value that a peer uses to uniquely define an SMC-R link within its own space. This means that a link user ID is unique within one peer only. Each peer defines its own link user ID for a link. The peers exchange this information once during link setup and it is never used architecturally again. The purpose of this identifier is for network management, display, and debugging purposes. For example an operator on a client could provide the operator on the server with the server's link user ID if he requires the server's operator to check on the operation of a link that the client is having trouble with.
- o Peer ID: The SMC-R peer ID uniquely identifies a specific instance of a specific TCP/IP stack. It is required because in clustered and load balancing environments, an IP address does not uniquely identify a TCP/IP stack. An RNIC's MAC/GID also doesn't uniquely or reliably identify a TCP/IP stack because RNICs can go up and down and even be redeployed to other TCP/IP stacks in a multiple partitioned or virtualized environment. The peer ID is not only unique per TCP/IP stack but is also unique per instance of a TCP/IP stack, meaning that if a TCP/IP stack is restarted, its peer ID changes.

2.3. SMC-R resilience and load balancing

The SMC-R multi-link architecture provides resilience for network high availability via failover capability to an alternate RoCE adapter.

The SMC-R multilink architecture does not define primary, secondary or alternate roles to the links. Instead there are multiple active links representing multiple redundant RoCE paths over the same LAN.

Assignment of TCP connections to links is unidirectional and asymmetric. This means that the client and server may each choose a separate link for their RDMA writes associated with a specific TCP connection.

If a hardware failure occurs or a QP failure associated with an individual link, then the TCP connections that were associated with the failing link are dynamically and transparently switched to use another available link. The server or the client can detect a

failure and immediately move their TCP connections and then notify their peer via the DELETE LINK LLC command. While the client can notify the server of an apparent link failure with the DELETE LINK LLC command, the server performs the actual link deletion.

The movement of TCP connections to another link can be accomplished with minimal coordination between the peers. The TCP connection movement is also transparent to and non disruptive to the TCP socket application workloads for most failure scenarios. After a failure, the surviving links and all associated hardware must handle the link group's workload.

As each SMC-R peer begins to move active TCP connections to another link all current RDMA write operations must be allowed to complete. Then the moving peer sends a signal to verify receipt of the last successful write by its peer. If this verification fails, the TCP connection must be reset. Once this verification is complete, all writes that failed may then be retried, in order, over the new link. Any data writes or CDC messages for which the sender did not receive write completion must be replayed before any subsequent data or CDC write operations are sent. LLC messages are not retried over the new link because they are dependent on a known link configuration, which has just changed because of the failure. The initiator of an LLC message exchange that fails will be responsible for retrying once the link group configuration stabilizes.

When a new link becomes available and is re-added to the link group then each peer is free to rebalance its current TCP connections as needed or only assign new TCP connections to the newly added link. Both the server and client are free to manage TCP connections across the link group as needed. TCP connection movement does not have to be stimulated by a link failure.

The SMC-R architecture also defines orderly vs. disorderly failover. The type is communicated in the LLC Delete Link command and is simply a means to indicate that the link has terminated (disorderly) or link termination is imminent (orderly). The orderly link deletion could be initiated via operator command or programmatically to bring down an idle link. For example an operator command could initiate orderly shut down of an adapter for service. Implementation of the two types is based on implementation requirements and is beyond the scope of the SMC-R architecture.

3. SMC-R Rendezvous architecture

Rendezvous is the process that SMC-R capable peers use to dynamically discover each others' capabilities, negotiate SMC-R connections, set up SMC-R links and link groups, and manage those link groups. A key aspect of SMC-R rendezvous is that it occurs dynamically and automatically, without requiring SMC link configuration to be defined by an administrator.

SMC-R Rendezvous starts with the TCP/IP three-way handshake during which connection peers use TCP options to announce their SMC-R capabilities. If both endpoints are SMC-R capable, then Connection Layer Control (CLC) messages are exchanged between the peers' SMC-R layers over the newly established TCP connection to negotiate SMC-R credentials. The CLC message mechanism is analogous to the messages exchanged by SSL for its handshake processing.

If a new SMC-R link is being set up, Link Layer Control (LLC) messages are used to confirm RDMA connectivity. LLC messages are also used by the SMC-R layers at each peer to manage the links and link groups.

Once an SMC-R link is set up or agreed to by the peers, the TCP sockets are passed to the peer applications which use them as normal. The SMC-R layer, which resides under the sockets layer, transmits the socket data between peers over RDMA using the SMC-R protocol, bypassing the TCP/IP stack.

3.1. TCP options

During the TCP/IP three-way handshake, the client and server indicate their support for SMC-R by including experimental TCP option 254 on the three-way handshake flows, in accordance with RFC 6994 "Shared Use of Experimental TCP Options". The ExID value used is the string 'SMCR' in EBCDIC (IBM-1047) encoding (0xE2D4C3D9). This ExID has been registered in the TCP ExIDs registry maintained by IANA.

After completion of the 3-way TCP handshake each peer queries its peer's options. If both peers set the TCP option on the three-way handshake, inline SMC-R negotiation occurs using CLC messages. If neither peer or only one peer set the TCP option, SMC-R cannot be used for the TCP connection, and the TCP connection completes setup using the IP fabric.

3.2. Connection Layer Control (CLC) messages

CLC messages are sent as data payload over the IP network using the TCP connection between SMC-R layers at the peers. They are analogous to the messages used to exchange parameters for SSL.

Use of CLC messages is detailed in the following sections. The following list provides a summary of the defined CLC messages and their purposes:

- o SMC PROPOSAL: Sent from the client to propose that this TCP connection is eligible to be moved to SMC-R. The client identifies itself and its subnet to the server and passes the SMC-R elements for a suggested RoCE path via the MAC and GID.
- o SMC ACCEPT: Sent from the server to accept the client's TCP connection SMC proposal. The server responds to the client's proposal by identifying itself to the client and passing the elements of a RoCE path that the client can use to to perform RDMA writes to the server. This consists of SMC-R link elements such as RoCE MAC, GID, RMB information etc.
- o SMC CONFIRM: Sent from the client to confirm the server's acceptance of SMC connection. The client responds to the server's acceptance by passing the elements of a RoCE path that the server can use to to perform RDMA writes to the client. This consists of SMC-R link elements such as RoCE MAC, GID, RMB information etc.
- o SMC DECLINE: Sent from either the server or the client to reject the SMC connection, indicating the reason the peer must decline the SMC proposal and allowing the TCP connection to revert back to IP connectivity.

3.3. LLC messages

Link Layer Control (LLC) messages are sent between peer SMC-R layers over an SMC-R link to manage the link or the link group. LLC messages are sent using RoCE sendmsg with inline data and are 44 bytes long. The 44 bytes size is based on what can fit into a RoCE Work Queue Element (WQE) without requiring the posting of receive buffers.

LLC messages generally follow a request-reply semantic. Each message has a request flavor and a reply flavor, and each request must be confirmed with a reply, except where otherwise noted. Use of LLC messages is detailed in the following sections. The following list provides a summary of the defined LLC messages and their purposes:

- o ADD LINK: Add a new link to a link group. Sent from the server to the client to initiate addition of a new link to the link group, or from the client to the server to request that the server initiate addition of a new link.
- o ADD LINK CONTINUATION: This is a continuation of ADD link that allows the ADD link to span multiple commands, because all the link information cannot be contained in a single ADD LINK message
- o CONFIRM LINK: Used to confirm that RoCE connectivity over a newly created SMC-R link is working correctly. Initiated by the server, and both this message and its reply must flow over the SMC-R link being confirmed.
- o DELETE LINK: When initiated by the server, deletes a specific link from the link group or deletes the entire link group. When initiated by the client, requests that the server delete a specific link or the entire link group.
- o CONFIRM RKEY: Informs the peer on the SMC-R link of the addition of an RMB to the link group.
- o CONFIRM RKEY CONTINUATION: This is a continuation of CONFIRM RKEY that allows the ADD link to span multiple commands, in the event that all of the information cannot be contained in a single CONFIRM RKEY message.
- o DELETE RKEY: Informs the peer on the SMC-R link of the deletion of one or more RMBs from the link group
- o TEST LINK: Verifies that an already-active SMC-R link is active and healthy
- o Optional LLC message: Any LLC message in which the two high order bits of the opcode are b'10' is an optional message and must be silently discarded by a receiving peer that does not support the opcode. No such messages are defined in this version of the architecture, however the concept is defined to allow for toleration of possible advanced, optional functions.

CONFIRM LINK and TEST LINK are sensitive to which link they flow on and must flow on the link being confirmed or tested. The other flows may flow over any active link in the link group. When there are multiple links in a link group, a response to an LLC message must flow over the same link that the original message flowed over, with the following exceptions:

- o ADD LINK request from a server in response to an ADD LINK from a client
- o DELETE LINK request from a server in response to a DELETE LINK from a client

3.4. CDC Messages

Connection Data Control (CDC) messages are sent over the RoCE fabric between peers using RoCE sendmsg with inline data, and are 44 bytes long which is based on the size that can fit into a RoCE Work Queue Element (WQE) without requiring the posting of receive buffers. CDC messages are used to describe the socket application data passed via RDMA write operations, and TCP connection state information including producer and consumer cursors, RMBE state information, and failover data validation.

3.5. Rendezvous flows

Rendezvous information for SMC-R is exchanged as TCP options on the TCP 3-way handshake flows to indicate capability, followed by inline TCP negotiation messages to actually do the SMC-R setup. Formats of all rendezvous options and messages discussed in this section are detailed in Appendix A.

3.5.1. First contact

First contact between RoCE peers occurs when a new SMC-R link group is being set up. This could be because no SMC-R links already exist between the peers, or the server decides to create a new SMC-R link group in parallel with an existing one.

3.5.1.1. TCP Options pre-negotiation

The client and server indicate their SMC-R capability to each other using TCP option 254 on the TCP 3-way handshake flows.

A client who wishes to do SMC-R will include TCP option 254 using an ExID equal to the EBCDIC (codepage IBM-1047) encoding of "SMCR" on its SYN flow.

A server that supports SMC-R will include TCP option 254 with the ExID value of EBCDIC "SMCR" on its SYN-ACK flow. Because the server is listening for connections and does not know where client connections will come from, the server implementation may choose to

unconditionally include this TCP option if it supports SMC-R. This may be required for server implementations where extensions to the TCP stack are not practical. For server implementations which can add code to examine and react to packets during the three-way handshake, the server should only include the SMC-R TCP option on SYN-ACK if the client included it on its SYN packet.

A client who supports SMC-R and meets the three conditions outlined above may optionally include the TCP option for SMC-R on its ACK flow, regardless of whether or not the server included it on its SYN-ACK flow. Some TCP/IP stacks may have to include it if the SMC-R layer cannot modify the options on the socket until the 3-way handshake completes. Proprietary servers should not include this option on the ACK flow, since including it on the SYN flow was sufficient to indicate the client's capabilities.

Once the initial three-way TCP handshake is completed, each peer examines the socket options. SMC-R implementations may do this by examining what was actually provided on the SYN and SYN-ACK packets or by performing a `getsockopt()` operation to determine the options set by the peer. If neither peer, or only one peer, specified the TCP option for SMC-R, then SMC-R cannot be used on this connection and it proceeds using normal IP flows and processing.

If both peers specified the TCP option for SMC-R, then the TCP connection is not started yet and the peers proceed to SMC-R negotiation using inline data flows. The socket is not yet turned over to the applications; instead the respective SMC layers exchange CLC messages over the newly formed TCP connection.

3.5.1.2. Client Proposal

If SMC-R is supported by both peers, the client sends an SMC Proposal CLC message to the server. On this flow from client to server it is not immediately apparent if this is a new or existing SMC-R link because in clustered environments a single IP address may represent multiple hosts. This type of cluster virtual IP address can be owned by a network based or host based layer 4 load balancer that distributes incoming TCP connections across a cluster of servers/hosts. Other clustered environments may also support the movement of a virtual IP address dynamically from one host in the cluster to another for high availability purposes. In summary, the client can not pre-determine that a connection is targeting the same host simply by matching the destination IP address for outgoing TCP connections. Therefore it cannot pre-determine the SMC-R link that will be used for a new TCP connection. This information will be

dynamically learned and the appropriate actions will be taken as the SMC-R negotiation handshake unfolds.

On the SMC-R proposal message, the initiator (client) proposes use of SMC-R by including its peer ID and GID and MAC addresses, as well as the IP subnet number of the outgoing interface (if IPv4) or the IP prefix list for the network that the proposal is sent over (if IPv6). At this point in the flow, the client makes no local commitments of resources for SMC-R.

When the server receives the SMC Proposal CLC message, it uses the peer ID provided by the client plus subnet or prefix information provided by the client, to determine if it already has a usable SMC-R link with this SMC-R peer. If there is one or more existing SMC-R links with this SMC-R peer, the server then decides which SMC link it will use for this TCP connection. See subsequent sections for the cases of reusing an existing SMC-R link or creating a parallel SMC link group between SMC-R peers.

If this is a first contact between SMC-R peers the server must validate that it is on the same LAN as the client before continuing. For IPv4, the server does this by verifying that it has an interface with an IP subnet number that matches the subnet number set by the client on the SMC Proposal. For IPv6 it does this by verifying that it is directly attached to at least one IP prefix that was listed by the client in its SMC Proposal message.

If server agrees to use SMC-R, the server begins setup of a new SMC-R link by allocating local QP and RMB resources (setting its QP state to INIT) and providing its full SMC-R information in an SMC Accept CLC message to the client over the TCP connection, along with a flag set indicating that this is a first contact flow. While the SMC Accept message could flow over any route back to the client depending upon IP routing, the SMC-R credentials provided must be for the common subnet or prefix between the server and client, as determined above. If the server cannot or does not want to do SMC-R with the client it sends an SMC Decline CLC message to the client and the connection data may begin flowing using normal TCP/IP flows.

3.5.1.3. Server acceptance

When the client receives the SMC Accept from the server, it uses the combination of the first contact flag, its GID/MAC and the GID/MAC returned by the server plus the LAN that the connection is setting up over and the QP number provided by the server to determine if this is a new or existing SMC-R link.

If it is an existing SMC-R link, and the client agrees to use that link for the TCP connection, see 3.5.2. Subsequent contact below. If it is a new SMC-R link between peers that already have an SMC link, then the server is starting a new SMC link group.

Assuming this is either a first contact between peers or the server is starting a new SMC link group, the client now allocates local QP and RMB resources for the SMC-R link (setting the QP state to RTR or "ready to receive"), associates them with the server QP as learned on the SMC Accept CLC message, and sends an SMC Confirm CLC message to the server over the TCP connection with its SMC-R link information included. The client also starts a timer to wait for the server to confirm the reliable connected QP as described below.

3.5.1.4. Client confirmation

Upon receipt of the client's SMC Confirm CLC message, the server associates its QP for this SMC-R link with the client's QP as learned on the SMC Confirm CLC message and sets its QP state to RTS (ready to send). Now the client and the server have reliable connected QPs.

3.5.1.5. Link (QP) confirmation

Since setting up the SMC-R link and its QPs did not require any network flows on the RoCE fabric, the client and server must now confirm connectivity over the RoCE fabric. To accomplish this, the server will send a "Confirm Link" Link Layer Control (LLC) message to the client over the RoCE fabric. The "Confirm Link" LLC message will provide the server's MAC, GID, and QP information for the connection, allow each partner to communicate the maximum number of links it can tolerate in this link group (the "link limit"), and will additionally provide two link IDs:

- o a one-byte server-assigned Link number that is used by both peers to identify the link within the link group and is only unique within a link group.
- o a four byte link user id. This opaque value is assigned by the server for the server's local use and is provided to the client for management purposes, for example to use in network management displays and products.

When the server sends this message, it will set a timer for receiving confirmation from the client.

When the client receives the server's confirmation "Confirm Link" LLC message it will cancel the confirmation timer it set when it sent the

SMC Confirm message. It will also advance its QP state to RTS and respond over the RoCE fabric with a "Confirm Link" response LLC message, providing its MAC, GID, QP number, link limit, confirming the one byte link number sent by the server, and providing its own four byte link user id to the server.

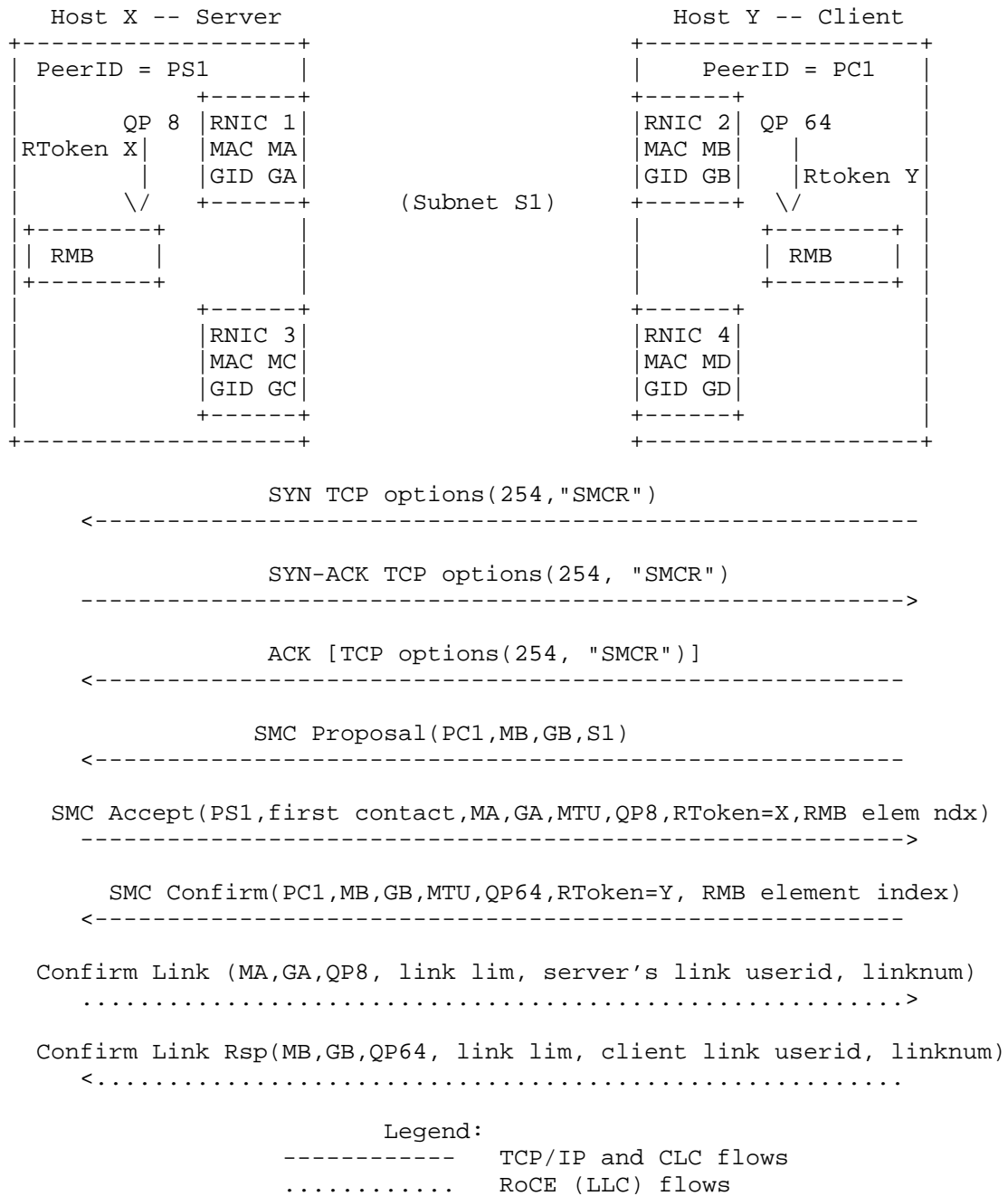


Figure 8 First contact rendezvous flows

Technically, the data for the TCP connection could now flow over the RoCE path. However if this is first contact, there is no alternate for this recently established RoCE path. Since in the current architecture there is no failover from RoCE to IP once connection data starts flowing, this means that a failure of this path would disrupt the TCP connection, meaning that the level of redundancy and failover is less than that provided by IP. If the network has alternate RoCE paths available, they would not be usable at this point, which is an unacceptable condition

3.5.1.6. Second SMC-R link setup

Because of the unacceptable situation described above, TCP data will not be allowed to flow on the newly established SMC-R link until a second path has been set up, or at least attempted.

If the server has a second RNIC available on the same LAN, it attempts to set up the second SMC-R link over that second RNIC. If it only has one RNIC available on the LAN, it will attempt to set up the second SMC-R link over that one RNIC. In the latter case, the server is attempting to set up an asymmetric link, in case the client does have a second RNIC on the LAN.

In either case the server allocates a new QP over the RNIC it is attempting to use for the second link, assigns a link number to the new link and also creates an RToken for the RMB over this second QP (note that this means that the first and second QP each has its own RToken to represent the same RMB). The server provides this information, as well as the MAC and GID of the RNIC it is attempting set up the second link over in an "Add Link" LLC message which it sends to the client over the SMC-R link that is already set up.

3.5.1.6.1. Client processing of "Add Link" LLC message from server

When the client receives the server's "Add Link" LLC message, it examines the GID and MAC provided by the server to determine if the server is attempting to use the same server-side RNIC as the existing SMC-R link, or a different one.

If the server is attempting to use the same server-side RNIC as the existing SMC-R link, then the client verifies that it has a second RNIC on the same LAN. If it does not, the client rejects the "Add Link" request from the server, because the resulting link would be a parallel link which is not supported within a link group. If the client does have a second RNIC on the same LAN, it accepts the request and an asymmetric link will be set up.

If the server is using a different server-side RNIC from the existing SMC-R link then the client will accept the request and a second SMC-R link will set up in this SMC-R link group. If the client has a second RNIC on the same LAN, that second RNIC will be used for the second SMC-R link, creating symmetric links. If the client does not have a second RNIC on the same LAN, it will use the same RNIC as was used for the initial SMC-R link, resulting in the setup of an asymmetric link in the SMC-R link group.

In either case, when the client accepts the server's "Add Link" request, it allocates a new QP on the chosen RNIC and creates an Rkey over that new QP for the client-side RMB for the SMC link group, then sends an "Add Link" reply LLC message to the server providing that information as well as echoing the Link number that was set by the server.

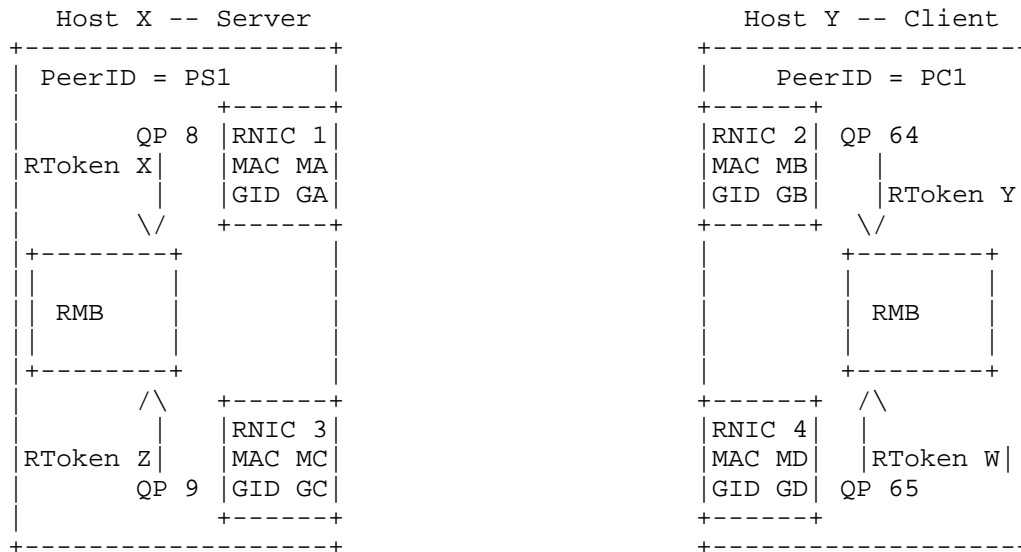
If the client rejects the server's "Add Link" request, it sends an "Add Link" reply LLC message to the server with the reason code for the rejection.

3.5.1.6.2. Server processing of "Add Link" reply LLC message from the client

If the client sends a negative response to the server or no reply is received, the server frees the RoCE resources it had allocated for the new link. Having a single link in an SMC-R link group is undesirable and the server's recovery is detailed in C.8. Failure to add second SMC-R link to a link group.

If the client sends a positive reply to the server with MAC/GID/QP/Rkey information, the server associates its QP for the new SMC-R link to the QP that the client provided. Now the new SMC-R link is in the same situation that the first was in after the client sent its ACK packet - there is a reliable connected QP over the new RoCE path, but there have been no RoCE flows to confirm that it's actually usable. So at this point the client and server will exchange "Confirm Link" LLC messages just like they did on the first SMC-R link.

If either peer receives a failure during this second "Confirm Link" LLC exchange (either an immediate failure which implies that the message did not reach the partner, or a timeout), it sends a "Delete Link" LLC message to the partner over the first (and now only) link in the link group which must be acknowledged before data can flow on the single link in the link group.



```
First SMC-R link setup as shown in Figure 8
<-.--.--.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.->

ADD link request (QP9,MC,GC, link number=2)
.....>

ADD link response (QP65,MD,GD, link number=2)
<.....

ADD link continuation request (RToken=Z)
.....>

ADD link continuation response(RToken=W)
<.....

Confirm Link(MC,GC,QP9,link number=2, link userid)
.....>

Confirm Link response(MD,GD,QP65,link number=2, link userid)
<.....
```

Legend:

-----	TCP/IP and CLC flows
.....	RoCE (LLC) flows

Figure 9 First contact, second link setup

3.5.1.6.3. Exchange of Rkeys on second SMC-R link

Note that in the scenario described here, first contact, there is only one RMB Rkey to exchange on the second SMC-R link and it is exchanged in the Add Link Continuation request and reply. In scenarios other than first contact, for example, adding a new SMC-R link to a longstanding link group with multiple RMBs, additional flows will be required to exchange additional RMB Rkeys. See 3.5.5.2.3. Adding a new SMC-R link to a link group with multiple RMBs for more details on these flows

3.5.1.6.4. Aborting SMC-R and falling back to IP

If both partners don't provide the SMC-R TCP option during the 3 way TCP handshake, the connection falls back to normal TCP/IP. During the SMC-R negotiation that occurs after the 3 way TCP handshake, either partner may break off SMC-R by sending an SMC Decline CLC message. The SMC Decline CLC message may be sent in place of any expected message, and may also be sent during the Confirm Link LLC exchange if there is a failure before any application data has flowed over the RoCE fabric. For more detail on exactly when an SMC Decline can flow during link group setup, see C.1. SMC Decline during CLC negotiation and C.2. SMC Decline during LLC negotiation

If this fallback to IP happens while setting up a new SMC-R link group, the RoCE resources allocated for this SMC-R link group relationship are torn down and it will be retried as a new SMC-R link group next time a connection starts between these peers with SMC-R proposed. Note that if this happens because one side doesn't support SMC-R, there will be very little to tear down as the TCP option will have failed to flow either on the initial SYN or the SYN-ACK, before either side had reserved any local RoCE resources.

3.5.2. Subsequent contact

"Subsequent contact" means setting up a new TCP connection between two peers that already have an SMC-R link group between them, and reusing the existing SMC-R link group. In this case it is not necessary to allocate new QPs. However it is possible that a new RMB has been allocated for this TCP connection, if the previous TCP connection used the last element available in the previously used RMB, or for any other implementation-dependent reason. For this reason, and for convenience and error checking, the same TCP option 254 followed by inline negotiation method described for initial contact will be used for subsequent contact, but the processing differs in some ways. That processing is described below.

3.5.2.1. SMC-R proposal

When the client begins the inline negotiation with the server, it does not know if this is a first contact or a subsequent contact. The client cannot know this information until it sees the server's peer ID to determine whether or not it already has an SMC-R link with this peer that it can use. There are several reasons why it is not sufficient to use the partner IP address, subnet, VLAN or other IP information to make this determination. The most obvious reason is distributed systems: if the server IP address is actually a virtual IP address representing a distributed cluster, the actual host serving this TCP connection may not be the same as the host that served the last TCP connection to this same IP address.

After the TCP three way handshake, assuming both partners indicate SMC-R capability, the client builds and sends the SMC Proposal CLC message to the server in exactly the same manner as it does in the first contact case, and in fact at this point doesn't know if it's first contact or subsequent contact. As in the first contact case, the client sends its Peer ID value, suggested RNIC GID/MAC, and IP subnet or prefix information.

Upon receiving the client's proposal, the server looks up the peer ID provided to determine if it already has a usable SMC-R link group with this peer. If it does already have a usable SMC-R link group, the server then needs to decide if it will use the existing SMC-R link group, or create a new link group. For the new link group case, see 3.5.3. First contact variation: creating a parallel link group, below.

For this discussion assume the server decides to use the existing SMC-R link group for the TCP connection, which is expected to be the most common case. The server is responsible for making this decision. Then the server needs to communicate that information to the client, but it is not necessary to allocate, associate, and confirm QPs for the chosen SMC-R link. All that remains to be done is to set up RMB space for this TCP connection.

If one of the RMBs already in use for this SMC-R link group has an available element that uses the appropriate buffer size, the server merely chooses one for this TCP connection and then sends an SMC Accept CLC message, providing the full RoCE information for the chosen SMC-R link to the client, using the same format as the SMC Accept CLC message described in the initial contact section above.

The server may choose to use the SMC-R link that matches the suggested MAC/GID provided by the client on the SMC Proposal for its

RDMA writes but is not obligated to. The final decision on which specific SMC-R link to assign a TCP connection to is an independent server and client decision.

It may be necessary for the server to allocate a new RMB for this connection. The reasons for this are implementation dependent and could include: no available space in existing RMB or RMBs, or desire to allocate a new RMB that uses a different buffer size from the ones already created, or any other implementation dependent reason. In this case the server will allocate the new RMB and then perform the flows described in 3.5.5.2.1. Adding a new RMB to an SMC-R link group. Once that processing is complete, the server then provides the full RoCE information, including the new Rkey, for this connection on an SMC Confirm CLC message to the client.

3.5.2.2. SMC-R acceptance

Upon receiving the SMC Accept CLC message from the server, the client examines the RoCE information provided by the server to determine if this is a first contact for a new SMC link group, or subsequent contact for an existing SMC-R link group. It is subsequent contact if the server side peer ID, GID, MAC and QP number provided on the packet match a known SMC-R link, and the "first contact" flag is not set. If this is not the case, for example the GID and MAC match but the QP is new, then the server is creating a new, parallel SMC-R link group and this is treated as a first contact.

A different RMB RToken does not indicate a first contact as the server may have allocated a new RMB, or be using several RMBs for this SMC-R link. The client needs the server's RMB information only for its RDMA writes to the server, and since there is no requirement for symmetric RMBs, this information is simply control information for the RDMA writes on this SMC-R link.

The client must validate that the RMB element being provided by the server is not in use by another TCP connection on this SMC-R link group. This validation must validate the new <rtoken, index> across all known <rtoken, index> on this link group. See 4.4.2. RMB element reuse and conflict resolution for the case in which the server tries to use an RMB element that is already in use on this link group.

Once the client has determined that this TCP connection is a subsequent contact over an existing SMC link, it performs a similar RMB allocation process as the server did: it either allocates an element from an RMB already associated with this SMC-R link, or it allocates a new RMB and associates it with this SMC-R link and then chooses an element out of it.

If the client allocates a new RMB for this TCP connection, it performs the processing described in 3.5.5.2.1. Adding a new RMB to an SMC-R link group. Once that processing is complete, the client provides its full RoCE information for this TCP connection on an SMC Confirm CLC message.

Because an SMC-R link with a verified connected QP already exists and is being reused, there is no need for verification or alternate QP selection flows or timers.

3.5.2.3. SMC-R confirmation

When the server receives the client's SMC Confirm CLC message on a subsequent contact, it verifies the following:

- o the RMB element provided by the client is not already in use by another TCP connection on this SMC-R link group (see section 4.4.2. RMB element reuse and conflict resolution for the case in which it is).
- o The MAC/GID/QP info provided by the client matches an active link within the link group. The client is free to select any valid / active link. The client is not required to select the same link as the server.

If this validation passes, the server stores the client's RMB information for this connection and the RoCE setup of the TCP connection is complete.

3.5.2.4. TCP data flow race with SMC Confirm CLC message

On a subsequent contact TCP/IP connection, a peer may send data as soon as it has received the peer RMB information for the connection. There are no additional RoCE confirmation flows, since the QPs on the SMC link are already reliably connected and verified.

In the majority of cases the first data will flow from the client to the server. The client must send the SMC Confirm CLC message before sending any connection data over the chosen SMC-R link, however the client need not wait for confirmation of this message, and in fact there will be no such confirmation. Since the server is required to have the RMB fully set up and ready to receive data from the client before sending SMC Accept CLC message, the client can begin sending data over the SMC-R link immediately upon completing the send of the SMC Confirm CLC message.

It is possible that data from the client will arrive into the server side RMB before the SMC Confirm CLC message from the client has been processed. In this case the server must handle this race condition, and not provide the arrived TCP data to the socket application until the SMC Confirm CLC message has been received and fully processed, opening the socket.

If the server has initial data to send to the client which is not a response to the client (this case should be rare), it can send the data immediately upon receiving and processing the SMC Confirm CLC message from the client. The client must have opened the TCP socket to the client application upon sending of SMC Confirm CLC message so the client will be ready to process data from the server.

3.5.3. First contact variation: creating a parallel link group

Recall that parallel SMC-R links within an SMC-R link group are not supported. These are multiple SMC-R links within a link group that use the same network path. However, multiple SMC-R link groups between the same peers are supported. This means that if multiple SMC-R links over the same RoCE path are desired, it is necessary to use multiple SMC-R link groups. While not a recommended practice, this could be done for platform specific reasons, like QP separation of different workloads. Only the server can drive the creation of multiple SMC-R link groups between peers.

At a high level, when the server decides to create an additional SMC-R link group with a client it already has an SMC-R link group with, the flows are basically the same as the normal "first contact" case described above. The following provides more detail and clarification of processing in this case.

When the server receives the SMC Proposal CLC message from the client and using the GID/MAC info determines that it already has an SMC-R link group with this client, the server can either reuse the existing SMC-R link group (detailed in 3.5.2. Subsequent contact above) or it can create a new SMC-R link group in addition to the existing one.

If the server decides to create a new SMC-R link group, it does the same processing it would have done for first contact: allocate QP and RMB resources as well as alternate QP resources, and communicate the QP and RMB information to the client on the SMC Accept CLC message with the "first contact" flag set.

When the client receives the server's SMC Accept CLC message with the new QP information and the "first contact" flag, it knows the server is creating a new SMC-R link group even though it already has an SMC-

R link group with the server. In this case the client will also allocate a new QP for this new SMC link and allocate an RMB for this link and generate an Rkey for it.

Note that multiple SMC-R link groups between the same peers must access different RMB resources, so new RMBs will be required. Using the same RMBs that are in use in another SMC-R link group is not permitted.

The client then associates its new QP with the server's new QP and sends its SMC Confirm CLC message back to the server providing the new QP/RMB information and sets its confirmation timer for the new SMC-R link.

When the server receives the client's SMC Confirm CLC message it associates its QP with the client's QP as learned on the SMC Confirm CLC message and sends a confirmation LLC message. The rest of the flow, with the confirmation QP and setup of additional SMC-R links, unfolds just like the first contact case.

3.5.4. Normal SMC-R link termination

The normal sockets API trigger points are used by the SMC-R layer to initiate SMC-R connection termination flows. The main design point for SMC-R normal connection flows is to use the SMC-R protocol to first shutdown the SMC-R connection and free up any SMC-R RDMA resources and then allow the normal TCP connection termination protocol (i.e. FIN processing) to drive cleanup of the TCP connection that exists on the IP fabric. This design point is very important in ensuring that RDMA resources such as the RMBEs are only freed and reused when both SMC-R end points are completely done with their RDMA Write operations to the partner's RMBE.

When the last TCP connection over an SMC-R link group terminates, the link group can be terminated. Similar to creation of SMC-R links and link groups, the primary responsibility for determining that normal termination is needed and initiating it lies with the server. Implementations may opt to set timers to keep SMC-R link groups up for a specified time after the last TCP connection ends, to avoid churn in cases when TCP connections come and go regularly.

The link or link group may also be terminated as a result of an operator initiated command. This command can be entered at either the client or the server. If entered at the client, the client requests that the server perform link or link group termination, and the responsibility for doing so ultimately lies with the server.

When the server determines that the SMC-R link group is to be terminated, it sends a DELETE LINK LLC message to the client, with a flag set indicating that all links in the link group are to be terminated. After receiving confirmation from the adapter that the DELETE LINK LLC message has been sent, the server can clean up its end of the link group (QPs, RMBs, etc). Upon receipt of the DELETE LINK message from the server, the client must immediately comply and clean up its end of the link group. Any TCP connections that the client believes to be active on the link group must be immediately terminated.

The client can request that the server delete the link group as well. The client does this by sending a DELETE LINK message to the server indicating that cleanup of all links is requested. The server must comply by sending a DELETE LINK to the client and processing as described above. If there are TCP connections active on the link group when the server receives this request, they are immediately terminated by sending a RST flow over the IP fabric.

3.5.5. Link group management flows

3.5.5.1. Adding and deleting links in an SMC-R link group

The server has the lead role in managing the composition of the link group. Links are added to link group by the server. The client may notify the server of new conditions that may result in the server adding a new link, but the server is ultimately responsible. In general links are deleted from the link group by the server, however in certain error cases the client may inform the server that a link must be deleted and treat it as deleted without waiting for action from the server. These flows are detailed in the following sections

3.5.5.1.1. Server initiated Add Link processing

As described in previous sections, the server initiates an Add Link exchange to create redundancy in a newly created link group. Once a link group is established the server may also initiate Add Link for other reasons, including:

- o Availability of additional resources on the server host to support an additional SMC-R link. This may include the provisioning of an additional RNIC, more storage becoming available to support additional QP resources, operator command, or any other implementation dependent reason. Note that, to be available for an existing link group, a new RNIC must be attached to the same RoCE LAN that the link group is using.

- o Receipt of notification from the client that additional resources on the client are available to support an additional SMC-R link. See 3.5.5.1.2. Client initiated Add Link processing.

Server initiated Add Link processing in an established SMC-R link group is the same as the Add Link processing described in 3.5.1.6. Second SMC-R link setup with the following changes:

- o If an asymmetric SMC-R link already exists in the link group a second asymmetric link will not be created. Only one asymmetric link is permitted in a link group.
- o TCP data flow on already existing link(s) in the link group is not halted or otherwise affected during the process of setting up the additional link.

In no case will the server initiate Add Link processing if the link group already has the maximum number of links negotiated by the partners.

3.5.5.1.2. Client initiated Add Link processing

If an additional RNIC becomes available for an existing SMC-R link group on the client's side, the client notifies the server by sending an Add Link request LLC message to the server. Unlike an Add Link request sent by the server to the client, this Add Link request merely informs the server that the client has a new RNIC. If the link group lacks redundancy, or has redundancy only on an asymmetric link with a single RNIC on the client side, the server must initiate an Add Link exchange in response to this message, to create or improve the link group's redundancy.

If the link group already has symmetric link redundancy but has fewer than the negotiated maximum number of links, the server may respond by initiating an Add Link exchange to create a new link using the client's new resource but is not required to.

If the link group already has the negotiated maximum number of links, the server must ignore the client's Add Link request LLC message.

Because the server is not required to respond to the client's Add Link LLC message in all cases, the client must not wait for a response or throw an error if one does not come.

3.5.5.1.3. Server initiated Delete Link Processing

Reasons that a server may delete a link include:

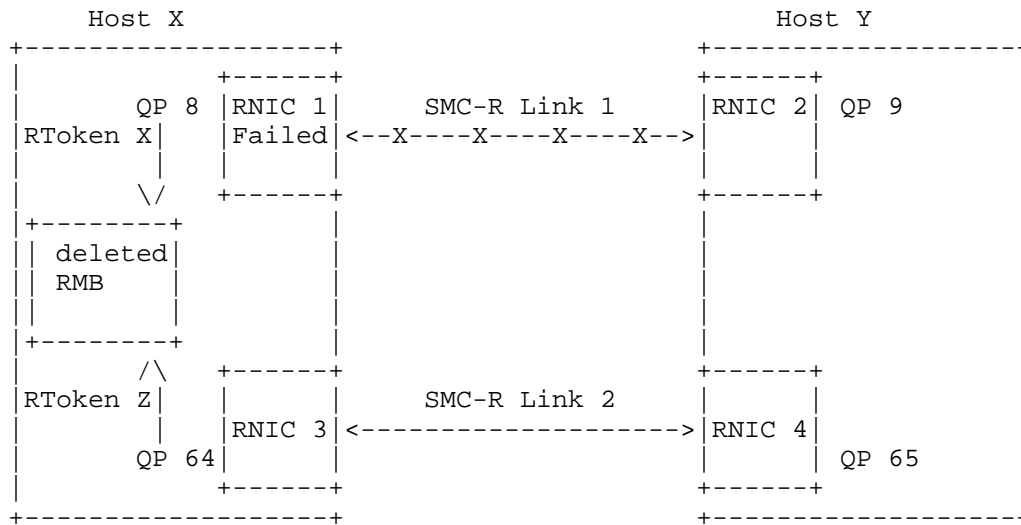
- o The link has not been used for TCP connections for an implementation defined time interval, and deleting the link will not cause the link group to lack redundancy
- o An error in resources supporting the link. These may include but are not limited to: RNIC errors, QP errors, software errors
- o The RNIC supporting this SMC-R link is being taken down, either because of an error case or because of an operator or software command.

If a link being deleted is supporting TCP connections, and there are one or more surviving links in the link group, the TCP connections are moved to the surviving links. For more information on this processing see 2.3. SMC-R resilience and load balancing.

The server deletes a link from the link group by sending a Delete Link request LLC message to the client over any of the usable links in the link group. Because the Delete Link LLC message specifies which link is to be deleted, it may flow over any link in the link group. The server must not clean up its RoCE resources for the link until the client responds.

The client responds to the server's Delete Link request LLC message by sending the server a Delete Link response LLC message. The client must respond positively; it cannot decline to delete the link. Once the server has received the client's Delete Link response, both sides may clean up their resources for the link.

Positive write completion or other indication from the RNIC on the client's side is sufficient to indicate to the client that the server has received the Delete Link response.



```
DELETE LINK(Request, link number = 1,
             .....>
             reason code = RNIC failure)
```

```
DELETE LINK(Response, link number = 1)
<.....
```

(note, architecturally this exchange can flow over either SMC-R link but most likely flows over link 2 since the RNIC for link 1 has failed)

Figure 10 Server initiated Delete Link flow

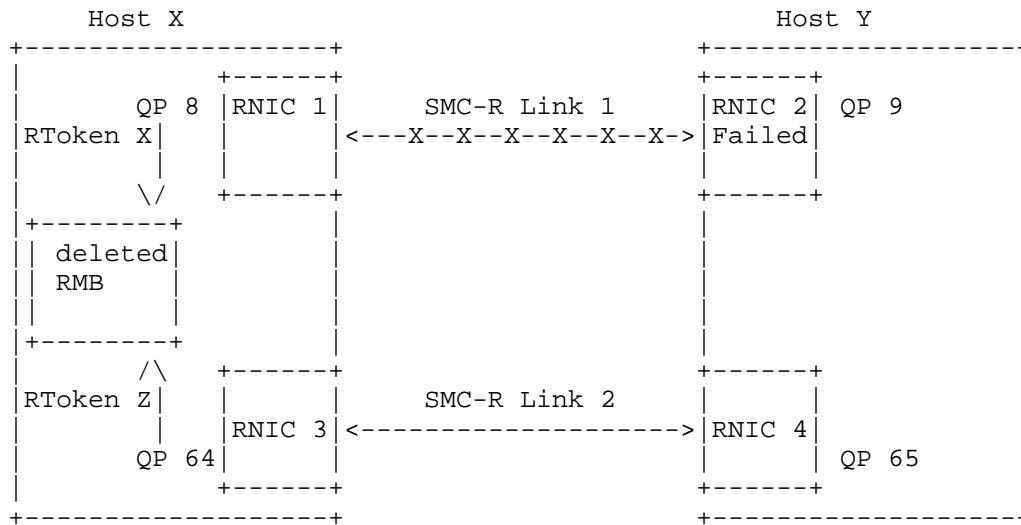
3.5.5.1.4. Client initiated Delete Link request

The client may request that the server delete a link for the same reasons that the server may delete a link, except for inactivity timeout.

Because the client depends on the server to delete links, there are two types of delete requests from client to server:

- o Orderly: the client is requesting that the server delete the link when able. This would result from an operator command to bring down the RNIC or some other nonfatal reason. In this case the server is required to delete the link, but may not do it right away.
- o Disorderly: the server must delete the link right away, because the client has experienced a fatal error with the link.

In either case the server responds by initiating a Delete Link exchange with the client as described in the previous section. The difference between the two is whether the server must do so immediately or can delay for an opportunity to gracefully delete the link.



```
DELETE LINK(Request, link number = 1, disorderly,
<.....
    reason code = RNIC failure)

DELETE LINK(Request, link number = 1,
.....>
    reason code = RNIC failure)

DELETE LINK(Response, link number = 1)
<.....

(note, architecturally this exchange can flow over either
SMC-R link but most likely flows over link 2 since
the RNIC for link 1 has failed)
```

Figure 11 Client-initiated Delete Link

3.5.5.2. Managing multiple Rkeys over multiple SMC-R links in a link group

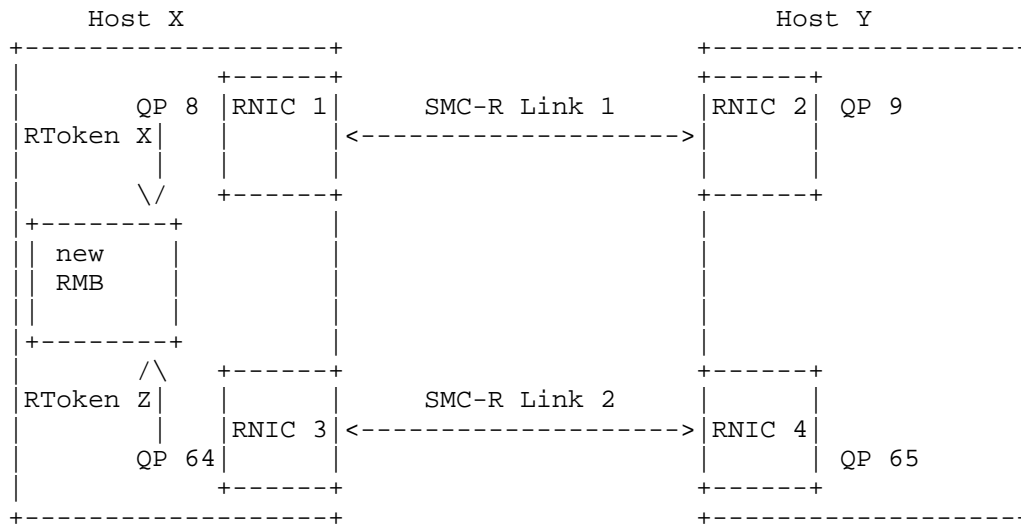
After the initial contact sequence completes and the number of TCP connections increases it is possible that the SMC peers could add additional RMBs to the Link Group. Recall that each peer independently manages its RMBs. Also recall that an RMB's RToken is specific to a QP, which means that when there are multiple SMC-R links in a link group, each RMB accessed with the link group requires a separate RToken for each SMC-R link in the group.

Each RMB that is added to a link must be added to all links within the Link Group. The set of RMBs created for the Link is called the "RToken Set". The RTokens must be exchanged with the peer. As RMBs are added and deleted, the RToken Set must remain in sync.

3.5.5.2.1. Adding a new RMB to an SMC-R link group

A new RMB can be added to an SMC-R link group on either the client or the server side. When an additional RMB is added to an existing SMC-R link group, that RMB must be associated with the QPs for each link in the link group. Therefore when an RMB is added to an SMC-R link group, its RMB RToken for each SMC-R link's QP must be communicated to the peer.

The tokens for a new RMB added to an existing SMC-R link group are communicated using "Confirm Rkey" LLC messages, as shown in Figure 12. The RToken set is specified as pairs: an SMC link number, paired with the new RMB's RToken over that SMC Link. To preserve failover capability, any TCP connection that uses a newly added RMB cannot go active until all RTokens for the RMB have been communicated for all the links in the link group.



```

CONFIRM RKEY(Request, Add,
    .....>
    RToken set((Link 1,RToken X),(Link2,RToken Z)))

CONFIRM RKEY(Response, Add,
    <.....
    RToken set((Link 1,RToken X),(Link2,RToken Z)))

(note, this exchange can flow over either SMC-R link)

```

Figure 12 Add RMB to existing link group

Implementations may choose to proactively add RMBs to link groups in anticipation of need. For example, an implementation may add a new RMB when all of its existing RMBs are over a certain threshold percentage used.

A new RMB may also be added to an existing link group on an as needed basis. For example, when a new TCP connection is added to the link group but there are no available RMB elements. In this case the CLC exchange is paused while the peer that requires the new RMB adds it. An example of this is illustrated in figure 13.

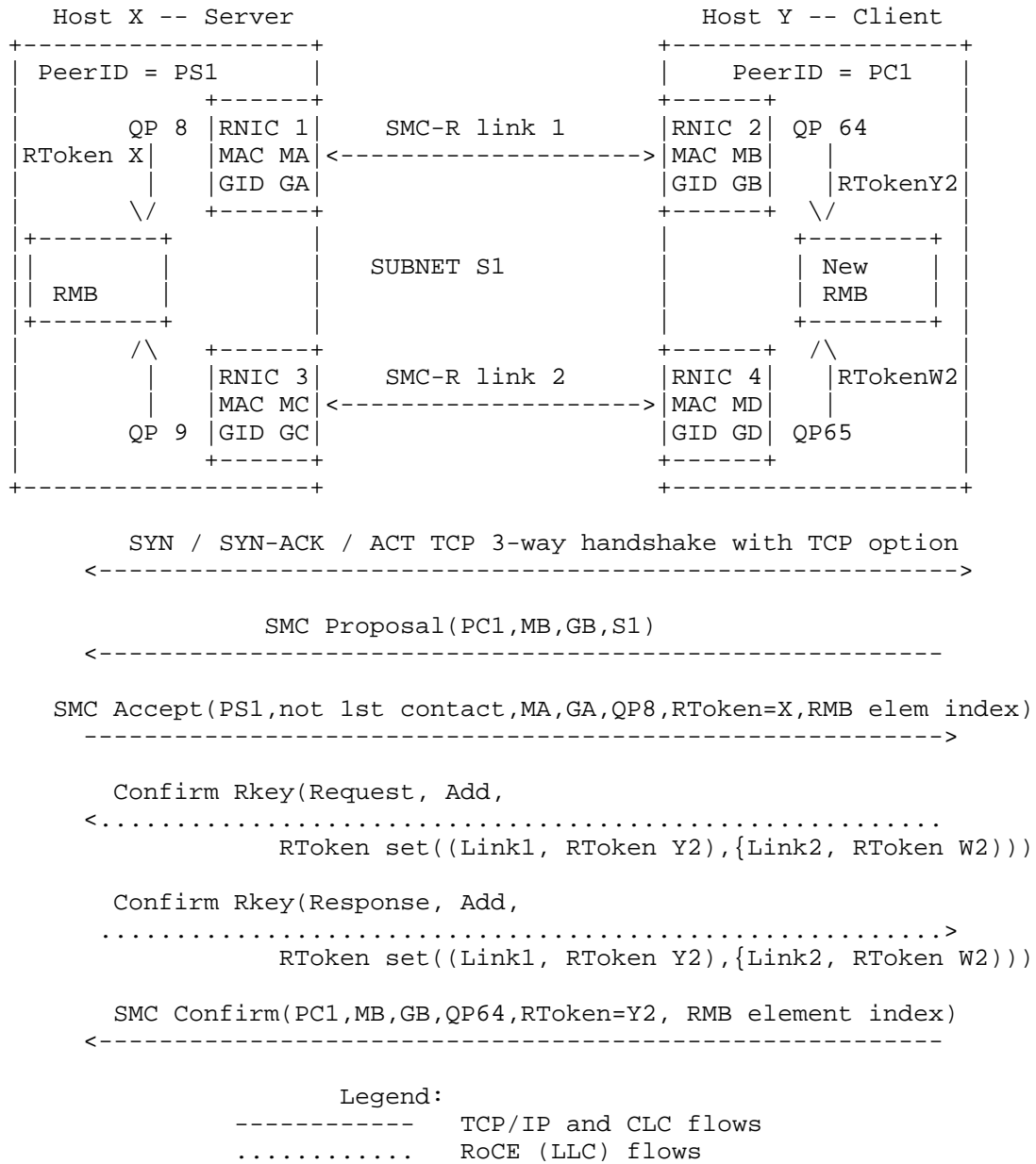
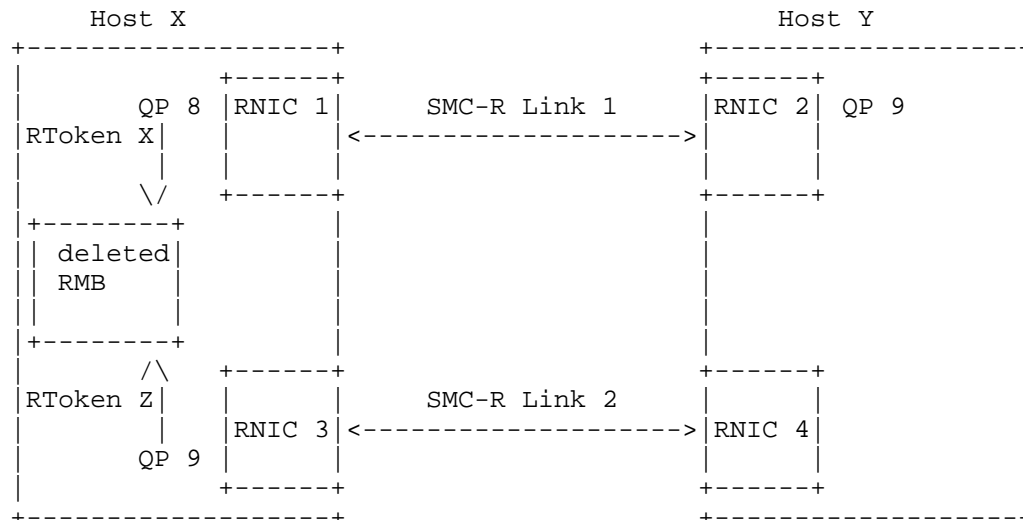


Figure 13 Client adds RMB during TCP connection setup

3.5.5.2.2. Deleting an RMB from an SMC-R link group

Either peer can delete one or more of its RMBs as long as it is not being used for any TCP connections. Ideally an SMC-R peer would use a timer to avoid freeing an RMB immediately after the last TCP connection stops using it, to keep the RMB available for later TCP connections and avoid thrashing with addition and deletion of RMBs. Once an SMC-R peer decides to delete an RMB, it sends a DELETE RKEY LLC message to its peer. It can then free the RMB once it receives a response from the peer. Multiple RMBs can be deleted in a DELETE RKEY exchange.

Note that in a DELETE RKEY message, it is not necessary to specify the full RToken for a deleted RMB. The RMB's Rkey over one link in the link group is sufficient to specify which RMB is being deleted.



```
DELETE RKEY(Request, Rkey list(Rkey X))
.....>

DELETE RKEY(Response, Rkey list(Rkey X))
<.....

(note, this exchange can flow over either SMC-R link)
```

Figure 14 Delete RMB from SMC-R link group

3.5.5.2.3. Adding a new SMC-R link to a link group with multiple RMBs

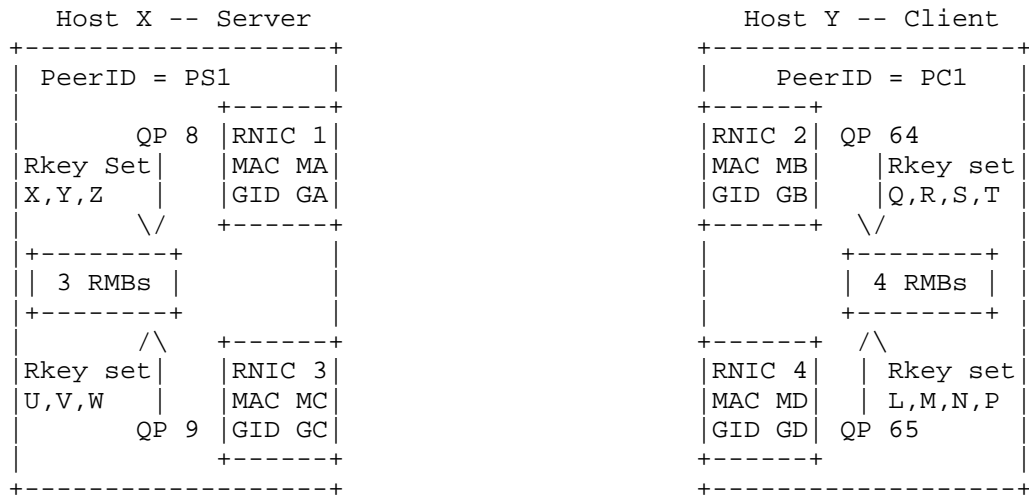
When a new SMC-R link is added to an existing link group, there could be multiple RMBs on each side already associated with the link group. There could also be a different number of RMBs on one side as on the other, because each peer manages its RMBs independently. Each of these RMBs will require a new RToken to be used on the new SMC-R link, and then those new RTokens must be communicated to the peer. This requires two-way communication as the server will have to communicate its RTokens to the client and vice versa.

RTokens are communicated between peers in pairs. Each RToken pair consists of:

- o The RToken for the RMB, as is already known on an existing SMC-R link in the link group
- o The RToken for the same RMB, to be used on the new SMC-R link.

These pairs are required to ensure that each peer knows which RTokens across QPs are equivalent.

The "Add Link" request and response LLC messages do not have room to contain any RToken pairs. "Add Link continuation" LLC messages are used to communicate these pairs, as shown in Figure 15. The "Add Link Continuation" LLC messages are sent on the same SMC-R link that the "Add Link" LLC messages were sent over, and in both the "Add Link" and the "Add Link Continuation" LLC messages, the first RToken in each RToken pair will be the RToken for the RMB as known on the SMC-R link that the LLC message is being sent over.



```

ADD link request (QP9,MC,GC, link number=2)
.....>

ADD link response (QP65,MD,GD, link number=2)
<.....

ADD link continuation req(RToken Pairs=((X,U),(Y,V),(Z,W)))
.....>

ADD link continuation rsp(RToken Pairs=((Q,L),(R,M),(S,N),(T,P)))
<.....

Confirm Link Req/Rsp exchange on link 2
<.....>

```

Legend:

----- TCP/IP and CLC flows

..... RoCE (LLC) flows

Figure 15 Exchanging Rkeys when a new link is added to a link group

3.5.5.3. Serialization of LLC exchanges, and collisions

LLC flows can be divided into two main groups for serializaion considerations.

The first group is LLC messages that are independent and can flow at any time. These are one-time, unsolicited messages that either do

not have a required response, or that have a simple response that does not interfere with the operations of another group of messages. These messages are:

- o TEST LINK from either the client or the server: This message requires a TEST LINK response to be returned, but does not affect the configuration of the link group or the Rkeys.
- o ADD LINK from the client to the server: This message is provided as an "FYI" to the server to let it know that the client has an additional RNIC available. The server is not required to act upon or respond to this message.
- o DELETE_LINK from the client to the server: This message informs the server that the client has either experienced an error or problem that requires a link or link group to be terminated, or that an operator has commanded that a link or link group be terminated. The server does not respond directly to the message, rather it initiates a DELETE LINK exchange as a result of receiving it.
- o DELETE LINK from the server to the client with the "delete entire link group" flag set: This message informs the client that the entire link group is being deleted.

The second group is LLC messages that are part of an exchange of LLC messages that affects link group configuration that must complete before another exchange of LLC messages that affects link group configuration can be processed. When a peer knows that one of these exchanges is in progress, it must not start another exchange. These exchanges are:

- o ADD LINK / ADD LINK response / ADD LINK CONTINUATION / ADD LINK CONTINUATION response / CONFIRM LINK / CONFIRM LINK RESPONSE: This exchange, by adding a new link, changes the configuration of the link group.
- o DELETE LINK / DELETE LINK response initiated by the server: This exchange, by deleting a link, changes the configuration of the link group.

- o CONFIRM RKEY / CONFIRM RKEY response or DELETE RKEY / DELETE RKEY response: This exchange changes the RMB configuration of the link group. RKeys can not change while links are being added or deleted (while ADD or DELETE LINK is in progress). However, CONFIRM RKEY and DELETE RKEY are unique in that both the client and server can independently manage (add or remove) their own RMBs. This allows each peer to concurrently change their RKeys and therefore concurrently send CONFIRM RKEY or DELETE RKEY requests. The concurrent CONFIRM RKEY or DELETE RKEY requests can be independently processed and do not represent a collision

Because the server is in control of the configuration of the link group, many timing windows and collisions are avoided but there are still some that must be handled.

3.5.5.3.1. Collisions with ADD LINK / CONFIRM LINK exchange

Colliding LLC message: TEST LINK

Action to resolve: Send immediate TEST LINK reply

Colliding LLC Message: ADD LINK from client to server

Action to resolve: Server ignores the ADD LINK message. When client receives server's ADD LINK, client will consider that message to be in response to its ADD LINK message and the flow works. Since both client and server know not to start this exchange if an ADD LINK operation is already underway, this can only occur if the client sends this message before receiving the server's ADD LINK and this message crosses with the server's ADD LINK message, therefore the server's ADD LINK arrives at the client immediately after the client sent this message.

Colliding LLC Message: DELETE LINK from client to server, specific link specified

Action to resolve: Server queues the DELETE link message and processes after the ADD LINK exchange completes. If it is an orderly link termination, it can wait until after this exchange continues. If it is disorderly and the link affected is the one that the current exchange is using, the server will discover the outage when a message in this exchange fails.

Colliding LLC Message: DELETE LINK from client to server, entire link group to be deleted

Action to resolve: Immediately clean up the link group

Colliding LLC message: CONFIRM RKEY from the client

Action to resolve: Send negative CONFIRM_RKEY response to the client. Once the current exchange finishes, client will have to recompute its Rkey set to include the new link, and start a new CONFIRM RKEY exchange.

3.5.5.3.2. Collisions during DELETE LINK exchange

Colliding LLC Message: TEST LINK from either peer

Action to resolve: Send immediate TEST LINK response

Colliding LLC message: ADD LNK from client to server

Action to resolve: Server queues the ADD LINK and processes it after the current exchange completes

Colliding LLC message: DELETE LINK from client to server (specific link)

Action to resolve: Server queues the DELETE link message and processes after the current exchange completes. If it is an orderly link termination, it can wait until after this exchange continues. If it is disorderly and the link affected is the one that the current exchange is using, the server will discover the outage when a message in this exchange fails

Colliding LLC message: DELETE LINK from either client or server, deleting the entire link group

Action to resolve: immediately clean up the link group

Colliding LLC message: CONFIRM_RKEY from client to server

Action to resolve: Send negative CONFIRM_RKEY response to the client. Once the current exchange finishes, client will have to recompute its Rkey set to include the new link, and start a new CONFIRM RKEY exchange

3.5.5.3.3. Collisions during CONFIRM_RKEY exchange

Colliding LLC Message: TEST LINK

Action to resolve: Send immediate TEST LINK reply

Colliding LLC message: ADD LINK from client to server

Action to resolve: Queue the ADD LINK and process it after the current exchange completes

Colliding LLC message: ADD LINK from server to client (CONFIRM RKEY exchange was initiated by the client and it crossed with the server initiating an ADD LINK exchange)

Action to resolve: Process the ADD LINK. Client will receive a negative CONFIRM RKEY from the server and will have to redo this CONFIRM RKEY exchange after the ADD LINK exchange completes.

Colliding LLC message: DELETE LINK from client to server, specific link to be deleted (CONFIRM RKEY exchange was initiated by the server and it crossed with the client's DELETE LINK request

Action to resolve: Server queues the DELETE link message and processes after the ADD LINK exchange completes. If it is an orderly link termination, it can wait until after this exchange continues. If it is disorderly and the link affected is the one that the current exchange is using, the server will discover the outage when a message in this exchange fails.

Colliding LLC message: DELETE LINK from server to client, specific link deleted (CONFIRM RKEY exchange was initiated by the client and it crossed with the server's DELETE LINK)

Action to resolve: Process the DELETE LINK. Client will receive a negative CONFIRM RKEY from the server and will have to redo this CONFIRM RKEY exchange after the ADD LINK exchange completes.

Colliding LLC message: DELETE LINK from either client or server, entire link group deleted

Action to resolve: immediately clean up the link group

Colliding LLC message: CONFIRM LINK from the peer that did not start the current CONFIRM LINK exchange

Action to resolve: Queue the request and process it after the current exchange completes.

4. SMC-R memory sharing architecture

4.1. RMB element allocation considerations

Each TCP connection using SMC-R must be allocated a RMBE by each SMC-R peer. This allocation is performed by each end point independently to allow each end point to select an RMBE that best matches the characteristics on its TCP socket end point. The RMBE associated with a TCP socket endpoint must have a Receive buffer that is at least as large as the TCP receive buffer size in effect for that connection. The receive buffer size can be determined by what is specified explicitly by the application using `setsockopt()` or implicitly via the system configured default value. This will allow sufficient data to be RDMA written by the SMC-R peer to fill an entire receive buffer size worth of data on a given data flow. Given that each RMB must have fixed length RMBEs this implies that an SMC-R end point may need to maintain multiple RMBs of various sizes for SMC-R connections on a given SMC link and can then select an RMBE that most closely fits a connection.

4.2. RMB and RMBE format

An RMB is a virtual memory buffer whose backing real memory is pinned, which is divided into a whole number of equal sized RMB Elements (RMBEs). Each RMBE begins with a four byte eye catcher for diagnostic and service purposes, followed by the receive data buffer. The contents of this diagnostic eyecatcher are implementation dependent and should be used by the local SMC-R peer to check for overlay errors by verifying an intact eyecatcher with every RMBE access.

The RMBE is a wrapping receive buffer for receiving RDMA writes from the peer. Cursors, as described below, are exchanged between peers to manage and track RDMA writes and local data reads from the RMBE for a TCP connection.

4.3. RMBE control information

RMBE control information consists of consumer and producer cursors, wrap counts, CDC message sequence numbers, control flags such as urgent data and writer blocked indicators, and TCP connection

information such as termination flags. This information is exchanged between SMC-R peers using CDC messages, which are passed using RDMA message passing with inline data, with the control information contained in the inline data. A TCP/IP stack implementing SMC-R must receive and store this information in its internal data structures as it is used to manage the RMBE and its data buffer.

The format and contents of the CDC message is described in detail in 4.3. RMBE control information. The following is a high level description of what this control information contains.

- o Connection state flags such as sending done, connection closed, failover data validation, and abnormal close
- o A sequence number that is managed by the sender. This sequence number starts at 1, is increased each send, and wraps to 0. This sequence number tracks the CDC message sent and is not related to the number of bytes sent. It is used for failover data validation.
- o Producer cursor: a wrapping offset into the receiver's RMBE data area. Set by the peer that is writing into the RMBE, it points to where the writing peer will write the next byte of data into an RMBE. This cursor is accompanied by a wrap sequence number to help the RMBE owner (the receiver) identify full window size wrapping writes. Note that this cursor must account for (i.e., skip over) the RMBE eyecatcher that is in the beginning of the data area.
- o Consumer cursor: a wrapping offset into the receiver's RMBE data area. Set by the owner of the RMBE (the peer that is reading from it), this cursor points to the offset of the next byte of data to be consumed by the peer in its own RMBE. The sender cannot write beyond this cursor into the receiver's RMBE without causing data loss. Like the producer cursor, this is accompanied by a wrap count to help the writer identify full window size wrapping reads. Note that this cursor must account for (i.e., skip over) the RMBE eyecatcher that is in the beginning of the data area.
- o Data flags such as urgent data, writer blocked indicator, and cursor update requests.

4.4. Use of RMBEs

4.4.1. Initializing and accessing RMBEs

The RMBE eyecatcher is initialized by the RMB owner prior to assigning it to a specific TCP connection and communicating its RMB

index to the SMC-R partner. After an RMBE index is communicated to the SMC-R partner the RMBE can only be referenced in "read only mode" by the owner and all updates to it are performed by the remote SMC-R partner via RDMA write operations.

Initialization of an RMBE must include the following:

- o Zeroing out the entire RMBE receive buffer, which helps minimize data integrity issues (e.g. data from a previous connection somehow being presented to the current connection).
- o Setting the beginning RMBE eye catcher. This eye catcher plays an important role in helping detect accidental overlays of the RMBE. The RMB owner must always validate these eye catchers before each new reference to the RMBE. If the eye catchers are found to be corrupted the local host must reset the TCP connection associated with this RMBE and log the appropriate diagnostic information.

4.4.2. RMB element reuse and conflict resolution

RMB elements can be reused once their associated TCP and SMC-R connections are terminated. Under normal and abnormal SMC-R connection termination processing both SMC-R peers must explicitly acknowledge that they are done using an RMBE before that element can be freed and reassigned to another SMC-R connection instance. For more details on SMC-R connection termination refer to section 4.8. However, there are some error scenarios where this 2 way explicit acknowledgement may not be completed. In these scenarios (mentioned explicitly elsewhere in this document) an RMBE owner may chose to re-assign this RMBE to a new SMC-R connection instance on this SMC link group. When this occurs the partner SMC-R peer must detect this condition during SMC-R rendezvous processing when presented with an RMBE that it believes is already in use for a different SMC-R connection. In this case, the SMC-R peer must abort the existing SMC-R connection associated with this RMBE. The abort processing Resets the TCP connection (if it is still active) but it must not attempt to perform any RDMA writes to this RMBE and must also ignore any data sitting in the local RMBE associated with the existing connection. It then proceeds to free up the local RMBE and notify the local application that the connection is being abnormally reset.

The remote SMC-R peer then proceeds to normal processing for this new SMC-R connection.

4.5. SMC-R protocol considerations

The following sections describe considerations for the SMC-R protocol as compared to the TCP protocol.

4.5.1. SMC-R protocol optimized window size updates

An SMC-R receiver host sends its Consumer Cursor information to the sender to convey the progress that the receiving application has made in consuming the sent data. The difference between the writer's Producer Cursor and the associated receiver's Consumer Cursor indicates the window size available for the sender to write into. This is somewhat similar to TCP window update processing and therefore has some similar considerations, such as silly window syndrome avoidance, whereby the TCP protocol has an optimization that minimizes the overhead of very small, unproductive window size updates associated with sub-optimal socket applications consuming very small amount of data on every receive() invocation. For SMC-R, the receiver only updates its Consumer Cursor via a unique CDC message under the following conditions:

- o The current window size (from a sender's perspective) is less than half of the Receive Buffer space and the Consumer Cursor update will result in a minimum increase in the window size of 10% of the Receive buffer space. Some examples:
 - a. Receive Buffer size: 64K, Current window size (from a sender's perspective): 50K. No need to update the Consumer Cursor. Plenty of space is available for the sender.
 - b. Receive Buffer size: 64K, Current window size (from a sender's perspective): 30K, Current window size from a receiver's perspective: 31K. No need to update the Consumer Cursor; even though the sender's window size < 1/2 of the 64K, the window update would only increase that by 1K which is < 1/10th of the 64K buffer size.
 - c. Receive Buffer size: 64K, Current window size (from a sender's perspective): 30K, Current window size from a receiver's perspective: 64K. The receiver updates the Consumer Cursor (sender's window size < 1/2 of the 64K, the window update would increase that by > 6.4K).

- o The receiver must always include a Consumer Cursor update whenever it sends a CDC message to the partner for another flow (i.e. send flow in the opposite direction). This allows the window size update to be delivered with no additional overhead. This is somewhat similar to TCP DelayAck processing and quite effective for request/response data patterns.
- o If a peer has set the B-bit in a CDC message then any consumption of data by the receiver causes a CDC message to be sent updating the consumer cursor until that a CDC message with that bit cleared is received from the peer.
- o The optimized window size updates are overridden when the sender sets the Consumer Cursor Update Requested flag in a CDC message to the receiver. When this indicator is on the consumer must send a Consumer Cursor update immediately when data is consumed by the local application or if the cursor has not been updated for a while (i.e. local copy consumer cursor does not match the last consumer cursor value sent to the the partner). This allows the sender to perform optional diagnostics for detecting a stalled receiver application (data has been sent but not consumed). It is recommended that the Consumer Cursor Update Requested flag only be sent for diagnostic procedures as it may result in non-optimal data path performance.

4.5.2. Small data sends

The SMC-R protocol makes no special provisions for handling small data segments sent across a stream socket. Data is always sent if sufficient window space is available. There are no special provisions for coalescing small data segments, similar to the TCP Nagle algorithm.

An implementation of SMC-R may optimize its sending processing by coalescing outbound data for a given SMC-R connection so that it can reduce the number of RDMA write operations it performed in a similar fashion to Nagle's algorithm. However, any such coalescing would require a timer on the sending host that would ensure that data was eventually sent. And the sending host would have to opt out of this processing if Nagle's algorithm had been disabled (programmatically or via system configuration).

4.5.3. TCP Keepalive processing

TCP keepalive processing allows applications to direct the local TCP/IP host to periodically "test" the viability of an idle TCP connection. Since SMC-R connections have both a TCP representation

along with an SMC-R representation there are unique keepalive processing considerations:

- o SMC-R layer keepalive processing: If keepalive is enabled for an SMC-R connection the local host maintains a keepalive timer that reflects how long an SMC-R connection has been idle. The local host also maintains a timestamp of last activity for each SMC link (for any SMC-R connection on that link). When it is determined that an SMC-R connection has been idle longer than the keepalive interval the host checks whether the SMC-R link has been idle for a duration longer than the keepalive timeout. If both conditions are met, the local host then performs a Test Link LLC command to test the viability of the SMC link over the RoCE fabric (RC-QPs). If a Test Link LLC command response is received within a reasonable amount of time then the link is considered viable and all connections using this link are considered viable as well. If however a response is not received in a reasonable amount of time or there's a failure in sending the Test Link LLC command then this is considered a failure in the SMC link and failover processing to an alternate SMC link must be triggered. If no alternate SMC link exists in the SMC link group then all the SMC-R connections on this link are abnormally terminated by resetting the TCP connections represented by these SMC-R connections. Given that multiple SMC-R connections can share the same SMC link, implementing an SMC link level probe using the Test Link LLC command will help reduce the amount of unproductive keepalive traffic for SMC-R connections; as long as some SMC-R connections on a given SMC link are active (i.e. have had I/O activity within the keepalive interval) then there is no need to perform additional link viability testing.
- o TCP layer keepalives processing: Traditional TCP "keepalive" packets are not as relevant for SMC-R connections given that the TCP path is not used for these connections once the SMC-R rendezvous processing is completed. All SMC-R connections by default have associated TCP connections that are idle. Are TCP keepalive probes still needed for these connections? There are two main scenarios to consider:
 1. TCP keepalives that are used determine whether the peer TCP endpoint is still active. This is not needed for SMC-R connections as the SMC-R level keepalives mentioned above will determine whether the remote endpoint connections are still active.

2. TCP keepalives that are used to ensure that TCP connections traversing an intermediate proxy maintain an active state. For example, stateful firewalls typically maintain state representing every valid TCP connection that traverses the firewall. These types of firewalls are known to expire idle connections by removing their state in the firewall to conserve memory. TCP keepalives are often used in this scenario to prevent firewalls from timing out otherwise idle connections. When using SMC-R, both end points must reside in the same layer 2 network (i.e. the same subnet). As a result, firewalls can not be injected in the path between two SMC-R endpoints. However, other intermediate proxies, such as TCP/IP layer load balancers may be injected in the path of two SMC-R endpoints. These types of load balancers also maintain connection state so that they can forward TCP connection traffic to the appropriate cluster end point. When using SMC-R these TCP connections will appear to be completely idle making them susceptible to potential timeouts at the LB proxy. As a result, for this scenario, TCP keepalives may still be relevant.

The following are the TCP level keepalive processing requirements for SMC-R enabled hosts:

- o SMC-R peers should allow TCP keepalives to flow on the TCP path of SMC-R connections based on existing TCP keepalive configuration and programming options. However, it is strongly recommended that platforms provide the ability to specify very granular keepalive timers (for example, single digit second timers) should consider providing a configuration option that limits the minimum keepalive timer that will be used for TCP layer keepalives on SMC-R connections. This is important to minimize the amount of TCP keepalive packets transmitted in the network for SMC-R connections.
- o SMC-R peers must always respond to inbound TCP layer keepalives (by sending ACKs for these packets) even if the connection is using SMC-R. Typically, once a TCP connection has completed the SMC-R rendezvous processing and using SMC-R for data flows, no new inbound TCP segments are expected on that TCP connection other than TCP termination segments (FIN, RST, etc). TCP keepalives are the one exception that must be supported. And since TCP keepalive probes do not carry any application layer data this has no adverse impact on the application's inbound data stream.

4.6. TCP connection failover between SMC-R links

A peer may change which SMC-R link within a link group it sends its writes over in the event of a link failure. Since each peer independently chooses which link to send writes over for a specific TCP connection, this process is done independently by each peer.

4.6.1. Validating data integrity

Even though RoCE is a reliable transport there is a small subset of failure modes that could cause unrecoverable loss of data. When an RNIC acknowledges receipt of an RDMA write to its peer, that creates a write completion event to the sending peer, which allows the sender to release any buffers it is holding for that write. In normal operation and in most failures, this operation is reliable.

However there are failure modes possible in which a receiving RNIC has acknowledged an RDMA write but then was not able to place the received data into its host memory, for example a sudden, disorderly failure of the interface between the RNIC and the host. While rare, these types of events must be guarded against to ensure data integrity. The process for switching SMC-R links during failover that is described in this section guards against this possibility, and is mandatory.

Each peer must track the current state of the CDC sequence numbers for a TCP connection. The sender must keep track of SS, which is the sequence number of the CDC message that described the last write acknowledged by the peer RNIC. In other words, SS describes the last write that the sender believes its peer has successfully received. The receiver must keep track of SR, the sequence number of the CDC message that described last write that it has successfully received, i.e., the data has been successfully placed into an RMBE.

When an RNIC fails and the sender changes SMC-R links, the sender must first send a CDC message with the 'F' flag set over the new SMC-R link. This is the failover data validation message. The sequence number in this CDC message is equal to SS. The CDC message key, the length, and SMC-R alert token are the only other fields in this CDC message that are significant. No reply is expected from this validation message, and once the sender has sent it, the sender may resume sending on the new SMC-R link as described in 4.6.2. below

Upon receipt of the failover validation message, the receiver must verify that its SR value for the TCP connection is equal to or greater than the sequence number in the failover validation message. If so, no further action is required and the TCP connection resumes

on the new SMC-R link. If SR is less than the sequence number value in the validation message, data has been lost and the receiver must immediately reset the TCP connection.

4.6.2. Resuming the TCP connection on a new SMCR link

When a connection is moved to a new SMC-R link and the failover validation message has been sent, the sender can immediately resume normal transmission. In order to preserve the application message stream the sender must replay any RDMA writes (and their associated CDC messages) that were in progress or failed when the previous SMC-R link failed, before sending new data on the new SMC-R link. The sender has two options for accomplishing this:

- o Preserve the sequence numbers "as is": Retry all failed and pending operations as they were originally done, including reposting all associated RDMA write operations and their associated CDC messages without making any changes. Then resume sending new data using new sequence numbers.
- o Combine pending messages and possibly add new data: Combine failed and pending messages into a single new write with a new sequence number. This allows the sender to combine pending messages into fewer operations. As a further optimization this write can also include new data, as long as all failed and pending data is also included. If this approach is taken, the sequence number must be increased beyond the last failed or pending sequence number.

4.7. RMB data flows

The following sections describe the RDMA wire flows for the SMC-R protocol after a TCP connection has switched into SMC-R mode (i.e. SMC-R rendezvous processing is complete and a pair of RMB elements has been assigned and communicated by the SMC-R peers). The ladder diagrams below include the following:

- o RMBE control information kept by each peer. Only a subset of the information is depicted, specifically only the fields that reflect the stream of data written by Host A and read by Host B.
- o Time line 0-x that shows the wire flows in a time relative fashion
- o Note that RMBE control information is only shown in a time interval if its value changed (otherwise assume the value is unchanged from previously depicted value)

- o The local copy of the producer and consumer cursors that is maintained by each host is not depicted in these figures. Note that the cursor values in the diagram reflect the necessity of skipping over the eyecatcher in the RMBE data area. They start and wrap at 4, not 0.

4.7.1. Scenario 1: Send flow, window size unconstrained

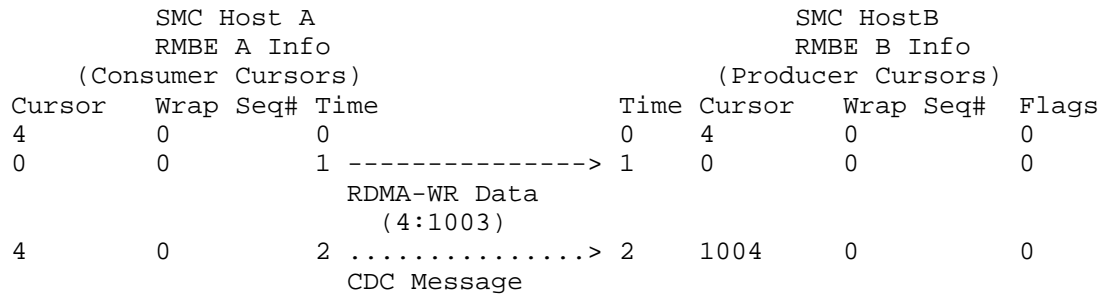


Figure 16 Scenario 1: Send flow, window size unconstrained

Scenario assumptions:

- o Kernel implementation
- o New SMC-R connection, no data has been sent on the connection
- o Host A: Application issues send for 1,000 bytes to Host B
- o Host B: RMBE receive buffer size is 10,000, application has issued a recv for 10,000 bytes

Flow description:

1. Application issues send() for 1,000 bytes, SMC-R layer copies data into a kernel send buffer. It then schedules an RDMA write operation to move the data into the peer's RMBE receive buffer, at relative position 4-1003 (to skip the four byte eyecatcher in the RMBE data area). Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation.

2. Host A sends a CDC message to update the Producer Cursor to byte 1004. This CDC message will deliver an interrupt to Host B. At this point, the SMC-R layer can return control back to the application. Host B, once notified of the completion of the previous RDMA operation, locates the RMBE associated with the RMBE alert token that was included in the message and proceeds to perform normal receive side processing, waking up the suspended application read thread, copying the data into the application's receive buffer, etc. It will use the Producer Cursor as an indicator of how much data is available to be delivered to the local application. After this processing is complete, the SMC-R layer will also update its local Consumer Cursor to match the Producer Cursor (i.e. indicating that all data has been consumed). Note that a message to the peer updating the Consumer Cursor is not needed at this time as the window size is unconstrained ($> 1/2$ of the receive buffer size). The window size is calculated using by taking the difference between the Producer and the Consumer cursors in the RMBEs ($10,000 - 1,004 = 8,996$).

4.7.2. Scenario 2: Send/Receive flow, window unconstrained

SMC Host A				SMC HostB				
RMBE A Info				RMBE B Info				
(Consumer Cursors)				(Producer Cursors)				
Cursor	Wrap	Seq#	Time	Time	Cursor	Wrap	Seq#	Flags
4	0	0		0	4	0		0
0	0	1	----->	1	0	0		0
RDMA-WR Data								
(4:1003)								
4	0	2>	2	1004	0		0
CDC Message								
0	0	3	<-----	3	1004	0		0
RDMA-WR Data								
(4:503)								
1004	0	4	<.....	4	1004	0		0
CDC Message								

Figure 17 Scenario 2: Send/Recv flow, window size unconstrained

Scenario assumptions:

- o New SMC-R connection, no data has been sent on the connection
- o Host A: Application issues send for 1,000 bytes to Host B
- o Host B: RMBE receive buffer size is 10,000, application has already issued a recv for 10,000 bytes. Once the receive is completed, the application sends a 500 byte response to Host A.

Flow description:

1. Application issues send() for 1,000 bytes, SMC-R layer copies data into a kernel send buffer. It then schedules an RDMA write operation to move the data into the peer's RMBE receive buffer, at relative position 4-1003. Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation.
2. Host A sends a CDC message to update the Producer Cursor to byte 1004. This CDC message will deliver an interrupt to Host B. At this point, the SMC-R layer can return control back to the application.

3. Host B, once notified of the receipt of the previous CDC message, locates the RMBE associated with the RMBE alert token and proceeds to perform normal receive side processing, waking up the suspended application read thread, copying the data into the application's receive buffer, etc. After this processing is complete, the SMC-R layer will also update its local Consumer Cursor to match the Producer Cursor (i.e. indicating that all data has been consumed). Note that an update of the Consumer Cursor to the peer is not needed at this time as the window size is unconstrained ($> 1/2$ of the receive buffer size). The application then performs a send() for 500 bytes to Host A. The SMC-R layer will copy the data into a kernel buffer and then schedule an RDMA Write into the partner's RMBE receive buffer. Note that this RDMA write operation includes no immediate data or notification to Host A.
4. Host B sends a CDC message to update the partner's RMBE Control information with the latest Producer Cursor (set to 503 and not shown in the diagram above) and to also inform the peer that the Consumer Cursor value is now 1004. It also updates the local Current Consumer Cursor and Last Sent Consumer Cursor to 1004. This CDC message includes notification since we are updating our Producer Cursor which requires attention by the peer host.

4.7.3. Scenario 3: Send Flow, window constrained

SMC Host A				SMC HostB				
RMBE A Info				RMBE B Info				
(Consumer Cursors)				(Producer Cursors)				
Cursor	Wrap	Seq#	Time	Time	Cursor	Wrap	Seq#	Flags
4	0	0		0	4	0		0
4	0	1	----->	1	4	0		0
RDMA-WR Data (4:3003)								
4	0	2>	2	3004	0		0
CDC Message								
4	0	3		3	3004	0		0
4	0	4	----->	4	3004	0		0
RDMA-WR Data (3004:7003)								
4	0	5>	5	7004	0		0
CDC Message								
7004	0	6	<.....	6	7004	0		0
CDC Message								

Figure 18 Scenario 3: Send Flow, window size constrained

Scenario assumptions:

- o New SMC-R connection, no data has been sent on this connection
- o Host A: Application issues send for 3,000 bytes to Host B and then another send for 4,000
- o Host B: RMBE receive buffer size is 10,000. Application has already issued a recv for 10,000 bytes

Flow description:

1. Application issues send() for 3,000 bytes, SMC-R layer copies data into a kernel send buffer. It then schedules an RDMA write operation to move the data into the peer's RMBE receive buffer, at relative position 4-3003. Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation.
2. Host A sends a CDC message to update its Producer Cursor to byte 3003. This CDC message will deliver an interrupt to Host B. At this point, the SMC-R layer can return control back to the application.
3. Host B, once notified of the receipt of the previous CDC message, locates the RMBE associated with the RMBE alert token and proceeds to perform normal receive side processing, waking up the suspended application read thread, copying the data into the application's receive buffer, etc. After this processing is complete, the SMC-R layer will also update its local Consumer Cursor to match the Producer Cursor (i.e. indicating that all data has been consumed). It will not however update the partner with this information as the window size is not constrained (10000-3000=7000 of available space). The application on Host B also issues a new recv() for 10,000.
4. On Host A, application issues a send() for 4,000 bytes. The SMC-R layer copies the data into a kernel buffer and schedules an async RDMA write into the peer's RMBE receive buffer at relative position 3003-7004. Note that no alert is provided to host B for this flow.
5. Host A sends a CDC message to update the Producer Cursor to byte 7004. This CDC message will deliver an interrupt to Host B. At this point, the SMC-R layer can return control back to the application.

6. Host B, once notified of the receipt of the previous CDC message, locates the RMBE associated with the RMBE alert token and proceeds to perform normal receive side processing, waking up the suspended application read thread, copying the data into the application's receive buffer, etc. After this processing is complete, the SMC-R layer will also update its local Consumer Cursor to match the Producer Cursor (i.e. indicating that all data has been consumed). It will then determine whether it needs to update the Consumer Cursor to the peer. The available window size is now 3,000 (10,000 - (Producer Cursor - Last Sent Consumer Cursor)) which is < 1/2 receive buffer size (10,000/2=5,000) and the advance of the window size is > 10% of the windows size (1,000). Therefore a CDC message is issued to update the Consumer Cursor to peer A.

4.7.4. Scenario 4: Large send, flow control, full window size writes

SMC Host A				SMC HostB				
RMBE A Info				RMBE B Info				
(Consumer Cursors)				(Producer Cursors)				
Cursor	Wrap	Seq#	Time	Time	Cursor	Wrap	Seq#	Flags
1004	1	0		0	1004	1	0	
1004	1	1	----->	1	1004	1	1	0
RDMA-WR Data (1004:9999)								
1004	1	2	----->	2	1004	1	2	0
RDMA-WR Data (4:1003)								
1004	1	3>	3	1004	2	3	Wrt
CDC Message								
1004	2	4	<.....	4	1004	2	4	Wrt
CDC Message								
1004	2	5	----->	5	1004	2	5	Wrt Blk
RDMA-WR Data (1004:9999)								
1004	2	6	----->	6	1004	2	6	Wrt Blk
RDMA-WR Data (4:1003)								
1004	2	7>	7	1004	3	7	Wrt
CDC Message								
1004	3	8	<.....	8	1004	3	8	Wrt
CDC Message								

Figure 19 Scenario 4: Large send, flow control, full window size writes

Scenario assumptions:

- o Kernel implementation
- o Existing SMC-R connection, Host B's receive window size is fully open (Peer Consumer Cursor = Peer Producer Cursor).
- o Host A: Application issues send for 20,000 bytes to Host B
- o Host B: RMB receive buffer size is 10,000, application has issued a recv for 10,000 bytes

Flow description:

1. Application issues send() for 20,000 bytes, SMC-R layer copies data into a kernel send buffer (assumes send buffer space of 20,000 is available for this connection). It then schedules an RDMA write operation to move the data into the peer's RMBE receive buffer, at relative position 1004-9999. Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation.
2. Host A then schedules an RDMA write operation to fill the remaining 1000 bytes of available space in the peer's RMBE receive buffer, at relative position 4-1003. Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation. Also note that an implementation of SMC-R may optimize this processing by combining step 1 and 2 into a single RDMA Write operation (with 2 different data sources).
3. Host A sends CDC message to update the Producer Cursor to byte 1004. Since the entire receive buffer space is filled, the Producer Writer Blocked flag (WrtBlk indicator above) is set and the Producer Window Wrap Sequence Number (Producer WrapSeq# above) is incremented. This CDC message will deliver an interrupt to Host B. At this point, the SMC-R layer can return control back to the application.

4. Host B, once notified of the receipt of the previous CDC message, locates the RMBE associated with the RMBE alert token and proceeds to perform normal receive side processing, waking up the suspended application read thread, copying the data into the application's receive buffer, etc. In this scenario, Host B notices that the Producer Cursor has not been advanced (same value as Consumer Cursor), however, it notices that the Producer Window Wrap Size Sequence number is different from its local value (1) indicating that a full window of new data is available. All the data in the receive buffer can be processed, the first segment (1004-9999) followed by the second segment (4-1003). Because the Producer Writer Blocked indicator was set, Host B schedules a CDC message to update its latest information to the peer: Consumer Cursor (1004), Consumer Window Wrap Size Sequence Number (2: the current Producer Window Wrap Sequence Number is used).
5. Host A, upon receipt of the CDC message locates the TCP connection associated with the alert token, and upon examining the control information provided notices that Host B has consumed all of the data (based on the Consumer Cursor and the Consumer Window Wrap Size Sequence number) and initiates the next RDMA write to fill the receive buffer at offset 1003-9999.
6. Host A then moves the next 1000 bytes into the beginning of the receive buffer (4-1003) by scheduling an RDMA write operation. Note at this point there are still 8 bytes remaining to be written.
7. Host A then sends a CDC message to set the Producer Writer Blocked indicator and to increment the Producer Window Wrap Size Sequence Number (3).
8. Host B, upon notification completes the same processing as step 4 above, including sending a CDC message to update the peer to indicate that all data has been consumed. At this point Host A can write the final 8 bytes to host B's RMBE into positions 1004-1011 (not shown).

4.7.5. Scenario 5: Send flow, urgent data, window size unconstrained

SMC Host A RMBE A Info (Consumer Cursors)				SMC HostB RMBE B Info (Producer Cursors)			
Cursor	Wrap	Seq#	Time	Time	Cursor	Wrap	Seq# Flag
1000	1	0		0	1000	1	0
1000	1	1	----->	1	1000	1	0
			RDMA-WR Data (1000:1499)				
1000	1	2>	2	1500	1	UrgP
			CDC Message				UrgA
1500	1	3	<.....	3	1500	1	UrgP
			CDC Message				UrgA
1500	1	4	----->	4	1500	1	UrgP
			RDMA-WR Data (1500:2499)				UrgA
1500	1	5>	5	2500	1	0
			CDC Message				

Figure 20 Scenario 5: send Flow, urgent data, window size open

Scenario assumptions:

- o Kernel implementation
- o Existing SMC-R connection, window size open, all data has been consumed by receiver.
- o Host A: Application issues send for 500 bytes with urgent data indicator (OOB) to Host B, then sends 1000 of normal data
- o Host B: RMBE Receive buffer size is 10,000, application has issued a recv for 10,000 bytes and is also monitoring the socket for urgent data

Flow description:

1. Application issues send() for 500 bytes of urgent data. SMC-R layer copies data into a kernel send buffer. It then schedules an RDMA write operation to move the data into the peer's RMBE receive buffer, at relative position 1000-1499. Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation.

2. Host A sends a CDC message to update its Producer Cursor to byte 1500 and to turn on the Producer Urgent Data Pending (UrgP) and Urgent Data Present (UrgA) flags. This CDC message will deliver an interrupt to Host B. At this point, the SMC-R layer can return control back to the application.
3. Host B, once notified of the receipt of the previous CDC message, locates the RMBE associated with the RMBE alert token, notices that the Urgent Data Pending flag is on and proceeds with Out of Band socket API notification. For example, satisfying any outstanding select() or poll() requests on the socket by indicating that urgent data is pending (i.e. by setting the exception bit on). The Urgent Data Present indicator allows Host B to also determine the position of the urgent data (Producer cursor points one byte beyond the last byte of urgent data). Host B can then perform normal receive side processing (including specific urgent data processing), copying the data into the application's receive buffer, etc. Host B then sends a CDC message to update the partner's RMBE Control area with its latest Consumer Cursor (1500). Note this CDC message must occur regardless of the current local window size that is available. The partner host (Host A) cannot initiate any additional RDMA writes until acknowledgement that the urgent data has been processed (or at least processed/remembered at the SMC-R layer).
4. Upon receipt of the message, Host A wakes up, sees that peer consumed all data up to and including the last byte of Urgent data and now resumes sending any pending data. In this case, the application had previously issued a send for 1000 bytes of normal data which would have been copied in the send buffer and control would have been returned to the application. Host A now initiates a RDMA write to move that data to the Peer's receive buffer at position 1500-2499.
5. Host A then sends a CDC message with inline data update its Producer Cursor value (2500) and turn off the Urgent Data Pending and Urgent Data Present flags. Host B wakes up, processes the new data (resumes application, copies data into the application receive buffer) and then proceeds to update the Local current consumer cursor (2500). Given that the window size is unconstrained there is no need for Consumer Cursor update in the peer's RMBE.

4.7.6. Scenario 6: Send flow, urgent data, window size closed

SMC Host A RMBE A Info (Consumer Cursors)				SMC HostB RMBE B Info (Producer Cursors)			
Cursor	Wrap	Seq#	Time	Time	Cursor	Wrap	Seq# Flag
1000	1	0		0	1000	2	Wrt Blk
1000	1	1>	1	1000	2	Wrt Blk UrgP
			CDC Message				
1000	2	2	<.....	2	1000	2	Wrt Blk UrgP
			CDC Message				
1000	2	3	----->	3	1000	2	Wrt Blk UrgP
			RDMA-WR data 1 (1000:1499)				
1000	2	4>	4	1500	2	UrgP UrgA
			CDC Message				
1500	2	5	<.....	5	1500	2	UrgP UrgA
			CDC Message				
1500	2	6	----->	6	1500	2	UrgP UrgA
			RDMA-WR data 1 (1500:2499)				
1000	2	7>	7	2500	2	0
			CDC Message				

Figure 21 Scenario 6: Send flow, urgent data, window size closed

Scenario assumptions:

- o Kernel implementation
- o Existing SMC-R connection, window size closed, writer is blocked.
- o Host A: Application issues send for 500 bytes with urgent data indicator (OOB) to Host B, then sends 1000 of normal data.
- o Host B: RMBE Receive buffer size is 10,000, application has no outstanding recv() (for normal data) and is monitoring the socket for urgent data.

Flow description:

1. Application issues send() for 500 bytes of urgent data. SMC-R layer copies data into a kernel send buffer (if available). Since the writer is blocked (window size closed) it cannot send the data immediately. It then sends a CDC message to notify the peer of the Urgent Data Pending (UrgP) indicator (the Writer Blocked indicator remains on as well). This serves as a signal to Host B that urgent data is pending in the stream. Control is also returned to the application at this point.
2. Host B, once notified of the receipt of the previous CDC message, locates the RMBE associated with the RMBE alert token, notices that the Urgent Data Pending flag is on and proceeds with Out of Band socket API notification. For example, satisfying any outstanding select() or poll() requests on the socket by indicating that urgent data is pending (i.e. by setting the exception bit on). At this point it is expected that the application will enter urgent data mode processing, expeditiously processing all normal data (by issuing recv API calls) so that it can get to the urgent data byte. Whether the application has this urgent mode processing or not, at some point the application will consume some or all of the pending data in the receive buffer. When this occurs, Host B will also send a CDC message with inline data to update its Consumer Cursor and Consumer Window Wrap Sequence Number to the peer. In the example above, a full window worth of data was consumed.
3. Host A, once awakened by the message will notice that the window size is now open on this connection (based on the Consumer Cursor and the Consumer Window Wrap Sequence Number which now matches the Producer Window Wrap Sequence Number) and resume sending of the urgent data segment by scheduling an RDMA write into relative position 1000-1499.
4. Host A then sends a CDC message to advance its Producer Cursor (1500) and to also notify Host B of the Urgent Data Present (UrgA) indicator (and turn off the Writer Blocked indicator). This signals to Host B that the urgent data is now in the local receive buffer and that the Producer Cursor points to the last byte of urgent data.
5. Host B wakes up, processes the urgent data and once the urgent data is consumed sends a CDC message with inline data to update its Consumer Cursor (1500).

6. Host A wakes up, sees that Host B has consumed the sequence number associated with the urgent data and then initiates the next RDMA write operation to move the 1000 bytes associated with the next send() of normal data into the peer's receive buffer at position (1500-2499). Note that send() API would have likely completed earlier in the process by copying the 1000 bytes into a send buffer and returning back to the application even though we could not send any new data until the urgent data was processed and acknowledged by Host B.
7. Host A sends a CDC message to advance its Producer Cursor to 2500 and to reset the Urgent Data Pending and Present flags. Host B wakes up and processes the inbound data.

4.8. Connection termination

Just as SMC-R connections are established using a combination of TCP connection establishment flows and SMC-R protocol flows, the termination of SMC-R connections also uses a similar combination of SMC-R protocol termination flows and normal TCP protocol connection termination flows. The following sections describe the SMC-R protocol normal and abnormal connection termination flows.

4.8.1. Normal SMC-R connection termination flows

Normal SMC-R connection flows are triggered via the normal stream socket API semantics, namely by the application issuing a close() or shutdown() API. Most applications, after consuming all incoming data and after sending any outbound data will then issue a close() API to indicate that they are done both sending and receiving data. Some applications, typically a small percentage, make use of the shutdown() API that allows then to indicate that the application is done sending data, receiving data or both sending and receiving data. The main use of this API is scenarios where a TCP application wants to alert its partner end point that it is done sending data, yet is still receiving data on its socket (shutdown for Write). Issuing shutdown for both sending and receiving data is really no different than issuing a close() and can therefore be treated in a similar fashion. Shutdown for read is typically not a very useful operation and in normal circumstances does not trigger any network flows to notify the partner TCP end point of this operation.

These same trigger points will be used by the SMC-R layer to initiate SMC-R connections termination flows. The main design point for SMC-R normal connection flows is to use the SMC-R protocol to first shutdown the SMC-R connection and free up any SMC-R RDMA resources and then allow the normal TCP connection termination protocol (i.e.

FIN processing) to drive cleanup of the TCP connection. This design point is very important in ensuring that RDMA resources such as the RMBEs are only freed and reused when both SMC-R end points are completely done with their RDMA Write operations to the partner's RMBE.

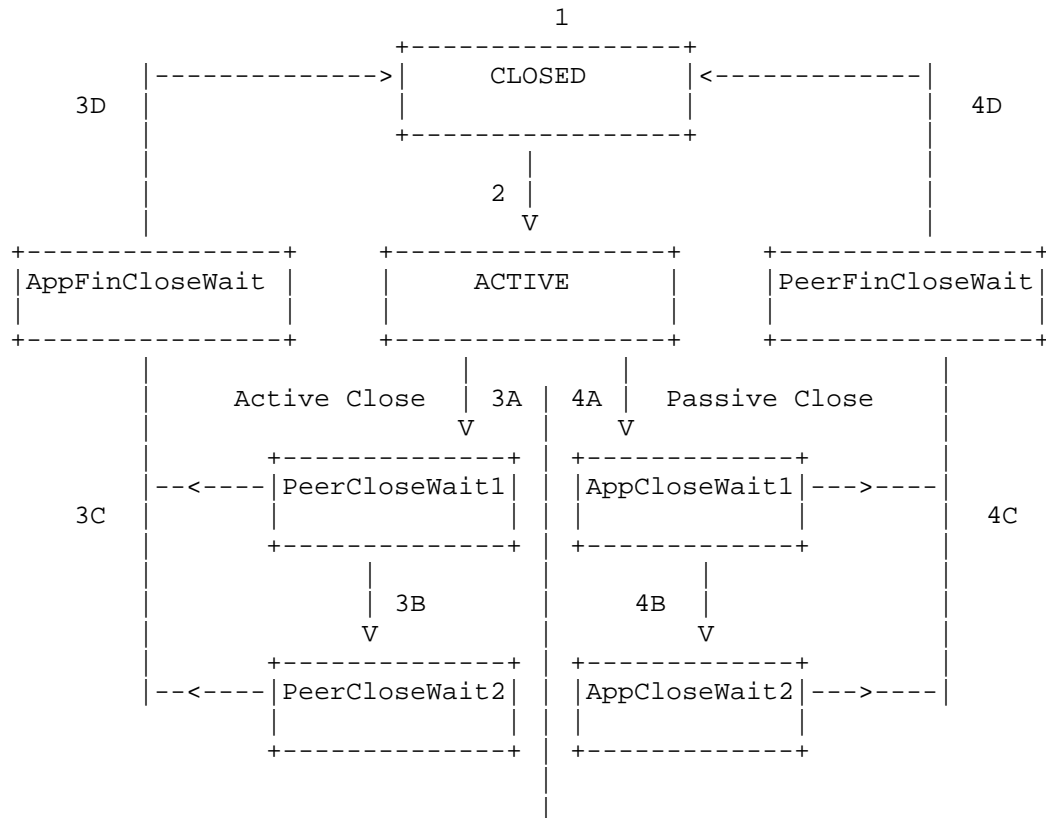


Figure 22 SMC-R connection states

Figure 23 describes the states that an SMC-R connection typically goes through. Note that there are variations to these states that can occur when an SMC-R connection is abnormally terminated, similar in a way to when a TCP connection is reset. The following are the high level state transitions for an SMC-R connection:

1. An SMC-R connection begins in the Closed state. This state is meant to reflect an RMBE that is not currently in use (was previously in use but no longer is or one that was never allocated)
2. An SMC-R connection progresses to the Active state once the SMC-R rendezvous processing has successfully completed, RMB element indices have been exchanged and SMC-R links have been activated. In this state, TCP connection is fully established, rendezvous processing has been completed and SMC-R peers can begin exchange of data via RDMA.
3. Active close processing (on SMC-R peer that is initiating the connection termination)

A. When an application on one of the SMC-R connection peers issues a close() or shutdown(write or both) the SMC-R layer on that host will initiate SMC-R connection termination processing. First if close() or shutdown(both) is issued it will check to see that there's no data in the local RMB element that has not been read by the application. If unread data is detected, the SMC-R connection must be abnormally reset - for more detail on this refer to "SMC-R connection reset". If no unread data is pending, it then checks to see whether any outstanding data is waiting to be written to the peer or if any outstanding RDMA writes for this SMC-R connection have not yet completed. If either of these two scenarios are true, an indicator that this connection is in a pending close state is saved in internal data structures representing this SMC-R connection and control is returned to the application. If all data to be written to the partner has completed this peer will send a CDC message to notify the peer of either the PeerConnectionClosed indicator (close or shutdown for both was issued) or the PeerDoneWriting indicator. This will provide stimulus to the partner SMC-R peer that the connection is terminating. At this point the local side of the SMC-R connection transitions in the PeerCloseWait1 state and control can be returned to the application. If this process could not be completed synchronously (close pending condition mentioned above) it is completed when all RDMA writes for data and control cursors have been completed.

B. At some point the SMC-R peer application (passive close) will consume all incoming data, realize that that partner is done sending data on this connection and proceed to initiate its own close of the connection once it has completed sending all data from its end. The partner application can initiate this connection termination processing via a close() or shutdown()

APIs. If the application does so by issuing a shutdown() for write, then the partner SMC-R layer will send a CDC message to notify the peer (active close side) of the PeerDoneWriting indicator. When the "active close" SMC-R peer wakes up as a result of the previous CDC message, it will notice that the PeerDoneWriting indicator is now on and transition to the PeerCloseWait2 state. This state indicates that the peer is done sending data and may still be reading data. The "active close" peer will also at this point need to ensure that any outstanding recv() calls for this socket are woken up and remember that that no more data is forthcoming on this connection (in case the local connection was shutdown() for write only)

C. This flow is a common transition from 3a or 3b above. When the SMC-R peer (passive close) consumes all data, updates all necessary cursors to the peer and the application closes its socket (close or shutdown for both) it will send a CDC message to the peer (the active close side) with the PeerConnectionClosed indicator set. At this point the connection can transition back to Closed state if the local application has already closed (or issued shutdown for both) the socket. Once in the Closed state, the RMBE can now be safely be reused for a new SMC-R connection. When the PeerConnectionClosed indicator is turned on, the SMC-R peer is indicating that it is done updating the partner's RMBE.

D. Conditional State: If the local application has not yet issued a close() or shutdown(both) yet, we need to wait until the application does so (ApplFinWaitState). Once it does, the local host will send a CDC message to notify the peer of the PeerConnectionClosed indicator and then transition to the Closed state.

4. Passive close processing (on SMC-R peer that receives an indication that the partner is closing the connection)

A. Upon receipt of an inbound RDMA write notice the SMC-R layer will detect that the PeerConnectionClosed indicator or PeerDoneWriting indicator is on. If any outstanding recv() calls are pending they are completed with an indicator that the partner has closed the connection (zero length data presented to application). If any pending data to be written and PeerConnectionClosed is on then an SMC-R connection reset must be performed. The connection then enters the ApplCloseWait1 state on the passive close side waiting for the local application to initiate its own close processing

B. If the local application issues a shutdown() for writing then the SMC-R layer will send a CDC message to notify the partner of the PeerDoneWriting indicator transition the local side of the SMC-R connection to the ApplCloseWait2 state.

C. When the application issues a close() or shutdown() for both, the local SMC-R peer will send a message informing the peer of the PeerConnectionClosed indicator and transition to the Closed state if the remote peer has also sent the local peer the PeerConnectionClosed indicator. If the peer has not sent the PeerConnectionClosed indicator, we transition into the PeerFinalCloseWait state.

D. The local SMC-R connection stays in this state until the peer sends the PeerConnectionClosed indicator in our RMBE. When the indicator is sent we transition to the Closed state and are then free to reuse this RMBE.

Note that each SMC-R peer needs to provide some logic that will prevent being stranded in termination state indefinitely. For example, if an Active Close SMC-R peer is in a PeerCloseWait (1 or 2) state awaiting the remote SMC-R peer to update its connection termination status it needs to provide a timer that will prevent it from waiting in that state indefinitely should the remote SMC-R peer not respond to this termination request. This could occur in error scenarios; for example, if the remote SMC-R peer suffered a failure prior to being able to respond to the termination request or the remote application is not responding to this connection termination request by closing its own socket. This latter scenario is similar to the TCP FINWAIT2 state that has been known to sometimes cause issues when remote TCP/IP hosts lose track of established connections and neglect to close them. Even though the TCP standards do not mandate a time out from the TCP FINWAIT2 state, most TCP/IP implementations implement a timeout for this state. A similar timeout will be required for SMC-R connections. When this timeout occurs, the local SMC-R peer performs TCP reset processing for this connection. However, no additional RDMA writes to the partner RMBE can occur at this point (we have already indicated that we are done updating the peer's RMBE). After the TCP connection is Reset the RMBE can be returned to the free pool for reallocation. See section 3.2.5 for more details.

Also note that it is possible to have two SMC-R end points initiate an Active close concurrently. In that scenario the flows above still apply, however, both end points follow the active close path (path 3).

4.8.1.1. Abnormal SMC-R connection termination flows

Abnormal SMC-R connection termination can occur for a variety of reasons, including:

- o The TCP connection associated with an SMC-R connection is reset. In the TCP protocol either end point can send a RST segment to abort an existing TCP connection when error conditions are detected for the connection or the application overtly requests that the connection be reset.
- o Normal SMC-R connection termination processing has unexpectedly stalled for a given connection. When the stall is detected (connection termination timeout condition) an abnormal SMC-R connection termination flow is initiated.

In these scenarios it is very important that resources associated with the affected SMC-R connections are properly cleaned up to ensure that there are no orphaned resources and that resources can reliably be reused for new SMC-R connections. Given that SMC-R relies heavily on the RDMA Write processing, special care needs to be taken to ensure that an RMBE is no longer being used by a SMC-R peer before logically reassigning that RMBE to a new SMC-R connection.

When an SMC-R peer initiates a TCP connection reset it also initiates an SMC-R abnormal connection flow at the same time. The SMC-R peers explicitly signal their intent to abnormally terminate an SMC-R connection and await explicit acknowledgement that the peer has received this notification and has also completed abnormal connection termination on its end. Note that TCP connection reset processing can occur in parallel to these flows.

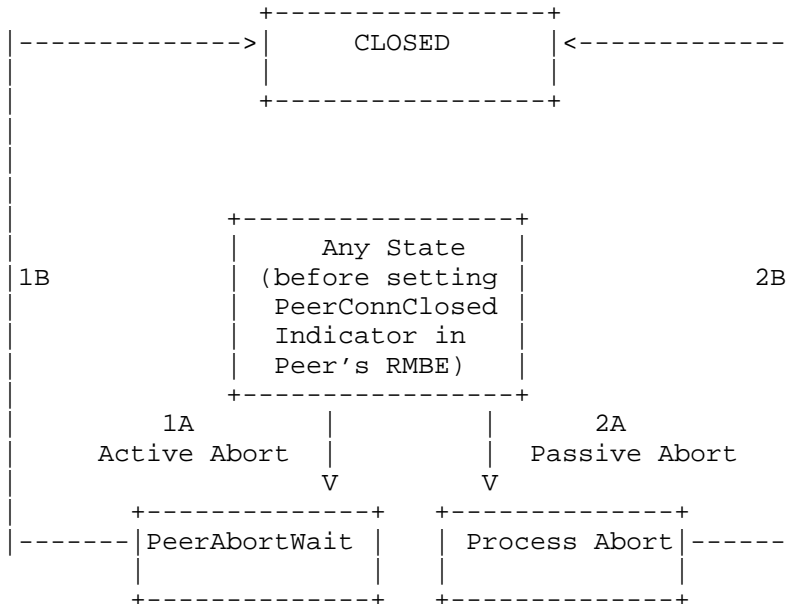


Figure 23 SMC-R abnormal connection termination state diagram

Figure 24 above shows the SMC-R abnormal connection termination state diagram:

1. Active abort designates the SMC-R peer that is initiating the TCP RST processing. At the time that the TCP RST is sent the active abort side must also
 - A. Send the PeerConnAbort indicator to the partner via RDMA messaging with inline data and then transition to the PeerAbortWait state. During this state it will monitor this SMC-R connection waiting for the peer to send its corresponding PeerConnAbort indicator but will ignore any other activity in this connection (i.e. new incoming data). It will also surface an appropriate error to any socket API calls issued against this socket (e.g. ECONNABORTED, ECONNRESET, etc.)
 - B. Once the peer sends the PeerConnAbort indicator to the local host, the local host can transition this SMC-R connection to the Closed state and reuse this RMBE. Note that the SMC-R peer that goes into the Active abort state must provide some protection

against staying in that state indefinitely should the remote SMC-R peer not respond by sending its own PeerConnAbort indicator to the local host. While this should be a rare scenario it could occur if the remote SMC-R peer (passive abort) suffered a failure right after the local SMC-R peer (active abort) sent the PeerConnAbort indicator. To protect against these types of failures, a timer can be set after entering the PeerAbortWait state and when if that timer pops before the peer has sent its local PeerConnAbort indicator (to the active abort side) then this RMBE can be returned to the free pool for possible re-allocation. See section 3.2.5 for more details.

2. Passive abort designates the SMC-R peer that is the recipient of an SMC-R abort from the peer designated by the PeerConnAbort indicator being sent by the peer in a CDC message. Upon receiving this request, the local peer must

- A. Indicate to the socket application that this connection has been aborted using the appropriate error codes, purge all in-flight data for this connection that is waiting to be read or waiting to be sent.

- B. Send a CDC message to notify the peer of the PeerConnAbort indicator and once that is completed transition this RMBE to the Closed state.

If an SMC-R peer receives a TCP RST for a given SMC-R connection it also initiates SMC-R abnormal connection termination processing if it has not already been notified (via the PeerConnAbort indicator) that the partner is severing the connection. It is possible to have two SMC-R endpoints concurrently be in an Active abort role for a given connection. In that scenario the flows above still apply but both end points take the active abort path (path 1).

4.8.1.2. Other SMC-R connection termination conditions

The following are additional conditions that have implications of SMC-R connection termination:

- o A SMC-R peer being gracefully shut down. If an SMC-R peer supports a graceful shutdown operation it should attempt to terminate all SMC-R connections as part of shutdown processing. This could be accomplished via LLC Delete Link requests on all active SMC Links.

- o Abnormal termination of an SMC-R peer. In this example, there may be no opportunity for the host to perform any SMC-R cleanup processing. In this scenario it is up to the remote peer to detect a RoCE communications failure with the failing host. This could trigger an SMC link switch but that would also surface RoCE errors causing the remote host to eventually terminate all existing SMC-R connections to this peer.
- o Loss of RoCE connectivity between two SMC-R peers. If two peers are no longer reachable across any links in their SMC Link group then both peers perform a TCP reset for the connections, surface an error to the local applications and free up all QP resources associated with the link group.

5. Security considerations

5.1. VLAN considerations

The concepts and access control of virtual LANs (VLANs) must be extended to also cover the RoCE network traffic flowing across the ethernet.

The RoCE VLAN configuration and accesses must mirror the IP VLAN configuration and accesses over the CEE fabric. This means that hosts, routers and switches that have access to specific VLANs on the IP fabric must also have the same VLAN access across the RoCE fabric. In other words, the SMC-R connectivity will follow the same virtual network access permissions as normal TCP/IP traffic.

5.2. Firewall considerations

As mentioned above, the RoCE fabric inherits the same VLAN topology/access as the IP fabric. RoCE is a layer 2 protocol that requires both end points to reside in the same layer 2 network (i.e. VLAN). RoCE traffic can not traverse multiple VLANs as there is no support for routing RoCE traffic beyond a single VLAN. As a result, SMC-R communications will also be confined to peers that are members of the same VLAN. IP based firewalls are typically inserted between VLANs (or physical lans) and rely on normal IP routing to insert themselves in the data path. Since RoCE (and by extension SMC-R) is not routable beyond the local VLAN, there is no ability to insert a firewall in the network path of two SMC-R peers.

5.3. Host-based IP Filters

Because SMC-R maintains the TCP three-way handshake for connection setup before switching to RoCE out of band, existing IP filters that control connection setup flows remain effective in an SMC-R environment. IP filters that operate on traffic flowing in an active TCP connection are not supported, because the connection data does not flow over IP.

5.4. Intrusion Detection Services

Similar to IP filters, intrusion detection services that operate on TCP connection setups are compatible with SMC-R with no changes required. However once the TCP connection has switched to RoCE out of band, packets are not available for examination.

5.5. IP Security (IPSec)

IP Security is not compatible with SMC-R because there are no IP packets to operate on. TCP connections that require IP security must opt out of SMC-R.

5.6. TLS/SSL

TLS/SSL is preserved in an SMC-R environment. The TLS/SSL layer resides above the SMC-R layer and outgoing connection data is encrypted before being passed down to the SMC-R layer for RDMA write. Similarly, incoming connection data goes through the SMC-R layer encrypted and is decrypted by the TLS/SSL layer as it is today.

The TLS/SSL handshake messages flow over the TCP connection after the connection has switched to SMC-R, so are exchanged using RDMA writes by the SMC-R layer, transparently to the TLS/SSL layer.

6. IANA considerations

The scarcity of TCP option codes available for assignment is understood and this architecture uses experimental TCP options following the conventions of RFC 6994 "Shared Use of Experimental TCP Options".

If this protocol achieves wide acceptance a discrete option code may be requested by subsequent versions of this protocol.

7. References

7.1. Normative References

- [ROCE] RDMA over Converged Ethernet specification, URL,
http://members.infinibandta.org/kwspub/spec/Annex_RoCE_final.pdf
- [IBTA] Infiniband Architecture specification, URL,
<http://www.infinibandta.org/specs>
- [RFC793] University of Southern California Information Services
Institute, "Transmission Control Protocol", RFC 793,
September 1981.
- [RFC4727] Fenner B., "Experimental Values in IPv4, IPv6, ICMPv4,
ICMPv6, UDP, and TCP Headers", RFC 4727, November 2006.

7.2. Informative References

- [RFC 6994] Touch, J., "Shared use of Experimental TCP Options",
draft URL, <https://tools.ietf.org/html/rfc6994>

8. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

9. Conventions used in this document

In the rendezvous flow diagrams, dashed lines (----) are used to indicate flows over the TCP/IP fabric and dotted lines (....) are used to indicate flows over the RoCE fabric.

In the data transfer ladder diagrams, dashed lines (----) are used to indicate RDMA write operations and dotted lines (....) are used to indicate CDC messages, which are RDMA messages with inline data that contain control information for the connection.

Appendix A. Formats

A.1. TCP option

The SMC-R TCP option is formatted in accordance with RFC 6994 "Shared Use of Experimental TCP Options". The ExID value is IBM-1047 (EBCDIC) encoding for 'SMCR'

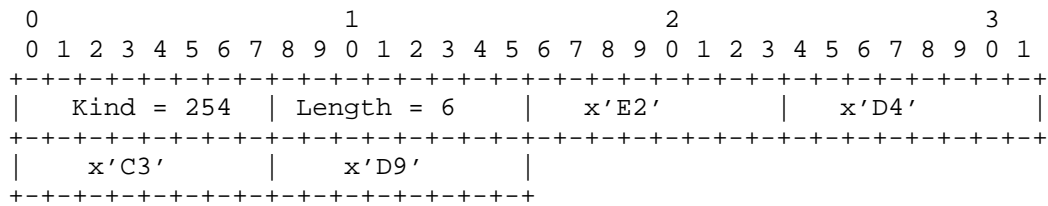


Figure 24 SMC-R TCP option format

A.2. CLC messages

The following rules apply to all CLC messages:

General rules on formats:

- o Reserved fields must be set to zero and not validated
- o Each message has an eyecatcher at the start and another eyecatcher at the end. These must both be validated by the receiver.
- o SMC version indicator: The only SMC-R version defined in this architecture is version 1. In the future, if peers have a mismatch of versions, the lowest common version number is used.

A.2.1. Peer ID format

All CLC messages contain a peer ID that uniquely identifies an instance of a TCP/IP stack. This peer ID is required to be universally unique across TCP/IP stacks and instances (including restarts) of TCP/IP stacks.

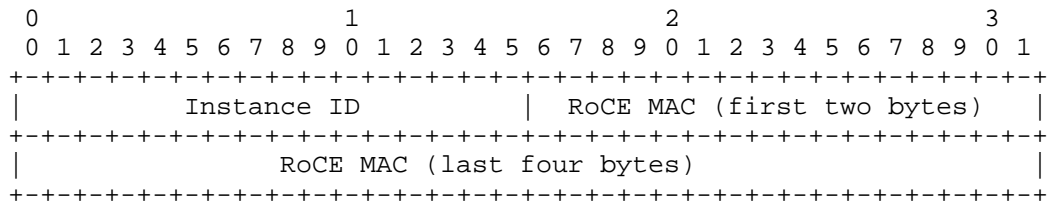


Figure 25 Peer ID format

Instance ID

A two-byte instance count that ensures that if the same RNIC MAC is later used in the peer ID for a different TCP/IP stack, for example if an RNIC is redeployed to another stack, the values are unique. It also ensures that if a TCP/IP stack is restarted, the instance ID changes. Value is implementation defined, with one suggestion being two bytes of the system clock.

RoCE MAC

The RoCE MAC address for one of the peer's RNICs. Note that in a virtualized environment this will be the virtual MAC of one of the peer's RNICs.

A.2.2. SMC Proposal CLC message format

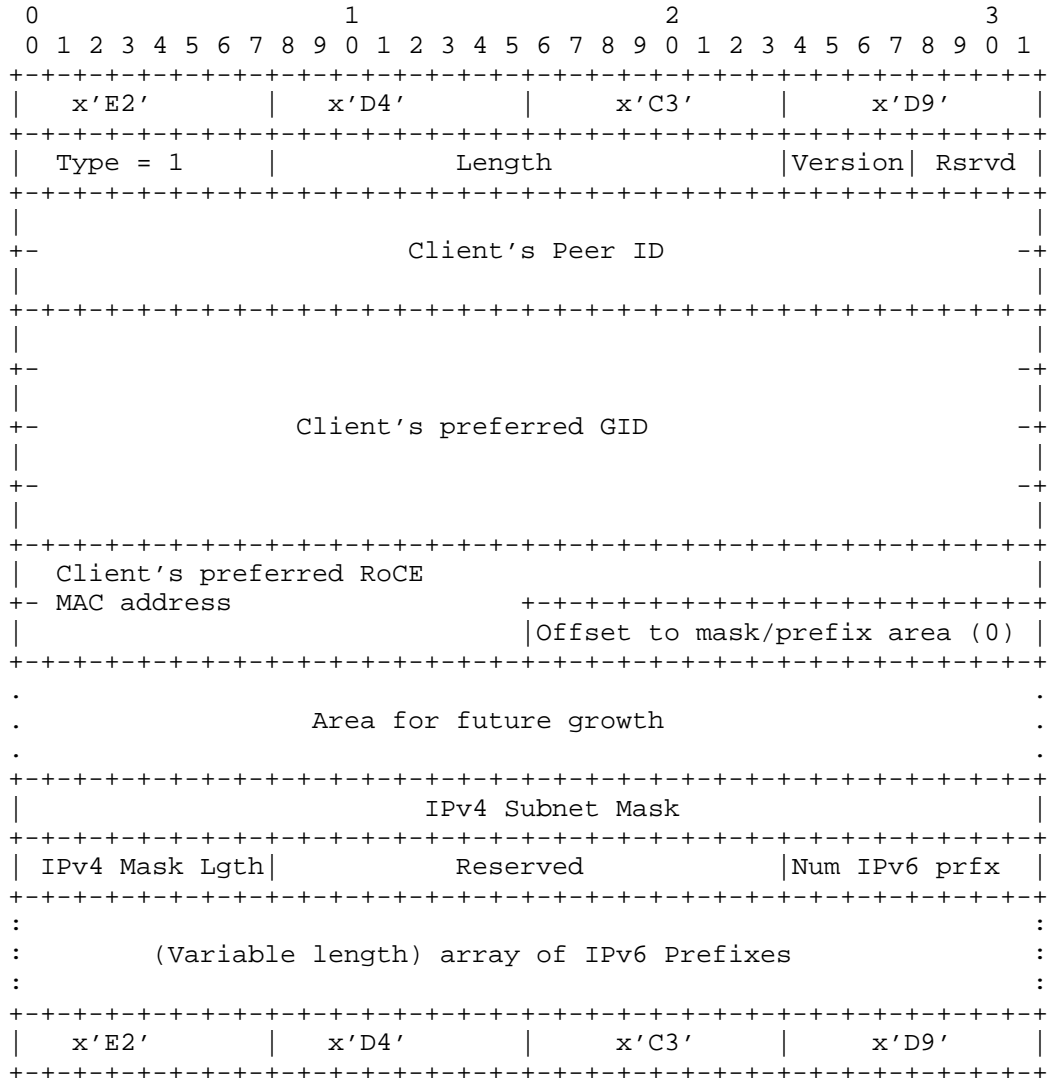


Figure 26 SMC Proposal CLC message format

The fields present in the SMC Proposal CLC message are:

Eyecatchers

Like all CLC messages, the SMC Proposal has beginning and ending eyecatchers to aid with verification and parsing. The hex digits spell 'SMCR' in IBM-1047 (EBCDIC)

Type

CLC message type 1 indicates SMC Proposal

Length

The length of this CLC message. If this an IPv4 flow, this value is 52. Otherwise it is variable depending upon how many prefixes are listed.

Version

Version of the SMC-R protocol. Version 1 is the only currently defined value

Client's Peer ID

As described in A.2.1. above

Client's preferred RoCE GID

This is the IPv6 address of the client's preferred RNIC on the RoCE fabric

Client's preferred RoCE MAC address

The MAC address of the client's preferred RNIC on the RoCE fabric. It is required as some operating systems do not have neighbor discovery or ARP support for RoCE RNICs.

Offset to mask/prefix area

Provides the number of bytes that must be skipped after this field, to access the IPv4 Subnet Mask and the fields that follow it. Allows for future growth of this signal. In this version of the architecture, this value is always zero.

Area for future growth

In this version of the architecture, this field does not exist. This indicates where additional information may be inserted into the signal in the future. "The Offset to mask/prefix area" field must be used to skip over this area.

IPv4 Subnet mask

If this message is flowing over an IPv4 TCP connection, the value of the subnet mask associated with the interface the client sent this message over. If this an IPv6 flow this field is all zeroes.

This field, along with all fields that follow it in this signal, must be accessed by skipping the number of bytes listed in the "Offset to mask/prefix area" field after the end of that field.

IPv4 Mask Lgth

If this message is flowing over an IPv4 TCP connection, the number of significant bits in the IPv4 subnet mask. If this an IPv6 flow, this field is zero.

Num IPv6 prfx

If this message is flowing over an IPv6 TCP connection, the number of IPv6 prefixes that follow, with a maximum value of 8. if this is an IPv4 flow this field is zero and is immediately followed by the ending eyecatcher.

Array of IPv6 Prefixes

For IPv6 TCP connections, a list of the IPv6 prefixes associated with the network the client sent this message over, up to a maximum of 8 prefixes.

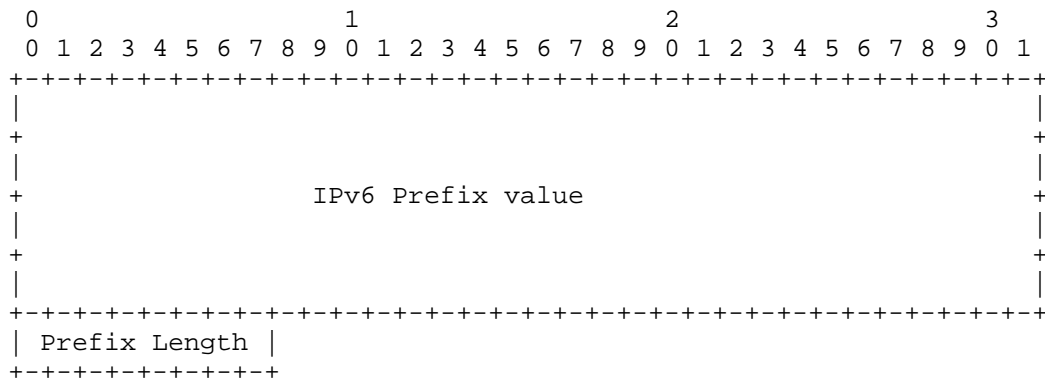


Figure 27 Format for IPv6 Prefix array element

A.2.3. SMC Accept CLC message format

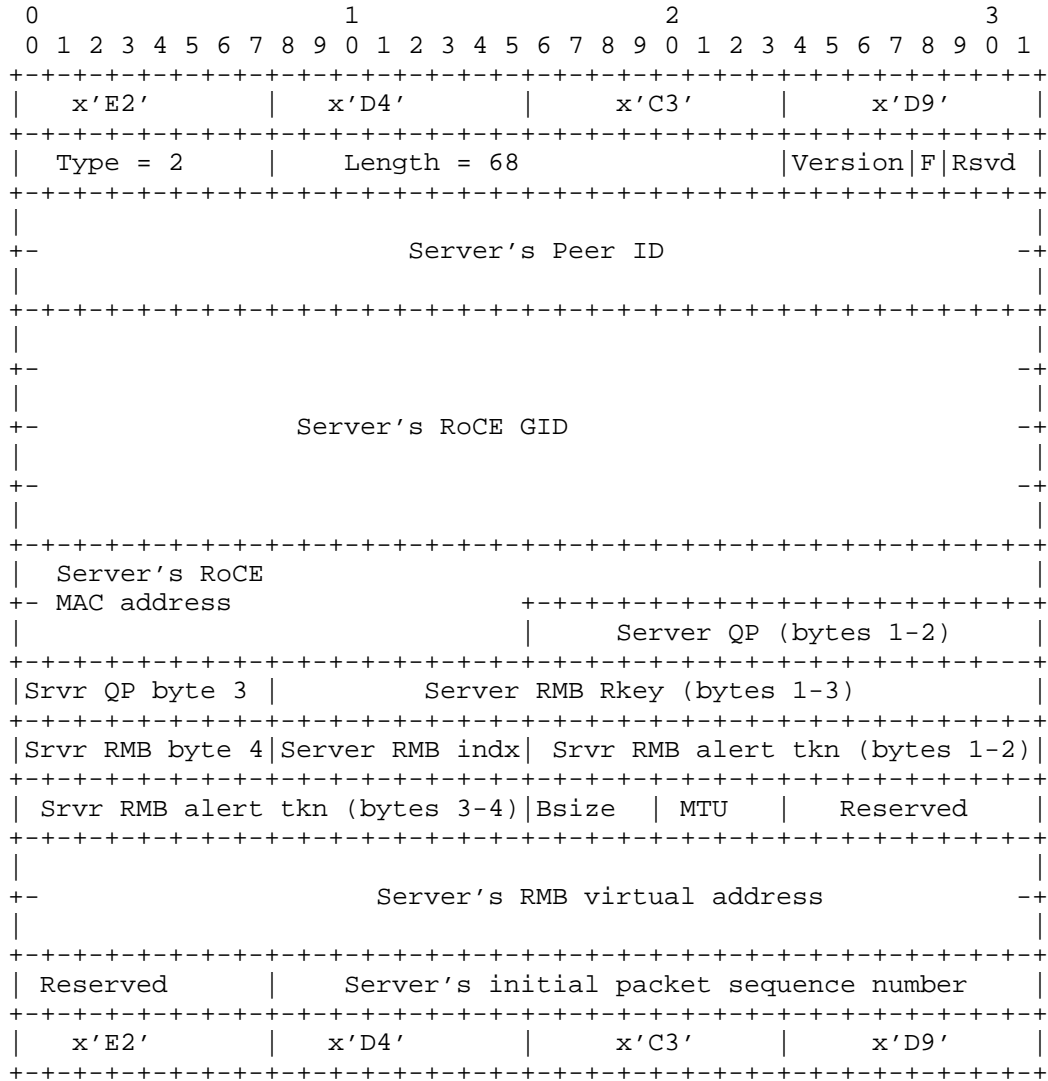


Figure 28 SMC Accept CLC message format

The fields present on the SMC Accept CLC message are:

Eyecatchers

Like all CLC messages, the SMC Accept has beginning and ending eyecatchers to aid with verification and parsing. The hex digits spell 'SMCR' in IBM-1047 (EBCDIC)

Type

CLC message type 2 indicates SMC Accept

Length

The SMC Accept CLC message is 68 bytes long

Version

Version of the SMC-R protocol. Version 1 is the only currently defined value.

F-bit

First Contact flag: A 1-bit flag that indicates that the server believes this TCP connection is the first SMC-R contact for this link group

Server's Peer ID

As described in A.2.1. above

Server's RoCE GID

This is the IPv6 address of the RNIC that the server chose for this SMC Link

Server's RoCE MAC address

The MAC address of the server's RNIC for the SMC link. It is required as some operating systems do not have neighbor discovery or ARP support for RoCE RNICs.

Server's QP number

The number for the reliably connected queue pair that the server created for this SMC link

Server's RMB Rkey

The RDMA Rkey for the RMB that the server created or chose for this TCP connection

Server's RMB element index

This indexes which element within the server's RMB will represent this TCP connection

Server's RMB element alert token

A platform defined, architecturally opaque token that identifies this TCP connection. Added by the client as immediate data on RDMA writes from the client to the server to inform the server that there is data for this connection to retrieve from the RMB element

Bsize:

Server's RMB element buffer size in four bits compressed notation: $x=4$ bits. Actual buffer size value is $(2^{(x+4)}) * 1K$. Smallest possible value is 16K. Largest size supported by this architecture is 512K.

MTU

An enumerated value indicating this peer's QP MTU size. The two peers exchange this value and the minimum of the peer's value will be used for the QP. This field should only be validated on a first contact exchange.

The enumerated MTU values are:

- 0: reserved
- 1: 256
- 2: 512
- 3: 1024
- 4: 2048
- 5: 4096
- 6-15: reserved

Server's RMB virtual address

The virtual address of the server's RMB as assigned by the server's RNIC.

Server's initial packet sequence number

The starting packet sequence number that this peer will use when sending to the other peer, so that the other peer can prepare its QP for the sequence number to expect.

A.2.4. SMC Confirm CLC message format

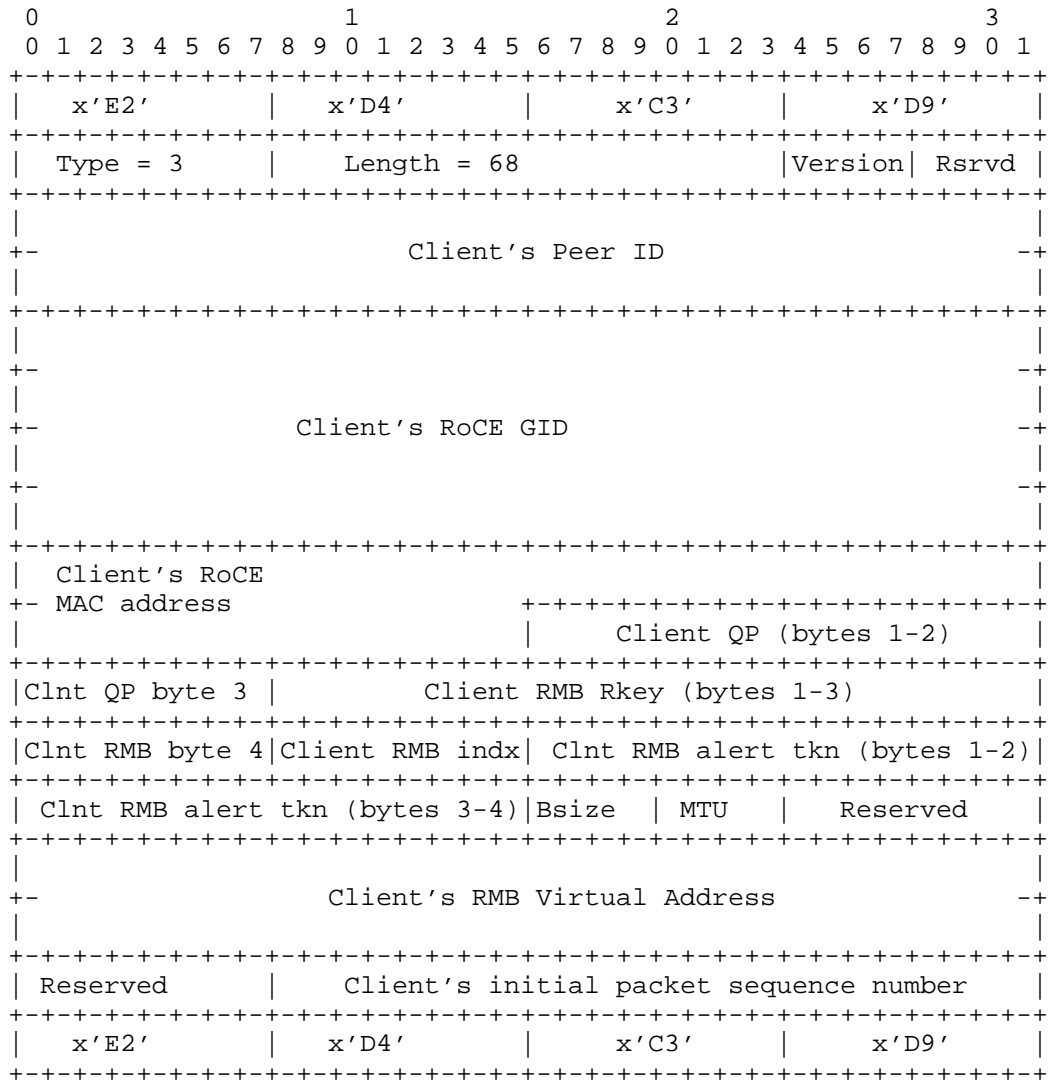


Figure 29 SMC Confirm CLC message format

The SMC Confirm CLC message is nearly identical to the SMC Accept except that it contains client information and lacks a first contact flag.

The fields present on the SMC Confirm CLC message are:

Eyecatchers

Like all CLC messages, the SMC Confirm has beginning and ending eyecatchers to aid with verification and parsing. The hex digits spell 'SMCR' in IBM-1047 (EBCDIC)

Type

CLC message type 3 indicates SMC Confirm

Length

The SMC Confirm CLC message is 68 bytes long

Version

Version of the SMC-R protocol. Version 1 is the only currently defined value.

Client's Peer ID

As described in A.2.1. above

Clients's RoCE GID

This is the IPv6 address of the RNIC that the client chose for this SMC Link

Client's RoCE MAC address

The MAC address of the client's RNIC for the SMC link. It is required as some operating systems do not have neighbor discovery or ARP support for RoCE RNICs.

Client's QP number

The number for the reliably connected queue pair that the client created for this SMC link

Client's RMB Rkey

The RDMA Rkey for the RMB that the client created or chose for this TCP connection

Client's RMB element index

This indexes which element within the client's RMB will represent this TCP connection

Client's RMB element alert token

A platform defined, architecturally opaque token that identifies this TCP connection. Added by the server as immediate data on RDMA writes from the server to the client to inform the client that there is data for this connection to retrieve from the RMB element

Bsize:

Client's RMB element buffer size in four bits compressed notation: $x=4$ bits. Actual buffer size value is $(2^{(x+4)}) * 1K$. Smallest possible value is 16K. Largest size supported by this architecture is 512K.

MTU

An enumerated value indicating this peer's QP MTU size. The two peers exchange this value and the minimum of the peer's value will be used for the QP. The values are enumerated in A.2.3. This value should only be validated on the first contact exchange.

Client's RMB virtual address

The virtual address of the server's RMB as assigned by the server's RNIC.

Client's initial packet sequence number

The starting packet sequence number that this peer will use when sending to the other peer, so that the other peer can prepare its QP for the sequence number to expect

.

A.2.5. SMC Decline CLC message format

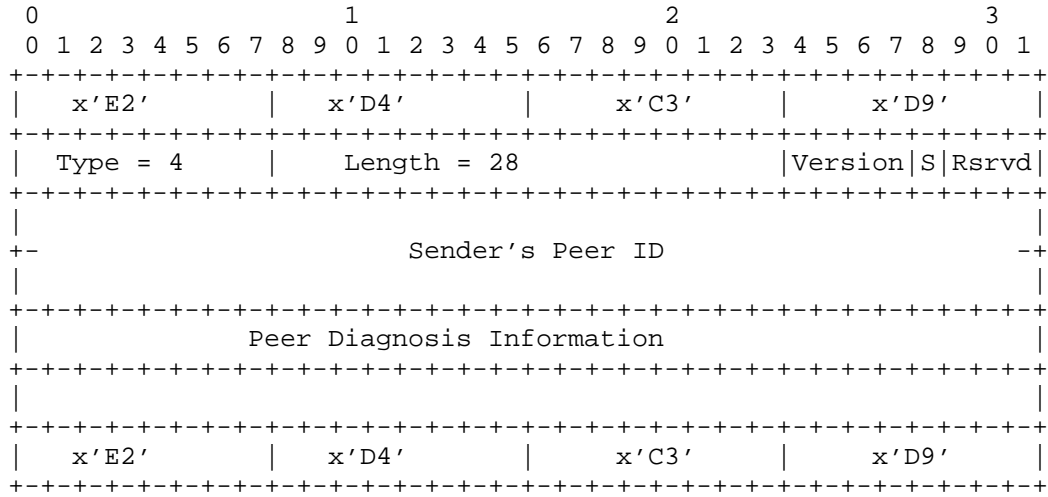


Figure 30 SMC Decline CLC message format

The fields present on the SMC Decline CLC message are:

Eyecatchers

Like all CLC messages, the SMC Decline has beginning and ending eyecatchers to aid with verification and parsing. The hex digits spell 'SMCR' in IBM-1047 (EBCDIC)

Type

CLC message type 4 indicates SMC Decline

Length

The SMC Decline CLC message is 28 bytes long

Version

Version of the SMC-R protocol. Version 1 is the only currently defined value.

S-bit

Synch Bit. Indicates that the link group is out of synch and receiving peer must clean up its representation of the link group

Sender's Peer ID

As described in A.2.1. above

Peer Diagnosis Information

Four bytes of diagnosis information provided by the peer. These values are defined by the individual peers and it is necessary to consult the peer's system documentation to interpret the results.

A.3. LLC messages

LLC messages are sent over an existing SMC-R link using RoCE message passing and are always 44 bytes long so that they fit into the space available in a single WQE without requiring the receiver to post receive buffers. If all 44 bytes are not needed, they are padded out with zeroes. LLC messages are in a request/response format. The message type is the same for request and response, and a flag indicates whether a message is flowing as a request or a response.

The two high order bits of an LLC message opcode indicate how it is to be handled by a peer that does not support the opcode.

If the high order bits of the opcode are b'00' then the peer must support the LLC message and indicate a protocol error if it does not.

If the high order bits of the opcode are b'10' then the peer must silently discard the LLC message if does not support the opcode. This requirement is inserted to allow for toleration of advanced, but optional function.

High order bits of b'11' indicate a Connection Data Control (CDC) message as described in A.4.

A.3.1. CONFIRM LINK LLC message format

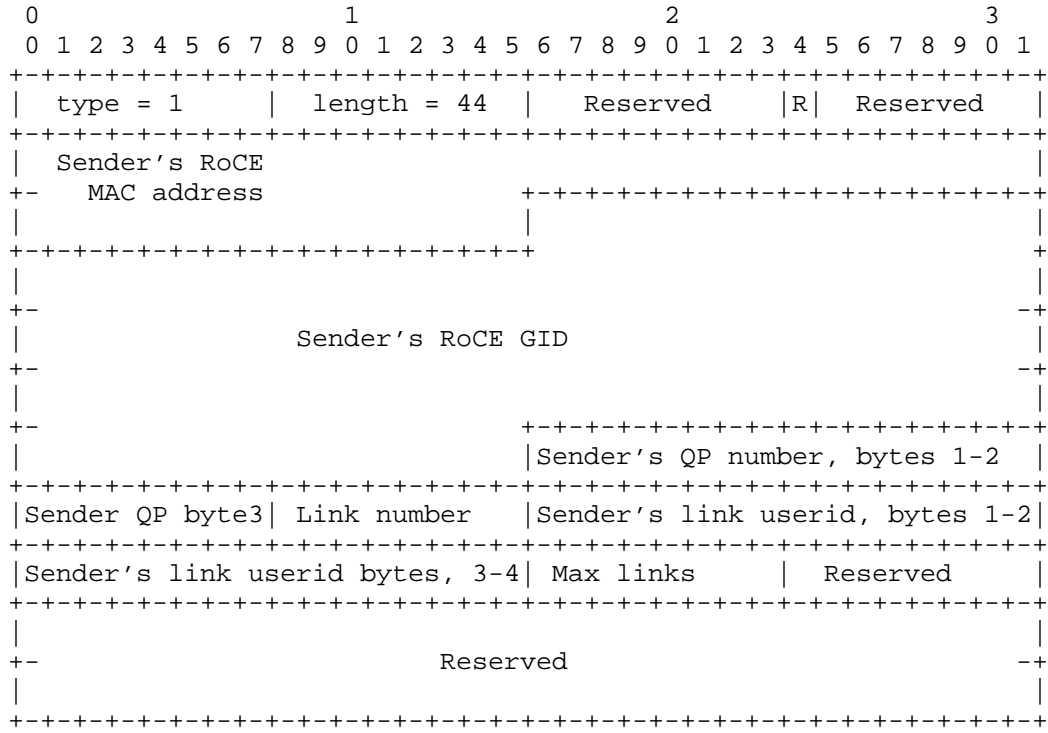


Figure 31 CONFIRM LINK LLC message format

The CONFIRM LINK LLC message is required to be exchanged between the server and client over a newly created SMC-R link to complete the setup of an SMC link. Its purpose is to confirm that the RoCE path is actually usable.

On first contact this flows after the server receives the SMC Confirm CLC message from the client over the IP connection. For additional links added to an SMC link group, it flows after the ADD LINK and ADD LINK CONTINUATION exchange. This flow provides confirmation that the queue pair is in fact usable. Each peer echoes its RoCE information back to the other.

Type

Type 1 indicates CONFIRM LINK

Length

All LLC messages are 44 bytes long

R

Reply flag. When set indicates this is a CONFIRM LINK REPLY

Sender's RoCE MAC address

The MAC address of the sender's RNIC for the SMC link. It is required as some operating systems do not have neighbor discovery or ARP support for RoCE RNICs.

Sender's RoCE GID

This is the IPv6 address of the RNIC that the sender is using for this SMC-R Link

Sender's QP number

The number for the reliably connected queue pair that the sender created for this SMC-R link

Link number

An identifier assigned by the server that uniquely identifies the link within the link group. This identifier is ONLY unique within a link group. Provided by the server and echoed back by the client

Link User ID

An opaque, implementation defined identifier assigned by the sender and provided to the receiver solely for purposes of display, diagnosis, network management, etc. The link user ID should be unique across the sender's entire software space, including all link other link groups.

Max Links

The maximum number of links the sender can support in a link group. The maximum for this link group is the the smaller of the values provided by the two peers.

A.3.2. ADD LINK LLC message format

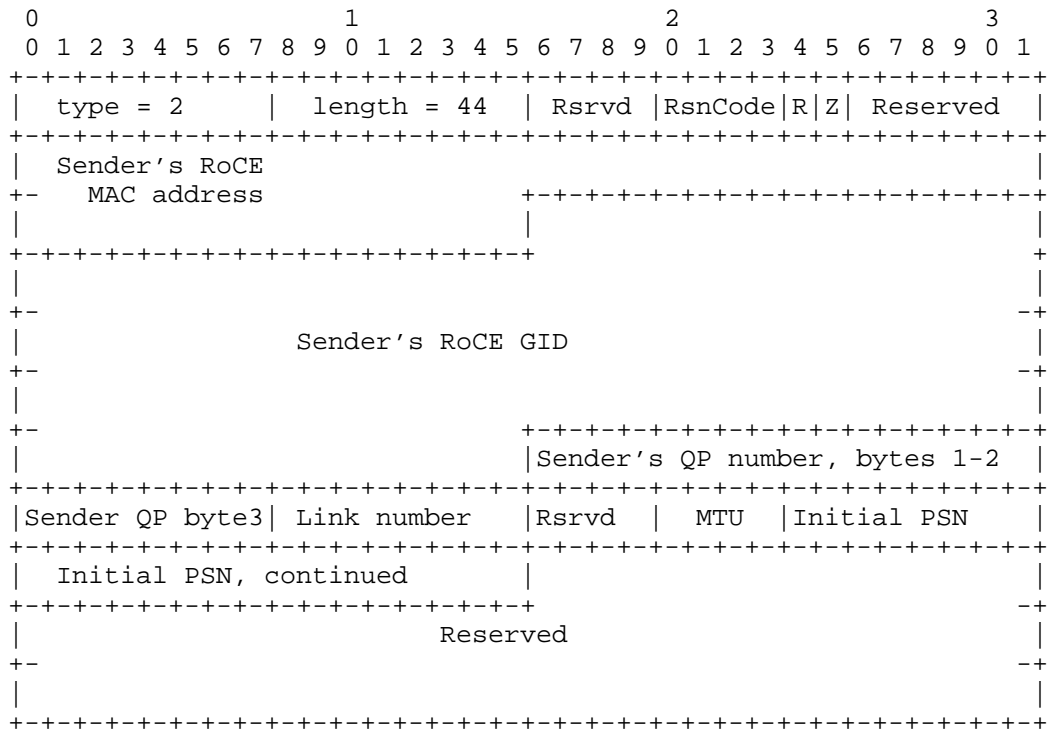


Figure 32 ADD LINK LLC message format

The ADD LINK LLC message is sent over an existing link in the link group when a peer wishes to add an SMC-R link to an existing SMC-R link group. It sent by the server to add a new SMC-R link to the group, or by the client to request that the server add a new link, for example when a new RNIC becomes active. When sent from the client to the server, it represents a request that the server initiate an ADD LINK exchange.

This message is sent immediately after the initial SMC link in the group completes, as described in 3.5.1. First contact. It can also be sent over an existing SMC-R link group at any time as new RNICs are added and become available. Therefore there can be as few as 1 new RMB RTokens to communicate, or several. Rtokens will be communicated using ADD LINK CONTINUATION messages.

The contents of the ADD LINK LLC message are:

Type

Type 2 indicates ADD LINK

Length

All LLC messages are 44 bytes long

RsnCode

If the Z (rejection) flag is set, this field provides the reason code. Values can be:

X'1' - no alternate path available: set when the server provides the same MAC/GID as an existing SMC-R link in the group, and the client does not have any additional RNICs available (i.e., server is attempting to set up an asymmetric link but none is available)

X'2' - Invalid MTU value specified

R

Reply flag. When set indicates this is an ADD LINK REPLY

Z

Rejection flag. When set on reply indicates that the server's ADD LINK was rejected by the client. When this flag is set, the reason code will also be set.

Sender's RoCE MAC address

The MAC address of the sender's RNIC for the new SMC-R link. It is required as some operating systems do not have neighbor discovery or ARP support for RoCE RNICs.

Sender's RoCE GID

The IPv6 address of the RNIC that the sender is using for the new SMC-R Link

Sender's QP number

The number for the reliably connected queue pair that the sender created for the new SMC-R link

Link number

An identifier for the new SMC-R link. This is assigned by the server and uniquely identifies the link within the link group. This identifier is ONLY unique within a link group. Provided by the server and echoed back by the client

MTU

An enumerated value indicating this peer's QP MTU size. The two peers exchange this value and the minimum of the peer's value will be used for the QP. The values are enumerated in A.2.3.

Initial PSN

The starting packet sequence number that this peer will use when sending to the other peer, so that the other peer can prepare its QP for the sequence number to expect.

A.3.3. ADD LINK CONTINUATION LLC message format

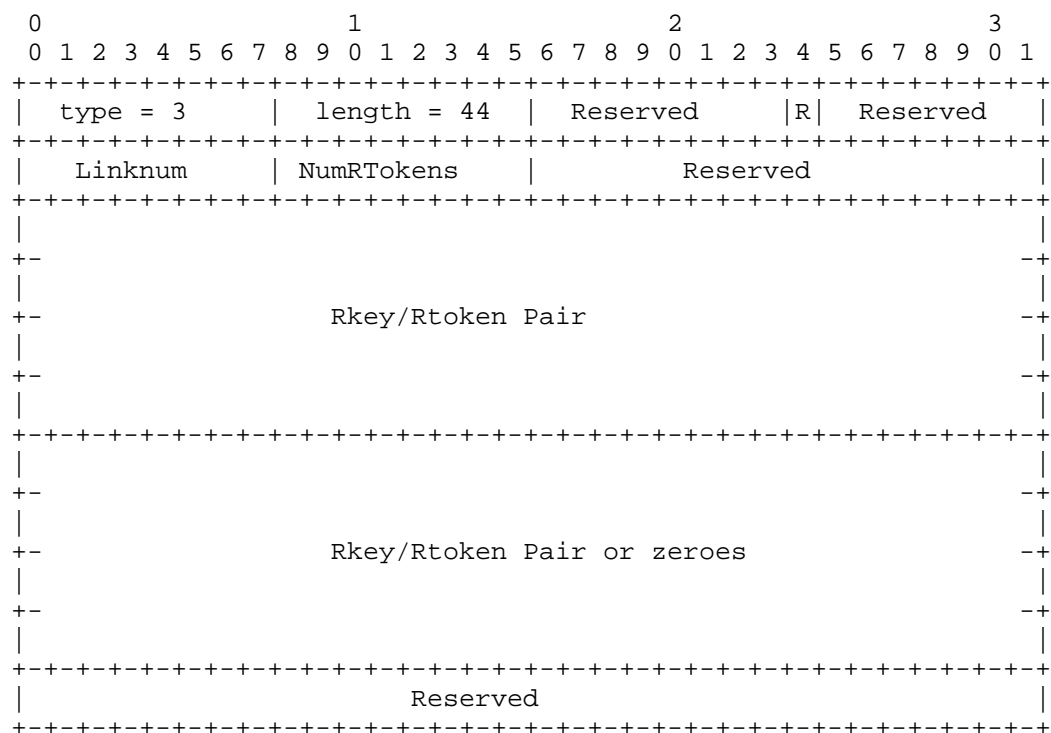


Figure 33 ADD LINK CONTINUATION LLC message format

When a new SMC-R link is added to an SMC-R link group, it is necessary to communicate the new link's RTokens for the RMBs that the SMC-r link group can access. This message follows the ADD LINK and provides the RTokens.

The server kicks off this exchange by sending the first ADD LINK CONTINUATION LLC message, and the server controls the exchange as described below.

- o If the client and the server require the same number of ADD LINK CONTINUATION messages to communicate their RTokens, the server starts the exchange by sending the client the first ADD LINK CONTINUATION request to the client with its RTokens, then the client responds with an ADD LINK CONTINUATION response with its RTokens, and so on until the exchange is completed.
- o If the server requires more ADD LINK CONTINUATION messages than the client, then after the client has communicated all its RTokens, the server continues to send ADD LINK CONTINUATION request messages to the client. The client continues to respond, using empty (number of RTokens to be communicated = 0) ADD LINK CONTINUATION response messages.
- o If the client requires more ADD LINK CONTINUATION messages than the server, then after communicating all its RTokens the server will continue to send empty ADD LINK CONTINUATION messages to the client to solicit replies with the client's RTokens, until all have been communicated.

The contents of this message are:

Type

Type 3 indicates ADD LINK CONTINUATION

Length

All LLC messages are 44 bytes long

R

Reply flag. When set indicates this is an ADD LINK CONTINUATION
REPLY

LinkNum

The link number of the new link within the SMC link group that Rkeys are being communicated for

NumRTokens

Number of RTokens remaining to be communicated (including the ones in this message). If the value is less than or equal to 2, this is the last message. If it is greater than 2, another continuation message will be required, and its value will be the value in this message minus 2, and so on until all Rkeys are communicated. The maximum value for this field is 255.

Up to 2 Rkey/RToken pairs

These consist of an Rkey for an RMB that is known on the SMC-R link that this message was sent over (the reference Rkey), paired with the same RMB's RToken over the new SMC link. A full RToken is not required for the reference because it is only being used to distinguish which RMB it applies to, not address it.

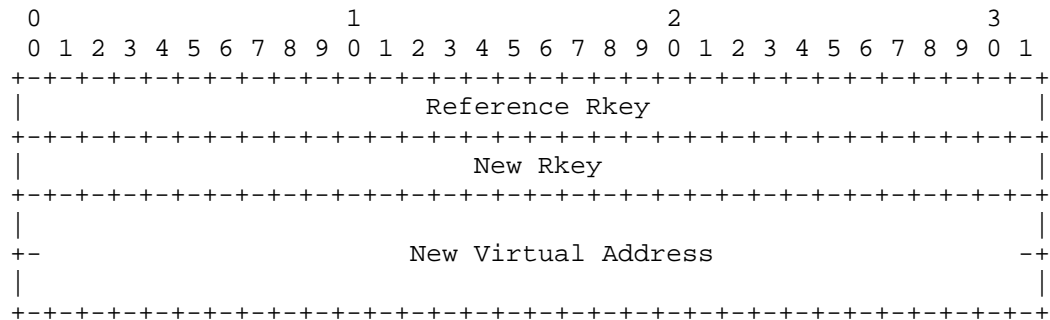


Figure 34 Rkey/Rtoken pair format

The contents of the RKey/RToken pair are:

Reference Rkey

The Rkey of the RMB as it is already known on the SMC-R link over which this message is being sent. Required so that the peer knows which RMB to associate the new Rtoken with.

New Rkey

The Rkey of this RMB as it is known over the new SMC-R link

New Virtual Address

The virtual address of this RMB as it is known over the new SMC-R link.

A.3.4. DELETE LINK LLC message format

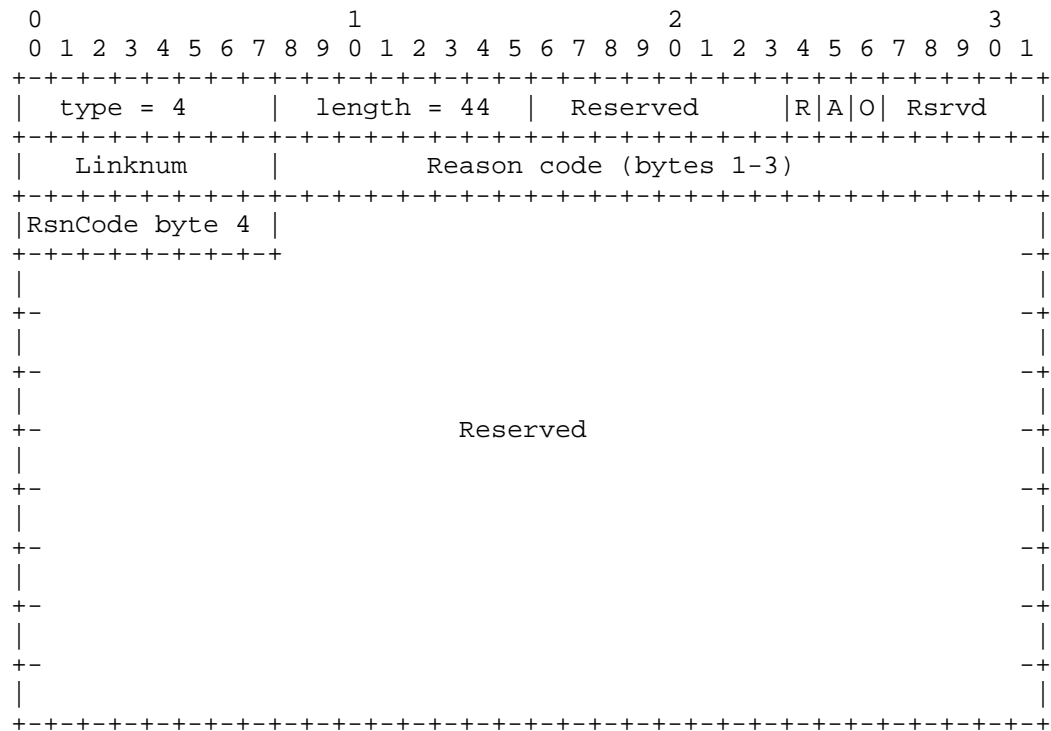


Figure 35 DELETE LINK LLC message format

When the client or server detects that a QP or SMC-R link goes down or needs to come down, it sends this message over one of the other links in the link group.

When the DELETE Link is sent from the client it only serves as a notification, and the client expects the server to send a DELETE LINK Request in response. To avoid races, only the server will initiate the actual DELETE LINK Request and Response sequence that results from notification from the client.

The server can also initiate the DELETE Link without notification from the client if it detects an error or if orderly link termination was initiated.

The client may also request termination of the entire link group and the server may terminate the entire link group using this message.

The contents of this message are:

Type

Type 4 indicates DELETE LINK

Length

All LLC messages are 44 bytes long

R

Reply flag. When set indicates this is an DELETE LINK REPLY

A

All flag. When set indicates that all links in the link group are to be terminated. This terminates the link group.

O

Orderly flag. Indicates orderly termination. Orderly termination is generally caused by an operator command rather than an error on the link. When the client requests orderly termination, the server may wait to complete other work before terminating.

LinkNum

The link number of the link to be terminated. If the A flag is set, this field has no meaning and is set to 0.

RsnCode

The termination reason code. Currently defined reason codes are:

Request Reason Codes:

- o X'00010000' = lost path
- o X'00020000' = operator initiated termination

- o X'00030000' = Program initiated termination (link inactivity)
- o X'00040000' = LLC protocol violation
- o X'00050000' = Asymmetric link no longer needed

Response Reason Codes:

- o X'00100000' = Unknown Link ID (no link)
- o Others TBD

A.3.5. CONFIRM RKEY LLC message format

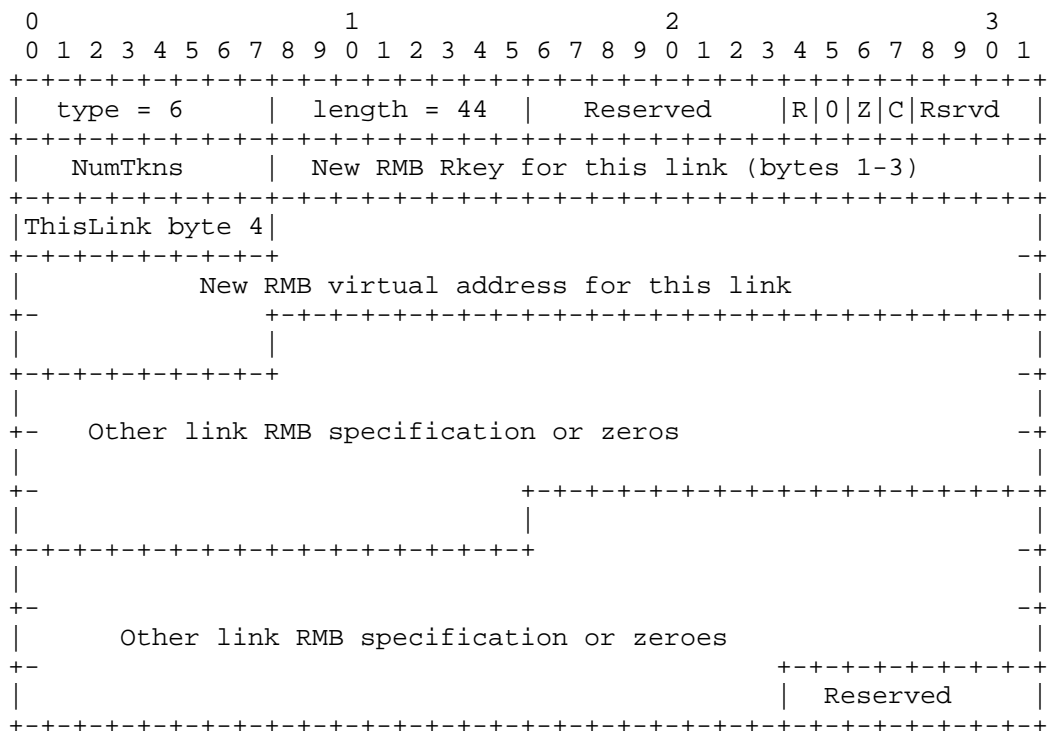


Figure 36 CONFIRM RKEY LLC message format

The CONFIRM_RKEY flow can be sent at any time from either the client or the server, to inform the peer that an RMB has been created or deleted. The creator of a new RMB must inform its peer of the new RMB's RToken for all SMC-R links in the SMC-R link group. The deleter of an RMB must inform its peer of the deleted RMB's RToken for all SMC-R links.

For RMB creation, the creator sends this message over the SMC link that the first TCP connection that uses the new RMB is using. This message contains the new RMB RToken for the SMC link that the message is sent over, then it lists the sender's SMC links in the link group paired with the new RToken for the new RMB for that link. This message can communicate the new RTokens for 3 QPs: the QP for the link this message is sent over, and 2 others. If there are more than 3 links in the SMC-R link group, CONFIRM_RKEY_CONTINUATION will be required.

For RMB deletion, the creator sends the same format of message with a delete flag set, to inform the peer that the RMB's RTokens on all links in the group are deleted.

In both cases, the peer responds by simply echoing the message with the response flag set. If the response is a negative response, the sender must recalculate the RToken set and start a new CONFIRM_RKEY exchange from the beginning. The timing of this retry is controlled by the C flag as described below.

The contents of this message are:

Type

Type 6 indicates CONFIRM RKEY

Length

All LLC messages are 44 bytes long

R

Reply flag. When set indicates this is a CONFIRM RKEY REPLY

0

Reserved bit

Z

Negative response flag

C

Configuration Retry bit. If this is a negative response and this flag is set, the originator should recalculate the Rkey set and retry this exchange as soon as the current configuration change

is completed. If this flag is not set on a negative response, the originator must wait for the next natural stimulus (for example, a new TCP connection started that requires a new RMB) before retrying.

NumTkns

The number of other link/RToken pairs, including those provided in this message, to be communicated. Note that this value does not include the Rtoken for the link this message was sent on (i.e., the maximum value is 2). If this value is three or fewer this is the only message in the exchange. If this value is greater than three, a CONFIRM RKEY CONTINUATION message will be required.

Note: in this version of the architecture, 8 is the maximum number of links supported in a link group.

New RMB Rkey for this link

The new RMB's Rkey as assigned on the link this message is being sent over.

New RMB virtual address for this link

The new RMB's virtual address as assigned on the link this messages is being sent over.

Other link RMB specification

The new RMB's specification on the other links in the link group, as shown in Figure 38.

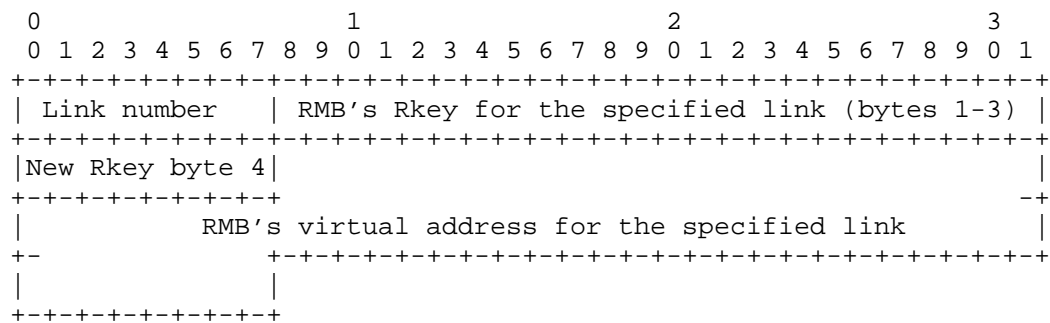


Figure 37 Format of link number/Rkey pairs

Link number

The link number for a link in the link group

RMB's Rkey for the specified link

The Rkey used to reach the RMB over the link whose number was specified in the link number field.

RMB's virtual address for the specified link

The virtual address used to reach the RMB over the link whose number was specified in the link number field.

A.3.6. CONFIRM RKEY CONTINUATION LLC message format

[illegible]

The CONFIRM RKEY CONTINUATION LLC message is used to communicate any additional RMB RTokens that did not fit into the CONFIRM RKEY message. Each of these messages can hold up to 3 RMB RTokens. The Numlinks field indicates how many RMB RTokens are to be communicated,

including the ones in this message. If the value is 3 or less, this is the last message of the group. If the value is 4 or higher, additional CONFIRM RKEY CONTINUATION messages will follow, and the Numlinks value will be a countdown until all are communicated.

Like the CONFIRM RKEY message, the peer responds by echoing the message back with the reply flag set.

The contents of this message are:

Type

Type 8 indicates CONFIRM RKEY CONTINUATION

Length

All LLC messages are 44 bytes long

R

Reply flag. When set indicates this is a CONFIRM RKEY CONTINUATION REPLY

O

Reserved bit

Z

Negative response flag

NumTknsLeft

The number of link/RToken pairs, including those provided in this message, that are remaining to be communicated. If this value is three or fewer this is the last message in the exchange. If this value is greater than three, another CONFIRM RKEY CONTINUATION message will be required. Note that in this version of the architecture, 8 is the maximum number of links supported in a link group.

Other link RMB specifications

The new RMB's specification on other links in the link group, as shown in Figure 38.

A.3.7. DELETE RKEY LLC message format

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
type = 9										length = 44										Reserved										R 0 Z Rsrvd									
Count										Error Mask										Reserved																			
First deleted Rkey																																							
Second deleted Rkey or zeros																																							
Third deleted Rkey or zeros																																							
Fourth deleted Rkey or zeros																																							
Fifth deleted Rkey or zeros																																							
Sixth deleted Rkey or zeros																																							
Seventh deleted Rkey or zeros																																							
Eighth deleted Rkey or zeros																																							
Reserved																																							

The DELETE_RKEY flow can be sent at any time from either the client or the server, to inform the peer that one or more RMBs have been deleted. Because the peer already knows every RMB's Rkey on each link in the link group, this message only specifies one Rkey for each RMB being deleted. The Rkey provided for each deleted RMB will be its Rkey as known on the SMC-R link that this message is sent over.

It is not necessary to provide the entire RToken. The Rkey alone is sufficient for identifying an existing RMB.

The peer responds by simply echoing the message with the response flag set. If the peer did not recognize an Rkey, a negative response flag will be set, however no aggressive recovery action beyond logging the error will be taken.

The contents of this message are:

Type

Type 9 indicates DELETE RKEY

Length

All LLC messages are 44 bytes long

R

Reply flag. When set indicates this is a DELETE RKEY REPLY

O

Reserved bit

Z

Negative response flag

Count

Number of RMBs being deleted by this message. Maximum value is 8

Error Mask

If this is a negative response, indicates which RMBs were not successfully deleted. Each bit corresponds to a listed RMB. So for example b'01010000' indicates that the second and fourth Rkeys weren't successfully deleted.

Deleted Rkeys

A list of Count Rkeys. Provided on the request flow and echoed back on the response flow. Each Rkey is valid on the link this message is sent over, and represents a deleted RMB. Up to eight RMBs can be deleted in this message.

A.3.8. TEST LINK LLC message format

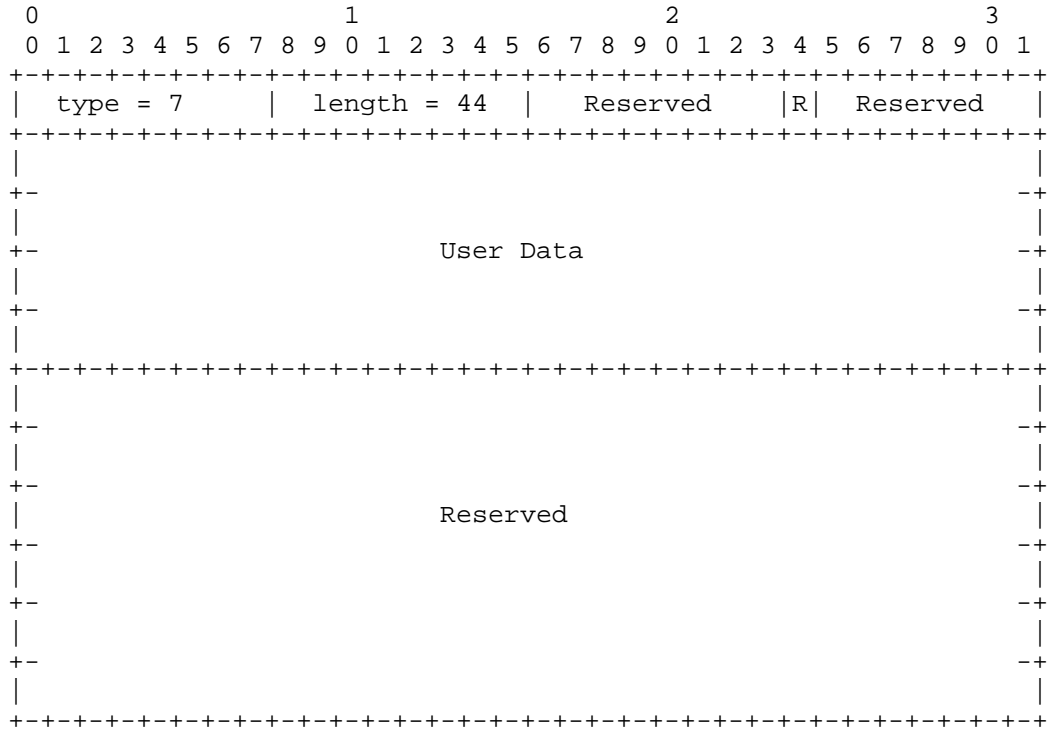


Figure 38 TEST LINK LLC message format

The TEST_LINK request can be sent from either peer to the other on an existing SMC-R link at any time to test that the SMC-R link is active and healthy at the software level. A peer which receives a TEST_LINK LLC message immediately sends back a TEST_LINK reply, echoing back the user data. Also refer to 4.5.3. TCP Keepalive processing.

The contents of this message are:

Type

Type 7 indicates TEST LINK

Length

All LLC messages are 44 bytes long

R

Reply flag. When set indicates this is a TEST LINK REPLY

User Data

The receiver of this message echoes the sender's data back in a TEST_LINK response LLC message

A.4. Connection Data Control (CDC) message format

The RMBE control data is communicated using Connection Data Control (CDC) messages, which use RDMA message passing using inline data, similar to LLC messages. Also similar to LLC messages, this data block is 44 bytes long to ensure that it can fit into private data areas of receive WQEs, without requiring the receiver to post receive buffers.

Unlike LLC messages, this data is integral to the data path so its processing must be prioritized and optimized similarly to other data path processing. While LLC messages may be processed on a slower path than data, these messages cannot be.

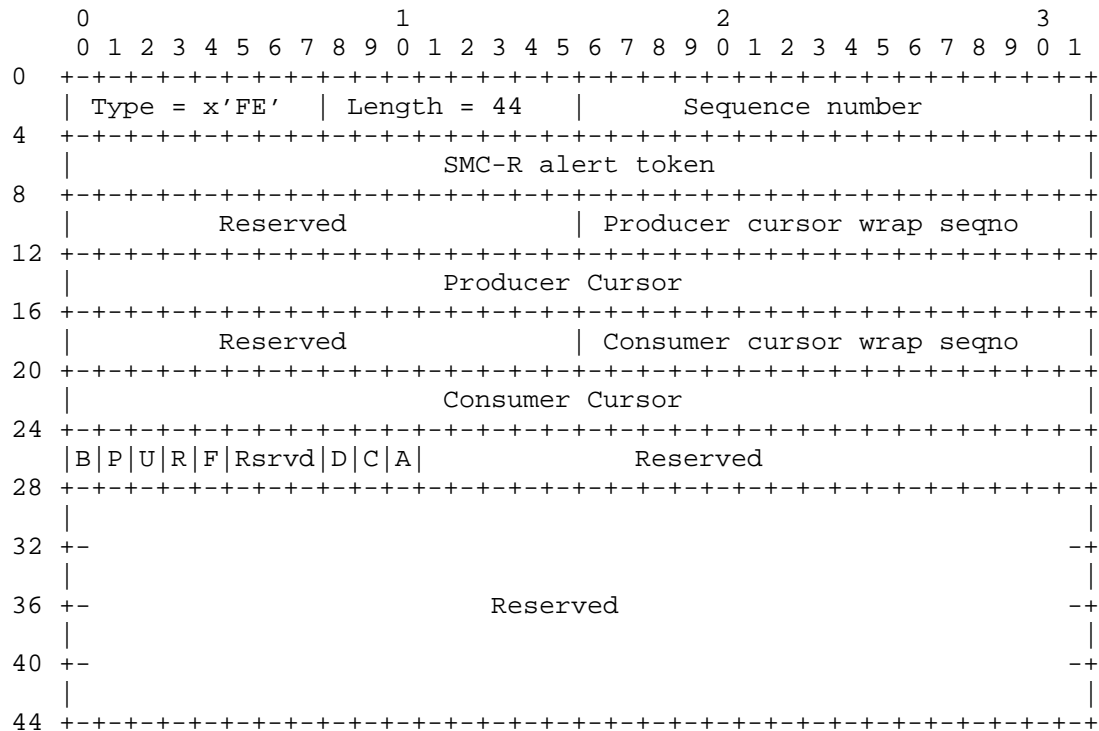


Figure 39 Connection Data Control (CDC) Message Format

Type = x'FE'

This type number has the two high order bits turned on to enable processing to quickly distinguish it from an LLC message

Length = 44

The length of inline data that does not require posting of a receive buffer.

Sequence number

A 2 byte unsigned integer that represents a wrapping sequence number. The initial value is one and this value can wrap to 0. Incremented with every control message send, except for the failover data validation message, and used to guard against processing an old control message out of sequence, and also used in failover data validation. In normal usage, if this number is

less than the last received value, discard this message. If greater, processes this message. Old control messages can be lost with no ill effect, but cannot be processed after newer ones.

If this is a failover validation CDC message (F flag set), then the receiver must verify that it has received and fully processed the RDMA write that was described by the CDC message with the sequence number in this message. If not, the TCP connection must be reset, to guard against data loss. Details of this processing are in section 4.6.1.

SMC-R alert token

The endpoint-assigned alert token that identifies which TCP connection on the link group this control message refers to.

Producer cursor wrap seqno

A 2 byte unsigned integer that represents wrapping counter incremented by the producer whenever the data written into this RMBE receiver buffer causes a wrap (i.e. the producer cursor wraps). This is used by the receiver to determine when new data is available even though the cursors appear unchanged such as when a full window size write is completed (Producer cursor of this RMBE sent by peer = Local Consumer Cursor) or in scenarios where the Producer Cursor sent for this RMBE < Local Consumer Cursor).

Producer cursor

Unsigned, 4 byte integer that is a wrapping offset into the RMBE data area. Points to the next byte of data to be written by the sender. Can advance up to the receiver's Consumer Cursor as known by the sender. When the urgent data present indicator is on then points one byte beyond the last byte of urgent data. When computing this cursor, the presence of the eyecatcher in the RMBE data area must be accounted for. The first writeable data location in the RMBE is at offset 4, so this cursor begins at 4 and wraps to 4.

Consumer cursor wrap seqno

2 byte unsigned integer that mirrors the value of the Producer cursor wrap sequence number when the last read from this RMBE occurred. Used as an indicator on how far along the consumer is in reading data (i.e. processed last wrap point or not). The

producer side can use this indicator to detect whether more data can be written to the partner in full window write scenarios (where the Producer Cursor = Consumer Cursor as known on the remote RMBE). In this scenario if the consumer sequence number equals the local producer sequence number the producer knows that more data can be written.

Consumer Cursor

Unsigned 4 byte integer that is a wrapping offset into the sender's RMBE data area. Points to the offset of the next byte of data to be consumed by the peer in its own RMBE. When computing this cursor, the presence of the eyecatcher in the RMBE data area must be accounted for. The first writeable data location in the RMBE is at offset 4, so this cursor begins at 4 and wraps to 4. The sender cannot write beyond this cursor into the peer's RMBE without causing data loss.

B-bit

Writer blocked indicator: Sender is blocked for writing, requires explicit notification when receive buffer space is available.

P-bit

Urgent data pending: Sender has urgent data pending for this connection

U-bit

Urgent data present: Indicates that urgent is data present in the RMBE data area, and the producer cursor points to one byte beyond the last byte of urgent data.

R-bit

Request for consumer cursor update: Indicates that a consumer cursor update is requested bypassing any window size optimization algorithms.

F-bit

Failover validation indicator: sent by a peer to guard against data loss during failover when the TCP connection is being moved to another SMC-R link in the link group. When this bit is set the only other fields in the CDC message that are significant are the type, length, SMC-R alert token and the sequence number. The

receiver must validate that it has fully processed the RDMA write described by the previous CDC message bearing the same sequence number as this validation message. If it has, no further action is required. If it has not, the TCP connection must be reset. This processing is described in detail in section 4.6.1.

D-bit

Sending done indicator: Sent by a peer when it is done writing new data into the receiver's RMBE data area.

C-bit

Peer Closed Connection indicator: Sent by a peer when it is completely done with this connection and will no longer be making any updates to the receiver's RMBE, and will also not be sending any more control messages.

A-bit

Abnormal Close indicator: Sent by a peer when the connection is abnormally terminated (for example, the TCP connection was Reset). When sent it indicates that the peer is completely done with this connection and will no longer be making any updates to this RMBE or sending any more control messages. It also indicates that the RMBE owner must flush any remaining data on this connection and surface an error return code to any outstanding socket APIs on this connection (same processing as receiving an RST segment on a TCP connection).

Appendix B.

Socket API considerations

A key design goal for SMC-R is to require no application changes for exploitation. It is confined to socket applications using stream (i.e. TCP protocol) sockets over IPv4 or IPv6. By virtue of the fact that the switch to the SMC-R protocol occurs after a TCP connection is established no changes are required in socket address family or in the IP addresses and ports that the socket application are using. Existing socket APIs that allow the application to retrieve local and remote socket address structures for an established TCP connection (for example, `getsockname()` and `getpeername()`) will continue to function as they have before. Existing DNS setup and APIs for resolving hostnames to IP addresses and vice versa also continue to function without any changes. In general all of the usual socket APIs that are used for TCP communicates (send APIs, recv APIs, etc.) will continue to function as they do today even if SMC-R is used as the underlying protocol.

Each SMC-R enabled implementation does however need to pay special attention to any socket APIs that have a reliance on the underlying TCP and IP protocols and ensure that their behavior in an SMC-R environment is reasonable and minimizes impact to the application. While the basic socket API set is fairly similar across different Operating Systems, when it comes to advanced socket API options there is more variability. Each implementation needs to perform a detailed analysis of its API options and SMC-R impact and implications. As part of that step a discussion or review with other implementations supporting SMC-R would be useful to ensure a consistent implementation.

`setsockopt()/getsockopt()` considerations

These APIs allow socket applications to manipulate socket, transport (TCP/UDP) and IP level options associated with a given socket. Typically, a platform restricts the number of IP options available to stream (TCP) socket applications given their connection oriented nature. The general guideline here is to continue processing these APIs in a manner that allows for application compatibility. Some options will be relevant to the SMC-R protocol and will require special processing under the covers. For example, the ability to manipulate TCP send and receive buffer sizes is still valid for SMC-R. However, other options may have no meaning for SMC-R. For example, if an application enabled the `TCP_NODELAY` option to disable Nagle's algorithm it should have no real effect in SMC-R communications as there is no notion of Nagle's algorithm with this new protocol. But the implementation must accept the `TCP_NODELAY` option as it does today and save it so that it can be later extracted

via `getsockopt()` processing. Note that any TCP or IP level options will still have an effect on any TCP/IP packets flowing for an SMC-R connection (i.e. as part of TCP/IP connection establishment and TCP/IP connection termination packet flows).

Under the covers manipulation of the TCP options will also include the SMC layer setting and reading the SMC-R experimental option before and after completion of the 3 way TCP handshake.

Appendix C. Rendezvous Error scenarios

Error scenarios in setting up and managing SMC-R links are discussed in this section.

C.1. SMC Decline during CLC negotiation

A peer to the SMC-R CLC negotiation can send SMC Decline in lieu of any expected CLC message to decline SMC and force the TCP connection back to IP fabric. There can be several reasons for an SMC Decline during the CLC negotiation including: RNIC went down, SMC-R forbidden by local policy, subnet (IPv4) or prefix (IPv6) doesn't match, lack of resources to perform SMC-R. In all cases when an SMC Decline is sent in lieu of an expected CLC message, no confirmation is required and the TCP connection immediately falls back to using the IP fabric.

To prevent ambiguity between CLC messages and application data, an SMC Decline cannot "chase" another CLC message. SMC Decline can only be sent in lieu of an expected CLC message. For example, if the client sends SMC Proposal then its RNIC goes down, it must wait for the SMC Accept for the server and then it can reply to that with an SMC Decline.

This "no chase" rule means that if this TCP connection is not a first contact between RoCE peers, a server cannot send SMC Decline after sending SMC Accept - it can only either break the TCP connection. Similarly, once the client sends SMC Confirm on a TCP connection that isn't first contact, it is committed to SMC-R for this TCP connection and cannot fall back to IP.

C.2. SMC Decline during LLC negotiation

For a TCP connection that represents first contact between RoCE pairs, it is possible for SMC to fail back to IP during the LLC negotiation. This is possible until the first contact SMC link is confirmed. For example, see Figure 40. After a first contact SMC link is confirmed, fallback to IP is no longer possible. The rule that this translates to is: a first contact peer can send SMC Decline at any time during LLC negotiation until it has successfully sent its CONFIRM LINK (request or response) flow. After that point, it cannot fall back to IP.

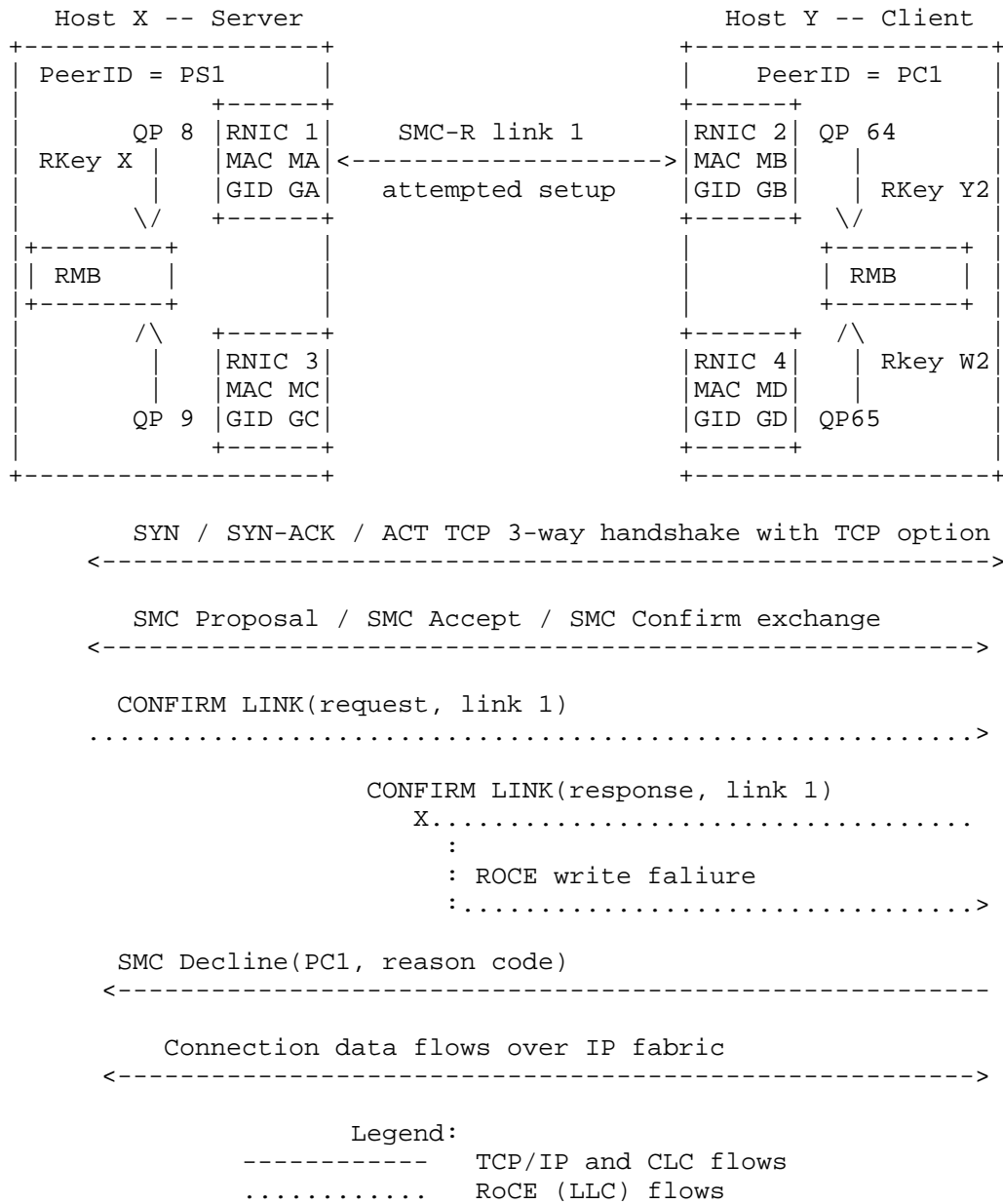


Figure 40 SMC Decline during LLC negotiation

C.3. The SMC Decline window

Because SMC-R does not support fall-back to IP for a TCP connection that is already using RDMA, there are specific rules on when SMC Decline, which signals a fall-back to IP because of an error or problem with the RoCE fabric, can be sent during TCP connection setup. There is a point of no return after which a connection cannot fall back to IP, and RoCE errors that occur after this point require the connection to be broken with a RST flow in the IP fabric.

For first contact, that point of no return is after the Add Link LLC message has been successfully sent for the second SMC-R link. Specifically, the server cannot fall back to IP after receiving either a positive write completion indication for the Add Link request, or after receiving the Add Link response from the client, whichever comes first. The client cannot fall back to IP after either sending a negative Add Link response, receiving a positive write complete on a positive Add Link response, or receiving a Confirm Link for the second SMC-R link from the server, whichever comes first.

For subsequent contact, that point of no return is after the last send of the CLC negotiation completes. This, in combination with the rule that error "chasers" are not allowed during CLC negotiation, means that the server cannot send SMC Decline after sending an SMC Accept, and the client cannot send an SMC Decline after sending an SMC Confirm.

C.4. Out of synch conditions during SMC-R negotiation

The SMC Accept CLC message contains a "first contact" flag that indicates to the client whether or not the server believes it is setting up a new link group, or using an existing link group. This flag is used to detect an out of synch condition between the client and the server. The scenario detected is as follows: There is a single existing SMC-R link between the peers. After the client sends the SMC Proposal CLC message, the existing SMC-R link between the client and the server fails. The client cannot chase the SMC Proposal CLC message with an SMC Decline CLC message in this case because the client does not yet know that the server would have wanted to choose the SMC-R link that just crashed. The QP that failed recovers before the server returns its SMC Accept CLC message. This means that there is a QP but no SMC link. Since the server had not yet learned of the SMC link failure when it sent the SMC Accept CLC message, it attempts to re-use the SMC link that just failed. This means the server would not set the "first contact" flag, indicating to the client that the server thinks it is reusing an SMC-

R link. However the client does not have an SMC-R link that matches the server's specification. Because the "first contact" flag is off, the client realizes it is out of synch with the server and sends SMC Decline to cause the connection to fall back to IP.

C.5. Timeouts during CLC negotiation

Because the SMC-R negotiation flows as TCP data, there are built-in timeouts and retransmits at the TCP layer for individual messages. Implementations also must protect the overall TCP/CLC handshake with a timer or timers to prevent connections from hanging indefinitely due to SMC-R processing. This can be done with individual timers for individual CLC messages or an overall timer for the entire exchange, which may include the TCP handshake and the CLC handshake under one timer or separate timers. This decision is implementation dependent.

If the TCP and/or CLC handshakes time out, the TCP connection must be terminated as it would be in a legacy IP environment when connection setup doesn't complete in a timely manner. Because the CLC flows are TCP messages, if they cannot be sent and received in a timely fashion, the TCP connection is not healthy and would not work if fallback to IP were attempted.

C.6. Protocol errors during CLC negotiation

Protocol errors occur during CLC negotiation when a message is received that is not expected. For example, a peer that is expecting a CLC message but instead receives application data has experienced a protocol error, and also indicates a likely software error as the two sides are out of synch. When application data is expected, this data is not parsed to ensure it's not a CLC message.

When a peer is expecting a CLC negotiation message, any parsing error except a bad enumerated value in that message must be treated as application data. The CLC negotiation messages are designed with beginning and ending eyecatchers to help verify that they are actually the expected message. If other parsing errors in an expected CLC message occur, such as incorrect length fields or incorrectly formatted fields, the message must be treated as application data.

All protocol errors with the exception of bad enumerated values must result in termination of the TCP connection. No fallback to IP is allowed in the case of a protocol error because if the protocols are out of synch, mismatched, or corrupted, then data and security integrity cannot be ensured.

The exception to this rule is enumerated values, for example the QP MTU values on SMC Accept and SMC Confirm. If a reserved value is received, the proper error response is to send SMC Decline and fall back to IP. The reason for this is that use of a reserved enumerated value indicates that the other partner likely has additional support that the receiving partner does not have. This indicated mismatch of SMC-R capabilities is not an integrity problem, but indicates that SMC-R cannot be used for this connection

C.7. Timeouts during LLC negotiation

Whenever a peer sends an LLC message to which a reply is expected, it sets a timer after the send posts to wait for the reply. An expected response may be a reply flavor of the LLC message (for example CONFIRM LINK REPLY) or a new LLC message (for example an ADD LINK CONTINUATION expected from the server by the client if there are more Rkeys to communicate).

On LLC flows that are part of a first contact setup of a link group, the value of the timer is implementation dependent but should be long enough to allow the other peer have a write complete timeout and 2-3 retransmits of an SMC Decline on the TCP fabric. For LLC flows that are maintaining the link group and not part of first contact setup of a link group, the timers may be shorter. Upon receipt of an expected reply the timer is cancelled. If a timer pops without a reply having been received, the sender must initiate a recovery action

During first contact processing, failure of an LLC verification timer is a should-not-occur which indicates a problem with one of the endpoints. The reason for this is that if there is a "routine" failure in the RoCE fabric that causes an LLC verification send to fail, the sender will get a write completion failure and will then send SMC Decline to the partner. The only time an LLC verification timer will expire on a first contact is when the sender thinks the send succeeded but it actually didn't. Because of the reliable connected nature of QP connections on the RoCE fabric, this indicates a problem with one of the peers, not with the RoCE fabric.

After the reliable connected QP for the first SMC-R link in a link group is set up on initial contact, the client sets a timer to wait for a RoCE verification message from the server that the QP is actually connected and usable. If the server experiences a failure sending its QP confirmation message, it will send SMC Decline, which should arrive at the client before the client's verification timer expires. If the client's timer expires without receiving either an SMC Decline or a RoCE message confirmation from the server, there is

a problem either with the server or with the TCP fabric. In either case the client must break the TCP connection and clean up the SMC-R link.

There are two scenarios in which the client's response to the QP verification message fails to reach the server. The main difference is whether or not the client has successfully completed the send of the CONFIRM LINK response.

In the normal case of a problem with the RoCE path, the client will learn of the failure by getting a write completion failure, before the server's timer expires. In this case, the client sends an SMC Decline CLC message to the server and the TCP connection falls back to IP.

If the client's send of the Confirmation message receives a positive return code but for some reason still does not reach the server, or the client's SMC Decline CLC message fails to reach the server after the client fails to send its RoCE confirmation message, then the server's timer will time out and the server must break the TCP connection by sending RST. This is expected to be a very rare case, because if the client cannot send its CONFIRM LINK RSP LLC message, the client should get a negative return code and initiate fallback to IP. A client receiving a positive return code on a send that fails to reach the server should be extremely rare.

C.7.1. Recovery actions for LLC timeouts and failures

The following table describes recovery actions for LLC timeouts. A write completion failure or other indication of failure to send on the send of the LLC command is treated the same as a timeout.

LLC Message: CONFIRM LINK from server (first contact, first link in the link group)

Timer waits for: CONFIRM LINK reply from client

Recovery action: Break the TCP connection by sending RST and clean up the link. The server should have received an SMC Decline from the client by now if the client had an LLC send failure.

LLC Message: CONFIRM LINK from server (first contact, second link in the link group)

Timer waits for: CONFIRM LINK reply from client

Recovery action: The second link was not successfully set up. Send DELETE LINK to the client. Connection data cannot flow in the first link in the link group, until the reply to this DELETE LINK is received, to prevent the peers from being out of synch on the state of the link group.

LLC Message: CONFIRM LINK from server (not first contact)

Timer Waits for: CONFIRM LINK reply from client

Recovery action: Clean up the new link and set a timer to retry. Send DELETE LINK to the client, in case the client has a longer timer interval, so the client can stop waiting

LLC Message: CONFIRM LINK REPLY from client (first contact)

Timer waits for: ADD LINK from server

Recovery action: Clean up the SMC-R link and break the TCP connection by sending RST over the IP fabric. There is a problem with the server. If the server had a send failure, it should have have sent SMC Decline by now.

LLC Message: ADD LINK from server (first contact)

Timer waits for: ADD LINK reply from client

Recovery action: Break the TCP connection with RST and clean up RoCE resources. The connection is past the point where the server can fall back to IP, and if the client had a send problem it should have sent SMC Decline by now.

LLC Message: ADD LINK from server (not first contact)

Timer waits for: ADD LINK reply from client

Recovery action: Clean up resources (QP, RMB keys, etc) for the new link and treat the link that the ADD LINK was sent over as if it had failed. If there is another link available to resend the ADD LINK and the link group still needs another link, retry the ADD LINK over another link in the link group.

LLC Message: ADD LINK REPLY from client (and there are more Rkeys to be communicated)

Timer waits for: ADD LINK CONTINUATION from server

Recovery action: Treat the same as ADD LINK timer failure

LLC Message: ADD LINK REPLY or ADD LINK CONTINUATION reply from client (and there are no more Rkeys to be communicated, for the second link in a first contact scenario)

Timer waits for: CONFIRM LINK from the server on the new link

Recovery action: The new link has failed to set up. Send DELETE LINK to the server. Do not consider the socket opened to the client application until receiving confirmation from the server in the form of a DELETE LINK request for this link and sending the reply (to prevent the partners from being out of synch on the state of the link group).

Set a timer to send another ADD LINK to the server if there is still an unused RNIC on the client side.

LLC Message: ADD LINK REPLY or ADD LINK CONTINUATION reply from the client (and there are no more Rkeys to be communicated)

Timer waits for: CONFIRM LINK from the server, over the new link

Recovery action: Send a DELETE LINK to the server for the new link, then clean up any resource allocated for the new link and set a timer to send ADD LINK to the server if there is still an unused RNIC on the client side. The new link has failed to set up, but the link that the ADD LINK exchange occurred over is unaffected.

LLC Message: ADD LINK CONTINUATION from server

Timer waits for: ADD LINK CONTINUATION REPLY from client

Recovery action: Treat the same as ADD LINK timer failure

LLC Message: ADD LINK CONTINUATION reply from client (first contact, and RMB count fields indicate that the server owes more ADD LINK CONTINUATION messages)

Timer waits for: ADD LINK CONTINUATION from the server

Recovery action: Clean up the SMC link and break the TCP connection by sending RST. There is a problem with the server.

If the server had a send failure, it should have have sent SMC Decline by now.

LLC Message: ADD LINK CONTINUATION reply from client (not first contact and RMB count fields indicate that the server owes more ADD LINK CONTINUATION messages)

Timer waits for: ADD LINK CONTINUATION from server

Recovery action: Treat as is if client detected link failure on the link the ADD LINK exchange is using. Send DELETE LINK to the server over another active link if one exists, otherwise clean up the link group.

LLC Message: DELETE LINK from client

Timer waits for: DELETE LINK request from server

Recovery action: If the scope of the request is to delete a single link, the surviving link, over which the client sent the DELETE LINK is no longer usable either. If this is the last link in the link group, end TCP connections over the link group by sending RST packets. If there are other surviving links in the link group, resend over a surviving link. Also send a DELETE LINK over a surviving link for the link that the client attempted to send the initial DELETE LINK message over. If the scope of the request is to delete the entire link group, try resending on other links in the link group until success is achieved. If all sends fail, tear down the link group and any TCP connections that exist on it.

LLC Message: DELETE LINK from server (scope: entire link group)

Timer waits for: Confirmation from the adapter that the message was delivered.

Recovery action: Tear down the link group and any TCP connections that exist over it.

LLC Message: DELETE LINK from server (scope: single link)

Timer waits for: DELETE LINK reply from the client

Recovery action: The link over which the client sent the DELETE LINK is no longer usable either. If this is the last link in the link group, end TCP connections over the link group by sending RST packets. If there are other surviving links in the link

group, resend over a surviving link. Also send a DELETE LINK over a surviving link for the link that the server attempted to send the initial DELETE LINK message over. If the scope of the request is to delete the entire link group, try resending on other links in the link group until success is achieved. If all sends fail, tear down the link group and any TCP connections that exist on it.

LLC Message: CONFIRM RKEY from the client

Timer waits for: CONFIRM RKEY REPLY from the server

Recovery action: Perform normal client procedures for detection of failed link. The link over which the message was sent has failed.

LLC Message: CONFIRM RKEY from the server

Timer waits for : CONFIRM RKEY REPLY from the client

Recovery action: Perform normal server procedures for detection of failed link. The link over which the message was sent has failed.

LLC Message: TEST LINK from the client

Timer waits for: TEST LINK REPLY from the server

Recovery action: Perform normal client procedures for detection of failed link. The link over which the message was sent has failed.

LLC Message: TEST LINK from the server

Timer waits for : TEST LINK REPLY from the client

Recovery action: Perform normal server procedures for detection of failed link. The link over which the message was sent has failed.

The following table describes recovery actions for invalid LLC messages. These could be misformatted or contain out of synch data.

LLC Message received: CONFIRM LINK from server

What could be bad: Incorrect link information

Recovery action: Protocol error. The link must be brought down by sending a DELETE LINK for the link over another link in the link group if one exists. If this is first contact, fall back to IP by sending SMC Decline to server.

LLC Message received: ADD LINK

What could be bad: Undefined enumerated MTU value

Recovery action: Send negative ADD LINK reply with reason code x'2'

LLC Message received: ADD LINK reply from client

What could be bad: Client side link information that would result in a parallel link being set up

Recovery action: Parallel links are not permitted. Delete the link by sending DELETE LINK to the client over another link in the link group.

LLC Message received: Any link group command from the server except DELETE LINK for the entire link group

What could be bad: Client has sent DELETE LINK for the link that the message was received on

Recovery action: Ignore the LLC message. Worst case the server will time out. Best case the DELETE LINK crosses with the command from the server and the server realizes it failed.

LLC Message received: ADD LINK CONTINUATION from the server or ADD LINK CONTINUATION REPLY from the client

What could be bad: Number of RMBs provided doesn't match count given on initial ADD LINK or ADD LINK reply message

Recovery action: Protocol error. Treat as if detected link outage

LLC Message received: DELETE LINK from client

What could be bad: Link indicated doesn't exist

Recovery action: If the link is in the process of being cleaned up, assume timing window and ignore message. Otherwise, send DELETE LINK REPLY with reason code 1.

LLC Message received: DELETE LINK from server

What could be bad: Link indicated doesn't exist

Recovery action: Send DELETE LINK REPLY with reason code 1.

LLC Message received: CONFIRM RKEY from either client or server

What could be bad: No Rkey provided for one or more of the links in the link group

Recovery action: Treat as if detected failure of the link(s) for which no RKEY was provided

LLC message received: DELETE RKEY

Specified RKey doesn't exist

Send negative DELETE RKEY response.

LLC message received: TEST LINK reply

What could be bad: User data doesn't match what was sent in the TEST LINK request

Recovery action: Treat as if detected that the link has gone down. This is a protocol error

LLC message received: Unknown LLC type with high order bits of opcode equal b'10'

What could be bad: This is an optional LLC message which the receiver does not support

Recovery action: Ignore (silently discard) the message

LLC message received: any unambiguously incorrect or out of synch LLC message

What it indicates: Link is out of sync

Recovery action: Treat as if detected that the link has gone down. Note that an unsupported or unknown LLC opcode whose two high order bits are b'10' is not an error, and must be silently discarded. Any other unknown or unsupported LLC opcode is an error.

C.8. Failure to add second SMC-R link to a link group

When there is any failure in setting up the second SMC-R link in an SMC-R link group, including confirmation timer expiration, the SMC-R link group is allowed to continue, without available failover. However this situation is extremely undesirable and the server must endeavor to correct it as soon as it can.

The server peer in the SMC-R link group must set a timer to drive it to retry setup of a failed additional SMC-R link. The server will immediately retry the SMC-R link setup when the first of the following events occurs:

- o The retry timer expires
- o A new RNIC becomes available to the server, on the same LAN as the SMC-R link group
- o An "Add Link" LLC request message is received from the client, which indicates availability of a new RNIC on the client side.

Authors' Addresses

Mike Fox
IBM
3039 Cornwallis Rd.
Research Triangle Park, NC 27709

Email: mjfox@us.ibm.com

Constantinos (Gus) Kassimis
IBM
3039 Cornwallis Rd.
Research Triangle Park, NC 27709

Email: kassimis@us.ibm.com

Jerry Stevens
IBM
3039 Cornwallis Rd.
Research Triangle Park, NC 27709

Email: sjerry@us.ibm.com

Storage Maintenance (StorM) Working Group
Internet Draft
Intended status: Proposed Standard
Expires: January 2014
Obsoletes: 5046

Michael Ko
Consultant
Alexander Nezhinsky
Mellanox
July 9, 2013

iSCSI Extensions for RDMA Specification
draft-ietf-storm-iser-15.txt

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January, 2014.

Abstract

iSCSI Extensions for Remote Direct Memory Access (RDMA) provides the RDMA data transfer capability to iSCSI by layering iSCSI on top of an RDMA-Capable Protocol. An RDMA-Capable Protocol provides RDMA Read and Write services, which enable data to be transferred directly into SCSI I/O Buffers without intermediate data copies. This document describes the extensions to the iSCSI protocol to support RDMA services as provided by an RDMA-Capable Protocol.

This document obsoletes RFC 5046.

Table of Contents

1	Definitions and Acronyms	6
1.1	Definitions	6
1.2	Acronyms	12
1.3	Conventions	14
2	Introduction	15
2.1	Motivation	15
2.2	iSCSI/iSER Layering	16
2.3	Architectural Goals	17
2.4	Protocol Overview	17
2.5	RDMA services and iSER	19
2.5.1	STag.....	19
2.5.2	Send.....	20
2.5.3	RDMA Write.....	21
2.5.4	RDMA Read.....	21
2.6	SCSI Read Overview	21
2.7	SCSI Write Overview	22
3	Upper Layer Interface Requirements	23
3.1	Operational Primitives offered by iSER	23
3.1.1	Send_Control.....	24
3.1.2	Put_Data.....	24
3.1.3	Get_Data.....	24
3.1.4	Allocate_Connection_Resources.....	25
3.1.5	Deallocate_Connection_Resources.....	25
3.1.6	Enable_Datamover.....	25
3.1.7	Connection_Terminate.....	26
3.1.8	Notice_Key_Values.....	26
3.1.9	Deallocate_Task_Resources.....	26
3.2	Operational Primitives used by iSER	27
3.2.1	Control_Notify.....	27
3.2.2	Data_Completion_Notify.....	27
3.2.3	Data_ACK_Notify.....	28
3.2.4	Connection_Terminate_Notify.....	28
3.3	iSCSI Protocol Usage Requirements	28
4	Lower Layer Interface Requirements	30
4.1	Interactions with the RCaP Layer	30
4.2	Interactions with the Transport Layer	31
5	Connection Setup and Termination	32
5.1	iSCSI/iSER Connection Setup	32
5.1.1	Initiator Behavior.....	33
5.1.2	Target Behavior.....	35
5.1.3	iSER Hello Exchange.....	36
5.2	iSCSI/iSER Connection Termination	39
5.2.1	Normal Connection Termination at the Initiator.....	39
5.2.2	Normal Connection Termination at the Target.....	40
5.2.3	Termination without Logout Request/Response PDUs.....	40

Internet-Draft	iSER Specification	July 2013
6	Login/Text Operational Keys	42
6.1	HeaderDigest and DataDigest	42
6.2	MaxRecvDataSegmentLength	42
6.3	RDMAExtensions	43
6.4	TargetRecvDataSegmentLength	44
6.5	InitiatorRecvDataSegmentLength	44
6.6	OFMarker and IFMarker	45
6.7	MaxOutstandingUnexpectedPDUs	45
6.8	MaxAHSLength	46
6.9	TaggedBufferForSolicitedDataOnly	46
6.10	iSERHelloRequired.....	47
7	iSCSI PDU Considerations	48
7.1	iSCSI Data-Type PDU	48
7.2	iSCSI Control-Type PDU	49
7.3	iSCSI PDUs	49
7.3.1	SCSI Command.....	49
7.3.2	SCSI Response.....	51
7.3.3	Task Management Function Request/Response.....	53
7.3.4	SCSI Data-out.....	54
7.3.5	SCSI Data-in.....	55
7.3.6	Ready To Transfer (R2T).....	57
7.3.7	Asynchronous Message.....	59
7.3.8	Text Request & Text Response.....	59
7.3.9	Login Request & Login Response.....	60
7.3.10	Logout Request & Logout Response	60
7.3.11	SNACK Request	60
7.3.12	Reject	60
7.3.13	NOP-Out & NOP-In	61
8	Flow Control and STag Management	62
8.1	Flow Control for RDMA Send Messages	62
8.1.1	Flow Control for Control-Type PDUs from the Initiator.....	62
8.1.2	Flow Control for Control-Type PDUs from the Target.....	65
8.2	Flow Control for RDMA Read Resources	66
8.3	STag Management	67
8.3.1	Allocation of STags.....	67
8.3.2	Invalidation of STags.....	67
9	iSER Control and Data Transfer	69
9.1	iSER Header Format	69
9.2	iSER Header Format for iSCSI Control-Type PDU	69
9.3	iSER Header Format for iSER Hello Message	72
9.4	iSER Header Format for iSER HelloReply Message	73
9.5	SCSI Data Transfer Operations	74
9.5.1	SCSI Write Operation.....	74
9.5.2	SCSI Read Operation.....	75
9.5.3	Bidirectional Operation.....	76
10	iSER Error Handling and Recovery	77
10.1	Error Handling.....	77

Internet-Draft	iSER Specification	July 2013
10.1.1	Errors in the Transport Layer	77
10.1.2	Errors in the RCaP Layer	78
10.1.3	Errors in the iSER Layer	78
10.1.4	Errors in the iSCSI Layer	80
10.2	Error Recovery.....	82
10.2.1	PDU Recovery	82
10.2.2	Connection Recovery	83
11	Security Considerations	84
12	IANA Considerations	85
13	References	86
13.1	Normative References.....	86
13.2	Informative References.....	86
14	Appendix A: Summary of Changes from RFC 5046	88
15	Appendix B: Message Format for iSER	90
15.1	iWARP Message Format for iSER Hello Message.....	90
15.2	iWARP Message Format for iSER HelloReply Message.....	91
15.3	iSER Header Format for SCSI Read Command PDU.....	92
15.4	iSER Header Format for SCSI Write Command PDU.....	93
15.5	iSER Header Format for SCSI Response PDU.....	94
16	Appendix C: Architectural discussion of iSER over InfiniBand	95
16.1	Host side of iSCSI & iSER connections in Infiniband.....	95
16.2	Storage side of iSCSI & iSER mixed network environment....	96
16.3	Discovery processes for an InfiniBand Host.....	96
16.4	IBTA Connection specifications.....	97
17	Acknowledgments	98

Table of Figures

Figure 1 Example of iSCSI/iSER Layering in Full Feature Phase ...	16
Figure 2 iSER Header Format	69
Figure 3 iSER Header Format for iSCSI Control-Type PDU	70
Figure 4 iSER Header Format for iSER Hello Message	72
Figure 5 iSER Header Format for iSER HelloReply Message	73
Figure 6 SendSE Message containing an iSER Hello Message	90
Figure 7 SendSE Message containing an iSER HelloReply Message ...	91
Figure 8 iSER Header Format for SCSI Read Command PDU	92
Figure 9 iSER Header Format for SCSI Write Command PDU	93
Figure 10 iSER Header Format for SCSI Response PDU	94
Figure 11 iSCSI and iSER on IB	95
Figure 12 Storage Controller with TCP, iWARP, and IB Connections	96

1 Definitions and Acronyms

1.1 Definitions

Advertisement (Advertised, Advertise, Advertisements, Advertises) -

The act of informing a remote iSER (iSCSI Extensions for RDMA) Layer that a local node's buffer is available to it. A Node makes a buffer available for incoming RDMA Read Request Message or incoming RDMA Write Message access by informing the remote iSER Layer of the Tagged Buffer identifiers (STag, Base Offset, and buffer length). Note that this Advertisement of Tagged Buffer information is the responsibility of the iSER Layer on either end and is not defined by the RDMA-Capable Protocol. A typical method would be for the iSER Layer to embed the Tagged Buffer's STag, Base Offset, and buffer length in a message destined for the remote iSER Layer.

Base Offset - A value when added to the Buffer Offset forms the Tagged Offset.

Completion (Completed, Complete, Completes) - Completion is defined as the process by which the RDMA-Capable Protocol layer informs the iSER Layer, that a particular RDMA Operation has performed all functions specified for the RDMA Operation.

Connection - A connection is a logical bidirectional communication channel between the initiator and the target, e.g., a TCP connection. Communication between the initiator and the target occurs over one or more connections. The connections carry control messages, SCSI commands, parameters, and data within iSCSI Protocol Data Units (iSCSI PDUs).

Connection Handle - An information element that identifies the particular iSCSI connection and is unique for a given iSCSI Layer and the underlying iSER Layer. Every invocation of an Operational Primitive is qualified with the Connection Handle.

Data Sink - The peer receiving a data payload. Note that the Data Sink can be required to both send and receive RCaP (RDMA-Capable Protocol) Messages to transfer a data payload.

Data Source - The peer sending a data payload. Note that the Data Source can be required to both send and receive RCaP Messages to transfer a data payload.

Datamover Interface (DI) - The interface between the iSCSI Layer and the Datamover Layer as described in [DA].

Datamover Layer - A layer that is directly below the iSCSI Layer and above the underlying transport layers. This layer exposes and uses a set of transport independent Operational Primitives for the communication between the iSCSI Layer and itself. The Datamover layer, operating in conjunction with the transport layers, moves the control and data information on the iSCSI connection. In this specification, the iSER Layer is the Datamover layer.

Datamover Protocol - A Datamover protocol is the wire-protocol that is defined to realize the Datamover layer functionality. In this specification, the iSER protocol is the Datamover protocol.

Inbound RDMA Read Queue Depth (IRD) - The maximum number of incoming outstanding RDMA Read Requests that the RDMA-Capable Controller can handle on a particular RCaP Stream at the Data Source. For some RDMA-Capable Protocol layers, the term "IRD" may be known by a different name. For example, for InfiniBand, the equivalent for IRD is the Responder Resources.

I/O Buffer - A buffer that is used in a SCSI Read or Write operation so SCSI data may be sent from or received into that buffer.

iSCSI - The iSCSI protocol as defined in [iSCSI] is a mapping of the SCSI Architecture Model of SAM-5 over TCP.

iSCSI control-type PDU - Any iSCSI PDU that is not an iSCSI data-type PDU and also not a SCSI Data-out PDU carrying solicited data is defined as an iSCSI control-type PDU. Specifically, it is to be noted that SCSI Data-out PDUs for unsolicited data are defined as iSCSI control-type PDUs.

iSCSI data-type PDU - An iSCSI data-type PDU is defined as an iSCSI PDU that causes data transfer via RDMA operations at the iSER layer, transparent to the remote iSCSI Layer, to take place between the peer iSCSI nodes on a full feature phase iSCSI connection. An iSCSI data-type PDU, when requested for transmission by the sender iSCSI Layer, results in the associated data transfer without the participation of the remote iSCSI Layer, i.e. the PDU itself is not delivered as-is to the remote iSCSI Layer. The following iSCSI PDUs constitute the set of iSCSI data-type PDUs - SCSI Data-In PDU and R2T PDU.

iSCSI Layer - A layer in the protocol stack implementation within an end node that implements the iSCSI protocol and interfaces with the iSER Layer via the Datamover Interface.

iSCSI PDU (iSCSI Protocol Data Unit) - The iSCSI Layer at the initiator and the iSCSI Layer at the target divide their communications into messages. The term "iSCSI protocol data unit" (iSCSI PDU) is used for these messages.

iSCSI/iSER Connection - An iSER-assisted iSCSI connection. An iSCSI connection that is not iSER-assisted always maps onto a TCP connection at the transport level. But an iSER-assisted iSCSI connection may not have an underlying TCP connection. For some RCaP implementation (e.g., iWARP), an iSER-assisted iSCSI connection has an underlying TCP connection. For other RCaP implementation (e.g., InfiniBand), there is no underlying TCP connection. (In the specific example of InfiniBand [IB], an iSER-assisted iSCSI connection is directly mapped onto the InfiniBand RC channel.)

iSCSI/iSER Session - An iSER-assisted iSCSI session. All connections of an iSCSI/iSER session are iSCSI/iSER connections.

iSER - iSCSI Extensions for RDMA, the protocol defined in this document.

iSER-assisted - A term generally used to describe the operation of iSCSI when the iSER functionality is also enabled below the iSCSI Layer for the specific iSCSI/iSER connection in question.

iSER-IRD - This variable represents the maximum number of incoming outstanding RDMA Read Requests that the iSER Layer at the initiator grants on a particular RCaP Stream.

iSER-ORD - This variable represents the maximum number of outstanding RDMA Read Requests that the iSER Layer can initiate on a particular RCaP Stream. This variable is maintained only by the iSER Layer at the target.

iSER Layer - The layer that implements the iSCSI Extensions for RDMA (iSER) protocol.

iWARP - A suite of wire protocols comprising of [RDMAP], [DDP], and [MPA] when layered above [TCP]. [RDMAP] and [DDP] may be layered above SCTP or other transport protocols.

Local Mapping - A task state record maintained by the iSER Layer that associates the Initiator Task Tag to the Local STag(s). The specifics of the record structure are implementation dependent.

Local Peer - The implementation of the RDMA-Capable Protocol on the local end of the connection. Used to refer to the local entity when describing protocol exchanges or other interactions between two Nodes.

Node - A computing device attached to one or more links of a network. A Node in this context does not refer to a specific application or protocol instantiation running on the computer. A Node may consist of one or more RDMA-Capable Controllers installed in a host computer.

Operational Primitive - An Operational Primitive is an abstract functional interface procedure that requests another layer to perform a specific action on the requestor's behalf or notifies the other layer of some event. The Datamover Interface between an iSCSI Layer and a Datamover layer within an iSCSI end node uses a set of Operational Primitives to define the functional interface between the two layers. Note that not every invocation of an Operational Primitive may elicit a response from the requested layer. A full discussion of the Operational Primitive types and request-response semantics available to iSCSI and iSER can be found in [DA].

Outbound RDMA Read Queue Depth (ORD) - The maximum number of outstanding RDMA Read Requests that the RDMA-Capable Controller can initiate on a particular RCaP Stream at the Data Sink. For some RDMA-Capable Protocol layer, the term "ORD" may be known by a different name. For example, for InfiniBand, the equivalent for ORD is the Initiator Depth.

Phase Collapse - Refers to the optimization in iSCSI where the SCSI status is transferred along with the final SCSI Data-in PDU from a target. See section 4.2 in [iSCSI].

RCaP Message - One or more packets of the network layer comprising a single RDMA operation or a part of an RDMA Read Operation of the RDMA-Capable Protocol. For iWARP, an RCaP Message is known as an RDMAP Message.

RCaP Stream - A single bidirectional association between the peer RDMA-Capable Protocol layers on two Nodes over a single transport-level stream. For iWARP, an RCaP Stream is known as an RDMAP Stream, and the association is created following a successful Login Phase during which iSER support is negotiated.

RDMA-Capable Protocol (RCaP) - The protocol or protocol suite that provides a reliable RDMA transport functionality, e.g., iWARP, InfiniBand, etc.

RDMA-Capable Controller - A network I/O adapter or embedded controller with RDMA functionality. For example, for iWARP, this could be an RNIC, and for InfiniBand, this could be a HCA (Host Channel Adapter) or TCA (Target Channel Adapter).

RDMA-enabled Network Interface Controller (RNIC) - A network I/O adapter or embedded controller with iWARP functionality.

RDMA Operation - A sequence of RCaP Messages, including control Messages, to transfer data from a Data Source to a Data Sink. The following RDMA Operations are defined - RDMA Write Operation, RDMA Read Operation, and Send Operation.

RDMA Protocol (RDMAP) - A wire protocol that supports RDMA Operations to transfer ULP (Upper Level Protocol) data between a Local Peer and the Remote Peer as described in [RDMAP].

RDMA Read Operation - An RDMA Operation used by the Data Sink to transfer the contents of a Data Source buffer from the Remote Peer to a Data Sink buffer at the Local Peer. An RDMA Read operation consists of a single RDMA Read Request Message and a single RDMA Read Response Message.

RDMA Read Request - An RCaP Message used by the Data Sink to request the Data Source to transfer the contents of a buffer. The RDMA Read Request Message describes both the Data Source and the Data Sink buffers.

RDMA Read Response - An RCaP Message used by the Data Source to transfer the contents of a buffer to the Data Sink, in response to an RDMA Read Request. The RDMA Read Response Message only describes the Data Sink buffer.

RDMA Write Operation - An RDMA Operation used by the Data Source to transfer the contents of a Data Source buffer from the Local Peer to a Data Sink buffer at the Remote Peer. The RDMA Write Message only describes the Data Sink buffer.

Remote Direct Memory Access (RDMA) - A method of accessing memory on a remote system in which the local system specifies the remote location of the data to be transferred. Employing an RDMA-Capable Controller in the remote system allows the access to take

place without interrupting the processing of the CPU(s) on the system.

Remote Mapping - A task state record maintained by the iSER Layer that associates the Initiator Task Tag to the Advertised STag(s) and the Base Offset(s). The specifics of the record structure are implementation dependent.

Remote Peer - The implementation of the RDMA-Capable Protocol on the opposite end of the connection. Used to refer to the remote entity when describing protocol exchanges or other interactions between two Nodes.

SCSI Layer - This layer builds/receives SCSI CDBs (Command Descriptor Blocks) and sends/receives them with the remaining command execute [SAM5] parameters to/from the iSCSI Layer.

Send - An RDMA Operation that transfers the content of a buffer from the Local Peer to an untagged buffer at the Remote Peer.

SendInvSE Message - A Send with Solicited Event and Invalidate Message.

SendSE Message - A Send with Solicited Event Message.

Sequence Number (SN) - DataSN for a SCSI Data-in PDU and R2TSN for an R2T PDU. The semantics for both types of sequence numbers are as defined in [iSCSI].

Session, iSCSI Session - The group of Connections that link an initiator SCSI port with a target SCSI port form an iSCSI session (equivalent to a SCSI I-T nexus). Connections can be added to and removed from a session even while the I-T nexus is intact. Across all connections within a session, an initiator sees one and the same target.

Steering Tag (STag) - An identifier of a Tagged Buffer on a Node (Local or Remote) as defined in [RDMAP] and [DDP]. For other RDMA-Capable Protocols, the Steering Tag may be known by different names but will be herein referred to as STags. For example, for Infiniband, a Remote STag is known as an R-Key, and a Local STag is known as an L-Key, and both will be considered STags.

Tagged Buffer - A buffer that is explicitly Advertised to the iSER Layer at the remote node through the exchange of an STag, Base Offset, and length.

Tagged Offset - The offset within a Tagged Buffer.

Traditional iSCSI - Refers to the iSCSI protocol as defined in [iSCSI] (i.e. without the iSER enhancements).

Untagged Buffer - A buffer that is not explicitly Advertised to the iSER Layer at the remote node.

1.2 Acronyms

Acronym	Definition

AHS	Additional Header Segment
BHS	Basic Header Segment
CO	Connection Only
CRC	Cyclic Redundancy Check
DDP	Direct Data Placement Protocol
DI	Datamover Interface
HCA	Host Channel Adapter
IANA	Internet Assigned Numbers Authority
IB	Infiniband
IETF	Internet Engineering Task Force
I/O	Input - Output
IO	Initialize Only
IP	Internet Protocol
IPoIB	IP over Infiniband
IPsec	Internet Protocol Security
iSER	iSCSI Extensions for RDMA
ITT	Initiator Task Tag

LO	Leading Only
MPA	Marker PDU Aligned Framing for TCP
NOP	No Operation
NSG	Next Stage (during the iSCSI Login Phase)
PDU	Protocol Data Unit
R2T	Ready To Transfer
R2TSN	Ready To Transfer Sequence Number
RCaP	RDMA-Capable Protocol
RDMA	Remote Direct Memory Access
RDMAP	Remote Direct Memory Access Protocol
RFC	Request For Comments
RNIC	RDMA-enabled Network Interface Controller
SAM5	SCSI Architecture Model - 5
SCSI	Small Computer Systems Interface
SNACK	Selective Negative Acknowledgment - also Sequence Number Acknowledgement for data
STag	Steering Tag
SW	Session Wide
TCA	Target Channel Adapter
TCP	Transmission Control Protocol
TMF	Task Management Function
TTT	Target Transfer Tag
ULP	Upper Level Protocol

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].

2.1 Motivation

The iSCSI protocol ([iSCSI]) is a mapping of the SCSI Architecture Model (see [SAM5] and [iSCSI-SAM]) over the TCP protocol. SCSI commands are carried by iSCSI requests and SCSI responses and status are carried by iSCSI responses. Other iSCSI protocol exchanges and SCSI Data are also transported in iSCSI PDUs.

Out-of-order TCP segments in the Traditional iSCSI model have to be stored and reassembled before the iSCSI protocol layer within an end node can place the data in the iSCSI buffers. This reassembly is required because not every TCP segment is likely to contain an iSCSI header to enable its placement and TCP itself does not have a built-in mechanism for signaling ULP message boundaries to aid placement of out-of-order segments. This TCP reassembly at high network speeds is quite counter-productive for the following reasons: wasted memory bandwidth in data copying, need for reassembly memory, wasted CPU cycles in data copying, and the general store-and-forward latency from an application perspective.

The generic term RDMA-Capable Protocol (RCaP) is used to refer to protocol stacks that provide the Remote Direct Memory Access (RDMA) functionality, such as iWARP and InfiniBand.

With the availability of RDMA-Capable Controllers within a host system, it is appropriate for iSCSI to be able to exploit the direct data placement function of the RDMA-Capable Controller like other applications.

iSCSI Extensions for RDMA (iSER) is designed precisely to take advantage of generic RDMA technologies - iSER's goal is to permit iSCSI to employ direct data placement and RDMA capabilities using a generic RDMA-Capable Controller. In summary, iSCSI/iSER protocol stack is designed to enable scaling to high speeds by relying on a generic data placement process and RDMA technologies and products, which enable direct data placement of both in-order and out-of-order data.

This document describes iSER as a protocol extension to iSCSI, both for convenience of description and also because it is true in a very strict protocol sense. However, it is to be noted that iSER is in reality extending the connectivity of the iSCSI protocol defined in [iSCSI], and the name iSER reflects this reality.

When the iSCSI protocol as defined in [iSCSI] (i.e. without the iSER enhancements) is intended in the rest of the document, the term "Traditional iSCSI" is used to make the intention clear.

This document obsoletes RFC 5046. See Section 14 for the list of changes from RFC 5046.

2.2 iSCSI/iSER Layering

iSCSI Extensions for RDMA (iSER) is layered between the iSCSI layer and the RCaP layer.

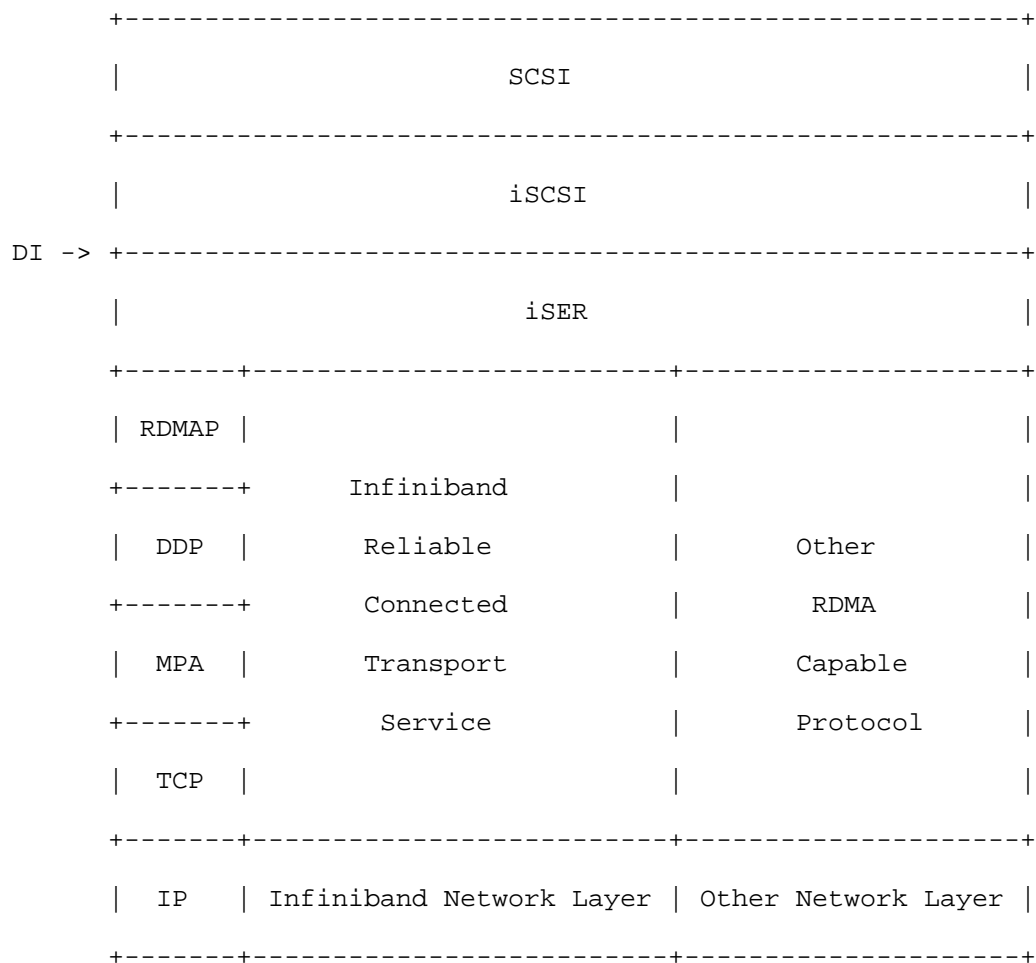


Figure 1 Example of iSCSI/iSER Layering in Full Feature Phase

Figure 1 shows an example of the relationship between SCSI, iSCSI, iSER, and the different RCaP layers. For TCP, the RCaP is iWARP. For Infiniband, the RCaP is the Reliable Connected Transport Service. Note that the iSCSI layer as described here supports the RDMA Extensions as used in iSER.

2.3 Architectural Goals

This section summarizes the architectural goals that guided the design of iSER.

1. Provide an RDMA data transfer model for iSCSI that enables direct in order or out of order data placement of SCSI data into pre-allocated SCSI buffers while maintaining in order data delivery.
2. Not require any major changes to SCSI Architecture Model [SAM5] and SCSI command set standards.
3. Utilize existing iSCSI infrastructure (sometimes referred to as "iSCSI ecosystem") including but not limited to MIB, bootstrapping, negotiation, naming & discovery, and security.
4. Enable a session to operate in the Traditional iSCSI data transfer mode if iSER is not supported by either the initiator or the target (not require iSCSI full feature phase interoperability between an end node operating in Traditional iSCSI mode, and an end node operating in iSER-assisted mode).
5. Allow initiator and target implementations to utilize generic RDMA-Capable Controllers such as RNICs, or implement iSCSI and iSER in software (not require iSCSI or iSER specific assists in the RCaP implementation or RDMA-Capable Controller).
6. Implement a light weight Datamover protocol for iSCSI with minimal state maintenance.

2.4 Protocol Overview

Consistent with the architectural goals stated in section 2.3, the iSER protocol does not require changes in the iSCSI ecosystem or any related SCSI specifications. iSER protocol defines the mapping of iSCSI PDUs to RCaP Messages in such a way that it is entirely feasible to realize iSCSI/iSER implementations that are based on generic RDMA-Capable Controllers. The iSER protocol layer requires minimal state maintenance to assist an iSCSI full feature phase connection, besides being oblivious to the notion of an iSCSI

session. The crucial protocol aspects of iSER may be summarized thus:

1. iSER-assisted mode is negotiated during the iSCSI login in the leading connection for each session, and an entire iSCSI session can only operate in one mode (i.e. a connection in a session cannot operate in iSER-assisted mode if a different connection of the same session is already in full feature phase in the Traditional iSCSI mode).
2. Once in iSER-assisted mode, all iSCSI interactions on that connection use RCaP Messages.
3. A Send Message is used for carrying an iSCSI control-type PDU preceded by an iSER header. See section 7.2 for more details on iSCSI control-type PDUs.
4. RDMA Write, RDMA Read Request, and RDMA Read Response Messages are used for carrying control and all data information associated with the iSCSI data-type PDUs (i.e., SCSI Data-In PDUs and R2T PDUs). iSER does not use SCSI Data-Out PDUs for solicited data, and SCSI Data-Out PDUs for unsolicited data are not treated as iSCSI data-type PDUs by iSER because RDMA is not used. See section 7.1 for more details on iSCSI data-type PDUs.
5. Target drives all data transfer (with the exception of iSCSI unsolicited data) for SCSI writes and SCSI reads, by issuing RDMA Read Requests and RDMA Writes respectively.
6. RCaP is responsible for ensuring data integrity. (For example, iWARP includes a CRC-enhanced framing layer called MPA on top of TCP; and for Infiniband, the CRCs are included in the Reliable Connection mode). For this reason, iSCSI header and data digests are negotiated to "None" for iSCSI/iSER sessions.
7. The iSCSI error recovery hierarchy defined in [iSCSI] is fully supported by iSER. (However, see section 7.3.11 on the handling of SNACK Request PDUs.)
8. iSER requires no changes to iSCSI security and text mode negotiation mechanisms.

Note that Traditional iSCSI implementations may have to be adapted to employ iSER. It is expected that the adaptation when required is likely to be centered around the upper layer interface requirements of iSER (section 3).

2.5 RDMA services and iSER

iSER is designed to work with software and/or hardware protocol stacks providing the protocol services defined in RCaP documents such as [RDMAP], [IB], etc. The following subsections describe the key protocol elements of RCaP services that iSER relies on.

2.5.1 STag

An STag is the identifier of an I/O Buffer unique to an RDMA-Capable Controller that the iSER Layer Advertises to the remote iSCSI/iSER node in order to complete a SCSI I/O.

In iSER, Advertisement is the act of informing the target by the initiator that an I/O Buffer is available at the initiator for RDMA Read or RDMA Write access by the target. The initiator Advertises the I/O Buffer by including the STag and the Base Offset in the header of an iSER Message containing the SCSI Command PDU to the target. The buffer length is as specified in the SCSI Command PDU.

The iSER Layer at the initiator Advertises the STag and the Base Offset for the I/O Buffer of each SCSI I/O to the iSER Layer at the target in the iSER header of a Send Message containing the SCSI Command PDU, unless the I/O can be completely satisfied by unsolicited data alone. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

The iSER Layer at the target provides the STag for the I/O Buffer that is the Data Sink of an RDMA Read Operation (section 2.5.4) to the RCaP layer on the initiator node - i.e. this is completely transparent to the iSER Layer at the initiator.

The iSER layer at the initiator SHOULD invalidate the Advertised STag upon a normal completion of the associated task. The Send with Invalidate Message, if supported by the RCaP layer (e.g., iWARP), can be used for automatic invalidation when it is used to carry the SCSI Response PDU. There are two exceptions to this automatic invalidation - bidirectional commands, and abnormal completion of a command. The iSER Layer at the initiator SHOULD explicitly invalidate the STag in these two cases. That iSER layer MUST check that STag invalidation has occurred whenever receipt of a Send with Invalidate message is the expected means of causing an STag to be invalidated, and MUST perform the STag invalidation if the STag has not already been invalidated (e.g., because a Send message was used instead of Send with Invalidate).

If the Advertised STag is not invalidated as recommended in the foregoing paragraph (e.g., in order to cache the STag for future reuse), the I/O Buffer remains exposed to the network for access by the RCaP. Such an I/O Buffer is capable of being read or written by the RCaP outside the scope of the iSCSI operation for which it was originally established, which has both robustness and security considerations. The robustness considerations are that the system containing the iSER initiator may react poorly to an unexpected modification of its memory. For the security considerations, see Section 11.

2.5.2 Send

Send is the RDMA Operation that is not addressed to an Advertised buffer, and uses Untagged buffers as the message is received.

The iSER Layer at the initiator uses the Send Operation to transmit any iSCSI control-type PDU to the target. As an example, the initiator uses Send Operations to transfer iSER Messages containing SCSI Command PDUs to the iSER Layer at the target.

An iSER layer at the target uses the Send Operation to transmit any iSCSI control-type PDU to the initiator. As an example, the target uses Send Operations to transfer iSER Messages containing SCSI Response PDUs to the iSER Layer at the initiator.

For interoperability, iSER implementations SHOULD accept and correctly process SendSE and SendInvSE messages. However, SendSE and SendInvSE messages are to be regarded as optimizations or enhancements to the basic Send message, and their support may vary by RCaP protocol and specific implementation. In general, these messages SHOULD NOT be used, unless the RCaP requires support for them in all implementations. If these messages are used, the implementation SHOULD be capable of reverting to use of Send in order to work with a receiver that does not support these message. Attempted use of these messages with a peer that does not support them may result in a fatal error that closes the RCaP connection. For example, these messages SHOULD NOT be used with the InfiniBand RCaP because InfiniBand does not require support for them in all cases. New iSER implementations SHOULD use Send (and not SendSE or SendInvSE) unless there are compelling reasons for doing otherwise. Similarly, iSER implementations SHOULD NOT rely on events triggered by SendSE and SendInvSE, as these messages may not be used.

RDMA Write is the RDMA Operation that is used to place data into an Advertised buffer at the Data Sink. The Data Source addresses the Message using an STag and a Tagged Offset that are valid on the Data Sink.

The iSER Layer at the target uses the RDMA Write Operation to transfer the contents of a local I/O Buffer to an Advertised I/O Buffer at the initiator. The iSER Layer at the target uses the RDMA Write to transfer whole or part of the data required to complete a SCSI Read command.

The iSER Layer at the initiator does not employ RDMA Writes.

2.5.4 RDMA Read

RDMA Read is the RDMA Operation that is used to retrieve data from an Advertised buffer at the Data Source. The sender of the RDMA Read Request addresses the Message using an STag and a Tagged Offset that are valid on the Data Source in addition to providing a valid local STag and Tagged Offset that identify the Data Sink.

The iSER Layer at the target uses the RDMA Read Operation to transfer the contents of an Advertised I/O Buffer at the initiator to a local I/O Buffer at the target. The iSER Layer at the target uses the RDMA Read to fetch whole or part of the data required to complete a SCSI Write Command.

The iSER Layer at the initiator does not employ RDMA Reads.

2.6 SCSI Read Overview

The iSER Layer at the initiator receives the SCSI Command PDU from the iSCSI Layer. The iSER Layer at the initiator generates an STag for the I/O Buffer of the SCSI Read and Advertises the buffer by including the STag and the Base Offset as part of the iSER header for the PDU. The iSER Message is transferred to the target using a Send Message. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

The iSER Layer at the target uses one or more RDMA Writes to transfer the data required to complete the SCSI Read.

The iSER Layer at the target uses a Send Message to transfer the SCSI Response PDU back to the iSER Layer at the initiator. The iSER Layer at the initiator invalidates the STag and notifies the iSCSI

Layer of the availability of the SCSI Response PDU. The Send with Invalidate Message, if supported by the RCaP layer (e.g., iWARP), can be used for automatic invalidation of the STag.

2.7 SCSI Write Overview

The iSER Layer at the initiator receives the SCSI Command PDU from the iSCSI Layer. If solicited data transfer is involved, the iSER Layer at the initiator generates an STag for the I/O Buffer of the SCSI Write and Advertises the buffer by including the STag and the Base Offset as part of the iSER header for the PDU. The iSER Message is transferred to the target using a Send Message. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

The iSER Layer at the initiator may optionally send one or more non-immediate unsolicited data PDUs to the target using Send Messages.

If solicited data transfer is involved, the iSER Layer at the target uses one or more RDMA Reads to transfer the data required to complete the SCSI Write.

The iSER Layer at the target uses a Send Message to transfer the SCSI Response PDU back to the iSER Layer at the initiator. The iSER Layer at the initiator invalidates the STag and notifies the iSCSI Layer of the availability of the SCSI Response PDU. The Send with Invalidate Message, if supported by the RCaP layer (e.g., iWARP), can be used for automatic invalidation of the STag.

3 Upper Layer Interface Requirements

This section discusses the upper layer interface requirements in the form of an abstract model of the required interactions between the iSCSI Layer and the iSER Layer. The abstract model used here is derived from the architectural model described in [DA]. [DA] also provides a functional overview of the interactions between the iSCSI Layer and the datamover layer as intended by the Datamover Architecture.

The interface requirements are specified by Operational Primitives. An Operational Primitive is an abstract functional interface procedure between the iSCSI Layer and the iSER Layer that requests one layer to perform a specific action on behalf of the other layer or notifies the other layer of some event. Whenever an Operational Primitive is invoked, the Connection_Handle qualifier is used to identify a particular iSCSI connection. For some Operational Primitives, a Data_Descriptor is used to identify the iSCSI/SCSI data buffer associated with the requested or completed operation.

The abstract model and the Operational Primitives defined in this section facilitate the description of the iSER protocol. In the rest of the iSER specification, the compliance statements related to the use of these Operational Primitives are only for the purpose of the required interactions between the iSCSI Layer and the iSER Layer. Note that the compliance statements related to the Operational Primitives in the rest of this specification only mandate functional equivalence on implementations, but do not put any requirements on the implementation specifics of the interface between the iSCSI Layer and the iSER Layer.

Each Operational Primitive is invoked with a set of qualifiers which specify the information context for performing the specific action being requested of the Operational Primitive. While the qualifiers are required, the method of realizing the qualifiers (e.g., by passing synchronously with invocation, or by retrieving from task context, or by retrieving from shared memory, etc.) is implementation dependent.

3.1 Operational Primitives offered by iSER

The iSER protocol layer MUST support the following Operational Primitives to be used by the iSCSI protocol layer.

3.1.1 Send_Control

Input qualifiers: Connection_Handle, BHS and AHS (if any) of the iSCSI PDU, PDU-specific qualifiers

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request the outbound transfer of an iSCSI control-type PDU (see section 7.2). Qualifiers that only apply for a particular control-type PDU are known as PDU-specific qualifiers, e.g., ImmediateDataSize for a SCSI Write command. For details on PDU-specific qualifiers, see section 7.3. The iSCSI Layer can only invoke the Send_Control Operational Primitive when the connection is in iSER-assisted mode.

3.1.2 Put_Data

Input qualifiers: Connection_Handle, content of a SCSI Data-in PDU header, Data_Descriptor, Notify_Enable

Return results: Not specified

This is used by the iSCSI Layer at the target to request the outbound transfer of data for a SCSI Data-in PDU from the buffer identified by the Data_Descriptor qualifier. The iSCSI Layer can only invoke the Put_Data Operational Primitive when the connection is in iSER-assisted mode.

The Notify_Enable qualifier is used to indicate to the iSER Layer whether or not it should generate an eventual local completion notification to the iSCSI Layer. See section 3.2.2 on Data_Completion_Notify for details.

3.1.3 Get_Data

Input qualifiers: Connection_Handle, content of an R2T PDU, Data_Descriptor, Notify_Enable

Return results: Not specified

This is used by the iSCSI Layer at the target to request the inbound transfer of solicited data requested by an R2T PDU into the buffer identified by the Data_Descriptor qualifier. The iSCSI Layer can only invoke the Get_Data Operational Primitive when the connection is in iSER-assisted mode.

The `Notify_Enable` qualifier is used to indicate to the iSER Layer whether or not it should generate the eventual local completion notification to the iSCSI Layer. See section 3.2.2 on `Data_Completion_Notify` for details.

3.1.4 `Allocate_Connection_Resources`

Input qualifiers: `Connection_Handle`, `Resource_Descriptor` (optional)

Return results: `Status`

This is used by the iSCSI Layers at the initiator and the target to request the allocation of all connection resources necessary to support RCaP for an operational iSCSI/iSER connection. The iSCSI Layer may optionally specify the implementation-specific resource requirements for the iSCSI connection using the `Resource_Descriptor` qualifier.

A return result of `Status=success` means the invocation succeeded, and a return result of `Status=failure` means that the invocation failed. If the invocation is for a `Connection_Handle` for which an earlier invocation succeeded, the request will be ignored by the iSER Layer and the result of `Status=success` will be returned. Only one `Allocate_Connection_Resources` Operational Primitive invocation can be outstanding for a given `Connection_Handle` at any time.

3.1.5 `Deallocate_Connection_Resources`

Input qualifiers: `Connection_Handle`

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request the deallocation of all connection resources that were allocated earlier as a result of a successful invocation of the `Allocate_Connection_Resources` Operational Primitive.

3.1.6 `Enable_Datamover`

Input qualifiers: `Connection_Handle`,
`Transport_Connection_Descriptor`, `Final_Login_Response_PDU` (optional)

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request that iSER-assisted mode be used for the connection. The `Transport_Connection_Descriptor` qualifier is used to identify the specific connection associated with the `Connection_Handle`. The iSCSI layer can only invoke the `Enable_Datamover` Operational Primitive when there was a corresponding prior resource allocation.

The `Final_Login_Response_PDU` input qualifier is applicable only for a target, and contains the final Login Response PDU that concludes the iSCSI Login Phase.

3.1.7 `Connection_Terminate`

Input qualifiers: `Connection_Handle`

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request that a specified iSCSI/iSER connection be terminated and all associated connection and task resources be freed. When this Operational Primitive invocation returns to the iSCSI layer, the iSCSI layer may assume full ownership of all iSCSI-level resources, e.g. I/O Buffers, associated with the connection.

3.1.8 `Notice_Key_Values`

Input qualifiers: `Connection_Handle`, number of keys, list of Key-Value pairs

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request the iSER Layer to take note of the specified Key-Value pairs which were negotiated by the iSCSI peers for the connection.

3.1.9 `Deallocate_Task_Resources`

Input qualifiers: `Connection_Handle`, ITT

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request the deallocation of all RCaP-specific resources allocated by the iSER Layer for the task identified by the ITT qualifier. The iSER Layer may require a certain number of RCaP-specific resources associated with the ITT for each new iSCSI task. In the normal course of execution, these task-level resources in the iSER Layer

are assumed to be transparently allocated on each task initiation and deallocated on the conclusion of each task as appropriate. In exception scenarios where the task does not conclude with a SCSI Response PDU, the iSER Layer needs to be notified of the individual task terminations to aid its task-level resource management. This Operational Primitive is used for this purpose, and is not needed when a SCSI Response PDU normally concludes a task. Note that RCaP-specific task resources are deallocated by the iSER Layer when a SCSI Response PDU normally concludes a task, even if the SCSI Status was not success.

3.2 Operational Primitives used by iSER

The iSER layer MUST use the following Operational Primitives offered by the iSCSI protocol layer when the connection is in iSER-assisted mode.

3.2.1 Control_Notify

Input qualifiers: Connection_Handle, an iSCSI control-type PDU

Return results: Not specified

This is used by the iSER Layers at the initiator and the target to notify the iSCSI Layer of the availability of an inbound iSCSI control-type PDU. A PDU is described as "available" to the iSCSI Layer when the iSER Layer notifies the iSCSI Layer of the reception of that inbound PDU, along with an implementation-specific indication as to where the received PDU is.

3.2.2 Data_Completion_Notify

Input qualifiers: Connection_Handle, ITT, SN

Return results: Not specified

This is used by the iSER Layer to notify the iSCSI Layer of the completion of outbound data transfer that was requested by the iSCSI Layer only if the invocation of the Put_Data Operational Primitive (see section 3.1.2) was qualified with Notify_Enable set. SN refers to the DataSN associated with the SCSI Data-In PDU.

This is used by the iSER Layer to notify the iSCSI Layer of the completion of inbound data transfer that was requested by the iSCSI Layer only if the invocation of the Get_Data Operational Primitive (see section 3.1.3) was qualified with Notify_Enable set. SN refers to the R2TSN associated with the R2T PDU.

3.2.3 Data_ACK_Notify

Input qualifier: Connection_Handle, ITT, DataSN

Return results: Not specified

This is used by the iSER Layer at the target to notify the iSCSI Layer of the arrival of the data acknowledgement (as defined in [iSCSI]) requested earlier by the iSCSI Layer for the outbound data transfer via an invocation of the Put_Data Operational Primitive where the A-bit in the SCSI Data-in PDU is set to 1. See section 7.3.5. DataSN refers to the expected DataSN of the next SCSI Data-in PDU which immediately follows the SCSI Data-in PDU with the A-bit set to which this notification corresponds, with semantics as defined in [iSCSI].

3.2.4 Connection_Terminate_Notify

Input qualifiers: Connection_Handle

Return results: Not specified

This is used by the iSER Layers at the initiator and the target to notify the iSCSI Layer of the unsolicited termination or failure of an iSCSI/iSER connection. The iSER Layer MUST deallocate the connection and task resources associated with the terminated connection before the invocation of this Operational Primitive. Note that the Connection_Terminate_Notify Operational Primitive is not invoked when the termination of the connection was earlier requested by the local iSCSI Layer.

3.3 iSCSI Protocol Usage Requirements

To operate in an iSER-assisted mode, the iSCSI Layers at both the initiator and the target MUST negotiate the RDMAExtensions key (see section 6.3) to "Yes" on the leading connection. If the RDMAExtensions key is not negotiated to "Yes", then iSER-assisted mode MUST NOT be used. If the RDMAExtensions key is negotiated to "Yes" but the invocation of the Allocate_Connection_Resources Operational Primitive to the iSER layer fails, the iSCSI layer MUST fail the iSCSI Login process or terminate the connection as appropriate. See section 10.1.3.1 for details.

If the RDMAExtensions key is negotiated to "Yes", the iSCSI Layer MUST satisfy the following protocol usage requirements from the iSER protocol:

1. The iSCSI Layer at the initiator MUST set ExpDataSN to 0 in Task Management Function Requests for Task Allegiance Reassignment for read/bidirectional commands, so as to cause the target to send all unacknowledged read data.
2. The iSCSI Layer at the target MUST always return the SCSI status in a separate SCSI Response PDU for read commands, i.e., there MUST NOT be a "phase collapse" in concluding a SCSI Read Command.
3. The iSCSI Layers at both the initiator and the target MUST support the keys as defined in section 6 on Login/Text Operational Keys. If used as specified, these keys MUST NOT be answered with NotUnderstood and the semantics as defined MUST be followed for each iSER-assisted connection.
4. The iSCSI Layer at the initiator MUST NOT issue SNACKs for PDUs.

4.1 Interactions with the RCaP Layer

The iSER protocol layer is layered on top of an RCaP layer (see Figure 1) and the following are the key features that are assumed to be supported by any RCaP layer:

- * The RCaP layer supports all basic RDMA operations, including RDMA Write Operation, RDMA Read Operation, and Send Operation.
- * The RCaP layer provides reliable, in-order message delivery and direct data placement.
- * When the iSER Layer initiates an RDMA Read Operation following an RDMA Write Operation on one RCaP Stream, the RDMA Read Response Message processing on the remote node will be started only after the preceding RDMA Write Message payload is placed in the memory of the remote node.
- * The RCaP layer encapsulates a single iSER Message into a single RCaP Message on the Data Source side. The RCaP layer decapsulates the iSER Message before delivering it to the iSER Layer on the Data Sink side.
- * For a RCaP layer that supports the Send with Invalidate Message (e.g., iWARP), when the iSER Layer provides the STag to be remotely invalidated to the RCaP layer for a Send with Invalidate Message, the RCaP layer uses this STag as the STag to be invalidated in the Send with Invalidate Message.
- * The RCaP layer uses the STag and Tagged Offset provided by the iSER Layer for the RDMA Write and RDMA Read Request Messages.
- * When the RCaP layer delivers the content of an RDMA Send Message to the iSER Layer, the RCaP layer provides the length of the RDMA Send message. This ensures that the iSER Layer does not have to carry a length field in the iSER header.
- * When the RCaP layer delivers the Send Message to the iSER Layer, it notifies the iSER Layer with the mechanism provided on that interface.
- * For a RCaP layer that supports the Send with Invalidate Message (e.g., iWARP), when the RCaP layer delivers a Send with Invalidate Message to the iSER Layer, it passes the value of the STag that was invalidated.

- * The RCaP layer propagates all status and error indications to the iSER Layer.
- * For a transport layer that operates in byte stream mode such as TCP, the RCaP implementation supports the enabling of the RDMA mode after Connection establishment and the exchange of Login parameters in byte stream mode. For a transport layer that provides message delivery capability such as [IB], the RCaP implementation supports the use of the messaging capability by the iSCSI Layer directly for the Login phase after connection establishment before enabling iSER-assisted mode. (In the specific example of InfiniBand [IB], the iSCSI Layer uses IB messages to transfer iSCSI PDUs for the Login phase after connection establishment before enabling iSER-assisted mode.)
- * Whenever the iSER Layer terminates the RCaP Stream, the RCaP layer terminates the associated Connection.

4.2 Interactions with the Transport Layer

After the iSER connection is established, the RCaP layer and the underlying transport layer are responsible for maintaining the Connection and reporting to the iSER Layer any Connection failures.

5 Connection Setup and Termination

5.1 iSCSI/iSER Connection Setup

During connection setup, the iSCSI Layer at the initiator is responsible for establishing a connection with the target. After the connection is established, the iSCSI Layers at the initiator and the target enter the Login Phase using the same rules as outlined in [iSCSI]. The connection transitions into the iSCSI full feature phase in iSER-assisted mode following a successful login negotiation between the initiator and the target in which iSER-assisted mode is negotiated and the connection resources necessary to support RCaP have been allocated at both the initiator and the target. The same connection MUST be used for both the iSCSI Login phase and the subsequent iSER-assisted full feature phase.

For a transport layer that operates in byte stream mode such as TCP, the RCaP implementation supports the enabling of the RDMA mode after Connection establishment and the exchange of Login parameters in byte stream mode. For a transport layer that provides message delivery capability such as [IB], the RCaP implementation supports the use of the messaging capability by the iSCSI Layer directly for the Login phase after connection establishment before enabling iSER-assisted mode.

iSER-assisted mode MUST NOT be enabled unless it is negotiated on the leading connection during the LoginOperationalNegotiation Stage of the iSCSI Login Phase. iSER-assisted mode is negotiated using the RDMAExtensions=<boolean-value> key. Both the initiator and the target MUST exchange the RDMAExtensions key with the value set to "Yes" to enable iSER-assisted mode. If both the initiator and the target fail to negotiate the RDMAExtensions key set to "Yes", then the connection MUST continue with the login semantics as defined in [iSCSI]. If the RDMAExtensions key is not negotiated to Yes, then for some RCaP implementation (such as [IB]), the existing connection may need to be torn down and a new connection may need to be established in TCP capable mode. (For InfiniBand this will require an [IPoIB] type connection.)

iSER-assisted mode is defined for a Normal session only and the RDMAExtensions key MUST NOT be negotiated for a Discovery session. Discovery sessions are always conducted using the transport layer as described in [iSCSI].

An iSER enabled node is not required to initiate the RDMAExtensions key exchange if its preference is for the Traditional iSCSI mode. The RDMAExtensions key, if offered, MUST be sent in the first

available Login Response or Login Request PDU in the LoginOperationalNegotiation stage. This is due to the fact that the value of some login parameters might depend on whether iSER-assisted mode is enabled or not.

iSER-assisted mode is a session-wide attribute. If both the initiator and the target negotiated RDMAExtensions="Yes" on the leading connection of a session, then all subsequent connections of the same session MUST enable iSER-assisted mode without having to exchange RDMAExtensions key during the iSCSI Login Phase. Conversely, if both the initiator and the target failed to negotiate RDMAExtensions to "Yes" on the leading connection of a session, then the RDMAExtensions key MUST NOT be negotiated further on any additional subsequent connection of the session.

When the RDMAExtensions key is negotiated to "Yes", the HeaderDigest and the DataDigest keys MUST be negotiated to "None" on all iSCSI/iSER connections participating in that iSCSI session. This is because, for an iSCSI/iSER connection, RCaP is responsible for providing error detection that is at least as good as a 32-bit CRC for all iSER Messages. Furthermore, all SCSI Read data are sent using RDMA Write Messages instead of the SCSI Data-in PDUs, and all solicited SCSI write data are sent using RDMA Read Response Messages instead of the SCSI Data-out PDUs. HeaderDigest and DataDigest which apply to iSCSI PDUs would not be appropriate for RDMA Read and RDMA Write operations used with iSER.

5.1.1 Initiator Behavior

If the outcome of the iSCSI negotiation is to enable iSER-assisted mode, then on the initiator side, prior to sending the Login Request with the T (Transit) bit set to 1 and the NSG (Next Stage) field set to FullFeaturePhase, the iSCSI Layer SHOULD request the iSER Layer to allocate the connection resources necessary to support RCaP by invoking the Allocate_Connection_Resources Operational Primitive. The connection resources required are defined by implementation and are outside the scope of this specification. The iSCSI Layer may invoke the Notice_Key_Values Operational Primitive before invoking the Allocate_Connection_Resources Operational Primitive to request the iSER Layer to take note of the negotiated values of the iSCSI keys for the Connection. The specific keys to be passed in as input qualifiers are implementation dependent. These may include, but are not limited to, MaxOutstandingR2T, ErrorRecoveryLevel, etc.

Among the connection resources allocated at the initiator is the Inbound RDMA Read Queue Depth (IRD). As described in section 9.5.1, R2Ts are transformed by the target into RDMA Read operations. IRD

limits the maximum number of simultaneously incoming outstanding RDMA Read Requests per an RCaP Stream from the target to the initiator. The required value of IRD is outside the scope of the iSER specification. The iSER Layer at the initiator MUST set IRD to 1 or higher if R2Ts are to be used in the connection. However, the iSER Layer at the initiator MAY set IRD to 0 based on implementation configuration which indicates that no R2Ts will be used on that connection. Initially, the iSER-IRD value at the initiator SHOULD be set to the IRD value at the initiator and MUST NOT be more than the IRD value.

On the other hand, the Outbound RDMA Read Queue Depth (ORD) MAY be set to 0 since the iSER Layer at the initiator does not issue RDMA Read Requests to the target.

Failure to allocate the requested connection resources locally results in a login failure and its handling is described in section 10.1.3.1.

The iSER Layer MUST return a success status to the iSCSI Layer in response to the Allocate_Connection_Resources Operational Primitive.

After the target returns the Login Response with the T bit set to 1 and the NSG field set to FullFeaturePhase, and a status class of 0 (Success), the iSCSI Layer MUST invoke the Enable_Datamover Operational Primitive with the following qualifiers. (See section 10.1.4.6 for the case when the status class is not Success.):

- a. Connection_Handle that identifies the iSCSI connection.
- b. Transport_Connection_Descriptor which identifies the specific transport connection associated with the Connection_Handle.

The iSER Layer MUST send the iSER Hello Message as the first iSER Message only if iSERHelloRequired is negotiated to "Yes". See Section 5.1.3 on iSER Hello Exchange.

If the iSCSI Layer on the initiator side allocates the connection resources to support RCaP only after it receives the final Login Response PDU from the target, then it may not be able to handle the number of unexpected iSCSI control-type PDUs (as declared by the MaxOutstandingUnexpectedPDUs key from the initiator) that can be sent by the target before the buffer resources are allocated at the initiator side. In this case the iSERHelloRequired key SHOULD be negotiated to "Yes" so that the initiator can allocate the

connection resources before sending the iSER Hello Message. See section 5.1.3 for more details.

5.1.2 Target Behavior

If the outcome of the iSCSI negotiation is to enable iSER-assisted mode, then on the target side, prior to sending the Login Response with the T (Transit) bit set to 1 and the NSG (Next Stage) field set to FullFeaturePhase, the iSCSI Layer MUST request the iSER Layer to allocate the resources necessary to support RCaP by invoking the `Allocate_Connection_Resources` Operational Primitive. The connection resources required are defined by implementation and are outside the scope of this specification. Optionally, the iSCSI Layer may invoke the `Notice_Key_Values` Operational Primitive before invoking the `Allocate_Connection_Resources` Operational Primitive to request the iSER Layer to take note of the negotiated values of the iSCSI keys for the Connection. The specific keys to be passed in as input qualifiers are implementation dependent. These may include, but not limited to, `MaxOutstandingR2T`, `ErrorRecoveryLevel`, etc.

Premature allocation of RCaP connection resources can expose an iSER target to a resource exhaustion attack on those resources via multiple iSER connections that progress only to the point at which the implementation allocates the RCaP connection resources. The countermeasure for this attack is initiator authentication; the iSCSI Layer MUST NOT request the iSER Layer to allocate the connection resources necessary to support RCaP until the iSCSI layer is sufficiently far along in the iSCSI Login Phase that it is reasonably certain that the peer side is not an attacker. In particular, if the Login Phase includes a `SecurityNegotiation` stage, the iSCSI Layer MUST defer the connection resource allocation (i.e. invoking the `Allocate_Connection_Resources` Operational Primitive) to the `LoginOperationalNegotiation` stage ([iSCSI]) so that the resource allocation occurs after the authentication phase is completed.

Among the connection resources allocated at the target is the Outbound RDMA Read Queue Depth (ORD). As described in section 9.5.1, R2Ts are transformed by the target into RDMA Read operations. The ORD limits the maximum number of simultaneously outstanding RDMA Read Requests per RCaP Stream from the target to the initiator. Initially, the iSER-ORD value at the target SHOULD be set to the ORD value at the target.

On the other hand, the IRD at the target MAY be set to 0 since the iSER Layer at the target does not expect RDMA Read Requests to be issued by the initiator.

Failure to allocate the requested connection resources locally results in a login failure and its handling is described in section 10.1.3.1.

If the iSER Layer at the target is successful in allocating the connection resources necessary to support RCaP, the following events MUST occur in the specified sequence:

1. The iSER Layer MUST return a success status to the iSCSI Layer in response to the Allocate_Connection_Resources Operational Primitive.
2. The iSCSI Layer MUST invoke the Enable_Datamover Operational Primitive with the following qualifiers:
 - a. Connection_Handle that identifies the iSCSI connection.
 - b. Transport_Connection_Descriptor which identifies the specific transport connection associated with the Connection_Handle.
 - c. The final transport layer (e.g. TCP) message containing the Login Response with the T bit set to 1 and the NSG field set to FullFeaturePhase
3. The iSER Layer MUST send the final Login Response PDU in the native transport mode to conclude the iSCSI Login Phase. If the underlying transport is TCP, then the iSER Layer MUST send the final Login Response PDU in byte stream mode.
4. After receiving the iSER Hello Message from the initiator, the iSER Layer MUST respond with the iSER HelloReply Message to be sent as the first iSER Message if iSERHelloRequired is negotiated to "Yes". If the iSER layer receives an iSER Hello Message when iSERHelloRequired is negotiated to "No", then this MUST be treated as an iSER protocol error. See section 5.1.3 on iSER Hello Exchange for more details.

Note: In the above sequence, the operations as described in bullets 3 and 4 MUST be performed atomically for iWARP connections. Failure to do this may result in race conditions.

5.1.3 iSER Hello Exchange

If iSERHelloRequired is negotiated to "Yes", the first iSER Message sent by the iSER Layer at the initiator to the target MUST be the iSER Hello Message. The iSER Hello Message is used by the iSER

Layer at the initiator to declare iSER parameters to the target. See section 9.3 on iSER Header Format for iSER Hello Message. Conversely, if iSERHelloRequired is negotiated to "No", then the iSER Layer at the initiator MUST NOT send an iSER Hello Message.

In response to the iSER Hello Message, the iSER Layer at the target MUST return the iSER HelloReply Message as the first iSER Message sent by the target if iSERHelloRequired is negotiated to "Yes". The iSER HelloReply Message is used by the iSER Layer at the target to declare iSER parameters to the initiator. See section 9.4 on iSER Header Format for iSER HelloReply Message. If the iSER layer receives an iSER Hello Message when iSERHelloRequired is negotiated to "No", then this MUST be treated as an iSER protocol error. See section 10.1.3.4 on iSER Protocol Errors for more details

In the iSER Hello Message, the iSER Layer at the initiator declares the iSER-IRD value to the target.

Upon receiving the iSER Hello Message, the iSER Layer at the target MUST set the iSER-ORD value to the minimum of the iSER-ORD value at the target and the iSER-IRD value declared by the initiator. The iSER Layer at the target MAY adjust (lower) its ORD value to match the iSER-ORD value if the iSER-ORD value is smaller than the ORD value at the target in order to free up the unused resources.

In the iSER HelloReply Message, the iSER Layer at the target declares the iSER-ORD value to the initiator.

Upon receiving the iSER HelloReply Message, the iSER Layer at the initiator MAY adjust (lower) its IRD value to match the iSER-ORD value in order to free up the unused resources, if the iSER-ORD value declared by the target is smaller than the iSER-IRD value declared by the initiator.

It is an iSER level negotiation failure if the iSER parameters declared in the iSER Hello Message by the initiator are unacceptable to the target. This includes the following:

- * The initiator-declared iSER-IRD value is greater than 0 and the target-declared iSER-ORD value is 0.
- * The initiator-supported and the target-supported iSER protocol versions do not overlap.

See section 10.1.3.2 on the handling of the error situation.

An initiator that conforms to [RFC5046] allocates connection resources before sending the Login Request with the T (Transit) bit set to 1 and the NSG (Next Stage) field set to FullFeaturePhase. (For brevity, this is referred to as "early" connection allocation.) The current iSER specification relaxes this requirement to allow an initiator to allocate connection resources after it receives the final Login Response PDU from the target. (For brevity, this is referred to as "late" connection allocation.) An initiator that employs "late" connection allocation may encounter problems (e.g., RCaP connection closure) with a target that sends unexpected iSCSI PDUs immediately upon transitioning to Full Feature Phase, as allowed by the negotiated value of the MaxOutstandingUnexpectedPDUs key. The only way to prevent this situation in full generality is to use iSER Hello Messages, as they enable the initiator to allocate its connection resources before sending its iSER Hello Message. The iSERHelloRequired key is used by the initiator to determine if it is dealing with a target that supports the iSER Hello exchanges. Fortunately, known iSER target implementations do not take full advantage of the number of allowed unexpected PDUs immediately upon transitioning into full feature phase, enabling an initiator workaround that involves a smaller quantity of connection resources prior to full-feature phase, as explained further below.

In the following summary where "late" connection allocation is practised, an initiator that follows [RFC5046] is referred to as an "old" initiator; otherwise it is referred to as a "new" initiator. Similarly, a target that does not support the iSERHelloRequired key (and responds with "NotUnderstood" when negotiating the iSERHelloRequired key) is referred to as an "old" target; otherwise it is referred to as a "new" target. Note that an "old" target can still support the iSER Hello exchanges but this fact is not known by the initiator. A "new" target can also respond with "No" when negotiating the iSERHelloRequired key. In this case its behavior with respect to "late" connection allocation is similar to an "old" target.

A "new" initiator will work fine with a "new" target.

For an "old" initiator and an "old" target, the failure by the initiator to handle the number of unexpected iSCSI control-type PDUs that are sent by the target before the buffer resources are allocated at the initiator can result in the failure of the iSER session caused by closure of the underlying RCaP connection. For the "old" target, there is known implementation that sends one unexpected iSCSI control-type PDU after sending the final Login Response and then waits awhile before sending the next one. This

tends to alleviate somewhat the buffer allocation problem at the initiator.

For a "new" initiator and an "old" target, the failure by the initiator to handle the number of unexpected iSCSI control-type PDUs that are sent by the target before the buffer resources are allocated at the initiator can result in the failure of the iSER session caused by closure of the underlying RCaP connection. A "new" initiator MAY choose to terminate the connection; otherwise it SHOULD do one of the following:

1. Allocate the connection resources before sending the final Login Request PDU.
2. Allocate one or more buffers for receiving unexpected control-type PDUs from the target before sending the final Login Request PDU. This reduces the possibility of the unexpected control-type PDUs causing the RCaP connection to close before the connection resources have been allocated.

For an "old" initiator and a "new" target, if the iSERHelloRequired key is not negotiated, a "new" target MUST still respond with the iSER HelloReply Message when it receives the iSER Hello Message. If the iSERHelloRequired key is negotiated to "No" or "NotUnderstood", a "new" target MAY choose to terminate the connection; otherwise it SHOULD delay sending any unexpected control-type PDUs until one of the following events has occurred:

1. A PDU is received from the initiator after it sends the final Login Response PDU.
2. A system configurable timeout period, say one second, has expired.

5.2 iSCSI/iSER Connection Termination

5.2.1 Normal Connection Termination at the Initiator

The iSCSI Layer at the initiator terminates an iSCSI/iSER connection normally by invoking the Send_Control Operational Primitive qualified with the Logout Request PDU. The iSER Layer at the initiator MUST use a Send Message to send the Logout Request PDU to the target. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP). After the iSER Layer at the initiator receives the Send Message containing the Logout Response PDU from the target, it MUST notify the iSCSI Layer by invoking the

Control_Notify Operational Primitive qualified with the Logout Response PDU.

After the iSCSI logout process is complete, the iSCSI layer at the target is responsible for closing the iSCSI/iSER connection as described in Section 5.2.2. After the RCaP layer at the initiator reports that the Connection has been closed, the iSER Layer at the initiator MUST deallocate all connection and task resources (if any) associated with the connection, invalidate the Local Mappings (if any) before notifying the iSCSI Layer by invoking the Connection_Terminate_Notify Operational Primitive.

5.2.2 Normal Connection Termination at the Target

Upon receiving the Send Message containing the Logout Request PDU, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Control_Notify Operational Primitive qualified with the Logout Request PDU. The iSCSI Layer completes the logout process by invoking the Send_Control Operational Primitive qualified with the Logout Response PDU. The iSER Layer at the target MUST use a Send Message to send the Logout Response PDU to the initiator. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP). After the iSCSI logout process is complete, the iSCSI Layer at the target MUST request the iSER Layer at the target to terminate the RCaP Stream by invoking the Connection_Terminate Operational Primitive.

As part of the termination process, the RCaP layer MUST close the Connection. When the RCaP layer notifies the iSER Layer after the RCaP Stream and the associated Connection are terminated, the iSER Layer MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local and Remote Mappings (if any).

5.2.3 Termination without Logout Request/Response PDUs

5.2.3.1 Connection Termination Initiated by the iSCSI Layer

The Connection_Terminate Operational Primitive MAY be invoked by the iSCSI Layer to request the iSER Layer to terminate the RCaP Stream without having previously exchanged the Logout Request and Logout Response PDUs between the two iSCSI/iSER nodes. As part of the termination process, the RCaP layer will close the Connection. When the RCaP layer notifies the iSER Layer after the RCaP Stream and the associated Connection are terminated, the iSER Layer MUST perform the following actions.

If the `Connection_Terminate` Operational Primitive is invoked by the iSCSI Layer at the target, then the iSER Layer at the target MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local and Remote Mappings (if any).

If the `Connection_Terminate` Operational Primitive is invoked by the iSCSI Layer at the initiator, then the iSER Layer at the initiator MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local Mappings (if any).

5.2.3.2 Connection Termination Notification to the iSCSI Layer

If the iSCSI/iSER connection is terminated without the invocation of `Connection_Terminate` from the iSCSI Layer, the iSER Layer MUST notify the iSCSI Layer that the iSCSI/iSER connection has been terminated by invoking the `Connection_Terminate_Notify` Operational Primitive.

Prior to invoking `Connection_Terminate_Notify`, the iSER Layer at the target MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local and Remote Mappings (if any).

Prior to invoking `Connection_Terminate_Notify`, the iSER Layer at the initiator MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local Mappings (if any).

If the remote iSCSI/iSER node initiated the closing of the Connection (e.g., by sending a TCP FIN or TCP RST), the iSER Layer MUST notify the iSCSI Layer after the RCaP layer reports that the Connection is closed by invoking the `Connection_Terminate_Notify` Operational Primitive.

Another example of a Connection termination without a preceding logout is when the iSCSI Layer at the initiator does an implicit logout (connection reinstatement).

6 Login/Text Operational Keys

Certain iSCSI login/text operational keys have restricted usage in iSER, and additional keys are used to support the iSER protocol functionality. All other keys defined in [iSCSI] and not discussed in this section may be used on iSCSI/iSER connections with the same semantics.

6.1 HeaderDigest and DataDigest

Irrelevant when: RDMAExtensions=Yes

Negotiations resulting in RDMAExtensions=Yes for a session implies HeaderDigest=None and DataDigest=None for all connections in that session and overrides both the default and an explicit setting.

6.2 MaxRecvDataSegmentLength

For an iSCSI connection belonging to a session in which RDMAExtensions=Yes was negotiated on the leading connection of the session, MaxRecvDataSegmentLength need not be declared in the Login Phase, and MUST be ignored if it is declared. Instead InitiatorRecvDataSegmentLength (as described in section 6.5) and TargetRecvDataSegmentLength (as described in section 6.4) keys are negotiated. The values of the local and remote MaxRecvDataSegmentLength are derived from the InitiatorRecvDataSegmentLength and TargetRecvDataSegmentLength keys.

In the full feature phase, the initiator MUST consider the value of its local MaxRecvDataSegmentLength (that it would have declared to the target) as having the value of InitiatorRecvDataSegmentLength, and the value of the remote MaxRecvDataSegmentLength (that would have been declared by the target) as having the value of TargetRecvDataSegmentLength. Similarly, the target MUST consider the value of its local MaxRecvDataSegmentLength (that it would have declared to the initiator) as having the value of TargetRecvDataSegmentLength, and the value of the remote MaxRecvDataSegmentLength (that would have been declared by the initiator) as having the value of InitiatorRecvDataSegmentLength.

Note that RFC 3720 requires that when a target receives a NOP-Out request with a valid Initiator Task Tag, it responds with a NOP-In with the same Initiator Task Tag that was provided in the NOP-Out request. Furthermore, it returns the first MaxRecvDataSegmentLength bytes of the initiator provided Ping Data. Since there is no MaxRecvDataSegmentLength common to the initiator and the target in

iSER, the length of the data sent with the NOP-Out request MUST NOT exceed InitiatorMaxRecvDataSegmentLength.

The MaxRecvDataSegmentLength key is applicable only for iSCSI control-type PDUs.

6.3 RDMAExtensions

Use: LO (leading only)

Senders: Initiator and Target

Scope: SW (session-wide)

RDMAExtensions=<boolean-value>

Irrelevant when: SessionType=Discovery

Default is No

Result function is AND

This key is used by the initiator and the target to negotiate the support for iSER-assisted mode. To enable the use of iSER-assisted mode, both the initiator and the target MUST exchange RDMAExtensions=Yes. iSER-assisted mode MUST NOT be used if either the initiator or the target offers RDMAExtensions=No.

An iSER-enabled node is not required to initiate the RDMAExtensions key exchange if it prefers to operate in the Traditional iSCSI mode. However, if the RDMAExtensions key is to be negotiated, an initiator MUST offer the key in the first Login Request PDU in the LoginOperationalNegotiation stage of the leading connection, and a target MUST offer the key in the first Login Response PDU with which it is allowed to do so (i.e., the first Login Response PDU issued after the first Login Request PDU with the C bit set to 0) in the LoginOperationalNegotiation stage of the leading connection. In response to the offered key=value pair of RDMAExtensions=yes, an initiator MUST respond in the next Login Request PDU with which it is allowed to do so, and a target MUST respond in the next Login Response PDU with which it is allowed to do so.

Negotiating the RDMAExtensions key first enables a node to negotiate the optimal value for other keys. Certain iSCSI keys such as MaxBurstLength, MaxOutstandingR2T, ErrorRecoveryLevel, InitialR2T, ImmediateData, etc., may be negotiated differently depending on

6.4 TargetRecvDataSegmentLength

Use: IO (Initialize only)

Senders: Initiator and Target

Scope: CO (connection-only)

Irrelevant when: RDMAExtensions=No

TargetRecvDataSegmentLength=<numerical-value-512-to-(2**24-1)>

Default is 8192 bytes

Result function is minimum

This key is relevant only for the iSCSI connection of an iSCSI session if RDMAExtensions=Yes was negotiated on the leading connection of the session. It is used by the initiator and the target to negotiate the maximum size of the data segment that an initiator may send to the target in an iSCSI control-type PDU in the full feature phase. For SCSI Command PDUs and SCSI Data-out PDUs containing non-immediate unsolicited data to be sent by the initiator, the initiator MUST send all non-Final PDUs with a data segment size of exactly TargetRecvDataSegmentLength whenever the PDUs constitute a data sequence whose size is larger than TargetRecvDataSegmentLength.

6.5 InitiatorRecvDataSegmentLength

Use: IO (Initialize only)

Senders: Initiator and Target

Scope: CO (connection-only)

Irrelevant when: RDMAExtensions=No

InitiatorRecvDataSegmentLength=<numerical-value-512-to-(2**24-1)>

Default is 8192 bytes

Result function is minimum

This key is relevant only for the iSCSI connection of an iSCSI session if RDMAExtensions=Yes was negotiated on the leading connection of the session. It is used by the initiator and the target to negotiate the maximum size of the data segment that a target may send to the initiator in an iSCSI control-type PDU in the full feature phase.

6.6 OFMarker and IFMarker

Irrelevant when: RDMAExtensions=Yes

Negotiations resulting in RDMAExtensions=Yes for a session implies OFMarker=No and IFMarker=No for all connections in that session and overrides both the default and an explicit setting.

6.7 MaxOutstandingUnexpectedPDUs

Use: LO (leading only), Declarative

Senders: Initiator and Target

Scope: SW (session-wide)

Irrelevant when: RDMAExtensions=No

MaxOutstandingUnexpectedPDUs=<numerical-value-from-2-to-(2**32-1) | 0>

Default is 0

This key is used by the initiator and the target to declare the maximum number of outstanding "unexpected" iSCSI control-type PDUs that it can receive in the full feature phase. It is intended to allow the receiving side to determine the amount of buffer resources needed beyond the normal flow control mechanism available in iSCSI. An initiator or target should select a value such that it would not impose an unnecessary constraint on the iSCSI Layer under normal circumstances. The value of 0 is defined to indicate that the declarer has no limit on the maximum number of outstanding "unexpected" iSCSI control-type PDUs that it can receive. See sections 8.1.1 and 8.1.2 for the usage of this key. Note that iSER Hello and HelloReply Messages are not iSCSI control-type PDUs and are not affected by this key.

For interoperability with implementations based on [RFC5046], this key SHOULD be negotiated because the default value of 0 in [RFC5046]

6.8 MaxAHSLength

Use: LO (leading only), Declarative

Senders: Initiator and Target

Scope: SW (session-wide)

Irrelevant when: RDMAExtensions=No

MaxAHSLength=<numerical-value-from-2-to-(2**32-1) | 0>

Default is 256

This key is used by the initiator and target to declare the maximum size of AHS in an iSCSI control-type PDU that it can receive in the full feature phase. It is intended to allow the receiving side to determine the amount of resources needed for receive buffering. An initiator or target should select a value such that it would not impose an unnecessary constraint on the iSCSI Layer under normal circumstances. The value of 0 is defined to indicate that the declarer has no limit on the maximum size of AHS in iSCSI control-type PDUs that it can receive.

For interoperability with implementations based on [RFC5046], an initiator or target MAY terminate the connection if it anticipates MaxAHSLength to be greater than 256 and the key is not understood by its peer.

6.9 TaggedBufferForSolicitedDataOnly

Use: LO (leading only), Declarative

Senders: Initiator

Scope: SW (session-wide)

RDMAExtensions=<boolean-value>

Irrelevant when: RDMAExtensions=No

Default is No

This key is used by the initiator to declare to the target the usage of the Write Base Offset in the iSER header of an iSCSI control-type PDU. When set to No, the Base Offset is associated with an I/O buffer that contains all the write data, including both unsolicited and solicited data. When set to Yes, the Base Offset is associated with an I/O buffer that only contains solicited data.

6.10 iSERHelloRequired

Use: LO (leading only), Declarative

Senders: Initiator

Scope: SW (session-wide)

RDMAExtensions=<boolean-value>

Irrelevant when: RDMAExtensions=No

Default is No

This key is relevant only for the iSCSI connection of an iSCSI session if RDMAExtensions=Yes was negotiated on the leading connection of the session. It is used by the initiator to declare to the target if the iSER Hello Exchange is required. When set to Yes, the iSER layers MUST perform the iSER Hello Exchange as described in 5.1.3. When set to No, the iSER layers MUST NOT perform the iSER Hello Exchange.

7 iSCSI PDU Considerations

When a connection is in the iSER-assisted mode, two types of message transfers are allowed between the iSCSI Layer at the initiator and the iSCSI Layer at the target. These are known as the iSCSI data-type PDUs and the iSCSI control-type PDUs and these terms are described in the following sections.

7.1 iSCSI Data-Type PDU

An iSCSI data-type PDU is defined as an iSCSI PDU that causes data transfer, transparent to the remote iSCSI layer, to take place between the peer iSCSI nodes in the full feature phase of an iSCSI/iSER connection. An iSCSI data-type PDU, when requested for transmission by the iSCSI Layer in the sending node, results in the data being transferred without the participation of the iSCSI Layers at the sending and the receiving nodes. This is due to the fact that the PDU itself is not delivered as-is to the iSCSI Layer in the receiving node. Instead, the data transfer operations are transformed into the appropriate RDMA operations which are handled by the RDMA-Capable Controller. The set of iSCSI data-type PDUs consists of SCSI Data-in PDUs and R2T PDUs.

If the invocation of the Operational Primitive by the iSCSI Layer to request the iSER Layer to process an iSCSI data-type PDU is qualified with `Notify_Enable` set, then upon completing the RDMA operation, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the `Data_Completion_Notify` Operational Primitive qualified with `ITT` and `SN`. There is no data completion notification at the initiator since the RDMA operations are completely handled by the RDMA-Capable Controller at the initiator and the iSER Layer at the initiator is not involved with the data transfer associated with iSCSI data-type PDUs.

If the invocation of the Operational Primitive by the iSCSI Layer to request the iSER Layer to process an iSCSI data-type PDU is qualified with `Notify_Enable` cleared, then upon completing the RDMA operation, the iSER Layer at the target MUST NOT notify the iSCSI Layer at the target and MUST NOT invoke the `Data_Completion_Notify` Operational Primitive.

If an operation associated with an iSCSI data-type PDU fails for any reason, the contents of the Data Sink buffers associated with the operation are considered indeterminate.

7.2 iSCSI Control-Type PDU

Any iSCSI PDU that is not an iSCSI data-type PDU and also not a SCSI Data-out PDU carrying solicited data is defined as an iSCSI control-type PDU. The iSCSI Layer invokes the Send_Control Operational Primitive to request the iSER Layer to process an iSCSI control-type PDU. iSCSI control-type PDUs are transferred using Send Messages of RCaP. Specifically, it is to be noted that SCSI Data-Out PDUs carrying unsolicited data are defined as iSCSI control-type PDUs. See section 7.3.4 on the treatment of SCSI Data-out PDUs.

When the iSER Layer receives an iSCSI control-type PDU, it MUST notify the iSCSI Layer by invoking the Control_Notify Operational Primitive qualified with the iSCSI control-type PDU.

7.3 iSCSI PDUs

This section describes the handling of each of the iSCSI PDU types by the iSER Layer. The iSCSI Layer requests the iSER Layer to process the iSCSI PDU by invoking the appropriate Operational Primitive. A Connection_Handle MUST qualify each of these invocations. In addition, BHS and the optional AHS of the iSCSI PDU as defined in [iSCSI] MUST qualify each of the invocations. The qualifying Connection_Handle, the BHS and the AHS are not explicitly listed in the subsequent sections.

7.3.1 SCSI Command

Type: control-type PDU

PDU-specific qualifiers (for SCSI Write or bidirectional command): ImmediateDataSize, UnsolicitedDataSize, DataDescriptorOut

PDU-specific qualifiers (for SCSI Read or bidirectional command): DataDescriptorIn

The iSER Layer at the initiator MUST send the SCSI command in a Send Message to the target. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

For a SCSI Write or bidirectional command, the iSCSI Layer at the initiator MUST invoke the Send_Control Operational Primitive as follows:

- * If there is immediate data to be transferred for the SCSI write or bidirectional command, the qualifier ImmediateDataSize MUST be

used to define the number of bytes of immediate unsolicited data to be sent with the write or bidirectional command, and the qualifier `DataDescriptorOut` MUST be used to define the initiator's I/O Buffer containing the SCSI Write data.

- * If there is unsolicited data to be transferred for the SCSI Write or bidirectional command, the qualifier `UnsolicitedDataSize` MUST be used to define the number of bytes of immediate and non-immediate unsolicited data for the command. The iSCSI Layer will issue one or more SCSI Data-out PDUs for the non-immediate unsolicited data. See Section 7.3.4 on SCSI Data-out.
- * If there is solicited data to be transferred for the SCSI Write or bidirectional command, as indicated by the Expected Data Transfer Length in the SCSI Command PDU exceeding the value of `UnsolicitedDataSize`, the iSER Layer at the initiator MUST do the following:
 - a. It MUST allocate a Write STag for the I/O Buffer defined by the qualifier `DataDescriptorOut`. `DataDescriptorOut` describes the I/O buffer starting with the immediate unsolicited data (if any), followed by the non-immediate unsolicited data (if any) and solicited data. When `TaggedBufferForSolicitedDataOnly` is negotiated to No, the Base Offset is associated with this I/O Buffer. When `TaggedBufferForSolicitedDataOnly` is negotiated to Yes, the Base Offset is associated with an I/O Buffer that contains only solicited data.
 - b. It MUST establish a Local Mapping that associates the Initiator Task Tag (ITT) to the Write STag.
 - c. It MUST Advertise the Write STag and the Base Offset to the target by sending them in the iSER header of the iSER Message (the payload of the Send Message of RCaP) containing the SCSI Write or bidirectional command PDU. The `SendSE` Message should be used if supported by the RCaP layer (e.g., iWARP). See section 9.2 on iSER Header Format for iSCSI Control-Type PDU.

For a SCSI Read or bidirectional command, the iSCSI Layer at the initiator MUST invoke the `Send_Control` Operational Primitive qualified with `DataDescriptorIn` which defines the initiator's I/O Buffer for receiving the SCSI Read data. The iSER Layer at the initiator MUST do the following:

- a. It MUST allocate a Read STag for the I/O Buffer and note the Base Offset for this I/O Buffer.
- b. It MUST establish a Local Mapping that associates the Initiator Task Tag (ITT) to the Read STag.
- c. It MUST Advertise the Read STag and the Base Offset to the target by sending them in the iSER header of the iSER Message (the payload of the Send Message of RCaP) containing the SCSI Read or bidirectional command PDU. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP). See section 9.2 on iSER Header Format for iSCSI Control-Type PDU.

If the amount of unsolicited data to be transferred in a SCSI Command exceeds TargetRecvDataSegmentLength, then the iSCSI Layer at the initiator MUST segment the data into multiple iSCSI control-type PDUs, with the data segment length in all PDUs generated except the last one having exactly the size TargetRecvDataSegmentLength. The data segment length of the last iSCSI control-type PDU carrying the unsolicited data can be up to TargetRecvDataSegmentLength.

When the iSER Layer at the target receives the SCSI Command, it MUST establish a Remote Mapping that associates the ITT to the Base Offset(s) and the Advertised STag(s) in the iSER header. The Write STag is used by the iSER Layer at the target in handling the data transfer associated with the R2T PDU(s) as described in section 7.3.6. The Read STag is used in handling the SCSI Data-in PDU(s) from the iSCSI Layer at the target as described in section 7.3.5.

7.3.2 SCSI Response

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorStatus

The iSCSI Layer at the target MUST invoke the Send_Control Operational Primitive qualified with DataDescriptorStatus which defines the buffer containing the sense and response information. The iSCSI Layer at the target MUST always return the SCSI status for a SCSI command in a separate SCSI Response PDU. "Phase collapse" for transferring SCSI status in a SCSI Data-in PDU MUST NOT be used. The iSER Layer at the target sends the SCSI Response PDU according to the following rules:

- * If no STags were Advertised by the initiator in the iSER Message containing the SCSI command PDU, then the iSER Layer at the

target MUST send a Send Message containing the SCSI Response PDU. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

- * If the initiator Advertised a Read STag in the iSER Message containing the SCSI Command PDU, then the iSER Layer at the target MUST send a Send Message containing the SCSI Response PDU. The header of the Send Message MUST carry the Read STag to be invalidated at the initiator. The Send with Invalidate Message, if supported by the RCaP layer (e.g., iWARP), can be used for the automatic invalidation of the STag.
- * If the initiator Advertised only the Write STag in the iSER Message containing the SCSI command PDU, then the iSER Layer at the target MUST send a Send Message containing the SCSI Response PDU. The header of the Send Message MUST carry the Write STag to be invalidated at the initiator. The Send with Invalidate Message, if supported by the RCaP layer (e.g., iWARP), can be used for the automatic invalidation of the STag.

When the iSCSI Layer at the target invokes the Send_Control Operational Primitive to send the SCSI Response PDU, the iSER Layer at the target MUST invalidate the Remote Mapping before transferring the SCSI Response PDU to the initiator.

Upon receiving a Send Message containing the SCSI Response PDU from the target, the iSER layer at the initiator MUST invalidate the STag(s) specified in the header. (If a Send with Invalidate Message is supported by the RCaP layer (e.g., iWARP) and is used to carry the SCSI Response PDU, the RCaP layer at the initiator will invalidate the STag. The iSER Layer at the initiator MUST ensure that the correct STag is invalidated. If both the Read and the Write STags were Advertised earlier by the initiator, then the iSER Layer at the initiator MUST explicitly invalidate the Write STag upon receiving the Send with Invalidate Message because the header of the Send with Invalidate Message can only carry one STag (in this case the Read STag) to be invalidated.)

The iSER Layer at the initiator MUST ensure the invalidation of the STag(s) used in a command before notifying the iSCSI Layer at the initiator by invoking the Control_Notify Operational Primitive qualified with the SCSI Response. This precludes the possibility of using the STag(s) after the completion of the command thereby causing data corruption.

When the iSER Layer at the initiator receives a Send Message containing the SCSI Response PDU, it SHOULD invalidate the Local

Mapping. The iSER Layer MUST ensure that all local STag(s) associated with the ITT are invalidated before notifying the iSCSI Layer of the SCSI Response PDU by invoking the Control_Notify Operational Primitive qualified with the SCSI Response PDU.

7.3.3 Task Management Function Request/Response

Type: control-type PDU

PDU-specific qualifiers (for TMF Request): DataDescriptorOut, DataDescriptorIn

The iSER Layer MUST use a Send Message to send the Task Management Function Request/Response PDU. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

For the Task Management Function Request with the TASK REASSIGN function, the iSER Layer at the initiator MUST do the following:

- * It MUST use the ITT as specified in the Referenced Task Tag from the Task Management Function Request PDU to locate the existing STags (if any) in the Local Mappings.
- * It MUST invalidate the existing STags (if any) and the Local Mappings.
- * It MUST allocate a Read STag for the I/O Buffer and note the Base Offset associated with the I/O Buffer as defined by the qualifier DataDescriptorIn if the Send_Control Operational Primitive invocation is qualified with DataDescriptorIn.
- * It MUST allocate a Write STag for the I/O Buffer and note the Base Offset associated with the I/O Buffer as defined by the qualifier DataDescriptorOut if the Send_Control Operational Primitive invocation is qualified with DataDescriptorOut.
- * If STags are allocated, it MUST establish new Local Mapping(s) that associate the ITT to the allocated STag(s).
- * It MUST Advertise the STags and the Base Offsets, if allocated, to the target in the iSER header of the Send Message carrying the iSCSI PDU, as described in section 9.2. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

For the Task Management Function Request with the TASK REASSIGN function for a SCSI Read or bidirectional command, the iSCSI Layer at the initiator MUST set ExpDataSN to 0 since the data transfer and

acknowledgements happen transparently to the iSCSI Layer at the initiator. This provides the flexibility to the iSCSI Layer at the target to request transmission of only the unacknowledged data as specified in [iSCSI].

When the iSER Layer at the target receives the Task Management Function Request with the TASK REASSIGN function, it MUST do the following:

- * It MUST use the ITT as specified in the Referenced Task Tag from the Task Management Function Request PDU to locate the Local and Remote Mappings (if any).
- * It MUST invalidate the local STags (if any) associated with the ITT.
- * It MUST replace the Base Offset(s) and the Advertised STag(s) in the Remote Mapping with the Base Offset(s) and the Advertised STag(s) in the iSER header. The Write STag is used in the handling of the R2T PDU(s) from the iSCSI Layer at the target as described in section 7.3.6. The Read STag is used in the handling of the SCSI Data-in PDU(s) from the iSCSI Layer at the target as described in section 7.3.5.

7.3.4 SCSI Data-out

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorOut

The iSCSI Layer at the initiator MUST invoke the Send_Control Operational Primitive qualified with DataDescriptorOut which defines the initiator's I/O Buffer containing unsolicited SCSI Write data.

If the amount of unsolicited data to be transferred as SCSI Data-out exceeds TargetRecvDataSegmentLength, then the iSCSI Layer at the initiator MUST segment the data into multiple iSCSI control-type PDUs, with the DataSegmentLength having the value of TargetRecvDataSegmentLength in all PDUs generated except the last one. The DataSegmentLength of the last iSCSI control-type PDU carrying the unsolicited data can be up to TargetRecvDataSegmentLength. The iSCSI Layer at the target MUST perform the reassembly function for the unsolicited data.

For unsolicited data, the iSER Layer at the initiator MUST use a Send Message to send the SCSI Data-out PDU. If the F bit is set to

1, the SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

Note that for solicited data, the SCSI Data-out PDUs are not used since R2T PDUs are not delivered to the iSCSI layer at the initiator; instead R2T PDUs are transformed by the iSER layer at the target into RDMA Read operations. (See section 7.3.6.)

7.3.5 SCSI Data-in

Type: data-type PDU

PDU-specific qualifiers: DataDescriptorIn

When the iSCSI Layer at the target is ready to return the SCSI Read data to the initiator, it MUST invoke the Put_Data Operational Primitive qualified with DataDescriptorIn which defines the SCSI Data-in buffer. See section 7.1 on the general requirement on the handling of iSCSI data-type PDUs. SCSI Data-in PDU(s) are used in SCSI Read data transfer as described in section 9.5.2.

The iSER Layer at the target MUST do the following for each invocation of the Put_Data Operational Primitive:

1. It MUST use the ITT in the SCSI Data-in PDU to locate the remote Read STag and the Base Offset in the Remote Mapping. The Remote Mapping was established earlier by the iSER Layer at the target when the SCSI Read Command was received from the initiator.
2. It MUST generate and send an RDMA Write Message containing the read data to the initiator.
 - a. It MUST use the remote Read STag as the Data Sink STag of the RDMA Write Message.
 - b. It MUST add the Buffer Offset from the SCSI Data-in PDU to the Base Offset from the Remote Mapping as the Data Sink Tagged Offset of the RDMA Write Message.
 - c. It MUST use DataSegmentLength from the SCSI Data-in PDU to determine the amount of data to be sent in the RDMA Write Message.
3. It MUST associate DataSN and ITT from the SCSI Data-in PDU with the RDMA Write operation. If the Put_Data Operational Primitive invocation was qualified with Notify_Enable set, then when the iSER Layer at the target receives a completion from the RCaP

layer for the RDMA Write Message, the iSER Layer at the target MUST notify the iSCSI Layer by invoking the Data_Completion_Notify Operational Primitive qualified with DataSN and ITT. Conversely, if the Put_Data Operational Primitive invocation was qualified with Notify_Enable cleared, then the iSER Layer at the target MUST NOT notify the iSCSI Layer on completion and MUST NOT invoke the Data_Completion_Notify Operational Primitive.

When the A-bit is set to 1 in the SCSI Data-in PDU, the iSER Layer at the target MUST notify the iSCSI Layer at the target when the data transfer is complete at the initiator. To perform this additional function, the iSER Layer at the target can take advantage of the operational ErrorRecoveryLevel if previously disclosed by the iSCSI Layer via an earlier invocation of the Notice_Key_Values Operational Primitive. There are two approaches that can be taken:

1. If the iSER Layer at the target knows that the operational ErrorRecoveryLevel is 2, or if the iSER Layer at the target does not know the operational ErrorRecoveryLevel, then the iSER Layer at the target MUST issue a zero-length RDMA Read Request Message following the RDMA Write Message. When the iSER Layer at the target receives a completion for the RDMA Read Request Message from the RCaP layer, implying that the RDMA-Capable Controller at the initiator has completed processing the RDMA Write Message due to the completion ordering semantics of RCaP, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Data_Ack_Notify Operational Primitive qualified with ITT and DataSN (see section 3.2.3).
2. If the iSER Layer at the target knows that the operational ErrorRecoveryLevel is 1, then the iSER Layer at the target MUST do one of the following:
 - a. It MUST notify the iSCSI Layer at the target by invoking the Data_Ack_Notify Operational Primitive qualified with ITT and DataSN (see section 3.2.3) when it receives the local completion from the RCaP layer for the RDMA Write Message. This is allowed since digest errors do not occur in iSER (see section 10.1.4.2) and a CRC error will cause the connection to be terminated and the task to be terminated anyway. The local RDMA Write completion from the RCaP layer guarantees that the RCaP layer will not access the I/O Buffer again to transfer the data associated with that RDMA Write operation.

- b. Alternatively, it MUST use the same procedure for handling the data transfer completion at the initiator as for ErrorRecoveryLevel 2.

It should be noted that the iSCSI Layer at the target cannot set the A-bit to 1 if the ErrorRecoveryLevel=0.

SCSI status MUST always be returned in a separate SCSI Response PDU. The S bit in the SCSI Data-in PDU MUST always be set to 0. There MUST NOT be a "phase collapse" in the SCSI Data-in PDU.

Since the RDMA Write Message only transfers the data portion of the SCSI Data-in PDU but not the control information in the header, such as ExpCmdSN, if timely updates of such information is crucial, the iSCSI Layer at the initiator MAY issue NOP-Out PDUs to request the iSCSI Layer at the target to respond with the information using NOP-In PDUs.

7.3.6 Ready To Transfer (R2T)

Type: data-type PDU

PDU-specific qualifiers: DataDescriptorOut

In order to send an R2T PDU, the iSCSI Layer at the target MUST invoke the Get_Data Operational Primitive qualified with DataDescriptorOut which defines the I/O Buffer for receiving the SCSI Write data from the initiator. See section 7.1 on the general requirements on the handling of iSCSI data-type PDUs.

The iSER Layer at the target MUST do the following for each invocation of the Get_Data Operational Primitive:

1. It MUST ensure a valid local STag for the I/O Buffer and a valid Local Mapping. This may involve allocating a valid local STag and establishing a Local Mapping.
2. It MUST use the ITT in the R2T to locate the remote Write STag and the Base Offset in the Remote Mapping. The Remote Mapping was established earlier by the iSER Layer at the target when the iSER Message containing the Advertised Write STag, the Base Offset and the SCSI Command PDU for a SCSI Write or bidirectional command was received from the initiator.
3. If the iSER-ORD value at the target is set to 0, the iSER Layer at the target MUST terminate the connection and free up the resources associated with the connection (as described in 5.2.3)

if it received the R2T PDU from the iSCSI Layer at the target. Upon termination of the connection, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Connection Terminate Notify Operational Primitive.

4. If the iSER-ORD value at the target is set to greater than 0, the iSER Layer at the target MUST transform the R2T PDU into an RDMA Read Request Message. While transforming the R2T PDU, the iSER Layer at the target MUST ensure that the number of outstanding RDMA Read Request Messages does not exceed iSER-ORD value. To transform the R2T PDU, the iSER Layer at the target:
 - a. MUST derive the local STag and local Tagged Offset from the DataDescriptorOut that qualified the Get_Data invocation.
 - b. MUST use the local STag as the Data Sink STag of the RDMA Read Request Message.
 - c. MUST use the local Tagged Offset as the Data Sink Tagged Offset of the RDMA Read Request Message.
 - d. MUST use the Desired Data Transfer Length from the R2T PDU as the RDMA Read Message Size of the RDMA Read Request Message.
 - e. MUST use the remote Write STag as the Data Source STag of the RDMA Read Request Message.
 - f. MUST add the Buffer Offset from the R2T PDU to the Base Offset from the Remote Mapping as the Data Source Tagged Offset of the RDMA Read Request Message.
5. It MUST associate R2TSN and ITT from the R2T PDU with the RDMA Read operation. If the Get_Data Operational Primitive invocation was qualified with Notify_Enable set, then when the iSER Layer at the target receives a completion from the RCaP layer for the RDMA Read operation, the iSER Layer at the target MUST notify the iSCSI Layer by invoking the Data_Completion_Notify Operational Primitive qualified with R2TSN and ITT. Conversely, if the Get_Data Operational Primitive invocation was qualified with Notify_Enable cleared, then the iSER Layer at the target MUST NOT notify the iSCSI Layer on completion and MUST NOT invoke the Data_Completion_Notify Operational Primitive.

When the RCaP layer at the initiator receives a valid RDMA Read Request Message, it will return an RDMA Read Response Message

containing the solicited write data to the target. When the RCaP layer at target receives the RDMA Read Response Message from the initiator, it will place the solicited data in the I/O Buffer referenced by the Data Sink STag in the RDMA Read Response Message.

Since the RDMA Read Request Message from the target does not transfer the control information in the R2T PDU such as ExpCmdSN, if timely updates of such information is crucial, the iSCSI Layer at the initiator MAY issue NOP-Out PDUs to request the iSCSI Layer at the target to respond with the information using NOP-In PDUs.

Similarly, since the RDMA Read Response Message from the initiator only transfers the data but not the control information normally found in the SCSI Data-out PDU, such as ExpStatSN, if timely updates of such information is crucial, the iSCSI Layer at the target MAY issue NOP-In PDUs to request the iSCSI Layer at the initiator to respond with the information using NOP-Out PDUs.

7.3.7 Asynchronous Message

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorSense

The iSCSI Layer MUST invoke the Send_Control Operational Primitive qualified with DataDescriptorSense which defines the buffer containing the sense and iSCSI event information. The iSER Layer MUST use a Send Message to send the Asynchronous Message PDU. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

7.3.8 Text Request & Text Response

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorTextOut (for Text Request), DataDescriptorIn (for Text Response)

The iSCSI Layer MUST invoke the Send_Control Operational Primitive qualified with DataDescriptorTextOut (or DataDescriptorIn) which defines the Text Request (or Text Response) buffer. The iSER Layer MUST use Send Messages to send the Text Request (or Text Response PDUs). The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

7.3.9 Login Request & Login Response

During the login negotiation, the iSCSI Layer interacts with the transport layer directly and the iSER Layer is not involved. See section 5.1 on iSCSI/iSER Connection Setup. If the underlying transport is TCP, the Login Request PDUs and the Login Response PDUs are exchanged when the connection between the initiator and the target is still in the byte stream mode.

The iSCSI Layer MUST NOT send a Login Request (or a Login Response) PDU during the full feature phase. A Login Request (or a Login Response) PDU, if used, MUST be treated as an iSCSI protocol error. The iSER Layer MAY reject such a PDU from the iSCSI Layer with an appropriate error code. If a Login Request PDU is received by the iSCSI Layer at the target, it MUST respond with a Reject PDU with a reason code of "protocol error".

7.3.10 Logout Request & Logout Response

Type: control-type PDU

PDU-specific qualifiers: None

The iSER Layer MUST use a Send Message to send the Logout Request or Logout Response PDU. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP). Section 5.2.1 and 5.2.2 describe the handling of the Logout Request and the Logout Response at the initiator and the target and the interactions between the initiator and the target to terminate a connection.

7.3.11 SNACK Request

Since HeaderDigest and DataDigest must be negotiated to "None", there are no digest errors when the connection is in iSER-assisted mode. Also since RCaP delivers all messages in the order they were sent, there are no sequence errors when the connection is in iSER-assisted mode. Therefore the iSCSI Layer MUST NOT send SNACK Request PDUs. A SNACK Request PDU, if used, MUST be treated as an iSCSI protocol error. The iSER Layer MAY reject such a PDU from the iSCSI Layer with an appropriate error code. If a SNACK Request PDU is received by the iSCSI Layer at the target, it MUST respond with a Reject PDU with a reason code of "protocol error".

7.3.12 Reject

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorReject

The iSCSI Layer MUST invoke the Send_Control Operational Primitive qualified with DataDescriptorReject which defines the Reject buffer. The iSER Layer MUST use a Send Message to send the Reject PDU. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

7.3.13 NOP-Out & NOP-In

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorNOPOut (for NOP-Out),
DataDescriptorNOPIn (for NOP-In)

The iSCSI Layer MUST invoke the Send_Control Operational Primitive qualified with DataDescriptorNOPOut (or DataDescriptorNOPIn) which defines the Ping (or Return Ping) data buffer. The iSER Layer MUST use Send Messages to send the NOP-Out (or NOP-In) PDU. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP).

8.1 Flow Control for RDMA Send Messages

Send Messages in RCaP are used by the iSER Layer to transfer iSCSI control-type PDUs. Each Send Message in RCaP consumes an Untagged Buffer at the Data Sink. However, neither the RCaP layer nor the iSER Layer provides an explicit flow control mechanism for the Send Messages. Therefore, the iSER Layer SHOULD provision enough Untagged buffers for handling incoming Send Messages to prevent buffer exhaustion at the RCaP layer. If buffer exhaustion occurs, it may result in the termination of the connection.

An implementation may choose to satisfy the buffer requirement by using a common buffer pool shared across multiple connections, with usage limits on a per connection basis and usage limits on the buffer pool itself. In such an implementation, exceeding the buffer usage limit for a connection or the buffer pool itself may trigger interventions from the iSER Layer to replenish the buffer pool and/or to isolate the connection causing the problem.

iSER also provides the MaxOutstandingUnexpectedPDUs key to be used by the initiator and the target to declare the maximum number of outstanding "unexpected" control-type PDUs that it can receive. It is intended to allow the receiving side to determine the amount of buffer resources needed beyond the normal flow control mechanism available in iSCSI.

The buffer resources required at both the initiator and the target as a result of control-type PDUs sent by the initiator is described in section 8.1.1. The buffer resources required at both the initiator and target as a result of control-type PDUs sent by the target is described in section 8.1.2.

8.1.1 Flow Control for Control-Type PDUs from the Initiator

The control-type PDUs that can be sent by an initiator to a target can be grouped into the following categories:

1. Regulated: Control-type PDUs in this category are regulated by the iSCSI CmdSN window mechanism and the immediate flag is not set.
2. Unregulated but Expected: Control-type PDUs in this category are not regulated by the iSCSI CmdSN window mechanism but are expected by the target.

3. Unregulated and Unexpected: Control-type PDUs in this category are not regulated by the iSCSI CmdSN window mechanism and are "unexpected" by the target.

8.1.1.1 Control-Type PDUs from the Initiator in the Regulated Category

Control-type PDUs that can be sent by the initiator in this category are regulated by the iSCSI CmdSN window mechanism and the immediate flag is not set.

The queuing capacity required of the iSCSI layer at the target is described in section 4.2.2.1 of [iSCSI]. For each of the control-type PDUs that can be sent by the initiator in this category, the initiator MUST provision for the buffer resources required for the corresponding control-type PDU sent as a response from the target. The following is a list of the PDUs that can be sent by the initiator and the PDUs that are sent by the target in response:

- a. When an initiator sends a SCSI Command PDU, it expects a SCSI Response PDU from the target.
- b. When the initiator sends a Task Management Function Request PDU, it expects a Task Management Function Response PDU from the target.
- c. When the initiator sends a Text Request PDU, it expects a Text Response PDU from the target.
- d. When the initiator sends a Logout Request PDU, it expects a Logout Response PDU from the target.
- e. When the initiator sends a NOP-Out PDU as a ping request with ITT != 0xffffffff and TTT = 0xffffffff, it expects a NOP-In PDU from the target with the same ITT and TTT as in the ping request.

The response from the target for any of the PDUs enumerated here may alternatively be in the form of a Reject PDU sent instead before the task is active, as described in section 7.3 of [iSCSI].

8.1.1.2 Control-Type PDUs from the Initiator in the Unregulated but Expected Category

For the control-type PDUs in the Unregulated but Expected category, the amount of buffering resources required at the target can be predetermined. The following is a list of the PDUs in this category:

- a. SCSI Data-out PDUs are used by the initiator to send unsolicited data. The amount of buffer resources required by the target can be determined using FirstBurstLength. Note that SCSI Data-out PDUs are not used for solicited data since the R2T PDU which is used for solicitation is transformed into RDMA Read operations by the iSER layer at the target. See section 7.3.4.
- b. A NOP-Out PDU with TTT != 0xffffffff is sent as a ping response by the initiator to the NOP-In PDU sent as a ping request by the target.

8.1.1.3 Control-Type PDUs from the Initiator in the Unregulated and Unexpected Category

PDUs in the Unregulated and Unexpected category are PDUs with the immediate flag set. The number of PDUs in this category which can be sent by an initiator is controlled by the value of MaxOutstandingUnexpectedPDUs declared by the target. (See section 6.7.) After a PDU in this category is sent by the initiator, it is outstanding until it is retired. At any time, the number of outstanding unexpected PDUs MUST NOT exceed the value of MaxOutstandingUnexpectedPDUs declared by the target.

The target uses the value of MaxOutstandingUnexpectedPDUs that it declared to determine the amount of buffer resources required for control-type PDUs in this category that can be sent by an initiator. For the initiator, for each of the control-type PDUs that can be sent in this category, the initiator MUST provision for the buffer resources if required for the corresponding control-type PDU that can be sent as a response from the target.

An outstanding PDU in this category is retired as follows. If the CmdSN of the PDU sent by the initiator in this category is x, the PDU is outstanding until the initiator sends a non-immediate control-type PDU on the same connection with CmdSN = y (where y is at least x) and the target responds with a control-type PDU on any connection where ExpCmdSN is at least y+1.

When the number of outstanding unexpected control-type PDUs equals MaxOutstandingUnexpectedPDUs, the iSCSI Layer at the initiator MUST NOT generate any unexpected PDUs which otherwise it would have generated, even if it is intended for immediate delivery.

8.1.2 Flow Control for Control-Type PDUs from the Target

Control-type PDUs that can be sent by a target and are expected by the initiator are listed in the Regulated category. (See section 8.1.1.1.)

For the control-type PDUs that can be sent by a target and are unexpected by the initiator, the number is controlled by `MaxOutstandingUnexpectedPDUs` declared by the initiator. (See section 6.7.) After a PDU in this category is sent by a target, it is outstanding until it is retired. At any time, the number of outstanding unexpected PDUs MUST NOT exceed the value of `MaxOutstandingUnexpectedPDUs` declared by the initiator. The initiator uses the value of `MaxOutstandingUnexpectedPDUs` that it declared to determine the amount of buffer resources required for control-type PDUs in this category that can be sent by a target. The following is a list of the PDUs in this category and the conditions for retiring the outstanding PDU:

- a. For an Asynchronous Message PDU with `StatSN = x`, the PDU is outstanding until the initiator sends a control-type PDU with `ExpStatSN` set to at least `x+1`.
- b. For a Reject PDU with `StatSN = x` which is sent after a task is active, the PDU is outstanding until the initiator sends a control-type PDU with `ExpStatSN` set to at least `x+1`.
- c. For a NOP-In PDU with `ITT = 0xffffffff` and `StatSN = x`, the PDU is outstanding until the initiator responds with a control-type PDU on the same connection where `ExpStatSN` is at least `x+1`. But if the NOP-In PDU is sent as a ping request with `TTT != 0xffffffff`, the PDU can also be retired when the initiator sends a NOP-Out PDU with the same `ITT` and `TTT` as in the ping request. Note that when a target sends a NOP-In PDU as a ping request, it must provision a buffer for the NOP-Out PDU sent as a ping response from the initiator.

When the number of outstanding unexpected control-type PDUs equals `MaxOutstandingUnexpectedPDUs`, the iSCSI Layer at the target MUST NOT generate any unexpected PDUs which otherwise it would have generated, even if its intent is to indicate an iSCSI error condition (e.g., Asynchronous Message, Reject). Task timeouts as in the initiator waiting for a command completion or other connection and session level exceptions will ensure that correct operational behavior will result in these cases despite not generating the PDU. This rule overrides any other requirements elsewhere which require that a Reject PDU MUST be sent.

(Implementation note: SCSI task timeout and recovery can be a lengthy process and hence SHOULD be avoided by proper provisioning of resources.)

(Implementation note: To ensure that the initiator has a means to inform the target that outstanding PDUs have been retired, the target should reserve the last unexpected control-type PDU allowable by the value of MaxOutstandingUnexpectedPDUs declared by the initiator for sending a NOP-In ping request with TTT != 0xffffffff to allow the initiator to return the NOP-Out ping response with the current ExpStatSN.)

8.2 Flow Control for RDMA Read Resources

If iSERHelloRequired is negotiated to "Yes", then the total number of RDMA Read operations that can be active simultaneously on an iSCSI/iSER connection depends on the amount of resources allocated as declared in the iSER Hello exchange described in section 5.1.3. Exceeding the number of RDMA Read operations allowed on a connection will result in the connection being terminated by the RCaP layer. The iSER Layer at the target maintains the iSER-ORD to keep track of the maximum number of RDMA Read Requests that can be issued by the iSER Layer on a particular RCaP Stream.

During connection setup (see section 5.1), iSER-IRD is known at the initiator and iSER-ORD is known at the target after the iSER Layers at the initiator and the target have respectively allocated the connection resources necessary to support RCaP, as directed by the Allocate_Connection_Resources Operational Primitive from the iSCSI Layer before the end of the iSCSI Login Phase. In the full feature phase, if iSERHelloRequired is negotiated to "Yes", then the first message sent by the initiator is the iSER Hello Message (see section 9.3) which contains the value of iSER-IRD. In response to the iSER Hello Message, the target sends the iSER HelloReply Message (see section 9.4) which contains the value of iSER-ORD. The iSER Layer at both the initiator and the target MAY adjust (lower) the resources associated with iSER-IRD and iSER-ORD respectively to match the iSER-ORD value declared in the HelloReply Message. The iSER Layer at the target MUST flow control the RDMA Read Request Messages to not exceed the iSER-ORD value at the target.

If iSERHelloRequired is negotiated to "No", then the maximum number of RDMA Read operations that can be active is negotiated via other means outside the scope of this document. For example, in InfiniBand, iSER connection setup uses InfiniBand CM MADs, with additional iSER information exchanged in the private data.

8.3 STag Management

An STag is an identifier of a Tagged Buffer used in an RDMA operation. The allocation and the subsequent invalidation of the STags are specified in this document if the STags are exposed on the wire by being Advertised in the iSER header or declared in the header of an RCaP Message.

8.3.1 Allocation of STags

When the iSCSI Layer at the initiator invokes the Send_Control Operational Primitive to request the iSER Layer at the initiator to process a SCSI Command, zero, one, or two STags may be allocated by the iSER Layer. See section 7.3.1 for details. The number of STags allocated depends on whether the command is unidirectional or bidirectional and whether solicited write data transfer is involved or not.

When the iSCSI Layer at the initiator invokes the Send_Control Operational Primitive to request the iSER Layer at the initiator to process a Task Management Function Request with the TASK REASSIGN function, besides allocating zero, one, or two STags, the iSER Layer MUST invalidate the existing STags (if any) associated with the ITT. See section 7.3.3 for details.

The iSER Layer at the target allocates a local Data Sink STag when the iSCSI Layer at the target invokes the Get_Data Operational Primitive to request the iSER Layer to process an R2T PDU. See section 7.3.6 for details.

8.3.2 Invalidation of STags

The invalidation of the STags at the initiator at the completion of a unidirectional or bidirectional command when the associated SCSI Response PDU is sent by the target is described in section 7.3.2.

When a unidirectional or bidirectional command concludes without the associated SCSI Response PDU being sent by the target, the iSCSI Layer at the initiator MUST request the iSER Layer at the initiator to invalidate the STags by invoking the Deallocate_Task_Resources Operational Primitive qualified with ITT. In response, the iSER Layer at the initiator MUST locate the STags (if any) in the Local Mapping. The iSER Layer at the initiator MUST invalidate the STags (if any) and the Local Mapping.

For an RDMA Read operation used to realize a SCSI Write data transfer, the iSER Layer at the target SHOULD invalidate the Data

Sink STag at the conclusion of the RDMA Read operation referencing the Data Sink STag (to permit the immediate reuse of buffer resources).

For an RDMA Write operation used to realize a SCSI Read data transfer, the Data Source STag at the target is not declared to the initiator and is not exposed on the wire. Invalidation of the STag is thus not specified.

When a unidirectional or bidirectional command concludes without the associated SCSI Response PDU being sent by the target, the iSCSI Layer at the target MUST request the iSER Layer at the target to invalidate the STags by invoking the Deallocate_Task_Resources Operational Primitive qualified with ITT. In response, the iSER Layer at the target MUST locate the local STags (if any) in the Local Mapping. The iSER Layer at the target MUST invalidate the local STags (if any) and the Local Mapping.

9 iSER Control and Data Transfer

For iSCSI data-type PDUs (see section 7.1), the iSER Layer uses RDMA Read and RDMA Write operations to transfer the solicited data. For iSCSI control-type PDUs (see section 7.2), the iSER Layer uses Send Messages of RCaP.

9.1 iSER Header Format

An iSER header MUST be present in every Send Message of RCaP. The iSER header is located in the first 28 bytes of the message payload of the Send Message of RCaP, as shown in Figure 2.

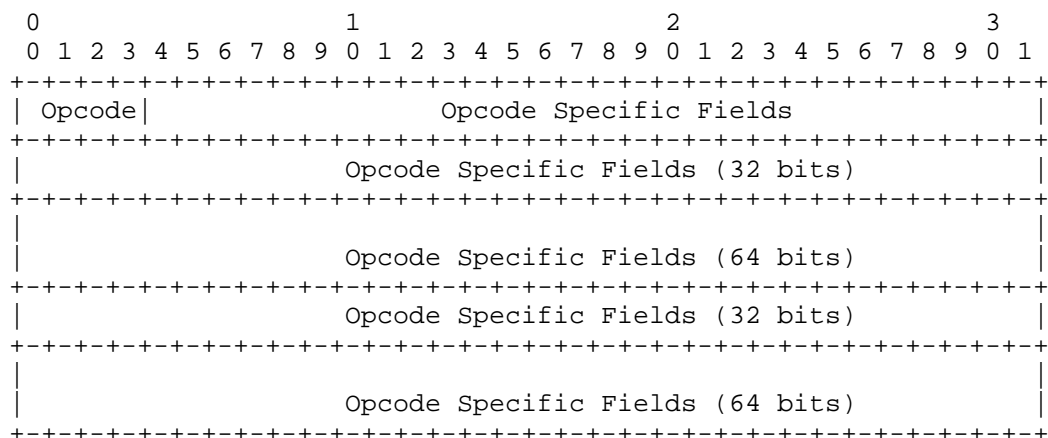


Figure 2 iSER Header Format

Opcode - Operation Code: 4 bits

The Opcode field identifies the type of iSER Messages:

0001b = iSCSI control-type PDU

0010b = iSER Hello Message

0011b = iSER HelloReply Message

All other opcodes are reserved.

9.2 iSER Header Format for iSCSI Control-Type PDU

The iSER Layer uses Send Messages of RCaP to transfer iSCSI control-type PDUs (see section 7.2). The message payload of each of the

Send Messages of RCaP used for transferring an iSER Message contains an iSER Header followed by an iSCSI control-type PDU.

The iSER header in a Send Message of RCaP carrying an iSCSI control-type PDU MUST have the format as described in Figure 3.

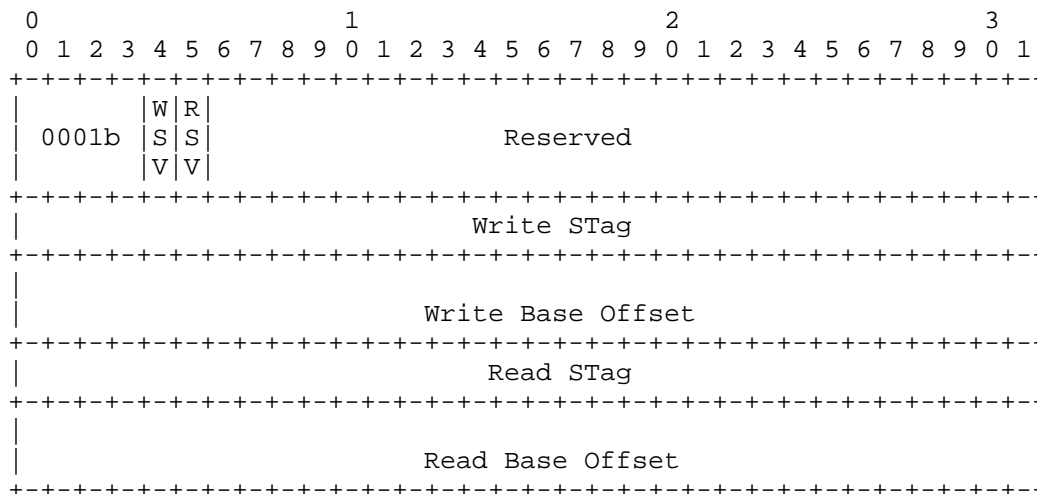


Figure 3 iSER Header Format for iSCSI Control-Type PDU

WSV - Write STag Valid flag: 1 bit

This flag indicates the validity of the Write STag field and the Write Base Offset field of the iSER Header. If set to one, the Write STag field and the Write Base Offset field in this iSER Header are valid. If set to zero, the Write STag field and the Write Base Offset field in this iSER Header MUST be ignored at the receiver. The Write STag Valid flag is set to one when there is solicited data to be transferred for a SCSI Write or bidirectional command, or when there are non-immediate unsolicited and solicited data to be transferred for the referenced task specified in a Task Management Function Request with the TASK REASSIGN function.

RSV - Read STag Valid flag: 1 bit

This flag indicates the validity of the Read STag field and the Read Base Offset field of the iSER Header. If set to one, the Read STag field and the Read Base Offset field in this iSER Header is valid. If set to zero, the Read STag field and the Read Base Offset field in this iSER Header MUST be ignored at the receiver. The Read STag Valid flag is set to one for a

SCSI Read or bidirectional command, or a Task Management Function Request with the TASK REASSIGN function.

Write STag - Write Steering Tag: 32 bits

This field contains the Write STag when the Write STag Valid flag is set to one. For a SCSI Write or bidirectional command, the Write STag is used to Advertise the initiator's I/O Buffer containing the solicited data. For a Task Management Function Request with the TASK REASSIGN function, the Write STag is used to Advertise the initiator's I/O Buffer containing the non-immediate unsolicited data and solicited data. This Write STag is used as the Data Source STag in the resultant RDMA Read operation(s). When the Write STag Valid flag is set to zero, this field MUST be set to zero and ignored on receive.

Write Base Offset: 64 bits

This field contains the Base Offset associated with the I/O Buffer for the SCSI Write command when the Write STag Valid flag is set to one. When the Write STag Valid flag is set to zero, this field MUST be set to zero and ignored on receive.

Read STag - Read Steering Tag: 32 bits

This field contains the Read STag when the Read STag Valid flag is set to one. The Read STag is used to Advertise the initiator's Read I/O Buffer of a SCSI Read or bidirectional command, or a Task Management Function Request with the TASK REASSIGN function. This Read STag is used as the Data Sink STag in the resultant RDMA Write operation(s). When the Read STag Valid flag is zero, this field MUST be set to zero and ignored on receive.

Read Base Offset: 64 bits

This field contains the Base Offset associated with the I/O Buffer for the SCSI Read command when the Read STag Valid flag is set to one. When the Read STag Valid flag is set to zero, this field MUST be set to zero and ignored on receive.

Reserved:

Reserved fields MUST be set to zero on transmit and MUST be ignored on receive.

9.3 iSER Header Format for iSER Hello Message

An iSER Hello Message MUST only contain the iSER header which MUST have the format as described in Figure 4. If iSERHelloRequired is negotiated to "Yes", then iSER Hello Message is the first iSER Message sent on the RCaP Stream from the iSER Layer at the initiator to the iSER Layer at the target.

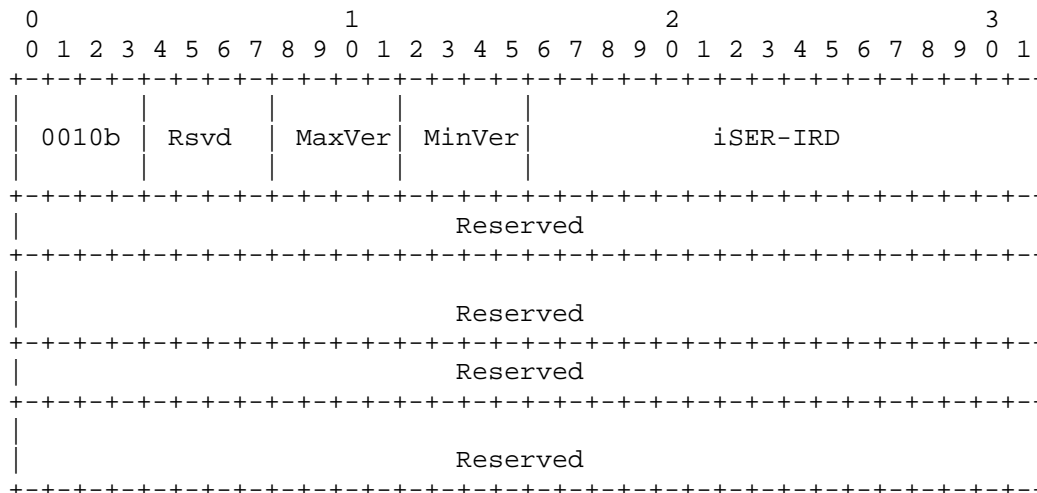


Figure 4 iSER Header Format for iSER Hello Message

MaxVer - Maximum Version: 4 bits

This field specifies the maximum version of the iSER protocol supported. It MUST be set to 10 to indicate the version of the specification described in this document.

MinVer - Minimum Version: 4 bits

This field specifies the minimum version of the iSER protocol supported. It MUST be set to 10 to indicate the version of the specification described in this document.

iSER-IRD: 16 bits

This field contains the value of the iSER-IRD at the initiator.

Reserved (Rsvd):

Reserved fields MUST be set to zero on transmit, and MUST be ignored on receive.

9.4 iSER Header Format for iSER HelloReply Message

An iSER HelloReply Message MUST only contain the iSER header which MUST have the format as described in Figure 5. If iSERHelloRequired is negotiated to "Yes", then the iSER HelloReply Message is the first iSER Message sent on the RCaP Stream from the iSER Layer at the target to the iSER Layer at the initiator.

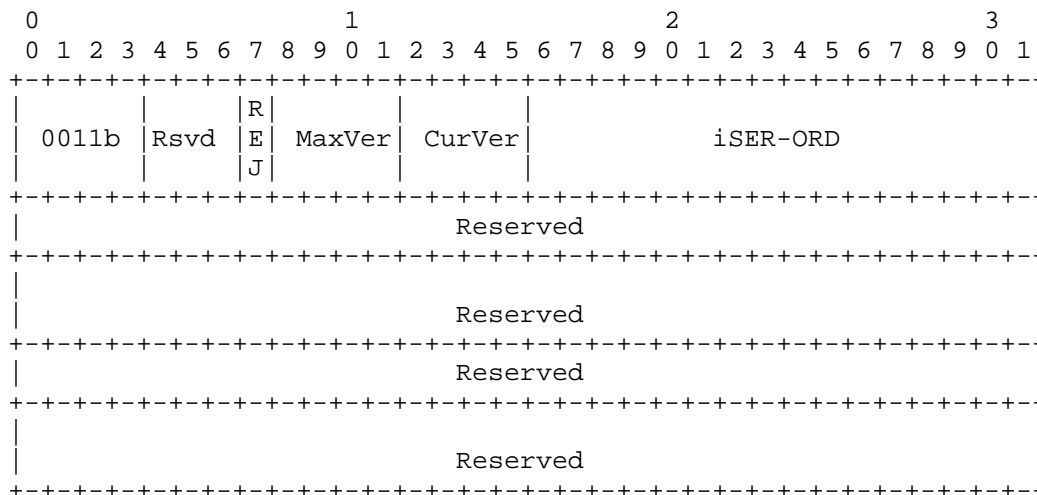


Figure 5 iSER Header Format for iSER HelloReply Message

REJ - Reject flag: 1 bit

This flag indicates whether the target is rejecting this connection. If set to one, the target is rejecting the connection.

MaxVer - Maximum Version: 4 bits

This field specifies the maximum version of the iSER protocol supported. It MUST be set to 10 to indicate the version of the specification described in this document.

CurVer - Current Version: 4 bits

This field specifies the current version of the iSER protocol supported. It MUST be set to 10 to indicate the version of the specification described in this document.

iSER-ORD: 16 bits

This field contains the value of the iSER-ORD at the target.

Reserved (Rsvd):

Reserved fields MUST be set to zero on transmit, and MUST be ignored on receive.

9.5 SCSI Data Transfer Operations

The iSER Layer at the initiator and the iSER Layer at the target handle each SCSI Write, SCSI Read, and bidirectional operation as described below.

9.5.1 SCSI Write Operation

The iSCSI Layer at the initiator MUST invoke the Send_Control Operational Primitive to request the iSER Layer at the initiator to send the SCSI Write Command. The iSER Layer at the initiator MUST request the RCaP layer to transmit a Send Message with the message payload consisting of the iSER header followed by the SCSI Command PDU and immediate data (if any). The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP). If there is solicited data, the iSER Layer MUST Advertise the Write STag and the Base Offset in the iSER header of the Send Message, as described in section 9.2. Upon receiving the Send Message, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Control_Notify Operational Primitive qualified with the SCSI Command PDU. See section 7.3.1 for details on the handling of the SCSI Write Command.

For the non-immediate unsolicited data, the iSCSI Layer at the initiator MUST invoke a Send_Control Operational Primitive qualified with the SCSI Data-out PDU. Upon receiving each Send Message containing the non-immediate unsolicited data, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Control_Notify Operational Primitive qualified with the SCSI Data-out PDU. See section 7.3.4 for details on the handling of the SCSI Data-out PDU.

For the solicited data, when the iSCSI Layer at the target has an I/O Buffer available, it MUST invoke the Get_Data Operational

Primitive qualified with the R2T PDU. See section 7.3.6 for details on the handling of the R2T PDU.

When the data transfer associated with this SCSI Write operation is complete, the iSCSI Layer at the target MUST invoke the Send_Control Operational Primitive when it is ready to send the SCSI Response PDU. Upon receiving a Send Message containing the SCSI Response PDU, the iSER Layer at the initiator MUST notify the iSCSI Layer at the initiator by invoking the Control_Notify Operational Primitive qualified with the SCSI Response PDU. See section 7.3.2 for details on the handling of the SCSI Response PDU.

9.5.2 SCSI Read Operation

The iSCSI Layer at the initiator MUST invoke the Send_Control Operational Primitive to request the iSER Layer at the initiator to send the SCSI Read Command. The iSER Layer at the initiator MUST request the RCaP layer to transmit a Send Message with the message payload consisting of the iSER header followed by the SCSI Command PDU. The SendSE Message should be used if supported by the RCaP layer (e.g., iWARP). The iSER Layer at the initiator MUST Advertise the Read STag and the Base Offset in the iSER header of the Send Message, as described in section 9.2. Upon receiving the Send Message, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Control_Notify Operational Primitive qualified with the SCSI Command PDU. See section 7.3.1 for details on the handling of the SCSI Read Command.

When the requested SCSI data is available in the I/O Buffer, the iSCSI Layer at the target MUST invoke the Put_Data Operational Primitive qualified with the SCSI Data-in PDU. See section 7.3.5 for details on the handling of the SCSI Data-in PDU.

When the data transfer associated with this SCSI Read operation is complete, the iSCSI Layer at the target MUST invoke the Send_Control Operational Primitive when it is ready to send the SCSI Response PDU. The SendInvSE Message should be used if supported by the RCaP layer (e.g., iWARP). Upon receiving the Send Message containing the SCSI Response PDU, the iSER Layer at the initiator MUST notify the iSCSI Layer at the initiator by invoking the Control_Notify Operational Primitive qualified with the SCSI Response PDU. See section 7.3.2 for details on the handling of the SCSI Response PDU.

9.5.3 Bidirectional Operation

The initiator and the target handle the SCSI Write and the SCSI Read portions of this bidirectional operation the same as described in Section 9.5.1 and Section 9.5.2 respectively.

10 iSER Error Handling and Recovery

RCaP provides the iSER Layer with reliable in-order delivery. Therefore, the error management needs of an iSER-assisted connection are somewhat different than those of a Traditional iSCSI connection.

10.1 Error Handling

iSER error handling is described in the following sections, classified loosely based on the sources of errors:

1. Those originating at the transport layer (e.g., TCP).
2. Those originating at the RCaP layer.
3. Those originating at the iSER Layer.
4. Those originating at the iSCSI Layer.

10.1.1 Errors in the Transport Layer

If the transport layer is TCP, then TCP packets with detected errors are silently dropped by the TCP layer and result in retransmission at the TCP layer. This has no impact on the iSER Layer. However, connection loss (e.g., link failure) and unexpected termination (e.g., TCP graceful or abnormal close without the iSCSI Logout exchanges) at the transport layer will cause the iSCSI/iSER connection to be terminated as well.

10.1.1.1 Failure in the Transport Layer Before RCaP Mode is Enabled

If the Connection is lost or terminated before the iSCSI Layer invokes the `Allocate_Connection_Resources` Operational Primitive, the login process is terminated and no further action is required.

If the Connection is lost or terminated after the iSCSI Layer has invoked the `Allocate_Connection_Resources` Operational Primitive, then the iSCSI Layer MUST request the iSER Layer to deallocate all connection resources by invoking the `Deallocate_Connection_Resources` Operational Primitive.

10.1.1.2 Failure in the Transport Layer After RCaP Mode is Enabled

If the Connection is lost or terminated after the iSCSI Layer has invoked the `Enable_Datamover` Operational Primitive, the iSER Layer MUST notify the iSCSI Layer of the connection loss by invoking the `Connection_Terminate_Notify` Operational Primitive. Prior to

10.1.1.2 Errors in the RCaP Layer

The RCaP layer does not have error recovery operations built in. If errors are detected at the RCaP layer, the RCaP layer will terminate the RCaP Stream and the associated Connection.

10.1.1.2.1 Errors Detected in the Local RCaP Layer

If an error is encountered at the local RCaP layer, the RCaP layer MAY send a Send Message to the Remote Peer to report the error if possible. (For iWARP, see [RDMAP] for the list of errors where a Terminate Message is sent.) The RCaP layer is responsible for terminating the Connection. After the RCaP layer notifies the iSER Layer that the Connection is terminated, the iSER Layer MUST notify the iSCSI Layer by invoking the Connection_Terminate_Notify Operational Primitive. Prior to invoking the Connection Terminate Notify Operational Primitive, the iSER layer MUST perform the actions described in Section 5.2.3.2.

10.1.1.2.2 Errors Detected in the RCaP Layer at the Remote Peer

If an error is encountered at the RCaP layer at the Remote Peer, the RCaP layer at the Remote Peer may send a Send Message to report the error if possible. If it is unable to send a Send Message, the Connection is terminated. This is treated the same as a failure in the transport layer after RDMA is enabled as described in section 10.1.1.2.

If an error is encountered at the RCaP layer at the Remote Peer and it is able to send a Send Message, the RCaP layer at the Remote Peer is responsible for terminating the connection. After the local RCaP layer notifies the iSER Layer that the Connection is terminated, the iSER Layer MUST notify the iSCSI Layer by invoking the Connection Terminate Notify Operational Primitive. Prior to invoking the Connection_Terminate_Notify Operational Primitive, the iSER layer MUST perform the actions described in Section 5.2.3.2.

10.1.1.3 Errors in the iSER Layer

The error handling due to errors at the iSER Layer is described in the following sections.

10.1.3.1 Insufficient Connection Resources to Support RCaP at Connection Setup

After the iSCSI Layer at the initiator invokes the `Allocate_Connection_Resources` Operational Primitive during the iSCSI login negotiation phase, if the iSER Layer at the initiator fails to allocate the connection resources necessary to support RCaP, it MUST return a status of failure to the iSCSI Layer at the initiator. The iSCSI Layer at the initiator MUST terminate the Connection as described in Section 5.2.3.1.

After the iSCSI Layer at the target invokes the `Allocate_Connection_Resources` Operational Primitive during the iSCSI login negotiation phase, if the iSER Layer at the target fails to allocate the connection resources necessary to support RCaP, it MUST return a status of failure to the iSCSI Layer at the target. The iSCSI Layer at the target MUST send a Login Response with a status class of 3 (Target Error), and a status code of "0302" (Out of Resources). The iSCSI Layers at the initiator and the target MUST terminate the Connection as described in Section 5.2.3.1.

10.1.3.2 iSER Negotiation Failures

If `iSERHelloRequired` is negotiated to "Yes" and the RCaP or iSER related parameters declared by the initiator in the iSER Hello Message is unacceptable to the iSER Layer at the target, the iSER Layer at the target MUST set the Reject (REJ) flag, as described in section 9.4, in the iSER HelloReply Message. The following are the cases when the iSER Layer MUST set the REJ flag to 1 in the HelloReply Message:

- * The initiator-declared `iSER-IRD` value is greater than 0 and the target-declared `iSER-ORD` value is 0.
- * The initiator-supported and the target-supported iSER protocol versions do not overlap.

After requesting the RCaP layer to send the iSER HelloReply Message, the handling of the error situation is the same as that for iSER format errors as described in section 10.1.3.3.

10.1.3.3 iSER Format Errors

The following types of errors in an iSER header are considered format errors:

- * Illegal contents of any iSER header field

- * Inconsistent field contents in an iSER header
- * Length error for an iSER Hello or HelloReply Message (see section 9.3 and 9.4)

When a format error is detected, the following events MUST occur in the specified sequence:

1. The iSER Layer MUST request the RCaP layer to terminate the RCaP Stream. The RCaP layer MUST terminate the associated Connection.
2. The iSER Layer MUST notify the iSCSI Layer of the connection termination by invoking the Connection_Terminate_Notify Operational Primitive. Prior to invoking the Connection_Terminate_Notify Operational Primitive, the iSER layer MUST perform the actions described in Section 5.2.3.2.

10.1.3.4 iSER Protocol Errors

If iSERHelloRequired is negotiated to "Yes", then the first iSER Message sent by the iSER Layer at the initiator MUST be the iSER Hello Message (see section 9.3). In this case the first iSER Message sent by the iSER Layer at the target MUST be the iSER HelloReply Message (see section 9.4). Failure to send the iSER Hello or HelloReply Message, as indicated by the wrong Opcode in the iSER header, is a protocol error. Conversely, if the iSER Hello Message is sent by the iSER Layer at the initiator when iSERHelloRequired is negotiated to "No", the iSER Layer at the target MAY treat this as a protocol error or respond with an iSER HelloReply Message. The handling of iSER protocol errors is the same as that for iSER format errors as described in section 10.1.3.3.

If the sending side of an iSER-enabled connection acts in a manner not permitted by the negotiated or declared login/text operational key values as described in section 6, this is a protocol error and the receiving side MAY handle this the same as for iSER format errors as described in section 10.1.3.3.

10.1.4 Errors in the iSCSI Layer

The error handling due to errors at the iSCSI Layer is described in the following sections. For error recovery, see section 10.2.

10.1.4.1 iSCSI Format Errors

When an iSCSI format error is detected, the iSCSI Layer MUST request the iSER Layer to terminate the RCaP Stream by invoking the `Connection_Terminate` Operational Primitive. For more details on the connection termination, see Section 5.2.3.1.

10.1.4.2 iSCSI Digest Errors

In the iSER-assisted mode, the iSCSI Layer will not see any digest error because both the `HeaderDigest` and the `DataDigest` keys are negotiated to "None".

10.1.4.3 iSCSI Sequence Errors

For Traditional iSCSI, sequence errors are caused by dropped PDUs due to header or data digest errors. Since digests are not used in iSER-assisted mode and the RCaP layer will deliver all messages in the order they were sent, sequence errors will not occur in iSER-assisted mode.

10.1.4.4 iSCSI Protocol Error

When the iSCSI Layer handles certain protocol errors by dropping the connection, the error handling is the same as that for iSCSI format errors as described in section 10.1.4.1.

When the iSCSI Layer uses the iSCSI Reject PDU and response codes to handle certain other protocol errors, no special handling at the iSER Layer is required.

10.1.4.5 SCSI Timeouts and Session Errors

This is handled at the iSCSI Layer and no special handling at the iSER Layer is required.

10.1.4.6 iSCSI Negotiation Failures

For negotiation failures that happen during the Login Phase at the initiator after the iSCSI Layer has invoked the `Allocate_Connection_Resources` Operational Primitive and before the `Enable_Datamover` Operational Primitive has been invoked, the iSCSI Layer MUST request the iSER Layer to deallocate all connection resources by invoking the `Deallocate_Connection_Resources` Operational Primitive. The iSCSI Layer at the initiator MUST terminate the Connection.

For negotiation failures during the Login Phase at the target, the iSCSI Layer can use a Login Response with a status class other than 0 (success) to terminate the Login Phase. If the iSCSI Layer has invoked the `Allocate_Connection_Resources` Operational Primitive and before the `Enable_Datamover` Operational Primitive has been invoked, the iSCSI Layer at the target MUST request the iSER Layer at the target to deallocate all connection resources by invoking the `Deallocate_Connection_Resources` Operational Primitive. The iSCSI Layer at both the initiator and the target MUST terminate the Connection.

During the iSCSI Login Phase, if the iSCSI Layer at the initiator receives a Login Response from the target with a status class other than 0 (Success) after the iSCSI Layer at the initiator has invoked the `Allocate_Connection_Resources` Operational Primitive, the iSCSI Layer MUST request the iSER Layer to deallocate all connection resources by invoking the `Deallocate_Connection_Resources` Operational Primitive. The iSCSI Layer MUST terminate the Connection in this case.

For negotiation failures during the full feature phase, the error handling is left to the iSCSI Layer and no special handling at the iSER Layer is required.

10.2 Error Recovery

Error recovery requirements of iSCSI/iSER are the same as that of Traditional iSCSI. All three `ErrorRecoveryLevels` as defined in [iSCSI] are supported in iSCSI/iSER.

- * For `ErrorRecoveryLevel 0`, session recovery is handled by iSCSI and no special handling by the iSER Layer is required.
- * For `ErrorRecoveryLevel 1`, see section 10.2.1 on PDU Recovery.
- * For `ErrorRecoveryLevel 2`, see section 10.2.2 on Connection Recovery.

The iSCSI Layer may invoke the `Notice_Key_Values` Operational Primitive during connection setup to request the iSER Layer to take note of the value of the operational `ErrorRecoveryLevel`, as described in sections 5.1.1 and 5.1.2.

10.2.1 PDU Recovery

As described in sections 10.1.4.2 and 10.1.4.3, digest and sequence errors will not occur in the iSER-assisted mode. If the RCaP layer

detects an error, it will close the iSCSI/iSER connection, as described in section 10.1.2. Therefore, PDU recovery is not useful in the iSER-assisted mode.

The iSCSI Layer at the initiator SHOULD disable iSCSI timeout-driven PDU retransmissions.

10.2.2 Connection Recovery

The iSCSI Layer at the initiator MAY reassign connection allegiance for non-immediate commands which are still in progress and are associated with the failed connection by using a Task Management Function Request with the TASK REASSIGN function. See section 7.3.3 for more details.

When the iSCSI Layer at the initiator does a task reassignment for a SCSI Write command, it MUST qualify the Send_Control Operational Primitive invocation with DataDescriptorOut which defines the I/O Buffer for both the non-immediate unsolicited data and the solicited data. This allows the iSCSI Layer at the target to use recovery R2Ts to request for data originally sent as unsolicited and solicited from the initiator.

When the iSCSI Layer at the target accepts a reassignment request for a SCSI Read command, it MUST request the iSER Layer to process SCSI Data-in for all unacknowledged data by invoking the Put_Data Operational Primitive. See section 7.3.5 on the handling of SCSI Data-in.

When the iSCSI Layer at the target accepts a reassignment request for a SCSI Write command, it MUST request the iSER Layer to process a recovery R2T for any non-immediate unsolicited data and any solicited data sequences that have not been received by invoking the Get_Data Operational Primitive. See section 7.3.6 on the handling of Ready To Transfer (R2T).

The iSCSI Layer at the target MUST NOT issue recovery R2Ts on an iSCSI/iSER connection for a task for which the connection allegiance was never reassigned. The iSER Layer at the target MAY reject such a recovery R2T received via the Get_Data Operational Primitive invocation from the iSCSI Layer at the target, with an appropriate error code.

The iSER Layer at the target will process the requests invoked by the Put_Data and Get_Data Operational Primitives for a reassigned task in the same way as for the original commands.

11 Security Considerations

When iSER is layered on top of an RCaP layer and provides the RDMA extensions to the iSCSI protocol, the security considerations of iSER are the same as that of the underlying RCaP layer. For iWARP, this is described in [RDMA] and [RDPSEC], plus the updates to both of those RFCs that are contained in [IPSEC-IPS].

Since iSER-assisted iSCSI protocol is still functionally iSCSI from a security considerations perspective, all of the iSCSI security requirements as described in [iSCSI] applies. If iSER is layered on top of a non-IP based RCaP layer, all the security protocol mechanisms applicable to that RCaP layer is also applicable to an iSCSI/iSER connection. If iSER is layered on top of a non-IP protocol, the IPsec mechanism as specified in [iSCSI] MUST be implemented at any point where the iSER protocol enters the IP network (e.g., via gateways), and the non-IP protocol SHOULD implement (optional to use) a packet by packet security protocol equal in strength to the IPsec mechanism specified by [iSCSI].

In order to protect target RCaP connection resources from possible resource exhaustion attacks, allocation of such resources for a new connection MUST be delayed until it is reasonably certain that the new connection is not part of a resource exhaustion attack (e.g., until after the SecurityNegotiation stage of Login), see section 5.1.2.

A valid STag exposes I/O Buffer resources to the network for access via the RCaP. The security measures for the RCaP and iSER described in the above paragraphs can be used to protect data in an I/O buffer from undesired disclosure or modification, and these measures are of heightened importance for implementations that retain (e.g., cache) STags for use in multiple tasks (e.g., iSCSI I/O operations) because the resources are exposed to the network for a longer period of time.

A complementary means of controlling I/O Buffer resource exposure is invalidation of the STag after completion of the associated task, which is RECOMMENDED in Section 2.5.1. The use of Send with Invalidate messages (which cause remote STag invalidation) is OPTIONAL, therefore the iSER layer MUST NOT rely on use of a Send with Invalidate by its Remote Peer to cause local STag invalidation. If an STag is expected to be invalid after completion of a task, the iSER layer MUST check the STag and invalidate it if it is still valid.

IANA is requested to add the following entries to the "iSCSI Login/Text Keys" registry of "iSCSI Parameters":

MaxAHSLength, [RFCXXXX]

TaggedBufferForSolicitedDataOnly, [RFCXXXX]

iSERHelloRequired, [RFCXXXX]

RFC Editor: Please replace XXXX in all instances of [RFCXXXX] above with the RFC number of this document and remove this note.

IANA is requested to update the following entries in the "iSCSI Login/Text Keys" registry of "iSCSI Parameters" to reference the RFC number of this draft when it is published as an RFC.

InitiatorRecvDataSegmentLength

MaxOutstandingUnexpectedPDUs

RDMAExtensions

TargetRecvDataSegmentLength

IANA is also requested to change the RFC5046 reference for the iSCSI Login/Text Keys registry to the RFC number of this document.

IANA is requested to update the registrations of the iSER Opcodes 1-3 in the iSER Opcodes registry to reference the RFC number of this draft when it is published as an RFC.

13.1 Normative References

- [RFC5046] M. Ko et al., "iSCSI Extensions for Remote Direct Memory Access", RFC 5046, October 2007
- [iSCSI] Chadalapaka et al., "iSCSI Protocol (Consolidated)", draft-ietf-storm-iscsi-cons-08.txt (work in progress), January 2013
- [RDMAP] R. Recio et al., "An RDMA Protocol Specification", RFC 5040, October 2007
- [DDP] H. Shah et al., "Direct Data Placement over Reliable Transports", RFC 5041, October 2007
- [MPA] P. Culley et al., "Marker PDU Aligned Framing for TCP Specification", RFC 5044, October 2007
- [RDDPSEC] J. Pinkerton et al., "DDP/RDMAP Security", RFC 5042, October 2007
- [TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981
- [RFC2119] Bradner, S., "Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997
- [IPS-IPSEC] D. Black et al., "Securing Block Storage Protocols over IP: RFC 3723 Requirements Update for IPsec v3", draft-ietf-storm-ipsec-ips-update-03 (work in progress), July 2013

13.2 Informative References

- [SAM5] T10/2104D rev r04, SCSI Architecture Model - 5 (SAM-5), Committee Draft.
- [iSCSI-SAM] F. Knight et al., "Internet Small Computer Systems Interface (iSCSI) SCSI Architecture Features Update", draft-ietf-storm-iscsi-sam-04.txt (work in progress), August 2011
- [DA] M. Chadalapaka et al., "Datamover Architecture for iSCSI", RFC 5047, October 2007
- [IB] InfiniBand Architecture Specification Volume 1 Release 1.2, October 2004

All changes are backward compatible with RFC 5046 except for item #8 which reflects all known implementations of iSER, each of which has implemented this change, despite its absence in RFC 5046. As a result, a hypothetical implementation based on RFC 5046 will not interoperate with an implementation based on this version of the specification.

1. Removed the requirement that a connection be opened in "normal" TCP mode and transitioned to zero-copy mode. This allows the spec to conform to existing implementation for both Infiniband and iWARP. Changes were made in sections 2, 3.1.6, 4.2, 5.1, 5.1.1, 5.1.2, 5.1.3, 10.1.3.4, and 11.
2. Added a clause in section 6.2 to clarify that MaxRecvDataSegmentLength must be ignored if it is declared in the Login Phase.
3. Added a clause in section 6.2 to clarify that the initiator must not send more than InitiatorMaxRecvDataSegmentLength worth of data when a NOP-Out request is sent with a valid Initiator Task Tag. Since InitiatorMaxRecvDataSegmentLength can be smaller than TargetMaxRecvDataSegmentLength, returning the original data in the NOP-Out request in this situation can overflow the receive buffer unless the length of the data sent with the NOP-Out request is less than InitiatorMaxRecvDataSegmentLength.
4. Added a SHOULD negotiate recommendation for MaxOutstandingUnexpectedPDUs in section 6.7.
5. Added MaxAHSLength key in section 6.8 to set a limit on the AHS Length. This is useful when posting receive buffers in knowing what the maximum possible message length is in a PDU which contains AHS.
6. Added TaggedBufferForSolicitedDataOnly key in section 6.9 to indicate how the memory region will be used. An initiator can treat the memory regions intended for unsolicited and solicited data differently, and can use different registration modes. In contrast, RFC 5046 treats the memory occupied by the data as a contiguous (or virtually contiguous, by means of scatter-gather mechanisms) and homogenous region. Adding a new key will allow different memory models to be accommodated. Changes were also made in section 7.3.1.

7. Added iSERHelloRequired key in section 6.10 to allow an initiator to allocate connection resources after the login process by requiring the use of the iSER Hello messages before sending iSCSI PDUs. The default is "No" since iSER Hello messages have not been implemented and are not in use. Changes were made in sections 5.1.1, 5.1.2, 5.1.3, 8.2, 9.3, 9.4, 10.1.3.2 and 10.1.3.4.
8. Added two 64-bit fields in iSER header in section 9.2 for the Read Base Offset and the Write Base Offset to accommodate a non-zero Base Offset. This allows one implementation such as the OFED stack to be used in both the Infiniband and the iWARP environment. Changes were made in the definition of Base Offset, Advertisement, and Tagged Buffer. Changes were also made in sections 2.4.1, 2.5, 2.6, 7.3.1, 7.3.3, 7.3.5, 7.3.6, 9.1, 9.3, 9.4, 9.5.1, and 9.5.2. This change is not backward compatible with RFC 5046, but is part of all known implementations of iSER at the time this document was developed.
9. Remove iWARP specific behavior. Changes were made in the definition section on RDMA Operation and Send Message Type. Clarifications were added in section 2.4.2 on the use of SendSE and SendInvSE. These clarifications reflect a removal of the requirements in RFC 5046 for the use of these messages, as implementations have not followed RFC 5046 in this area. Changes affecting Send with Invalidate were made in sections 2.4.1, 2.5, 2.6, 4.1, and 7.3.2. Changes affecting Terminate were made in sections 10.1.2.1 and 10.1.2.2. Changes were made in section 15 to remove iWARP headers.
10. Removed denial of service descriptions for the initiator in section 5.1.1 since it is applicable for the target only.
11. Clarified in section 2.4.1 that STag invalidation is the initiator's responsibility for security reasons, and the initiator cannot rely on the target using an Invalidate version of Send. Added text in section 11 on Stag invalidation.

15 Appendix B: Message Format for iSER

This section is for information only and is NOT part of the standard.

15.1 iWARP Message Format for iSER Hello Message

The following figure depicts an iSER Hello Message encapsulated in an iWARP SendSE Message.

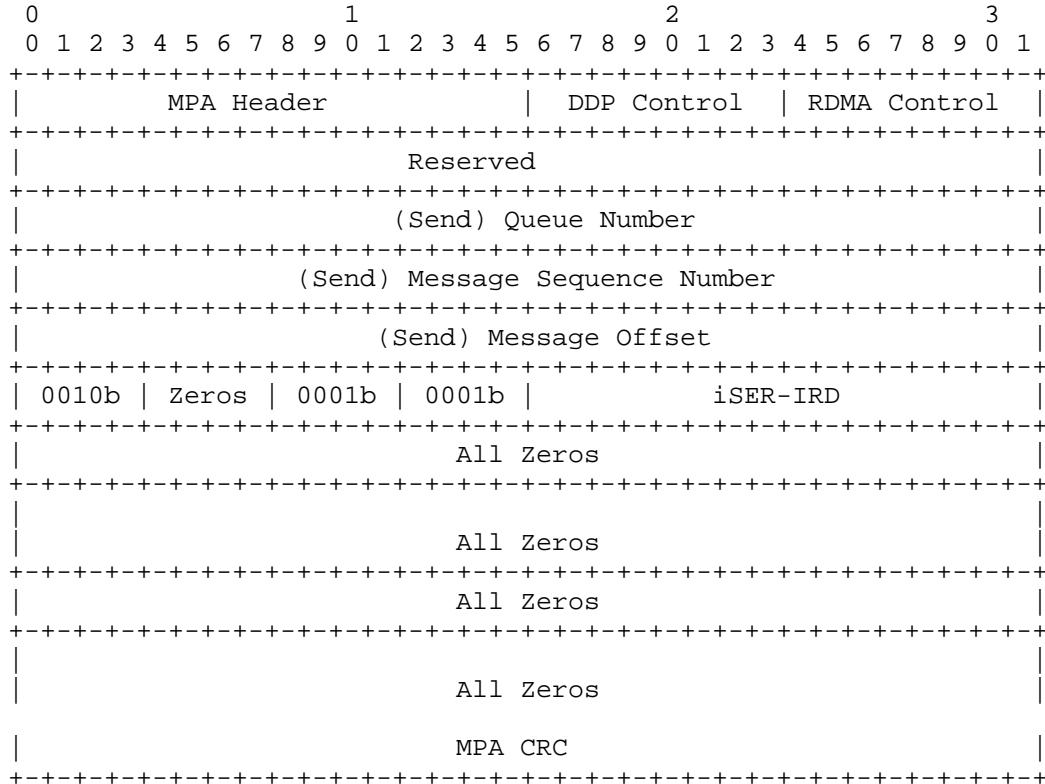


Figure 6 SendSE Message containing an iSER Hello Message

15.2 iWARP Message Format for iSER HelloReply Message

The following figure depicts an iSER HelloReply Message encapsulated in an iWARP SendSE Message. The Reject (REJ) flag is set to 0.

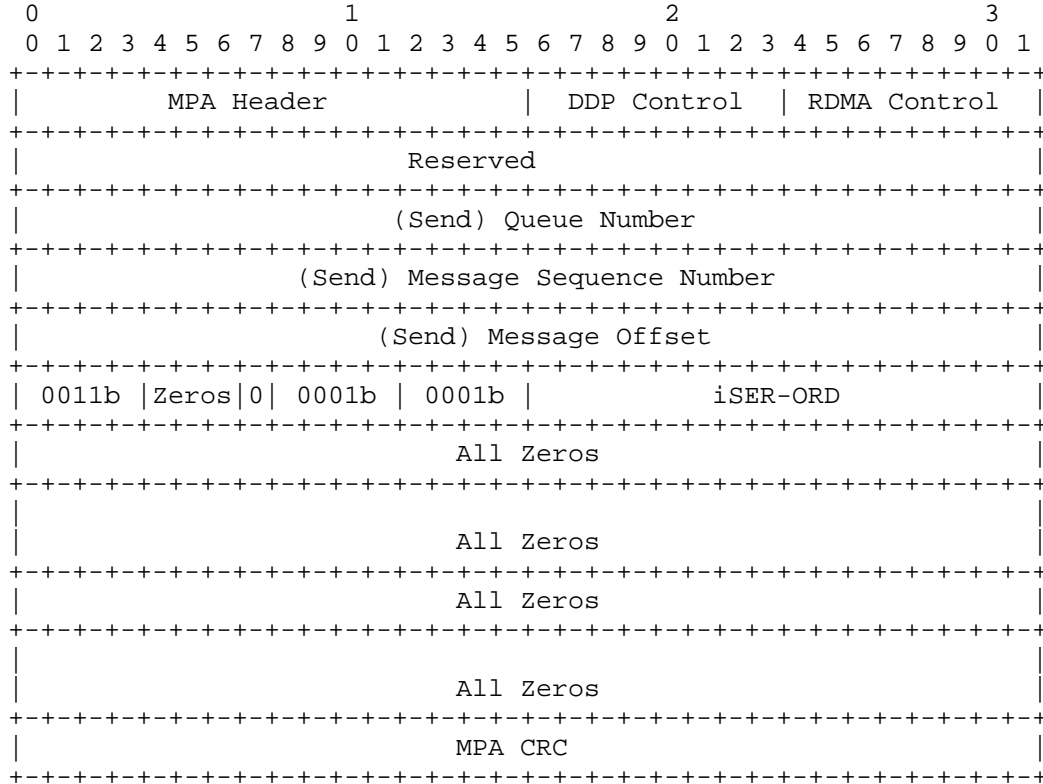


Figure 7 SendSE Message containing an iSER HelloReply Message

15.3 iSER Header Format for SCSI Read Command PDU

The following figure depicts a SCSI Read Command PDU embedded in an iSER Message. For this particular example, in the iSER header, the Write STag Valid flag is set to zero, the Read STag Valid flag is set to one, the Write STag field is set to all zeros, the Write Base Offset field is set to all zeros, the Read STag field contains a valid Read STag, and the Read Base Offset field contains a valid Base Offset for the Read Tagged Buffer.

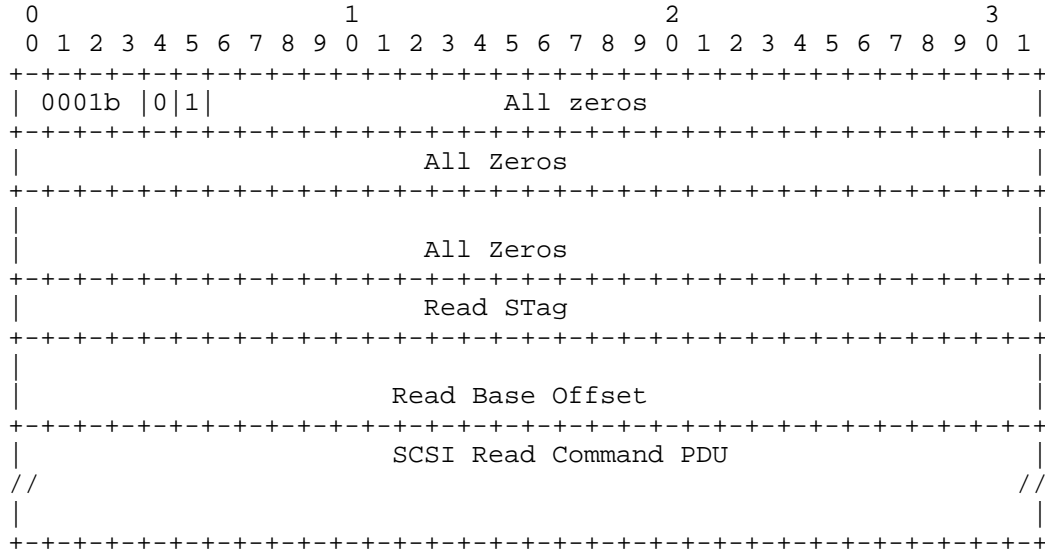


Figure 8 iSER Header Format for SCSI Read Command PDU

15.4 iSER Header Format for SCSI Write Command PDU

The following figure depicts a SCSI Write Command PDU embedded in an iSER Message. For this particular example, in the iSER header, the Write STag Valid flag is set to one, the Read STag Valid flag is set to zero, the Write STag field contains a valid Write STag, the Write Base Offset field contains a valid Base Offset for the Write Tagged Buffer, the Read STag field is set to all zeros since it is not used, and the Read Base Offset field is set to all zeros.

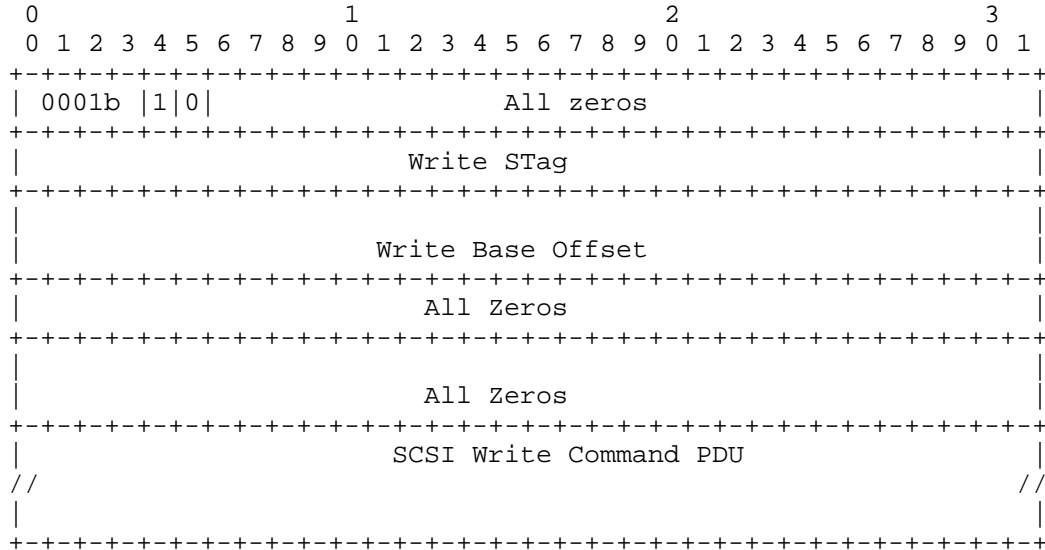


Figure 9 iSER Header Format for SCSI Write Command PDU

The following figure depicts a SCSI Response PDU embedded in an iSER Message:

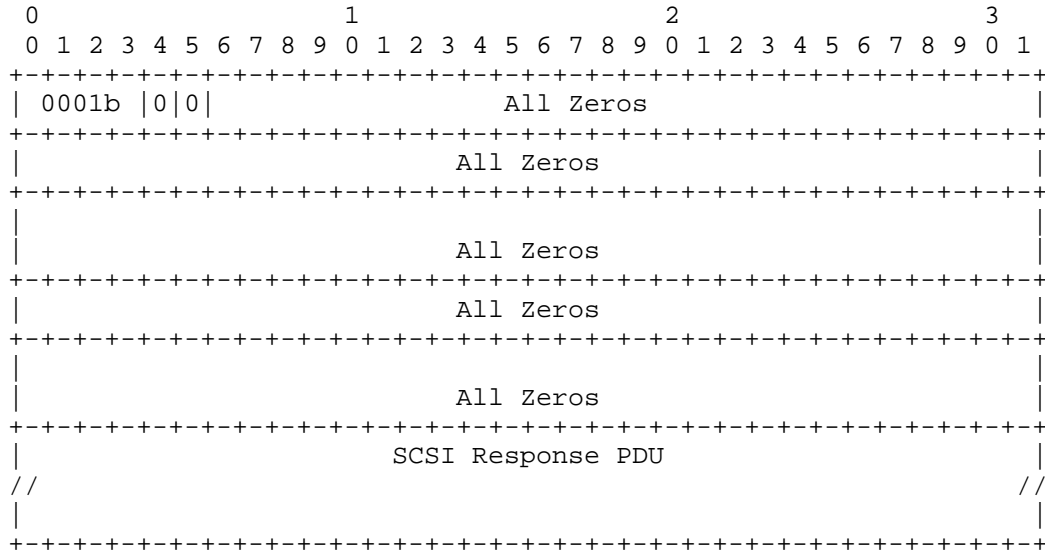


Figure 10 iSER Header Format for SCSI Response PDU

This section explains how an InfiniBand network (with Gateways) would be structured. It is informational only and is intended to provide insight on how iSER is used in an InfiniBand environment.

16.1 Host side of iSCSI & iSER connections in Infiniband

Figure 11 defines the topologies in which iSCSI and iSER will be able to operate on an InfiniBand Network.

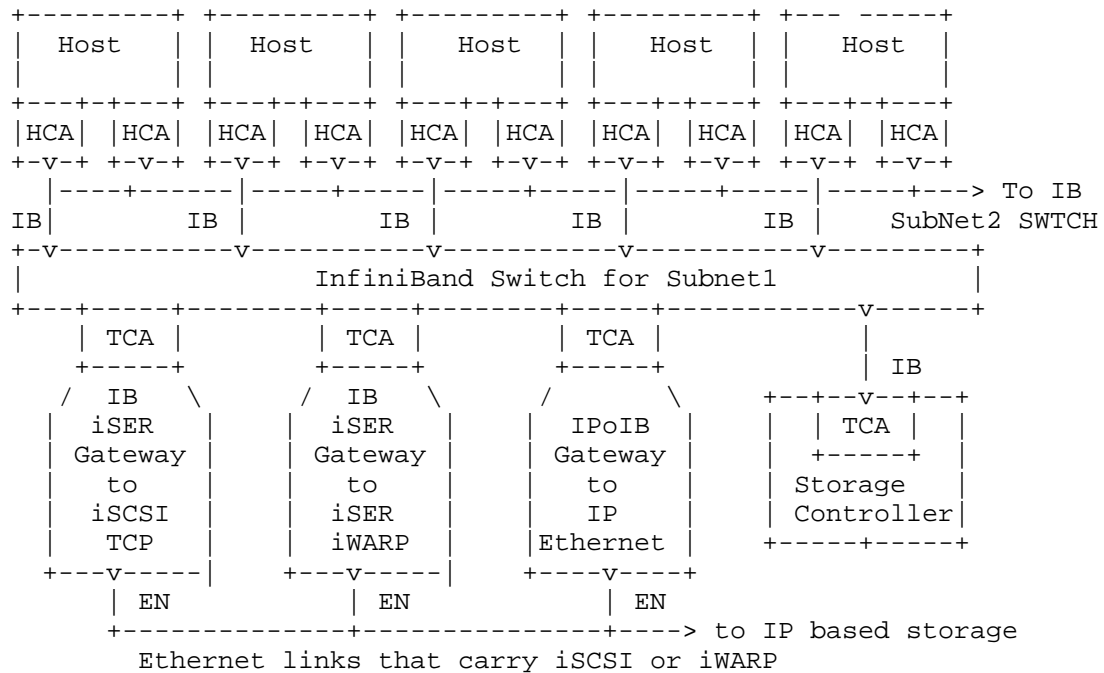


Figure 11 iSCSI and iSER on IB

In Figure 11, the Host systems are connected via the InfiniBand Host Channel Adapters (HCAs) to the InfiniBand links. With the use of IB switch(es), the InfiniBand links connect the HCA to InfiniBand Target Channel Adapters (TCAs) located in gateways or Storage Controllers. An iSER-capable IB-IP Gateway converts the iSER Messages encapsulated in IB protocols to either standard iSCSI, or iSER Messages for iWARP. An [iPOIB] Gateway converts the InfiniBand [iPOIB] protocol to IP protocol, and in the iSCSI case, permits iSCSI to be operated on an IB Network between the Hosts and the [iPOIB] Gateway.

16.2 Storage side of iSCSI & iSER mixed network environment

Figure 12 shows a storage controller that has three different portal groups: one supporting only iSCSI (TPG-4), one supporting iSER/iWARP or iSCSI (TPG-2), and one supporting iSER/IB (TPG-1).

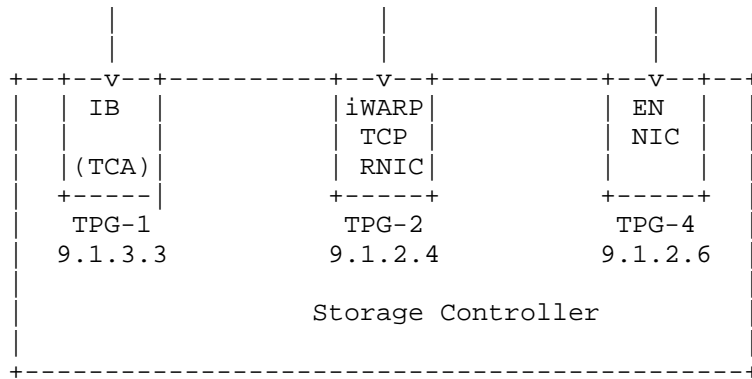


Figure 12 Storage Controller with TCP, iWARP, and IB Connections

The normal iSCSI portal group advertising processes (via SLP, iSNS, or SendTargets) are available to a Storage Controller.

16.3 Discovery processes for an InfiniBand Host

An InfiniBand Host system can gather portal group IP address from SLP, iSNS, or the SendTargets discovery processes by using TCP/IP via [IPoIB]. After obtaining one or more remote portal IP addresses, the Initiator uses the standard IP mechanisms to resolve the IP address to a local outgoing interface and the destination hardware address (Ethernet MAC or IB GID of the target or a gateway leading to the target). If the resolved interface is an [IPoIB] network interface, then the target portal can be reached through an InfiniBand fabric. In this case the Initiator can establish an iSCSI/TCP or iSCSI/iSER session with the Target over that InfiniBand interface, using the Hardware Address (InfiniBand GID) obtained through the standard Address Resolution (ARP) processes.

If more than one IP address are obtained through the discovery process, the Initiator should select a Target IP address that is on the same IP subnet as the Initiator if one exists. This will avoid a potential overhead of going through a gateway when a direct path exists.

In addition a user can configure manual static IP route entries if a particular path to the target is preferred.

16.4 IBTA Connection specifications

It is outside the scope of this document, but it is expected that the InfiniBand Trade Association (IBTA) has or will define:

- * The iSER ServiceID
- * A Means for permitting a Host to establish a connection with a peer InfiniBand end-node, and that peer indicating when that end-node supports iSER, so the Host would be able to fall back to iSCSI/TCP over [IPoIB].
- * A Means for permitting the Host to establish connections with IB iSER connections on storage controllers or IB iSER connected Gateways in preference to [IPoIB] connected Gateways/Bridges or connections to Target Storage Controllers that also accept iSCSI via [IPoIB].
- * A Means for combining the IB ServiceID for iSER and the IP port number such that the IB Host can use normal IB connection processes, yet ensure that the iSER target peer can actually connect to the required IP port number.

The authors acknowledge the following individuals for identifying implementation issues and/or suggesting resolutions to the issues clarified in this document: Robert Russell, Arne Redlich, David Black, Mallikarjun Chadalapaka, Tom Talpey, Felix Marti, Robert Sharp, Caitlin Bestler, Hemal Shah, Spencer Dawkins, Pete Resnick, Ted Lemon, Pete McCann, and Steve Kent. Credit also goes to the authors of the original iSER Specification [RFC5046], including Michael Ko, Mallikarjun Chadalapaka, John Hufferd, Uri Elzur, Hemal Shah, and Patricia Thaler. This document benefited from all of their contributions.

Author's Address

Michael Ko
Email: mkosjc@gmail.com

Alexander Nezhinsky
Mellanox Technologies
13 Zarchin St.
Raanana 43662, Israel
Phone: +972-74-712-9000
Email: alexandern@mellanox.com, nezhinsky@gmail.com

Copyright Notice

Copyright (c) 2012 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.

Storage Maintenance (storm) Working Group
Internet Draft
Intended status: Standards Track
Expires: October 2014

Hemal Shah
Broadcom Corporation
Felix Marti
Wael Nouredine
Asgeir Eiriksson
Chelsio Communications, Inc.
Robert Sharp
Intel Corporation
April 16, 2014

RDMA Protocol Extensions
draft-ietf-storm-rdmap-ext-10.txt

Status of this Memo

This Internet-Draft is submitted to IETF 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 16, 2014.

Copyright Notice

Copyright (c) 2014 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.

Abstract

This document specifies extensions to the IETF Remote Direct Memory Access Protocol (RDMA RFC5040). RDMA provides read and write services directly to applications and enables data to be transferred directly into Upper Layer Protocol (ULP) Buffers without intermediate data copies. The extensions specified in this document provide the following capabilities and/or improvements: Atomic Operations and Immediate Data.

Table of Contents

1. Introduction.....	3
1.1. Discovery of RDMA Extensions.....	4
2. Requirements Language.....	5
3. Glossary.....	5
4. Header Format Extensions.....	7
4.1. RDMA Control and Invalidate STag Fields.....	7
4.2. RDMA Message Definitions.....	9
5. Atomic Operations.....	9
5.1. Atomic Operation Details.....	11
5.1.1. FetchAdd.....	11
5.1.2. CmpSwap.....	12
5.2. Atomic Operations.....	14
5.2.1. Atomic Operation Request Message.....	14
5.2.2. Atomic Operation Response Message.....	18
5.3. Atomicity Guarantees.....	19
5.4. Atomic Operations Ordering and Completion Rules.....	19
6. Immediate Data.....	21
6.1. RDMA Interactions with ULP for Immediate Data.....	21
6.2. Immediate Data Header Format.....	22
6.3. Immediate Data or Immediate Data with SE Message.....	22
6.4. Ordering and Completions.....	23
7. Ordering and Completions Table.....	23
8. Error Processing.....	26
8.1. Errors Detected at the Local Peer.....	26
8.2. Errors Detected at the Remote Peer.....	27

9. Security Considerations.....	28
10. IANA Considerations.....	28
10.1. RDMAP Message Atomic Operation Subcodes.....	28
10.2. RDMAP Queue Numbers.....	29
11. References.....	30
11.1. Normative References.....	30
11.2. Informative References.....	31
12. Acknowledgments.....	32
Appendix A. DDP Segment Formats for RDMA Messages.....	33
A.1. DDP Segment for Atomic Operation Request.....	33
A.2. DDP Segment for Atomic Response.....	35
A.3. DDP Segment for Immediate Data and Immediate Data with SE35	

1. Introduction

The RDMA Protocol [RFC5040] provides capabilities for zero copy data communications that preserve memory protection semantics, enabling more efficient network protocol implementations. The RDMA Protocol is part of the iWARP family of specifications which also include RFC 5041 [RFC5041], RFC 5044 [RFC5044], and RFC 6581 [RFC6581]. This document specifies the following extensions to the RDMA Protocol (RDMAP):

- o Atomic operations on remote memory locations. Support for atomic operation enhances the usability of RDMAP in distributed shared memory environments.
- o Immediate Data messages allow the ULP at the sender to provide a small amount of data. When an Immediate Data message is sent following an RDMA Write Message, the combination of the two messages is an implementation of RDMA Write with Immediate message that is found in other RDMA transport protocols.

Other RDMA transport protocols define the functionality added by these extensions leading to differences in RDMA applications and/or Upper Layer Protocols. Removing these differences in the transport protocols simplifies these applications and ULPs and that is the main motivation for the extensions specified in this document.

RSockets [RSOCKETS] is an example of RDMA enabled middleware that provides a socket interface as the upper edge interface and utilizes RDMA to provide more efficient networking for sockets based applications. RSockets is aware of Immediate Data support in InfiniBand [IB]. RSockets cannot utilize the RDMA Write with

Immediate Data operation from InfiniBand . The addition of the Immediate Data operation specified in this draft will alleviate this difference in RSocketS when running on InfiniBand and iWARP.

Structured high performance computing applications based on the MPI interface [MPI] may use Atomic Operations defined in this specification. DAT Atomics [DAT_ATOMICS] is an example of RDMA enabled middleware that provides a portable RDMA programming interface for various RDMA transport protocols. DAT Atomics includes a primitive for InfiniBand that is not supported by iWARP RDMA Network Interface Controllers or RNICs. The addition of Atomic Operations as specified in this draft will allow atomic operations in DAT Atomics to work for both InfiniBand and RNICs interchangeably.

For more background on RDMA Protocol applicability, see Applicability of Remote Direct Memory Access Protocol (RDMA) and Direct Data Placement Protocol (DDP) [RFC5045].

1.1. Discovery of RDMAP Extensions

Today there are RDMA applications and/or ULPs that are aware of the existence of Atomic and Immediate data operations for RDMA transports such as InfiniBand and application programming interfaces such as Open Fabrics Verbs [OFAVERBS]. Today, these applications need to be aware that RDMAP does not support certain of these operations. Typically the availability of these capabilities is exposed to the applications through adapter query interfaces in software. Applications then have to decide to use or not to use Immediate Data or Atomic Operations based on the results of the query interfaces. Such query interfaces typically return the scope of atomicity guarantees, not the individual Atomic Operations supported. Therefore, this specification requires all Atomic Operations defined within to be supported if an RNIC supports any Atomic Operations.

In cases where heterogeneous hardware, with differing support for Atomic Operations and Immediate Data Operations, is deployed for use by RDMA applications and/or ULPs, applications are either statically configured to use or not use optional features or use application specific negotiation mechanisms. For the extensions covered by this document, it is RECOMMENDED that RDMA applications and/or ULPs negotiate at the application or ULP level the usage of these extensions. The definition of such application specific mechanism is outside the scope of this specification. For backward

compatibility, existing applications and/or ULPs should not assume that these extensions are supported.

In the absence of application specific negotiation of the features defined within this specification, the new operations can be attempted and reported errors can be used to determine a remote peer's capabilities. In the case of Atomics, a FetchAdd operation with Add Data set to 0 can safely be used to determine the existence of Atomic Operations without modifying the content of a remote peer's memory. A Remote Operation Error / Unexpected OpCode error will be reported by the remote peer in the case of an Immediate Data or Atomic Operation as described if not supported by the remote peer.

2. 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 RFC-2119 [RFC2119].

3. Glossary

This document is an extension of RFC 5040 and key words are defined in the glossary of the referenced document.

Atomic Operation - is an operation that results in an execution of a memory operation at a specific ULP Buffer address on a remote node using the Tagged Buffer data transfer model. The consumer can use Atomic Operations to read, modify and write memory at the destination ULP Buffer address while at the same time guaranteeing that no other Atomic Operation read or write accesses to the ULP Buffer address targeted by the Atomic Operation will occur across any other RDMAP Streams on an RNIC at the Responder.

Atomic Operation Request - An RDMA Message used by the Data Source to perform an Atomic Operation at the Responder.

Atomic Operation Response - An RDMA Message used by the Responder to describe the completion of an Atomic Operation at the Responder.

CmpSwap - is an Atomic Operation that is used to compare and swap a value at a specific address on a remote node.

FetchAdd - is an Atomic Operation that is used to atomically increment a value at a specific ULP Buffer address on a remote node.

Immediate Data - a small fixed size portion of data sent from the Data Source to a Data Sink

Immediate Data Message - An RDMA Message used by the Data Source to send Immediate Data to the Data Sink

Immediate Data with Solicited Event (SE) Message - An RDMA Message used by the Data Source to send Immediate Data with Solicited Event to the Data Sink

iWARP - A suite of wire protocols comprised of RFC 5040, RFC 5041, RFC 5044, and RFC 6581.

Requester - the sender of an RDMA Atomic Operation request.

Responder - the receiver of an RDMA Atomic Operation request.

RNIC - RDMA Network Interface Controller. In this context, this would be a network I/O adapter or embedded controller with iWARP functionality.

ULP - Upper Layer Protocol. The protocol layer above the one currently being referenced. The ULP for RFC 5040 / RFC 5041 is expected to be an OS, Application, adaptation layer, or proprietary device. The RFC 5040 / RFC 5041 documents do not specify a ULP -- they provide a set of semantics that allow a ULP to be designed to utilize RFC 5040 / RFC 5041.

4. Header Format Extensions

The control information of RDMA Messages is included in DDP protocol RFC 5041 defined header fields. RFC 5040 defines the RDMAP header formats layered on the DDP header definition. This specification extends RFC 5040 with the following new formats:

- . Four new RDMA Messages carry additional RDMAP headers. The Immediate Data operation and Immediate Data with Solicited Event operation include 8 bytes of data following the RDMAP header. Atomic Operations include Atomic Request or Atomic Response headers following the RDMAP header. The RDMAP header for Atomic Request messages is 52 bytes long as specified in Figure 4. The RDMAP header for Atomic Response Messages is 32 bytes long as specified in Figure 5.
- . Introduction of a new queue for untagged buffers (QN=3) used for Atomic Response tracking.

4.1. RDMAP Control and Invalidate STag Fields

For reference, Figure 1 depicts the format of the DDP Control and RDMAP Control fields, in the style and convention of RFC 5040:

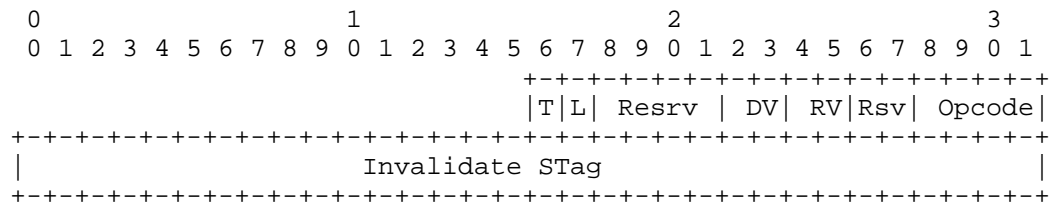


Figure 1 DDP Control and RDMAP Control Fields

The DDP Control Field consists of the T,L, Resrv and DV fields RFC 5041. The RDMAP Control Field consists of the RV, Rsv and Opcode fields RFC 5040.

This specification adds additional values for the RDMA Opcode field to those specified in RFC 5040. Figure 2 defines the new values of RDMA Opcode field that are used for the RDMA Messages defined in this specification.

Figure 2As shown in Figure 2, STag and Tagged Offset are not applicable for the RDMA Messages defined in this specification. Figure 2 also shows the appropriate Queue Number for each Opcode.

All RDMA Messages defined in this specification MUST have:

The RDMA Version (RV) field: 01b.

Opcode field: Set to one of the values in Figure 2.

Invalidate STag: Set to zero by the sender, ignored by the receiver.

RDMA Opcode	Message Type	Tagged Flag	STag and TO	Queue Number	In- validate STag	Message Length Communicated between DDP and RDMAP
1000b	Immediate Data	0	N/A	0	N/A	Yes
1001b	Immediate Data with SE	0	N/A	0	N/A	Yes
1010b	Atomic Request	0	N/A	1	N/A	Yes
1011b	Atomic Response	0	N/A	3	N/A	Yes

Figure 2 Additional RDMA Usage of DDP Fields

Note: N/A means Not Applicable.

This extension defines RDMAP use of Queue Number 3 for Untagged Buffers for Atomic Responses. This queue is used for tracking outstanding Atomic Requests.

All other DDP and RDMAP control fields are set as described in RFC 5040.

4.2. RDMA Message Definitions

The following figure defines which RDMA Headers are used on each new RDMA Message and which new RDMA Messages are allowed to carry ULP payload:

RDMA Message OpCode	Message Type	RDMA Header Used	ULP Message allowed in the RDMA Message
1000b	Immediate Data	Immediate Data Header	No
1001b	Immediate Data with SE	Immediate Data Header	No
1010b	Atomic Request	Atomic Request Header	No
1011b	Atomic Response	Atomic Response Header	No

Figure 3 RDMA Message Definitions

5. Atomic Operations

The RDMA Protocol Specification in RFC 5040 does not include support for Atomic Operations which are an important building block for implementing distributed shared memory.

This document extends the RDMA Protocol specification with a set of basic Atomic Operations, and specifies their resource and ordering rules. The Atomic Operations specified in this document provide equivalent functionality to the InfiniBand RDMA transport as well as extended Atomic Operations defined in Open Fabrics Verbs, to allow applications that use these primitives to work interchangeably over iWARP. Other operations are left for future consideration.

Atomic operations as specified in this document execute a 64-bit memory operation at a specified destination ULP Buffer address on a Responder node using the Tagged Buffer data transfer model. The operations atomically read, modify and write back the contents of the destination ULP Buffer address and guarantee that Atomic Operations on this ULP Buffer address by other RDMAP Streams on the same RNIC do not occur between the read and the write caused by the Atomic Operation. Therefore, the Responder RNIC MUST implement mechanisms to prevent Atomic Operations to a memory registered for Atomic Operations while an Atomic Operation targeting the memory is in progress. The Requester of an atomic operation cannot rely on atomic operation behavior at the Responder across multiple RNICs or with respect to other applications/ULPs running at the Responder that can access the ULP Buffer. It is OPTIONAL for an RNIC to provide such behavior when implementing the atomic operations specified in this document. An RNIC that supports Atomic Operations as specified in this document MUST implement both the FetchAdd operation as specified in section 5.1.1 and CmpSwap operation as specified in section 5.1.2. The advertisement of Tagged Buffer information for Atomic Operations is outside the scope of this specification and is handled by the ULPs.

Implementation note: It is RECOMMENDED that the applications do not use the ULP Buffer addresses used for Atomic Operations for other RDMA operations due to the lack of atomicity guarantees between operations other than Atomic Operations.

Implementation note: Errors related to the alignment in the following sections cover Atomic Operations targeted at a ULP Buffer address that is not aligned to a 64-bit boundary.

Atomic Operation Request Messages use the same remote addressing mechanism as RDMA Reads and Writes. The ULP Buffer address specified in the request is in the address space of the Remote Peer to which the Atomic Operation is targeted.

Atomic Operation Response Messages MUST use the Untagged Buffer model with QN=3. Queue number 3 will be used to track outstanding Atomic Operation Request messages at the Requestor. When the Atomic Operation Response message is received, the MSN will be used to locate the corresponding Atomic Operation request in order to complete the Atomic Operation request.

5.1. Atomic Operation Details

The following sub-sections describe the Atomic Operations in more details.

5.1.1. FetchAdd

The FetchAdd Atomic Operation requests the Responder to read a 64-bit Original Remote Data Value at a 64-bit aligned ULP Buffer address in the Responder's memory, to perform FetchAdd operation on multiple fields of selectable length specified by 64-bit "Add Mask", and write the result back to the same ULP Buffer address. The Atomic addition is performed independently on each one of these fields. A bit set in the Add Mask field specifies the field boundary; for each field, a bit is set at the most significant bit position for each field, causing any carry out of that bit position to be discarded when the addition is performed.

FetchAdd Atomic Operations MUST target ULP Buffer addresses that are 64-bit aligned. FetchAdd Atomic Operations that target ULP Buffer addresses that are not 64-bit aligned MUST be surfaced as errors and the Responder's memory MUST NOT be modified in such cases. Additionally an error MUST be surfaced and a terminate message MUST be generated. The setting of "Add Mask" field to 0x0000000000000000 results in Atomic Add of 64-bit Original Remote Data Value and 64-bit "Add Data".

The pseudo code below describes masked FetchAdd Atomic Operation.

```
bit_location = 1
carry = 0
Remote Data Value = 0
for bit = 0 to 63
{
    if (bit != 0 ) bit_location = bit_location << 1
    val1 = (Original Remote Data Value & bit_location) >> bit
    val2 = (Add Data & bit_location) >> bit
```

```
    sum = carry + val1 + val2

    carry = (sum & 2) >> 1

    sum = sum & 1

    if (sum)

        Remote Data Value |= bit_location

    carry = ((carry) && (!(Add Mask & bit_location)))

}
```

The FetchAdd operation is performed in the endian format of the target memory. The "Original Remote Data Value" is converted from the endian format of the target memory for return and returned to the Requester. The fields are in big-endian format on the wire.

The Requester specifies:

- o Remote STag
- o Remote Tagged Offset
- o Add Data
- o Add Mask

The Responder returns:

- o Original Remote Data

5.1.2. CmpSwap

The CmpSwap Atomic Operation requires the Responder to read a 64-bit value at a 64-bit aligned ULP Buffer address in the Responder's memory, to perform an AND logical operation using the 64 bit "Compare Mask" field in the Atomic Operation Request header, then to compare it with the result of a logical AND operation of the "Compare Mask" and the "Compare Data" fields in the header, and, if the two values are equal, to swap masked bits in the same ULP Buffer address with the masked Swap Data. If the two masked compare values are not equal, the contents of the Responder's memory are not changed. In either case, the original value read from the ULP Buffer

address is converted from the endian format of the target memory for return and returned to the Requester. The fields are in big-endian format on the wire.

The Requester specifies:

- o Remote STag
- o Remote Tagged Offset
- o Swap Data
- o Swap Mask
- o Compare Data
- o Compare Mask

The Responder returns:

- o Original Remote Data Value

The following pseudo code describes the masked CmpSwap operation result.

```
if (!((Compare Data ^ Original Remote Data Value) &
      Compare Mask))
then
    Remote Data Value =
        (Original Remote Data Value & ~(Swap Mask))
        | (Swap Data & Swap Mask)
else
    Remote Data Value = Original Remote Data Value
```

After the operation, the remote data buffer MUST contain the "Original Remote Data Value" (if comparison did not match) or the masked "Swap Data" (if the comparison did match). CmpSwap Atomic Operations MUST target ULP Buffer addresses that are 64-bit aligned.

If a CmpSwap Atomic Operation is attempted on a target ULP Buffer address that is not 64-bit aligned:

- o The operation MUST NOT be performed,
- o The Responder's memory MUST NOT be modified,
- o The result MUST be surfaced as an error, and
- o A terminate message MUST be generated (see Section 8.2. for the terminate message contents)

5.2. Atomic Operations

The Atomic Operation Request and Response are RDMA Messages. An Atomic Operation makes use of the DDP Untagged Buffer Model. Atomic Operation Request messages MUST use the same Queue Number as RDMA Read Requests (QN=1). Reusing the same Queue Number for Atomic Request messages allows the Atomic Operations to reuse the same infrastructure (e.g. ORD/IRD flow control) as defined for RDMA Read Requests. Atomic Operation Response messages MUST set Queue Number (QN) to 3 in the DDP header.

The RDMA Message OpCode for an Atomic Request Message is 1010b. The RDMA Message OpCode for an Atomic Response Message is 1011b.

5.2.1. Atomic Operation Request Message

The Atomic Operation Request Message carries an Atomic Operation Header that describes the ULP Buffer address in the Responder's memory. The Atomic Operation Request header immediately follows the DDP header. The RDMAP layer passes to the DDP layer a RDMAP Control Field. The following figure depicts the Atomic Operation Request Header that is used for all Atomic Operation Request Messages:

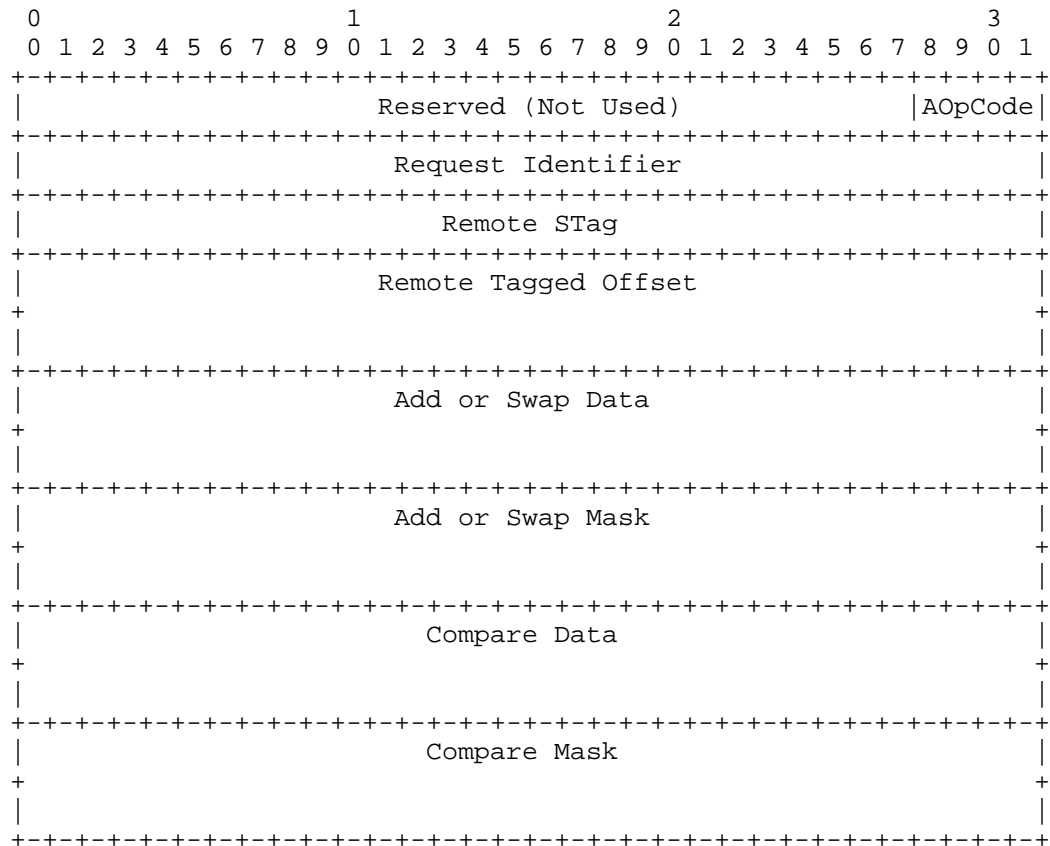


Figure 4 Atomic Operation Request Header

Reserved (Not Used): 28 bits

This field is set to zero on transmit, ignored on receive.

Atomic Operation Code (AOpCode): 4 bits.

See Figure 5. All Atomic Operation Codes from Figure 5 MUST be implemented by an RNIC that supports Atomic Operations.

Request Identifier: 32 bits.

The Request Identifier specifies a number that is used to identify Atomic Operation Request Message. The value used in this field is selected by the RNIC that sends the message, and is reflected back to the Local Peer in the Atomic Operation Response message.

Remote STag: 32 bits.

The Remote STag identifies the Remote Peer's Tagged Buffer targeted by the Atomic Operation. The Remote STag is associated with the RDMAP Stream through a mechanism that is outside the scope of the RDMAP specification.

Remote Tagged Offset: 64 bits.

The Remote Tagged Offset specifies the starting offset, in octets, from the base of the Remote Peer's Tagged Buffer targeted by the Atomic Operation. The Remote Tagged Offset MAY start at an arbitrary offset but MUST represent a 64-bit aligned ULP Buffer address.

Add or Swap Data: 64 bits.

The Add or Swap Data field specifies the 64-bit "Add Data" value in an Atomic FetchAdd Operation or the 64-bit "Swap Data" value in an Atomic Swap or CmpSwap Operation.

Add or Swap Mask: 64 bits

This field is used in masked Atomic Operations (FetchAdd and CmpSwap) to perform a bitwise logical AND operation as specified in the definition of these operations. For non-masked Atomic Operations (Swap), this field is set to ffffffffffffffffh on transmit and ignored by the receiver.

Compare Data: 64 bits.

The Compare Data field specifies the 64-bit "Compare Data" value in an Atomic CmpSwap Operation. For Atomic FetchAdd and Atomic Swap operation, the Compare Data field is set to zero on transmit and ignored by the receiver.

Compare Mask: 64 bits

This field is used in masked Atomic Operation CmpSwap to perform a bitwise logical AND operation as specified in the definition of these operations. For Atomic Operations FetchAdd and Swap, this field is set to ffffffffh on transmit and ignored by the receiver.

Atomic Operation Code	Atomic Operation	Add or Swap Data	Add or Swap Mask	Compare Data	Compare Mask
0000b	FetchAdd	Add Data	Add Mask	N/A	N/A
0010b	CmpSwap	Swap Data	Swap Mask	Valid	Valid

Figure 5 Atomic Operation Message Definitions

The Atomic Operation Request Message has the following semantics:

1. An Atomic Operation Request Message MUST reference an Untagged Buffer. That is, the Local Peer's RDMAP layer MUST request that the DDP mark the Message as Untagged.
2. One Atomic Operation Request Message MUST consume one Untagged Buffer.
3. The Responder's RDMAP layer MUST process an Atomic Operation Request Message. A valid Atomic Operation Request Message MUST NOT be delivered to the Responder's ULP (i.e., it is processed by the RDMAP layer).
4. At the Responder, an error MUST be surfaced in response to delivery to the Remote Peer's RDMAP layer of an Atomic Operation Request Message with an Atomic Operation Code that the RNIC does not support.
5. An Atomic Operation Request Message MUST reference the RDMA Read Request Queue. That is, the Requester's RDMAP layer MUST request that the DDP layer set the Queue Number field to one.
6. The Requester MUST pass to the DDP layer Atomic Operation Request Messages in the order they were submitted by the ULP.

2. An Atomic Operation Response Message MUST consume an Untagged Buffer. That is, the Responder RDMAP layer MUST request that the DDP mark the Message as Untagged.
3. An Atomic Operation Response Message MUST reference the Queue Number 3. That is, the Responder's RDMAP layer MUST request that the DDP layer set the Queue Number field to 3.
4. The Responder MUST ensure that a sufficient number of Untagged Buffers are available on the RDMA Read Request Queue (Queue with DDP Queue Number 1) to support the maximum number of Atomic Operation Requests negotiated by the ULP in addition to the maximum number of RDMA Read Requests negotiated by the ULP.
5. The Requester MUST ensure that a sufficient number of Untagged Buffers are available on the RDMA Atomic Response Queue (Queue with DDP Queue Number 3) to support the maximum number of Atomic Operation Requests negotiated by the ULP.
6. The RDMAP layer MUST Deliver the Atomic Operation Response Message to the ULP.
7. At the Requester, when an invalid Atomic Operation Response Message is delivered to the Remote Peer's RDMAP layer, an error is surfaced.
8. When the Responder receives Atomic Operation Request messages, the Responder RDMAP layer MUST pass Atomic Operation Response Messages to the DDP layer, in the order that the Atomic Operation Request Messages were received by the RDMAP layer, at the Responder.

5.3. Atomicity Guarantees

Atomicity of the Read-Modify-Write (RMW) on the Responder's node by the Atomic Operation MUST be assured in the context of concurrent atomic accesses by other RDMA Streams on the same RNIC.

5.4. Atomic Operations Ordering and Completion Rules

In addition to the ordering and completion rules described in RFC 5040, the following rules apply to implementations of the Atomic operations.

1. For an Atomic operation, the Requester MUST NOT consider the contents of the Tagged Buffer at the Responder to be modified by that specific Atomic Operation until the Atomic Operation Response Message has been Delivered to RDMAP at the Requester.
2. Atomicity guarantees MUST be provided within the scope of a single RNIC.

Implementation Note: This requirement for atomicity among operations is limited to the scope of a single RNIC. Atomicity guarantees are OPTIONAL with respect to access to the Tagged Buffer by any other method than an Atomic Operation via the same RNIC. Examples of such accesses that may not be atomic with respect to an Atomic Operation include accesses via other RNICs and local processor memory access to the Tagged Buffer.

3. Atomic Operation Request Messages MUST NOT start processing at the Responder until they have been Delivered to RDMAP by DDP.
4. Atomic Operation Response Messages MAY be generated at the Responder after subsequent RDMA Write Messages or Send Messages have been Placed or Delivered.
5. Atomic Operation Response Message processing at the Responder MUST be started only after the Atomic Operation Request Message has been Delivered by the DDP layer (thus, all previous RDMA Messages on that DDP Stream have been Delivered).
6. Send Messages MAY be Completed at the Responder before prior incoming Atomic Operation Request Messages have completed their response processing.
7. An Atomic Operation MUST NOT be Completed at the Requester until the DDP layer Delivers the associated incoming Atomic Operation Response Message.
8. If more than one outstanding Atomic Request Messages are supported by both peers, the Atomic Operation Request Messages MUST be processed in the order they were delivered by the DDP layer on the Responder. Atomic Operation Response Messages MUST be submitted to the DDP layer on the Responder in the order the Atomic Operation Request Messages were Delivered by DDP.

6. Immediate Data

The Immediate Data operation is typically used in conjunction with an RDMA Write Operation to improve ULP processing efficiency. The efficiency is gained by causing an RDMA Completion to be generated immediately following the RDMA Write operation. This RDMA Completion delivers 8 bytes of immediate data at the Remote Peer. The combination of an RDMA Write Message followed by an Immediate Data Operation has the same behavior as the RDMA Write with Immediate Data operation found in InfiniBand. An Immediate Data operation that is not preceded by an RDMA Write operation causes an RDMA Completion.

6.1. RDMAP Interactions with ULP for Immediate Data

For Immediate Data operations, the following are the interactions between the RDMAP Layer and the ULP:

- . At the Data Source:
 - . The ULP passes to the RDMAP Layer the following:
 - . Eight bytes of ULP Immediate Data
 - . When the Immediate Data operation Completes, an indication of the Completion results.
- . At the Data Sink:
 - . If the Immediate Data operation is Completed successfully, the RDMAP Layer passes the following information to the ULP Layer:
 - . Eight bytes of Immediate Data
 - . An Event, if the Data Sink is configured to generate an Event.
 - . If the Immediate Data operation is Completed in error, the Data Sink RDMAP Layer will pass up the corresponding error information to the Data Sink ULP and send a Terminate Message to the Data Source RDMAP Layer. The Data Source RDMAP Layer will then pass up the Terminate Message to the ULP.

6.2. Immediate Data Header Format

The Immediate Data and Immediate Data with SE Messages carry immediate data as shown in Figure 7. The RDMAP layer passes to the DDP layer an RDMAP Control Field and 8 bytes of Immediate Data. The first 8 bytes of the data following the DDP header contains the Immediate Data. See section A.3. for the DDP segment format of an Immediate Data or Immediate Data with SE Message.

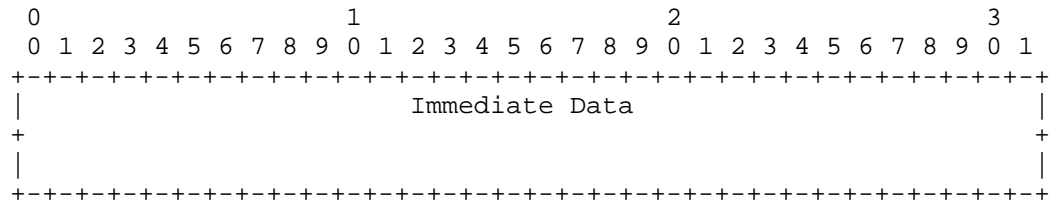


Figure 7 Immediate Data or Immediate Data with SE Message Header

Immediate Data: 64 bits.

Eight bytes of data transferred from the Data Source to an untagged buffer at the Data Sink.

6.3. Immediate Data or Immediate Data with SE Message

The Immediate Data or Immediate Data with SE Message uses the DDP Untagged Buffer Model to transfer Immediate Data from the Data Source to the Data Sink.

- . An Immediate Data or Immediate Data with SE Message MUST reference an Untagged Buffer. That is, the Local Peer's RDMAP Layer MUST request that the DDP layer mark the Message as Untagged.
- . One Immediate Data or Immediate Data with SE Message MUST consume one Untagged Buffer.
- . At the Remote Peer, the Immediate Data or Immediate Data with SE Message MUST be Delivered to the Remote Peer's ULP in the order they were sent.

- . For an Immediate Data or Immediate Data with SE Message, the Local Peer's RDMAP Layer MUST request that the DDP layer set the Queue Number field to zero.
- . For an Immediate Data or Immediate Data with SE Message, the Local Peer's RDMAP Layer MUST request that the DDP layer transmit 8 bytes of data.
- . The Local Peer MUST issue Immediate Data and Immediate Data with SE Messages in the order they were submitted by the ULP.
- . The Remote Peer MUST check that Immediate Data and Immediate Data with SE Messages include exactly 8 bytes of data from the DDP layer. The DDP header carries the length field that is reported by the DDP layer.

6.4. Ordering and Completions

Ordering and completion rules for Immediate Data are the same as those for a Send operation as described in section 5.5 of RFC 5040.

7. Ordering and Completions Table

The following table summarizes the ordering relationships for Atomic and Immediate Data operations from the standpoint of Local Peer issuing the Operations. Note that in the table that follows, Send includes Send, Send with Invalidate, Send with Solicited Event, and Send with Solicited Event and Invalidate. Also note that in the table below, Immediate Data includes Immediate Data and Immediate Data with Solicited Event.

First Operation	Second Operation	Placement Guarantee at Remote Peer	Placement Guarantee at Local Peer	Ordering Guarantee at Remote Peer
Immediate Data	Send	No Placement Guarantee between Send Payload and Immediate Data	Not Applicable	Completed in Order

Immediate Data	RDMA Write	No Placement Guarantee between RDMA Write Payload and Immediate Data	Not Applicable	Not Applicable
Immediate Data	RDMA Read	No Placement Guarantee between Immediate Data and RDMA Read Request	RDMA Read Response will not be Placed until Immediate Data is Placed at Remote Peer	RDMA Read Response Message will not be generated until Immediate Data has been Completed
Immediate Data	Atomic	No Placement Guarantee between Immediate Data and Atomic Request	Atomic Response will not be Placed until Immediate Data is Placed at Remote Peer	Atomic Response Message will not be generated until Immediate Data has been Completed
Immediate Data or Send	Immediate Data	No Placement Guarantee	Not Applicable	Completed in Order
RDMA Write	Immediate Data	No Placement Guarantee	Not Applicable	Immediate Data is Completed after RDMA Write is Placed and Delivered
RDMA Read	Immediate Data	No Placement Guarantee between Immediate Data and RDMA Read Request	Immediate Data MAY be Placed before RDMA Read Response is generated	Not Applicable

Atomic	Immediate Data	No Placement Guarantee between Immediate Data and Atomic Request	Immediate Data MAY be Placed before Atomic Response is generated	Not Applicable
Atomic	Send	No Placement Guarantee between Send Payload and Atomic Request	Send Payload MAY be Placed before Atomic Response is generated	Not Applicable
Atomic	RDMA Write	No Placement Guarantee between RDMA Write Payload and Atomic Request	RDMA Write Payload MAY be Placed before Atomic Response is generated	Not Applicable
Atomic	RDMA Read	No Placement Guarantee between Atomic Request and RDMA Read Request	No Placement Guarantee between Atomic Response and RDMA Read Response	RDMA Read Response Message will not be generated until Atomic Response Message has been generated
Atomic	Atomic	Placed in order	No Placement Guarantee between two Atomic Responses	Second Atomic Request Message will not be processed until first Atomic Response has been generated

Send	Atomic	No Placement Guarantee between Send Payload and Atomic Request	Atomic Response will not be Placed at the Local Peer Until Send Payload is Placed at the Remote Peer	Atomic Response Message will not be generated until Send has been Completed
RDMA Write	Atomic	No Placement Guarantee between RDMA Write Payload and Atomic Request	Atomic Response will not be Placed at the Local Peer Until RDMA Write Payload is Placed at the Remote Peer	Not Applicable
RDMA Read	Atomic	No Placement Guarantee between Atomic Request and RDMA Read Request	No Placement Guarantee between Atomic Response and RDMA Read Response	Atomic Response Message will not be generated until RDMA Read Response has been generated

8. Error Processing

In addition to error processing described in section 7 of RFC 5040, the following rules apply for the new RDMA Messages defined in this specification.

8.1. Errors Detected at the Local Peer

The Local Peer MUST send a Terminate Message for each of the following cases:

1. For errors detected while creating an Atomic Request, Atomic Response, Immediate Data, or Immediate Data with SE Message, or other reasons not directly associated with an incoming Message, the Terminate Message and Error code are sent instead of the Message. In this case, the Error Type and Error Code fields are included in the Terminate Message, but the Terminated DDP Header and Terminated RDMA Header fields are set to zero.
2. For errors detected on an incoming Atomic Request, Atomic Response, Immediate Data, or Immediate Data with Solicited Event (after the Message has been Delivered by DDP), the Terminate Message is sent at the earliest possible opportunity, preferably in the next outgoing RDMA Message. In this case, the Error Type, Error Code, and Terminated DDP Header fields are included in the Terminate Message, but the Terminated RDMA Header field is set to zero.

8.2. Errors Detected at the Remote Peer

On incoming Atomic Requests, Atomic Responses, Immediate Data, and Immediate Data with Solicited Event, the following MUST be validated:

- . The DDP layer MUST validate all DDP Segment fields.
- . The RDMA OpCode MUST be valid.
- . The RDMA Version MUST be valid.

On incoming Atomic requests the following additional validation MUST be performed:

- . The RDMAP layer MUST validate that the Remote Peer's Tagged ULP Buffer address references a 64-bit aligned ULP Buffer address. In the case of an error, the RDMAP layer MUST generate a Terminate Message indicating RDMA Layer Remote Operation Error with Error Code Name "Catastrophic Error, Localized to RDMAP Stream" as described in Section 4.8 of RFC 5040. Implementation Note: A ULP implementation can avoid this error by having the target ULP buffer of an atomic operation 64-bit aligned.

9. Security Considerations

This document specifies extensions to the RDMA Protocol specification in RFC 5040, and as such the Security Considerations discussed in Section 8 of RFC 5040 apply. In particular, Atomic Operations use ULP Buffer addresses for the Remote Peer buffer addressing used in RFC 5040 as required by the RFC 5042 [RFC5042] security model.

RDMAP and related protocols may be used by applications that exhibit distinctive traffic characteristics such as message timing, source, destination and size patterns. Examples include structured high performance computing applications based on the MPI interface. For such applications, analysis of encrypted traffic could reveal sensitive information, e.g., the nature of the application, size of data set being used, and information about the application's rate of progress. Such information can be hidden from passive observation via use of ESPv3 Traffic Flow Confidentiality [RFC4303] to obfuscate the encrypted traffic's characteristics. ESPv3 implementation requirements for RDMAP are specified in [RFC7146].

10. IANA Considerations

IANA is requested to add the following entries to the "RDMAP Message Operation Codes" registry of "RDDP Registries":

0x8, Immediate Data, [RFCXXXX]

0x9, Immediate Data with Solicited Event, [RFCXXXX]

0xA, Atomic Request, [RFCXXXX]

0xB, Atomic Response, [RFCXXXX]

In addition, the following registry is requested to be added to "RDDP Registries". The following section specifies the registry, its initial contents and the administration policy in more detail.

RFC Editor: Please replace XXXX in all instances of [RFCXXXX] above with the RFC number of this document and remove this note.

10.1. RDMAP Message Atomic Operation Subcodes

Name of the registry: "RDMAP Message Atomic Operation Subcodes"

Namespace details: RDMA Message Atomic Operation Subcodes are 4-bit values [RFCXXXX].

Information that must be provided to assign a new value: An IESG-approved standards-track specification defining the semantics and interoperability requirements of the proposed new value and the fields to be recorded in the registry.

Fields to record in the registry: RDMA Message Atomic Operation Subcode, Atomic Operation, RFC Reference.

Initial registry contents:

0x0, FetchAdd, [RFCXXXX]

0x1, Reserved

0x2, CmpSwap, [RFCXXXX]

Note: An experimental RDMA Message Operation Code has already been allocated; hence there is no need for an experimental RDMA Message Atomic Operation Subcode.

All other values are Unassigned and available to IANA for assignment. New RDMA Message Atomic Operation Subcodes should be assigned sequentially in order to better support implementations that process RDMA Message Atomic Operations in hardware.

Allocation Policy: Standards Action ([RFC5226])

RFC Editor: Please replace XXXX in all instances of [RFCXXXX] above with the RFC number of this document and remove this note.

10.2. RDMA Queue Numbers

Name of the registry: "RDMA DDP Untagged Queue Numbers"

Namespace details: RDMA DDP Untagged Queue numbers are 32-bit values [RFCXXXX].

Information that must be provided to assign a new value: An IESG-approved standards-track specification defining the semantics and interoperability requirements of the proposed new value and the fields to be recorded in the registry.

Fields to record in the registry: RDMAP DDP Untagged Queue Numbers, Queue Usage Description, RFC Reference.

Initial registry contents:

0x00000000, Queue 0 (Send operation Variants), [RFC5040]

0x00000001, Queue 1 (RDMA Read Request operations), [RFC5040]

0x00000002, Queue 2 (Terminate operations), [RFC5040]

0x00000003, Queue 3 (Atomic Response operations), [RFCXXXX]

Note: An experimental RDMAP Message Operation Code has already been allocated; hence there is no need for an experimental RDMAP DDP Untagged Queue Number.

All other values are Unassigned and available to IANA for assignment. New RDMAP queue numbers should be assigned sequentially in order to better support implementations that perform RDMAP queue selection in hardware.

Allocation Policy: Standards Action ([RFC5226])

RFC Editor: Please replace XXXX in all instances of [RFCXXXX] above with the RFC number of this document and remove this note.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4303] S. Kent, "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
- [RFC5040] Recio, R. et al., "A Remote Direct Memory Access Protocol Specification", RFC 5040, October 2007.
- [RFC5041] Shah, H. et al., "Direct Data Placement over Reliable Transports", RFC 5041, October 2007.

- [RFC5042] Pinkerton, J. and E. Deleganes, "Direct Data Placement Protocol (DDP) / Remote Direct Memory Access Protocol (RDMAP) Security", October 2007.
- [RFC5226] T. Narten and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", May 2008.
- [RFC7146] D. Black and P. Koning, "Securing Block Storage Protocols over IP: RFC 3723 Requirements Update for IPsec v3", April 2014.

RFC Editor: Please remove reference to RFC5226 if the associated IANA Considerations reference is also removed before publication.

11.2. Informative References

- [IB] InfiniBand Trade Association, "InfiniBand Architecture Specification Volumes 1 and 2", Release 1.1, November 2002, available from <http://www.infinibandta.org/specs>.
- [R SOCKETS] RSocketS, RDMA enabled Sockets library for Open Fabrics, available from <http://git.openfabrics.org/?p=~shefty/librdmacm.git;a=summary>.
- [RFC5044] P. Culley, U. Elzur, R. Recio, S. Bailey, J. Carrier, "Marker PDU Aligned Framing for TCP Specification", October 2007.
- [RFC5045] C. Bestler and L. Coene, "Applicability of Remote Direct Memory Access Protocol (RDMA and Direct Data Placement Protocol (DDP)", October 2007.
- [RFC6581] A. Kanevsky, C. Bestler, R. Sharp, S. Wise, "Enhanced Remote Direct Memory Access (RDMA) Connection Establishment", April 2012.
- [OFAVERBS] Open Fabrics Alliance Verbs Enhanced Atomic Operations, "[PATCH 0/2] Add support for enhanced atomic operations", available from <http://www.spinics.net/lists/linux-rdma/msg02405.html>.
- [DAT_ATOMICS] DAT Collaborative, User Direct Access Programming Library, "Ratified DAT IB extension spec", available from http://www.datcollaborative.org/DAT_IB_Extensions.pdf.

[MPI] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard, Version 3.0", available from <http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf>, September 2012.

12. Acknowledgments

The authors would like to acknowledge the following contributors who provided valuable comments and suggestions.

- o David Black
- o Arkady Kanevsky
- o Bernard Metzler
- o Jim Pinkerton
- o Tom Talpey
- o Steve Wise
- o Don Wood

This document was prepared using 2-Word-v2.0.template.dot.

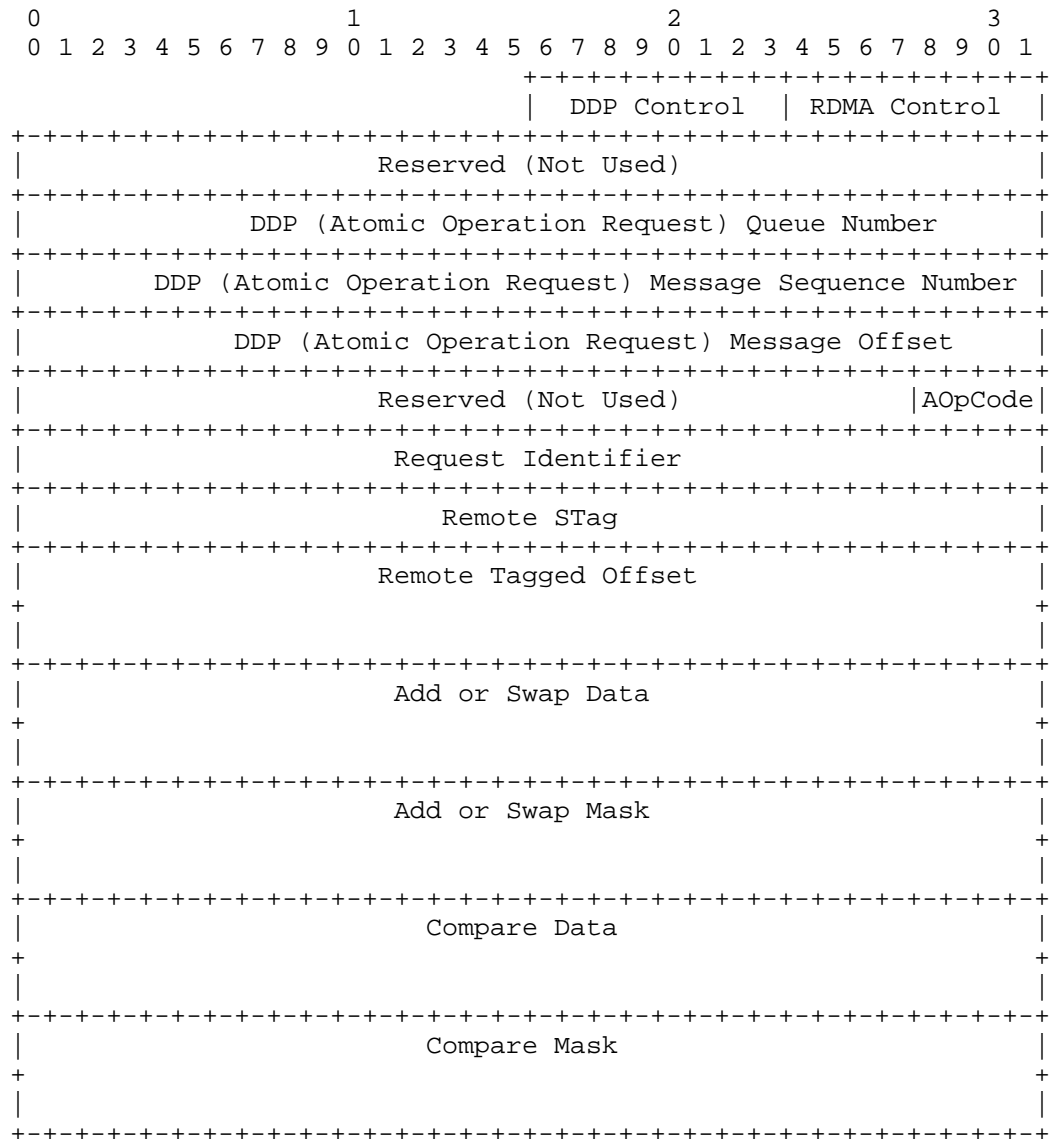
Appendix A.

DDP Segment Formats for RDMA Messages

This appendix is for information only and is NOT part of the standard. It simply depicts the DDP Segment format for the various RDMA Messages.

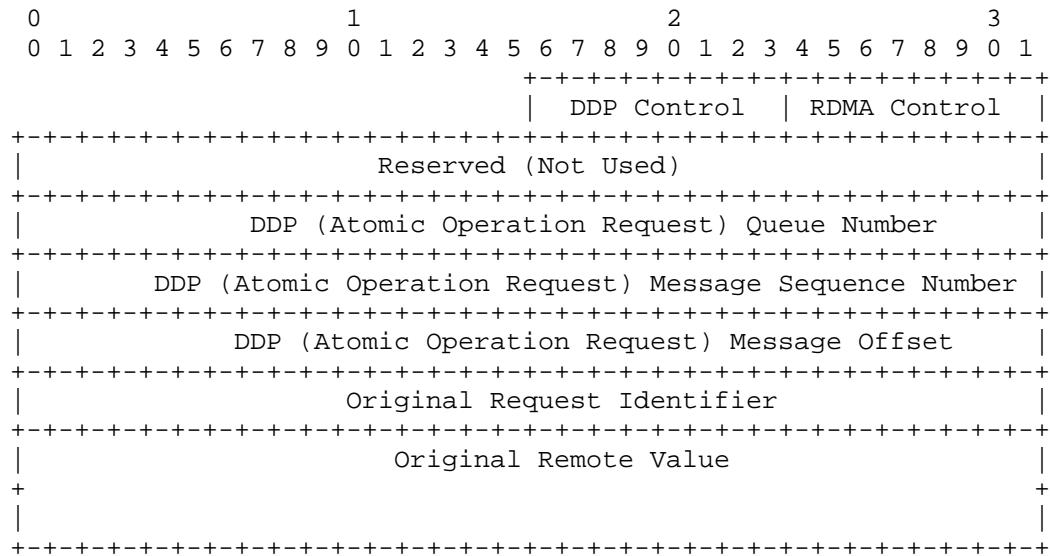
A.1. DDP Segment for Atomic Operation Request

The following figure depicts an Atomic Operation Request, DDP Segment:



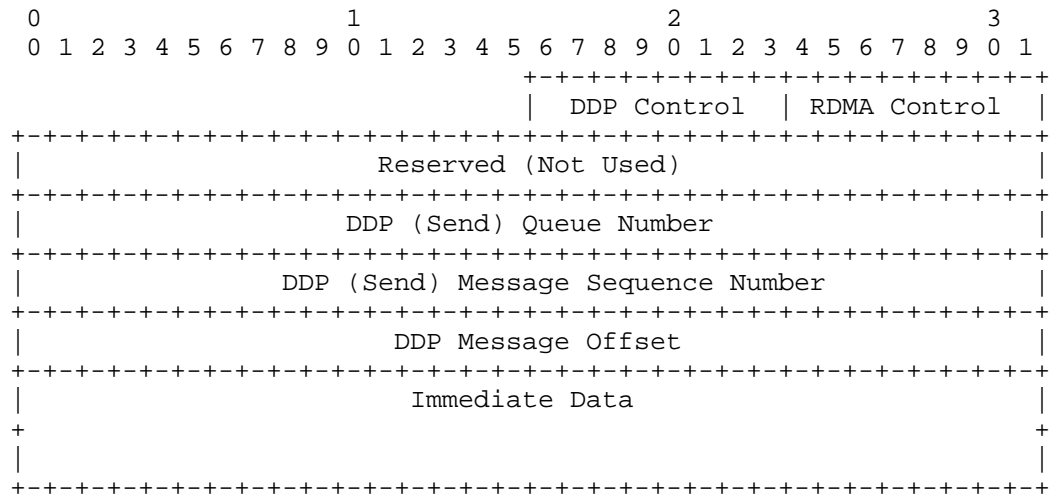
A.2. DDP Segment for Atomic Response

The following figure depicts an Atomic Operation Response, DDP Segment:



A.3. DDP Segment for Immediate Data and Immediate Data with SE

The following figure depicts an Immediate Data or Immediate data with SE, DDP Segment:



Authors' Addresses

Hemal Shah
Broadcom Corporation
5300 California Avenue
Irvine, CA 92617
Phone: 1-949-926-6941
Email: hemal@broadcom.com

Felix Marti
Chelsio Communications, Inc.
370 San Aleso Ave.
Sunnyvale, CA 94085
Phone: 1-408-962-3600
Email: felix@chelsio.com

Asgeir Eiriksson
Chelsio Communications, Inc.
370 San Aleso Ave.
Sunnyvale, CA 94085
Phone: 1-408-962-3600
Email: asgeir@chelsio.com

Wael Nouredine
Chelsio Communications, Inc.
370 San Aleso Ave.
Sunnyvale, CA 94085
Phone: 1-408-962-3600
Email: wael@chelsio.com

Robert Sharp
Intel Corporation
1300 South Mopac Expy, Mailstop: AN4-4B
Austin, TX 78746
Phone: 1-512-362-1407
Email: robert.o.sharp@intel.com

