TCPM working group
Internet Draft
Intended Status: Informational
Expires: 12/1/2013

M. Fox
C. Kassimis
J. Stevens
IBM

May 31, 2013

Shared Memory Communications over RDMA
draft-fox-tcpm-shared-memory-rdma-02.txt

Abstract

This document describes the 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

This document is a specification of the 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.

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.

1.1. Summary of changes in this draft

Significant changes in this architecture since the previous draft:

- o Modified the RDMA data transfer protocol to use message passing instead of RDMA writes to a dedicated control area to communicate control information between peers.

- o Added MTU and and Packet sequence number to certain messages

- o Added support for optional LLC messages, to allow for toleration of future, optional function.

1.2. Protocol overview

SMC-R defines the concept of the SMC-R Link, which is a logical point-to-point link 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.

An SMC-R link manages 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 stack 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.

Multiple SMC-R links between the same two TCP/IP stack peers are also supported. 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 redundant links are available

for load balancing as well as seamless failover. A set of SMC-R links that provides redundant connectivity is called a link group.

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, 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.

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.

1.3. 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 VLAN 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, access to the same RMB(s) and the same TCP server / client

SMC-R peer

The SMC-R Peer stack 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.

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 stack 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 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 to SMC-R stack, where typically each peer stack would reside on separate hosts.

```
                           ,,.--..,_
    +----+             _-``         `-,            +-----+
    |QP 8|            -   RoCE         ',          |QP 64|
    |    |          /     VLAN M         .         |     |
    +----+--------+/                     \+-------+-----+
     | RNIC 1     |    SMC-R Link         | RNIC 2     |
     |            |<--------------------->|            |
     +------------+ ,                    /+------------+
             MAC A (GID A)            MAC B (GID B)
                      .                .`
                       `',          ,-`
                          ``''--''``
```

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 is associated with a single and specific VLAN. VLAN exploitation is required for SMC-R as it is a key isolation attribute of this architecture. The RoCE fabric is the same physical fabric used for standard TCP/IP over Ethernet communications, with Converged Enhanced Ethernet (CEE_enabled) switches.

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 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).

```
         Host X                                     Host Y
    +-------------------+        ,.--.,_       +-------------------+
    |                   |     .'`       '.     |                   |
    | Protection        |   ,'            `,   |    Protection     |
    | Domain X          |  /                \  |    Domain Y       |
    |            +------+ /                  \ +------+            |
    |       QP 8 |RNIC 1| |   SMC-R Link     | |RNIC 2| QP 64      |
    |        |   |      |<-------------------->|      |   |        |
    |        |   |      ||                    ||      |   |        |
    |        |   +------+|    VLAN A          |+------+   |        |
    |        |          ||                    ||          |        |
    |        |          | |   RoCE           | |          |        |
    |        |RTokenX)  | \                  / |RToken (Y)|        |
    |        |          |  \                /  |          |        |
    |        V          |   `.            ,'   |          V        |
    | +--------+        |     '._       ,'     |        +--------+ |
    | |        |        |        `''-'``       |        |        | |
    | | RMB    |        |                      |        | RMB    | |
    | |        |        |                      |        |        | |
    | +--------+        |                      |        +--------+ |
    +-------------------+                      +-------------------+
```

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. 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.

```
               ---  +-----------+----------------+
               /\   |Eyecatcher |                |
                |   +-----------+                |
                |   |                            |
      RMB Element 1 |                            |
                |   |   Receive Buffer           |
                |   |                            |
                |   |                            |
               \/   |                            |
               ---  +-----------+----------------+
               /\   |Eyecather  |                |
                |   +-----------+                |
                |   |                            |
      RMB Element 2 |                            |
                |   |   Receive Buffer           |
                |   |                            |
                |   |                            |
               \/   |                            |
               ---  +----------------------------+
                    |           .                |
                    |           .                |
                    |           .                |
                    |           .                |
                    |    (up to 255 elements)    |
                    +----------------------------+
```

Figure 3   RMB Format

Figure 3 illustrates the basic format of an RMB. The RMB is a contiguous block of pinned memory that can support up to 255 TCP connections to exactly one remote SMC-R peer. Each RMB is therefore associated with the SMC-R links 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 allocated with multiple entries for efficiency; multiple TCP connections across an SMC link 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 entries in an RMB and the size of each RMB Element is entirely governed by the owning peer subject to the SMC-R architecture rules. 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. Each RMB supports multiple RMB Elements, one per TCP connection; however, all RMB elements within a given RMB must have the same size. 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 control area 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 and each peer in an SMC-R link manages allocation of its RMBs. RMB allocation can be asymmetric. For example, server X can allocate 2 RMBs to an SMC-R link while server Y allocates 5. This provides maximum implementation flexibility to allow hosts optimize RMB management for their own local requirements.

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 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 peer. This cursor is provided by the peer using RDMA message passing, and tells the local stack how far it can consume data in the RMBE write buffer.

- o The peer consumer cursor: This is a wrapping offset into the peer's RMB element's receive buffer that points to the next byte of data to be consumed by the peer in its own RMBE. This stack cannot write into the peer's RMBE beyond this point without causing data loss. This cursor is also provided by the peer using RDMA message passing.

Each TCP connection peer maintains its cursors for a TCP connection's RMBE in its local control structures. In other words, the stack who writes into a peer's RMBE provides its producer cursor to the peer whose RMB it has written into. The stack 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 RDMA message passing to update its producer cursor to peer A, to indicate to peer A how much data is available for peer A to consume. The RDMA 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 an RDMA 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

Delet   they are optimized to minimize the number of RDMA control messages
   required, as described in 4   ..6. RMB data
flows.

```
      Peer A's RMBE Control Info           Peer B's RMBE Control Info
     +--------------------------+          +--------------------------+
     |                          |          |                          |
      /----Peer producer cursor |    +-----+-Peer consumer cursor     |
    /|                          |    |     |                          |
   | +--------------------------+    |     +--------------------------+
   |  Peer A's RMBE                  |
   | +--------------------------+    |
   | |            +------------------+
   | |            |             |
   | |            \/            |
   | |             +------------|
   | |-------------+/////////// |
   | |//RMA data written by /// |
   | |/// peer B that is ////// |
   | |/available to be consumed/|
   | |///////////////////////// |
   | |///////// +---------------|
   | |----------+/\             |
   | |            |             |
    \|            |             |
     \           /              |
     |\---------/               |
     |                          |
     |                          |
```

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 RDMA message passing with the control information contained in inline

Delet   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 be 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.

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 types

Equal links within a link group also have another "Link 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 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.

```
         Host X                                     Host Y
    +-------------------+                      +-------------------+
    |                   |                      |                   |
    | Protection        |                      |    Protection     |
    | Domain X          |                      |    Domain Y       |
    |            +------+                      +------+            |
    |       QP 8 |RNIC 1|     SMC-R Link 1     |RNIC 2| QP 64      |
    |RToken X|   |      |<-------------------->|      |  |         |
    |        |   |      |                      |      |  |RToken Y|
    |       \/   +------+                      +------+ \/         |
    |+--------+         |                      |        +--------+ |
    ||        |         |                      |        |        | |
    || RMB    |         |                      |        | RMB    | |
    ||        |         |                      |        |        | |
    |+--------+         |                      |        +--------+ |
    |       /\   +------+                      +------+ /\         |
    |RToken Z|   |      |     SMC-R Link 2     |      |  |RToken W|
    |        |   |RNIC 3|<-------------------->|RNIC 4|  |         |
    |       QP 9 |      |                      |      | QP 65      |
    |            +------+                      +------+            |
    +-------------------+                      +-------------------+
```

Figure 5 Symmetric SMC-R links

```
         Host X                                     Host Y
    +-------------------+                      +-------------------+
    |                   |                      |                   |
    | Protection        |                      |    Protection     |
    | Domain X          |                      |    Domain Y       |
    |            +------+                      +------+            |
    |       QP 8 |RNIC 1|     SMC-R Link 1     |RNIC 2| QP 64      |
    |RToken X|   |      |<-------------------->|      |   |        |
    |        |   |      |                   .->|      |   |RToken Y|
    |       \/   +------+                 .`   +------+  \/        |
    |+--------+         |               .`     |        +--------+ |
    ||        |         |             .`       |        |        | |
    || RMB    |         |           .`         |        | RMB    | |
    ||        |         |         .`SMC-R      |        |        | |
    |+--------+         |       .` Link 2      |        +--------+ |
    |       /\   +------+     .`               +------+            |
    |Rtoken Z|   |      |   .`                 |      |down or     |
    |        |   |RNIC 3|<-`                   |RNIC 4|unavailable |
    |       QP 9 |      |                      |      |            |
    |            +------+                      +------+            |
    +-------------------+                      +-------------------+
```

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.

```
         Host X                                    Host Y
    +-------------------+                     +-------------------+
    |                   |                     |                   |
    | Protection        |                     |    Protection     |
    | Domain X          |                     |    Domain Y       |
    |            +------+  SMC-R link 1       +------+            |
    |       QP 8 |RNIC 1|<------------------->|RNIC 2| QP 64      |
    |RToken X|   |      |                     |      |   |        |
    |        |   |      |<------------------->|      |   |Rtoken Y|
    |       \/   +------+  SMC-R link 2       +------+  \/        |
    |+--------+   QP 9  |                     | QP 65  +--------+ |
    ||        |    |    |                     |  |     |        | |
    || RMB    |<-- +    |                     |  +---->| RMB    | |
    ||        |         |                     |        |        | |
    |+--------+         |                     |        +--------+ |
    |            +------+                     +------+            |
    |     down or|      |                     |      |down or     |
    |  unavailale|RNIC 3|                     |RNIC 4|unavailable |
    |            |      |                     |      |            |
    |            +------+                     +------+            |
    +-------------------+                     +-------------------+
```

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 stacks 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 following 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.

- 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 stack 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.

- Peer ID: The SMC-R peer ID uniquely identifies a specific instance of a specific stack. It is required because in clustered and load balancing environments, an IP address does not uniquely identify a stack. An RNIC's MAC/GID also doesn't uniquely or reliably identify a stack because RNICs can go up and down and even be redeployed to other stacks in a multiple partitioned or virtualized environment. The peer ID is not only unique per stack but is also unique per instance of a stack, meaning that if a 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 VLAN.

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 be 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. The server must perform the actual link deletion.

The movement of TCP connections to another link can be accomplished without notifying or coordinating with the peer. The TCP connection

movement is also transparent to and non disruptive to the TCP socket application workloads. After a failure, the surviving links and all associated hardware must handle the link group's workload.

As each SMC-R stack begins to move active TCP connections to another link all current RDMA write operations must be allowed to complete and then may be retried over the new link if the previously completed RDMA write operation did not successfully complete.

When a new link becomes available and is re-added to the link group then each stack 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 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.

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 253 on the three-way handshake flows, in accordance with draft-ietf-tcpm-experimental-options-01.txt. The magic number value used is the string 'SMCR' in EBCDIC (IBM-1047) encoding (0xE2D4C3D9).

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 newly opened 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 ink elements such as RoCE MAC, GID, RMB information etc.

- 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 ink elements such as RoCE MAC, GID, RMB information etc.

- 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:

- 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.

- 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

- 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.

- 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.

- CONFIRM RKEY: Informs the peer on the SMC-R link of the addition or deletion of one or more RMBs in the link group

- TEST LINK: Verifies that an already-active SMC-R link is active and healthy

- 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:

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

- DELETE LINK request from a server in response to a DELETE LINK from a client

3.4. Rendezvous flows

Rendezvous information for SMC-R is be exchanged as TCP options on the TCP 3-way handshake flows to indicate capability, followed by in-line 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.

3.4.1. 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.

3.4.1.1. TCP Options pre-negotiation

The client and server indicate their SMC-R capability to each other using TCP option 253 on the TCP 3-way handshake flows.

A client who wishes to do SMC-R will include TCP option 253 using a magic number equal to the EBCDIC (codepage IBM-1047) encoding of "SMCR" on its SYN flow.

A server that supports SMC-R will include TCP option 253 with the magic number 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 unconditionally includes this TCP option if it supports SMC-R.   This may be required for servers such as Linux where proprietary extensions to the TCP stack are not practical. For proprietary servers 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 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. Proprietary stacks may do this by examining what was actually provided on the SYN and SYN-ACK packets, and open stacks may do this 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, similar to the SSL negotiation model. The socket is not yet turned over to the applications; instead the respective SMC layers exchange CLC messages over the newly formed TCP connection.

3.4.1.2. 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 and the server must validate that it is on the same VLAN 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.   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.

3.4.1.3. 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 VLAN 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

Delet   link for the TCP connection, see
3   ..4.2. 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.

3.4.1.4. 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.

3.4.1.5. 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.

```
      Host X -- Server                           Host Y -- Client
   +-------------------+                      +-------------------+
   | PeerID = PS1      |                      |    PeerID = PC1   |
   |            +------+                      +------+            |
   |       QP 8 |RNIC 1|                      |RNIC 2| QP 64      |
   |RToken X|   |MAC MA|                      |MAC MB|   |        |
   |        |   |GID GA|                      |GID GB|   |Rtoken Y|
   |       \/   +------+      (Subnet S1)     +------+  \/        |
   |+--------+         |                      |        +--------+ |
   || RMB    |         |                      |        | RMB    | |
   |+--------+         |                      |        +--------+ |
   |            +------+                      +------+            |
   |            |RNIC 3|                      |RNIC 4|            |
   |            |MAC MC|                      |MAC MD|            |
   |            |GID GC|                      |GID GD|            |
   |            +------+                      +------+            |
   +-------------------+                      +-------------------+

                     SYN TCP options(253,"SMCR")
        <---------------------------------------------------------

                     SYN-ACK TCP options(253, "SMCR")
        --------------------------------------------------------->

                     ACK [TCP options(254, "SMCR")]
        <--------------------------------------------------------

                    SMC Proposal(PC1,MB,GB,S1)
        <--------------------------------------------------------

      SMC Accept(PS1,first contact,MA,GA,QP8,RToken=X,RMB element index)
        --------------------------------------------------------->

          SMC Confirm(PC1,MB,GB,QP64,RToken=Y, RMB element index)
        <--------------------------------------------------------

     Confirm Link (MA,GA,QP8, link lim, server's link userid, linknum)
        .........................................................>

     Confirm Link Rsp(MB,GB,QP64, link lim, client link userid, linknum)
        <........................................................

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

                  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

3.4.1.6. 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 VLAN, it attempts to set up the second SMC-R link over that second RNIC. If it only has one RNIC available on the VLAN, 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 VLAN.

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.

3.4.1.6.1. 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 VLAN. 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 VLAN, 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 VLAN, 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 VLAN, 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.

3.4.1.6.2. 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

Delet   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.

```
       Host X -- Server                           Host Y -- Client
    +-------------------+                      +-------------------+
    | PeerID = PS1      |                      |    PeerID = PC1   |
    |            +------+                      +------+            |
    |       QP 8 |RNIC 1|                      |RNIC 2| QP 64      |
    |RToken X|   |MAC MA|                      |MAC MB|   |        |
    |        |   |GID GA|                      |GID GB|   |RToken Y|
    |       \/   +------+                      +------+  \/        |
    |+--------+         |                      |        +--------+ |
    ||        |         |                      |        |        | |
    || RMB    |         |                      |        | RMB    | |
    ||        |         |                      |        |        | |
    |+--------+         |                      |        +--------+ |
    |       /\   +------+                      +------+  /\        |
    |        |   |RNIC 3|                      |RNIC 4|  |         |
    |RToken Z|   |MAC MC|                      |MAC MD|  |RToken W|
    |       QP 9 |GID GC|                      |GID GD| QP 65      |
    |            +------+                      +------+            |
    +-------------------+                      +-------------------+

                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

3.4.1.6.3. 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

Delet   3   ..4.5.2.3. Adding a new SMC-R link to a link group with multiple RMBs
for more details on these flows

3.4.1.6.4. 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

Delet   can flow during link group setup, see
C   ..1. SMC Decline during CLC
negotiation and C   ..2. SMC Decline during LLC
negotiation   Delet
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.

3.4.2. 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 253 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.

3.4.2.1. 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

Delet   case, see 3   ..4.3. 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 Confirm CLC message, providing the full RoCE information for the chosen SMC-R link to the client, using the same format as the SMC Confirm 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

Delet   flows described in 3   ..4.5.2.1. 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.

3.4.2.2. 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

Delet   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

Delet   performs the processing described in
3   ..4.5.2.1. 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.

3.4.2.3. 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

Delet   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.

3.4.2.4. 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 TCP 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.

3.4.3. 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

Delet   SMC-R link group (detailed in
3   ..4.2. 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.

3.4.4. 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.

3.4.5. Link group management flows

3.4.5.1. 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

3.4.5.1.1. 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 VLAN that the link group is using.

o Receipt of notification from the client that additional resources

Delet   on the client are available to support an additional SMC-R link.
See 3   ..4.5.1.2. Client initiated Add Link processing.

Server initiated Add Link processing in an established SMC-R link

Delet   group is the same as the Add Link processing described in
3   ..4.1.6.
Second SMC-R link setup with the following changes:

- 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.

- 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.

3.4.5.1.2. 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.

3.4.5.1.3. Server initiated Delete Link Processing

Reasons that a server may delete a link include:

- 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

- An error in resources supporting the link. These may include but are not limited to: RNIC errors, QP errors, software errors

- 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

Delet   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.

```
        Host X                                     Host Y
   +--------------------+                      +--------------------+
   |            +-------+                      +-------+            |
   |       QP 8 |RNIC 1|     SMC-R Link 1     |RNIC 2| QP 9         |
   |RToken X|   |Failed|<--X----X----X----X-->|       |             |
   |        |   |       |                      |       |             |
   |       \/   +-------+                      +-------+            |
   |+--------+          |                      |                    |
   || deleted|          |                      |                    |
   || RMB    |          |                      |                    |
   ||        |          |                      |                    |
   |+--------+          |                      |                    |
   |       /\   +-------+                      +-------+            |
   |RToken Z|   |       |     SMC-R Link 2     |       |             |
   |        |   |RNIC 3|<--------------------->|RNIC 4|             |
   |       QP 64|       |                      |       | QP 65       |
   |            +-------+                      +-------+            |
   +--------------------+                      +--------------------+

         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

3.4.5.1.4. 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:

- 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.

- 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.

```
        Host X                                         Host Y
   +--------------------+                    +--------------------+
   |            +------+                      +------+            |
   |       QP 8 |RNIC 1|     SMC-R Link 1     |RNIC 2| QP 9       |
   |RToken X|   |      |<---X--X--X--X--X--X->|Failed|            |
   |        |   |      |                      |      |            |
   |       \/   +------+                      +------+            |
   |+--------+         |                      |                   |
   || deleted|         |                      |                   |
   || RMB    |         |                      |                   |
   ||        |         |                      |                   |
   |+--------+         |                      |                   |
   |       /\   +------+                      +------+            |
   |RToken Z|   |      |     SMC-R Link 2     |      |            |
   |        |   |RNIC 3|<-------------------->|RNIC 4|            |
   |       QP 64|      |                      |      | QP 65      |
   |            +------+                      +------+            |
   +--------------------+                    +--------------------+

         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

3.4.5.2. 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.

3.4.5.2.1. 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.

```
         Host X                                     Host Y
    +-------------------+                      +-------------------+
    |            +------+                      +------+            |
    |       QP 8 |RNIC 1|     SMC-R Link 1     |RNIC 2| QP 9       |
    |RToken X|   |      |<-------------------->|      |            |
    |        |   |      |                      |      |            |
    |       \/   +------+                      +------+            |
    |+--------+         |                      |                   |
    || new    |         |                      |                   |
    || RMB    |         |                      |                   |
    ||        |         |                      |                   |
    |+--------+         |                      |                   |
    |       /\   +------+                      +------+            |
    |RToken Z|   |      |     SMC-R Link 2     |      |            |
    |        |   |RNIC 3|<-------------------->|RNIC 4|            |
    |       QP 64|      |                      |      | QP 65      |
    |            +------+                      +------+            |
    +-------------------+                      +-------------------+

          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.

```
       Host X -- Server                           Host Y -- Client
    +-------------------+                      +-------------------+
    | PeerID = PS1      |                      |    PeerID = PC1   |
    |            +------+                      +------+            |
    |       QP 8 |RNIC 1|    SMC-R link 1      |RNIC 2| QP 64      |
    |RToken X|   |MAC MA|<-------------------->|MAC MB|   |        |
    |        |   |GID GA|                      |GID GB|   |RTokenY2|
    |       \/   +------+                      +------+  \/        |
    |+--------+         |                      |        +--------+ |
    ||        |         |   SUBNET S1          |        | New    | |
    || RMB    |         |                      |        | RMB    | |
    |+--------+         |                      |        +--------+ |
    |       /\   +------+                      +------+  /\        |
    |        |   |RNIC 3|    SMC-R link 2      |RNIC 4|   |RTokenW2|
    |        |   |MAC MC|<-------------------->|MAC MD|   |        |
    |       QP 9 |GID GC|                      |GID GD| QP65       |
    |            +------+                      +------+            |
    +-------------------+                      +-------------------+

            SYN / SYN-ACK / ACT TCP 3-way handshake with TCP option
         <--------------------------------------------------------->

                     SMC Proposal(PC1,MB,GB,S1)
         <--------------------------------------------------------

       SMC Accept(PS1,not 1st contact,MA,GA,QP8,RToken=X,RMB elem index)
         --------------------------------------------------------->

           Confirm Rkey(Request, Add,
         <........................................................
                      RToken set((Link1, RToken Y2),{Link2, RToken W2)))

           Confirm Rkey(Response, Add,
          ........................................................>
                      RToken set((Link1, RToken Y2),{Link2, RToken W2)))

           SMC Confirm(PC1,MB,GB,QP64,RToken=Y2, RMB element index)
         <--------------------------------------------------------

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

Figure 13 Client adds RMB during TCP connection setup

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

Either peer can delete one of its RMBs as long as it is not being used for any TCP connections. Ideally an SMC-R host 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 CONFIRM RKEY(Delete) 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 CONFIRM RKEY(delete) exchange.

```
        Host X                                 Host Y
   +-------------------+                      +-------------------+
   |            +------+                      +------+            |
   |       QP 8 |RNIC 1|     SMC-R Link 1     |RNIC 2| QP 9       |
   |RToken X|   |      |<-------------------->|      |            |
   |        |   |      |                      |      |            |
   |       \/   +------+                      +------+            |
   |+--------+         |                      |                   |
   || deleted|         |                      |                   |
   || RMB    |         |                      |                   |
   ||        |         |                      |                   |
   |+--------+         |                      |                   |
   |       /\   +------+                      +------+            |
   |RToken Z|   |      |     SMC-R Link 2     |      |            |
   |        |   |RNIC 3|<-------------------->|RNIC 4|            |
   |       QP 9 |      |                      |      |            |
   |            +------+                      +------+            |
   +-------------------+                      +-------------------+

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

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

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

Figure 14 Delete RMB from SMC-R link group

3.4.5.2.3. 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.

```
      Host X -- Server                           Host Y -- Client
   +-------------------+                       +-------------------+
   | PeerID = PS1      |                       |    PeerID = PC1   |
   |            +------+                       +------+            |
   |       QP 8 |RNIC 1|                       |RNIC 2| QP 64      |
   |Rkey Set|   |MAC MA|                       |MAC MB|   |Rkey set|
   |X,Y,Z   |   |GID GA|                       |GID GB|   |Q,R,S,T |
   |       \/   +------+                       +------+  \/        |
   |+--------+         |                       |        +--------+ |
   || 3 RMBs |         |                       |        | 4 RMBs | |
   |+--------+         |                       |        +--------+ |
   |       /\   +------+                       +------+  /\        |
   |Rkey set|   |RNIC 3|                       |RNIC 4|  | Rkey set|
   |U,V,W   |   |MAC MC|                       |MAC MD|  | L,M,N,P |
   |       QP 9 |GID GC|                       |GID GD| QP 65      |
   |            +------+                       +------+            |
   +-------------------+                       +-------------------+

          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

3.4.5.3. 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 links 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.

- CONFIRM RKEY / CONFIRM 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 is 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 requests. The concurrent CONFIRM RKEY requests can be independently processed and does 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.

3.4.5.3.1. 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.

3.4.5.3.2. 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

3.4.5.3.3. 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 peer SMC-R host 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 contiguous pinned memory that 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 diagnostic eyecatcher should be used by the local SMC-R stack to check for overlay errors by verifying an intact eye catcher 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, control flags such are urgent data and writer blocked indicators, and TCP connection information such as termination flags. This information is exchanged between SMC-R peers using RDMA message passing with inline data, with the control information contained in

the inline data. An SMC-R stack 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 this inline data is described in detail in

Delet   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, and abnormal close

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.

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.

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 ..7.

Delet   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 RDMA 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 update 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 an RDMA 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 The optimized window size updates are overridden when the sender sets the Consumer Cursor Update Requested flag in the an RDMA control 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:

- 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.

- 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:

- SMC-R hosts 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 that 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.

- SMC-R hosts 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. 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 partner SMC-R hosts). 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.

4.6.1. Scenario 1: Send flow, window size 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
0        0         0                     0    0        0          0
0        0         1 ---------------> 1    0        0          0
                     RDMA-WR Data
                       (0:999)
0        0         2 ...............> 2    1000     0          0
                     RDMA MSG Control
                        data
```

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 0-999. Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation.

2. Host A sends an RDMA message with inline data to update the Producer Cursor to byte 1000. This RDMA 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 if 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 (10,000-1,000=9,000).

4.6.2. 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
   0         0         0                     0    0        0          0
   0         0         1 ----------------> 1    0        0          0
                         RDMA-WR Data
                           (0:999)
   0         0         2 ................> 2    1000     0          0
                         RDMA MSG Control
                            data
   0         0         3 <---------------  3    1000     0          0
                         RDMA-WR Data
                           (0:499)
   1000      0         4 <...............  4    1000     0          0
                          RDMA MSG Control
                          data
```

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 0-999. Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation.

2. Host A issues an RDMA message with inline data to update the Producer Cursor to byte 1000. This RDMA 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 RDMA 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 an RDMA message to update the partner's RMBE Control information with the latest Producer Cursor (set to 500 and not shown in the diagram above) and to also inform the peer that the Consumer Cursor value is now 1000. It also updates the local Current Consumer Cursor and Last Sent Consumer Cursor to 1000. This RDMA message includes notification since we are updating our Producer Cursor which requires attention by the peer host.

4.6.3. 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
   0        0         0                     0    0        0          0
   0        0         1 ---------------> 1    0        0          0
                        RDMA-WR Data
                          (0:2999)
   0        0         2 ...............> 2    3000     0          0
                        RDMA MSG Control
                           data
   0        0         3                     3    3000     0          0
   0        0         4 ---------------> 4    3000     0          0
                        RDMA-WR Data
                          (3000:6999)
   0        0         5 ................> 5    7000     0          0
                        RDMA MSG Control
                           data
   7000     0         6 <................ 6    7000     0          0
                        RDMA MSG Control
                           data
```

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 0-2,999. Note that no immediate data or alert (i.e. interrupt) is provided to host B for this RDMA operation.

2. Host A sends an RDMA message with inline data to update its Producer Cursor to byte 3000. This RDMA 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 RDMA 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 3000-6999. Note that no alert is provided to host B for this flow.

5. Host A sends an RDMA message with inline data to update the Producer Cursor to byte 7000. This RDMA 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 RDMA 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 < 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 an RDMA message is issued to update the Consumer Cursor to peer A.

4.6.4. 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
   1000      1         0                    0     1000      1          0
   1000      1         1 ---------------> 1     1000      1          0
                         RDMA-WR Data
                           (1000:9999)
   1000      1         2 ---------------> 2     1000      1          0
                         RDMA-WR Data
                           (0:999)
   1000      1         3 ...............> 3     1000      2          Wrt
                         RDMA MSG Control                            Blk
                            data
   1000      2         4 <............... 4     1000      2          Wrt
                         RDMA MSG Control                            Blk
                           data
   1000      2         5 ---------------> 5     1000      2          Wrt
                         RDMA-WR Data                                Blk
                           (1000:9999)
   1000      2         6 ---------------> 6     1000      2          Wrt
                         RDMA-WR Data                                Blk
                          (0:999)
   1000      2         7 ...............> 7     1000      3          Wrt
                         RDMA MSG Control                            Blk
                           data
   1000      3         8 <............... 8     1000      3          Wrt
                         RDMA MSG Control                            Blk
                           data
```

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 1000-9999. 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 data into the peer's RMBE receive buffer, at relative position 0-999. 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 RDMA write message with inline data to update the Producer Cursor to byte 1000. 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 RDMA 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 RDMA 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 (1000-9999) followed by the second segment (0-999). Because the Producer Writer Blocked indicator was set, Host B schedules an RDMA message to update its latest information to the peer: Consumer Cursor (1000), Consumer Window Wrap Size Sequence Number (2: the current Producer Window Wrap Sequence Number is used).

5. Host A, upon receipt of the message locates the RMBE 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 1000-9999.

6. Host A then moves the remaining 1000 bytes into the beginning of the receive buffer (0-999) by scheduling an RDMA write operation.

7. Host A then sends an RDMA message with inline data 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 an RDMA message to update the peer to indicate that all data has been consumed.

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

```
            SMC Host A                              SMC HostB
           RMBE A Info                             RMBE B Info
       (Consumer Cursors)                      (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
                        RDMA MSG Control                            UrgA
                          data
   1500     1         3 <............... 3    1500     1         UrgP
                        RDMA MSG Control                            UrgA
                          data
   1500     1         4 ---------------> 4    1500     1         UrgP
                        RDMA-WR Data                                UrgA
                          (1500:2499)
   1500     1         5 ...............> 5    2500     1         0
                        RDMA MSG Control
                         data
```

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. sends an RDMA message with inline data 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 RDMA 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 RDMA 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 an RDMA message to update the partner's RMBE Control area with its latest Consumer Cursor (1500). Note this RDMA 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 an RDMA 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.

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

```
            SMC Host A                              SMC HostB
           RMBE A Info                            RMBE B Info
       (Consumer Cursors)                      (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
                         RDMA MSG control                             Blk
                           data                                       UrgP

   1000      2         2 <............... 2    1000     2          Wrt
                         RDMA MSG Control                             Blk
                           data                                       UrgP

   1000      2         3 ---------------> 3    1000     2          Wrt
                         RDMA-WR data  l                              Blk
                           (1000:1499)                                UrgP

   1000      2         4 ...............> 4    1500     2          UrgP
                         RDMA MSG control                             UrgA
                           data
   1500      2         5 <............... 5    1500     2          UrgP
                         RDMA MSG Control                             UrgA
                          data
   1500      2         6 ---------------> 6    1500     2          UrgP
                         RDMA-WR data  l                              UrgA
                           (1500:2499)
   1000      2         7 ...............> 7    2500     2          0
                         RDMA MSG control
                           data
```

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 an RDMA message with inline data 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 RDMA 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 an RDMA 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 the sends an RDMA message with inline data 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 an RDMA 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 an RDMA message operation 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.

4.7. 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.

4.7.1. 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.

```
                                      1
                            +-----------------+
            |-------------->|     CLOSED      |<-------------|
        3D  |               |                 |              |  4D
            |               +-----------------+              |
            |                       |                        |
            |                     2 |                        |
            |                       V                        |
    +----------------+     +-----------------+     +----------------+
    |AppFinCloseWait |     |     ACTIVE      |     |PeerFinCloseWait|
    |                |     |                 |     |                |
    +----------------+     +-----------------+     +----------------+
            |                   |         |                   |
            |     Active Close  | 3A | 4A |  Passive Close    |
            |                   V    |    V                   |
            |       +--------------+ | +-------------+        |
            |--<----|PeerCloseWait1| | |AppCloseWait1|--->----|
        3C  |       |              | | |             |        |  4C
            |       +--------------+ | +-------------+        |
            |             |          |         |              |
            |             | 3B       |     4B  |              |
            |             V          |         V              |
            |       +--------------+ | +-------------+        |
            |--<----|PeerCloseWait2| | |AppCloseWait2|--->----|
                    |              | | |             |
                    +--------------+ | +-------------+
                                     |
                                     |
```

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 an RDMA message with Inline Data 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 an RDMA message with immediate data 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 RDMA 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 an RDMA 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 an RDMA 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 an RDMA message with inline data 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 peer has also sent the local stack 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 needs to provide some logic that will prevent being stranded in termination state indefinitely. For example, if an Active Close SMC-R host 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 host performs TCP reset processing for this connection. However, no additional RDMA messages 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).

4.7.1.1. 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 host 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.

```
                     +-----------------+
     |-------------->|     CLOSED      |<-------------|
     |               |                 |              |
     |               +-----------------+              |
     |                                                |
     |                                                |
     |                                                |
     |             +-----------------+                |
     |             |     Any State   |                |
     |1B           | (before setting |              2B|
     |             |  PeerConnClosed |                |
     |             |  Indicator in   |                |
     |             |  Peer's RMBE)   |                |
     |             +-----------------+                |
     |        1A         |         |      2A          |
     |     Active Abort  |         |  Passive Abort   |
     |                   V         V                  |
     |       +--------------+   +--------------+      |
     |-------|PeerAbortWait |   | Process Abort|------|
             |              |   |              |
             +--------------+   +--------------+
```

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 sendsthe 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 host (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 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 an RDMA 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 an RDMA message with inline data to notify the peer of the PeerConnAbort indicator and once that is completed transition this RMBE to the Closed state.

If an SMC-R host 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).

4.7.1.2. Other SMC-R connection termination conditions

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

o A SMC-R host being gracefully shut down. If an SMC-R host 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.

- Abnormal termination of an SMC-R host. 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.

- 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 stacks 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. 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 RMDA 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 draft-ietf-tcpm-experimental-options-01.txt.

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

[Tou2012] Touch, J., "Shared use of Experimental TCP Options", draft URL, http://tools.ietf.org/html/draft-ietf-tcpm-experimental-options-01

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 RDMA messages with inline data.

Appendix A.   Formats

A.1. TCP option

The SMC-R TCP option is formatted in accordance with draft-ietf-tcpm-experimental-options-01.txt. The magic number is IBM-1047 (EBCDIC) encoding for 'SMCR'

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Kind = 253  | Length = 6    |   x'E2'       |   x'D4'       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |    x'C3'      |    x'D9'      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 stack. This peer ID is required to be universally unique across stacks and instances (including restarts) of stacks.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Instance ID          |  RoCE MAC (first two bytes)   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                RoCE MAC (last four bytes)                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 stack, for example if an RNIC is redeployed to another stack, the values are unique. It also ensures that if a 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 stack's RNICs. Note that in a virtualized environment this will be the virtual MAC of one of the stack's RNICs.

A.2.2. SMC Proposal CLC 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   x'E2'       |   x'D4'       |     x'C3'     |     x'D9'     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Type = 1     |           Length              |Version| Rsrvd |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                       Client's Peer ID                      -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-                Client's preferred GID                       -+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Client's preferred RoCE                                      |
   +- MAC address                     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |         Reserved              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         IPv4 Subnet Mask                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | IPv4 Mask Lgth|           Reserved            |Num IPv6 prfx  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   :                                                               :
   :        (Variable length) array of IPv6 Prefixes               :
   :                                                               :
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   x'E2'       |   x'D4'       |     x'C3'     |     x'D9'     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Delet
      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.

   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

   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.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                  IPv6 Prefix value                            +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Prefix Length |
+-+-+-+-+-+-+-+-+
```

Figure 27 Format for IPv6 Prefix array element

A.2.3. SMC Accept CLC 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   x'E2'       |   x'D4'       |     x'C3'     |     x'D9'     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Type = 2     |    Length = 68                |Version|F|Rsvd |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                       Server's Peer ID                      -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-                Server's RoCE GID                            -+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Server's RoCE                                                |
   +- MAC address                  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |     Server QP (bytes 1-2)     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+---+
   |Srvr QP byte 3 |         Server RMB Rkey (bytes 1-3)           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Srvr RMB byte 4|Server RMB indx| Srvr RMB alert tkn (bytes 1-2)|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Srvr RMB alert tkn (bytes 3-4)|Bsize  | MTU   |   Reserved    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                     Server's RMB virtual address            -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Reserved      |    Server's initial packet sequence number    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   x'E2'       |   x'D4'       |     x'C3'     |     x'D9'     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 64 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

Delet

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.

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 stack will use when sending to the peer, so that the peer can prepare its QP for the sequence number to expect.

A.2.4. SMC Confirm CLC 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   x'E2'       |   x'D4'       |     x'C3'     |     x'D9'     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Type = 3     |    Length = 60                |Version| Rsrvd |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                      Client's Peer ID                       -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                                                             -+
|                                                               |
+-               Client's RoCE GID                             -+
|                                                               |
+-                                                             -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Client's RoCE                                                |
+- MAC address                  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |     Client QP (bytes 1-2)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-----+
|Clnt QP byte 3 |         Client RMB Rkey (bytes 1-3)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Clnt RMB byte 4|Client RMB indx| Clnt RMB alert tkn (bytes 1-2)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Clnt RMB alert tkn (bytes 3-4)|Bsize  | MTU   |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-                 Client's RMB Virtual Address                -+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reserved      |    Client's initial packet sequence number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   x'E2'       |   x'D4'       |     x'C3'     |     x'D9'     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 60 bytes long

Version

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

Client's Peer ID

Delet

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

Delet   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.

   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 stack will use when
      sending to the peer, so that the peer can prepare its QP for the
      sequence number to expect

      .

A.2.5. SMC Decline CLC 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   x'E2'       |   x'D4'       |     x'C3'     |     x'D9'     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Type = 4     |    Length = 28                |Version| Rsrvd |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                       Sender's Peer ID                      -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Reason code                |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+      Reserved                -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   x'E2'       |   x'D4'       |     x'C3'     |     x'D9'     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

Sender's Peer ID

Delet

As described in A   ..2.1. above

Reason Code

A two byte reason code set by the sender.

Values tbd

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.

Delet   High order bits of b'11' indicate an RMBE control message as
described in A   ..4.

A.3.1. CONFIRM LINK 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 = 1     |  length = 44  |Version| Rsrvd |R|  Reserved   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Sender's RoCE                                                |
   +-   MAC address                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   +-                                                             -+
   |                 Sender's RoCE GID                             |
   +-                                                             -+
   |                                                               |
   +-                              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |Sender's QP number, bytes 1-2  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Sender QP byte3| Link number   |Sender's link userid, bytes 1-2|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Sender's link userid bytes, 3-4| Max links     |  Reserved     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                           Reserved                          -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Version

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

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 stack, 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

```
    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 = 2     |  length = 44  |Version|RsnCode|R|Z| Reserved  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Sender's RoCE                                                |
   +-   MAC address                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
   |                                                               |
   +-                                                             -+
   |                 Sender's RoCE GID                             |
   +-                                                             -+
   |                                                               |
   +-                              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                               |Sender's QP number, bytes 1-2  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Sender QP byte3| Link number   |Rsrvd  |  MTU  |Initial PSN    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Initial PSN, continued       |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                              -+
   |                          Reserved                             |
   +-                                                             -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Delet   group completes, as described in
3   ..4.1. 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

Version

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

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)

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

Delet   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 stack will use when sending to the peer, so that the peer can prepare its QP for the sequence number to expect.

A.3.3. ADD LINK CONTINUATION 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 = 3     |  length = 44  |Version| Rsrvd |R|  Reserved   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Linknum     | NumRTokens    |         Reserved              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-                  Rkey/Rtoken Pair                           -+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-                  Rkey/Rtoken Pair or zeroes                 -+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Reserved                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

- 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.

- 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

Version

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

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.

   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.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Reference Rkey                       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             New Rkey                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                       New Virtual Address                   -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
                    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

```
     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 = 4     |  length = 44  |Version| Rsrvd |R|A|O| Rsrvd   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Linknum     |         Reason code (bytes 1-3)               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |RsnCode byte 4 |                                               |
    +-+-+-+-+-+-+-+-+                                              -+
    |                                                               |
    +-                                                             -+
    |                                                               |
    +-                                                             -+
    |                                                               |
    +-                            Reserved                         -+
    |                                                               |
    +-                                                             -+
    |                                                               |
    +-                                                             -+
    |                                                               |
    +-                                                             -+
    |                                                               |
    +-                                                             -+
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Version

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

R

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

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' = stack (program) initiated termination (link inactivity)

o X'00040000' = LLC protocol violation

o Others TBD

Response Reason Codes:

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

o X'00200000' = Unknown Link Group (no links)

o Others TBD

A.3.5. CONFIRM 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 = 6     |  length = 44  |Version| Rsrvd |R|D|Z| Rsrvd   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   NumLinks    |  New RMB Rkey for this link (bytes 1-3)       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |ThisLink byte 4|                                               |
    +-+-+-+-+-+-+-+-+                                              -+
    |           New RMB virtual address for this link               |
    +-              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |               |                                               |
    +-+-+-+-+-+-+-+-+                                              -+
    |                                                               |
    +-   Other link RMB specification or zeros                     -+
    |                                                               |
    +-                              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                               |                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                              -+
    |                                                               |
    +-                                                             -+
    |      Other link RMB specification or zeroes                   |
    +-                                              +-+-+-+-+-+-+-+-+
    |                                               |  Reserved     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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 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 contents of this message are:

Type

Type 6 indicates CONFIRM RKEY

Length

All LLC messages are 44 bytes long

Version

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

R

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

D

Delete flag. When set indicates that the indicated RMB is being deleted

Z

Negative response flag. Set when an attempt to send CONFIRM RKEY collides with a configuration change in the link group. When set on a reply, indicates that the sender must recalculate the Rkey and and redo this exchange after the current configuration change is completed.

NumLinks

The number link/RToken pairs, including those provided in this message, to be communicated.

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.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Link number   | RMB's Rkey for the specified link (bytes 1-3) |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |New Rkey byte 4|                                               |
 +-+-+-+-+-+-+-+-+                                              -+
 |           RMB's virtual address for the specified link        |
 +-              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               |
 +-+-+-+-+-+-+-+-+
```
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. TEST LINK 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 = 7     |  length = 44  |Version| Rsrvd |R|  Reserved   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-                         User Data                           -+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-                                                             -+
   |                          Reserved                             |
   +-                                                             -+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-                                                             -+
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
                   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 stack level. A stack which receives a TEST_LINK

Delet   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

Version

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

R

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

User Data

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

A.4. RMBE control message format

The RMBE control data is communicated using 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 it 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.

```
        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
    0  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       | Type = x'FE'  | Length = 44   |      Sequence number          |
    4  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                       SMC-R alert token                       |
    8  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |         Reserved              | Producer cursor wrap seqno    |
   12  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                       Producer Cursor                         |
   16  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |         Reserved              | Consumer cursor wrap seqno    |
   20  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                       Consumer Cursor                         |
   24  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |B|P|U|R| Rsrvd |D|C|A|             Reserved                    |
   28  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
   32  +-                                                             -+
       |                                                               |
   36  +-                         Reserved                            -+
       |                                                               |
   40  +-                                                             -+
       |                                                               |
   44  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 39 RMBE Control Message Format

Type = x'FE'

This type number has the high order bit 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. Incremented with every control message send, and used to guard against processing an old control message out of sequence. 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.

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.

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. 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.

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 CMC 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.

```
          Host X -- Server                            Host Y -- Client
      +--------------------+                      +--------------------+
      | PeerID = PS1       |                      |    PeerID = PC1    |
      |            +------+                       +------+             |
      |       QP 8 |RNIC 1|    SMC-R link 1       |RNIC 2| QP 64       |
      | RKey X |   |MAC MA|<--------------------->|MAC MB|   |         |
      |        |   |GID GA|   attempted setup     |GID GB|   | RKey Y2|
      |       \/   +------+                       +------+  \/         |
      |+--------+          |                      |        +--------+ |
      || RMB    |          |                      |        | RMB    | |
      |+--------+          |                      |        +--------+ |
      |       /\   +------+                       +------+  /\         |
      |        |   |RNIC 3|                       |RNIC 4|   | Rkey W2|
      |        |   |MAC MC|                       |MAC MD|   |         |
      |       QP 9 |GID GC|                       |GID GD| QP65        |
      |            +------+                       +------+             |
      +--------------------+                      +--------------------+

              SYN / SYN-ACK / ACT TCP 3-way handshake with TCP option
           <----------------------------------------------------------->

              SMC Proposal / SMC Accept / SMC Confirm exchange
           <---------------------------------------------------------->

             CONFIRM LINK(request, link 1)
           ..........................................................>

                             CONFIRM LINK(response, link 1)
                                X...................................
                                  :
                                  : ROCE write faliure
                                  :..................................>

             SMC Decline(PC1, reason code)
            <---------------------------------------------------------

                Connection data flows over IP fabric
            <--------------------------------------------------------->

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

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 to 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 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 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.

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 is 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)

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 (not first contact)

Timer Waits for: CONFIRM LINK reply from client

Recovery action: Clean up the new link and set a timer to retry.

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.

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 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: 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 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 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: 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: assume timing window and ignore message.

LLC Message received: CONFIRM RKEY form 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: 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: 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 VLAN 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