Remote Direct Memory Access Protocl (RDMAP)

Protocol Overview

RDMAP provides seven data transfer operations. Except for the RDMA Read operation, each operation generates exactly one RDMA Message. Following is a brief overview of the RDMA Operations and RDMA Messages:

1. Send - A Send operation uses a Send Message to transfer data from the Data Source into a buffer that has not been explicitly Advertised by the Data Sink. The Send Message uses the DDP Untagged Buffer Model to transfer the ULP Message into the Data Sink's Untagged Buffer.

2. Send with Invalidate - A Send with Invalidate operation uses a Send with Invalidate Message to transfer data from the Data Source into a buffer that has not been explicitly Advertised by the Data Sink. The Send with Invalidate Message includes all functionality of the Send Message, with one addition: an STag field is included in the Send with Invalidate Message. After the message has been Placed and Delivered at the Data Sink, the Remote Peer's buffer identified by the STag can no longer be accessed remotely until the Remote Peer's ULP re-enables access and Advertises the buffer.

3. Send with Solicited Event (Send with SE) - A Send with Solicited Event operation uses a Send with Solicited Event Message to transfer data from the Data Source into an Untagged Buffer at the Data Sink. The Send with Solicited Event Message is similar to the Send Message, with one addition: when the Send with Solicited Event Message has been Placed and Delivered, an Event may be generated at the recipient, if the recipient is configured to generate such an Event.

4. Send with Solicited Event and Invalidate (Send with SE and Invalidate) - A Send with Solicited Event and Invalidate operation uses a Send with Solicited Event and Invalidate Message to transfer data from the Data Source into a buffer that has not been explicitly Advertised by the Data Sink. The Send with Solicited Event and Invalidate Message is similar to the Send with Invalidate Message, with one addition: when the Send with Solicited Event and Invalidate Message has been Placed and Delivered, an Event may be generated at the recipient, if the recipient is configured to generate such an Event.

5. Remote Direct Memory Access Write - An RDMA Write operation uses an RDMA Write Message to transfer data from the Data Source to a previously Advertised Buffer at the Data Sink.

   The ULP at the Remote Peer, which in this case is the Data Sink, enables the Data Sink Tagged Buffer for access and Advertises the buffer's size (length), location (Tagged Offset), and Steering Tag (STag) to the Data Source through a ULP-specific mechanism. The ULP at the Local Peer, which in this case is the Data Source, initiates the RDMA Write operation. The RDMA Write Message uses the DDP Tagged Buffer Model to transfer the ULP Message into the Data Sink's Tagged Buffer. Note: the STag associated with the Tagged Buffer remains valid until the ULP at the Remote Peer

invalidates it or the ULP at the Local Peer invalidates it through a Send with Invalidate or Send with Solicited Event and Invalidate.

6. Remote Direct Memory Access Read - The RDMA Read operation transfers data to a Tagged Buffer at the Local Peer, which in this case is the Data Sink, from a Tagged Buffer at the Remote Peer, which in this case is the Data Source. The ULP at the Data Source enables the Data Source Tagged Buffer for access and Advertises the buffer's size (length), location (Tagged Offset), and Steering Tag (STag) to the Data Sink through a ULP-specific mechanism. The ULP at the Data Sink enables the Data Sink Tagged Buffer for access and initiates the RDMA Read operation. The RDMA Read operation consists of a single RDMA Read Request Message and a single RDMA Read Response Message, and the latter may be segmented into multiple DDP Segments.

   The RDMA Read Request Message uses the DDP Untagged Buffer Model to Deliver the STag, starting Tagged Offset, and length for both the Data Source and Data Sink Tagged Buffers to the Remote Peer's RDMA Read Request Queue.

   The RDMA Read Response Message uses the DDP Tagged Buffer Model to Deliver the Data Source's Tagged Buffer to the Data Sink, without any involvement from the ULP at the Data Source.

   Note: the Data Source STag associated with the Tagged Buffer remains valid until the ULP at the Data Source invalidates it or the ULP at the Data Sink invalidates it through a Send with Invalidate or Send with Solicited Event and Invalidate. The Data Sink STag associated with the Tagged Buffer remains valid until the ULP at the Data Sink invalidates it.

7. Terminate - A Terminate operation uses a Terminate Message to transfer to the Remote Peer information associated with an error that occurred at the Local Peer. The Terminate Message uses the DDP Untagged Buffer Model to transfer the Message into the Data Sink's Untagged Buffer.

RDMAP Layering

RDMAP is dependent on DDP, subject to the requirements defined in Section 3.1, "Transport Requirements and Assumptions". Figure 1, "RDMAP Layering", depicts the relationship between Upper Layer Protocols (ULPs), RDMAP, DDP protocol, the framing layer, and the transport. For LLP protocol definitions of each LLP, see [MPA], [TCP], and [SCTP].

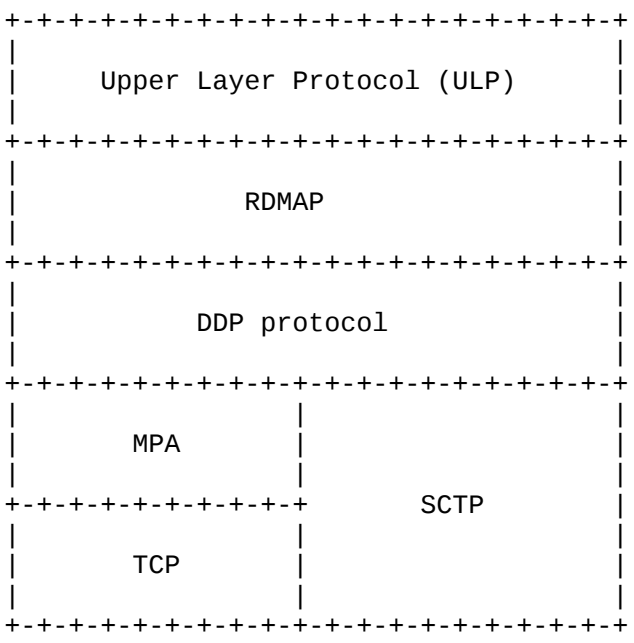

Figure 1: RDMAP Layering

If RDMAP is layered over DDP/MPA/TCP, then the respective headers and ULP Payload are arranged as follows (Note: For clarity, MPA header and CRC fields are included but MPA markers are not shown):

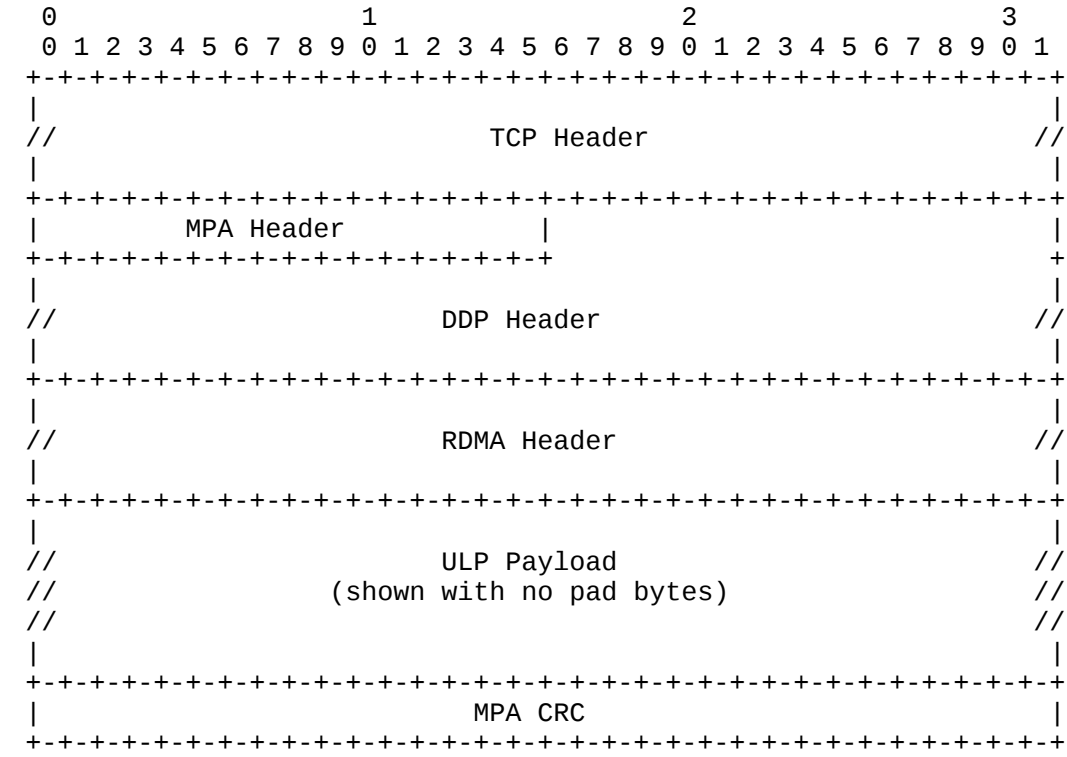

Figure 2: Example of MPA, DDP, and RDMAP Header Alignment over TCP

Direct Data Placement Protocol (DDP)

Architectural Goals

DDP has been designed with the following high-level architectural goals:

* Provide a buffer model that enables the Local Peer to Advertise a named buffer (i.e., a Tag for a buffer) to the Remote Peer, such that across the network the Remote Peer can Place data into the buffer at Remote-Peer-specified locations. This is referred to as the Tagged Buffer Model.

* Provide a second receive buffer model that preserves ULP message boundaries from the Remote Peer and keeps the Local Peer's buffers anonymous (i.e., Untagged). This is referred to as the Untagged Buffer Model.

* Provide reliable, in-order Delivery semantics for both Tagged and Untagged Buffer Models.

* Provide segmentation and reassembly of Upper Layer Protocol (ULP) messages. ULP corresponds in most cases to RDMA protocol.

* Enable the ULP Buffer to be used as a reassembly buffer, without a need for a copy, even if incoming DDP Segments arrive out of order. This requires the protocol to separate Data Placement of ULP Payload contained in an incoming DDP Segment from Data Delivery of completed ULP Messages.

* If the Lower Layer Protocol (LLP) supports multiple LLP Streams within an LLP Connection, provide the above capabilities independently on each LLP Stream and enable the capability to be exported on a per-LLP-Stream basis to the ULP. LLP corresponds to SCTP or TCP with modifications.

Protocol Overview

DDP supports two basic data transfer models - a Tagged Buffer data transfer model and an Untagged Buffer data transfer model.

The Tagged Buffer data transfer model requires the Data Sink to send the Data Source an identifier for the ULP Buffer, referred to as a Steering Tag (STag). The STag is transferred to the Data Source using a ULP-defined method. Once the Data Source ULP has an STag for a destination ULP Buffer, it can request that DDP send the ULP data to the destination ULP Buffer by specifying the STag to DDP. Note that the Tagged Buffer does not have to be filled starting at the beginning of the ULP Buffer. The ULP Data Source can provide an arbitrary offset into the ULP Buffer.

The Untagged Buffer data transfer model enables data transfer to occur without requiring the Data Sink to Advertise a ULP Buffer to the Data Source. The Data Sink can queue up a series of receive ULP Buffers. An Untagged DDP Message from the Data Source consumes an Untagged Buffer at the Data Sink. Because DDP is message oriented, even if the Data Source sends a DDP Message payload smaller than the receive ULP Buffer, the partially filled receive ULP Buffer is delivered to the ULP anyway. If the Data Source sends a DDP Message payload larger than the receive ULP Buffer, it results in an error.