                Direct Data Placement over Reliable Transports

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   The Direct Data Placement protocol provides information to Place the
   incoming data directly into an upper layer protocol's receive buffer
   without intermediate buffers. This removes excess CPU and memory
   utilization associated with transferring data through the
   intermediate buffers.

Table of Contents

Table of Figures

1. Introduction

   Note: The capitalization of certain words in this document indicates
   they are being used with the specific meaning given in the glossary
   (Section 2).

   Direct Data Placement Protocol (DDP) enables an Upper Layer Protocol
   (ULP) to send data to a Data Sink without requiring the Data Sink to
   Place the data in an intermediate buffer - thus, when the data
   arrives at the Data Sink, the network interface can Place the data
   directly into the ULP's buffer. This can enable the Data Sink to
   consume substantially less memory bandwidth than a buffered model
   because the Data Sink is not required to move the data from the
   intermediate buffer to the final destination. Additionally, this can
   enable the network protocol to consume substantially fewer CPU cycles
   than if the CPU was used to move the data, and this can remove the
   bandwidth limitation of only being able to move data as fast as the
   CPU can copy the data.

   DDP preserves ULP record boundaries (messages) while providing a
   variety of data transfer mechanisms and completion mechanisms to be
   used to transfer ULP messages.

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in RFC 2119 [RFC2119].

1.1. 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 ULP messages.

   * 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.

1.2. 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.

   There are several key differences between the Tagged and Untagged
   Buffer Model:

   * For the Tagged Buffer Model, the Data Source specifies which
     received Tagged Buffer will be used for a specific Tagged DDP
     Message (sender-based ULP Buffer management). For the Untagged
     Buffer Model, the Data Sink specifies the order in which
     Untagged Buffers will be consumed as Untagged DDP Messages are
     received (receiver-based ULP Buffer management).

   * For the Tagged Buffer Model, the ULP at the Data Sink must
     Advertise the ULP Buffer to the Data Source through a ULP

specific mechanism before data transfer can occur. For the
Untagged Buffer Model, data transfer can occur without an end-
to-end explicit ULP Buffer Advertisement. Note, however, that
the ULP needs to address flow control issues.

* For the Tagged Buffer Model, a DDP Message can start at an
  arbitrary offset within the Tagged Buffer. For the Untagged
  Buffer Model, a DDP Message can only start at offset 0.

* The Tagged Buffer Model allows multiple DDP Messages targeted to
  a Tagged Buffer with a single ULP Buffer Advertisement. The
  Untagged Buffer Model requires associating a receive ULP Buffer
  for each DDP Message targeted to an Untagged Buffer.

Either data transfer model Places a ULP Message into a DDP Message.
Each DDP Message is then sliced into DDP Segments that are intended
to fit within a lower-layer-protocol's (LLP) Maximum Upper Layer
Protocol Data Unit (MULPDU). Thus, the ULP can post arbitrarily
sized ULP Messages, containing up to $2^{32} - 1$ octets of ULP Payload,
and DDP slices the ULP message into DDP Segments, which are
reassembled transparently at the Data Sink.

DDP provides in-order delivery for the ULP. However, DDP
differentiates between Data Delivery and Data Placement. DDP
provides enough information in each DDP Segment to allow the ULP
Payload in each inbound DDP Segment payloads to be directly Placed
into the correct ULP Buffer, even when the DDP Segments arrive out-
of-order. Thus, DDP enables the reassembly of ULP Payload contained
in DDP Segments of a DDP Message into a ULP Message to occur within
the ULP Buffer, therefore eliminating the traditional copy out of the
reassembly buffer into the ULP Buffer.

A DDP Message's payload is Delivered to the ULP when:

* all DDP Segments of a DDP Message have been completely received,
  and the payload of the DDP Message has been Placed into the
  associated ULP Buffer,

* all prior DDP Messages have been Placed, and

* all prior DDP Message Deliveries have been performed.

The LLP under DDP may support a single LLP Stream of data per
connection (e.g., TCP [TCP]) or multiple LLP Streams of data per
connection (e.g., SCTP [SCTP]). But in either case, DDP is specified
such that each DDP Stream is independent and maps to a single LLP
Stream. Within a specific DDP Stream, the LLP Stream is required to

provide in-order, reliable Delivery. Note that DDP has no ordering
guarantees between DDP Streams.

A DDP protocol could potentially run over reliable Delivery LLPs or
unreliable Delivery LLPs. This specification requires reliable, in
order Delivery LLPs.

1.3. DDP Layering

DDP is intended to be LLP independent, subject to the requirements
defined in section 3. However, DDP was specifically defined to be
part of a family of protocols that were created to work well
together, as shown in Figure 1, DDP Layering. For LLP protocol
definitions of each LLP, see Marker PDU Aligned Framing for TCP
Specification [MPA] and Stream Control Transmission Protocol (SCTP)
Direct Data Placement (DDP) Adaptation [SCTPDDP].

DDP enables direct data Placement capability for any ULP, but it has
been specifically designed to work well with Remote Direct Memory
Access Protocol (RDMAP) (see [RDMAP]), and is part of the iWARP
protocol suite.

```
                      +------------------+
                      |                  |
                      |     RDMA ULP     |
                      |                  |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |             |                 |
         |    ULP      |     RDMAP       |
         |             |                 |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                               |
         |         DDP protocol          |
         |                               |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |             |                 |
         |    MPA      |                 |
         |             |                 |
         |             |                 |
         +-+-+-+-+-+-+-+-+     SCTP       |
         |             |                 |
         |    TCP      |                 |
         |             |                 |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
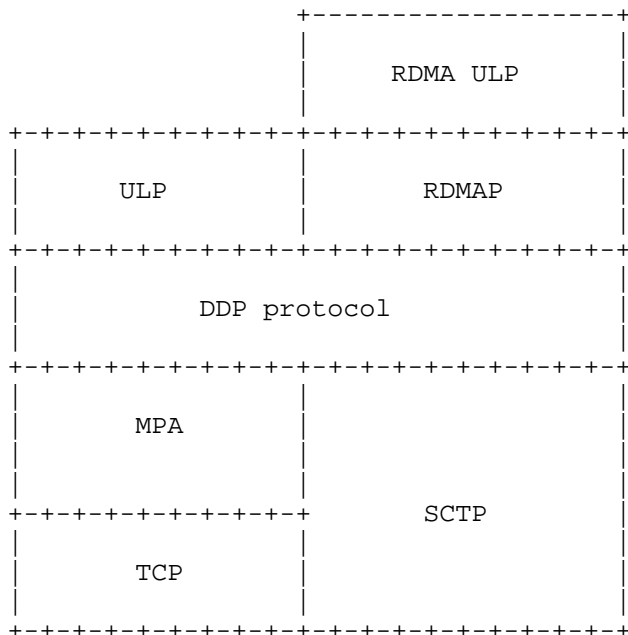
Figure 1: DDP Layering

If DDP is layered below RDMAP and on top of MPA and TCP, then the
respective headers and payload are arranged as follows (Note: For
clarity, MPA header and CRC are included, but framing markers are not
shown.):

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     //                       TCP Header                            //
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |            MPA Header          |                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
     |                                                               |
     //                       DDP Header                            //
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     //                       RDMAP Header                          //
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     //                                                             //
     //                       RDMAP ULP Payload                     //
     //                                                             //
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                           MPA CRC                             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
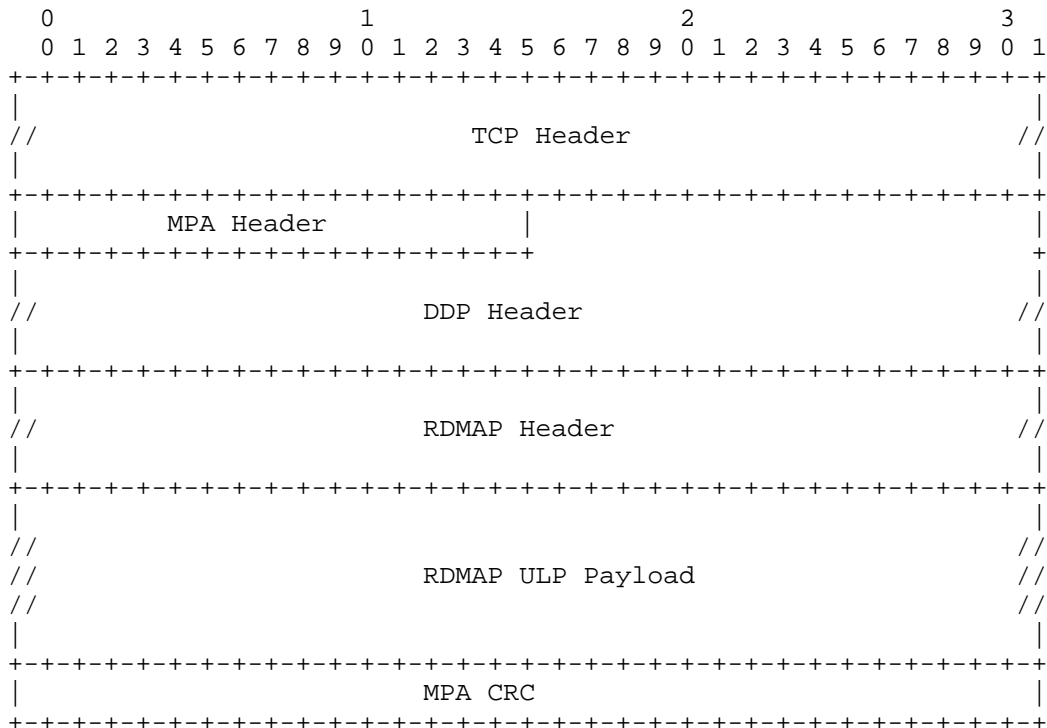
             Figure 2: MPA, DDP, and RDMAP Header Alignment

2. Glossary

2.1. General

   Advertisement (Advertised, Advertise, Advertisements, Advertises) -
      The act of informing a Remote Peer that a local RDMA Buffer is
      available to it. A Node makes available an RDMA Buffer for
      incoming RDMA Read or RDMA Write access by informing its RDMA/DDP
      peer of the Tagged Buffer identifiers (STag, base address,
      length). This Advertisement of Tagged Buffer information is not
      defined by RDMA/DDP and is left to the ULP. A typical method
      would be for the Local Peer to embed the Tagged Buffer's Steering
      Tag, address, and length in a Send message destined for the
      Remote Peer.

Data Delivery (Delivery, Delivered, Delivers) - Delivery is defined
    as the process of informing the ULP or consumer that a particular
    message is available for use. This is specifically different
    from "Placement", which may generally occur in any order, while
    the order of "Delivery" is strictly defined. See "Data
    Placement".

Data Sink - The peer receiving a data payload. Note that the Data
    Sink can be required to both send and receive RDMA/DDP Messages
    to transfer a data payload.

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

Delivery (Delivered, Delivers) - See Data Delivery in Section 2.1.

iWARP - A suite of wire protocols comprised of RDMAP [RDMAP], DDP
    (this specification), and Marker PDU Aligned Framing for TCP
    (MPA) [MPA]. The iWARP protocol suite may be layered above TCP,
    SCTP, or other transport protocols.

Local Peer - The RDMA/DDP protocol implementation on the local end of
    the connection. Used to refer to the local entity when
    describing a protocol exchange or other interaction between two
    Nodes.

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

Placement (Placed, Places) - See "Data Placement" in Section 2.3

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

RNIC - RDMA Enabled Network Interface Controller. In this context,
    this would be a network I/O adapter or embedded controller with
    iWARP functionality.

ULP - Upper Layer Protocol. The protocol layer above the protocol
    layer currently being referenced. The ULP for RDMA/DDP is
    expected to be an Operating System (OS), application, adaptation
    layer, or proprietary device. The RDMA/DDP documents do not

specify a ULP -- they provide a set of semantics that allow a ULP
to be designed to utilize RDMA/DDP.

ULP Message - The ULP data that is handed to a specific protocol
layer for transmission. Data boundaries are preserved as they
are transmitted through iWARP.

ULP Payload - The ULP data that is contained within a single protocol
segment or packet (e.g., a DDP Segment).

2.2. LLP

LLP - Lower Layer Protocol. The protocol layer beneath the protocol
layer currently being referenced. For example, for DDP, the LLP
is SCTP DDP Adaptation, MPA, or other transport protocols. For
RDMA, the LLP is DDP.

LLP Connection - Corresponds to an LLP transport-level connection
between the peer LLP layers on two nodes.

LLP Stream - Corresponds to a single LLP transport-level stream
between the peer LLP layers on two Nodes. One or more LLP
Streams may map to a single transport-level LLP Connection. For
transport protocols that support multiple streams per connection
(e.g., SCTP), an LLP Stream corresponds to one transport-level
stream.

MULPDU - Maximum Upper Layer Protocol Data Unit (MULPDU). The
current maximum size of the record that is acceptable for DDP to
pass to the LLP for transmission.

ULPDU - Upper Layer Protocol Data Unit. The data record defined by
the layer above MPA.

2.3. Direct Data Placement (DDP)

Data Placement (Placement, Placed, Places) - For DDP, this term is
specifically used to indicate the process of writing to a Data
Buffer by a DDP implementation. DDP Segments carry Placement
information, which may be used by the receiving DDP
implementation to perform Data Placement of the DDP Segment ULP
Payload. See "Data Delivery" and "Direct Data Placement".

DDP Abortive Teardown - The act of closing a DDP Stream without
attempting to complete in-progress and pending DDP Messages.

DDP Graceful Teardown - The act of closing a DDP Stream such that all
    in-progress and pending DDP Messages are allowed to complete
    successfully.

DDP Control Field - A fixed 8-bit field in the DDP Header.

DDP Header - The header present in all DDP Segments. The DDP Header
    contains control and Placement fields that are used to define the
    final Placement location for the ULP Payload carried in a DDP
    Segment.

DDP Message - A ULP-defined unit of data interchange, which is
    subdivided into one or more DDP Segments. This segmentation may
    occur for a variety of reasons, including segmentation to respect
    the maximum segment size of the underlying transport protocol.

DDP Segment - The smallest unit of data transfer for the DDP
    protocol. It includes a DDP Header and ULP Payload (if present).
    A DDP Segment should be sized to fit within the Lower Layer
    Protocol MULPDU.

DDP Stream - A sequence of DDP messages whose ordering is defined by
    the LLP. For SCTP, a DDP Stream maps directly to an SCTP stream.
    For MPA, a DDP Stream maps directly to a TCP connection, and a
    single DDP Stream is supported. Note that DDP has no ordering
    guarantees between DDP Streams.

DDP Stream Identifier (ID) - An identifier for a DDP Stream.

Direct Data Placement - A mechanism whereby ULP data contained within
    DDP Segments may be Placed directly into its final destination in
    memory without processing of the ULP. This may occur even when
    the DDP Segments arrive out of order. Out-of-order Placement
    support may require the Data Sink to implement the LLP and DDP as
    one functional block.

Direct Data Placement Protocol (DDP) - Also, a wire protocol that
    supports Direct Data Placement by associating explicit memory
    buffer placement information with the LLP payload units.

Message Offset (MO) - For the DDP Untagged Buffer Model, specifies
    the offset, in octets, from the start of a DDP Message.

Message Sequence Number (MSN) - For the DDP Untagged Buffer Model,
    specifies a sequence number that is increasing with each DDP
    Message.

   Protection Domain (PD) - A mechanism used to associate a DDP Stream
      and an STag. Under this mechanism, the use of an STag is valid
      on a DDP Stream if the STag has the same Protection Domain
      Identifier (PD ID) as the DDP Stream.

   Protection Domain Identifier (PD ID) - An identifier for the
      Protection Domain.

   Queue Number (QN) - For the DDP Untagged Buffer Model, identifies a
      destination Data Sink queue for a DDP Segment.

   Steering Tag - An identifier of a Tagged Buffer on a Node, valid as
      defined within a protocol specification.

   STag - Steering Tag

   Tagged Buffer - A buffer that is explicitly Advertised to the Remote
      Peer through exchange of an STag, Tagged Offset, and length.

   Tagged Buffer Model - A DDP data transfer model used to transfer
      Tagged Buffers from the Local Peer to the Remote Peer.

   Tagged DDP Message - A DDP Message that targets a Tagged Buffer.

   Tagged Offset (TO) - The offset within a Tagged Buffer on a Node.

   ULP Buffer - A buffer owned above the DDP layer and Advertised to the
      DDP layer either as a Tagged Buffer or an Untagged ULP Buffer.

   ULP Message Length - The total length, in octets, of the ULP Payload
      contained in a DDP Message.

   Untagged Buffer - A buffer that is not explicitly Advertised to the
      Remote Peer.

   Untagged Buffer Model - A DDP data transfer model used to transfer
      Untagged Buffers from the Local Peer to the Remote Peer.

   Untagged DDP Message - A DDP Message that targets an Untagged Buffer.

3. Reliable Delivery LLP Requirements

   Any protocol that can serve as an LLP to DDP MUST meet the following
   requirements.

   1. LLPs MUST expose MULPDU and MULPDU changes. This is required so
       that the DDP layer can perform segmentation aligned with the
       MULPDU and can adapt as MULPDU changes come about. The corner
       case of how to handle outstanding requests during a MULPDU change
       is covered by the requirements below.

   2. In the event of a MULPDU change, DDP MUST NOT be required by the
       LLP to re-segment DDP Segments that have been previously posted
       to the LLP. Note that under pathological conditions the LLP may
       change the Advertised MULPDU more frequently than the queue of
       previously posted DDP Segment transmit requests is flushed.
       Under this pathological condition, the LLP transmit queue can
       contain DDP Messages for which multiple updates to the
       corresponding MULPDU have occurred subsequent to posting of the
       messages. Thus, there may be no correlation between the queued
       DDP Segment(s) and the LLP's current value of MULPDU.

   3. The LLP MUST ensure that, if it accepts a DDP Segment, it will
       transfer it reliably to the receiver or return with an error
       stating that the transfer failed to complete.

   4. The LLP MUST preserve DDP Segment and Message boundaries at the
       Data Sink.

   5. The LLP MAY provide the incoming segments out of order for
       Placement, but if it does, it MUST also provide information that
       specifies what the sender-specified order was.

   6. LLP MUST provide a strong digest (at least equivalent to CRC32-C)
       to cover at least the DDP Segment. It is believed that some of
       the existing data integrity digests are not sufficient, and that
       direct memory transfer semantics requires a stronger digest than,
       for example, a simple checksum.

   7. On receive, the LLP MUST provide the length of the DDP Segment
       received. This ensures that DDP does not have to carry a length
       field in its header.

   8. If an LLP does not support teardown of an LLP Stream independent
       of other LLP Streams, and a DDP error occurs on a specific DDP
       Stream, then the LLP MUST label the associated LLP Stream as an
       erroneous LLP Stream and MUST NOT allow any further data transfer

on that LLP Stream after DDP requests the associated DDP Stream
to be torn down.

9. For a specific LLP Stream, the LLP MUST provide a mechanism to
    indicate that the LLP Stream has been gracefully torn down. For
    a specific LLP Connection, the LLP MUST provide a mechanism to
    indicate that the LLP Connection has been gracefully torn down.

    Note that, if the LLP does not allow an LLP Stream to be torn
    down independently of the LLP Connection, the above requirements
    allow the LLP to notify DDP of both events at the same time.

10. For a specific LLP Connection, when all LLP Streams are either
    gracefully torn down or are labeled as erroneous LLP Streams, the
    LLP Connection MUST be torn down.

11. The LLP MUST NOT pass a duplicate DDP Segment to the DDP layer
    after it has passed all the previous DDP Segments to the DDP
    layer and the associated ordering information for the previous
    DDP Segments and the current DDP Segment.

4. Header Format

   DDP has two different header formats: one for Data Placement into
   Tagged Buffers, and the other for Data Placement into Untagged
   Buffers. See Section 5.1 for a description of the two models.

4.1. DDP Control Field

   The first 8 bits of the DDP Header carry a DDP Control Field that is
   common between the two formats. It is shown below in Figure 3,
   offset by 16 bits to accommodate the MPA header defined in [MPA].
   The MPA header is only present if DDP is layered on top of MPA.

```
 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
                                +-+-+-+-+-+-+-+-+
                                |T|L| Rsvd  |DV |
                                +-+-+-+-+-+-+-+-+
```
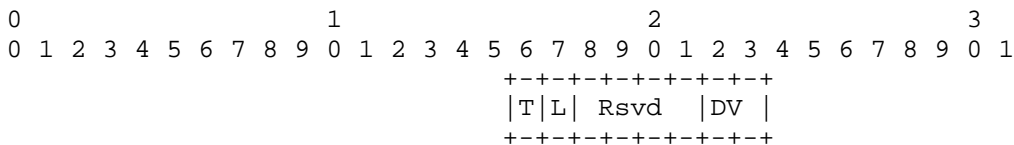
                      Figure 3: DDP Control Field

   T - Tagged flag: 1 bit.

      Specifies the Tagged or Untagged Buffer Model. If set to one,
      the ULP Payload carried in this DDP Segment MUST be Placed into a
      Tagged Buffer.

If set to zero, the ULP Payload carried in this DDP Segment MUST
be Placed into an Untagged Buffer.

L - Last flag: 1 bit.

Specifies whether the DDP Segment is the last segment of a DDP
Message. It MUST be set to one on the last DDP Segment of every
DDP Message. It MUST NOT be set to one on any other DDP Segment.

The DDP Segment with the L bit set to 1 MUST be posted to the LLP
after all other DDP Segments of the associated DDP Message have
been posted to the LLP. For an Untagged DDP Message, the DDP
Segment with the L bit set to 1 MUST carry the highest MO.

If the Last flag is set to one, the DDP Message payload MUST be
Delivered to the ULP after:

o Placement of all DDP Segments of this DDP Message and all
   prior DDP Messages, and

o Delivery of each prior DDP Message.

If the Last flag is set to zero, the DDP Segment is an
intermediate DDP Segment.

Rsvd - Reserved: 4 bits.

Reserved for future use by the DDP protocol. This field MUST be
set to zero on transmit, and not checked on receive.

DV - Direct Data Placement Protocol Version: 2 bits.

The version of the DDP Protocol in use. This field MUST be set
to one to indicate the version of the specification described in
this document. The value of DV MUST be the same for all the DDP
Segments transmitted or received on a DDP Stream.

4.2. DDP Tagged Buffer Model Header

Figure 4 shows the DDP Header format that MUST be used in all DDP
Segments that target Tagged Buffers. It includes the DDP Control
Field previously defined in Section 4.1. (Note: In Figure 4, the DDP
Header is offset by 16 bits to accommodate the MPA header defined in
[MPA]. The MPA header is only present if DDP is layered on top of
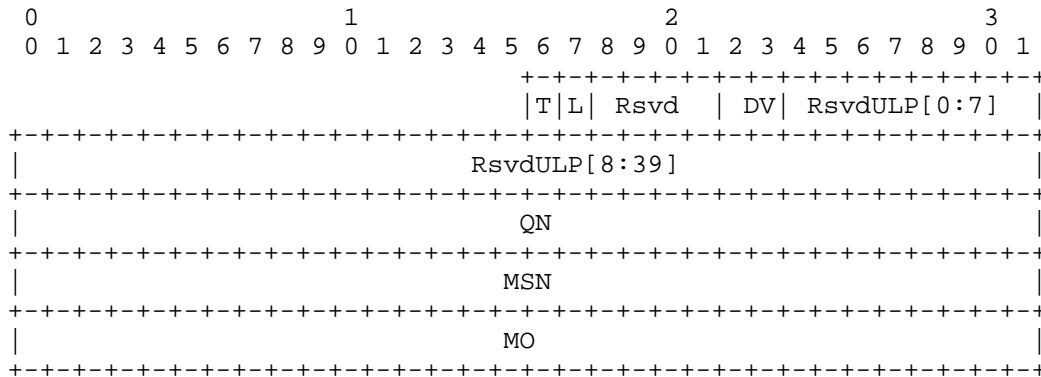MPA.)

```
     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
                                    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                    |T|L| Rsvd  | DV|   RsvdULP     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                             STag                              |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     +                             TO                                +
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                    Figure 4: Tagged Buffer DDP Header

     T is set to one.

     RsvdULP - Reserved for use by the ULP: 8 bits.

        The RsvdULP field is opaque to the DDP protocol and can be
        structured in any way by the ULP. At the Data Source, DDP MUST
        set RsvdULP Field to the value specified by the ULP. It is
        transferred unmodified from the Data Source to the Data Sink. At
        the Data Sink, DDP MUST provide the RsvdULP field to the ULP when
        the DDP Message is delivered. Each DDP Segment within a specific
        DDP Message MUST contain the same value for this field. The Data
        Source MUST ensure that each DDP Segment within a specific DDP
        Message contains the same value for this field.

     STag - Steering Tag: 32 bits.

        The Steering Tag identifies the Data Sink's Tagged Buffer. The
        STag MUST be valid for this DDP Stream. The STag is associated
        with the DDP Stream through a mechanism that is outside the scope
        of the DDP Protocol specification. At the Data Source, DDP MUST
        set the STag field to the value specified by the ULP. At the
        Data Sink, the DDP MUST provide the STag field when the ULP
        Message is delivered. Each DDP Segment within a specific DDP
        Message MUST contain the same value for this field and MUST be
        the value supplied by the ULP. The Data Source MUST ensure that
        each DDP Segment within a specific DDP Message contains the same
        value for this field.

     TO - Tagged Offset: 64 bits.

        The Tagged Offset specifies the offset, in octets, within the
        Data Sink's Tagged Buffer, where the Placement of ULP Payload
        contained in the DDP Segment starts. A DDP Message MAY start at
        an arbitrary TO within a Tagged Buffer.

4.3. DDP Untagged Buffer Model Header

   Figure 5 shows the DDP Header format that MUST be used in all DDP
   Segments that target Untagged Buffers. It includes the DDP Control
   Field previously defined in Section 4.1. (Note: In Figure 5, the DDP
   Header is offset by 16 bits to accommodate the MPA header defined in
   [MPA]. The MPA header is only present if DDP is layered on top of
   MPA.)

```
 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
                                +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                |T|L| Rsvd  | DV| RsvdULP[0:7]  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          RsvdULP[8:39]                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                              QN                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                              MSN                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                              MO                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                    Figure 5: Untagged Buffer DDP Header

   T is set to zero.

   RsvdULP - Reserved for use by the ULP: 40 bits.

      The RsvdULP field is opaque to the DDP protocol and can be
      structured in any way by the ULP. At the Data Source, DDP MUST
      set RsvdULP Field to the value specified by the ULP. It is
      transferred unmodified from the Data Source to the Data Sink. At
      the Data Sink, DDP MUST provide RsvdULP field to the ULP when the
      ULP Message is Delivered. Each DDP Segment within a specific DDP
      Message MUST contain the same value for the RsvdULP field. At
      the Data Sink, the DDP implementation is NOT REQUIRED to verify
      that the same value is present in the RsvdULP field of each DDP
      Segment within a specific DDP Message and MAY provide the value
      from any one of the received DDP Segment to the ULP when the ULP
      Message is Delivered.

   QN - Queue Number: 32 bits.

      The Queue Number identifies the Data Sink's Untagged Buffer queue
      referenced by this header. Each DDP segment within a specific
      DDP message MUST contain the same value for this field and MUST
      be the value supplied by the ULP at the Data Source. The Data
      Source MUST ensure that each DDP Segment within a specific DDP
      Message contains the same value for this field.

   MSN - Message Sequence Number: 32 bits.

      The Message Sequence Number specifies a sequence number that MUST
      be increased by one (modulo 2^32) with each DDP Message targeting
      the specific Queue Number on the DDP Stream associated with this
      DDP Segment. The initial value for MSN MUST be one. The MSN
      value MUST wrap to 0 after a value of 0xFFFFFFFF. Each DDP
      segment within a specific DDP message MUST contain the same value
      for this field. The Data Source MUST ensure that each DDP
      Segment within a specific DDP Message contains the same value for
      this field.

   MO - Message Offset: 32 bits.

      The Message Offset specifies the offset, in octets, from the
      start of the DDP Message represented by the MSN and Queue Number
      on the DDP Stream associated with this DDP Segment. The MO
      referencing the first octet of the DDP Message MUST be set to
      zero by the DDP layer.

4.4. DDP Segment Format

   Each DDP Segment MUST contain a DDP Header. Each DDP Segment may
   also contain ULP Payload. Following is the DDP Segment format:

```
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |  DDP  |                                       |
           | Header|          ULP Payload (if any)         |
           |       |                                       |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
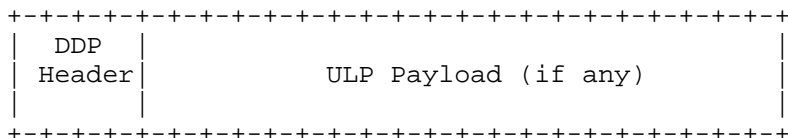
                    Figure 6: DDP Segment Format

5. Data Transfer

   DDP supports multi-segment DDP Messages. Each DDP Message is
   composed of one or more DDP Segments. Each DDP Segment contains a
   DDP Header. The DDP Header contains the information required by the
   receiver to Place any ULP Payload included in the DDP Segment.

5.1. DDP Tagged or Untagged Buffer Models

   DDP uses two basic buffer models for the Placement of the ULP
   Payload: Tagged Buffer Model and Untagged Buffer Model.

5.1.1. Tagged Buffer Model

   The Tagged Buffer Model is used by the Data Source to transfer a DDP
   Message into a Tagged Buffer at the Data Sink that has been
   previously Advertised to the Data Source. An STag identifies a
   Tagged Buffer. For the Placement of a DDP Message using the Tagged
   Buffer Model, the STag is used to identify the buffer, and the TO is
   used to identify the offset within the Tagged Buffer into which the
   ULP Payload is transferred. The protocol used to Advertise the
   Tagged Buffer is outside the scope of this specification (i.e., ULP
   specific). A DDP Message can start at an arbitrary TO within a
   Tagged Buffer.

   Additionally, a Tagged Buffer can potentially be written multiple
   times. This might be done for error recovery or because a buffer is
   being re-used after some ULP specific synchronization mechanism.

5.1.2. Untagged Buffer Model

   The Untagged Buffer Model is used by the Data Source to transfer a
   DDP Message to the Data Sink into a queued buffer.

   The DDP Queue Number is used by the ULP to separate ULP messages into
   different queues of receive buffers. For example, if two queues were
   supported, the ULP could use one queue to post buffers handed to it
   by the application above the ULP, and it could use the other queue
   for buffers that are only consumed by ULP-specific control messages.
   This enables the separation of ULP control messages from opaque ULP
   Payload when using Untagged Buffers.

   The DDP Message Sequence Number can be used by the Data Sink to
   identify the specific Untagged Buffer. The protocol used to
   communicate how many buffers have been queued is outside the scope of
   this specification. Similarly, the exact implementation of the
   buffer queue is outside the scope of this specification.

5.2. Segmentation and Reassembly of a DDP Message

   At the Data Source, the DDP layer MUST segment the data contained in
   a ULP message into a series of DDP Segments, where each DDP Segment
   contains a DDP Header and ULP Payload, and MUST be no larger than the
   MULPDU value Advertised by the LLP. The ULP Message Length MUST be
   less than 2^32. At the Data Source, the DDP layer MUST send all the
   data contained in the ULP message. At the Data Sink, the DDP layer
   MUST Place the ULP Payload contained in all valid incoming DDP
   Segments associated with a DDP Message into the ULP Buffer.

   DDP Message segmentation at the Data Source is accomplished by
   identifying a DDP Message (which corresponds one-to-one with a ULP
   Message) uniquely and then, for each associated DDP Segment of a DDP
   Message, by specifying an octet offset for the portion of the ULP
   Message contained in the DDP Segment.

   For an Untagged DDP Message, the combination of the QN and MSN
   uniquely identifies a DDP Message. The octet offset for each DDP
   Segment of a Untagged DDP Message is the MO field. For each DDP
   Segment of a Untagged DDP Message, the MO MUST be set to the octet
   offset from the first octet in the associated ULP Message (which is
   defined to be zero) to the first octet in the ULP Payload contained
   in the DDP Segment.

   For example, if the ULP Untagged Message was 2048 octets, and the
   MULPDU was 1500 octets, the Data Source would generate two DDP
   Segments, one with MO = 0, containing 1482 octets of ULP Payload, and
   a second with MO = 1482, containing 566 octets of ULP Payload. In
   this example, the amount of ULP Payload for the first DDP Segment was
   calculated as:

        1482 = 1500 (MULPDU) - 18 (for the DDP Header)

   For a Tagged DDP Message, the STag and TO, combined with the in-order
   delivery characteristics of the LLP, are used to segment and
   reassemble the ULP Message. Because the initial octet offset (the TO
   field) can be non-zero, recovery of the original ULP Message boundary
   cannot be done in the general case without an additional ULP Message.

      Implementers' note: One implementation, valid for some ULPs such
      as RDMAP, is to not directly support recovery of the ULP Message
      boundary for a Tagged DDP Message. For example, the ULP may wish
      to have the Local Peer use small buffers at the Data Source even
      when the ULP at the Data Sink has Advertised a single large
      Tagged Buffer for this data transfer. In this case, the ULP may
      choose to use the same STag for multiple consecutive ULP
      Messages. Thus, a non-zero initial TO and re-use of the STag

effectively enable the ULP to implement segmentation and
reassembly due to ULP-specific constraints. See [RDMAP] for
details of how this is done.

A different implementation of a ULP could use an Untagged DDP
Message (sent after the Tagged DDP Message) that details the
initial TO for the STag that was used in the Tagged DDP Message.
And finally, another implementation of a ULP could choose to
always use an initial TO of zero such that no additional message
is required to convey the initial TO used in a Tagged DDP
Message.

Regardless of whether the ULP chooses to recover the original ULP
Message boundary at the Data Sink for a Tagged DDP Message, DDP
supports segmentation and reassembly of the Tagged DDP Message. The
STag is used to identify the ULP Buffer at the Data Sink, and the TO
is used to identify the octet-offset within the ULP Buffer referenced
by the STag. The ULP at the Data Source MUST specify the STag and
the initial TO when the ULP Message is handed to DDP.

For each DDP Segment of a Tagged DDP Message, the TO MUST be set to
the octet offset from the first octet in the associated ULP Message
to the first octet in the ULP Payload contained in the DDP Segment,
plus the TO assigned to the first octet in the associated ULP
Message.

For example, if the ULP Tagged Message was 2048 octets with an
initial TO of 16384, and the MULPDU was 1500 octets, the Data Source
would generate two DDP Segments: one with TO = 16384, containing the
first 1486 octets of ULP payload, and a second with TO = 17870,
containing 562 octets of ULP payload. In this example, the amount of
ULP payload for the first DDP Segment was calculated as:

    1486 = 1500 (MULPDU) - 14 (for the DDP Header)

A zero-length DDP Message is allowed and MUST consume exactly one DDP
Segment. Only the DDP Control and RsvdULP Fields MUST be valid for a
zero-length Tagged DDP Segment. The STag and TO fields MUST NOT be
checked for a zero-length Tagged DDP Message.

For either Untagged or Tagged DDP Messages, the Data Sink is not
required to verify that the entire ULP Message has been received.

5.3. Ordering Among DDP Messages

   Messages passed through the DDP MUST conform to the ordering rules
   defined in this section.

   At the Data Source, DDP:

      * MUST transmit DDP Messages in the order they were submitted to
        the DDP layer,

      * SHOULD transmit DDP Segments within a DDP Message in increasing
        MO order for Untagged DDP Messages, and in increasing TO order
        for Tagged DDP Messages.

   At the Data Sink, DDP (Note: The following rules are motivated by LLP
   implementations that separate Placement and Delivery.):

      * MAY perform Placement of DDP Segments out of order,

      * MAY perform Placement of a DDP Segment more than once,

      * MUST Deliver a DDP Message to the ULP at most once,

      * MUST Deliver DDP Messages to the ULP in the order they were sent
        by the Data Source.

5.4. DDP Message Completion and Delivery

   At the Data Source, DDP Message transfer is considered completed when
   the reliable, in-order transport LLP has indicated that the transfer
   will occur reliably. Note that this in no way restricts the LLP from
   buffering the data at either the Data Source or Data Sink. Thus, at
   the Data Source, completion of a DDP Message does not necessarily
   mean that the Data Sink has received the message.

   At the Data Sink, DDP MUST Deliver a DDP Message if and only if all
   of the following are true:

      * the last DDP Segment of the DDP Message had its Last flag set,

      * all of the DDP Segments of the DDP Message have been Placed,

      * all preceding DDP Messages have been Placed, and

      * each preceding DDP Message has been Delivered to the ULP.

At the Data Sink, DDP MUST provide the ULP Message Length to the ULP
when an Untagged DDP Message is Delivered. The ULP Message Length
may be calculated by adding the MO and the ULP Payload length in the
last DDP Segment (with the Last flag set) of an Untagged DDP Message.

At the Data Sink, DDP MUST provide the RsvdULP Field of the DDP
Message to the ULP when the DDP Message is delivered.

6. DDP Stream Setup and Teardown

   This section describes LLP independent issues related to DDP Stream
   setup and teardown.

6.1. DDP Stream Setup

   It is expected that the ULP will use a mechanism outside the scope of
   this specification to establish an LLP Connection, and that the LLP
   Connection will support one or more LLP Streams (e.g., MPA/TCP or
   SCTP). After the LLP sets up the LLP Stream, it will enable a DDP
   Stream on a specific LLP Stream at an appropriate point.

   The ULP is required to enable both endpoints of an LLP Stream for DDP
   data transfer at the same time, in both directions; this is necessary
   so that the Data Sink can properly recognize the DDP Segments.

6.2. DDP Stream Teardown

   DDP MUST NOT independently initiate Stream Teardown. DDP either
   responds to a stream being torn down by the LLP or processes a
   request from the ULP to tear down a stream. DDP Stream teardown
   disables DDP capabilities on both endpoints. For connection-oriented
   LLPs, DDP Stream teardown MAY result in underlying LLP Connection
   teardown.

6.2.1. DDP Graceful Teardown

   It is up to the ULP to ensure that DDP teardown happens on both
   endpoints of the DDP Stream at the same time; this is necessary so
   that the Data Sink stops trying to interpret the DDP Segments.

   If the Local Peer ULP indicates graceful teardown, the DDP layer on
   the Local Peer SHOULD ensure that all ULP data would be transferred
   before the underlying LLP Stream and Connection are torn down, and
   any further data transfer requests by the Local Peer ULP MUST return
   an error.

If the DDP layer on the Local Peer receives a graceful teardown
request from the LLP, any further data received after the request is
considered an error and MUST cause the DDP Stream to be abortively
torn down.

If the Local Peer LLP supports a half-closed LLP Stream, on the
receipt of an LLP graceful teardown request of the DDP Stream, DDP
SHOULD indicate the half-closed state to the ULP, and continue to
process outbound data transfer requests normally. Following this
event, when the Local Peer ULP requests graceful teardown, DDP MUST
indicate to the LLP that it SHOULD perform a graceful close of the
other half of the LLP Stream.

If the Local Peer LLP supports a half-closed LLP Stream, on the
receipt of a ULP graceful half-closed teardown request of the DDP
Stream, DDP SHOULD keep data reception enabled on the other half of
the LLP Stream.

6.2.2. DDP Abortive Teardown

As previously mentioned, DDP does not independently terminate a DDP
Stream. Thus, any of the following fatal errors on a DDP Stream MUST
cause DDP to indicate to the ULP that a fatal error has occurred:

    * Underlying LLP Connection or LLP Stream is lost.

    * Underlying LLP reports a fatal error.

    * DDP Header has one or more invalid fields.

If the LLP indicates to the ULP that a fatal error has occurred, the
DDP layer SHOULD report the error to the ULP (see Section 7.2, DDP
Error Numbers) and complete all outstanding ULP requests with an
error. If the underlying LLP Stream is still intact, DDP SHOULD
continue to allow the ULP to transfer additional DDP Messages on the
outgoing half connection after the fatal error was indicated to the
ULP. This enables the ULP to transfer an error syndrome to the
Remote Peer. After indicating to the ULP a fatal error has occurred,
the DDP Stream MUST NOT be terminated until the Local Peer ULP
indicates to the DDP layer that the DDP Stream should be abortively
torn down.

7. Error Semantics

   All LLP errors reported to DDP SHOULD be passed up to the ULP.

7.1. Errors Detected at the Data Sink

   For non-zero-length Untagged DDP Segments, the DDP Segment MUST be
   validated before Placement by verifying:

   1. The QN is valid for this stream.

   2. The QN and MSN have an associated buffer that allows Placement of
       the payload.

       Implementers' note: DDP implementations SHOULD consider lack of
       an associated buffer as a system fault. DDP implementations MAY
       try to recover from the system fault using LLP means in a ULP-
       transparent way. DDP implementations SHOULD NOT permit system
       faults to occur repeatedly or frequently. If there is not an
       associated buffer, DDP implementations MAY choose to disable the
       stream for the reception and report an error to the ULP at the
       Data Sink.

   3. The MO falls in the range of legal offsets associated with the
       Untagged Buffer.

   4. The sum of the DDP Segment payload length and the MO falls in the
       range of legal offsets associated with the Untagged Buffer.

   5. The Message Sequence Number falls in the range of legal Message
       Sequence Numbers, for the queue defined by the QN. The legal
       range is defined as being between the MSN value assigned to the
       first available buffer for a specific QN and the MSN value
       assigned to the last available buffer for a specific QN.

       Implementers' note: for a typical Queue Number, the lower limit
       of the Message Sequence Number is defined by whatever DDP
       Messages have already been completed. The upper limit is defined
       by however many message buffers are currently available for that
       queue. Both numbers change dynamically as new DDP Messages are
       received and completed, and new buffers are added. It is up to
       the ULP to ensure that sufficient buffers are available to handle
       the incoming DDP Segments.

   For non-zero-length Tagged DDP Segments, the segment MUST be
   validated before Placement by verifying:

   1. The STag is valid for this stream.

    2. The STag has an associated buffer that allows Placement of the
        payload.

    3. The TO falls in the range of legal offsets registered for the
        STag.

    4. The sum of the DDP Segment payload length and the TO falls in the
        range of legal offsets registered for the STag.

    5. A 64-bit unsigned sum of the DDP Segment payload length and the
        TO does not wrap.

    If the DDP layer detects any of the receive errors listed in this
    section, it MUST cease placing the remainder of the DDP Segment and
    report the error(s) to the ULP. The DDP layer SHOULD include in the
    error report the DDP Header, the type of error, and the length of the
    DDP segment, if available. DDP MUST silently drop any subsequent
    incoming DDP Segments. Since each of these errors represents a
    failure of the sending ULP or protocol, DDP SHOULD enable the ULP to
    send one additional DDP Message before terminating the DDP Stream.

7.2. DDP Error Numbers

    The following error numbers MUST be used when reporting errors to the
    ULP. They correspond to the checks enumerated in section 7.1. Each
    error is subdivided into a 4-bit Error Type and an 8-bit Error Code.

    Error   Error
    Type   Code   Description

    0x0   0x00   Local Catastrophic

    0x1   Tagged Buffer Error
             0x00   Invalid STag
             0x01   Base or bounds violation
             0x02   STag not associated with DDP Stream
             0x03   TO wrap
             0x04   Invalid DDP version

    0x2   Untagged Buffer Error
             0x01   Invalid QN
             0x02   Invalid MSN - no buffer available
             0x03   Invalid MSN - MSN range is not valid
             0x04   Invalid MO
             0x05   DDP Message too long for available buffer
             0x06   Invalid DDP version

    0x3   Rsvd   Reserved for the use by the LLP

8. Security Considerations

    This section discusses both protocol-specific considerations and the
    implications of using DDP with existing security mechanisms. The
    security requirements for the DDP implementation are provided at the
    end of the section. A more detailed analysis of the security issues
    around the implementation and the use of the DDP can be found in
    [RDMASEC].

    The IPsec requirements for RDDP are based on the version of IPsec
    specified in RFC 2401 [IPSEC] and related RFCs, as profiled by RFC
    3723 [RFC3723], despite the existence of a newer version of IPsec
    specified in RFC 4301 [RFC4301] and related RFCs [RFC4303],
    [RFC4306]. One of the important early applications of the RDDP
    protocols is their use with iSCSI [iSER]; RDDP's IPsec requirements
    follow those of IPsec in order to facilitate that usage by allowing a
    common profile of IPsec to be used with iSCSI and the RDDP protocols.
    In the future, RFC 3723 may be updated to the newer version of IPsec;
    the IPsec security requirements of any such update should apply
    uniformly to iSCSI and the RDDP protocols.

8.1. Protocol-Specific Security Considerations

    The vulnerabilities of DDP to active third-party interference are no
    greater than any other protocol running over transport protocols such
    as TCP and SCTP over IP. A third party, by injecting spoofed packets
    into the network that are Delivered to a DDP Data Sink, could launch
    a variety of attacks that exploit DDP-specific behavior. Since DDP
    directly or indirectly exposes memory addresses on the wire, the
    Placement information carried in each DDP Segment must be validated,
    including invalid STag and octet-level granularity base and bounds
    check, before any data is Placed. For example, a third-party
    adversary could inject random packets that appear to be valid DDP
    Segments and corrupt the memory on a DDP Data Sink. Since DDP is IP
    transport protocol independent, communication security mechanisms
    such as IPsec [IPSEC] may be used to prevent such attacks.

8.2. Association of an STag and a DDP Stream

    There are several mechanisms for associating an STag and a DDP
    Stream. Two required mechanisms for this association are a
    Protection Domain (PD) association and a DDP Stream association.

    Under the Protection Domain (PD) association, a unique Protection
    Domain Identifier (PD ID) is created and used locally to associate an
    STag with a set of DDP Streams. Under this mechanism, the use of the
    STag is only permitted on the DDP Streams that have the same PD ID as
    the STag. For an incoming DDP Segment of a Tagged DDP Message on a

DDP Stream, if the PD ID of the DDP Stream is not the same as the PD
ID of the STag targeted by the Tagged DDP Message, then the DDP
Segment is not Placed, and the DDP layer MUST surface a local error
to the ULP. Note that the PD ID is locally defined and cannot be
directly manipulated by the Remote Peer.

Under the DDP Stream association, a DDP Stream is identified locally
by a unique DDP Stream identifier (ID). An STag is associated with a
DDP Stream by using a DDP Stream ID. In this case, for an incoming
DDP Segment of a Tagged DDP Message on a DDP Stream, if the DDP
Stream ID of the DDP Stream is not the same as the DDP Stream ID of
the STag targeted by the Tagged DDP Message, then the DDP Segment is
not Placed and the DDP layer MUST surface a local error to the ULP.
Note that the DDP Stream ID is locally defined and cannot be directly
manipulated by the Remote Peer.

A ULP SHOULD associate an STag with at least one DDP Stream. DDP
MUST support Protection Domain association and DDP Stream association
mechanisms for associating an STag and a DDP Stream.

8.3. Security Requirements

[RDMASEC] defines the security model and general assumptions for
RDMAP/DDP. This subsection provides the security requirements for
the DDP implementation. For more details on the type of attacks,
type of attackers, trust models, and resource sharing for the DDP
implementation, the reader is referred to [RDMASEC].

DDP has several mechanisms that deal with a number of attacks. These
attacks include, but are not limited to:

1. Connection to/from an unauthorized or unauthenticated endpoint.
2. Hijacking of a DDP Stream.
3. Attempts to read or write from unauthorized memory regions.
4. Injection of RDMA Messages within a stream on a multi-user
    operating system by another application.

DDP relies on the LLP to establish the LLP Stream over which DDP
Messages will be carried. DDP itself does nothing to authenticate
the validity of the LLP Stream of either of the endpoints. It is the
responsibility of the ULP to validate the LLP Stream. This is highly
desirable due to the nature of DDP.

Hijacking of an DDP Stream would require that the underlying LLP
Stream is hijacked. This would require knowledge of Advertised
Buffers in order to directly Place data into a user buffer.
Therefore, this is constrained by the same techniques mentioned to
guard against attempts to read or write from unauthorized memory
regions.

DDP does not require a node to open its buffers to arbitrary attacks
over the DDP Stream. It may access ULP memory only to the extent
that the ULP has enabled and authorized it to do so. The STag access
control model is defined in [RDMASEC]. Specific security operations
include:

1. STags are only valid over the exact byte range established by the
    ULP. DDP MUST provide a mechanism for the ULP to establish and
    revoke the TO range associated with the ULP Buffer referenced by
    the STag.
2. STags are only valid for the duration established by the ULP.
    The ULP may revoke them at any time, in accordance with its own
    upper layer protocol requirements. DDP MUST provide a mechanism
    for the ULP to establish and revoke STag validity.
3. DDP MUST provide a mechanism for the ULP to communicate the
    association between a STag and a specific DDP Stream.
4. A ULP may only expose memory to remote access to the extent that
    it already had access to that memory itself.
5. If an STag is not valid on a DDP Stream, DDP MUST pass the
    invalid access attempt to the ULP. The ULP may provide a
    mechanism for terminating the DDP Stream.

Further, DDP provides a mechanism that directly Places incoming
payloads in user-mode ULP Buffers. This avoids the risks of prior
solutions that relied upon exposing system buffers for incoming
payloads.

For the DDP implementation, two components MUST be provided: an
RDMA-enabled NIC (RNIC) and a Privileged Resource Manager (PRM).

8.3.1. RNIC Requirements

The RNIC MUST implement the DDP wire Protocol and perform the
security semantics described below.

1. An RNIC MUST ensure that a specific DDP Stream in a specific
    Protection Domain cannot access an STag in a different Protection
    Domain.

2. An RNIC MUST ensure that if an STag is limited in scope to a
    single DDP Stream, no other DDP Stream can use the STag.

3. An RNIC MUST ensure that a Remote Peer is not able to access
    memory outside the buffer specified when the STag was enabled for
    remote access.

4. An RNIC MUST provide a mechanism for the ULP to establish and
    revoke the association of a ULP Buffer to an STag and TO range.

5. An RNIC MUST provide a mechanism for the ULP to establish and
    revoke read, write, or read and write access to the ULP Buffer
    referenced by an STag.

6. An RNIC MUST ensure that the network interface can no longer
    modify an Advertised Buffer after the ULP revokes remote access
    rights for an STag.

7. An RNIC MUST NOT enable firmware to be loaded on the RNIC
    directly from an untrusted Local Peer or Remote Peer, unless the
    Peer is properly authenticated (by a mechanism outside the scope
    of this specification. The mechanism presumably entails
    authenticating that the remote ULP has the right to perform the
    update), and the update is done via a secure protocol, such as
    IPsec.

8.3.2. Privileged Resources Manager Requirement

   The PRM MUST implement the security semantics described below.

1. All Non-Privileged ULP interactions with the RNIC Engine that
    could affect other ULPs MUST be done using the Privileged
    Resource Manager as a proxy.

2. All ULP resource allocation requests for scarce resources MUST
    also be done using a Privileged Resource Manager.

3. The Privileged Resource Manager MUST NOT assume different ULPs
    share Partial Mutual Trust unless there is a mechanism to ensure
    that the ULPs do indeed share partial mutual trust.

4. If Non-Privileged ULPs are supported, the Privileged Resource
    Manager MUST verify that the Non-Privileged ULP has the right to
    access a specific Data Buffer before allowing an STag for which
    the ULP has access rights to be associated with a specific Data
    Buffer.

5. The Privileged Resource Manager SHOULD prevent a Local Peer from
    allocating more than its fair share of resources. If an RNIC
    provides the ability to share receive buffers across multiple DDP
    Streams, the combination of the RNIC and the Privileged Resource

          Manager MUST be able to detect if the Remote Peer is attempting
          to consume more than its fair share of resources so that the
          Local Peer can apply countermeasures to detect and prevent the
          attack.

8.4. Security Services for DDP

     DDP uses IP-based network services; therefore, all exchanged DDP
     Segments are vulnerable to spoofing, tampering and information
     disclosure attacks. If a DDP Stream may be subject to impersonation
     attacks, or stream hijacking attacks, it is highly RECOMMENDED that
     the DDP Stream be authenticated, integrity protected, and protected
     from replay attacks. It MAY use confidentiality protection to
     protect from eavesdropping.

8.4.1. Available Security Services

     IPsec can be used to protect against the packet injection attacks
     outlined above. Because IPsec is designed to secure arbitrary IP
     packet streams, including streams where packets are lost, DDP can run
     on top of IPsec without any change.

     DDP security may also profit from SSL or TLS security services
     provided for TCP or SCTP based ULPs [TLS] as well as from DTLS [DTLS]
     security services provided beneath the transport protocol. See
     [RDMASEC] for further discussion of these approaches and the
     rationale for selection of IPsec security services for the RDDP
     protocols.

8.4.2. Requirements for IPsec Services for DDP

     IPsec packets are processed (e.g., integrity checked and possibly
     decrypted) in the order they are received, and a DDP Data Sink will
     process the decrypted DDP Segments contained in these packets in the
     same manner as DDP Segments contained in unsecured IP packets.

     The IP Storage working group has defined the normative IPsec
     requirements for IP Storage [RFC3723]. Portions of this
     specification are applicable to the DDP. In particular, a compliant
     implementation of IPsec services MUST meet the requirements as
     outlined in Section 2.3 of [RFC3723]. Without replicating the
     detailed discussion in [RFC3723], this includes the following
     requirements:

     1. The implementation MUST support IPsec ESP [RFC2406], as well as
         the replay protection mechanisms of IPsec. When ESP is utilized,
         per-packet data origin authentication, integrity, and replay
         protection MUST be used.

   2. It MUST support ESP in tunnel mode and MAY implement ESP in
       transport mode.

   3. It MUST support IKE [RFC2409] for peer authentication,
       negotiation of security associations, and key management, using
       the IPsec DOI [RFC2407].

   4. It MUST NOT interpret the receipt of an IKE delete message as a
       reason for tearing down the DDP stream. Since IPsec acceleration
       hardware may only be able to handle a limited number of active
       IPsec Security Associations (SAs), idle SAs may be dynamically
       brought down and a new SA be brought up again, if activity
       resumes.

   5. It MUST support peer authentication using a pre-shared key, and
       MAY support certificate-based peer authentication using digital
       signatures. Peer authentication using the public key encryption
       methods [RFC2409] SHOULD NOT be used.

   6. It MUST support IKE Main Mode and SHOULD support Aggressive Mode.
       IKE Main Mode with pre-shared key authentication SHOULD NOT be
       used when either of the peers uses a dynamically assigned IP
       address.

   7. Access to locally stored secret information (pre-shared or
       private key for digital signing) must be suitably restricted,
       since compromise of the secret information nullifies the security
       properties of the IKE/IPsec protocols.

   8. It MUST follow the guidelines of Section 2.3.4 of [RFC3723] on
       the setting of IKE parameters to achieve a high level of
       interoperability without requiring extensive configuration.

   Furthermore, implementation and deployment of the IPsec services for
   DDP should follow the Security Considerations outlined in Section 5
   of [RFC3723].

9. IANA Considerations

   This document requests no direct action from IANA. The following
   consideration is listed here as commentary.

   If DDP were enabled a priori for a ULP by connecting to a well-known
   port, this well-known port would be registered for the DDP with IANA.
   The registration of the well-known port would be the responsibility
   of the ULP specification.

10. References

10.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
             Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC2406] Kent, S. and Atkinson, R., "IP Encapsulating Security
             Payload (ESP)", RFC 2406, November 1998.

   [RFC2407] Piper, D., "The Internet IP Security Domain of
             Interpretation of ISAKMP", RFC 2407, November 1998.

   [RFC2409] Harkins, D. and Carrel, D., "The Internet Key Exchange
             (IKE)", RFC 2409, November 1998.

   [RFC3723] Aboba, B., Tseng, J., Walker, J., Rangan, V., Travostino,
             F., "Securing Block Storage Protocols over IP", RFC 3723,
             April 2004.

   [IPSEC]   Kent, S. and R. Atkinson, "Security Architecture for the
             Internet Protocol", RFC 2401, November 1998.

   [MPA]   Culley, P., Elzur, U., Recio, R., Bailey, S., and J.
             Carrier, "Marker PDU Aligned Framing for TCP
             Specification", RFC 5044, October 2007.

   [RDMAP]   Recio, R., Culley, P., Garcia, D., and J. Hilland, "A
             Remote Direct Memory Access Protocol Specification", RFC
             5040, October 2007.

   [RDMASEC] Pinkerton, J. and E. Deleganes, "Direct Data Placement
             Protocol (DDP) / Remote Direct Memory Access Protocol
             (RDMAP) Security", RFC 5042, October 2007.

   [SCTP]   Stewart, R., Ed., "Stream Control Transmission Protocol",
             RFC 4960, September 2007.

   [SCTPDDP] Bestler, C. and R. Stewart, "Stream Control Transmission
             Protocol (SCTP) Direct Data Placement (DDP) Adaptation",
             RFC 5043, October 2007.

   [TCP]   Postel, J., "Transmission Control Protocol", STD 7, RFC
             793, September 1981.

10.2. Informative References

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
             Internet Protocol", RFC 4301, December 2005.

   [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC
             4303, December 2005.

   [RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC
             4306, December 2005.

   [DTLS]   Rescorla, E. and N. Modadugu, "Datagram Transport Layer
             Security", RFC 4347, April 2006.

   [TLS]   Dierks, T. and E. Rescorla, "The Transport Layer Security
             (TLS) Protocol Version 1.1", RFC 4346, April 2006.

   [iSER]   Ko, M., Chadalapaka, M., Hufferd, J., Elzur, U., Shah, H.,
             and P. Thaler, "Internet Small Computer System Interface
             (iSCSI) Extensions for Remote Direct Memory Access (RDMA)",
             RFC 5046, October 2007.

Appendix A. Receive Window Sizing

   This appendix provides guidance to LLP implementers.

   Reliable, sequenced, LLPs include a mechanism to Advertise the amount
   of receive buffer space a sender may consume. This is generally
   called a "receive window".

   DDP allows data to be transferred directly to predefined buffers at
   the Data Sink. Accordingly, the LLP receive window size need not be
   affected by the reception of a DDP Segment, if that segment is placed
   before additional segments arrive.

   The LLP implementation SHOULD maintain an Advertised receive window
   large enough to enable a reasonable number of segments to be
   outstanding at one time. The amount to Advertise depends on the
   desired data rate, and the expected or actual round-trip delay
   between endpoints.

   The amount of actual buffers maintained to "back up" the receive
   window is left up to the implementation. This amount will depend on
   the rate that DDP Segments can be retired; there may be some cases
   where segment processing cannot keep up with the incoming packet
   rate. If this occurs, one reasonable way to slow the incoming packet
   rate is to reduce the receive window.

   Note that the LLP should take care to comply with the applicable
   RFCs; for instance, for TCP, receivers are highly discouraged from
   "shrinking" the receive window (reducing the right edge of the window
   after it has been Advertised).

Appendix B. Contributors

   Many thanks to the following individuals for their contributions.

   John Carrier
   Cray Inc.
   411 First Avenue S, Suite 600
   Seattle, WA 98104-2860
   Phone: 206-701-2090
   EMail: carrier@cray.com

   Hari Ghadia
   Gen10 Technology, Inc.
   1501 W Shady Grove Road
   Grand Prairie, TX 75050
   Phone: (972) 301 3630
   EMail: hghadia@gen10technology.com

Caitlin Bestler
Broadcom Corporation
16215 Alton Parkway
Irvine, CA 92619-7013 USA
Phone: +1 (949) 926-6383
EMail: caitlinb@Broadcom.com

Uri Elzur
Broadcom Corporation
5300 California Avenue
Irvine, CA 92617, USA
Phone: 949.926.6432
EMail: uri@broadcom.com

Mike Penna
Broadcom Corporation
16215 Alton Parkway
Irvine, CA 92619-7013 USA
Phone: +1 (949) 926-7149
EMail: MPenna@Broadcom.com

Patricia Thaler
Broadcom Corporation
16215 Alton Parkway
Irvine, CA 92619-7013 USA
Phone: +1 (949) 926-8635
EMail: pthaler@broadcom.com

Ted Compton
EMC Corporation
Research Triangle Park, NC 27709 USA
Phone: +1 (919) 248-6075
EMail: compton_ted@emc.com

Jim Wendt
Hewlett-Packard Company
8000 Foothills Boulevard
Roseville, CA 95747-5668 USA
Phone: +1 (916) 785-5198
EMail: jim_wendt@hp.com

Mike Krause
Hewlett-Packard Company, 43LN
19410 Homestead Road
Cupertino, CA 95014 USA
Phone: +1 (408) 447-3191
EMail: krause@cup.hp.com

Dave Minturn
Intel Corporation
MS JF1-210
5200 North East Elam Young Parkway
Hillsboro, OR 97124 USA
Phone: +1 (503) 712-4106
EMail: dave.b.minturn@intel.com

Howard C. Herbert
Intel Corporation
MS CH7-404
5000 West Chandler Blvd.
Chandler, AZ 85226 USA
Phone: +1 (480) 554-3116
EMail: howard.c.herbert@intel.com

Tom Talpey
Network Appliance
1601 Trapelo Road #16
Waltham, MA 02451 USA
Phone: +1 (781) 768-5329
EMail: thomas.talpey@netapp.com

Dwight Barron
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070-2698 USA
Phone: +1 (281) 514-2769
EMail: Dwight.Barron@Hp.com

Dave Garcia
24100 Hutchinson Rd.
Los Gatos, CA 95033 USA
Phone: +1 (831) 247-4464
Email: Dave.Garcia@StanfordAlumni.org

Jeff Hilland
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070-2698 USA
Phone: +1 (281) 514-9489
EMail: jeff.hilland@hp.com

Barry Reinhold
Lamprey Networks
Durham, NH 03824 USA
Phone: +1 (603) 868-8411
EMail: bbr@LampreyNetworks.com

Authors' Addresses

   Hemal Shah
   Broadcom Corporation
   5300 California Avenue
   Irvine, CA 92617 USA
   Phone: +1 (949) 926-6941
   EMail: hemal@broadcom.com

   James Pinkerton
   Microsoft Corporation
   One Microsoft Way
   Redmond, WA 98052 USA
   Phone: +1 (425) 705-5442
   EMail: jpink@microsoft.com

   Renato Recio
   IBM Corporation
   11501 Burnett Road
   Austin, TX 78758 USA
   Phone: +1 (512) 838-1365
   EMail: recio@us.ibm.com

   Paul R. Culley
   Hewlett-Packard Company
   20555 SH 249
   Houston, TX 77070-2698 USA
   Phone: +1 (281) 514-5543
   EMail: paul.culley@hp.com