INTERNET-DRAFT
draft-ietf-rddp-ddp-00.txt

Hemal Shah
Intel Corporation
James Pinkerton
Microsoft Corporation
Renato Recio
IBM Corporation
Paul Culley
Hewlett-Packard Company

Expires: August, 2003

Direct Data Placement over Reliable Transports

1 Status of this Memo

This document is an Internet-Draft and is subject to all provisions of Section 10 of RFC2026.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

2 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

1 Status of this Memo....1
2 Abstract....1
3 Introduction....4
3.1 Architectural Goals....4
3.2 Protocol Overview....5
3.3 DDP Layering....7
4 Glossary....9
4.1 General....9
4.2 LLP....10
4.3 Direct Data Placement (DDP)....10
5 Reliable Delivery LLP Requirements....13
6 Header Format....15
6.1 DDP Control Field....15
6.2 DDP Tagged Buffer Model Header....16
6.3 DDP Untagged Buffer Model Header....17
6.4 DDP Segment Format....19
7 Data Transfer....20
7.1 DDP Tagged or Untagged Buffer Models....20
7.1.1 Tagged Buffer Model....20
7.1.2 Untagged Buffer Model....20
7.2 Segmentation and Reassembly of a DDP Message....21
7.3 Ordering Among DDP Messages....22
7.4 DDP Message Completion & Delivery....23
8 DDP Stream Setup & Teardown....25
8.1 DDP Stream Setup....25
8.2 DDP Stream Teardown....25
8.2.1 DDP Graceful Teardown....25
8.2.2 DDP Abortive Teardown....26
9 Error Semantics....27
9.1 Errors detected at the Data Sink....27
9.2 DDP Error Numbers....28
10 Security Considerations....29
10.1 Protocol-specific Security Considerations....29
10.2 Using IPSec with DDP....29
10.3 Association of an STag and a DDP Stream....29
10.4 Other Security Considerations....30
11 IANA Considerations....32
12 References....33
12.1 Normative References....33
12.2 Informative References....33
13 Appendix....34
13.1 Receive Window sizing....34
14 Author's Addresses....35
15 Acknowledgments....36
16 Full Copyright Statement....39

Table of Figures

Figure 1 DDP Layering.....7
Figure 2 MPA, DDP, and RDMAP Header Alignment.....8
Figure 3 DDP Control Field.....15
Figure 4 Tagged Buffer DDP Header.....16
Figure 5 Untagged Buffer DDP Header.....18
Figure 6 DDP Segment Format.....19

3 Introduction

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 also enable the network protocol to consume substantially fewer CPU cycles than if the CPU was used to move the data, and removes 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.

3.1 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 which 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 LLP supports multiple LLP streams within a 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.

3.2 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 because if a DDP Message arrives for an Untagged Buffer without an associated receive ULP buffer, the DDP Message is dropped, the DDP Stream is disabled for reception, and an error is reported to the ULP at the Data Sink.

* 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 arbitrary size 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) or multiple LLP streams of data per connection (e.g. 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.

3.3 DDP Layering

DDP is intended to be LLP independent, subject to the requirements defined in section 5. 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 [MPA], [TCP], and [SCTP].

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

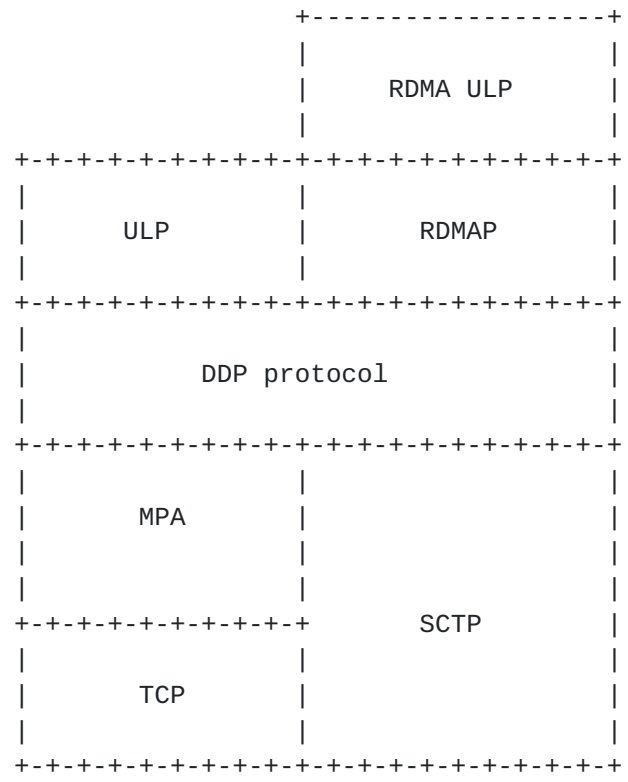


Figure 1 DDP Layering

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.):

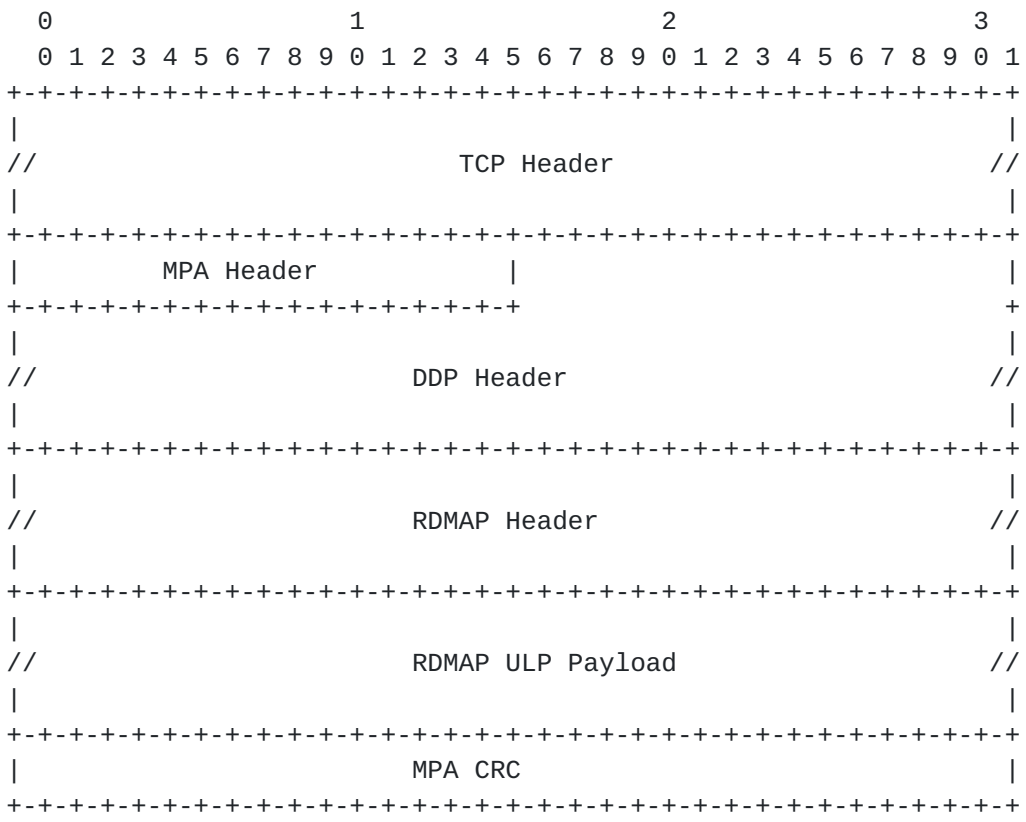


Figure 2 MPA, DDP, and RDMAP Header Alignment

4 Glossary

4.1 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.

iWARP - A suite of wire protocols comprised of RDMAP [RDMAP], DDP [DDP], and 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 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 RNICs installed in a host computer.

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.

ULP - Upper Layer Protocol. The protocol layer above the protocol

layer currently being referenced. The ULP for RDMA/DDP is expected to be an 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).

4.2 LLP

LLP - Lower Layer Protocol. The protocol layer beneath the protocol layer currently being referenced. For example, for DDP the LLP is SCTP, 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), a LLP Stream corresponds to one transport-level stream.

MULPDU - Maximum ULPDU. 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.

4.3 Direct Data Placement (DDP)

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

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".

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) û 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) û 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) û 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, Target 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.

Target 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 - is the total length 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.

5 Reliable Delivery LLP Requirements

1. LLPs MUST expose MULPDU & 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 which were posted multiple MULPDU updates previously, 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 require 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 a 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.

6 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 7.1 for a description of the two models.

6.1 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.

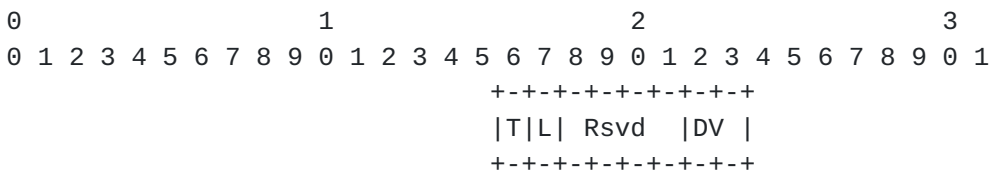


Figure 3 DDP Control Field

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:

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

. 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.

6.2 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 6.1. (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.)



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.

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.

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.

6.3 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 6.1. (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
```

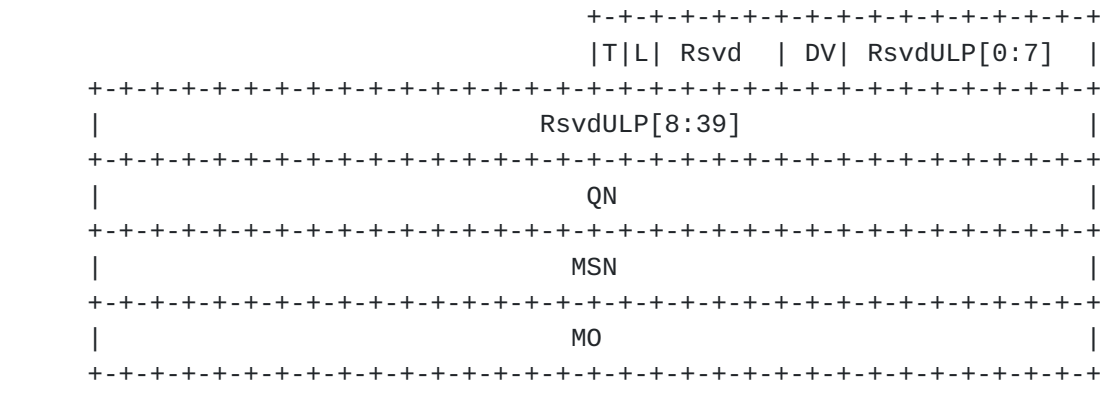


Figure 5 Untagged Buffer DDP Header

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.

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.

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.

6.4 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:

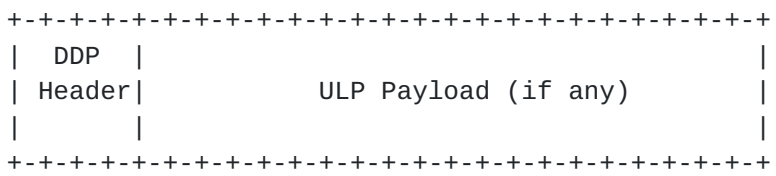


Figure 6 DDP Segment Format

7 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.

7.1 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.

7.1.1 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.

7.1.2 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 which 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.

7.2 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 enables 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 which 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 Tagged 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.

7.3 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.

7.4 DDP Message Completion & 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.

8 DDP Stream Setup & Teardown

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

8.1 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.

8.2 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 teardown 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.

8.2.1 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 & 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 a 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-close teardown request of the DDP Stream, DDP SHOULD keep data reception enabled on the other half of the LLP stream.

8.2.2 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 catastrophic 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 9.2, 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 torndown.

9 Error Semantics

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

9.1 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.

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. For DDP Messages using Untagged Buffer model, 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.

9.2 DDP Error Numbers

The following error numbers MUST be used when reporting receive errors to the ULP. They correspond to the checks enumerated in section 9.1. Each error is subdivided into a 4-bit Error Type and an 8 bit Error Code.

Error Type	Error 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

10 Security Considerations

This section discusses both protocol-specific considerations and the implications of using DDP with existing security mechanisms.

10.1 Protocol-specific Security Considerations

The vulnerabilities of DDP to active third-party interference are no greater than any other protocol running over TCP. 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] or TLS [TLS] may be used to prevent such attacks.

10.2 Using IPSec with DDP

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. 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.

10.3 Association of an STag and a DDP Stream

There are several mechanisms for associating an STag and a DDP Stream. Two reasonable 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 and a DDP Stream. DDP MUST support Protection Domain association and DDP Stream association mechanisms for associating an STag and a DDP Stream.

10.4 Other Security Considerations

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 and is therefore 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 by a (forthcoming) document. 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.

11 IANA Considerations

If DDP was 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.

12 References

12.1 Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

[MPA] P. Culley et al., "Markers with PDU Alignment", RDMA Consortium Draft Specification draft-cully-iwarp-mpa-01.doc, October 2002

[RDMAP] R. Recio et al., "RDMA Protocol Specification", RDMA Consortium Draft Specification draft-recio-iwarp-01, October

2002

[SCTP] R. Stewart et al., "Stream Control Transmission Protocol", RFC 2960, October 2000.

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

12.2 Informative References

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, November 1998.

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

13 Appendix

13.1 Receive Window sizing

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).

14 Author's Addresses

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

Hemal Shah
Intel Corporation
MS PTL1
1501 South Mopac Expressway, #400
Austin, TX 78746 USA
Phone: +1 (512) 732-3963
Email: hemal.shah@intel.com

15 Acknowledgments

John Carrier
Adaptec, Inc.
691 S. Milpitas Blvd.
Milpitas, CA 95035 USA
Phone: +1 (360) 378-8526
Email: john_carrier@adaptec.com

Hari Ghadia
Adaptec, Inc.
691 S. Milpitas Blvd.,
Milpitas, CA 95035 USA
Phone: +1 (408) 957-5608
Email: hari_ghadia@adaptec.com

Patricia Thaler
Agilent Technologies, Inc.
1101 Creekside Ridge Drive, #100

M/S-RG10
Roseville, CA 95678
Phone: +1-916-788-5662
email: pat_thaler@agilent.com

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

Uri Elzur
Broadcom Corporation
16215 Alton Parkway
Irvine, California 92619-7013 USA
Phone: +1 (949) 585-6432
Email: Uri@Broadcom.com

Ted Compton
EMC Corporation
Research Triangle Park, NC 27709, USA
Phone: 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
375 Totten Pond Road
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
Hewlett-Packard Company
19333 Vallco Parkway
Cupertino, Ca. 95014 USA
Phone: +1 (408) 285-6116
Email: dave.garcia@hp.com

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

16 Full Copyright Statement

This document and the information contained herein is provided on an "AS IS" basis and ADAPTEC INC., AGILENT TECHNOLOGIES INC., BROADCOM CORPORATION, CISCO SYSTEMS INC., EMC CORPORATION, HEWLETT-PACKARD COMPANY, INTERNATIONAL BUSINESS MACHINES CORPORATION, INTEL CORPORATION, MICROSOFT CORPORATION, NETWORK APPLIANCE INC., THE INTERNET SOCIETY, AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright (c) 2002 ADAPTEC INC., BROADCOM CORPORATION, CISCO SYSTEMS INC., EMC CORPORATION, HEWLETT-PACKARD COMPANY, INTERNATIONAL BUSINESS MACHINES CORPORATION, INTEL CORPORATION, MICROSOFT CORPORATION, NETWORK APPLIANCE INC., All Rights Reserved.