draft-culley-iwarp-mpa-v1.0

P. Culley
   Hewlett-Packard Company
U. Elzur
   Broadcom Corporation
R. Recio
   IBM Corpration
S. Bailey
   Sandburst Corporation
et. al.

October 25, 2002

Marker PDU Aligned Framing for TCP Specification (Version 1.0)

1 Status of this Memo

This document is a Release Specification of the RDMA Consortium.
Copies of this document and associated errata may be found at
http://www.rdmaconsortium.org.

2 Abstract

A framing protocol is defined for TCP that is fully compliant with
applicable TCP RFCs and fully interoperable with existing TCP
implementations. The framing mechanism is designed to work as a
"shim" between TCP and higher-level protocols, preserving the
reliable, in-order delivery of TCP while adding the preservation of
higher-level protocol record boundaries.

Table of Contents

Table of Figures

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

3   Introduction

This section discusses the reason for creating MPA on TCP and a
general overview of the protocol. Later sections show the MPA
headers (see section 6 on page 10), and detailed protocol
requirements and characteristics (see section 7 on page 12), as well
as Connection Semantics (section 8 on page 19), Error Semantics
(section 9 on page 20), and Security Considerations (section 10 on
page 21).

3.1 Motivation

A generalized framing mechanism for the TCP transport protocol [TCP]
is desirable to some Upper Layer Protocols (ULP). One ULP that can
benefit from the framing mechanism is Direct Data Placement (DDP).
The ability to locate the Upper Layer Protocol Data Unit (ULPDU)
boundary is useful to a hardware network adapter that uses DDP to
directly place the data in the application buffer based on the
control information carried in the ULPDU header. This may be done
without requiring that the packets arrive in order. One potential
benefit of this capability is the avoidance of the memory copy
overhead. Another potential benefit is the smaller memory
requirement for handling out of order packets and dropped packets.

MPA is intended for ULPs that are specifically designed to utilize
"records" (ULPDUSs) rather than a stream of octets.

Many approaches have been proposed for the generalized framing
mechanism. Some are probabilistic in nature and others are
deterministic. A probabilistic approach is characterized by a
detectable value embedded in the byte stream. It is probabilistic
because under some conditions the receiver may incorrectly interpret
application data as the detectable value. Under these conditions,
the protocol may fail with unacceptable frequency. A deterministic
approach is characterized by embedded controls at known locations in
the byte stream. Because the receiver can guarantee it will only
examine the data stream at locations that are known to contain the
embedded control, the protocol can never misinterpret application
data as being embedded control data. For unambiguous handling of an
out of order packet, the deterministic approach is preferred.

The MPA protocol provides a generalized framing mechanism for TCP
using the deterministic approach. It allows the location of the
ULPDU to be determined in the TCP stream even if the TCP segments
arrive out of order.

3.2 Protocol Overview

   MPA is described as a extra layer above TCP and below the ULP. The
   end-to-end data flow is:

   1. The ULP negotiates the use of MPA at both ends of a connection.

   2. The ULP hands data records (ULPDUs) to MPA at the sender.

   3. MPA creates a Framed Protocol Data Unit (FPDU) by pre-pending a
       header, inserting markers, and appending a CRC after the ULPDU
       and PAD (if any). MPA delivers the FPDU to TCP.

   4. The MPA-aware TCP sender puts the FPDUs into the TCP stream. It
       segments the TCP stream in such a way that each TCP segment
       contains a single FPDU. TCP then passes each segment to the IP
       layer for transmission.

   5. The TCP receiver may be MPA-aware or may not be MPA-aware. If it
       is MPA-aware, it may separate passing the TCP payload to MPA from
       passing the TCP payload ordering information to MPA. In either
       case, RFC compliant TCP wire behavior is observed at both the
       sender and receiver.

   6. The MPA receiver locates and assembles complete FPDUs within the
       stream, verifies their integrity, and removes MPA markers,
       ULPDU_Length, PAD and CRC.

   7. MPA then provides the complete ULPDUs to the ULP. MPA may also
       separate passing MPA payload to the ULP from passing the MPA
       payload ordering information.

   The layering of PDUs with MPA is shown in Figure 1.

```
            +------------------+
            |   ULP client     |
            +------------------+  <- ULPDUs
            |      MPA         |
            +------------------+  <- FPDUs (containing ULPDUs)
            |     TCP*         |
            +------------------+  <- TCP Segments (containing FPDUs)
            |    IP etc.       |
            +------------------+

                        * TCP or MPA-aware TCP.
```
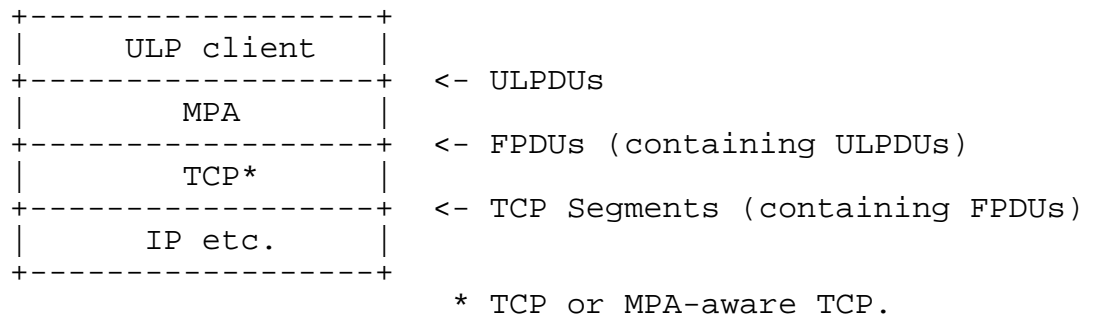
             Figure 1 ULP MPA TCP Layering

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

MPA-aware TCP is a TCP layer which potentially contains some
additional semantics as defined in this document. MPA is implemented
as a data stream ULP for TCP and is therefore RFC compliant. MPA-
aware TCP is RFC compliant.

MPA with an MPA-aware TCP allows an implementation to recover ULPDUs
that may be received out of order. This enables an implementation
with an appropriate ULP at the receiver to save a significant amount
of intermediate storage by storing the ULPDUs in the right locations
in the ULP buffers when they arrive, rather than waiting until full
ordering can be restored.

MPA implementations that support recovery of out of order ULPDUs
should also support a mechanism to indicate the ordering of ULPDUs as
the sender transmitted them and indicate when missing intermediate
segments arrive. These mechanisms allow ULPs to reestablish record
ordering and report arrival of complete groups of records.

One last area that MPA addresses is data integrity. Many users of
TCP have noted that the TCP checksum is not as strong as could be
desired [CRCTCP]. Studies have shown that the TCP checksum indicates
segments in error at a much higher rate than the underlying link
characteristics would indicate. With these higher error rates, the
chance that an error will escape detection, when using only the TCP
checksum for data integrity, becomes a concern. A stronger integrity
check can reduce the chance of data errors being missed.

MPA includes a CRC check to increase the ULPDU data integrity to the
level provided by other modern protocols, such as SCTP [SCTP].

MPA combined with an MPA-aware TCP can only ensure FPDU Alignment
with the TCP Header if the FPDU is less than or equal to TCP's EMSS.
Thus if FPDU alignment is desired by the ULP, the ULP must cooperate
with MPA to ensure FPDUs lengths do not exceed the EMSS under normal
conditions.

4 Glossary

Delivery - (Delivered, Delivers) - For MPA, Delivery is defined as
    the process of informing the ULP or consumer that a particular
    PDU is ordered for use. This is specifically different from
    "passing the PDU to the ULP", which may generally occur in any
    order, while the order of "Delivery" is strictly defined.

EMSS - Effective Maximum Segment Size. EMSS is the smaller of the
    TCP maximum segment size (MSS) as defined in RFC 793 [TCP], and
    the current path Maximum Transfer Unit (MTU) [PathMTU].

FPDU - Framing Protocol Data Unit. The unit of data created by an
    MPA sender.

FPDU Alignment - the property that a TCP segment begins with an FPDU.

PDU - protocol data unit

MPA - Marker-based ULP PDU Aligned Framing for TCP protocol.   This
    document defines the MPA protocol.

MULPDU - Maximum ULPDU. The current maximum size of the record that
    is acceptable for the ULP to pass to MPA for transmission.

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 MPA on TCP devices installed in a host
    computer.

Remote Peer - The MPA 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 MPA is expected to
    be DDP [DDP], or an OS, application, adaptation layer, or
    proprietary protocol. This document does not specify a ULP - it
    provides a set of semantics that allow a ULP to be designed to
    utilize MPA.

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

5 LLP and ULP requirements

5.1 TCP implementation Requirements to support MPA

   To provide optimum performance, a transmit side TCP implementation
   SHOULD:

   *   With an EMSS large enough to contain the FPDU, segment the
        outgoing TCP stream such that the first octet of every FPDU is
        aligned with the beginning of a TCP segment, and is entirely
        contained in the TCP segment.

   *   Report the current EMSS to the MPA transmit layer.

   When an MPA implementation supports handling out of order ULPDUs, the
   receive side TCP implementation SHOULD:

   *   Pass incoming TCP segments to MPA as soon as they have been
        received and validated, even if not received in order. The TCP
        layer MUST have committed to keeping each segment before it can
        be passed to the MPA. This means that the segment must have
        passed the TCP, IP, and lower layer data integrity validation
        (i.e., checksum), must be in the receive window, must not be a
        duplicate, must be part of the same epoch (if timestamps are used
        to verify this) and any other checks required by TCP RFCs. The
        segment MUST NOT be passed to MPA more than once unless
        explicitly requested (see Section 9).

        This is not to imply that the data must be completely ordered
        before use. An implementation may accept out of order segments,
        SACK them [RFC2018], and pass them to the ULP when the reception
        of the segments needed to fill in the gaps arrive. Such an
        implementation can "commit" to the data early on, and will not
        overwrite it even if (or when) duplicate data arrives. MPA
        expects to utilize this "commit" to allow the passing of ULPDUs
        to the ULP when they arrive, independent of ordering.

   *   Provide a mechanism to indicate the ordering of TCP segments as
        the sender transmitted them. One possible mechanism might be
        attaching the TCP sequence number to each segment.

   *   Provide a mechanism to indicate when a given TCP segment (and the
        prior TCP stream) is complete. One possible mechanism might be
        to utilize the leading (left) edge of the TCP Receive Window.

   MPA on TCP implementations that do not provide the semantics listed
   above will interoperate with those that do, but may negate many of
   the performance and resource advantages that ULPs designed for MPA
   would expect.

The LLP MUST inform MPA when the LLP connection is closed or has
begun closing the connection (e.g. received a FIN).

5.2 MPA's interactions with the ULP

ULPs require MPA to maintain ULP record boundaries from the sender to
the receiver. When using MPA on TCP to send data, the ULP provides
records (ULPDUs) to MPA. MPA will use the reliable transmission
abilities of TCP to transmit the data, and will insert appropriate
additional information into the TCP stream to allow the MPA receiver
to locate the record boundary information.

As such, MPA accepts complete records (ULPDUs) from the ULP at the
sender and returns them to the ULP at the receiver.

MPA provides information to the ULP on the current maximum size of
the record that is acceptable to send (MULPDU). The ULP SHOULD be
able to limit each record size to MULPDU. The range of MULPDU values
MUST be between 128 octets and 64768 octets, inclusive.

The sending ULP MUST NOT post a ULPDU larger than 64768 octets to
MPA. The ULP MAY post a ULPDU of any size between one and 64768
octets, however MPA is NOT REQUIRED to support a ULPDU length that is
greater than the current MULPDU.

While the maximum theoretical length supported by the MPA header
ULPDU_Length field is 65535, TCP over IP requires the IP datagram
maximum length to be 65535 octets. To enable MPA to support FPDU
Alignment, the maximum size of the ULP payload must fit within an IP
datagram. Thus the ULPDU limit of 64768 octets was derived by taking
the maximum IP datagram length, subtracting from it the maximum total
length of the sum of the IPv4 header, TCP header, IPv4 options, TCP
options, and the worst case MPA overhead, and then rounding the
result down to a 128 byte boundary.

On receive, MPA MUST pass each ULPDU with its length to the ULP when
it has been validated.

If an MPA implementation supports passing out of order ULPDUs to the
ULP, the MPA implementation SHOULD:

*   Pass each ULPDU with its length to the ULP as soon as it has been
    fully received and validated.

*   Provide a mechanism to indicate the ordering of ULPDUs as the
    sender transmitted them. One possible mechanism might be
    providing the TCP sequence number for each ULPDU.

*   Provide a mechanism to indicate when a given ULPDU (and prior
    ULPDUs) are complete. One possible mechanism might be to allow
    the ULP to see the current outgoing TCP Ack sequence number.

*   Provide an indication to the ULP that the LLP has closed or has
    begun to close the connection (e.g. received a FIN).

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
```

6   FPDU Formats

    MPA senders create FPDUs out of ULPDUs. The format of an FPDU shown
    below MUST be used for all MPA FPDUs. For purposes of clarity,
    markers are not shown in Figure 2.
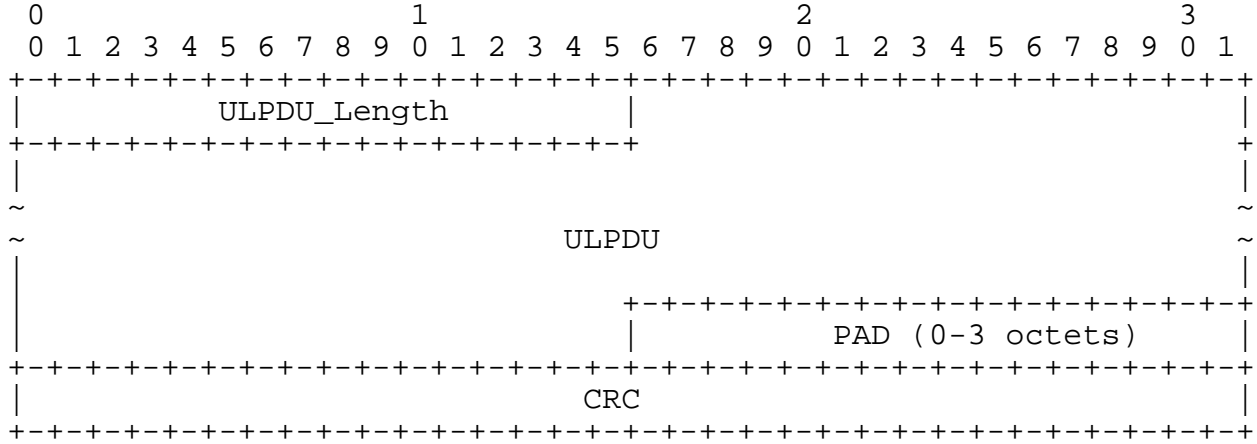
```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                 ULPDU_Length             |                    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                                  +
      |                                                               |
      ~                                                               ~
      ~                             ULPDU                             ~
      |                                                               |
      |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                               |          PAD (0-3 octets)     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                             CRC                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
                        Figure 2 FPDU Format

    ULPDU_Length: 16 bits (unsigned integer). This is the number of
    octets of the contained ULPDU. It does not include the length of the
    FPDU header itself, the pad, the CRC, or of any markers that fall
    within the ULPDU. The 16-bit ULPDU Length field is large enough to
    support the largest IP datagrams for IPv4 or IPv6.

    PAD: The PAD field trails the ULPDU and contains between zero and
    three octets of data. The pad data MUST be set to zero by the sender
    and ignored by the receiver (except for CRC checking). The length of
    the pad is set so as to make the size of the FPDU an integral
    multiple of four.

    CRC: 32 bits, this CRC is used to verify the entire contents of the
    FPDU, using CRC32c.

    The FPDU adds a minimum of 6 octets to the length of the ULPDU. In
    addition, the total length of the FPDU will include the length of any
    markers and from 0 to 3 pad bytes added to round-up the ULPDU size.

6.1 Marker Format

The format of a marker MUST be as specified in Figure 3:
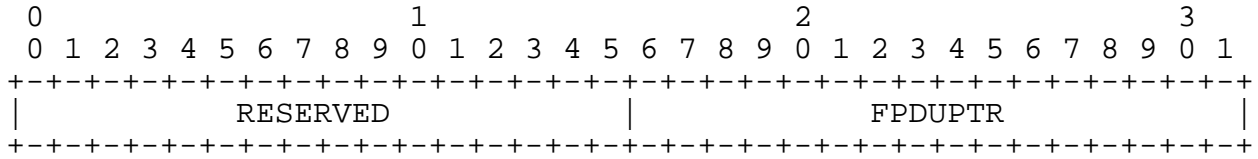
```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                RESERVED       |               FPDUPTR         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
                     Figure 3 Marker Format

RESERVED: The Reserved field MUST be set to zero on transmit and
ignored on receive (except for CRC calculation).

FPDUPTR: The FPDU Pointer is a relative pointer, 16-bits long,
interpreted as an unsigned integer, that indicates the number of
octets in the TCP stream from the beginning of the FPDU to the first
octet of the entire marker.

7 Data Transfer Semantics

   This section discusses some characteristics and behavior of the MPA
   protocol as well as implications of that protocol.

7.1 MPA Markers

   MPA senders MUST insert a marker into the data stream at a 512 octet
   periodic interval in the TCP Sequence Number Space. The marker
   contains a 16 bit unsigned integer referred to as the FPDUPTR (FPDU
   Pointer).

   If the FPDUPTR's value is non-zero, the FPDU Pointer is a 16 bit
   relative back-pointer. FPDUPTR MUST contain the number of octets in
   the TCP stream from the beginning of the current FPDU to the first
   octet of the marker, unless the marker falls between FPDUs. Thus the
   location of the first byte of the previous FPDU header can be
   determined by subtracting the value of the given marker from the
   current byte-stream sequence number (e.g. TCP sequence number) of the
   first byte of the marker. Note that this computation must take into
   account that the TCP sequence number could have wrapped between the
   marker and the header.

   An FPDUPTR value of 0x0000 is a special case - it is used when the
   marker falls exactly between FPDUs. In this case, the marker MUST be
   placed in the following FPDU and viewed as being part of that FPDU
   (e.g. for CRC calculation). Thus an FPDUPTR value of 0x0000 means
   that immediately following the marker is an FPDU header.

   Since all FPDUs are integral multiples of 4 octets, the bottom two
   bits of the FPDUPTR as calculated by the sender are zero. MPA
   reserves these bits so they MUST be treated as zero for computation
   at the receiver.

   The MPA markers MUST be inserted immediately following MPA connection
   establishment, and at every 512th octet of the TCP byte stream
   thereafter. As a result, the first marker has an FPDUPTR value of
   0x0000. If the first marker begins at byte sequence number SeqStart,
   then markers are inserted such that the first byte of the marker is
   at byte sequence number SeqNum if the remainder of (SeqNum -
   SeqStart) mod 512 is zero. Note that SeqNum can wrap.

   For example, if the TCP sequence number were used to calculate the
   insertion point of the marker, the starting TCP sequence number is
   unlikely to be zero, and 512 octet multiples are unlikely to fall on
   a modulo 512 of zero. If the MPA connection is started at TCP
   sequence number 11, then the 1st marker will begin at 11, and
   subsequent markers will begin at 523, 1035, etc.

If an FPDU is large enough to contain multiple markers, they MUST all
point to the same point in the TCP stream: the first octet of the
FPDU.

If a marker interval contains multiple FPDUs (the FPDUs are small),
the marker MUST point to the start of the FPDU containing the marker
unless the marker falls between FPDUs, in which case the marker MUST
be zero.

The following example shows an FPDU containing a marker.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          ULPDU Length (0x0010)    |                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                            +
|                                                               |
+                                                               +
|                       ULPDU (octets 0-9)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            (0x0000)               |      FPDU ptr (0x000C)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     ULPDU (octets 10-15)                      |
|                         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         |             PAD (2 octets:0,0)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           CRC                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
                Figure 4 Example FPDU Format with Marker

MPA Receivers MUST preserve ULPDU boundaries when passing data to the
ULP. MPA Receivers MUST pass the ULPDU data and the ULPDU Length to
the ULP and not the markers, headers, and CRC.

7.2 CRC Calculation

When sending an FPDU, the sender MUST include a valid CRC field. The
CRC field in the MPA FPDU, MUST be computed in the manner described
in the iSCSI Protocol [iSCSI] document for Header and Data Digests.

The fields which MUST be included in the CRC calculation when sending
an FPDU are as follows:

1) If the first octet of the FPDU is the "ULPDU Length" field, the
    CRC-32c is calculated from the first octet of the "ULPDU Length"
    header, through all ULP payload and markers (if present), to the
    last octet of the PAD (if present), inclusive. If there is a
    marker immediately following the PAD, the marker is included in
    the CRC calculation for this FPDU.

2) If the first octet of the FPDU is a marker, (i.e. the marker fell
    between FPDUs, and thus is required to be included in the second
    FPDU), the CRC-32c is calculated from the first octet of the
    marker, through the "ULPDU Length" header, through all ULP
    payload and markers (if present), to the last octet of the PAD
    (if present), inclusive.

3) After calculating the CRC-32c, the resultant value is placed into
    the CRC field at the end of the FPDU.

When an FPDU is received, the receiver MUST first perform the
following:

1) Calculate the CRC of the incoming FPDU in the same fashion as
    defined above.

2) Verify that the calculated CRC-32c value is the same as the
    received CRC-32c value found in the FPDU CRC field. If not, the
    receiver MUST treat the FPDU as an invalid FPDU.

The procedure for handling invalid FPDUs is covered in the Error
Section (see section 9 on page 20)

The following is an annotated hex dump of an example FPDU sent as the
first FPDU on the stream. As such, it starts with a marker. The FPDU
contains 24 octets of the contained ULPDU, which are all zeros. The
CRC32c has been correctly calculated and can be used as a reference.
See the [DDP] and [RDMA] specification for definitions of the DDP
Control field, Queue, MSN, MO, and Send Data.

```
   Octet Contents  Annotation                                      1
   Count                                                           2
                                                                   3
   0000    00 00   Marker: Reserved                                4
   0002    00 00          FPDUPTR                                  5
   0004    00 2a   Length                                          6
   0006    40 03   DDP Control Field, Send with Last flag set      7
   0008    00 00   Reserved (STag position with no STag)           8
   000a    00 00                                                   9
   000c    00 00   Queue = 0                                      10
   000e    00 00                                                  11
   0010    00 00   MSN = 1                                        12
   0012    00 01                                                  13
   0014    00 00   MO = 0                                         14
   0016    00 00                                                  15
   0018    00 00                                                  16
                   Send Data (24 octets of zeros)                 17
   002e    00 00                                                  18
   0030    4C 86   CRC32c                                         19
   0032    B3 84                                                  20
           Figure 5 Annotated Hex Dump of an FPDU                 21
                                                                  22
```

The following is an example sent as the second FPDU of the stream   23
where the first FPDU (which is not shown here) had a length of 492   24
octets and was also a Send to Queue 0 with Last Flag set. This   25
example contains a marker.   26

```
                                                                  27
   Octet Contents  Annotation                                     28
   Count                                                          29
                                                                  30
   01ec    00 2a   Length                                         31
   01ee    40 03   DDP Control Field: Send with Last Flag set     32
   01f0    00 00   Reserved (STag position with no STag)          33
   01f2    00 00                                                  34
   01f4    00 00   Queue = 0                                      35
   01f6    00 00                                                  36
   01f8    00 00   MSN = 2                                        37
   01fa    00 02                                                  38
   01fc    00 00   MO = 0                                         39
   01fe    00 00                                                  40
   0200    00 00   Marker: Reserved                               41
   0202    00 14          FPDUPTR                                 42
   0204    00 00                                                  43
                   Send Data (24 octets of zeros)                 44
   021a    00 00                                                  45
   021c    A1 9C   CRC32c                                         46
   021e    D1 03                                                  47
       Figure 6 Annotated Hex Dump of an FPDU with Marker         48
                                                                  49
                                                                  50
                                                                  51
```

7.3 MPA on TCP Sender Segmentation

   The various TCP RFCs allow considerable choice in segmenting a TCP
   stream. In order to optimize FPDU recovery at the MPA receiver, MPA
   specifies additional segmentation rules.

   MPA MUST encapsulate the ULPDU such that there is exactly one ULPDU
   contained in one FPDU.

   An MPA-aware TCP sender SHOULD segment the outbound TCP stream such
   that there is exactly one FPDU per TCP segment.

   An MPA-aware TCP sender SHOULD, with an EMSS large enough to contain
   the FPDU, segment the outgoing TCP stream such that the first octet
   of every FPDU is aligned with the beginning of a TCP segment, and is
   entirely contained in the TCP segment.

      Implementation note: To achieve the previous segmentation rule,
      TCP's Nagle [NagleDAck] algorithm SHOULD be disabled.

   There are exceptions to the above rule. Once an ULPDU is provided to
   MPA, the MPA on TCP sender MUST transmit it or fail the connection;
   it cannot be repudiated. As a result, during changes in MTU and
   EMSS, or when TCP's Receive Window size (RWIN) becomes too small, it
   may be necessary to send FPDUs that do not conform to the
   segmentation rule above.

   A possible, but less desirable, alternative is to use IP
   fragmentation on accepted FPDUs to deal with MTU reductions or
   extremely small EMSS.

   The sender MUST still format the FPDU according to FPDU format as
   shown in Figure 2.

   On a retransmission, TCP does not necessarily preserve original TCP
   segmentation boundaries. This can lead to the loss of FPDU alignment
   and containment within a TCP segment during TCP retransmissions. An
   MPA-Aware TCP SHOULD try to preserve original TCP segmentation
   boundaries on a retransmission.

7.3.1 FPDU Size Considerations

   MPA defines the Maximum Upper Layer Protocol Data Unit (MULPDU) as
   the size of the largest ULPDU fitting in an EMSS-sized FPDU. MULPDU
   is EMSS minus the FPDU overhead (6 octets) minus space for markers
   and pad octets.

The maximum ULPDU Length for a single ULPDU MUST be computed as:

MULPDU = EMSS - (6 + 4 * Ceiling(EMSS / 512) + EMSS mod 4)

The formula above accounts for the worst-case number of markers.

The ULP SHOULD provide ULPDUs that are as large as possible, but less than or equal to MULPDU.

If the TCP implementation needs to adjust EMSS to support MTU changes, the MULPDU value is changed accordingly.

In certain rare situations, the EMSS may shrink to very small sizes. If this occurs, the MPA on TCP sender MUST not shrink the MULPDU below 128 bytes and is not required to follow the segmentation rules in Section 7.3 MPA on TCP Sender Segmentation on page 16. The value 128 is chosen as to allow ULP designers a reasonable amount of room to implement their protocol. Typical WAN scenarios will not reduce the EMSS below 512 octets.

7.4 MPA Receiver FPDU Identification

An MPA receiver MUST first verify the FPDU before passing the ULPDU to the ULP. To do this, the receiver MUST:

*   locate the start of the FPDU unambiguously,

*   verify its CRC.

If the above conditions are true, the MPA receiver passes the ULPDU to the ULP.

To detect the start of the FPDU unambiguously one of the following MUST be used:

1:   In an ordered TCP stream, the ULPDU Length field in the current FPDU when FPDU has a valid CRC, can be used to identify the beginning of the next FPDU.

2:   A Marker can always be used to locate the beginning of an FPDU (in FPDUs with valid CRCs). Since the location of the marker is known in the octet stream (sequence number space), the marker can always be found.

3:   Having found an FPDU by means of a Marker, following contiguous FPDUs can be found by using the ULPDU Lengths (from FPDUs with valid CRCs) to establish the next FPDU boundary.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

The ULPDU Length field MUST be used to determine if the entire FPDU
is present before forwarding the ULPDU to the ULP.

CRC calculation is discussed in section 7.2 on page 13 above.

7.4.1 Re-segmenting Middle boxes and non-conforming senders

Since fully conforming MPA on TCP senders start FPDUs on TCP segment
boundaries, a receiving ULP on MPA on TCP implementation may be able
to optimize the reception of data in various ways.

However, MPA receivers MUST NOT depend on FPDU Alignment on TCP
segment boundaries.

Some MPA senders may be unable to conform to the sender requirements
because their implementation of TCP is not designed with MPA in mind.
Even if the sender is fully conformant, the network may contain
"middle boxes" which modify the TCP stream by changing the
segmentation. This is generally interoperable with TCP and its users
and MPA must be no exception.

The presence of markers in MPA allows an MPA receiver to recover the
FPDUs despite these obstacles, although it may be necessary to
utilize additional buffering at the receiver to do so.

8 Connection Semantics   1
                                                                 2
8.1 Connection setup   3
                                                                 4
   MPA requires that the ULP MUST activate the framing mode on a TCP   5
   half connection at the same location in the octet stream at both the 6
   sender and the receiver. This is required in order for the marker   7
   scheme to correctly locate the markers.   8
                                                                 9
   MPA MAY be utilized separately in each direction, or enabled in both 10
   directions at once; it is up to the ULP.   11
                                                                 12
   This can be accomplished several ways, and is left up to the ULP: 13
                                                                 14
   *   The ULP MAY require MPA framing immediately after TCP connection 15
        setup. This has the advantage that no additional negotiation is 16
        needed (at least for MPA). In this case the marker MUST be the 17
        first four octets sent (this marker has the special value 0x0000, 18
        meaning it belongs to the FPDU that follows).   19
                                                                 20
   *   The ULP MAY negotiate the start of MPA. The exchange establishes 21
        that MPA (as well as other ULPs) will be used, and exactly   22
        locates the point in the octet stream where MPA is to begin   23
        operation. Again, the marker is the first four octets sent (this 24
        marker has the special value 0x0000, meaning it belongs to the 25
        FPDU that follows). Note that such a negotiation protocol is 26
        outside the scope of this specification.   27
                                                                 28
8.2 Normal Connection Teardown   29
                                                                 30
   Each half connection of MPA terminates when the ULP closes the   31
   corresponding TCP half connection.   32
                                                                 33
   A mechanism SHOULD be provided by MPA to the ULP for the ULP to be   34
   made aware that a graceful close of the LLP connection has been   35
   received by the LLP (e.g. FIN is received).   36
                                                                 37
                                                                 38
                                                                 39
                                                                 40
                                                                 41
                                                                 42
                                                                 43
                                                                 44
                                                                 45
                                                                 46
                                                                 47
                                                                 48
                                                                 49
                                                                 50
                                                                 51

9 Error Semantics

The following errors MUST be detected by MPA and the codes SHOULD be provided to the ULP:

    Code Error

    1   TCP connection closed, terminated or lost. This includes lost by timeout, too many retries, RST received or FIN received.

    2   Received MPA CRC does not match the calculated value for the FPDU.

    3   In the event that the CRC is valid, received MPA marker and 'ULPDU Length' fields do not agree on the start of a FPDU. If the FPDU start determined from previous ULPDU Length fields does not match with the MPA marker position, MPA SHOULD deliver an error to the ULP. It may not be possible to make this check as a segment arrives, but the check SHOULD be made when a gap creating an out of order sequence is closed and any time a marker points to an already identified FPDU. It is OPTIONAL for a receiver to check each marker, if multiple markers are present in an FPDU, or if the segment is received in order.

When conditions 2 or 3 above are detected, an MPA-aware TCP implementation MAY choose to silently drop the TCP segment rather than reporting the error to the ULP. In this case, the sending TCP will retry the segment, usually correcting the error, unless the problem was at the source. In that case, the source will usually exceed the number of retries and terminate the connection.

Once MPA delivers an error of any type, it MUST not deliver any additional FPDUs on that half connection.

MPA MUST NOT close the TCP connection following a reported error. Closing the connection is the responsibility of the ULP.

10 Security Considerations

   This section discusses the security considerations for MPA.

10.1 Protocol-specific Security Considerations

   The vulnerabilities of MPA to third-party attacks are no greater than
   any other protocol running over TCP. A third party, by sending
   packets into the network that are delivered to an MPA receiver, could
   launch a variety of attacks that take advantage of how MPA operates.
   For example, a third party could send random packets that are valid
   for TCP, but contain no FPDU headers. An MPA receiver reports an
   error to the ULP when any packet arrives that cannot be validated as
   an FPDU when properly located on an FPDU boundary. This would have a
   severe impact on performance. Communication security mechanisms such
   as IPsec [IPSEC] or TLS [TLS] may be used to prevent such attacks.
   Independent of how MPA operates, a third party could use ICMP packets
   to reduce the path MTU to such a small size that performance would
   likewise be severely impacted. Range checking on path MTU sizes in
   ICMP packets may be used to prevent such attacks.

10.2 Using IPSec With MPA

   IPsec can be used to protect against the packet injection attacks
   outlined above. Because IPsec is designed to secure individual IP
   packets, MPA can run above IPsec without change. IPsec packets are
   processed (e.g., integrity checked and decrypted) in the order they
   are received, and an MPA receiver will process the decrypted FPDUs
   contained in these packets in the same manner as FPDUs contained in
   unsecured IP packets.

11 IANA Considerations

   If a well-known port is chosen as the mechanism to identify a ULP on
   MPA on TCP, the well-known port must be registered with IANA.
   Because the use of the port is ULP specific, registration of the port
   with IANA is left to the ULP.

12 References   1
                                                                 2
12.1 Normative References   3
                                                                 4
   [iSCSI] Satran, Julian, draft-ietf-iscsi-15.txt, July 30, 2002 (work   5
        in progress)   6
                                                                 7
   [PathMTU] Mogul, J., and Deering, S., "Path MTU Discovery", RFC 1191,   8
        November 1990.   9
                                                                 10
   [RFC2018] Mathis, M., ahdavi, J., Floyd, S., Romanow, A., "TCP   11
        Selective Acknowledgment Options", RFC 2018, October 1996.   12
                                                                 13
   [RFC2026] Bradner, S., "The Internet Standards Process -- Revision   14
        3", BCP 9, RFC 2026, October 1996.   15
                                                                 16
   [TCP] Postel, J., "Transmission Control Protocol - DARPA Internet   17
        Program Protocol Specification", RFC 793, September 1981.   18
                                                                 19
12.2 Informative References   20
                                                                 21
   [CRCTCP] Stone J., Partridge, C., "When the CRC and TCP checksum   22
        disagree", ACM Sigcomm, Sept. 2000.   23
                                                                 24
   [DDP] H. Shah et al., "Direct Data Placement over Reliable   25
        Transports", RDMA Consortium Draft Specification   26
        draft-shah-iwarp-ddp-v1.0, October 2002   27
        (see http://www.rdmaconsortium.org/)   28
                                                                 29
   [IPSEC] Atkinson, R., Kent, S., "Security Architecture for the   30
        Internet Protocol", RFC 2401, November 1998.   31
                                                                 32
   [NagleDAck] Minshall G., Mogul, J., Saito, Y., Verghese, B.,   33
        "Application performance pitfalls and TCP's Nagle algorithm",   34
        Workshop on Internet Server Performance, May 1999.   35
                                                                 36
   [RDMA] R. Recio et al., "RDMA Protocol Specification", RDMA   37
        Consortium Draft Specification   38
        draft-recio-iwarp-rdmap-v1.0, October 2002   39
        (see http://www.rdmaconsortium.org/)   40
                                                                 41
   [SCTP] R. Stewart et al., "Stream Control Transmission Protocol", RFC   42
        2960, October 2000.   43
                                                                 44
   [STONE] Stone, J., "Checksums in the Internet", Doctoral dissertation   45
        - August 2001   46
                                                                 47
   [TLS] Dierks, T. and others, "The TLS Protocol, Version 1.0", RFC   48
        2246, January 1999.   49
                                                                 50
                                                                 51

[Williams93] Williams, R., "A PAINLESS GUIDE TO CRC ERROR DETECTION
     ALGORITHMS" - Internet publication, August 1993,
     http://www.geocities.com/SiliconValley/Pines/8659/crc.htm.

[RFC792] Internet Control Message Protocol. J. Postel. Sep-01-1981

[RFC1122] Requirements for Internet hosts - communication layers.
     R.T.   Braden. Oct-01-1989.

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
```

13 Appendix

This appendix is for information only and is NOT part of the standard.

13.1 Receiver implementation

13.1.1 Transport & Network Layer Reassembly Buffers

The use of reassembly buffers (either TCP reassembly buffers or IP fragmentation reassembly buffers) is implementation dependent. When MPA is enabled, reassembly buffers are needed if FPDU Alignment is lost or if IP fragmentation occurs. This is because the incoming out of order segment may not contain enough information for MPA to process all of the FPDU. In the usual case this should be a transient condition due to a reduction in the path MTU, so a solution does not need to be high performance. For cases where a re-segmenting middle box is present, the presence of markers significantly reduces the amount of buffering needed.

Recovery from IP Fragmentation must be transparent to the MPA Consumers.

13.1.1.1 Network Layer Reassembly Buffers

Most IP implementations set the IP Don't Fragment bit. Thus upon a
path MTU change, intermediate devices drop the IP datagram if it is
too large and reply with an ICMP message which tells the source TCP
that the path MTU has changed. This causes TCP to emit segments
conformant with the new path MTU size. Thus IP fragments under most
conditions should never occur at the receiver. But it is possible.

There are several options for implementation of network layer
reassembly buffers:

1. drop any IP fragments, and reply with an ICMP message according
    to [RFC792] (fragmentation needed and DF set) to tell the Remote
    Peer to resize its TCP segment

2. support an IP reassembly buffer, but have it of limited size
    (possibly the same size as the local link's MTU). The end Node
    would normally never advertise a path MTU larger than the local
    link MTU. It is recommended that a dropped IP fragment cause an
    ICMP message to be generated according to RFC792.

3. multiple IP reassembly buffers, of effectively unlimited size.

4. support an IP reassembly buffer for the largest IP datagram (64
    KB).

5. support for a large IP reassembly buffer which could span
    multiple IP datagrams.

An implementation should support at least 2 or 3 above, to avoid
dropping packets that have traversed the entire fabric.

There is no end-to-end ACK for IP reassembly buffers, so there is no
flow control on the buffer. The only end-to-end ACK is a TCP ACK,
which can only occur when a complete IP datagram is delivered to TCP.
Because of this, under worst case, pathological scenarios, the
largest IP reassembly buffer is the TCP receive window (to buffer
multiple IP datagrams that have all been fragmented).

Note that if the Remote Peer does not implement re-segmentation of
the data stream upon receiving the ICMP reply updating the path MTU,
it is possible to halt forward progress because the opposite peer
would continue to retransmit using a transport segment size that is
too large. This deadlock scenario is no different than if the fabric
MTU (not last hop MTU) was reduced after connection setup, and the
remote Node's behavior is not compliant with [RFC1122].

13.1.1.2 TCP Reassembly buffers

   A TCP reassembly buffer is also needed. TCP reassembly buffers are
   needed if FPDU Alignment is lost when using TCP with MPA or when the
   MPA FPDU spans multiple TCP segments (which is an exceptional case).
   This is a transient condition that only occurs when a path MTU has
   been reduced, unless there is a middle-box in the fabric that is re-
   segmenting the TCP stream.

   Since lost FPDU Alignment often means that FPDUs are incomplete, an
   MPA on TCP implementation must have a reassembly buffer large enough
   to recover an FPDU that is less than or equal to the MTU of the
   locally attached link (this should be the largest possible advertised
   TCP path MTU). If the MTU is smaller than 140 octets, the buffer MUST
   be at least 140 octets long to support the minimum FPDU size. The
   140 octets allows for the minimum MULPDU of 128, 2 octets of pad, 2
   of ULPDU_Length, 4 of CRC, and space for a possible marker. As usual,
   additional buffering may provide better performance.

   Note that if the TCP segment were not stored, it is possible to
   deadlock the MPA algorithm. If the path MTU is reduced, FPDU
   Alignment requires the source TCP to re-segment the data stream to
   the new path MTU. The source MPA will detect this condition and
   reduce the MPA segment size, but any FPDUs already posted to the
   source TCP will be re-segmented and lose FPDU Alignment. If the
   destination does not support a TCP reassembly buffer, these segments
   can never be successfully transmitted and the protocol deadlocks.

   When a complete FPDU is received, processing continues normally.

14 Author's Addresses   1
                                                              2
   Stephen Bailey   3
        Sandburst Corporation   4
        600 Federal Street   5
        Andover, MA   01810 USA   6
        Phone: +1 978 689 1614   7
        Email: steph@sandburst.com   8
                                                              9
   Paul R. Culley   10
        Hewlett-Packard Company   11
        20555 SH 249   12
        Houston, Tx. USA 77070-2698   13
        Phone: 281-514-5543   14
        Email: paul.culley@hp.com   15
                                                              16
   Uri Elzur   17
        Broadcom   18
        16215 Alton Parkway   19
        CA, 92618   20
        Phone: 949.585.6432   21
        Email: uri@broadcom.com   22
                                                              23
   Renato J Recio   24
        IBM   25
        Internal Zip 9043   26
        11400 Burnett Road   27
        Austin, Texas 78759   28
        Phone: 512-838-3685   29
        Email: recio@us.ibm.com   30
                                                              31
   John Carrier   32
        Adaptec Inc.   33
        691 South Milpitas Blvd.   34
        Milpitas, CA 95035   35
        Phone: 360-378-8526   36
        Email: John_Carrier@adaptec.com   37
                                                              38
                                                              39
                                                              40
                                                              41
                                                              42
                                                              43
                                                              44
                                                              45
                                                              46
                                                              47
                                                              48
                                                              49
                                                              50
                                                              51

15 Acknowledgments

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

Jeff Hilland   1
    Hewlett-Packard Company   2
    20555 SH 249   3
    Houston, Tx. USA 77070-2698   4
    Phone: 281-514-9489   5
    Email: jeff.hilland@hp.com   6
                                                                 7
Mike Ko   8
    IBM   9
    650 Harry Rd.   10
    San Jose, CA 95120   11
    Phone: (408) 927-2085   12
    Email: mako@us.ibm.com   13
                                                                14
Mike Krause   15
    Hewlett-Packard Corporation, 43LN   16
    19410 Homestead Road   17
    Cupertino, CA 95014 USA   18
    Phone: +1 (408) 447-3191   19
    Email: krause@cup.hp.com   20
                                                                21
Dave Minturn   22
    Intel Corporation   23
    MS JF1-210   24
    5200 North East Elam Young Parkway   25
    Hillsboro, Oregon 97124   26
    Phone: 503-712-4106   27
    Email: dave.b.minturn@intel.com   28
                                                                29
Jim Pinkerton   30
    Microsoft, Inc.   31
    One Microsoft Way   32
    Redmond, WA, USA 98052   33
    Email: jpink@microsoft.com   34
                                                                35
Hemal Shah   36
    Intel Corporation   37
    MS PTL1   38
    1501 South Mopac Expressway, #400   39
    Austin, Texas 78746   40
    Phone: 512-732-3963   41
    Email: hemal.shah@intel.com   42
                                                                43
Allyn Romanow   44
    Cisco Systems   45
    170 W Tasman Drive   46
    San Jose, CA 95134 USA   47
    Phone: +1 408 525 8836   48
    Email: allyn@cisco.com   49
                                                                50
                                                                51

Tom Talpey   1
    Network Appliance   2
    375 Totten Pond Road   3
    Waltham, MA 02451 USA   4
    Phone: +1 (781) 768-5329   5
    EMail: thomas.talpey@netapp.com   6
                                                                     7
Patricia Thaler   8
    Agilent Technologies, Inc.   9
    1101 Creekside Ridge Drive, #100   10
    M/S-RG10   11
    Roseville, CA 95678   12
    Phone: +1-916-788-5662   13
    email: pat_thaler@agilent.com   14
                                                                   15
Jim Wendt   16
    Hewlett Packard Corporation   17
    8000 Foothills Boulevard MS 5668   18
    Roseville, CA 95747-5668 USA   19
    Phone: +1 916 785 5198   20
    Email: jim_wendt@hp.com   21
                                                                   22
Jim Williams   23
    Emulex Corporation   24
    580 Main Street   25
    Bolton, MA 01740 USA   26
    Phone: +1 978 779 7224   27
    Email: jim.williams@emulex.com   28
                                                                   29
                                                                   30
                                                                   31
                                                                   32
                                                                   33
                                                                   34
                                                                   35
                                                                   36
                                                                   37
                                                                   38
                                                                   39
                                                                   40
                                                                   41
                                                                   42
                                                                   43
                                                                   44
                                                                   45
                                                                   46
                                                                   47
                                                                   48
                                                                   49
                                                                   50
                                                                   51

16 Full Copyright Statement   1
                                                                     2
                                                                         18
                                                                         19
                                                                         20
                                                                         21
                                                                         22
                                                                         23
                                                                         24
                                                                         25
                                                                         26
                                                                         27
                                                                         28
                                                                         29
                                                                         30
                                                                         31
                                                                         32
                                                                         33
                                                                         34
                                                                         35
                                                                         36
                                                                         37
                                                                         38
                                                                         39
                                                                         40
                                                                         41
                                                                         42
                                                                         43
                                                                         44
                                                                         45
                                                                         46
                                                                         47
                                                                         48
                                                                         49
                                                                         50
                                                                         51