Transport Area Working Group   G. Renker
Internet-Draft   G. Fairhurst
Intended status: Standards Track   University of Aberdeen
Expires: May 1, 2008   October 29, 2007

MIB for the UDP-Lite protocol
draft-ietf-tsvwg-udplite-mib-03

Status of this Memo

Copyright Notice

Abstract

   This document specifies a Management Information Base (MIB) module
   for the Lightweight User Datagram Protocol. It defines a set of new
   MIB objects to characterise the behaviour and performance of
   transport layer endpoints deploying UDP-Lite. UDP-Lite resembles
   UDP, but differs from the semantics of UDP by the addition of a
   single option. This adds the capability for variable-length data
   checksum coverage, which can benefit a class of applications that
   prefer delivery of (partially) corrupted datagram payload data in
   preference to discarding the datagram.

Table of Contents

1. Introduction

   The Lightweight User Datagram Protocol (UDP-Lite) [RFC3828] (also
   known as UDPLite) is an IETF standards-track transport protocol. The
   operation of UDP-Lite is similar to the User Datagram Protocol (UDP)
   [RFC0768], but can also serve applications in error-prone network
   environments that prefer to have partially damaged payloads delivered
   rather than discarded. This is achieved by changing the semantics of
   the UDP Length field to that of a Checksum Coverage field. If this
   feature is not used, UDP-Lite is semantically identical to UDP.

   The interface of UDP-Lite differs from that of UDP by the addition of
   a single option, which communicates a length value. At the sender
   this specifies the intended datagram checksum coverage; at the
   receiver it signifies a minimum coverage threshold for incoming
   datagrams. This length value may also be modified during the
   lifetime of a connection. UDP-Lite does not provide mechanisms to
   negotiate the checksum coverage between the sender and receiver.
   Where required, this needs to be communicated by another protocol.
   DCCP [RFC4340] for instance includes a capability to negotiate
   checksum coverage values.

   This document defines a set of runtime statistics (variables) that
   facilitate both network management/monitoring as well as unified
   comparisons between different protocol implementations and operating
   environments. To provide a common interface for users and
   implementors of UDP-Lite modules, the definitions of these runtime
   statistics are provided as a MIB module using the SMIv2 format
   [RFC2578].

1.1. Relationship to the UDP-MIB

   The similarities between UDP and UDP-Lite suggest that the MIB module
   for UDP-Lite should resemble that of UDP [RFC4113], with extensions
   corresponding to the additional capabilities of UDP-Lite. The UDP-
   Lite MIB module is placed beneath the mib-2 subtree, adhering to the
   familiar structure of the UDP-MIB module to ease integration.

   In particular, these well-known basic counters are supported:

   o InDatagrams

   o NoPorts

   o InErrors

   o OutDatagrams

The following read-only variables have been added to the basic
structure used in the UDP-MIB module:

   InPartialCov: The number of received datagrams, with a valid
   format and checksum, whose checksum coverage is strictly less than
   the datagram length.

   InBadChecksum: The number of received datagrams with an invalid
   checksum (i.e. where the receiver-recalculated UDP-Lite checksum
   does not match that in the Checksum field). Unlike NoPorts, this
   error type also counts as InErrors.

   OutPartialCov: The number of sent datagrams with a valid format
   and checksum whose checksum coverage is strictly less than the
   datagram length.

All non-error counters used in this document are 64-bit counters.
This is a departure from UDP, which traditionally used 32-bit
counters and mandates 64-bit counters only on fast networks
[RFC4113]. This choice is justified by the fact that UDP-Lite is a
more recent protocol, and that network speeds continue to grow.

Another difference from the UDP MIB module is that the UDP-Lite MIB
module does not support an IPv4-only listener table. This feature
was present only for compatibility reasons and is superseded by the
more informative endpoint table. Two columnar objects have been
added to this table:

   udpliteEndpointMinCoverage: The minimum acceptable receiver
   checksum coverage length [RFC3828]. This value may be manipulated
   by the application attached to the receiving endpoint.

   udpliteEndpointViolCoverage: This object is optional and counts
   the number of valid datagrams with a checksum coverage value less
   than the corresponding value of udpliteEndpointMinCoverage.
   Although being otherwise valid, such datagrams are discarded
   rather than passed to the application. This object thus serves to
   separate cases of violated coverage from other InErrors.

The second entry is not required to manage the transport protocol and
hence is not mandatory. It may be implemented to assist in debugging
application design and configuration.

The UDP-Lite MIB module also provides a discontinuity object to help
determine whether one or more of its counters experienced a
discontinuity event. This is an event, other than re-initialising
the management system, which invalidates the management entity's
understanding of the counter values.

For example, if UDP-Lite is implemented as a loadable operating
system module, a module load or unload would produce a discontinuity.
By querying the value of udpliteStatsDiscontinuityTime, a management
entity can determine whether or not a discontinuity event has
occurred.

1.2. Relationship to HOST-RESOURCES-MIB and SYSAPPL-MIB

The UDP-Lite endpoint table contains one columnar object,
udpliteEndpointProcess, reporting a unique value which identifies a
distinct piece of software associated with this endpoint. (When more
than one piece of software is associated with this endpoint, a
representative is chosen; so that consecutive queries consistently
refer to the same identifier, as long as the representative piece of
software is running and still associated with the endpoint.)

The value of udpliteEndpointProcess is reported as an Unsigned32; and
it shares with the hrSWRunIndex of the HOST-RESOURCES-MIB [RFC2790]
and the sysApplElmtRunIndex of the SYSAPPL-MIB [RFC2287] the
requirement that, wherever possible, this should be the native and
unique identification number employed by the system.

If the SYSAPPL-MIB module is available, the value of
udpliteEndpointProcess should correspond to the appropriate value of
sysApplElmtRunIndex. If not available, an alternative should be used
(e.g. the hrSWRunIndex of the HOST-RESOURCES-MIB module).

1.3. Interpretation of the MIB Variables

Figure 1 shows an informal survey of the packet processing path, with
reference to counter names in brackets.

```
Received UDP-Lite Datagrams
   |
   |                  +- Full Coverage --------------------+-> Deliver
   |                  |                                     |
   +- Valid Header--+              +- >= Rec. Coverage --+
   |  (InDatagrams) |              |
   |                  +- Partial -----+
   |                  (InPartialCov)  |
   |                                  +-  < Rec. Coverage --+
   |                          (EndpointViolCoverage) |
   |                                                  |
   |                                                  |
   +- Header Error ---+                               |
   |                  |                               |
   +- Checksum Error -+-----------------------------------+-> Discard
   |  (InBadChecksum)                                 (InErrors)
   |
   +- Port Error -------------------------------------------> Discard
      (NoPorts)
```
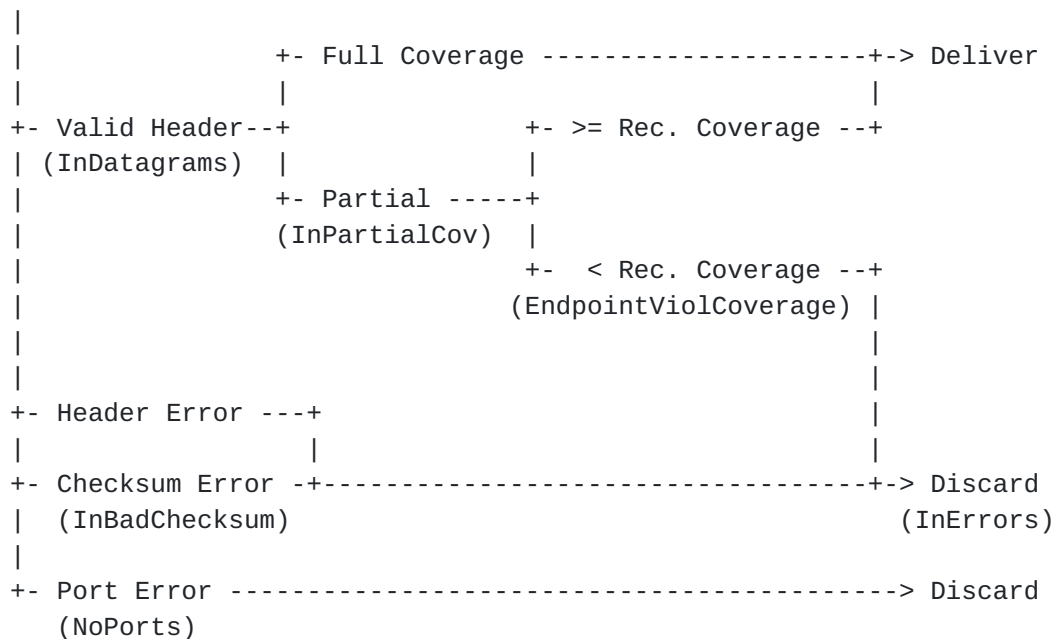
                    Figure 1: UDP-Lite Input Processing Path

   A platform-independent test of the UDP-Lite implementations in two
   connected end hosts may be performed as follows.

   On the sending side, OutDatagrams and OutPartialCov are observed.
   The ratio OutPartialCov/OutDatagrams describes the fraction (between
   0 and 1) of datagrams using partial checksum coverage.

   On the receiving side, InDatagrams, InPartialCov, and InErrors are
   monitored. If datagrams are received from the given sender, InErrors
   is close to zero, and InPartialCov is zero, no partial coverage is
   employed. If no datagrams are received and InErrors increases
   proportionally with the sending rate, a configuration error is likely
   (a wrong value of receiver minimum checksum coverage).

   The InBadChecksum counter reflects errors that may persist following
   end-host processing, router processing, or link processing (this
   includes illegal coverage values as defined in [RFC3828], since
   checksum and checksum coverage are mutually inter-dependent). In
   particular, InBadChecksum can serve as an indicator of the residual
   link bit error rate: on links with higher bit error rates, a lower
   value of the checksum coverage may help to reduce both the values of
   InErrors and InBadChecksum.

   By observing these values and adapting the configuration, a setting
   may then be found that is more adapted to the specific type of link,
   as well as the type of payload; indicated by a reduction of the

number of discarded datagrams (InErrors), leading to an improved
performance.

The above statistics are elementary and can be used to derive the
following information:

o   The total number of incoming datagrams is InDatagrams + InErrors +
    NoPorts

o   The number of InErrors that were discarded due to problems other
    than bad checksum is InErrors - InBadChecksum

o   The number of InDatagrams that have full coverage is InDatagrams -
    InPartialCov.

o   The number of OutDatagrams that have full coverage is OutDatagrams
    - OutPartialCov.

The following Case diagram [CASE] summarises the relationships
between the counters on the input processing path.

```
        Transport Layer Interface
     -----------------------------------------------------------------
                    /\
                    ||
                 --------------------------- InDatagrams
                    ||                            ^
                    ||                            |
                    ||                            |
                    ||---------------------> InPartialCov
                    ||                            |
                    ||                            |
                    ||                            v
                    ||                   EndpointViolCoverage
                    ||                            |
       NoPorts <--------||                        |
                    ||                            |
                    ||------> InBadChecksum ------>|
                    ||                            |
                    ||                            |
                    ||                            v
                    ||----------------------> InErrors
                    ||
                    ||
     -----------------------------------------------------------------
             Network Layer Interface

          Figure 2: Counters for received UDP-Lite Datagrams
```
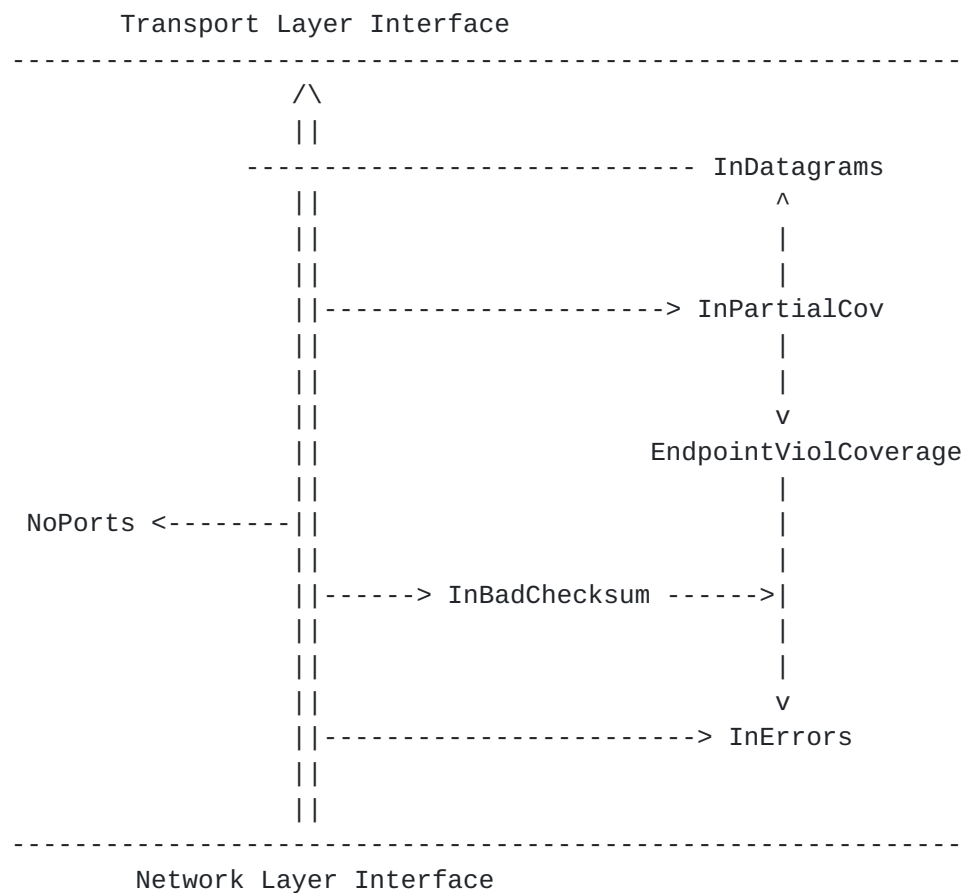
A configuration error may occur when a sender chooses a coverage
value for the datagrams that it sends that is less than the minimum
coverage configured by the intended recipient. The minimum coverage
is set on a per-session basis by the application associated with the
listening endpoint, and its current value is recorded in the
udpliteEndpointTable. Reception of valid datagrams with a checksum
coverage value less than this threshold results in dropping the
datagram [RFC3828] and incrementing InErrors. To improve debugging
of such (misconfigured) cases, an implementer may choose to support
the optional udpliteEndpointViolCoverage entry in the endpoint table
(Section 1.1.) that specifically counts datagrams falling in this
category. Without this feature, failure due to misconfiguration can
not be distinguished from datagram processing failure.

1.4. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in BCP 14, [RFC2119].

2. The Internet-Standard Management Framework

   For a detailed overview of the documents that describe the current
   Internet-Standard Management Framework, please refer to section 7 of
   RFC 3410 [RFC3410].

   Managed objects are accessed via a virtual information store, termed
   the Management Information Base or MIB. MIB objects are generally
   accessed through the Simple Network Management Protocol (SNMP).
   Objects in the MIB are defined using the mechanisms defined in the
   Structure of Management Information (SMI). This memo specifies a MIB
   module that is compliant to the SMIv2, which is described in STD 58,
   RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580
   [RFC2580].

[3](#). Definitions

```
UDPLITE-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    mib-2, Unsigned32,
    Counter32, Counter64    FROM SNMPv2-SMI          -- [RFC2578]

    TimeStamp               FROM SNMPv2-TC           -- [RFC2579]

    MODULE-COMPLIANCE,
    OBJECT-GROUP            FROM SNMPv2-CONF          -- [RFC2580]

    InetAddress,
    InetAddressType,
    InetPortNumber          FROM INET-ADDRESS-MIB;    -- [RFC4001]
```

```
udpliteMIB MODULE-IDENTITY
    LAST-UPDATED "200710290000Z"        -- Mon, 29th Oct 2007
    ORGANIZATION "IETF TSV Working Group (TSVWG)"
    CONTACT-INFO
            "IETF TSV Working Group
             http://www.ietf.org/html.charters/tsvwg-charter.html
             Mailing List: tsvwg@ietf.org

             Gerrit Renker, Godred Fairhurst
             Electronics Research Group
             School of Engineering, University of Aberdeen
             Fraser Noble Building, Aberdeen AB24 3UE, UK"
    DESCRIPTION
            "The MIB module for managing UDP-Lite implementations.
             Copyright (C) The IETF Trust (2007). This version of
             this MIB module is part of RFC ZZZ; see the RFC
             itself for full legal notices."

    -- RFC Ed.: replace ZZZ with actual RFC number & remove this note

    REVISION "200710290000Z"            -- Mon, 29th Oct 2007
    DESCRIPTION
            "Initial SMIv2 revision, based on the format of the UDP
             MIB module (RFC 4113) and published as RFC ZZZ."

    -- RFC Ed.: replace ZZZ with actual RFC number & remove this note

    ::= { mib-2 XXX }

    -- RFC Ed.: replace XXX with OBJECT-IDENTIFIER & remove this note


udplite      OBJECT IDENTIFIER ::= { udpliteMIB 1 }


udpliteInDatagrams OBJECT-TYPE          -- as in UDP-MIB
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The total number of UDP-Lite datagrams that were
             delivered to UDP-Lite users.
             Discontinuities in the value of this counter can occur
             at re-initialisation of the management system, and at
             other times as indicated by the value of
             udpliteStatsDiscontinuityTime."
    ::= { udplite 1 }
```

```
    udpliteInPartialCov OBJECT-TYPE          -- new in UDP-Lite
        SYNTAX      Counter64
        MAX-ACCESS read-only
        STATUS      current
        DESCRIPTION
                "The total number of UDP-Lite datagrams that were
                 delivered to UDP-Lite users (applications) and whose
                 checksum coverage was strictly less than the datagram
                 length.
                 Discontinuities in the value of this counter can occur
                 at re-initialisation of the management system, and at
                 other times as indicated by the value of
                 udpliteStatsDiscontinuityTime."
        ::= { udplite 2 }


    udpliteNoPorts OBJECT-TYPE               -- as in UDP-MIB
        SYNTAX      Counter32
        MAX-ACCESS read-only
        STATUS      current
        DESCRIPTION
                "The total number of received UDP-Lite datagrams for
                 which there was no listener at the destination port.

                 Discontinuities in the value of this counter can occur
                 at re-initialisation of the management system, and at
                 other times as indicated by the value of
                 udpliteStatsDiscontinuityTime."
        ::= { udplite 3 }


    udpliteInErrors OBJECT-TYPE              -- as in UDP-MIB
        SYNTAX      Counter32
        MAX-ACCESS read-only
        STATUS      current
        DESCRIPTION
                "The number of received UDP-Lite datagrams that could not
                 be delivered for reasons other than the lack of an
                 application at the destination port.
                 Discontinuities in the value of this counter can occur
                 at re-initialisation of the management system, and at
                 other times as indicated by the value of
                 udpliteStatsDiscontinuityTime."
        ::= { udplite 4 }
```

```
    udpliteInBadChecksum OBJECT-TYPE        -- new in UDP-Lite
        SYNTAX      Counter32
        MAX-ACCESS read-only
        STATUS      current
        DESCRIPTION
                "The number of received UDP-Lite datagrams whose checksum
                 could not be validated. This includes illegal checksum
                 coverage values, as their use would lead to incorrect
                 checksums.
                 Discontinuities in the value of this counter can occur
                 at re-initialisation of the management system, and at
                 other times as indicated by the value of
                 udpliteStatsDiscontinuityTime."
        REFERENCE "RFC 3828, section 3.1"
        ::= { udplite 5 }


    udpliteOutDatagrams OBJECT-TYPE        -- as in UDP-MIB
        SYNTAX      Counter64
        MAX-ACCESS read-only
        STATUS      current
        DESCRIPTION
                "The total number of UDP-Lite datagrams sent from this
                 entity.
                 Discontinuities in the value of this counter can occur
                 at re-initialisation of the management system, and at
                 other times as indicated by the value of
                 udpliteStatsDiscontinuityTime."
        ::= { udplite 6 }


    udpliteOutPartialCov OBJECT-TYPE        -- new in UDP-Lite
        SYNTAX      Counter64
        MAX-ACCESS read-only
        STATUS      current
        DESCRIPTION
                "The total number of udpliteOutDatagrams whose
                 checksum coverage was strictly less than the
                 datagram length.
                 Discontinuities in the value of this counter can occur
                 at re-initialisation of the management system, and at
                 other times as indicated by the value of
                 udpliteStatsDiscontinuityTime."
        ::= { udplite 7 }


    udpliteEndpointTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF UdpLiteEndpointEntry
```

         MAX-ACCESS not-accessible
         STATUS   current
         DESCRIPTION
               "A table containing information about this entity's
                UDP-Lite endpoints on which a local application is
                currently accepting or sending datagrams.

                The address type in this table represents the address
                type used for the communication, irrespective of the
                higher-layer abstraction. For example, an application
                using IPv6 'sockets' to communicate via IPv4 between
                ::ffff:10.0.0.1 and ::ffff:10.0.0.2 would use
                InetAddressType ipv4(1).

                Like the udpTable in RFC 4113, this table also allows
                the representation of an application that completely
                specifies both local and remote addresses and ports. A
                listening application is represented in three possible
                ways:

                1) An application that is willing to accept both IPv4
                   and IPv6 datagrams is represented by a
                   udpliteEndpointLocalAddressType of unknown(0) and a
                   udpliteEndpointLocalAddress of ''h (a zero-length
                   octet-string).

                2) An application that is willing to accept only IPv4
                   or only IPv6 datagrams is represented by a
                   udpliteEndpointLocalAddressType of the appropriate
                   address type and a udpliteEndpointLocalAddress of
                   '0.0.0.0' or '::' respectively.

                3) An application that is listening for datagrams only
                   for a specific IP address but from any remote
                   system is represented by a
                   udpliteEndpointLocalAddressType of the appropriate
                   address type, with udpliteEndpointLocalAddress
                   specifying the local address.

                In all cases where the remote address is a wildcard,
                the udpliteEndpointRemoteAddressType is unknown(0),
                the udpliteEndpointRemoteAddress is ''h (a zero-length
                octet-string), and the udpliteEndpointRemotePort is 0.

                If the operating system is demultiplexing UDP-Lite
                packets by remote address/port, or if the application
                has 'connected' the socket specifying a default remote
                address/port, the udpliteEndpointRemote* values should

                  be used to reflect this."
         ::= { udplite 8 }

      udpliteEndpointEntry OBJECT-TYPE
          SYNTAX   UdpLiteEndpointEntry
          MAX-ACCESS not-accessible
          STATUS   current
          DESCRIPTION
                  "Information about a particular current UDP-Lite endpoint.

                  Implementers need to pay attention to the sizes of
                  udpliteEndpointLocalAddress/RemoteAddress, as OIDs of
                  column instances in this table must have no more than
                  128 sub-identifiers in order to remain accessible with
                  SNMPv1, SNMPv2c, and SNMPv3."
          INDEX   { udpliteEndpointLocalAddressType,
                     udpliteEndpointLocalAddress,
                     udpliteEndpointLocalPort,
                     udpliteEndpointRemoteAddressType,
                     udpliteEndpointRemoteAddress,
                     udpliteEndpointRemotePort,
                     udpliteEndpointInstance }
          ::= { udpliteEndpointTable 1 }

      UdpLiteEndpointEntry ::= SEQUENCE {
              udpliteEndpointLocalAddressType   InetAddressType,
              udpliteEndpointLocalAddress   InetAddress,
              udpliteEndpointLocalPort   InetPortNumber,
              udpliteEndpointRemoteAddressType InetAddressType,
              udpliteEndpointRemoteAddress   InetAddress,
              udpliteEndpointRemotePort   InetPortNumber,
              udpliteEndpointInstance   Unsigned32,
              udpliteEndpointProcess   Unsigned32,
              udpliteEndpointMinCoverage   Unsigned32,
              udpliteEndpointViolCoverage   Counter32
          }

```
udpliteEndpointLocalAddressType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS not-accessible
    STATUS      current
    DESCRIPTION
            "The address type of udpliteEndpointLocalAddress.  Only
             IPv4, IPv4z, IPv6, and IPv6z addresses are expected, or
             unknown(0) if datagrams for all local IP addresses are
             accepted."
    ::= { udpliteEndpointEntry 1 }
```

    udpliteEndpointLocalAddress OBJECT-TYPE
        SYNTAX   InetAddress
        MAX-ACCESS not-accessible
        STATUS   current
        DESCRIPTION
               "The local IP address for this UDP-Lite endpoint.

                The value of this object can be represented in three
                possible ways, depending on the characteristics of the
                listening application:

                1. For an application that is willing to accept both
                   IPv4 and IPv6 datagrams, the value of this object
                   must be ''h (a zero-length octet-string), with
                   the value of the corresponding instance of the
                   EndpointLocalAddressType object being unknown(0).

                2. For an application that is willing to accept only
                   IPv4 or only IPv6 datagrams, the value of this
                   object must be '0.0.0.0' or '::', respectively,
                   while the corresponding instance of the
                   EndpointLocalAddressType object represents the
                   appropriate address type.

                3. For an application that is listening for data
                   destined only to a specific IP address, the value
                   of this object is the specific IP address for
                   which this node is receiving packets, with the
                   corresponding instance of the
                   EndpointLocalAddressType object representing the
                   appropriate address type.

                As this object is used in the index for the
                udpliteEndpointTable, implementors should be careful
                not to create entries that would result in OIDs with
                more than 128 sub-identifiers; this is because of SNMP
                and SMI limitations."
        ::= { udpliteEndpointEntry 2 }

    udpliteEndpointLocalPort OBJECT-TYPE
        SYNTAX   InetPortNumber
        MAX-ACCESS not-accessible
        STATUS   current
        DESCRIPTION
               "The local port number for this UDP-Lite endpoint."
        ::= { udpliteEndpointEntry 3 }

    udpliteEndpointRemoteAddressType OBJECT-TYPE
        SYNTAX   InetAddressType
        MAX-ACCESS not-accessible
        STATUS   current
        DESCRIPTION
                "The address type of udpliteEndpointRemoteAddress. Only
                 IPv4, IPv4z, IPv6, and IPv6z addresses are expected, or
                 unknown(0) if datagrams for all remote IP addresses are
                 accepted. Also, note that some combinations of
                 udpliteEndpointLocalAdressType and
                 udpliteEndpointRemoteAddressType are not supported. In
                 particular, if the value of this object is not
                 unknown(0), it is expected to always refer to the
                 same IP version as udpliteEndpointLocalAddressType."
        ::= { udpliteEndpointEntry 4 }

    udpliteEndpointRemoteAddress OBJECT-TYPE
        SYNTAX   InetAddress
        MAX-ACCESS not-accessible
        STATUS   current
        DESCRIPTION
                "The remote IP address for this UDP-Lite endpoint. If
                 datagrams from any remote system are to be accepted,
                 this value is ''h (a zero-length octet-string).
                 Otherwise, it has the type described by
                 udpliteEndpointRemoteAddressType and is the address of
                 the remote system from which datagrams are to be
                 accepted (or to which all datagrams will be sent).

                 As this object is used in the index for the
                 udpliteEndpointTable, implementors should be careful
                 not to create entries that would result in OIDs with
                 more than 128 sub-identifiers; this is because of SNMP
                 and SMI limitations."
        ::= { udpliteEndpointEntry 5 }

    udpliteEndpointRemotePort OBJECT-TYPE
        SYNTAX   InetPortNumber
        MAX-ACCESS not-accessible
        STATUS   current
        DESCRIPTION
                "The remote port number for this UDP-Lite endpoint. If
                 datagrams from any remote system are to be accepted,
                 this value is zero."
        ::= { udpliteEndpointEntry 6 }

    udpliteEndpointInstance OBJECT-TYPE
        SYNTAX   Unsigned32 (1..'ffffffff'h)
        MAX-ACCESS not-accessible
        STATUS   current
        DESCRIPTION
                "The instance of this tuple. This object is used to
                 distinguish among multiple processes 'connected' to
                 the same UDP-Lite endpoint. For example, on a system
                 implementing the BSD sockets interface, this would be
                 used to support the SO_REUSEADDR and SO_REUSEPORT
                 socket options."
        ::= { udpliteEndpointEntry 7 }

    udpliteEndpointProcess OBJECT-TYPE
        SYNTAX   Unsigned32
        MAX-ACCESS read-only
        STATUS   current
        DESCRIPTION
                "A unique value corresponding to a piece of software
                 running on this endpoint.

                 If this endpoint is associated with more than one piece
                 of software, the agent should choose one of these; such
                 that subsequent reads will consistently return the same
                 value, as long as the representative piece of software
                 is running and still associated with the endpoint. The
                 implementation may use any algorithm satisfying these
                 constraints (e.g. choosing the entity with the oldest
                 start time).

                 This identifier is platform-specific. Wherever possible,
                 it should use the system's native, unique identification
                 number as value.

                 If the SYSAPPL-MIB module is available, the value should
                 be the same as sysApplElmtRunIndex. If not available, an
                 alternative should be used (e.g. the hrSWRunIndex of the
                 HOST-RESOURCES-MIB module).

                 If it is not possible to uniquely identify the pieces of
                 software associated with this endpoint, then the value
                 zero should be used. (Note that zero is otherwise a
                 valid value for sysApplElmtRunIndex.)"
        ::= { udpliteEndpointEntry 8 }

    udpliteEndpointMinCoverage OBJECT-TYPE -- new in UDP-Lite
        SYNTAX   Unsigned32
        MAX-ACCESS read-only
        STATUS   current
        DESCRIPTION
                "The minimum checksum coverage expected by this endpoint.
                 A value of 0 indicates that only fully covered datagrams
                 are accepted."
        REFERENCE "RFC 3828, section 3.1"
        ::= { udpliteEndpointEntry 9 }

    udpliteEndpointViolCoverage OBJECT-TYPE -- new / optional in UDP-Lite
        SYNTAX   Counter32
        MAX-ACCESS read-only
        STATUS   current
        DESCRIPTION
                "The number of datagrams received by this endpoint whose
                 checksum coverage violated the minimum coverage threshold
                 set for this connection (i.e. all valid datagrams whose
                 checksum coverage was strictly smaller than the minimum,
                 as defined in RFC 3828).
                 Discontinuities in the value of this counter can occur
                 at re-initialisation of the management system, and at
                 other times as indicated by the value of
                 udpliteStatsDiscontinuityTime."
        ::= { udpliteEndpointEntry 10 }

    udpliteStatsDiscontinuityTime OBJECT-TYPE
        SYNTAX   TimeStamp
        MAX-ACCESS read-only
        STATUS   current
        DESCRIPTION
                "The value of sysUpTime at the most recent occasion at
                 which one or more of the UDP-Lite counters suffered a
                 discontinuity.
                 A value of zero indicates no such discontinuity has
                 occurred since the last re-initialisation of the local
                 management subsystem."
        ::= { udplite 9 }

    -- Conformance Information

    udpliteMIBConformance OBJECT IDENTIFIER ::= { udpliteMIB 2 }

    udpliteMIBCompliance MODULE-COMPLIANCE

```
          STATUS     current
          DESCRIPTION
                "The compliance statement for systems that implement
                 UDP-Lite.

                 There are a number of INDEX objects that cannot be
                 represented in the form of OBJECT clauses in SMIv2,
                 but for which we have the following compliance
                 requirements, expressed in OBJECT clause form in this
                 description clause:

                 -- OBJECT      udpliteEndpointLocalAddressType
                 -- SYNTAX      InetAddressType { unknown(0), ipv4(1),
                 --                                ipv6(2), ipv4z(3),
                 --                                ipv6z(4) }
                 -- DESCRIPTION
                 --    Support for dns(16) is not required.
                 -- OBJECT      udpliteEndpointLocalAddress



                 -- SYNTAX      InetAddress (SIZE(0|4|8|16|20))
                 -- DESCRIPTION
                 --    Support is only required for zero-length
                 --    octet-strings, and for scoped and unscoped
                 --    IPv4 and IPv6 addresses.
                 -- OBJECT      udpliteEndpointRemoteAddressType
                 -- SYNTAX      InetAddressType { unknown(0), ipv4(1),
                 --                                ipv6(2), ipv4z(3),
                 --                                ipv6z(4) }
                 -- DESCRIPTION
                 --    Support for dns(16) is not required.
                 -- OBJECT      udpliteEndpointRemoteAddress
                 -- SYNTAX      InetAddress (SIZE(0|4|8|16|20))
                 -- DESCRIPTION
                 --    Support is only required for zero-length
                 --    octet-strings, and for scoped and unscoped
                 --    IPv4 and IPv6 addresses.
                "
          MODULE  -- this module
              MANDATORY-GROUPS { udpliteBaseGroup,
                                 udplitePartialCsumGroup,
                                 udpliteEndpointGroup    }
           GROUP              udpliteAppGroup
           DESCRIPTION
                "This group is optional and provides supplementary
                 information about the effectiveness of using minimum
                 checksum coverage thresholds on endpoints."
```

```
       ::= { udpliteMIBConformance 1 }


   udpliteMIBGroups OBJECT IDENTIFIER ::= { udpliteMIBConformance 2 }


   udpliteBaseGroup OBJECT-GROUP          -- as in UDP
       OBJECTS   { udpliteInDatagrams, udpliteNoPorts, udpliteInErrors,
                   udpliteOutDatagrams, udpliteStatsDiscontinuityTime }
       STATUS     current
       DESCRIPTION
             "The group of objects providing for counters of
              basic UDP-like statistics."
       ::= { udpliteMIBGroups 1 }


   udplitePartialCsumGroup OBJECT-GROUP  -- specific to UDP-Lite
   OBJECTS   { udpliteInPartialCov,
              udpliteInBadChecksum,
              udpliteOutPartialCov  }
       STATUS     current
       DESCRIPTION
             "The group of objects providing for counters of
              transport-layer statistics exclusive to UDP-Lite."
       ::= { udpliteMIBGroups 2 }


   udpliteEndpointGroup OBJECT-GROUP
       OBJECTS    { udpliteEndpointProcess, udpliteEndpointMinCoverage }
       STATUS     current
       DESCRIPTION
             "The group of objects providing for the IP version
              independent management of UDP-Lite 'endpoints'."
       ::= { udpliteMIBGroups 3 }


   udpliteAppGroup OBJECT-GROUP
       OBJECTS    { udpliteEndpointViolCoverage }
       STATUS     current
       DESCRIPTION
             "The group of objects that provide application-level
              information for the configuration management of
              UDP-Lite 'endpoints'."
       ::= { udpliteMIBGroups 4 }

   END
```

4. Security Considerations

   There are no management objects defined in this MIB module that have
   a MAX-ACCESS clause of read-write and/or read-create. So, if this
   MIB module is implemented correctly, then there is no risk that an
   intruder can alter or create any management objects of this MIB
   module via direct SNMP SET operations.

   Some of the readable objects in this MIB module (i.e., objects with a
   MAX-ACCESS other than not-accessible) may be considered sensitive or
   vulnerable in some network environments. It is thus important to
   control even GET and/or NOTIFY access to these objects and possibly
   to even encrypt the values of these objects when sending them over
   the network via SNMP. These are the tables and objects and their
   sensitivity/vulnerability:

   Since UDP-Lite permits the delivery of (partially) corrupted data to
   an end host, the counters defined in this MIB module may be used to
   infer information about the characteristics of the end-to-end path
   over which the datagrams are communicated.

   The indices of the udpliteEndpointTable contain information about the
   listeners on an entity. In particular, the udpliteEndpointLocalPort
   and udpliteLocalPort objects in the indices can be used to identify
   what ports are open on the machine and which attacks are likely to
   succeed, without the attacker having to run a port scanner. The
   table also identifies the currently listening UDP-Lite ports. This
   could be used to infer the type of application associated with the
   port at the receiver. The udpliteEndpointMinCoverage provides
   information about the requirements of the transport service
   associated with a specific UDP-Lite port. This provides additional
   detail concerning the type of application associated with the port at
   the receiver.

   SNMP versions prior to SNMPv3 did not include adequate security.
   Even if the network itself is secure (for example by using IPsec),
   even then, there is no control as to who on the secure network is
   allowed to access and GET/SET (read/change/create/delete) the objects
   in this MIB module.

   It is RECOMMENDED that implementers consider the security features as
   provided by the SNMPv3 framework (see RFC 3410 [RFC3410], section 8),
   including full support for the SNMPv3 cryptographic mechanisms (for
   authentication and privacy).

   Further, deployment of SNMP versions prior to SNMPv3 is NOT
   RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to
   enable cryptographic security. It is then a customer/operator

responsibility to ensure that the SNMP entity giving access to an
instance of this MIB module is properly configured to give access to
the objects only to those principals (users) that have legitimate
rights to indeed GET or SET (change/create/delete) them.

5. IANA Considerations

The MIB module in this document uses the following IANA-assigned
OBJECT IDENTIFIER values recorded in the SMI Numbers registry:

```
+------------+------------------------+
| Descriptor | OBJECT IDENTIFIER value |
+------------+------------------------+
| udpliteMIB |      { mib-2 XXX }     |
+------------+------------------------+
```

==> Note to the RFC Editor (to be removed prior to publication):

    The IANA is requested to assign a value for "XXX" under the 'mib-2'
    subtree and to record the assignment in the SMI Numbers registry.
    When the assignment has been made, the RFC Editor is asked to replace
    "XXX" (here and in the MIB module) with the assigned value and to
    remove this note.

6. Acknowledgments

The design of the MIB module presented in this document owes much to the format of the module presented in [RFC4113].

==> NOTE TO THE RFC EDITOR: PLEASE REMOVE THIS LOG PRIOR TO PUBLICATION

   Revision 00 of draft-ietf-tsvwg-udplite-mib was obsoleted by revision
   01 due to a syntax error (quote transposed with full-stop) in the MIB
   module, which is corrected in rev-01. Thanks to Magnus Westerlund
   for identifying this.

   Draft draft-ietf-tsvwg-udplite-mib-00 was published as a work item of
   tsvwg, June 2007.

   The following changelog lists the changes up to revision 02 (which
   became rev-00/01 of this document) of the preceding individual draft
   submission draft-renker-tsvwg-udplite-mib.

   Changes introduced in rev-01:

   o General:

           - incremented revision number to 01

           - updated date to November

           - rephrased abstract

   o Section 1:

           - rephrased the beginning of the second paragraph

   o Section 1.1:

           - rephrased some items

           - added missing InBadChecksum heading

           - updated text to refer to 64bit counters

   o Section 1.3:

           - removed 'x' in 'datagrams'

           - rephrased for clarity

           - Figure 1: missing bracketed text should be InErrors

           - Figure 1: correction - NoPorts are not counted as InDatagrams

o [Section 2](#):

      - made the "Editor's Note" stand out more

o [Section 3](#) / MIB:

      - upgraded 11 32bit counters to 64bit

      - moved from experimental to mib-2

      - updated revision date

o [Section 4](#):

      - some minor changes

o [Section 5](#):

      - again highlighted the Editor's Note by using `==>' to make it
      consistent

Changes introduced in rev-02:

o General:

      - updated month, date, and revision

      - changed `transport layer entities' to `endpoints' in abstract

o [Section 1](#):

      - added missing comma after `option'

      - split explanatory clause after colon into standalone clause

o [Section 1.1](#):

      - added a bullet list of standard counters known from the UDP
      MIB

      - added a note that NoPorts does not increment InErrors

      - removed description of InBadCoverage (no longer in MIB, see
      below)

      - qualified note with on 64-bit counters wrt `non-error'
      counters, reflecting the change that all error counters have
      been changed to Counter32

- Nit: changed `fairly recent' -> `more recent'

- removed traces of InBadCoverage (see below)

o [Section 1.3](#):

- corrected figure 1 (NoPorts was inconsistent with text; Rec
Coverage was incorrectly labelled, it didn't show
EndpointViolCoverage; slight reformatting)

- Nit: `wrong value' -> `a wrong value'

- clarified that InBadChecksum can also serve as indicator of
path bit error rate

- Nit: `which' -> `that' in `a setting may then ...' and `A
configuration error may ...'

- removed traces of InBadCoverage (see below)

o [Section 3](#) (the MIB):

- updated LAST-UPDATED and REVISION to match the revision date

- updated the Copyright in DESCRIPTION from 2006 to 2007

- converted all error counters to Counter32, following IETF
input for rev-01

- added a required IMPORTS statement for Counter32 to UDPLITE-
MIB DEFINITIONS

- demoted the following error counters from Counter64 to
Counter32: udpliteNoPorts, udpliteInErrors,
updliteInBadChecksum, udpliteEndpointViolCoverage

- removed second, redundant MIB-2 number of udplite, following
suggestion by Bill Fenner.

- updated warning about maximum number of sub-identifiers in
udpliteEndpointEntry, to reflect new structure

- udplite now comes as entry number 1 in the udpliteMIB tree

- accordingly changed the identifier of udpliteMIBConformance
to 2

            - updated all RFC editor notes with regard to the assignment of
            mib-2 identifier numbers (now only a single one required)

            - updated section `IANA Considerations' with regard to this
            change

            - removed the InBadCoverage counter since it is subsumed by
            InBadChecksum (see below)

            - updated the following identifier numbers for consistency
            after removal: udpliteInBadChecksum (6 -> 5),
            udpliteOutDatagrams (7 -> 6), udpliteOutPartialCov (8 -> 7),
            udpliteEndpointTable (9 -> 8)

            - removed udpliteInBadCoverage from udplitePartialCsumGroup

            - extended the definition of InBadChecksum to include invalid
            checksum coverage

            - added RFC 3828 references for udpliteEndpointMinCoverage and
            udpliteEndpointViolCoverage

            - updated the description of udpliteEndpointInstance with
            respect to draft-persson-v6ops-mib-issue-01.txt, section 3.2

    o Section 7 (References):

            - RFC 4113 (UDP MIB) becomes informative, not normative

    o Note on the removal of InBadCoverage: Following rev-01, this
        counter has been removed since - with the exception of obviously
        buggy UDP-Lite stacks - it is subsumed by InBadChecksum. An
        illegal checksum value which is not the cause of a buggy
        implementation will in practically all cases lead to an incorrect
        checksum value, so that one counter implies information inherent
        in the other.

    ====> END OF NOTE TO THE RFC EDITOR <====

7. References

7.1. Normative References

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

[RFC2578]   McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J.,
            Rose, M., and S. Waldbusser, "Structure of Management
            Information Version 2 (SMIv2)", STD 58, RFC 2578,
            April 1999.

[RFC2579]   McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J.,
            Rose, M., and S. Waldbusser, "Textual Conventions for
            SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580]   McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J.,
            Rose, M., and S. Waldbusser, "Conformance Statements for
            SMIv2", STD 58, RFC 2580, April 1999.

[RFC3828]   Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., and
            G. Fairhurst, "The Lightweight User Datagram Protocol
            (UDP-Lite)", RFC 3828, July 2004.

[RFC4001]   Daniele, M., Haberman, B., Routhier, S., and J.
            Schoenwaelder, "Textual Conventions for Internet Network
            Addresses", RFC 4001, February 2005.

7.2. Informative References

[CASE]   Case, J. and C. Partridge, "Case Diagrams: A First Step to
            Diagrammed Management Information Bases", ACM Computer
            Communications Review, 19(1):13-16, January 1989.

[RFC0768]   Postel, J., "User Datagram Protocol", STD 6, RFC 768,
            August 1980.

[RFC2287]   Krupczak, C. and J. Saperia, "Definitions of System-Level
            Managed Objects for Applications", RFC 2287,
            February 1998.

[RFC2790]   Waldbusser, S. and P. Grillo, "Host Resources MIB",
            RFC 2790, March 2000.

[RFC3410]   Case, J., Mundy, R., Partain, D., and B. Stewart,
            "Introduction and Applicability Statements for Internet-
            Standard Management Framework", RFC 3410, December 2002.

   [RFC4113] Fenner, B. and J. Flick, "Management Information Base for
              the User Datagram Protocol (UDP)", RFC 4113, June 2005.

   [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram
              Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

Authors' Addresses

    Gerrit Renker
    University of Aberdeen
    School of Engineering
    Fraser Noble Building
    Aberdeen AB24 3UE
    Scotland

    Email: gerrit@erg.abdn.ac.uk
    URI: http://www.erg.abdn.ac.uk

    Godred Fairhurst
    University of Aberdeen
    School of Engineering
    Fraser Noble Building
    Aberdeen AB24 3UE
    Scotland

    Email: gorry@erg.abdn.ac.uk
    URI: http://www.erg.abdn.ac.uk

Full Copyright Statement

Intellectual Property

Acknowledgment