MIB for the UDP-Lite protocol

draft-renker-tsvwg-udplite-mib-00

Presentation to TSV WG

Gorry Fairhurst & Gerrit Renker

University of Aberdeen

UDP-Lite Background [RFC 3828]

- Main difference to RFC 768: *partial checksums*
  - *Serves applications in bit-error-prone environments*
  - *UDP discards even when irrelevant parts corrupt*
  - *Length* field re-interpreted as *Checksum Coverage*
  - *To benefit, needs special Link drivers*
- *Application areas:*
  - *wireless, mobile, 3G/4G multimedia*
  - *VoIP, TVoIP, video streaming*
  - *video / audio conferencing*
  - *multicast-enabled (contrast to DCCP / RFC 4340)*

UDP-Lite Implementation Status

- Linux kernel submission for rel. 2.6.20
- A working *stack* with applications for over 1 year!
  - fully extended to UDP-Litev4/v6
  - code shared/integrated with UDP
  - basic NAT/Firewall/XFRM (IPsec) support
- *Applications*:
  - VLC, ttcp, Ethereal (supports v4/v6 UDP-Lite)
  - easy porting from UDP:
    - socket(s, SOCK_DGRAM, IPPROTO_UDPLITE);
    - setsockopt(s, ..., UDPLITE_SEND_CSCOV, &CsCov, ...);

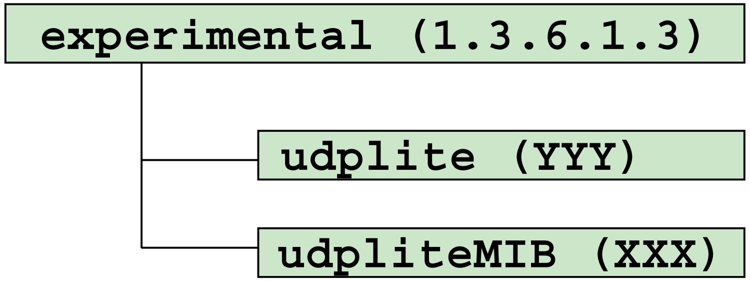
Contents of the MIB

- Shares v4/v6 basics with UDP-MIB (RFC 4113):
  - **InDatagrams, NoPorts, InErrors, OutDatagrams**
  - But: UDP-Lite has no special high-capacity counters

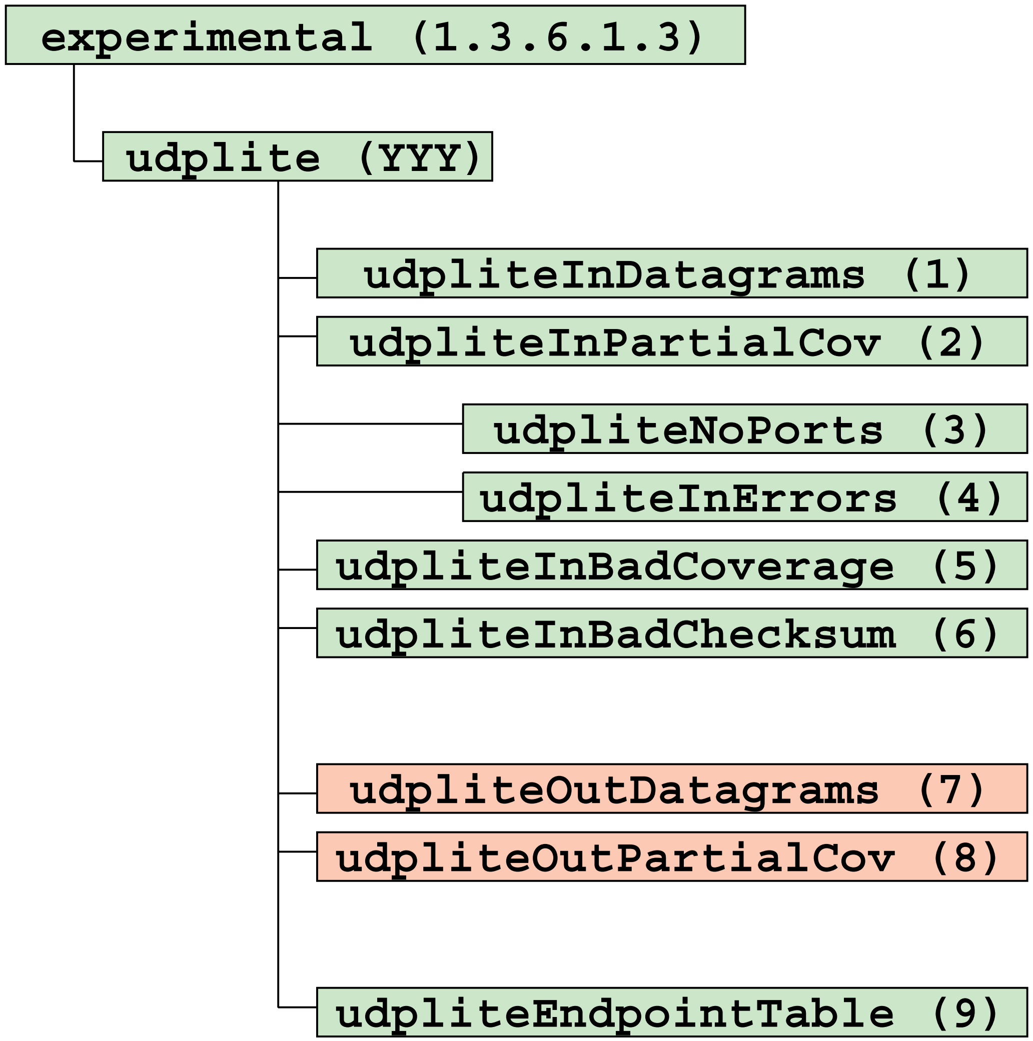
- **New** in UDP-Lite MIB:
  - **InPartialCov** – *InDatagram* with partial coverage
  - **InBadCoverage** – *InError* with bad coverage value
  - **InBadChecksum** – *InError* due to failed checksum
  - **OutPartialCov** – *OutDatagram* with partial coverage
  - A new **endpoint table** - More later

UDP-Lite MIB Components

- **udplite**: MIB proper (YYY - to be assigned)
- **udpliteMIB**: compliance statements (XXX)



Global components: 32bit Counters


experimental (1.3.6.1.3)

udplite (YYY)

udpliteInDatagrams (1)

udpliteInPartialCov (2)

udpliteNoPorts (3)

udpliteInErrors (4)

udpliteInBadCoverage (5)

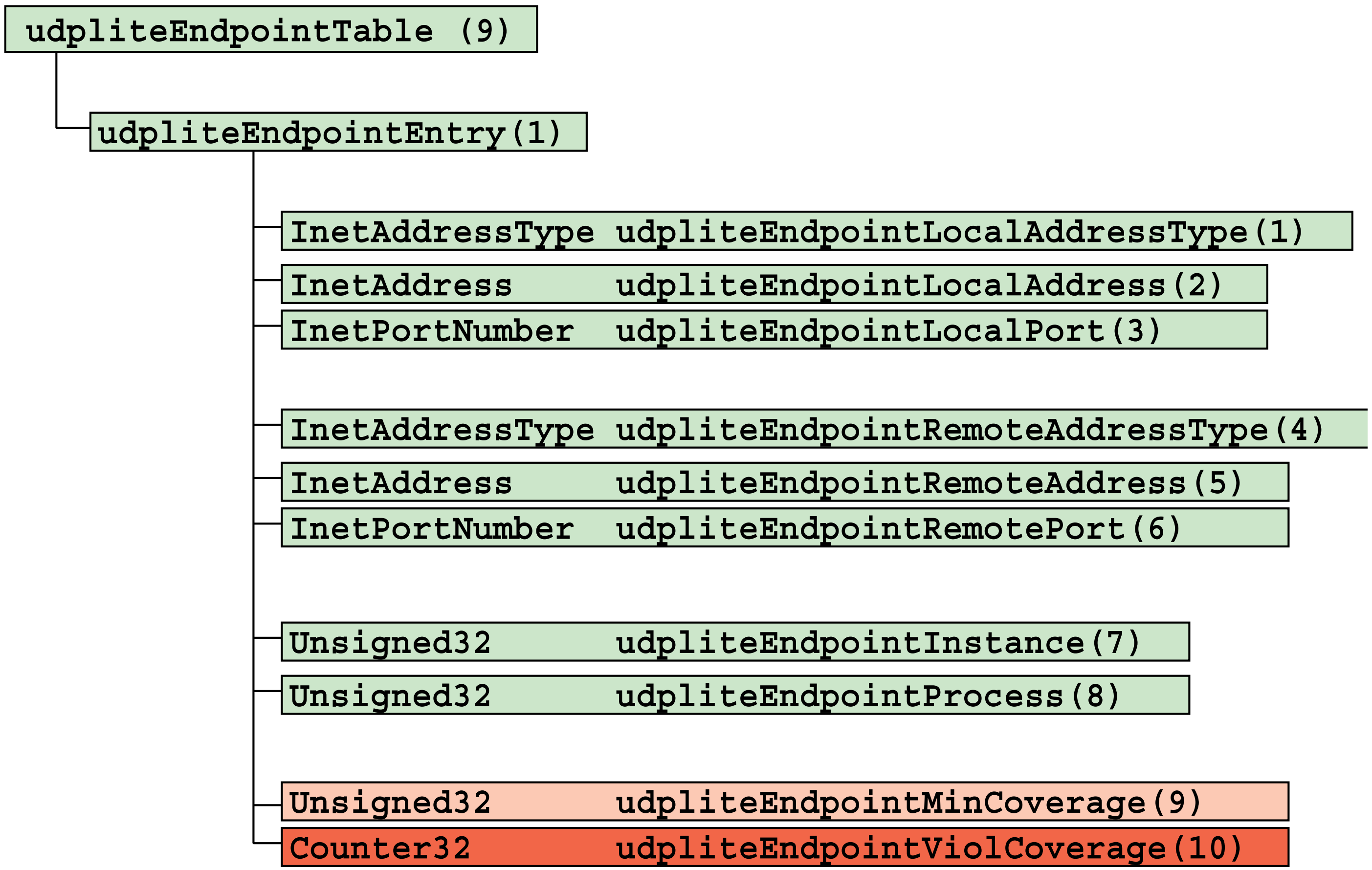
udpliteInBadChecksum (6)

udpliteOutDatagrams (7)

udpliteOutPartialCov (8)

udpliteEndpointTable (9)

udpliteEndpointTable



udpliteEndpointTable (9)

udpliteEndpointEntry(1)

InetAddressType udpliteEndpointLocalAddressType(1)

InetAddress udpliteEndpointLocalAddress(2)

InetPortNumber udpliteEndpointLocalPort(3)

InetAddressType udpliteEndpointRemoteAddressType(4)

InetAddress udpliteEndpointRemoteAddress(5)

InetPortNumber udpliteEndpointRemotePort(6)

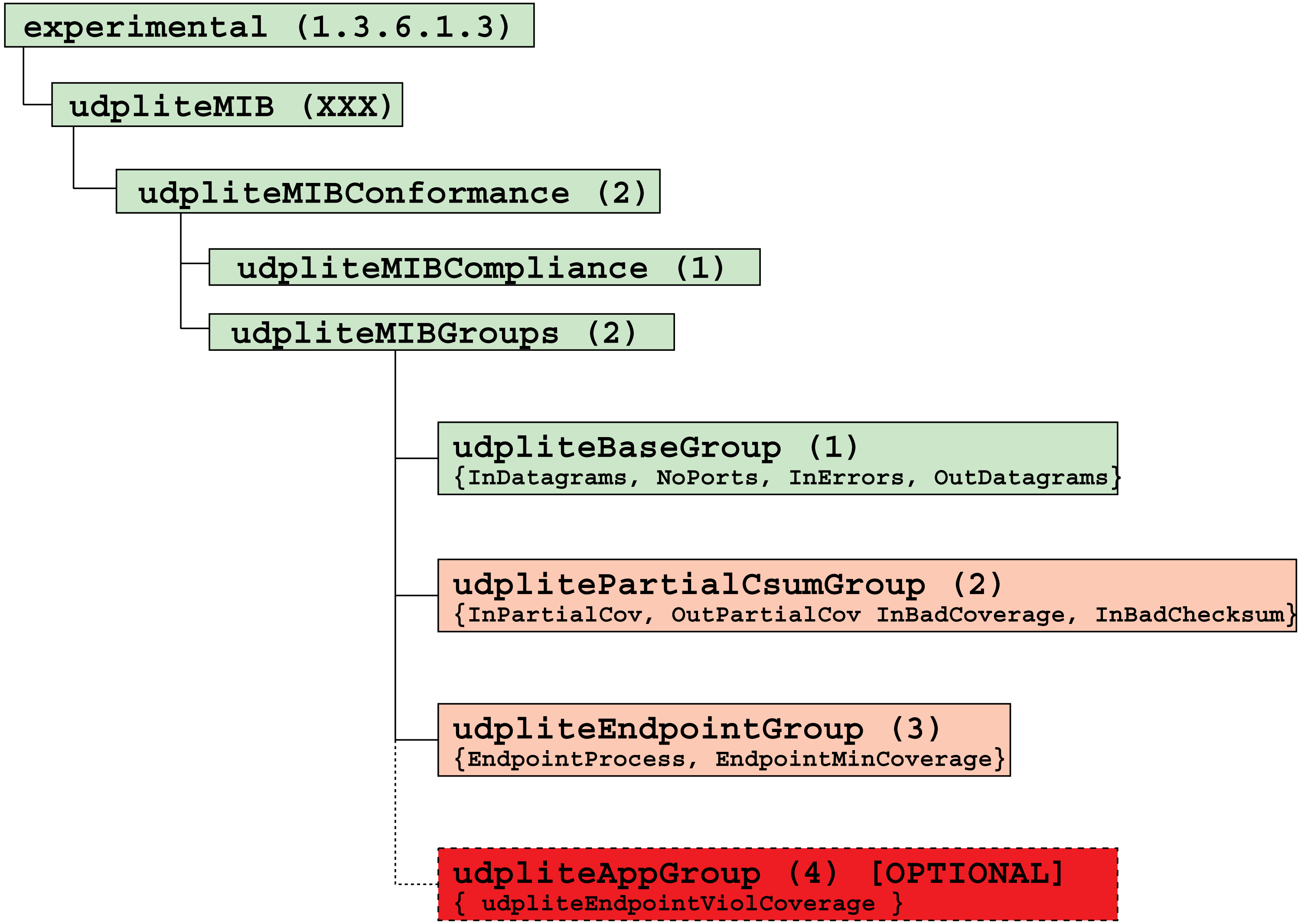
Unsigned32 udpliteEndpointInstance(7)

Unsigned32 udpliteEndpointProcess(8)

Unsigned32 udpliteEndpointMinCoverage(9)

Counter32 udpliteEndpointViolCoverage(10)

Conformance Statements



experimental (1.3.6.1.3)

udpliteMIB (XXX)

udpliteMIBConformance (2)

udpliteMIBCompliance (1)

udpliteMIBGroups (2)

udpliteBaseGroup (1)
{InDatagrams, NoPorts, InErrors, OutDatagrams}

udplitePartialCsumGroup (2)
{InPartialCov, OutPartialCov InBadCoverage, InBadChecksum}

udpliteEndpointGroup (3)
{EndpointProcess, EndpointMinCoverage}

udpliteAppGroup (4) [OPTIONAL]
{ udpliteEndpointViolCoverage }

Proposed Changes for rev -01

* "always below the coverage of incoming xdatagrams)." ==> `incoming datagrams'
* InBadChecksum was missing in Section 1.1, in the paragraph

> "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). These errors are also ..."

==> prepended the missing `InBadChecksum: ’

* updated email address to gerrit@erg.abdn.ac.uk after Lars' email

* changed

```
    udpliteMIBConformance OBJECT IDENTIFIER ::= { udpliteMIB 2 }
```

to read

```
    udpliteMIBConformance OBJECT IDENTIFIER ::= { udpliteMIB 1 }
```

===> Reason: udpliteMIBConformance was the only sub-element of udpliteMIB
May be something needing discussion when the draft is processed further.

More…?

Conclusions & Further Work

- Would be good to involve other opinions/ideas!
- We’d like to know from people with MIB experience....
- Any wisdom about 32 bit vs 64 bit counters?
  - 32 bit chosen as more widely available…
  - But: what about Gigabit speeds?

- Can we make this a tsvwg work item?