Consideration of LLDPv2

Paul Congdon
Paul Unbehagen
Why do we need to update LLDP?

- LLDP is widely deployed in many environments
- The number of TLVs sent in LLDPDUs continues to grow
  - New standards continue to defined new objects
  - A large number of Vendor Specific TLVs
- Alternative protocols are being proposed to get around the single PDU size limit
- Relying on Jumbo frames to support more TLVs is problematic in many environments
- Summary: We need to be able to exchange more TLVs.
Objectives for a new version

• Support the ability to send more than 1 PDUs worth of TLVs
• Support the ability to communicate with an LLDPv1 implementation (only the first PDUs worth of TLVs).
• Ensure the integrity of the full set of TLVs is received by partner
  • NOTE: This can be useful in v1 implementations as well
• Consider if there are other optimizations to address
  • E.g. Less frequent updates
  • E.g. New reachability addresses (Nearest-station or Nearest-Router)
Current LLDP operation reminder

NOTE: Think of the Remote and Local MIBs as a database that must fit into a single PDU
Replace all values of the Remote MIB with contents of LLDPDU when something changes
Proposal

- Define a new mandatory (for v2 implementations) TLV that appears just after the current mandatory set of 3 TLVs.
  - ChassisID TLV + PortID TLV + TTL TLV + (new) ExtensionPDU TLV

- In the new TLV, define a vector that specifies:
  - The number of extension PDUs to be sent
  - An identity of each PDU (e.g. hash, checksum, version number or PDU number)
  - Acknowledges the receipt of partner extension PDUs

- The first v2 PDU looks like a standard v1 PDU with the extra ExtensionPDU TLV (i.e. will be received by v1 implementations).

- The new extension PDUs need to be ignored by v1 LLDP in a non-intrusive way. Options:
  - Force an error in the v2 PDUs – will cause error counters to increment
  - Use a new Ethertype for v2 extension PDUs - preferred

- The new PDUs need to have a mandatory format as well:
  - Includes at least the first two mandatory TLVs of a v1 PDU (ChassisID + PortID)
  - Includes new TLV that identifies the extension PDU.

- Optimizations:
  - There is no need to resend extension PDUs if nothing has changed.
  - Only periodically send the 1st PDU.
  - TTL in 1st PDU relates to all extension PDUs.

- NOTE: The maximum size of a TLV defines the maximum number of extension PDUs that can be included. (depends on identity field)
NOTE: Send primary LLDPDU and all extension PDU when something changes locally
If extension data has NOT changed, no need to send anything other than primary LLDPDU
## Example Extension TLV

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>(n+m)*16 + 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLV type = Y</td>
<td>TLV information string</td>
<td>Number of Tx extension PDUs</td>
<td>Number of Rx extension PDUs</td>
<td>n MD5 sums</td>
</tr>
<tr>
<td>(7 bits)</td>
<td>(9 bits)</td>
<td>(4 bits)</td>
<td>(4 bits)</td>
<td>of Tx extension PDUs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(0 &lt;= n &lt; 16)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>m MD5 sums</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of Rx extension PDUs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(0 &lt;= m &lt; 16)</td>
</tr>
</tbody>
</table>

- **TLV Type**
  - probably use the next reserved type (i.e. 9)
- **Number of Tx and Rx extension PDUs**
  - If using MD5 Sum of 16 bytes, can only pack 30 sums into a TLV
- **MD5 Sums**
  - Should cover the entire extension LLDPDU
### Example Extension PDU

<table>
<thead>
<tr>
<th>DA</th>
<th>SA</th>
<th>LLDP Extension Ethertype</th>
<th>Chassis ID TLV</th>
<th>Port ID TLV</th>
<th>Extension Desc TLV</th>
<th>Optional TLV</th>
<th>...</th>
<th>Optional TLV</th>
<th>End of LLDPDU TLV</th>
</tr>
</thead>
</table>

- **LLDP Extension Ethertype**
  - New Ethertype allows LLDPv1 implementations to ignore these frames

- **Chassis ID + Port ID are mandatory**
  - Note TTL from 1st PDU should apply and is not needed here

- **Extension Description TLV**
  - Numbers the extension PDU in the sequence of all extension PDUs
**Example Extension Description TLV**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLV type = Y (7 bits)</td>
<td>TLV information string length (9 bits)</td>
<td>PDU Number (4 bits)</td>
<td>Max PDU Number (4 bits)</td>
</tr>
</tbody>
</table>

- **TLV Type**
  - Another new base TLV type (i.e. 10)
- **PDU Number and Max PDU Number**
  - For example PDU 1 of 5

- **TLV header**

- **TLV information string**
Questions / comments / TBDs

• How to define the extension PDU TLV?
  • It needs to contain a vector of information for all extension PDUs
  • It needs to acknowledge received extension PDUs.
  • The smaller the identity field, the more extension PDUs that can be supported (e.g. CRC-16 or MD5 Hash?)
  • We could define two extension TLVs – one for Tx and one for Rx to support more extension PDUs
  • Should the MD5 Hash cover all PDUs or individual?

• When to send the 1st PDU as an ACK of received extension PDUs?
  • Need a final bit in the extension PDUs or a PDU number scheme?
  • Define another End of LLDPDU TLV?

• Retransmission strategy? SACK or just retransmit the entire sequence?