802.1ABdh – Multi-Frame LLDP

Overview and Questions

Version 0

Stephen Haddock

July 13, 2020

802.1ABdh Motivation

- LLDP currently limited to conveying information that can be contained in a single frame.
  - InfoSource sends a single frame containing a LLDPDU with all TLVs.
  - InfoTarget records received TLVs in a "remote MIB" corresponding to the transmitter.
    - InfoTarget deletes any information received in a TLV in a previous LLDPDU if that TLV is not present in the most recent LLDPDU.
- Amendment increases the limit on the information that can be conveyed by allowing multiple frames containing TLVs.
  - TLVs themselves must still fit within a single frame.
  - Total amount of information still limited (will discuss later).
- Based on proposal by Paul Bottorff and Paul Congdon
  - http://www.ieee802.org/1/files/public/docs2020/dh-bottorff-xlldp-0320-v03.pdf

Basic (proposed) mechanism

1. InfoSource sends a normal LLDPDU containing a new TLV:
   - “Manifest” TLV contains a description of one or more extension PDUs, each of which can contain one or more TLVs.
     - Assuming for now that the Manifest contains a complete list of all X-PDUs, not just those that have changed.
   - If InfoTarget does not implement multi-frame it will record the Manifest TLV but take no further action (normal behavior for an unrecognized TLV).
   - If InfoTarget implements multi-frame it will record the Manifest TLV and begin requesting extension PDUs from InfoSource.
2. InfoTarget sends a request for an extension PDU (XREQ-PDU) to the InfoSource (unicast).
   - Only a single XREQ-PDU can be outstanding at any time, but a single XREQ-PDU can request extension PDUs.
3. InfoSource sends the requested extension PDUs (X-PDU) to the InfoTarget (unicast).

Interoperability with “legacy” LLDP implementations

- Legacy implementations, acting as an InfoSource, send a normal LLDPDU to InfoTarget(s).
  - Sent multicast, since there may be multiple connected InfoTargets.
  - Each InfoTarget receives and records normal LLDPDU, whether or not the InfoTarget implements multi-frame.
- Multi-frame implementations, acting as an InfoSource, send a normal LLDPDU containing a Manifest TLV to InfoTarget(s).
  - InfoSource does not need to know if InfoTarget is a legacy implementation.
  - If InfoTarget is a legacy implementation it will record the Manifest TLV but take no further action.
- Legacy implementations never receive an XREQ-PDU because they do not send an LLDPDU containing a Manifest TLV.
- Legacy implementations never receive an X-PDU because they do not send XREQ-PDUs.
- If, perchance, a legacy implementation does receive an XREQ-PDU or X-PDU, want to make sure it is discarded.

Identifying XLLDP Frames

- Current proposal suggests using a new Ethertype (with sub-type and revision number) to identify frames containing XREQ-PDU and X-PDU.
  - Not clear this is necessary.
- Current LLDP specification requires discarding any LLDPDU that does not have a Chassis ID TLV as the first TLV.
- Both the XREQ-PDU and X-PDU contain a new TLV (XREQ-TLV and XID-TLV respectively). Putting this as the first TLV would be sufficient to identify the PDU and ensure that it is discarded by legacy implementations.
  - Allows using the current LLDP Ethertype for XREQ-PDU and X-PDU.
  - Legacy implementations would increment error counters when discard an XREQ-PDU or X-PDU, but that is OK and even desirable since a legacy implementation should never receive them when the protocol is properly implemented.
- Is there some other reason for using a new Ethertype that I am missing?

Information size limit

- In legacy implementations the maximum amount of information that can be conveyed by LLDP is limited by the number and size of TLVs that can be included in a single frame.
  - Depends on the media connecting the LLDP systems (which could be a virtual connection).
  - May be further constrained by the application environment (e.g. TSN).
  - How does the implementation know the MTU?
- In a multi-frame implementation the maximum amount of information is limited by the number of X-PDU identifiers in that can fit in a Manifest TLV (and in a single frame).
  - Maximum size of a TLV information string (511 octets) limits the number of X-PDU identifiers to 84. This seems like plenty, but it may be further constrained by other considerations:
    - Frame size limitations (as above).
    - Other TLVs that need to be in the “normal” LLDPDU to ensure receipt by legacy implementations.
- Could eliminate the information size limit by allowing X-PDUs to contain a Manifest TLV (that identifies additional X-PDUs).

Database updates

- Legacy implementations always do full database updates.
  - InfoSource sends single LLDPDU that includes all TLVs (full database).
  - InfoTarget validates all TLVs, and then sends all together to update the remote MIB (atomic operation).
  - If an LLDPDU contains a TLV that is not valid, or does not contain a TLV that was present in a previous LLDPDU, that TLV information is deleted from the remote MIB.
- For multi-frame, want to be able to do incremental updates.
  - InfoSource sends normal LLDPDU with a Manifest TLV containing descriptors of all X-PDUs, even those with TLVs that have not changed.
  - InfoTarget sends XREQ-PDU for any X-PDUs whose descriptors in the most recently received Manifest are different from X-PDUs already received.
    - The remote MIB is updated with TLV information received in changed X-PDUs when all X-PDUs in the most recently received Manifest have been received, so update is still atomic.
    - If going to retain the ability to delete a TLV from the remote MIB, then need a way to distinguish between a TLV that is no longer relevant versus a TLV that has not changed.

Deleting TLV info from remote MIB

1. Implicit delete
   - When InfoTarget receives a Manifest TLV and the subsequent X-PDUs, the InfoTarget retains a list of the TLVs contained in each X-PDU.
   - InfoSource sends a new Manifest TLV containing a modified X-PDU descriptor for the X-PDU that no longer contains the TLV to be deleted.
   - InfoTarget requests and receives the modified X-PDU, and by comparing with the retained list of expected TLVs, the InfoTarget can identify the TLV to be deleted.
2. Explicit delete
   - The InfoSource sends a new Manifest TLV containing a modified X-PDU descriptor for the X-PDU containing the TLV to be deleted.
   - Upon request from the InfoTarget, the InfoSource sends the modified X-PDU that includes an invalid TLV for the TLV to be deleted (e.g. infostring length set to zero).
   - InfoTarget detects the invalid TLV and deletes it from the remote MIB.
   - InfoSource needs to include the invalid TLV in each new revision of the X-PDU because it never gets a confirmation that the InfoTarget received the X-PDU.
3. Re-initialize entire remote MIB
   - The InfoSource increments the revision number of all X-PDUs, regardless of whether there is any change to the included TLVs, and sends a new Manifest TLV.
   - InfoTarget sends requests for all X-PDUs.
   - When InfoTarget detects that it has new revisions of every X-PDU it does a complete update of the remote MIB, including deleting any old information from a TLV no longer present in any X-PDU.

Fixed mapping of TLVs to X-PDUs?

- Presumably it would be desirable to impose as few constraints as possible on how the InfoSource maps TLVs to X-PDUs.
  - Ideally the InfoSource would be free to change the mapping of TLVs to X-PDUs over time.
    - Especially if the length of data contained in a TLV can vary due to changes in operating conditions.
- I think all that is necessary for the InfoSource to move a TLV from one X-PDU to another is to send a new Manifest with an updated X-PDU descriptor for both TLVs.
  - Am I missing something here?

Thank You