Problem with 802.1ASds and Multiple Time Domains

Adriaan Niess, XC/EYA2, 2024-03-01
802.1ASds Fails with Multiple Time Domains

Problem Description

- Example Scenario shows if primary GM crashes on HDE link, it can also prevent the secondary GM from doing a failover. Essentially the secondary GM gets *deadlocked*.
- In general the problem can occur whenever a gPTP node is time receiver and time transmitter at the same time on different domains.
802.1ASds Failover Bug
Example Szenario (1)

- A is primary GM (time transmitter for domain 1)
- B is secondary GM (time receiver on domain 1, time transmitter for domain 2)
802.1ASds Failover Bug
Example Szenario (2)

- Let's assume all nodes connected to bus are synchronized
- Note: There is one instance of asCapableAcrossDomains defined per Link Port (not PTP Port)
  - Or if no CMLDS is used there is a single instance of asCapableAcrossDomains across all domains
- Initial state of B is shown below

```
  asCapableAcrossDomains = True
  asCapable[domain=2] = True
```
802.1ASds Failover Bug
Example Szenario (3)

- Now the primary GM crashes
- Domain 1 is now offline!

\[
\text{asCapableAcrossDomains} = \text{True}
\]
\[
\text{asCapable[domain=2]} = \text{True}
\]
802.1ASds Failover Bug

Example Szenario (4)

- At some point B will send a PDelayRequest to A (for simplicity I draw the message as unicast)
- A does not answer with PDelayResponse
- After PDelayIntervalTimer elapses, B sets asCapableAcrossDomains=False in MDPDelayReq state machine
- On B: asCapableAcrossDomains=False implies asCapable=False (for all domains)

- **Domain 2 is now offline too!** Because B‘s MDSyncSendSM won't send Sync frames if asCapable is False!
- The use of asCapableAcrossDomains does not make sense for HDE links!
Possible Solution

- If only the instance specific Pdelay mechanism is used for HDE links anyway:
  - I suggest removing the variable asCapableAcrossDomains altogether (from chapter 19)

- MDPdelayReq state machine from chapter 11 could be made more generic to still be reused in chapter 19 by introducing a function setAsCapable() which could behave differently depending if instance-specific mechanism or CMLDS is used.
  - Whenever instance specific mechanism is used we can set asCapable directly without even considering the value of asCapableAcrossDomains.