802.1ASbt presentation on Pdelay_Req storm issue

Bob Noseworthy (ren@iol.unh.edu)
University of New Hampshire’s InterOperability Laboratory (UNH-IOL)
IEEE 802.1 TSN TG
2014.05.14
Introduction

• Faulty Pdelay Responses (Response or Response Follow Ups) trigger immediate retransmission of Pdelay Req
  – For persistent faults, this triggers a ‘high’ rate of Pdelay messages (as fast as responses come back)
  – This may burden a Bridge connected to both proper and faulty link partners
The MDPdelayReq State Machine (per 802.1AS-cor)

- Focus is on the transitions to the Reset state from:
  - WAITING_FOR_PDELA Y_RESP
- And
  - WAITING_FOR_PDELA Y_RESP_FOLLOW_UP
RESET to PDELAY_REQ states

The following as tutorial:

- Upon entering the RESET state, the state actions are executed and then immediately (unconditionally ‘UCT’) transition to SEND_PDELAY_REQ
- A new Pdelay_Req frame is sent immediately upon entering SEND_PDELAY_REQ
WAITING_FOR_PDELAY_RESP to RESET

\[
\text{(currentTime} - \text{pdelayIntervalTimer} >= \text{pdelayReqInterval}) || \\
(\text{rcvdPdelayResp} && \\
( \text{rcvdPdelayRespPtr->requestingPortIdentity.clockIdentity != thisClock} ) || \\
( \text{rcvdPdelayRespPtr->requestingPortIdentity.portNumber != thisPort} ) || \\
( \text{rcvdPdelayRespPtr->sequenceId != txPdelayReqPtr->sequenceId} ) ) 
\]

- Timeout occurs so go to reset (this is fine); or,
- \text{rcvdPdelayResp} && not thisClock, not thisPort, or \text{rcvdPdelayRespPtr->sequenceId != txPdelayReqPtr->sequenceId}

- In this latter case, as soon as an errored Pdelay_Resp is received, a fresh Pdelay_Req frame will be sent, potentially triggering a storm of invalid Pdelay_Resp and Pdelay_Req retransmissions
- Transition to WAITING_FOR_PDELAY_RESP_FOLLOW_UP is conditioned on rcvdPdelayResp with proper thisClock, thisPort and seq. Id.
WAITING_FOR_PDELAY_RESP_FOLLOW_UP to RESET

\[
\text{(currentTime - pdelayIntervalTimer} \geq \text{pdelayReqInterval)} \text{ || (rcvdPdelayResp && (rcvdPdelayRespPtr->sequenceId == txPdelayReqPtr->sequenceId))}
\]

- Timeout occurs so go to reset (this is fine); or,
- \((\text{rcvdPdelayResp && (rcvdPdelayRespPtr->sequenceId == txPdelayReqPtr->sequenceId)})\)
  - Causes transition to RESET if a 2nd Pdelay_Resp is received with same Seq.Id

- In this latter case, as soon as an errored Duplicate Pdelay_Resp is received, a fresh Pdelay_Req frame will be sent, potentially \text{triggering a storm of invalid Pdelay_Resp and Pdelay_Req retransmissions}

- Transition to WAITING_FOR_PDELAY_INTERVAL_TIMER is conditioned on rcvdPdelayRespFollowUp with proper sourcePortIdentity (thisClock, thisPort) and seq. Id (if not received, then the timeout condition will take us back to RESET)
Proposed solution

• **Option 1:** Change the transition from RESET to SEND_PDELAY_REQ from “UCT” to “currentTime – pdelayIntervalTimer >= pdelayReqInterval”
  – In the event that a timeout occurred (causing the transition to RESET) then this condition will already be true (allowing for immediate transition to SEND_PDELAY_REQ)
  – In the case of an errored frame, this will cause a delay before the next Pdelay_Req is sent

• **Option 2:** Make all transitions INTO the RESET state conditioned only on “currentTime – pdelayIntervalTimer >= pdelayReqInterval”

• Option 1 is preferred (see next slide)
What happens in the RESET state

<table>
<thead>
<tr>
<th>RESET</th>
</tr>
</thead>
<tbody>
<tr>
<td>initPdelayRespReceived = FALSE;</td>
</tr>
<tr>
<td>rcvdPdelayResp = FALSE;</td>
</tr>
<tr>
<td>if (lostResponses &lt;= allowedLostResponses)</td>
</tr>
<tr>
<td>lostResponses += 1;</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>isMeasuringDelay = FALSE;</td>
</tr>
<tr>
<td>asCapable = FALSE;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

• Option 1 is preferred, as Option 2 will delay setting asCapable to FALSE upon receipt of some errored Pdelay exchanges
  – note, ‘today’ there is still a delay in some error cases, such as a Pdelay_Resp_Follow_Up with improper Seq.Id
Summary

• Faulty responses trigger flood of frames on a link at the rate of the Pdelay responses
• To introduce delay and force Pdelay_Req messages to be paced at their intended rate, change either:
  – the transition from RESET to SEND_PDELAY_REQ from “UCT” to “currentTime – pdelayIntervalTimer >= pdelayReqInterval”
  Or,
  – remove all conditions on the transitions into Reset leaving only “currentTime – pdelayIntervalTimer >= pdelayReqInterval”