pDelay and Other AS Cor Issues

v1 – March, 2013

802.1 TSN Face to Face – Orlando, FL

Don Pannell - dpannell@marvell.com

Issues

- **Multiple people have made various comments in the last AS Cor Sponsor ballot in the area of pDelay Response and Residency Time maximum values**
- **This presentation summarizes the issues & gives more background information so an informed decision can be made**
- **In Summary the problem & proposed solutions are:**
  - pDelay Response time should be increased from 10 to 125 mSec
  - Residency Time should be increased from 10 to 125 mSec
  - Increase the number of Lost pDelay Responses before asCapable is dropped from 3 to 10
  - Define what a Late pDelay Response is and what to do with them

pDelay Response Issue

pDelay Response – History

- **An original goal of AVB was to be low cost (its roots are from the Residential Ethernet Study group)**
  - For IEEE 802.1AS that meant a goal to run it on an 8051 CPU!
- **Originally AS specified a 1 mSec pDelay Response time**
  - Near AS's completion this was change to 10 mSec as concern grew about meeting the 1 mSec number
  - Simulations were run showing negligible clock accuracy impact between 1, 10 or even 50 mSec pDelay Response times
- **Now it is clear that 10 mSec can not be met 100% of the time cost effectively**
  - It has been seen in testing where asCapable is dropped on a port due to this reason and all streams through that port stop
  - This is against the high reliability/availabilty goals AVB

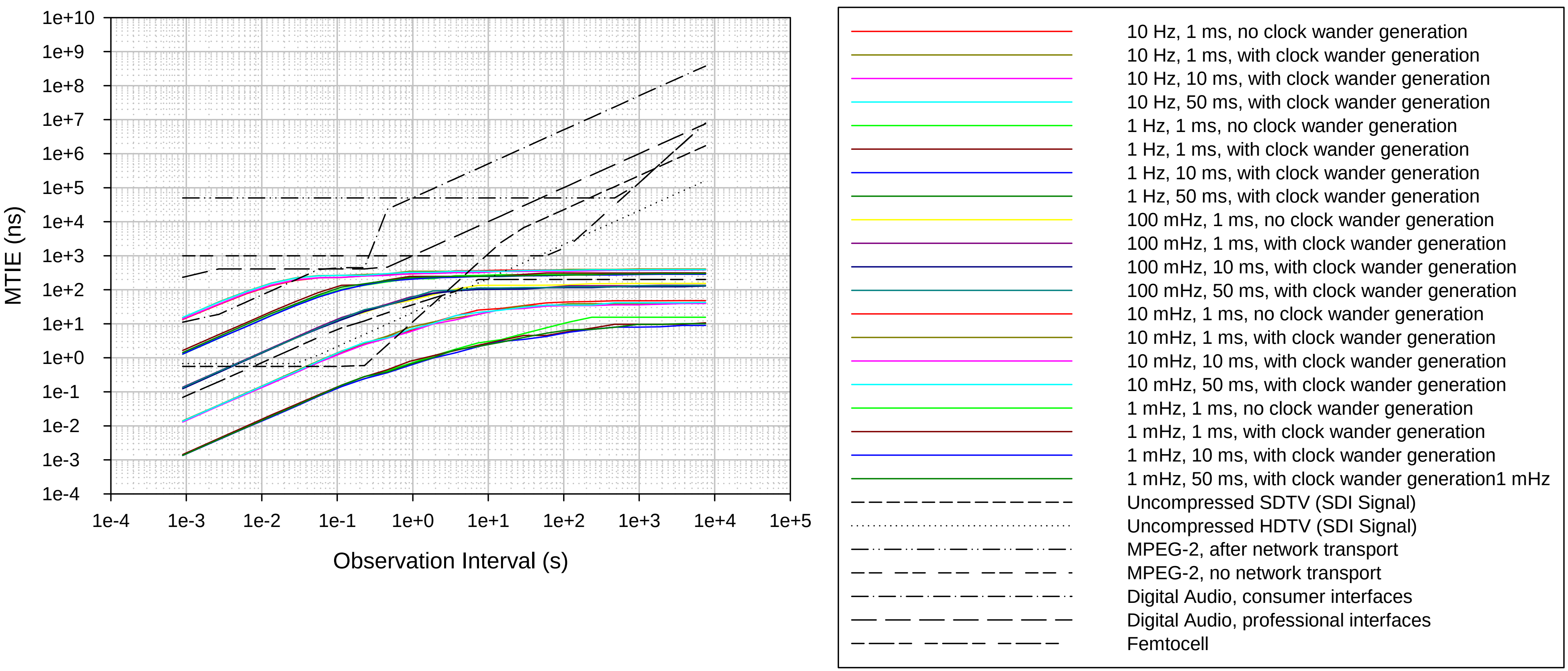
pDelay Response – New Information

- **To get a product to pass the 10 mSec requirement the pDelay Response code needs to be in the kernel**
  - Doable, but harder to support
- **Even then, with 3 hours of testing on a 24 port bridge the 10 mSec number could not be met 100% of the time**
  - What about a 48 port or 256 port or larger bridge?
- **Windows uses a 10 mSec task clock**
  - So a Windows based end node can never meet a 10 mSec Response time!
  - Not supporting Windows based end nodes would be against the Broad Market Potential & consumer goal that was in the AVB PARs

pDelay Response – Simulation Data

- **From as-garner-simulation-results-wander-gen-new-res-time-0710**

Comparison of jitter/wander accumulation MTIE at time-aware system (node) 8
10 Hz, 1 Hz, 100 mHz, 10 mHz, and 1 mHz endpoint filter bandwidths
1, 10, 50 ms residence time and Pdelay turnaround time (with clock wander generation)
1 ms residence time and Pdelay turnaround time (without clock wander generation
Sync Interval = 0.125 s
Pdelay Interval = 1.0 s



pDelay Response – The Math

- Why does the response time have so little effect?
- The response time error comes if the clock rate changes between the two response time time stamps
- During 125 mSec period a 1 ppm error is 0.000000125 or 0.125 ns (1/1,000,000 * 0.125 seconds)
- Assuming the clock changes at the past assumed maximum rate of 4 ppm / Sec, it would change a max of ½ a ppm during 125 mSec (1 Sec / 8 = 125 mSec)
- So the error due to a 125 mSec pDelay Response time is less ½ * 0.125 ns → Much less than the 40ns or 8ns time stamp resolution

pDelay Response – The Proposed Change

- **Current pDelay Response time max is 10 mSec**
- **Propose change this to 125 mSec or before the next pDelay Request whichever is smaller**
  - A link's gPTP slave port can request the link's master port to change its pDelay Request rate – so this needs to track
- **A max of 125 mSec is explained next**

pDelay Response – The Changes Impact

- Before the proposed solution of 125 mSec was commented on, questions were asked of existing users of gPTP if this would be a problem
- All but one vendor could handle calculations up until the next pDelay Request (up to 1 Sec or more)
- The one vendor could support up to ~130 mSec with changes
- So 125 mSec is proposed

- Again this is needed for large bridges and broad market potential end nodes

Residency Time Issue

Residency Time – History

- While Residency Time issues have not been seen in testing, bridges with high port counts will have the same problem meeting the current 10 mSec Residency Time requirements for the same reasons as pDelay Response
- The issues have not been seen in testing as a link partner device cannot know what the residency time of the upsteam device was
- But testing can measure this in bridges
- We already know it is likely 24 port bridges cannot guarantee 10 mSec Residency Time
- What about 256 port or more bridges?
- Is this really a required parameter for AS accuracy?

Residency Time – Simulations & Math

- The priviously referenced simulations modified BOTH the pDelay Response Time and the Residency Time together at 1, 10 and 50 mSec
- The results show that these different times have little effect on the IEEE 802.1AS accuracy
- And the previously shown Math is valid for both:
- The increased error introduced by an increase in the pDelay Response or the Residency Times is far under the Time Stamp sampling error of 40ns or even 8ns

Residency Time – The Proposed Change

- **Current Residency time max is 10 mSec**
- **Propose change this to 125 mSec or before the next Sync whichever is less**
  - A link's gPTP slave port can request the link's master port to change its Sync rate – so this needs to track
- **Again this is needed for large bridges and broad market potential low cost bridges**
  - Why require a high cost CPU in a bridge just to meet the current spec when no performance gain is achieved?
  - And the CPU is probably busy with other more important work

Lost pDelay Response Issue

Lost pDelay – History

- **According to the current standard, if pDelay Response messages are Lost 3 times in a row, asCapable is dropped**
- **This was intended to detect if a new link partner supports gPTP (and AVB in general)**
- **Historically 3 has been a good number for these kinks of things**
  - A Tx CRC may occur on the 1st attempt
  - An Rx CRC may occur on the 2nd attempt
  - The odds of a third attempt failing due to a CRC are huge
- **But once a link partner is known to be asCapable (and AVB for that matter), do you want to drop it so quickly?**

Lost pDelay – Considerations

- **Consider this after streams are running on the link:**
  - A Tx CRC may occur on the 1st attempt
  - An Rx CRC may occur on the 2nd attempt
  - On the 3rd attempt the CPU is too busy so pDelay is considered lost
  - Now asCapable is dropped and all the streams are stopped!
- **I think the goal of AVB was to be more resilient that past solutions**
- **So even though the above scenario is not likely, its effect is so devastating to steams**
  - And other more likely scenarios may create the same havoc
- **This is against the high reliability goals of AVB**

Lost pDelay – The Proposed Change

- **Current count of lost pDelays to drop asCapable is 3**
- **Propose change this to 10**
  - This is a simple change
  - A better change would be to have a different limit at 1st link up when asCapable is false vs. when asCapable is true
  - i.e., make it harder to drop asCapable once it is established
- **Changing this count to 10 has little down side**
  - If a link partner crashes without a link down, then asCapable will be true for 10 seconds instead of 3, but likely no steams were getting processed by the crashed device anyway
- **The goal of AVB is to keep streams flowing!**

Late pDelay Issue

Late pDelay – History

- There is no current definition for a Late pDelay Response
- But what do you call a frame that comes latter than the pDelay Response time but before the next pDelay Request?
- One proposal is to count these as Lost
- This approach has led to the current issue of asCapable being dropped while streams are flowing
- What if a large number of pDelays are Late?

Late pDelay – The Proposed Change

- **Define Late pDelays and treat them as Lost but only if the Current count of lost pDelays to drop asCapable is changed to 10**
- **Including this definition and changing this count to 10 has little down side**
  - If a link partner crashes without a link down, then asCapable will be true for 10 seconds instead of 3, but likely no steams were getting processed by the crashed device anyway
- **The goal of AVB is to keep streams flowing!**
- **We need to decide what to do.**
- **Thanks for your time!**