60802 Time Sync – Mean Link Delay Averaging & Normative Requirements

David McCall – Intel Corporation

Version 2
References


Content

• Background
• 60802’s recommended meanLinkDelay Algorithm
  • Startup behaviour
  • Dos and Don’ts
• Implications for Simulations – Recent vs Future
  • Time Series
    • IIR Factor of 100 vs 1000
    • Settling time of 50s
    • Implications & Recommendations
  • Monte Carlo
    • Average of 50 path delay calculations
    • Implications & Recommendations
• Normative Requirements
Background

• 802.1AS-2020 assumes \texttt{meanLinkDelay} and provides some informative language about approaches to averaging but no normative requirements
  • See IEEE 802.1AS Clause 11.2.1.

• Initial Monte Carlo simulations implemented a simple “% effective”
  • 90% effective removed 90% of the error
  • Even without the algorithm, meanLinkDelay error was much smaller than other factors, and almost entirely due to Timestamp Error

• More recent Monte Carlo simulations modelled 50 previous path delay measurements (for each node) and took an average

• Latest Time Series simulations implemented the recommended algorithm (from Annex D), but used a factor of 100 vs recommendation of 1000

• Additional, focused, simulations investigating behaviour of meanLinkDelay averaging would be useful...and are provided in this contribution.
Recommended meanLinkDelay Algorithm

From IEC/IEEE 60802, Annex D
**Path Delay Measurement**

\[ m_{Path Delay} = \frac{(t_4(x) - t_1(x)) - (t_3(x) - t_2(x))}{Rate\ Ratio(x)} = 10\,\text{ns} \]
Mean Link Delay Averaging

\[ m_{PathDelay}(x) = \frac{(t_4 - t_1) - \frac{(t_3 - t_2)}{NRR}}{2} \]

**If** \( x = 1 \)

\[ meanLinkDelay(x) = m_{PathDelay}(x) \]

**If** \( x < 1000 \) then \( \alpha = \frac{1}{x} \) else \( \alpha = \frac{1}{1000} \)

\[ meanLinkDelay(x) = ((1 - \alpha) \times meanLinkDelay(x - 1)) + \alpha \times m_{PathDelay}(x) \]
Startup Behaviour – Recommended Algorithm

\[
\begin{align*}
x = 1 & \quad 1 \\
x = 2 & \quad 1 \leftrightarrow 2 \\
x = 3 & \quad 1 \leftrightarrow 2 \\
x = 4 & \quad 1 \leftrightarrow 2 \\
x = 5 & \quad 1 \leftrightarrow 2 \\
x = 6 & \quad 1 \leftrightarrow 2 \leftrightarrow 3 \\
x = 7 & \quad 1 \leftrightarrow 2 \leftrightarrow 3 \\
x = 8 & \quad 1 \leftrightarrow 2 \leftrightarrow 3 \\
x = 9 & \quad 1 \leftrightarrow 2 \leftrightarrow 3 \\
x = 10 & \quad 1 \leftrightarrow 2 \leftrightarrow 3
\end{align*}
\]
Two Simulation Types

• Single Run
  • Plots progress of meanLinkDelay

• Multiple Runs
  • 100,000 runs
  • Plots progress of
    • Central 90% percentile
    • Min & Max
    • Mean ±6σ
  • Also...
    • Probability Distributions
    • QQ Plots

• Both versions...
  • Only model Timestamp Errors
  • TSGE: ±4 ns
  • DTSE: +6 ns
  • Actual Link Delay: 100 ns
  • Interval: 125 ms (no variation)

For normal distribution, one sample would fall outside ±6σ on average approximately every 500,000,000 samples. For a 125 ms interval that’s approximately once every 2 years. [This is true for a steady state...which this is not...but we’ll come back to that.]
Single Run – 2 min
Single Run – 2 min
Multiple Runs – 5 mins

$6\sigma$: $\pm 1.15$ ns
$90\%$: $\pm 0.32$ ns
Multiple Runs – Distribution at 5 mins

![Graph showing mean link delay distribution at 5 mins. The graph indicates that 90% of the data falls within a certain range.](image)

- **Mean Link Delay at Elapsed time 300 s**
- **Probability Density**
- **Parameters:**
  - Min = 99.22
  - Max = 100.84
  - Mean = 100
  - Std = 1.147

90% of the data falls within the shaded area.
Multiple Runs – First 10 s

6σ: ±50.0 ns
90%: ±13.7 ns
Min: -32.28 ns  Max: +30.96 ns

6σ: ±5.55 ns
90%: ±1.53 ns
Min:-4.03 ns  Max:+4.31 ns

4 Measurements x ±10 ns Timestamp Error
↓
Max ± 40 ns Error
Multiple Runs – Distribution at 0 s

![Distribution Plot]

- Min = 67.72
- Max = 130.96
- Mean = 99.97
- Std = 50.036
Multiple Runs – QQ Plot at 0s

Distribution **can not** be regarded as Gaussian/Normal at 0 s
Multiple Runs – Distribution at 10 s
Multiple Runs – QQ Plot at 10s

Distribution can be regarded as Gaussian/Normal at 10s
Multiple Runs – 5th Minute

±6σ range is still narrowing after 5 mins
Multiple Runs – 5⁰ Minute

±6σ range is still narrowing after 5 mins
Multiple Runs – 30 mins

±6σ range is stable – within ±1.13 ns – after **7 mins**!
# Recommended Algorithm – Startup Behaviour

<table>
<thead>
<tr>
<th>After...</th>
<th>6σ Range (ns)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Measurement</td>
<td>N/A [Feasible Limit ± 40ns]</td>
<td>Distribution can’t be treated as Gaussian / Normal. Simulate via TSGE &amp; DTSE.</td>
</tr>
<tr>
<td>10 s</td>
<td>±5.55</td>
<td></td>
</tr>
<tr>
<td>20 s</td>
<td>±3.94</td>
<td></td>
</tr>
<tr>
<td>30 s</td>
<td>±3.21</td>
<td></td>
</tr>
<tr>
<td>1 min</td>
<td>±2.28</td>
<td></td>
</tr>
<tr>
<td>2 min</td>
<td>±1.61</td>
<td></td>
</tr>
<tr>
<td>3 min</td>
<td>±1.33</td>
<td></td>
</tr>
<tr>
<td>4 min</td>
<td>±1.20</td>
<td></td>
</tr>
<tr>
<td>5 min</td>
<td>±1.15</td>
<td></td>
</tr>
<tr>
<td>6 min</td>
<td>±1.13</td>
<td></td>
</tr>
<tr>
<td>7 min</td>
<td>±1.12</td>
<td>After 7 mins of simulation 6σ Range varies between ±1.13 ns and ±1.11 ns</td>
</tr>
</tbody>
</table>
Network Level – 100 Hops

• New Monte Carlo Simulation!
  • Assumes normal distribution for meanLinkDelay error
  • Uses data from Multiple Run MLD simulation to determine standard deviation for normal distribution
    • Apart from simulation of 1st measurement, which models each timestamp error (since the distribution at this stage can’t be regarded as Gaussian/Normal)
  • Each run simulates a 100-hop network with meanLinkDelay error accumulating down the chain
  • 1 Million Runs x 11 (plus some 100,000 Runs for QQ Plots)
  • Generates probability distribution plots for accumulated meanLinkDelay error at 64 and 100 hops
MLD Error $6\sigma$ Range Over 100 Hops

Recommended Algorithm

If there is no averaging, MLD error consumes entire PTP Instance networking-level dTE budget.
# 60802 Time Error Budget – 1,000 ns

<table>
<thead>
<tr>
<th>Network Aspect</th>
<th>Error Type</th>
<th>Network Level Error Budget (ns)</th>
<th>Normative or Informative?</th>
</tr>
</thead>
<tbody>
<tr>
<td>All PTP Instances</td>
<td>Constant Time Error</td>
<td>200</td>
<td>Normative</td>
</tr>
<tr>
<td></td>
<td>Dynamic Time Error</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>All PTP Links</td>
<td>Constant Time Error</td>
<td>200</td>
<td>Informative</td>
</tr>
<tr>
<td></td>
<td>Dynamic Time Error</td>
<td>Negligible</td>
<td></td>
</tr>
</tbody>
</table>

- 50 ns at Grandmaster and End Instance for dTE between network and application level (50 ns each)
- 600 ns PTP Instance dTE budget – 100 ns = 500 ns
Is $6\sigma$ the Correct Range?

• If $6\sigma$ is the right range to consider for constant steady state operation over a number of years...
  • 1 MLD error outside the range every couple of years
  • (but unlikely to occur at the same time as a similarly large error in the same direction of other contributors towards dTE...and likely to be mitigated by endpoint filtering...i.e. you can’t simply add together $6\sigma$ values)

• Might a narrower range be a better measure for startup behaviour?
  • If a network starts up once per day and takes 7 mins to fully settle, $6\sigma$ range means 1 MLD error outside the range, during startup, on average every 200 years.
Reasons to Stick with 6σ Range

• We’re interested in how long the MLD averaging algorithm takes to reach steady state performance, where the 6σ range is appropriate.

• Other errors are considered in terms of 6σ range.

• A lot depends on how often the network is initialised.

• The 6σ range is constantly shrinking throughout the startup period until steady state behaviour is achieved. Difficult to model accurately.

• 6σ range is good enough to indicate algorithm behaviour and error contribution relative to other factors.

So..sticking with 6σ range.
Meaning of $6\sigma$ Range

• When considering startup behaviour, this is the $6\sigma$ range value that is useful for considering over the long term operation of the network.
  • It is not as directly useful for considering the likelihood of an out-of-range error during the startup period.
  • But it’s still useful for considering and comparing startup behaviours and the magnitude of errors that are likely to be seen.
MLD Error 6σ Range Over 100 Hops

Recommended Algorithm

"1st Measurement" indicates a likely worst case error if no algorithm is employed.
MLD Error $6\sigma$ Range without Algorithm at Hop 100 - QQ Plot

Distribution at 1st hop **can not** be regarded as Gaussian. Distribution at 100th hop **can**.
MLD Error 6σ Range Over 100 Hops

Recommended Algorithm

“1st Measurement” indicates a likely worst case error if no algorithm is employed.
MLD Error $6\sigma$ Range Over 100 Hops

Recommended Algorithm

"1st Measurement" indicates a likely worst case error if no algorithm is employed.
MLD Error 6σ Range at 1 Hop vs. 100 Hops

1st Measurement,
1 hop: ±50 ns (not a normal distribution)
100 hops: ±500 ns (is a normal distribution)
MLD Error 6σ Range at 1 Hop vs. 100 Hops

After 10 s, 1 hop: 100 ±5.55 ns, 100 hops: ±55.5 ns

After 20 s, 1 hop: ±3.94 ns, 100 hops: ±39.3 ns

After 30 s, 1 hop: ±3.21 ns, 100 hops: ±32.1 ns

After 1 min, 1 hop: ±2.28 ns, 100 hops: ±22.8 ns
MLD Error $6\sigma$ Range at 1 Hop vs. 100 Hops

After 2 mins, 1 hop: ±1.61 ns, 100 hops: ±16.1 ns

After 3 mins, 1 hop: ±1.33 ns, 100 hops: ±13.3 ns

After 4 mins, 1 hop: ±1.20 ns, 100 hops: ±12.0 ns

After 5 mins, 1 hop: ±1.15 ns, 100 hops: ±11.5 ns

After 6 mins, 1 hop: ±1.13 ns, 100 hops: ±11.3 ns

After 7 mins, 1 hop: ±1.12 ns, 100 hops: ±11.1 ns

After 100 hops, MLD error standard deviation is approx $10 \times$ error after 1 hop.
Don’t Truncate to 0 ns Link Delay!

• This might seem intuitive (zero or negative link delay is impossible absent time travel) but it causes problems.

• There are two possibilities...
  • Truncate each Path Delay measurement to zero, prior to the IIR filter.
  • Truncate the meanLinkDelay to zero, if the IIR filter is less than zero.
Link Delay 5 ns – 1 Hop
Without Truncating of Path Delay Measurement
Link Delay 5 ns – 1 Hop
With Truncation of Path Delay Measurement

In this example, mean of meanLinkDelay stabalises at 6.42 ns.
Link Delay 5 ns – 100 Hops – After 10 s
Without Truncation of Path Delay Measurement or meanLinkDelay Value

Mean = 0.0059 ns
Link Delay 5 ns – 100 Hops – After 10 s
Without Truncation of Path Delay Measurement – With Truncation of meanLinkDelay Value

Impact isn’t as large, but Mean = 0.51 ns (versus Mean = 0.0059 ns without truncation).
Ramping the Filter Factor is Valuable

• There is a big difference between ramping the filter factor...

\[
\text{If } x < 1000 \text{ then } \alpha = \frac{1}{x} \text{ else } \alpha = \frac{1}{1000}
\]

• ...verses not ramping...

\[
\alpha = \frac{1}{1000}
\]
Ramp vs No Ramp
Ramp vs No Ramp

Without Ramping, meanLinkDelay takes 13 mins to settle, vs 7 mins with Ramping
Don’t forget to initialise the filter with the first measurement...

• This...

\[
\text{If } x = 1 \quad \text{meanLinkDelay}(x) = mPathDelay(x)
\]

• ...is very different from this...

\[
\text{If } x = 1 \quad \text{meanLinkDelay}(x) = 0 + \alpha \times mPathDelay(x)
\]

• ...which also implies no ramping of the filter factor.
MLD Error With Incorrect Initialisation
MLD Error With Incorrect Initialisation

Without correct initialisation (and without ramping) meanLinkDelay error can take 10 mins to stabilise
Dos & Don’ts Summary

• Do
  • Ramp the IRR filter factor
  • Initialise the filter with the first measurement

• Don’t
  • Truncate path delay measurements or meanLinkDelay to zero
Implications for Simulations
Time Series Simulations

• Recent Time Series simulations used a factor of 100 (not the recommended 1000)
• Time Series simulations also discard the first 50 s of data for the purposes of calculating dTE statistics
  • Simulation time is 3,100 s i.e. 3,050 is used for calculating dTE statistics
MLD Error – Factor of 100
MLD Error – Factor of 100

Factor of 100 settles to $\pm 3.6$ ns after 27 s (versus $\pm 3.39$ ns for factor of 1000; $\pm 1.13$ ns after 7 mins)
Time Series Simulations – Implications

• Recent Time Series simulations effectively had MLD distribution with $6\sigma$ Range of $\pm 3.6$ ns per node; $\pm 36$ ns over 100 hops.

• If recommended settings are used, MLD distribution $6\sigma$ Range of $\pm 1.13$ ns per node; $\pm 11.3$ ns over 100 hops...but only after 7 mins
  - If the recommended algorithm is used, the Time Series starts assessing dTE after 50 seconds, when MLD distribution $6\sigma$ Range is $\pm 2.5$ ns.

• Each replication spends 370 s where MLD distribution $6\sigma$ Range after 100 hops is...
  - ...between $\pm 9$ ns and $\pm 25$ s worse than recommended settings for 370 s
  - $\pm 25$ ns worse than recommend settings for 2,680 s

• $\pm 25$ ns worse is the steady-state behaviour
Time Series Simulations – Recommendations

• For simulations targeting steady-state behaviour
  • Initialise IIR filter for each node with a value from the normal distribution:
    • Mean: actual path delay
    • Standard Deviation: 1.88 ns
  • Valid for TSGE = ±4 ns; DTSE = ±6 ns
  • For other combinations of TSGE and DTSE, matching Monte Carlo simulations can generate the appropriate standard deviation

• For simulations targeting initialisation behaviour, use recommended algorithm
Monte Carlo Simulations

• Initial simulations used a % effective value to emulate an algorithm
  • For a given % effective, that % of the error was eliminated.
  • Simulations typically used a value of 95% effective
  • This contribution illustrates a reduction from MLD Error distribution 6σ Range at 100 hops of ±500 ns to ±11.1 ns, i.e. 97.8% effective.

• Recent simulations took an average of 50 Pdelay calculations, which is the equivalent of the IRR filter after 6.125 s
  • Effective MLD Error distribution 6σ Range at 100 hops of ±70.1 ns
  • This is ±59 ns worse than what can the recommended algorithm can be achieved at steady state.
Monte Carlo Simulations – Recommendations

• For MLD Error (from Timestamp Error) use a value from the normal distribution:
  • Mean: actual path delay
  • Standard Deviation: 1.88 ns
  • This calculation won’t just be more accurate than previous method (average of 50 path delay calculations), it will be much faster than most recent method

• Valid for TSGE = ±4 ns; DTSE = ±6 ns

• For other combinations of TSGE and DTSE, matching Monte Carlo simulations can generate the appropriate standard deviation
Normative Requirements
Background

- Discussion during comment resolution of d1.1 that a normative requirement on meanLinkDelay accuracy is both possible and useful
  - meanLinkDelay is accessible via the relevant managed object (see IEEE 802.1AS-2020 14.16.6)
- There was no comment on d1.1 about this, so the discussion was deemed out of scope, but likely to be subject of comments on d1.2
  - May be rejected as out-of-scope for d1.2...but will then come back at SA Ballot
meanLinkDelay Normative Requirement?

- We could require performance equivalent to the steady state performance from the Monte Carlo simulation, but...
- That doesn’t take into consideration some other sources of meanLinkDelay error (e.g. Clock Drift and Timestamp Error affecting NRR estimation)
  - Effect is orders of magnitude less than direct Timestamp error
- Maybe overkill?
  - We already have normative requirements on overall error generation that cover accurate measurement of meanLinkDelay + residenctime
  - If meanLinkDelay algorithm is mostly effective, the impact of meanLinkDelay error will be swamped by other sources of error
- What level of meanLinkDelay error starts to have a negative impact on overall dTE?
100 Hops – MLDerrorSDx6 1.13 ns
Recommended meanLinkDelay Algorithm – Steady State

minDTE_SUM = -620 ns  maxDTE_SUM = 632 ns  sdDTE_SUM = 159 ns [σ]
100 Hops – MLDerrorSDx6 5.55 ns
Recommended meanLinkDelay Algorithm – After 10 s

\[ \text{minDTE}_{\text{SUM}} = -627 \text{ ns} \quad \text{maxDTE}_{\text{SUM}} = 640 \text{ ns} \quad \text{sdDTE}_{\text{SUM}} = 159 \text{ ns} [\sigma] \]
Normative Requirement – meanLinkDelay
Suggestions & Questions

• Noting that...
  • 1 sample in approx. 500 million falls outside 6σ range
  • 1 sample in approx. 3.5 million falls outside 5σ range
  • No algorithm → OK algorithm is a big difference
  • A simple IIR filter with factor 100 can deliver 6σ range ±3.6 ns after 27 s
    • Equivalent 5σ range is ±3.0 ns
    • An IRR filter with factor 1,000 delivers 5σ range is ±2.5 ns
  • Going from 6σ Range 1.13 → 5.55 ns doesn’t have a large negative effect on dTE, given other sources of error.

• Suggest a normative requirement of ±3 ns
  • Give enough information in the informative annex about behaviour of the recommended algorithm for test spec to be written appropriately.
Thank you
Default Configuration

hops <- 100 # Minimum 1 hop
runs <- 100000
#
# Input Errors, Parameters & Correction Factors
driftType <- 3 # 1 = DO NOT USE - Historical - Uniform Probability Distribution between MIN & MAX ppm/s
  # 2 = Probability Based on Linear Temp Ramp
  # 3 = Probability Based on Half-Sinusoidal Temp Ramp
  # 4 = Probability Based on Quarter-Sinusoidal Temp Ramp
# Clock Drift Probability from Temp Curve & XO Offset/Temp Relationship
tempMax <- +85 # degC - Maximum temperature
tempMin <- -40 # degC - Minimum temperature
tempRampRate <- 1 # degC/s - Drift Rate for Linear Temp Ramp
tempRampPeriod <- 95 # s - Drift Period for Sinusoidal & Half-Sinusoidal Temp Ramps
tempHold <- 30 # s - Hold Period at MIN and MAX temps before next temp ramp down or up
GMscale <- 0.5 # Ratio of GM stability vs. standard XO. 1 is same. 0 is perfectly stable.
nonGMscale <- 1 # Ratio of non-GM (and non-ES) node stability vs. standard XO. 1 is same. 0 is perfectly stable.
Default Configuration

TSGEx <= 4 # +/- ns - Error due to Timestamp Granularity on TX
TSGEr <= 4 # +/- ns - Error due to Timestamp Granularity on RX
DTSEx <= 6 # +/- ns - Dynamic Timestamp Error on TX
DTSER <= 6 # +/- ns - Dynamic Timestamp Error on RX
syncInterval <= 125 # ms - Nominal Interval between two Sync messages
SImode <= 3 # Mode for generating Tsync2sync *HARD CODED to MODE 3*

  # 1 = Gamma Distribution, defaulting to 90% of Tsync2sync falling within 10% of the nominal syncInterval. Truncated at SImax (higher values above are reduced to SImax)
  #   No truncation of low values
  # 2 = Gamma Distribution, defaulting to 90% of Tsync2sync falling within 10% of the nominal syncInterval. Truncated at SImax (higher values are reduced to SImax)
  #   Truncated at SImin (lower values are increased to SImin)
  # 3 = Uniform, linear distribution between syncInterval x SImin and syncInterval x SImax
Default Configuration

SIscale <- 1 # Scaling factor for Mode 1 & 2 Tsync2sync vs regular distribution.
    # Scaling factor of 3 would mean 90% of Tsync2sync falling within 30% of the nominal syncInterval
SImax <- 1.048  # For mode 1 & 2, Max truncation factor (e.g. 2x syncInterval) limit for Tsync2sync; higher values reduced to SImax
    # For mode 3, upper limit of uniform linear distribution
SImin <- 0.952  # For mode 1 & 2, Min truncation factor (e.g. 0.5 x syncInterval) limit for Tsync2sync; higher values reduced to SImin
    # For mode 3, lower limit of uniform linear distribution
pathDelayMin <- 5 # ns - 1m cable = 5ns path delay
pathDelayMax <- 500 # ns - 100m cable = 500ns path delay
residenceTime <- 15 # ms - TresidenceTime maximum; higher values truncated
RTmin <- 1 # TresidenceTime minimum; lower values truncated
RTmean <- 5 # TresidenceTime mean
RTsd <- 1.8 # TresidenceTime sigma; 3.4ppm will fall outside 6-sigma either side of the mean
Default Configuration

MLDerrorSDx6 <- 1.13 # ns - 6x Standard Deviation of meanLinkDelay Error
   # Calculated from separate simulation.
   # 1.13 ns is for steady state after IIR filter with alpha = 1000 has stabalised

mNRRsmoothingNA <- 4 # Whole Number >=1 - Combined N & A value for "smoothing" calculated mNRR (mNRRc)
   # Calculate mNRR using timestamps from Nth Sync message in the past
   # Then take average of previous A mNRRcalculations.

mNRRcompNAP <- 8 # Whole Number >=1
   # For NRR drift rate error correction calculations, take two measurements, mNRRa and mNRRb.
   # Both use timestamps from Nth Sync message in the past, then take average of previous A calculations.
   # Calculation mNRRb starts P calculations in the past from mNRRa, where P = mNRRcompNAP * 2.
   # If 0, there is no NRR drift rate error correction.