60802 Time Sync – Mean Link Delay & Timestamp Granularity

David McCall (Intel Corporation)

Version 1

References

[1] – Geoff Garner, "Initial 60802 Error Generation Time Series Simulation Results Version 1", contribution to IEC/IEEE 60802, 22nd January 2024

[2] – Geoff Garner, "Further 60802 Error Generation Time Series Simulation Results Version 1", contribution to IEC/IEEE 60802 13th March 2024

Content

- Background & Baseline Assumptions
- Description of Simulation
- $mPathDelay$ Measurements and $meanLinkDelay$ Errors
  - DTSE then TSGE (matching implementations)
  - Effect of Clock Offsets & Pdelay Turnaround
- Simulations to match Time Series Simulations in [1] & [2]
  - TSGE then DTSE

Note: In previous contributions I've often looked at modelling errors. In this contribution, I'm looking at the behaviour of implementations as a result of errors. The simulation models timestamps, not just errors.

Background

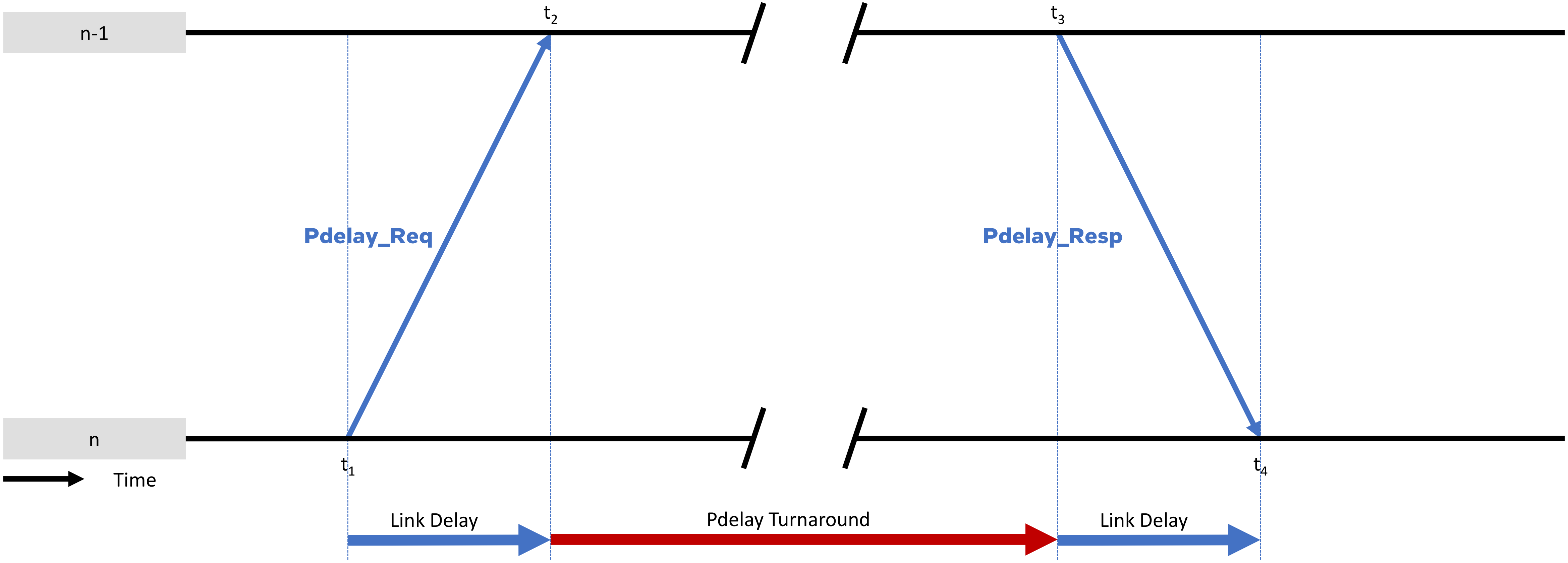
- Mean Link Delay ($meanLinkDelay$) measurements in [1] and [2] demonstrated some unexpected behaviour, with step changes between two values either side of the actual value.
  - See slides 22 to 27
- This contribution is a detailed breakdown of the way timestamp errors, clock offsets and errors due to clock drift can contribute to errors in the individual Path Delay measurements ($mPathDelay$). It investigates how these errors, combined with the $meanLinkDelay$ filtering (averaging) algorithm might produce the observed (simulated) behaviour.

Baseline Assumptions

- Unless mentioned, the "usual" 60802 configuration applies…
    - Same timestamp and clock drift errors as [1]
        - Timestamp Granularity: 8 ns
        - Dynamic Timestamp Error: ±6 ns
    - Same parameters and configuration as [1]
        - Pdelay Interval: uniform distribution, -119 ms to 131 ms (note: different from Time Series, but does not have an effect on the end results)
        - Pdelay Turnaround: normal distribution, mean 10 ms, standard deviation 1.8 ms
            - Truncated to 1 ms and 15 ms (values outside range are rounded up or down respectively)
- $mPathDelay$ means raw measurements of Path Delay, before input to the Mean Link Delay filter
- $meanLinkDelay$ means output from Mean Link Delay filter
    - Filter is as described in clause D.5.7 of 60802 and is the same as used in [1]
- Note: Clock Drift is **not** currently simulated (but constant Clock Offsets are)

Description of Simulation

Path Delay Measurement



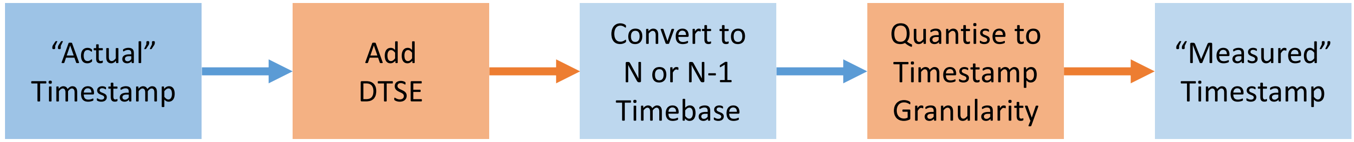
$$PathDelay = \frac{\left(t_4(x) - t_1(x)\right) - \dfrac{\left(t_3(x) - t_2(x)\right)}{Neighbor\ Rate\ Ratio(x)}}{2}$$

Description of Simulation – 1

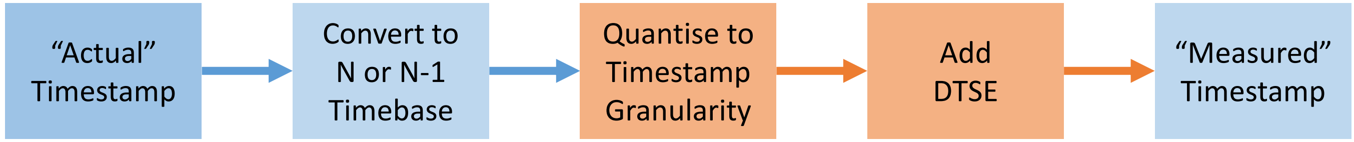
- RStudio script for R
- 1 Vector (1 dimensional array) per variable, of length **runs**, where **runs** is the number of Pdelay_Req / _Resp message exchanges being simulated.
  - **runs** is typically 10,000
- Simulation has a concept of an “ideal” clock against which two “real” clocks are measured
  - One clock for node N, which is measuring meanLinkDelay
  - One clock for node N-1, the Path Delay to which node N is measuring.
  - Each “real” clock has a constant offset (ppm) relative to the ideal clock
    - The offset can be 0 ppm

Description of Simulation – 2

- Models a series of Pdelay_Req and Pdelay_Resp messages, each separated by a randomly generated Pdelay Interval and with a randomly generated Pdelay Turnaround time
  - Both are in terms of “ideal” clock
- Models “actual” timestamps for t_1, t_2, t_3 and t_4 in terms of “ideal” clock
- Then:



- Note that Time Series Simulation does this (and this RStudio simulation has an option to do the same)…



Description of Simulation – 3

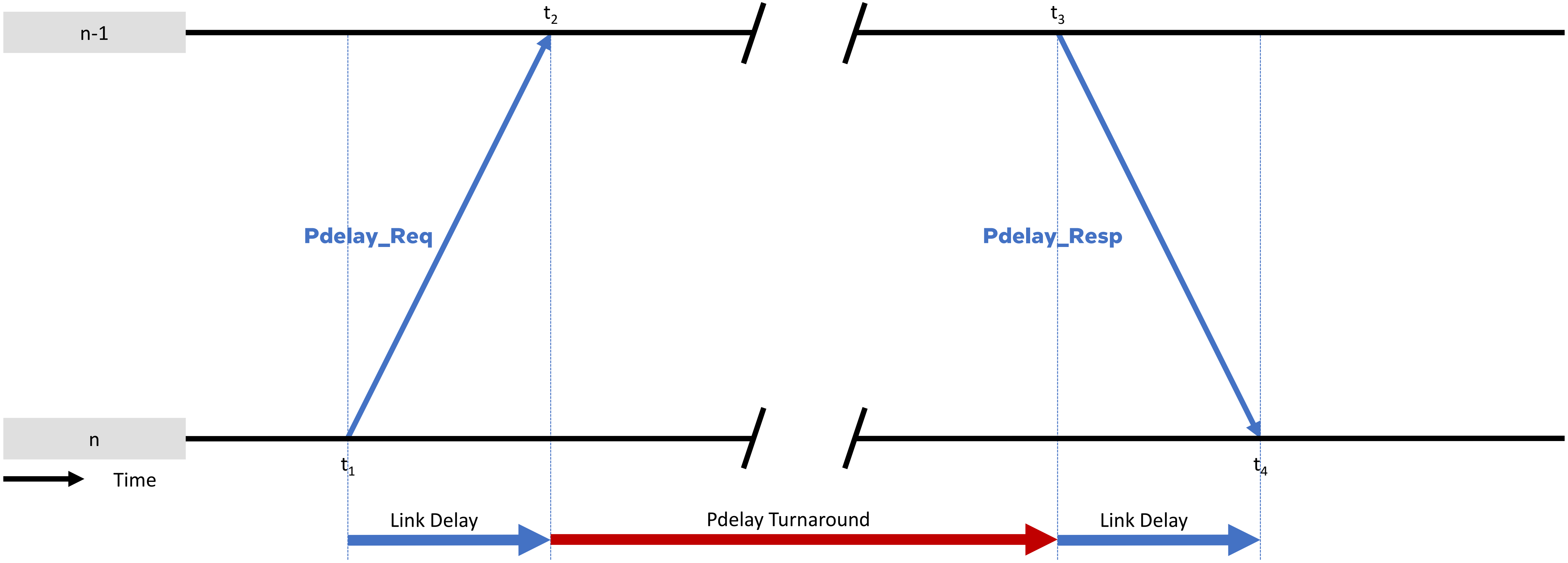
- Carries out mPathDelay calculation:

$$PathDelay = \frac{\left(t_4(x) - t_1(x)\right) - \frac{\left(t_3(x) - t_2(x)\right)}{Neighbor\ Rate\ Ratio(x)}}{2}$$

  - NRR is the actual NRR with the option for added error
    - Error is not calculated (as it is from a different process that is not modelled) but rather taken as an output from [1]. Same error file is used for all simulations.
- Carries out meanLinkDelay calculation using algorithm defined in D.5.7
  - Uses initial value of Path Delay with an error of normal distribution, mean 0 ns, standard deviation 0.1 ns. (Avoids need to simulate start-up behaviour.)

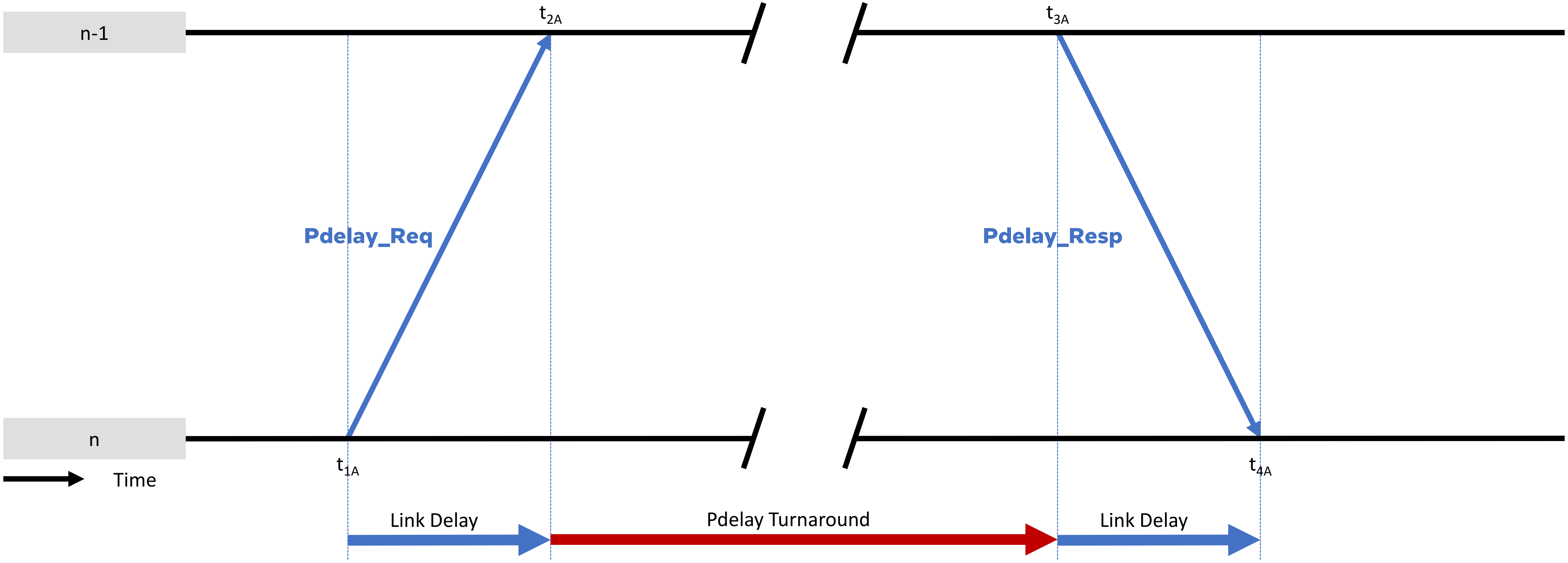
$mPathDelay$ Measurements & $meanLinkDelay$ Errors

Path Delay Measurement



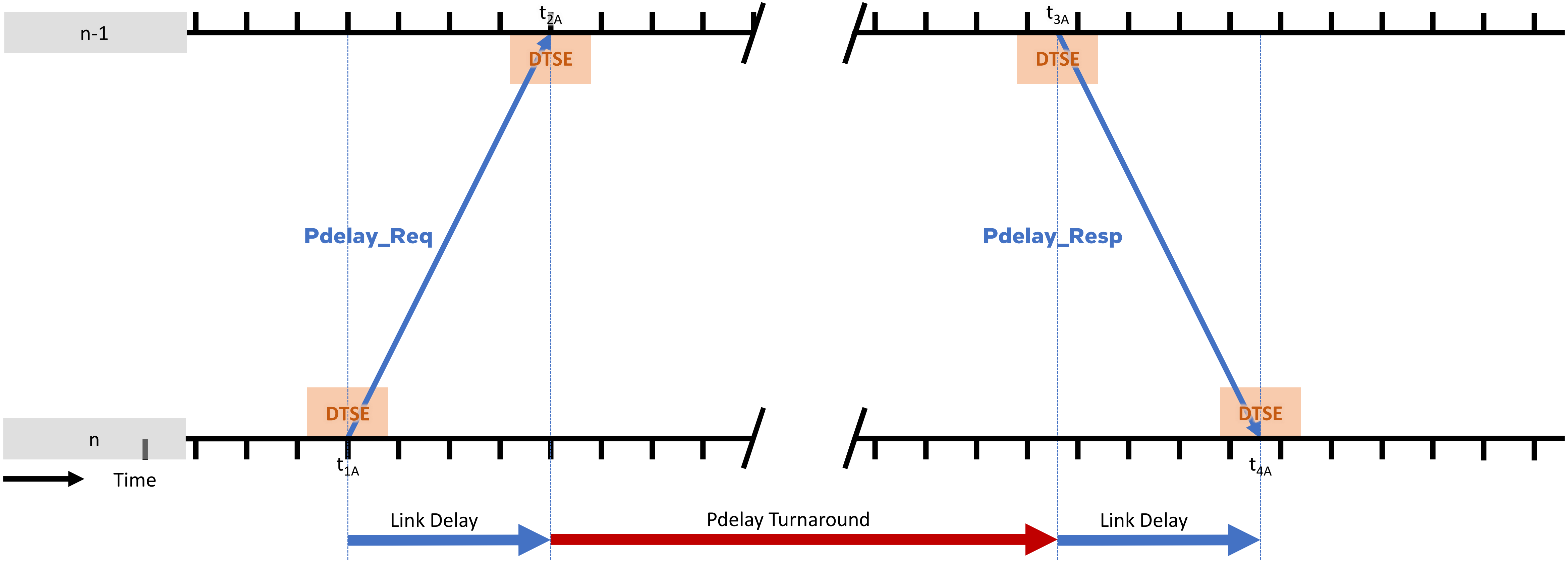
$$PathDelay = \frac{\left(t_4(x) - t_1(x)\right) - \dfrac{\left(t_3(x) - t_2(x)\right)}{Neighbor\ Rate\ Ratio(x)}}{2}$$

Path Delay Measurement – Actual



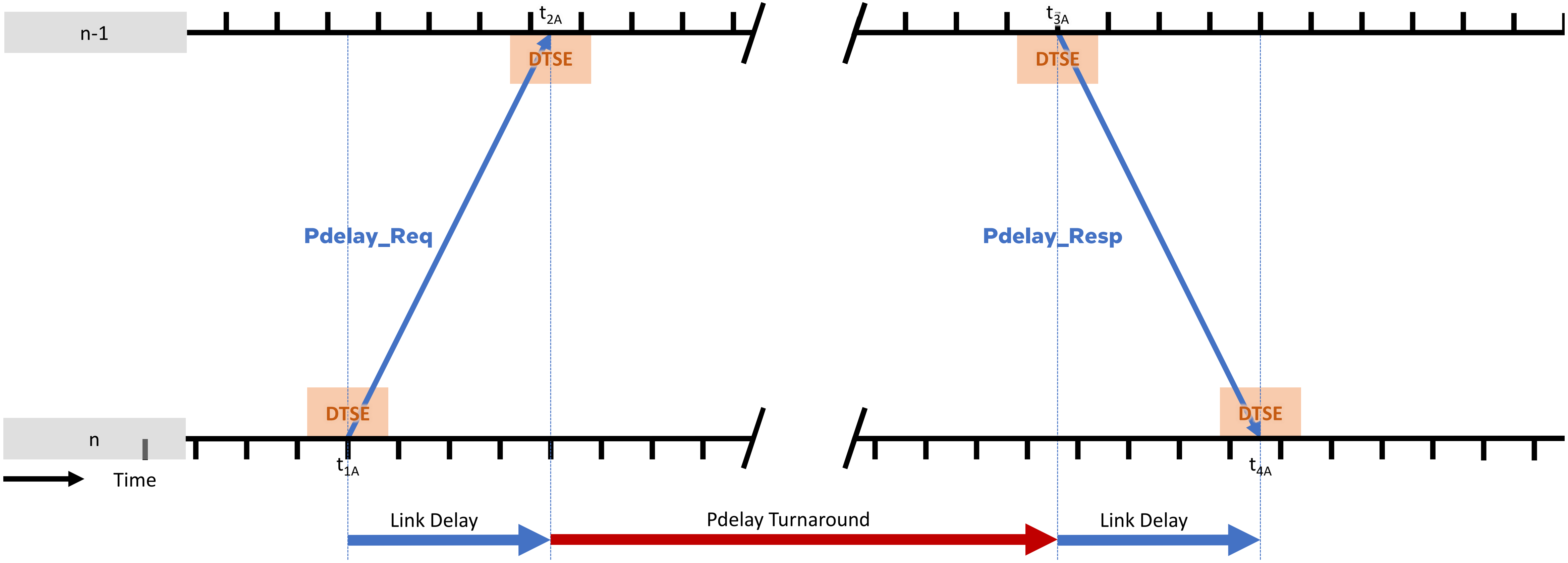
$$PathDelay = \frac{\left(t_{4A}(x) - t_{1A}(x)\right) - \dfrac{\left(t_{3A}(x) - t_{2A}(x)\right)}{Neighbor\ Rate\ Ratio(x)}}{2}$$

Path Delay Measurement – Actual



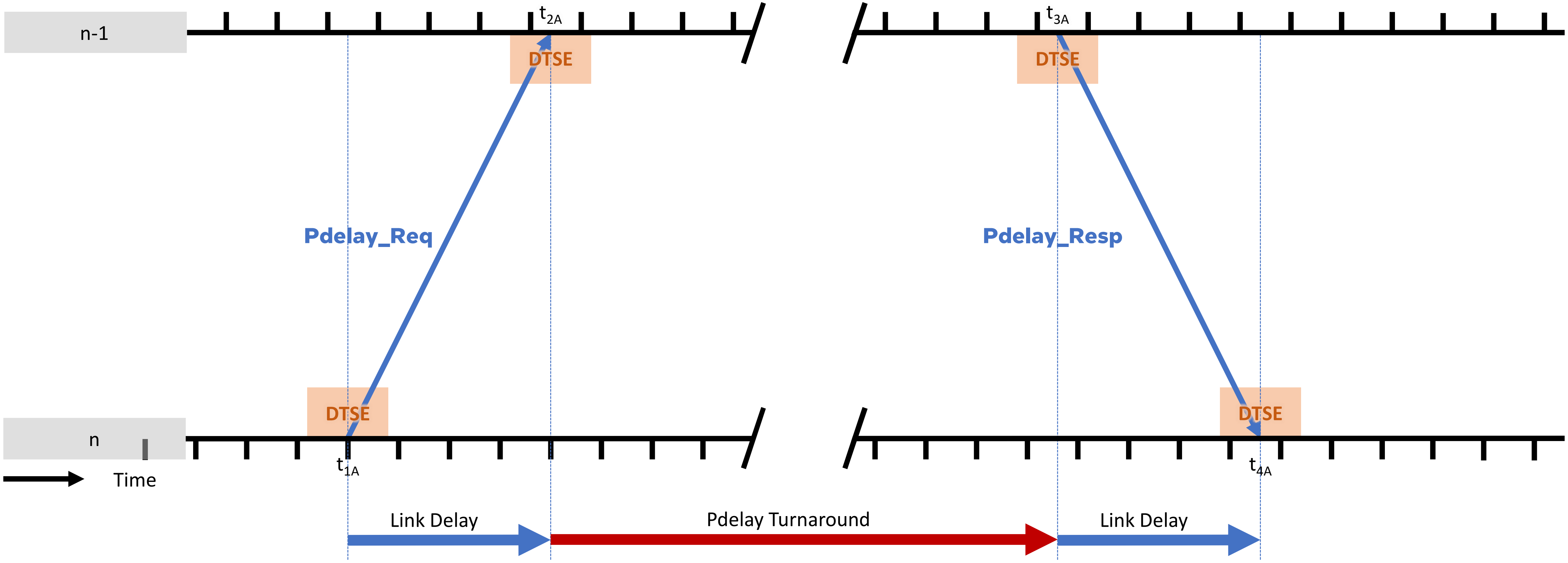
$$PathDelay = \frac{\left(t_{4m}(x) - t_{1m}(x)\right) - \dfrac{\left(t_{3m}(x) - t_{2m}(x)\right)}{Neighbor\ Rate\ Ratio(x)}}{2}$$

Path Delay Measurement – Actual



$$PathDelay = \frac{\left(t_{4m}(x) - t_{1m}(x)\right) - \dfrac{\left(t_{3m}(x) - t_{2m}(x)\right)}{Neighbor\ Rate\ Ratio(x)}}{2}$$

Path Delay Measurement – Actual



$$PathDelay = \frac{\left(t_{4m}(x) - t_{1m}(x)\right) - \dfrac{\left(t_{3m}(x) - t_{2m}(x)\right)}{Neighbor\ Rate\ Ratio(x)}}{2}$$

Path Delay Measurement – Actual

4ns Granularity

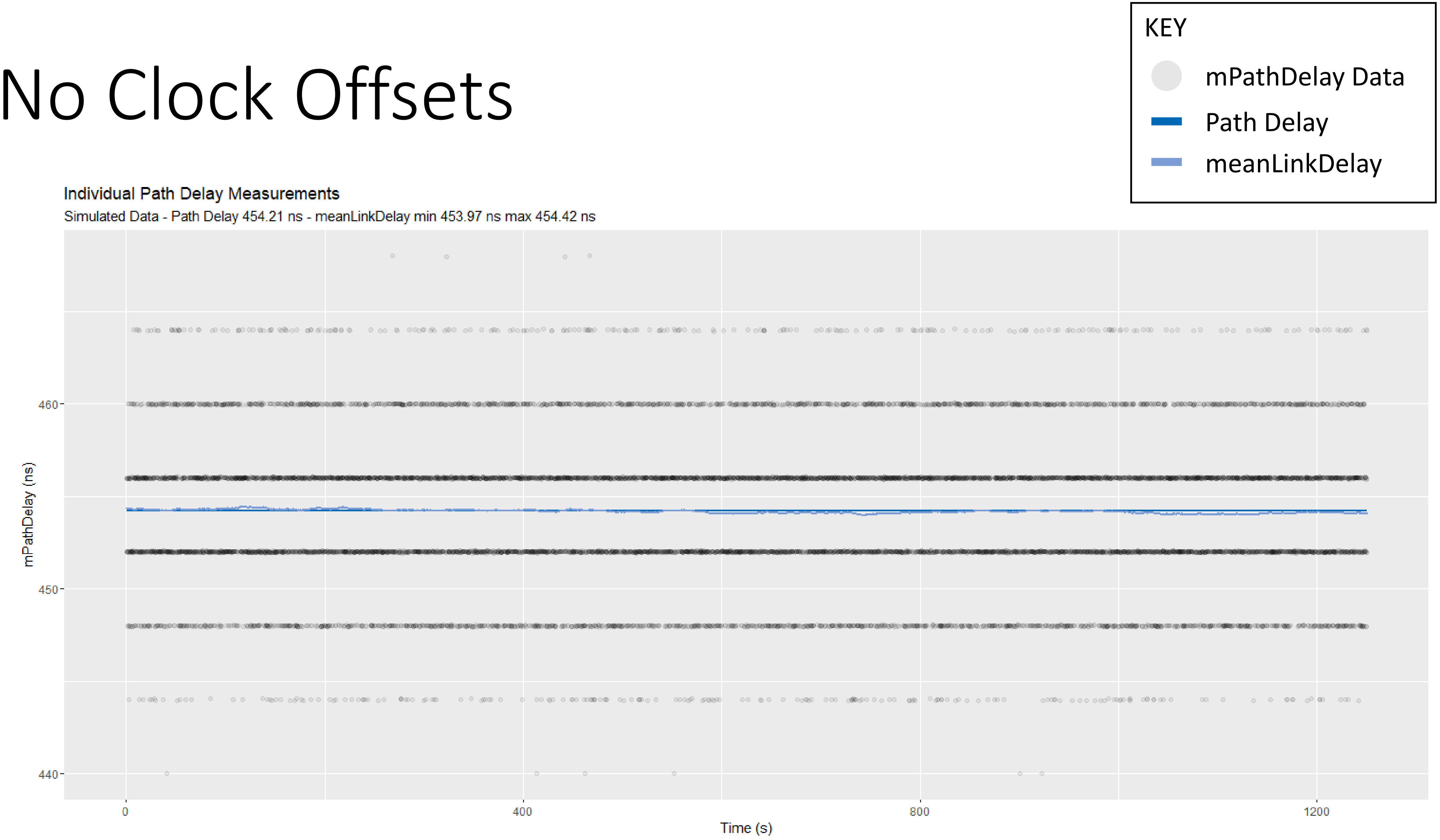
8ns Granularity

8ns Granularity

$$PathDelay = \frac{\boxed{\left(t_{4m}(x) - t_{1m}(x)\right)} - \frac{\boxed{\left(t_{3m}(x) - t_{2m}(x)\right)}}{Neighbor\ Rate\ Ratio(x)}}{2}$$

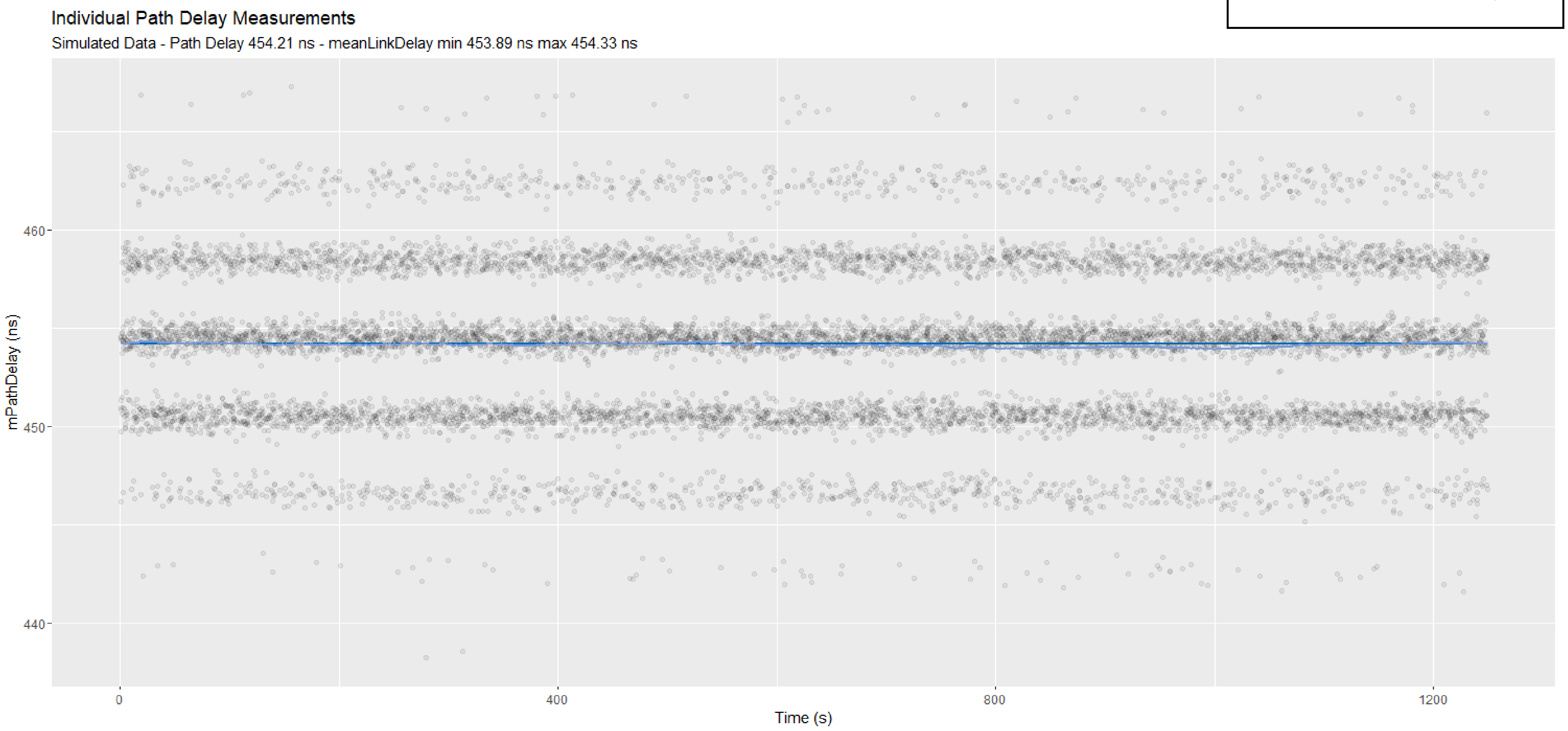
If NRR is 1 (0 ppm)

No Clock Offsets



n-1 node +0.5 ppm

KEY
- mPathDelay Data
- Path Delay
- meanLinkDelay



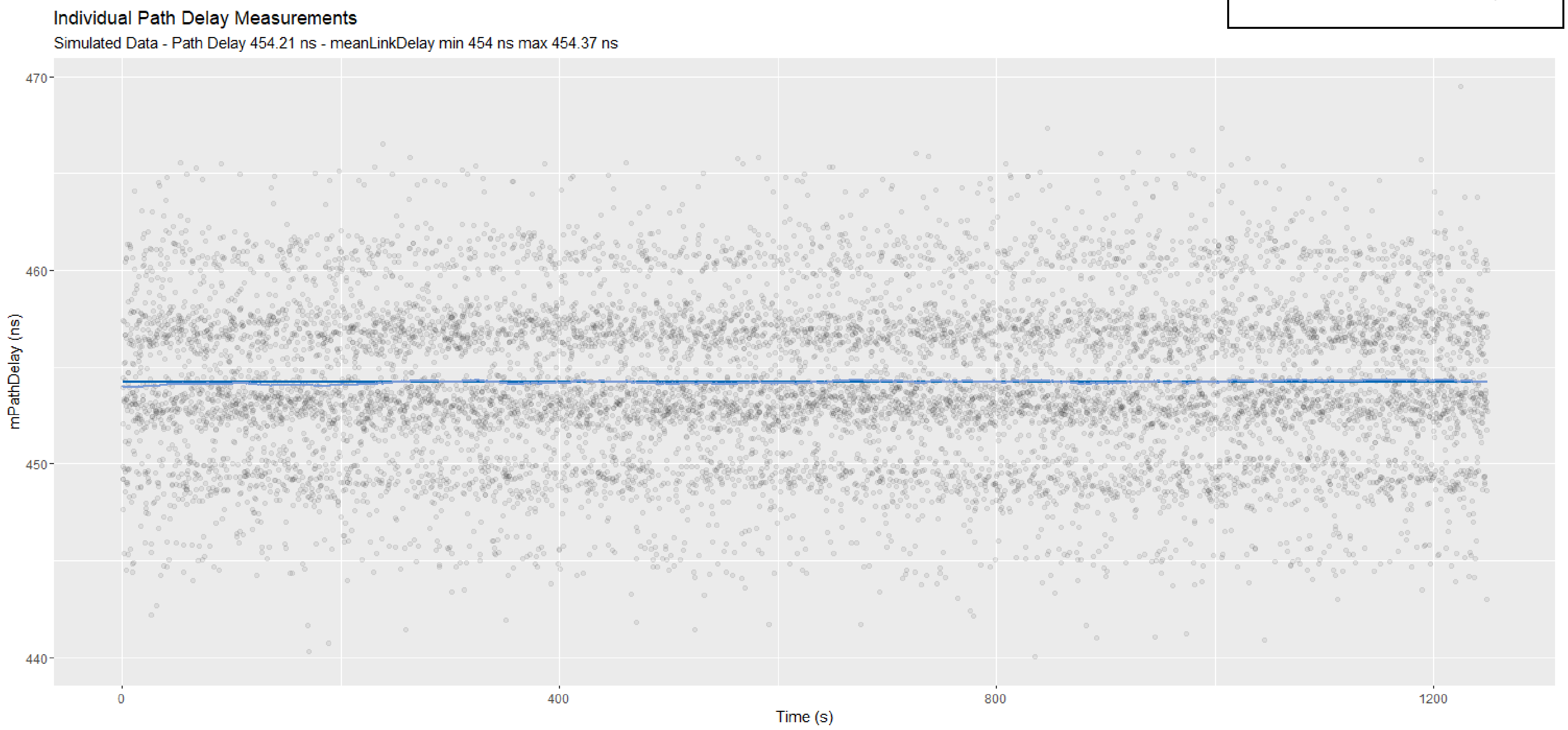
n-1 node +1 ppm

KEY

mPathDelay Data

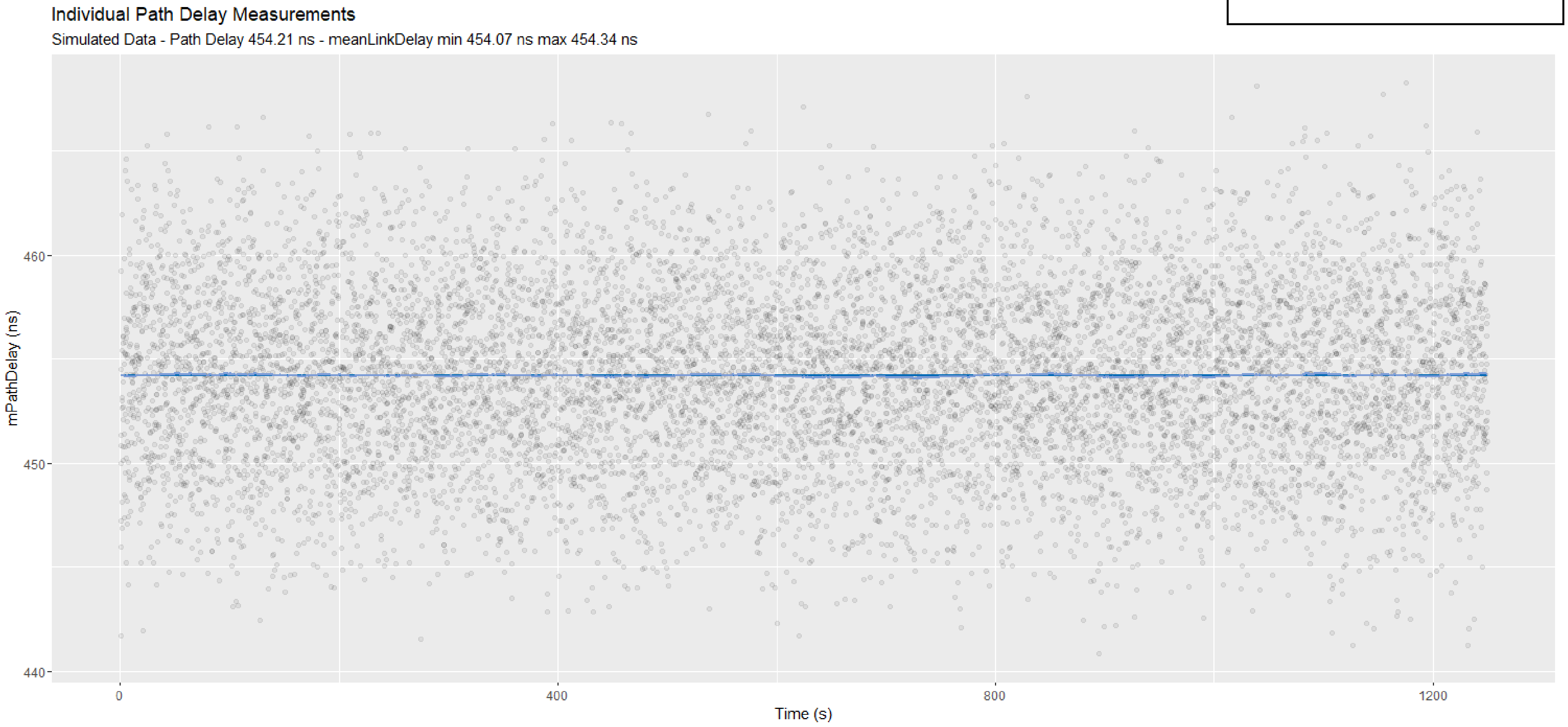
Path Delay

meanLinkDelay



n-1 node +5 ppm

KEY
mPathDelay Data
Path Delay
meanLinkDelay



Observations

- Zero clock offset (note: simulation is “perfect”, i.e. clocks are exactly in sync apart from initial phase offset) illustrates 4ns quantisation steps according to TSGE.
- Adding a clock offset spreads out the measurements around the quantisation steps <u>**and**</u> shifts the steps themselves, within broad limits.
- If the clock offset is large enough, the quantisation steps are swamped and the datapoints appear to be dispersed around the actual Path Delay

Proposition

- The way the data is spread around the quantisation steps is related to the probability distribution of Pdelay Turnaround.
  - Test by varying the distribution of Pdelay Turnaround
- The shift of the quantisation steps is affected by both the distribution of Pdelay Turnaround and clock offset(s)
  - Test by varying the distribution of Pdelay Turnaround and the clock offsets

n-1 node +0.5 ppm

Pdelay Turnaround – Truncated Normal Distribution

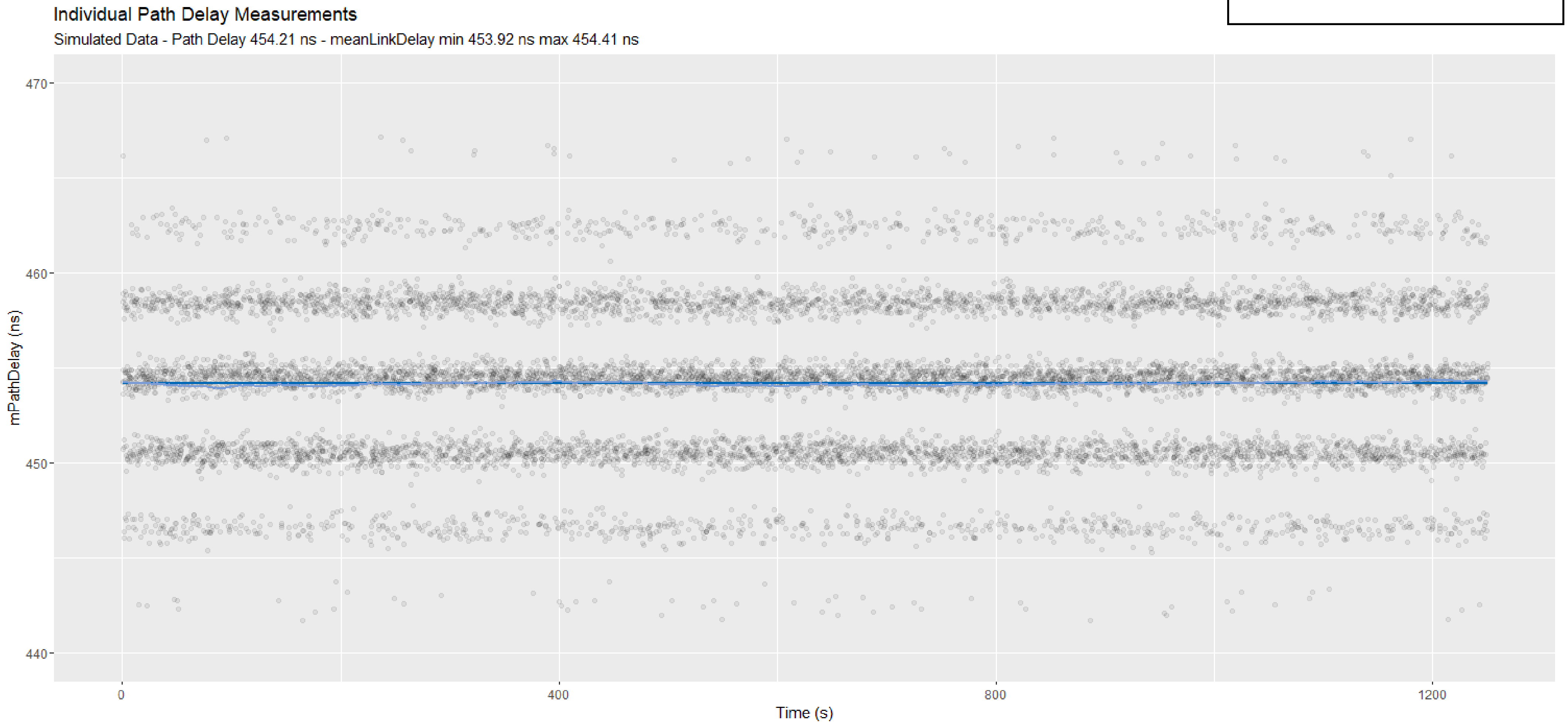
(μ =10; σ=1.8; truncated 1 to 15 ms)

KEY

mPathDelay Data

Path Delay

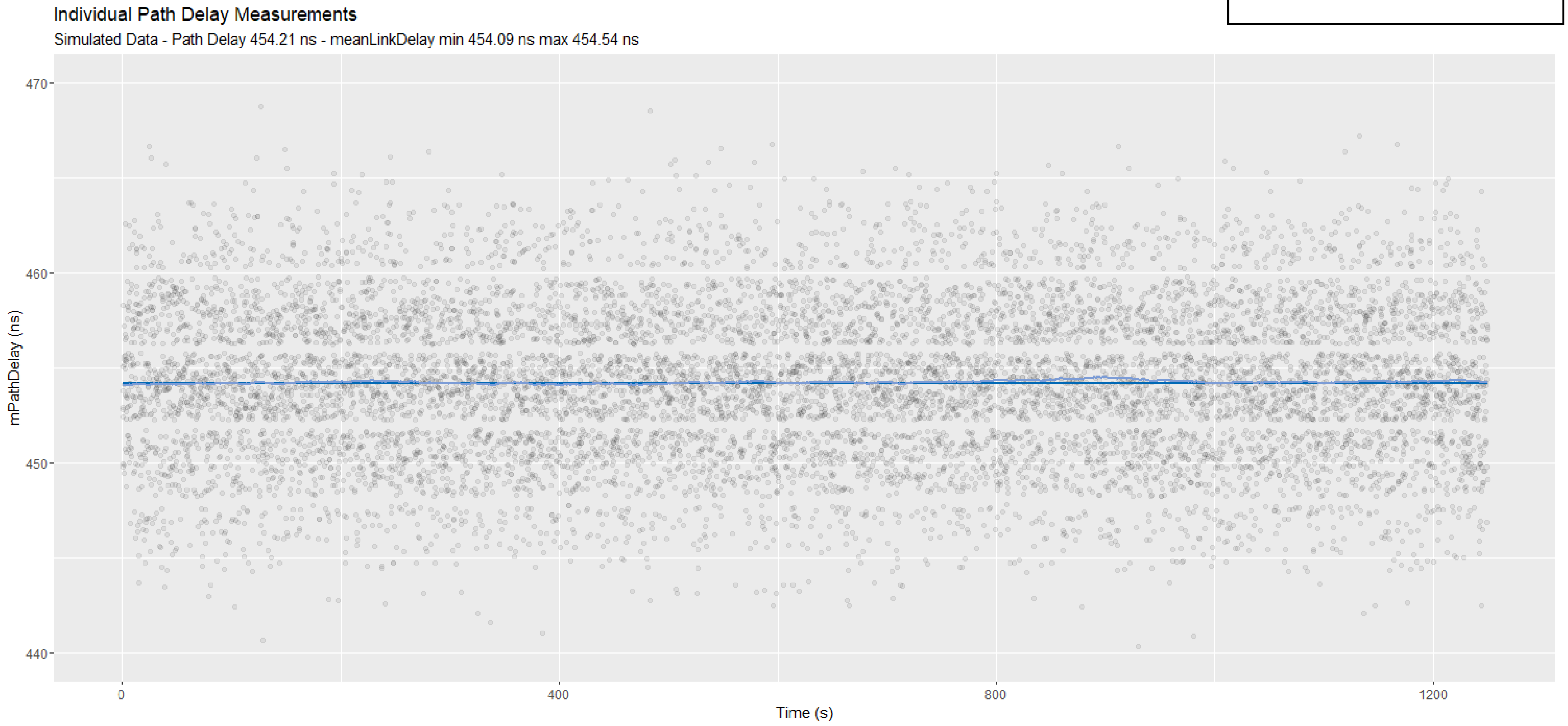
meanLinkDelay



n-1 node +0.5 ppm

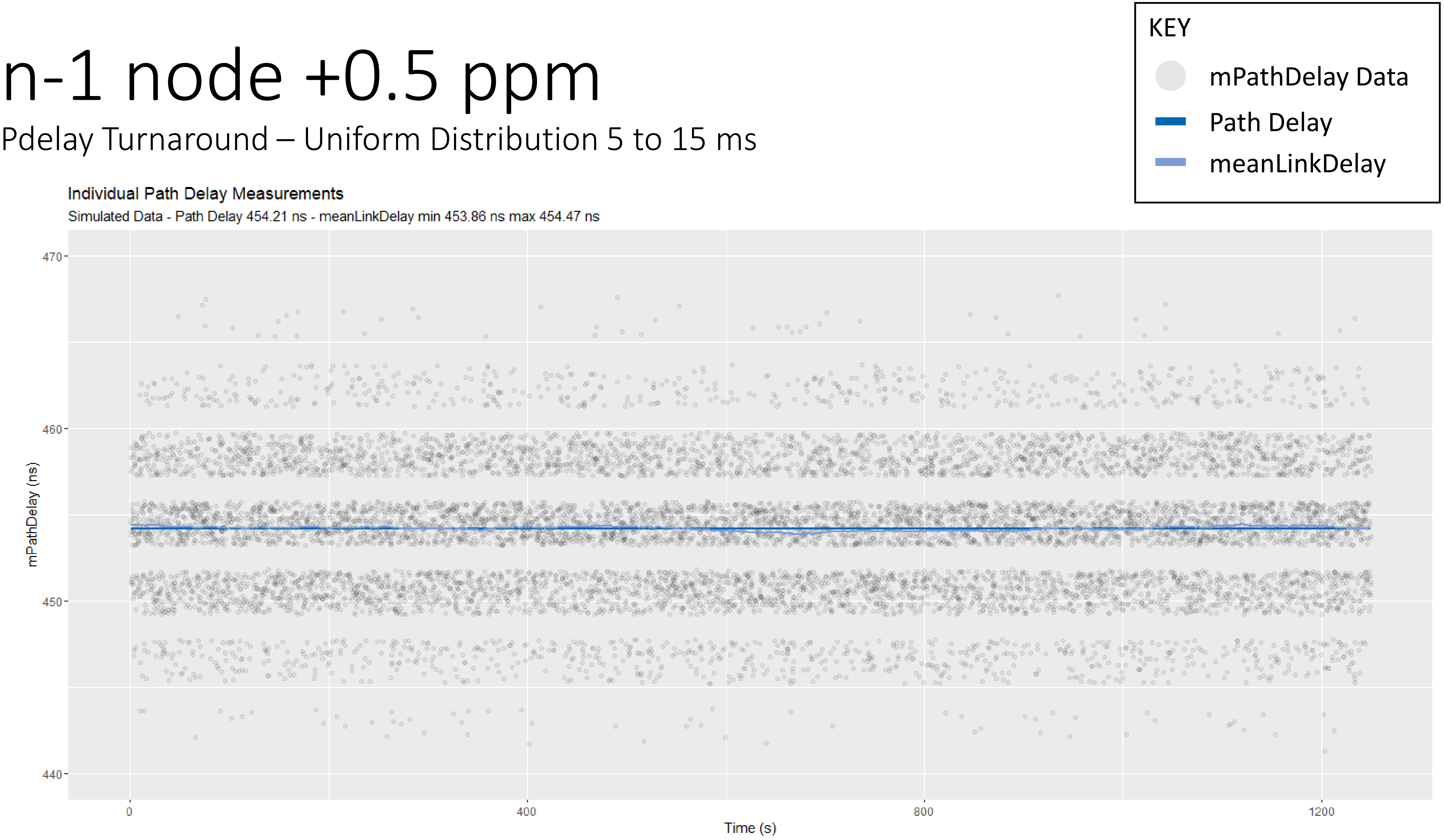
Pdelay Turnaround – Uniform Distribution 1 to 15 ms

KEY
- mPathDelay Data
- Path Delay
- meanLinkDelay



n-1 node +0.5 ppm

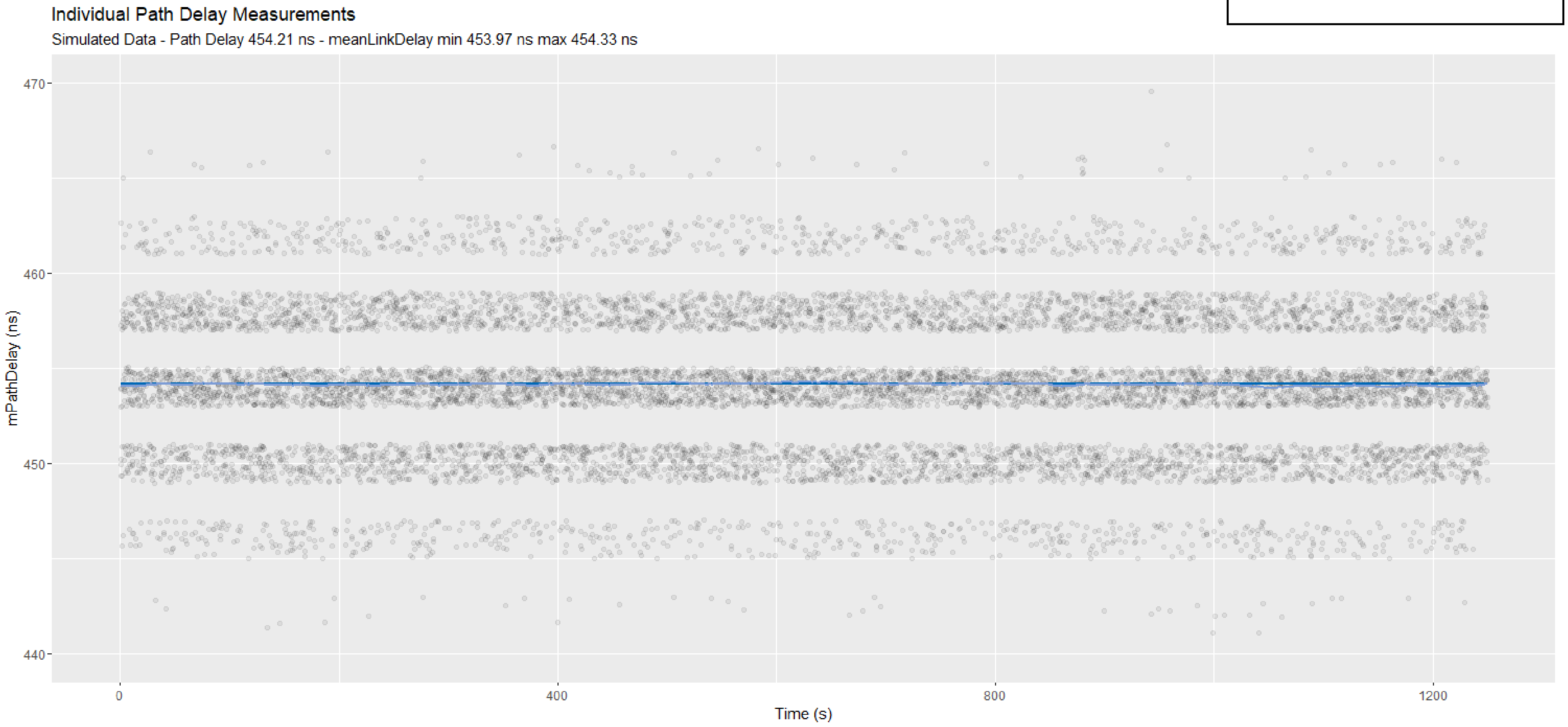
Pdelay Turnaround – Uniform Distribution 5 to 15 ms



n-1 node +0.4 ppm

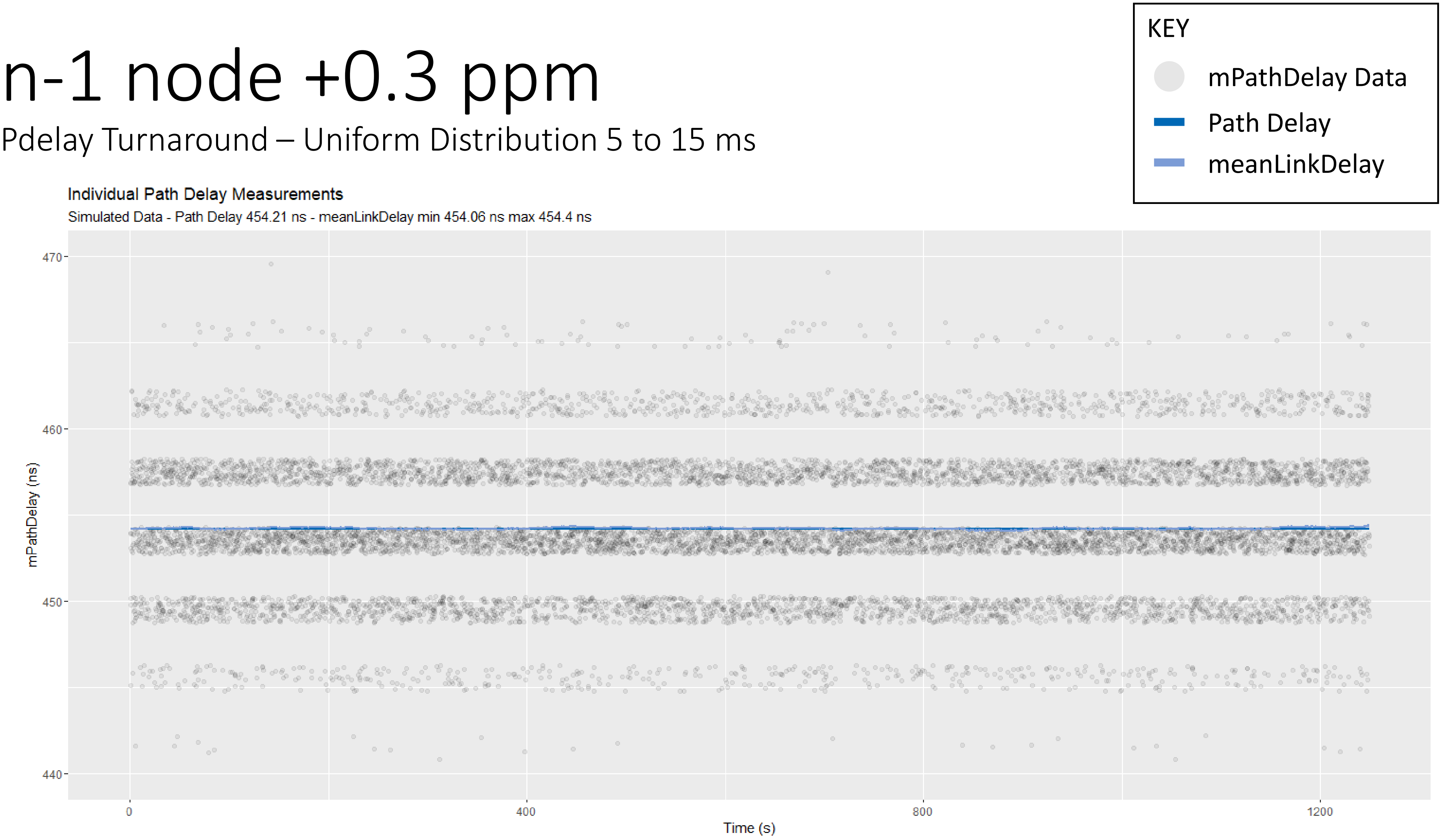
Pdelay Turnaround – Uniform Distribution 5 to 15 ms

KEY
- mPathDelay Data
- Path Delay
- meanLinkDelay



n-1 node +0.3 ppm

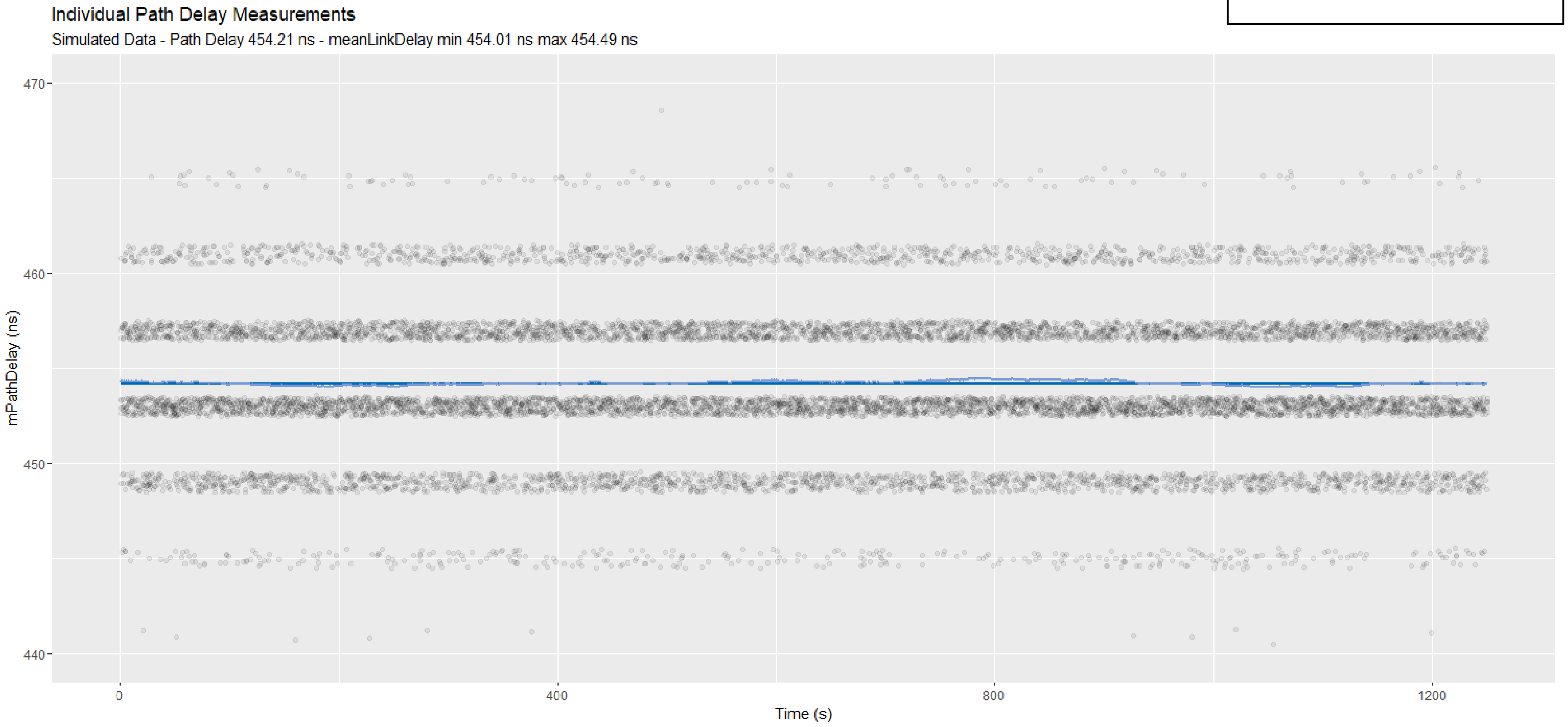
Pdelay Turnaround – Uniform Distribution 5 to 15 ms



n-1 node +0.2 ppm

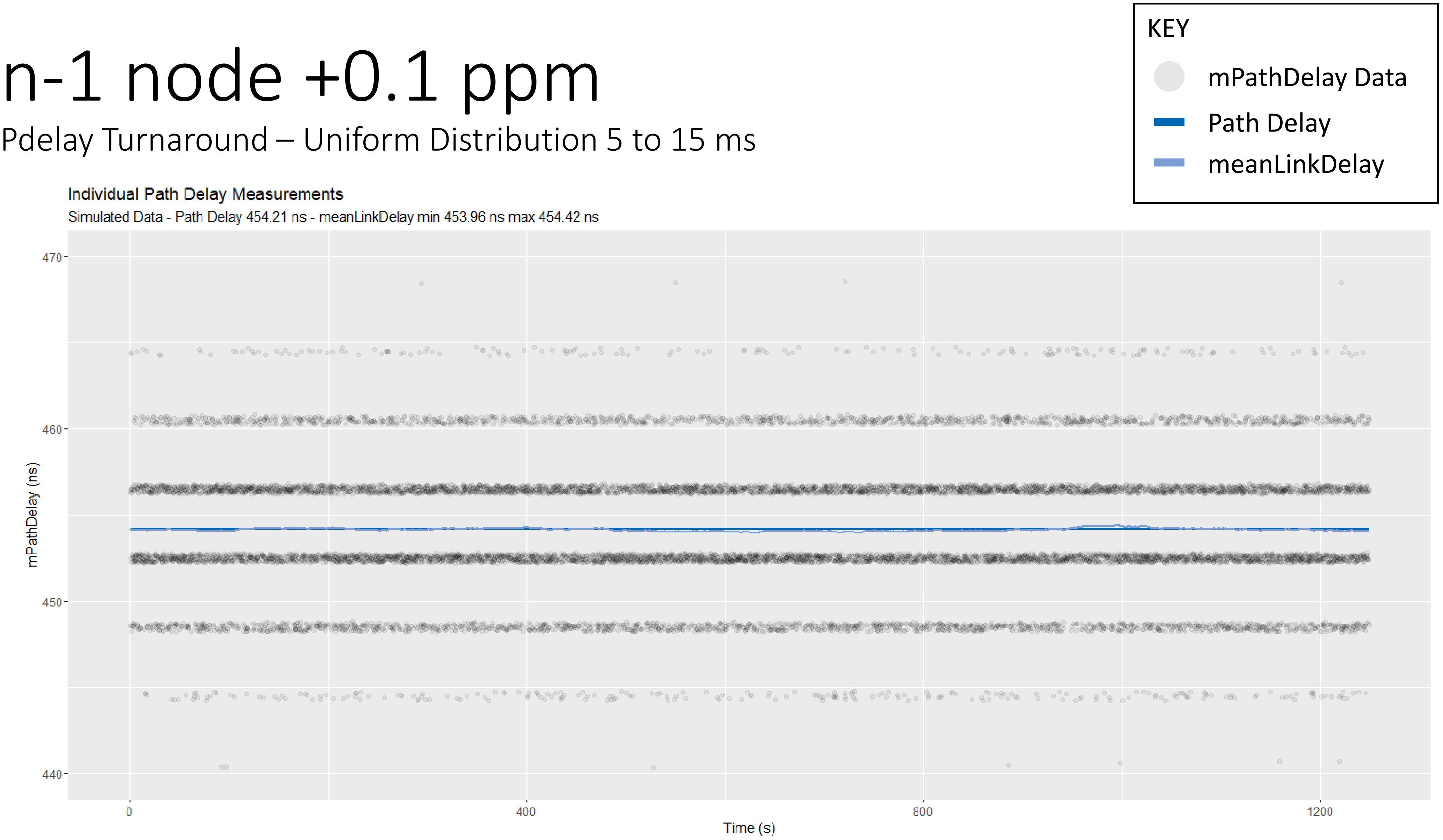
Pdelay Turnaround – Uniform Distribution 5 to 15 ms

KEY
- mPathDelay Data
- Path Delay
- meanLinkDelay



n-1 node +0.1 ppm

Pdelay Turnaround – Uniform Distribution 5 to 15 ms

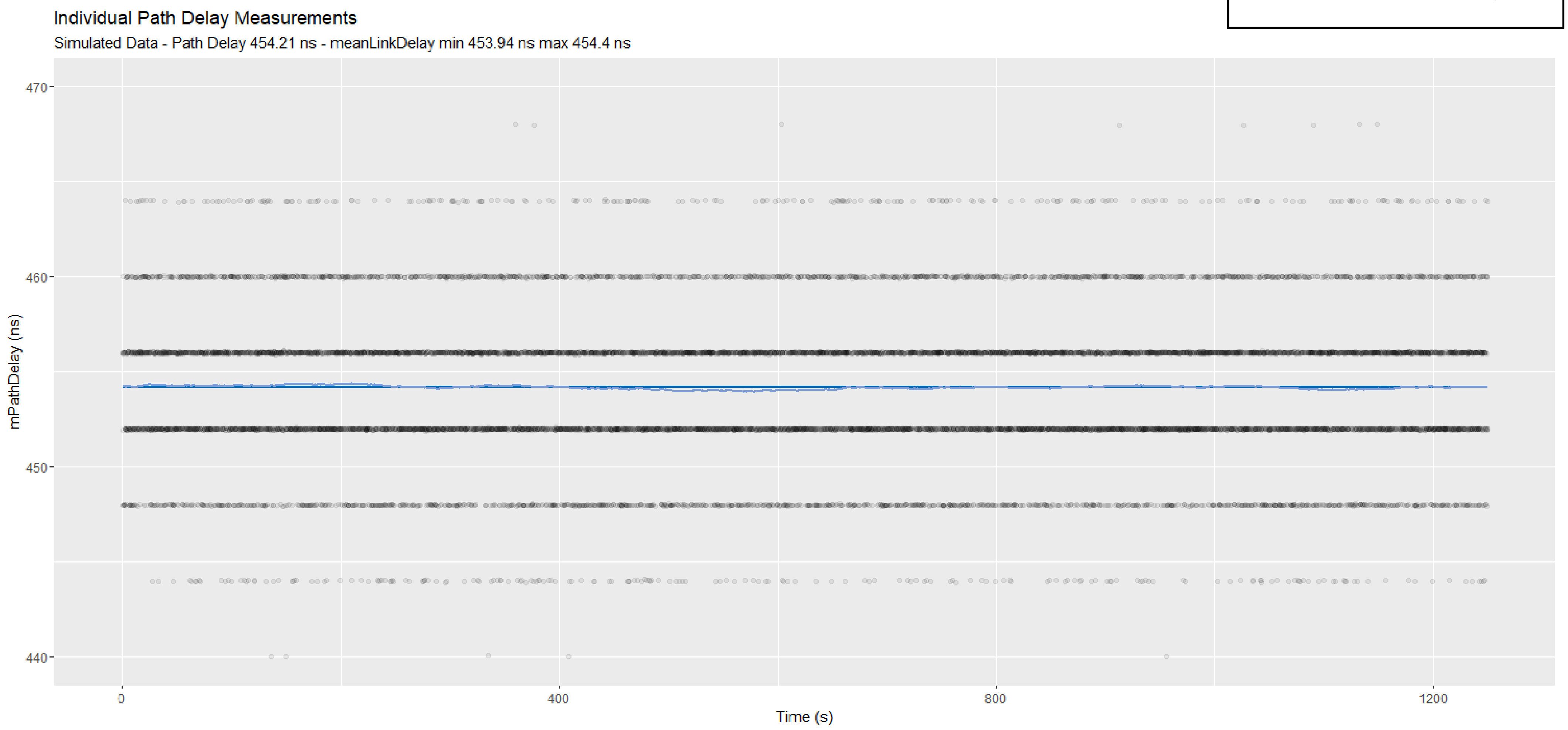


No Clock Offsets

Pdelay Turnaround – Uniform Distribution 5 to 15 ms

KEY

- mPathDelay Data
- Path Delay
- meanLinkDelay



Observations

- The combination of TSGE, DTSE, Pdelay Turnaround distribution and Clock Offsets determines the quantisation steps and distribution around those steps of $mPathDelay$ measurements.
- The underlying distribution of mPathDelay measurements around the actual Path Delay value remains the same, and is determined by DTSE and TSGE.
- Regardless of the above factors, taking a long “average” a described in Clause D.5.7, yields stable meanLinkDelay values well within the normative requirement of ±3 ns

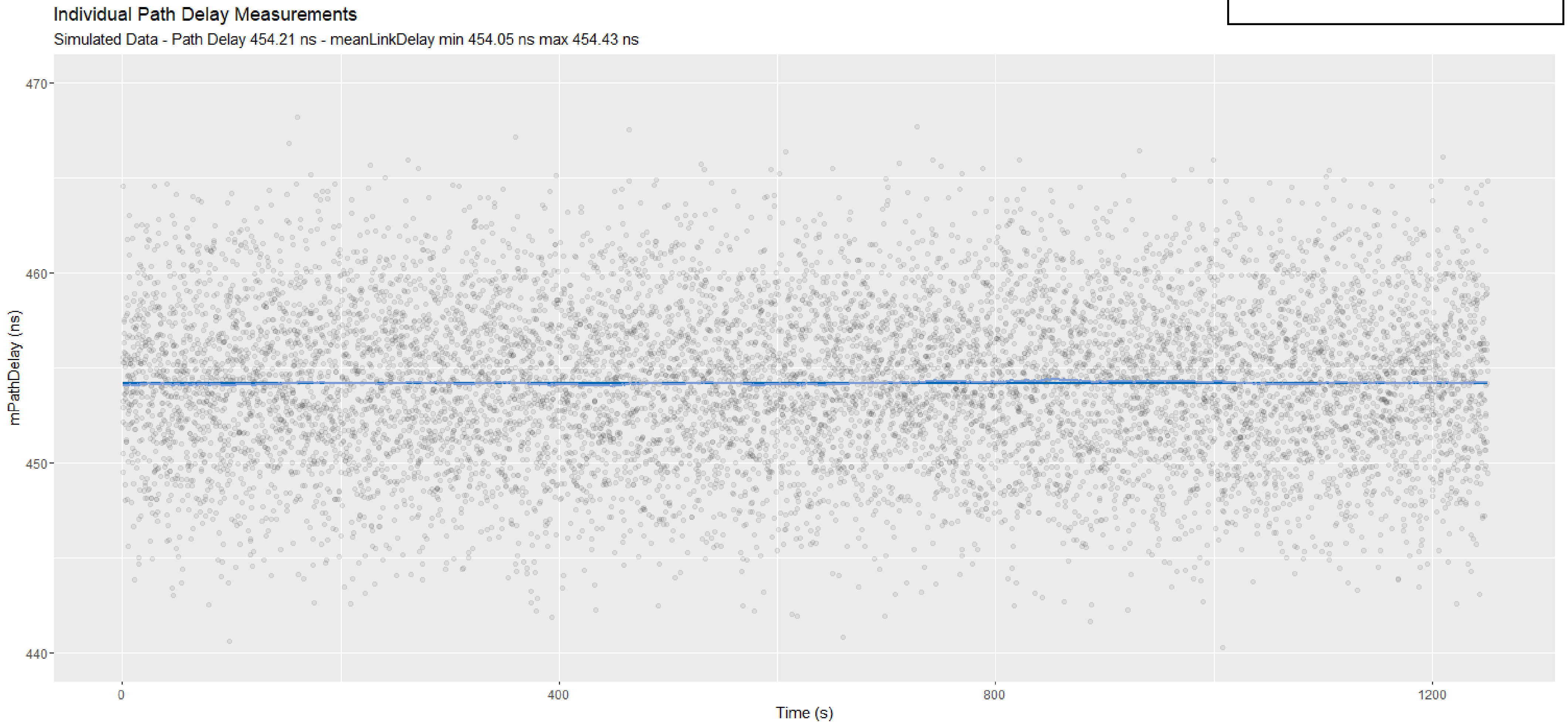
n-1 node +0.5 ppm

Pdelay Turnaround – Truncated Normal Distribution

(µ =10; σ=1.8; truncated 1 to Pdelay Turnaround – Truncated Normal Distribution15 ms)

KEY

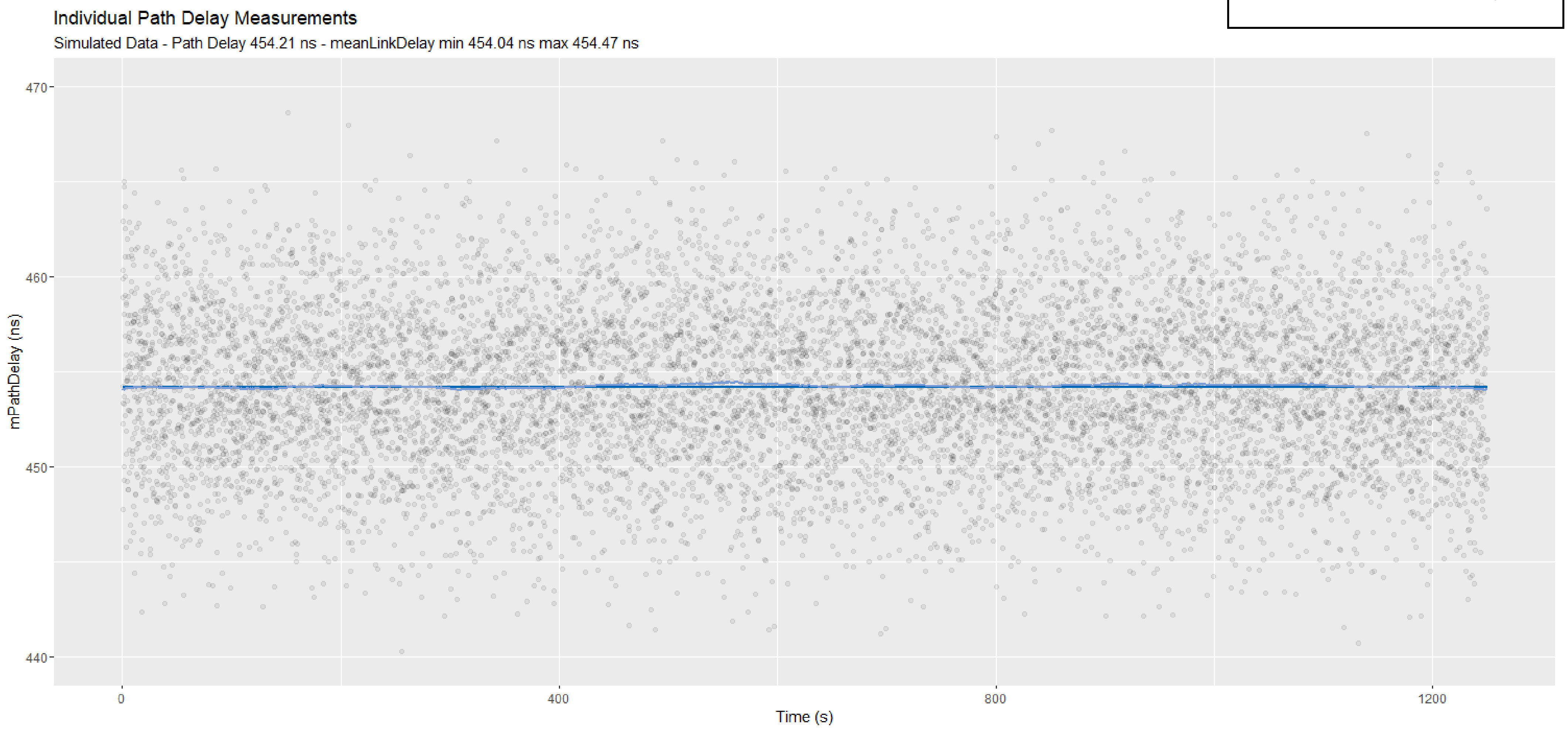
- mPathDelay Data
- Path Delay
- meanLinkDelay



n-1 node +0.5 ppm

Pdelay Turnaround – Uniform Distribution 5 to 15 ms

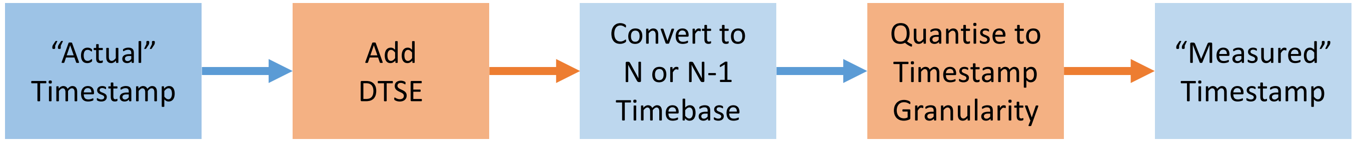
KEY
- mPathDelay Data
- Path Delay
- meanLinkDelay



Simulations to Match Time Series Simulations

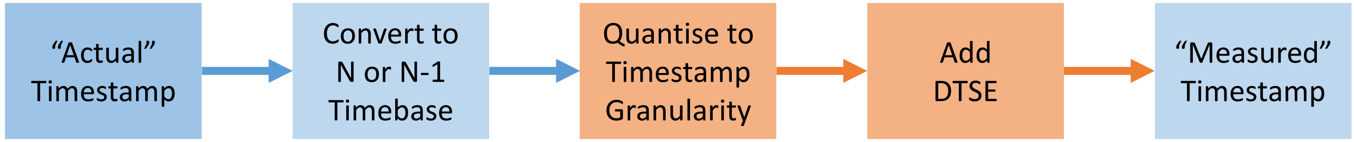
Difference

- Simulations in previous section do this…



…which matches most implementations.

- Time Series Simulations in [1] and [2] do this…



…which the RStudio simulation can also match.

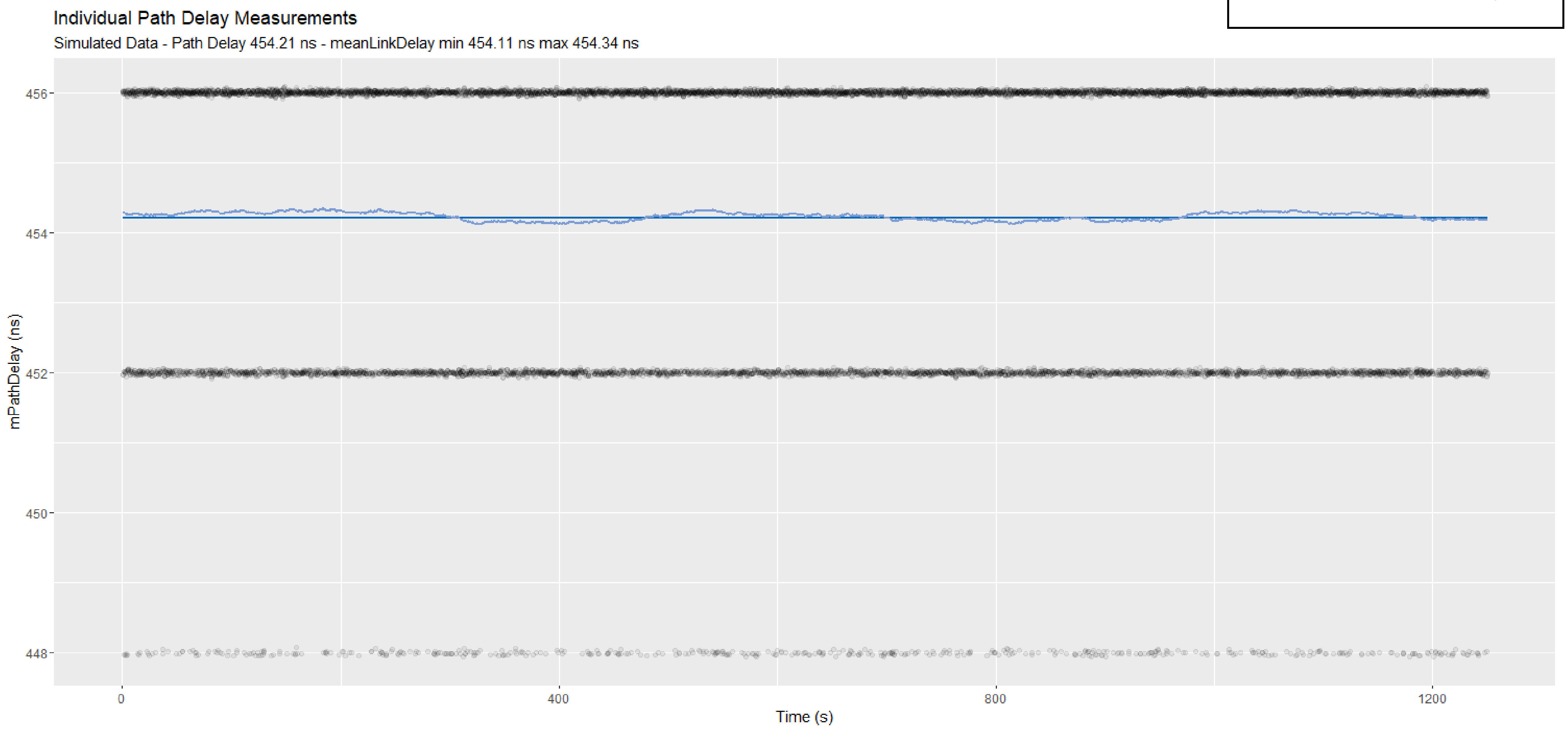
No DTSE; No Clock Offset

KEY

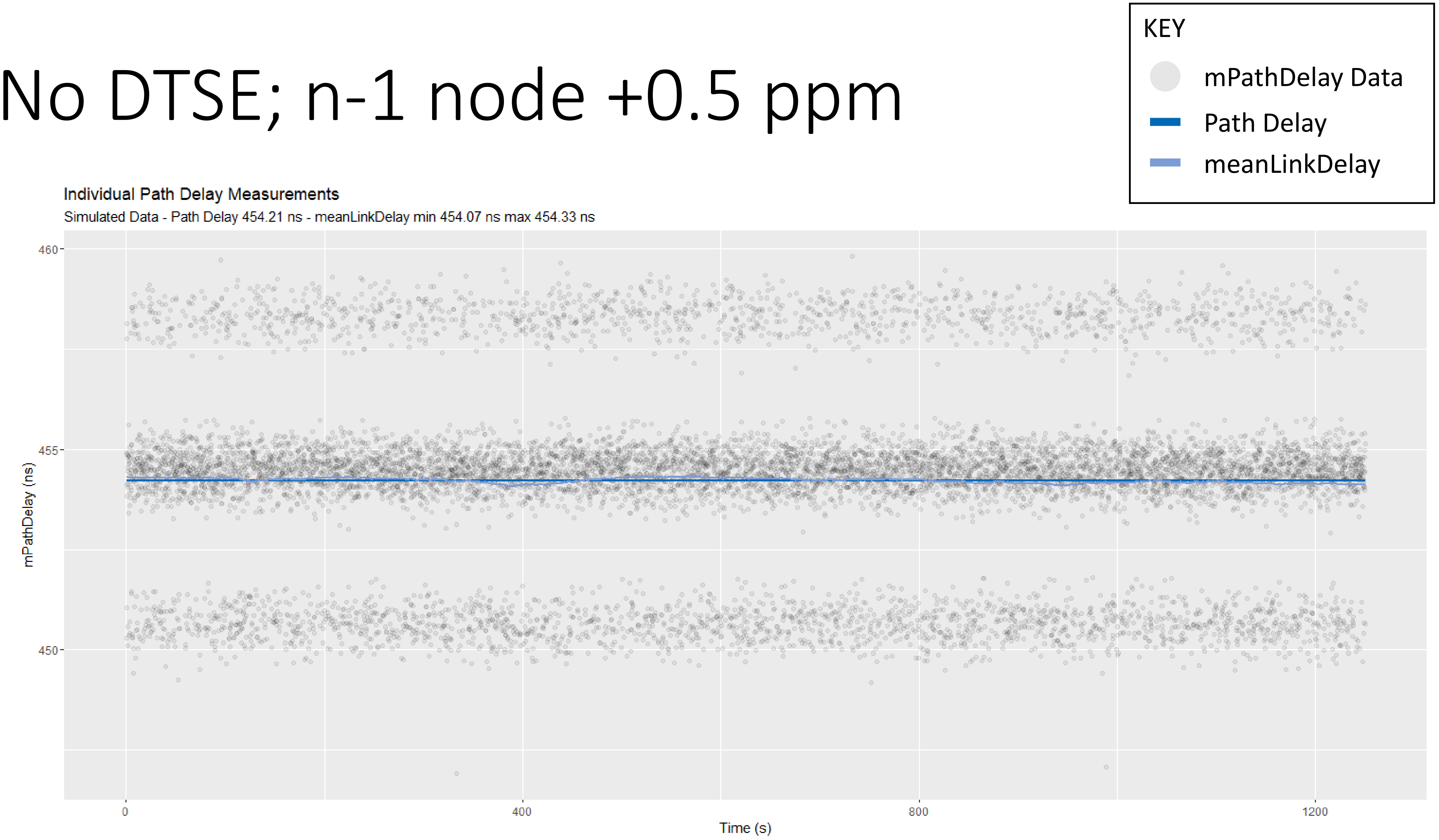
mPathDelay Data

Path Delay

meanLinkDelay

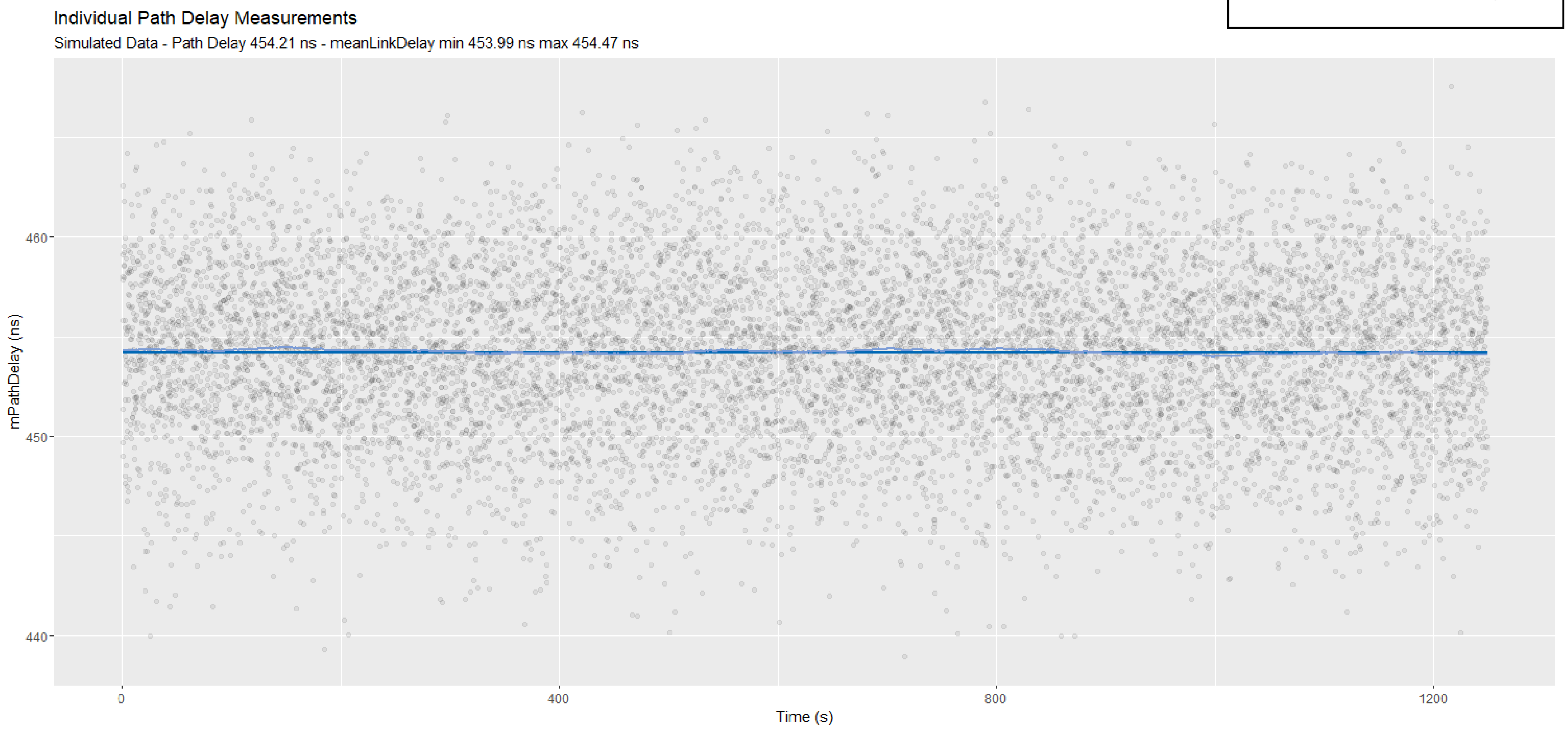


No DTSE; n-1 node +0.5 ppm

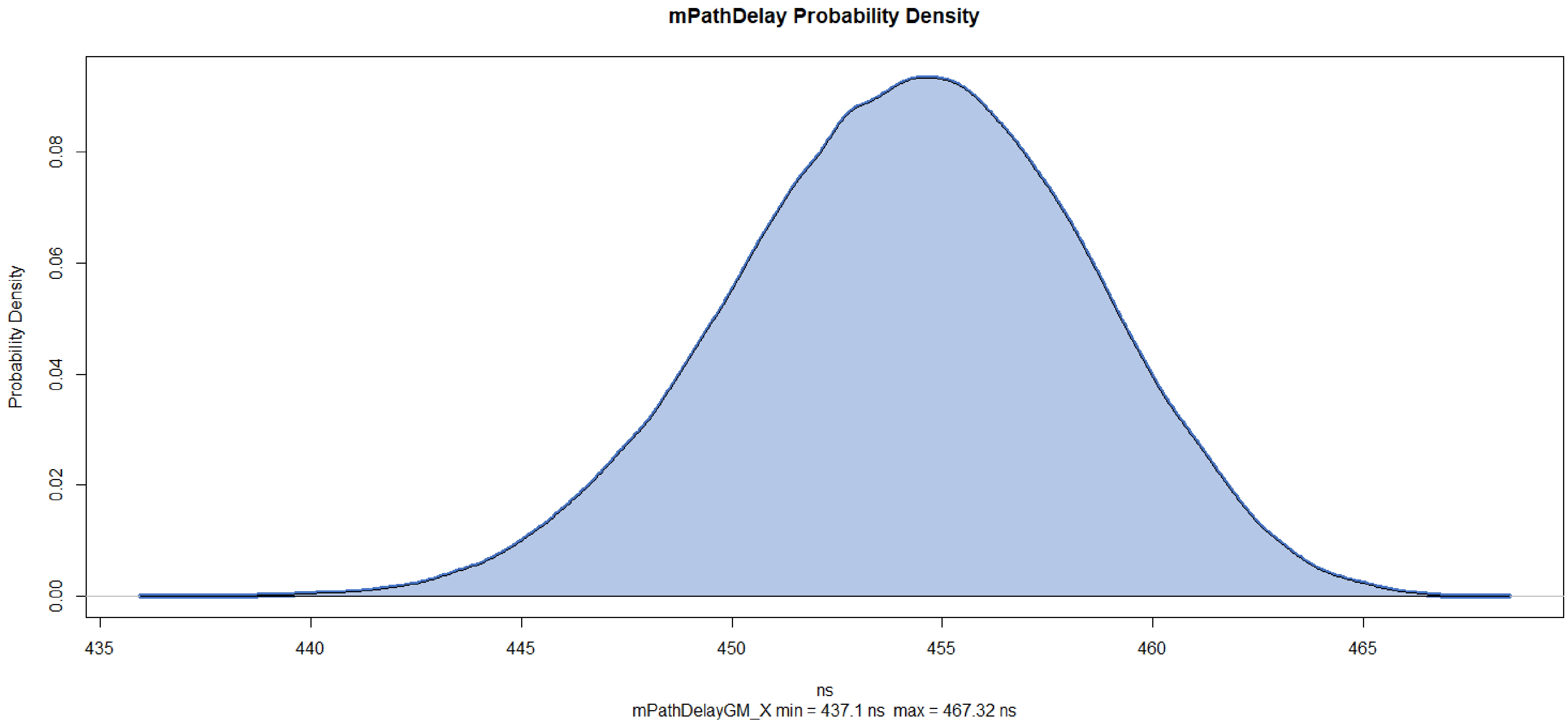


±6 ns DTSE; No Clock Offset

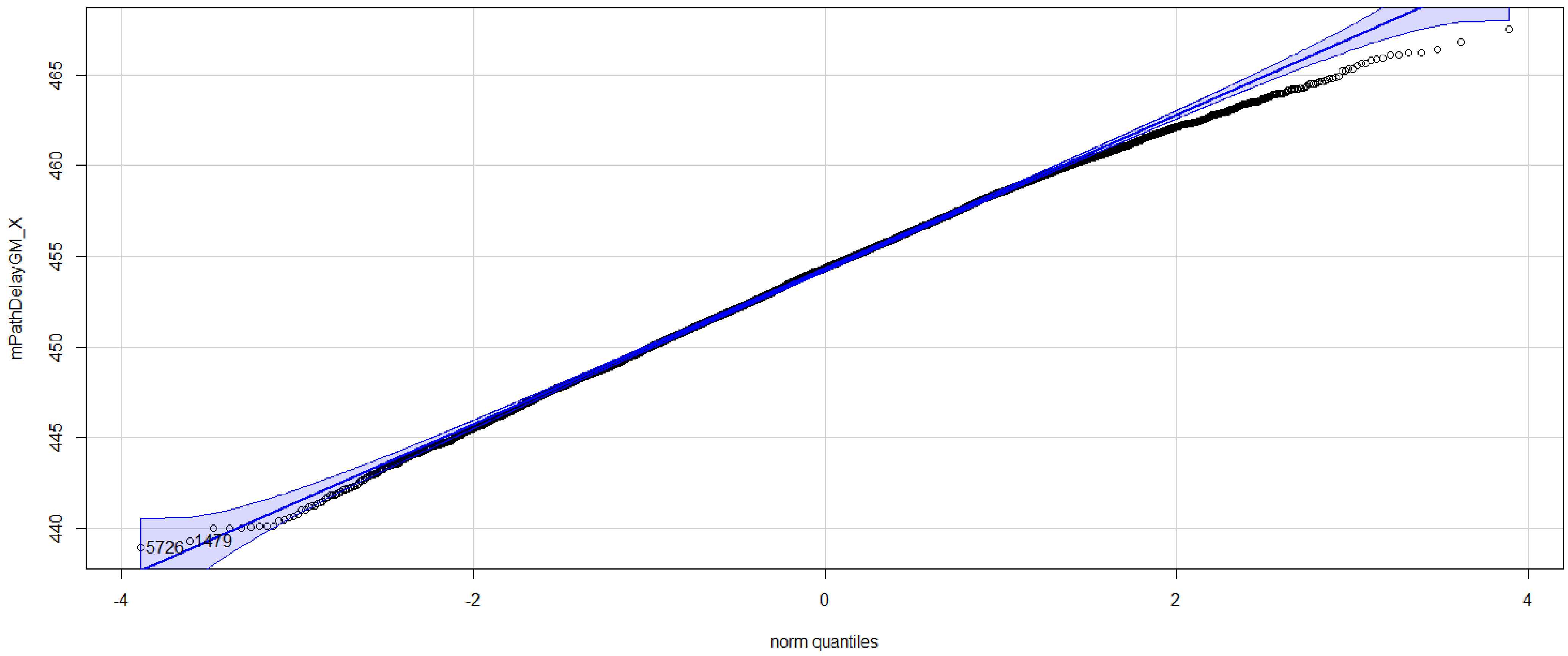
KEY
- mPathDelay Data
- Path Delay
- meanLinkDelay



±6 ns DTSE; No Clock Offset – Probability Density (100,000 Messages)

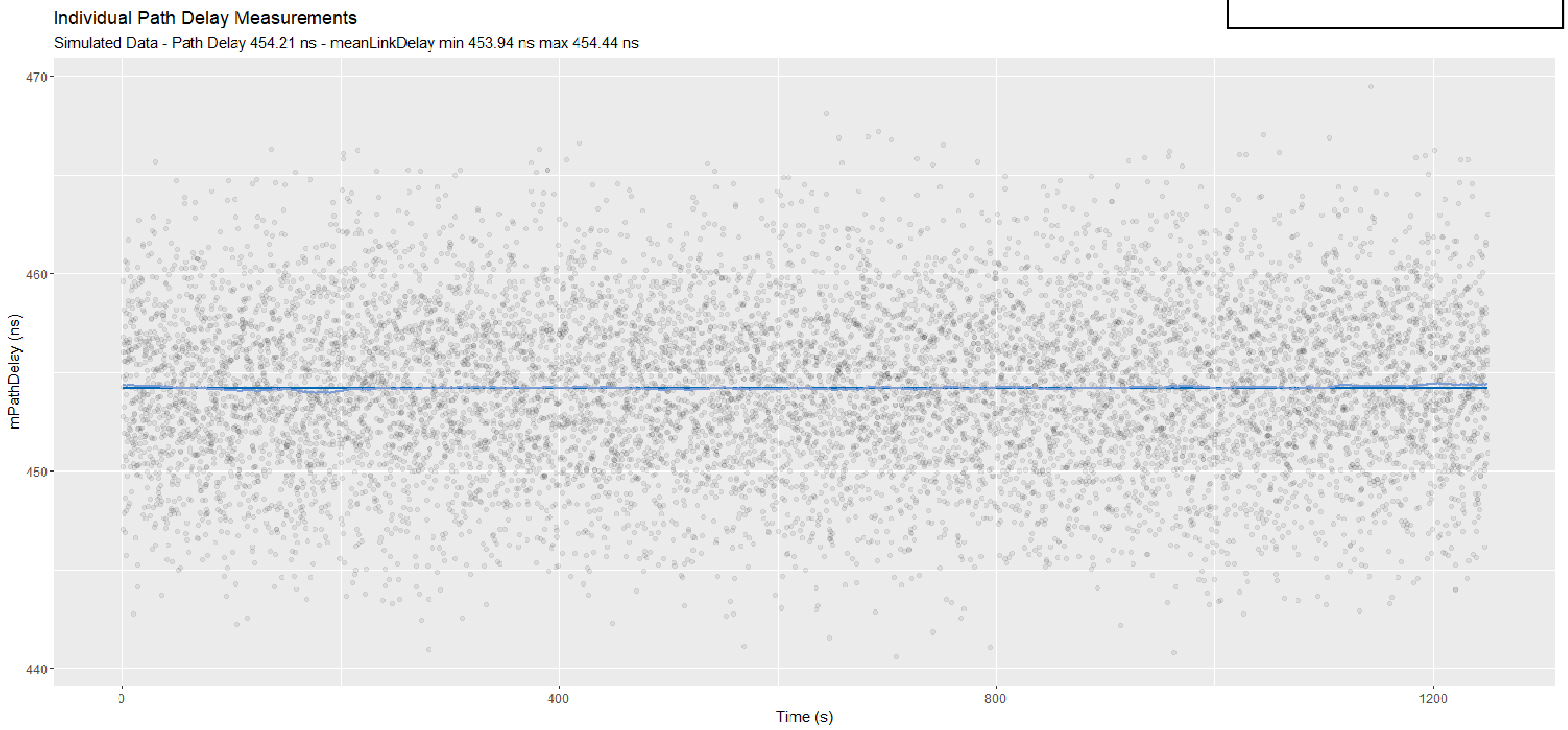


±6 ns DTSE; No Clock Offset – QQ Plot

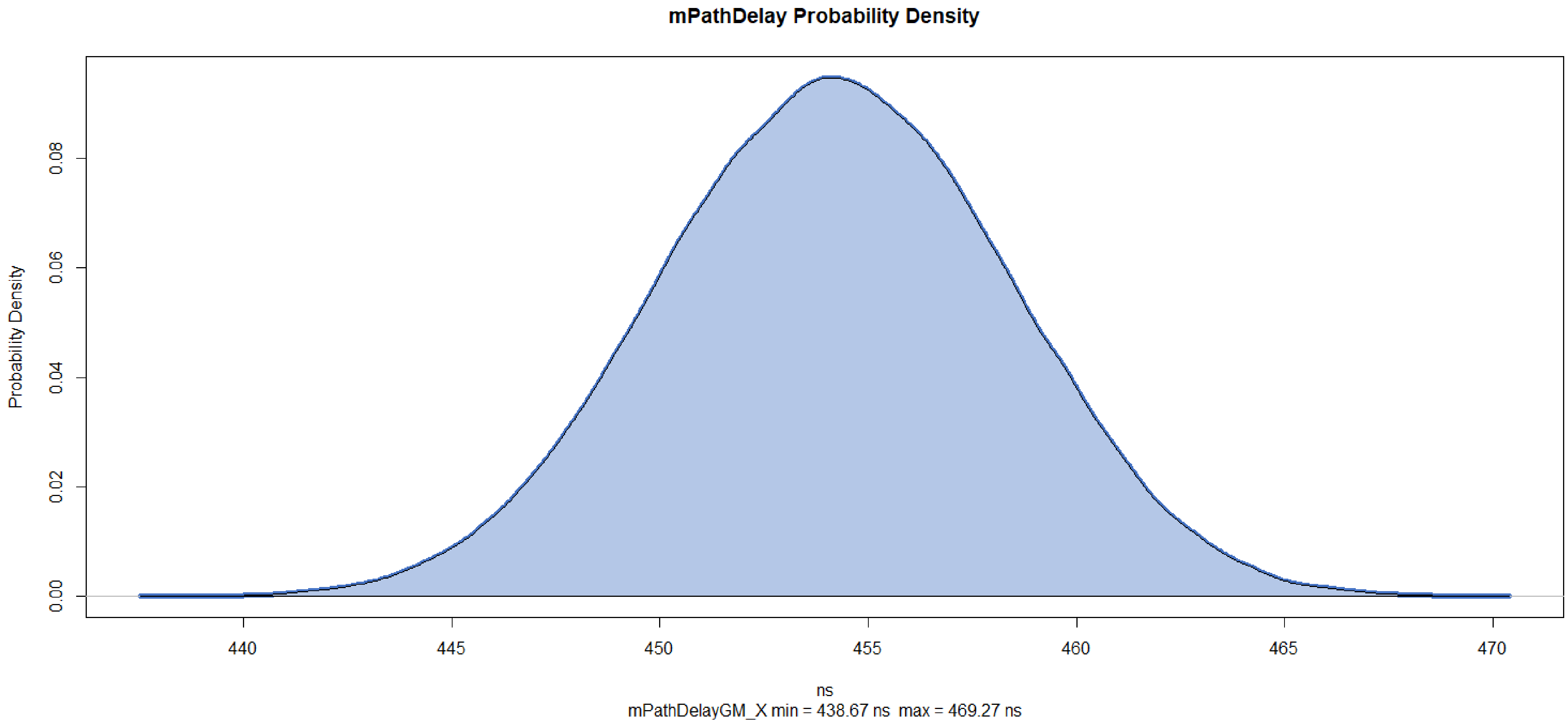


±6 ns DTSE; n-1 node +0.5 ppm

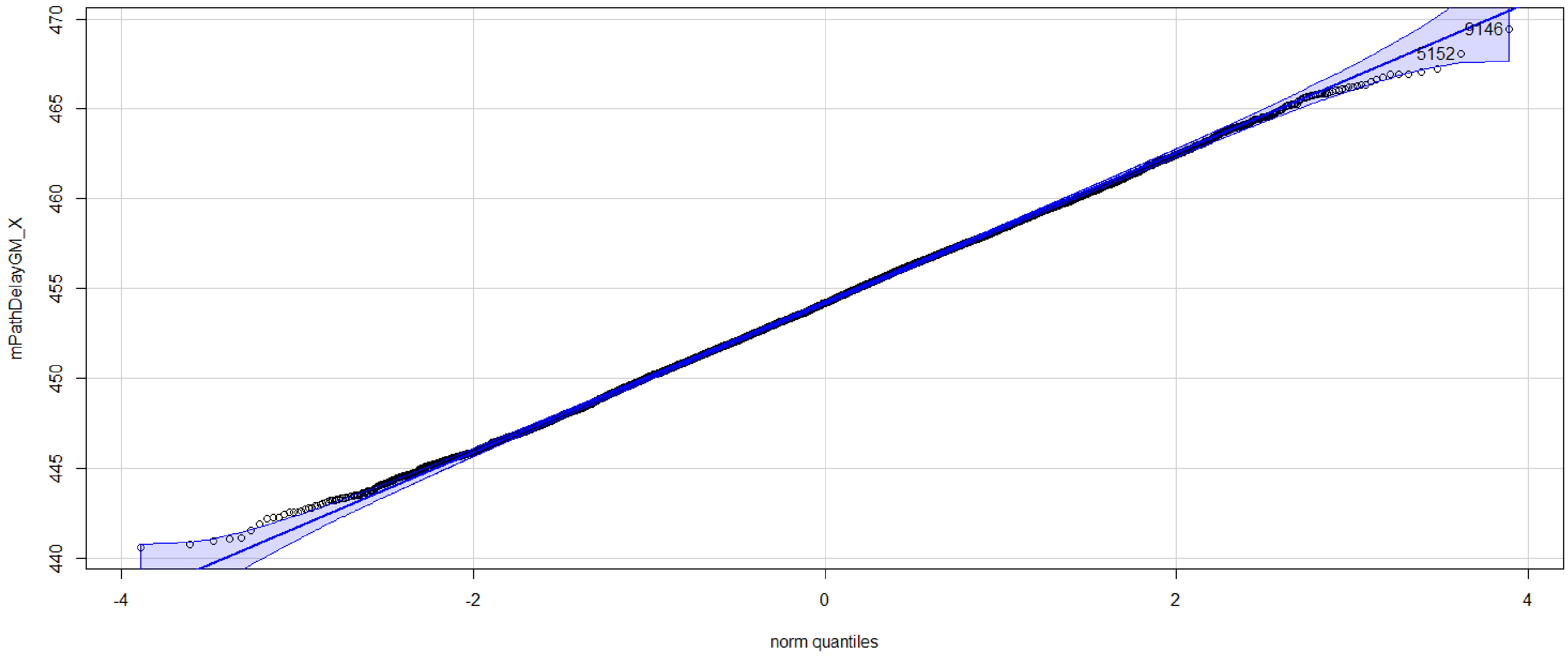
KEY
- mPathDelay Data
- Path Delay
- meanLinkDelay



±6 ns DTSE; n-1 node +0.5 ppm – Probability Density (100,000 Messages)



±6 ns DTSE; n-1 node +0.5 ppm – QQ Plot



Observations & Recommendation

- RStudio simulation shows some similarities with [1] and [2]
  - Quantisation steps due to TSGE with distribution either side of steps due to DTSE.
- But no evidence of “sticking” on a particular step for any length of time with consequent visible step changes
- Without step changes, normative requirements on meanLinkDelay are easily met.
  - This is regardless of whether Timestamp Granularity is applied before or after Dynamic Timestamp Error
- Currently there is no theory for why [1] and [2] exhibit the step-change behaviour. (Especially what causes a move from one step to another.)
- **Recommendation: do not alter meanLinkDelay normative requirements from d2.2 values.**

Thank you