# 60802 Dynamic Time Sync Error – NRR Medians, Algorithms & Analysis Validation

David McCall & Kevin Stanton (Intel)

January 2022 IEEE 802 – 802.1 TSN – IEEE/IEC 60802

# Abstract

- Industrial Automation Systems require microsecond-accurate time across long daisy-chains of devices using IEEE Std. 802.1AS™-2020 as specified by IEEE/IEC 60802.
- Simulated protocol and system parameters have thus far either been judged impractical or have failed to meet the time-accuracy requirement.
- An analysis of how errors accumulate suggested that a Monte Carlo method analysis could support fast iteration of potential scenarios and deliver insights into cause and effect. See...
  - 60802-McCall-et-al-Time-Sync-Error-Model-0921-v03.pdf
  - 60802-McCall-Stanton-Time-Sync-Error-Model-and-Analysis-2021-11-v02.pdf
- In this contribution we:
  - Revisit the effect of taking a median of past NRR calculations
  - Introduce approaches for reducing Dynamic Time Error via algorithms that...
    - Average pDelay measurements when calculating Mean Link Delay
    - Compensate for Clock Drift
  - Raise questions about how the use of such algorithms might be required in the specification
  - Propose next steps for validating Monte Carlo analysis results against Time Series Simulations

# Content

- Background & Recap
- Neighbor Rate Ratio "Smoothing"
  - Calculating NRR using previous pDelayResp timestamps & using a median of previous calculations
- Algorithms
  - Mean Link Delay
  - Clock Drift Compensation
  - How to include algorithm requirements in the specification?
- Next Steps
  - Including Validating Monte Carlo Analysis against Time Series Simulations

# Background & Recap

# Background & Recap – 1

- Monte Carlo Analysis seems to be a useful tool to gain insights into how Dynamic Time Errors accumulate and develop an approach to achieving the group's goal of 1ms accuracy over 100 hops
- From the presentation in November, optimisation of pDelay Interval and use of three algorithms seems to provide a good chance of sufficient accuracy. The three algorithms identified are...
  - NRR Smoothing – use of previous pDelayResp timestamps when calculating NRR
  - Averaging Mean Link Delay
  - Compensating for Clock Drift
- No details of the last two algorithms were discussed; they were modelled as a "% effective" at removing associated errors.

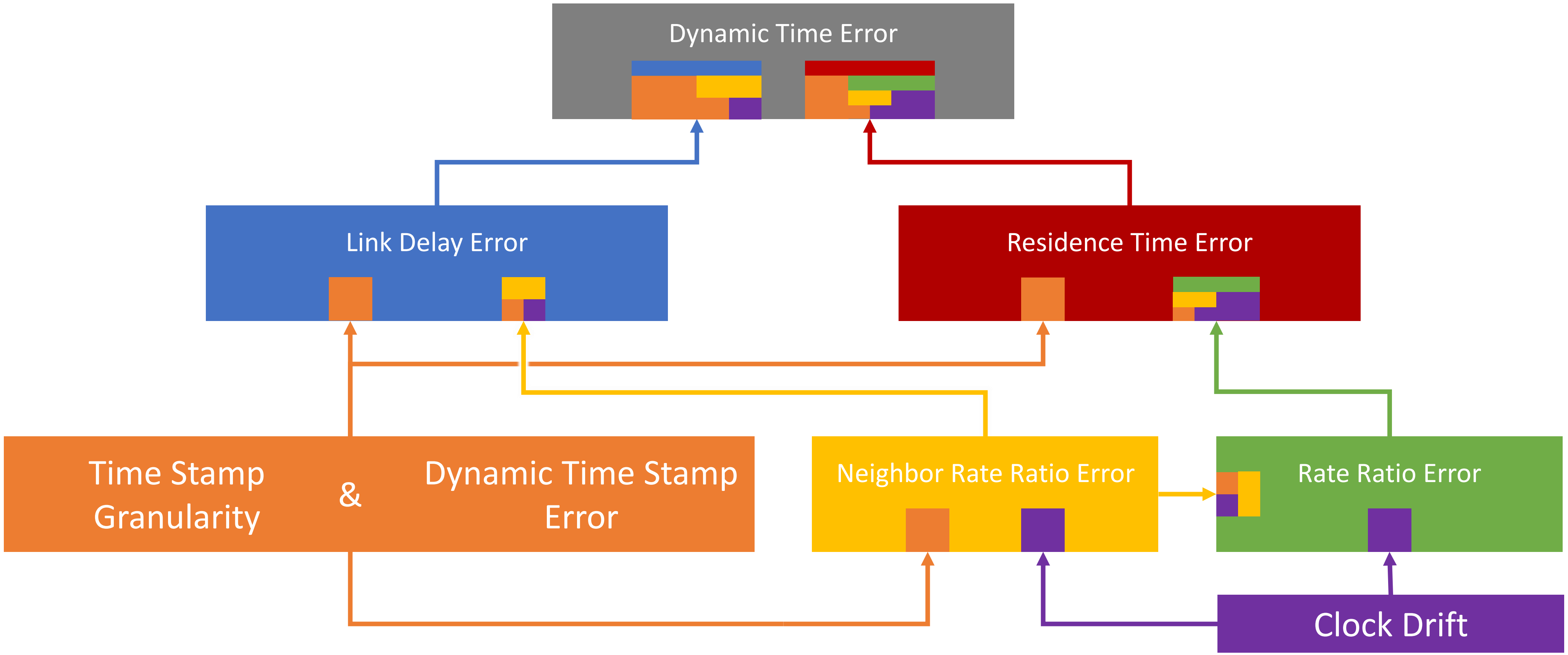
# Background & Recap – 2

- A second aspect of mNRR Smoothing was judged to be ineffective: use of a median of past NRR calculations.
  - This is **not** the case.
- The plan after the last presentation was for a small group to collaborate on deciding how to validate the Monte Carlo Analysis against the Time Series Simulation
  - This started...but has not progressed far due to need to include taking a median of past NRR calculations in the Monte Carlo Analysis and then examine its effect.

# Background & Recap – 3

- Hence three sections to this presentation...
  - Examination of taking a median of past NRR calculations
  - Introduction of potential algorithms for...
    - Mean Link Delay Averaging
    - Clock Drift Compensation
  - Proposal for Time Series Simulations to validate Monte Carlo Analysis

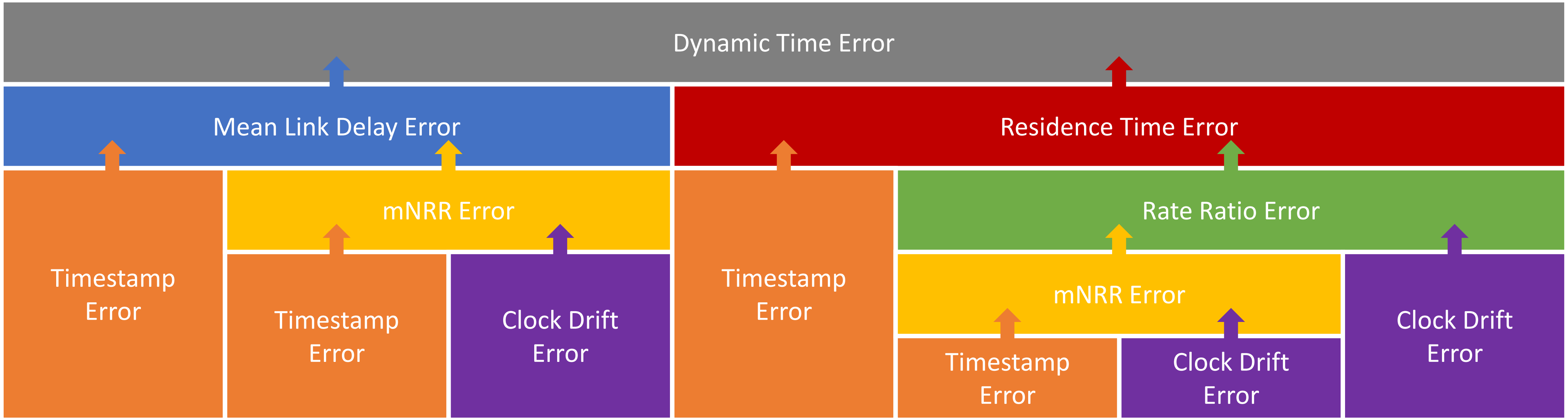
# Time Sync – How Errors Add Up




**All errors in this analysis are caused by either Clock Drift or Timestamp Errors**

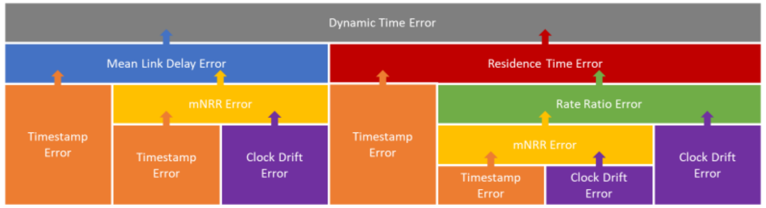
# Graphical Representation of Error Accumulation

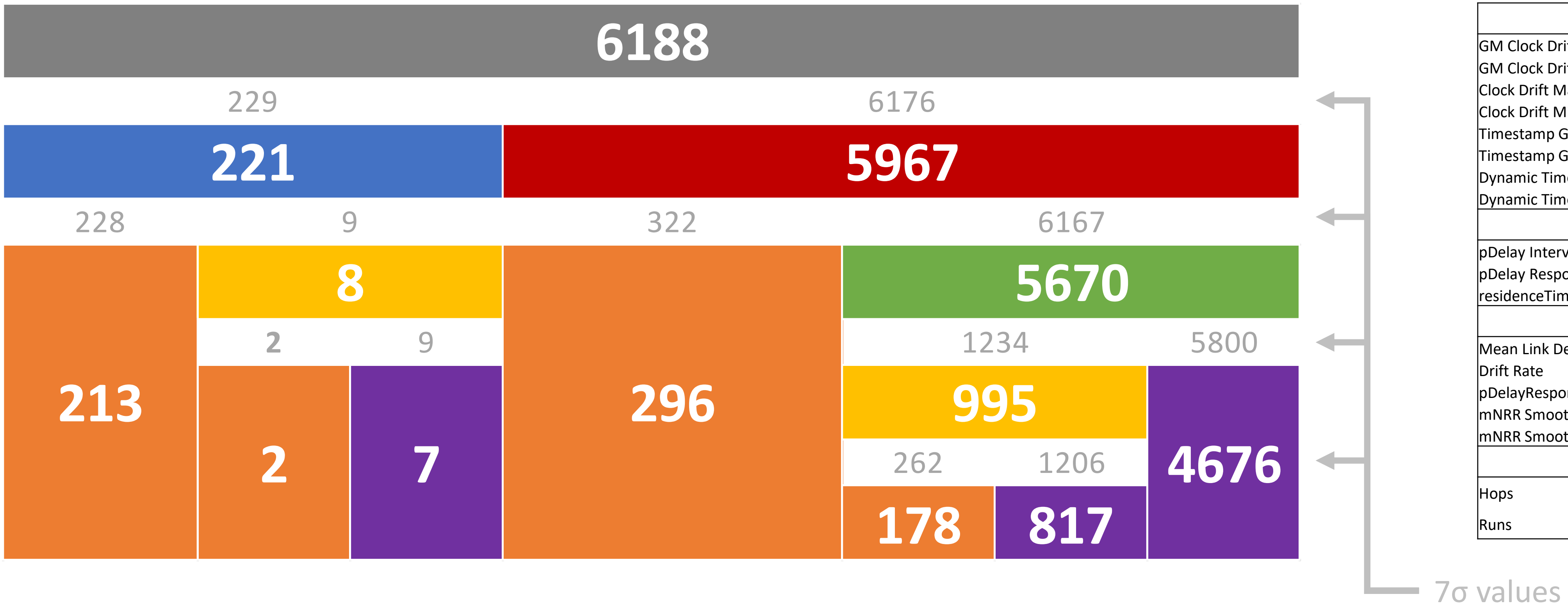
- Each error breaks down into two parts




- The relative weight of each part can be judged by their 7σ values

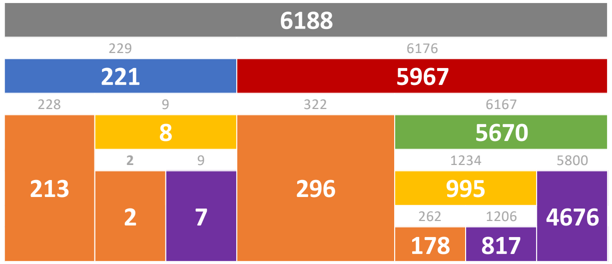
# Graphical Representation of Error Accumulation

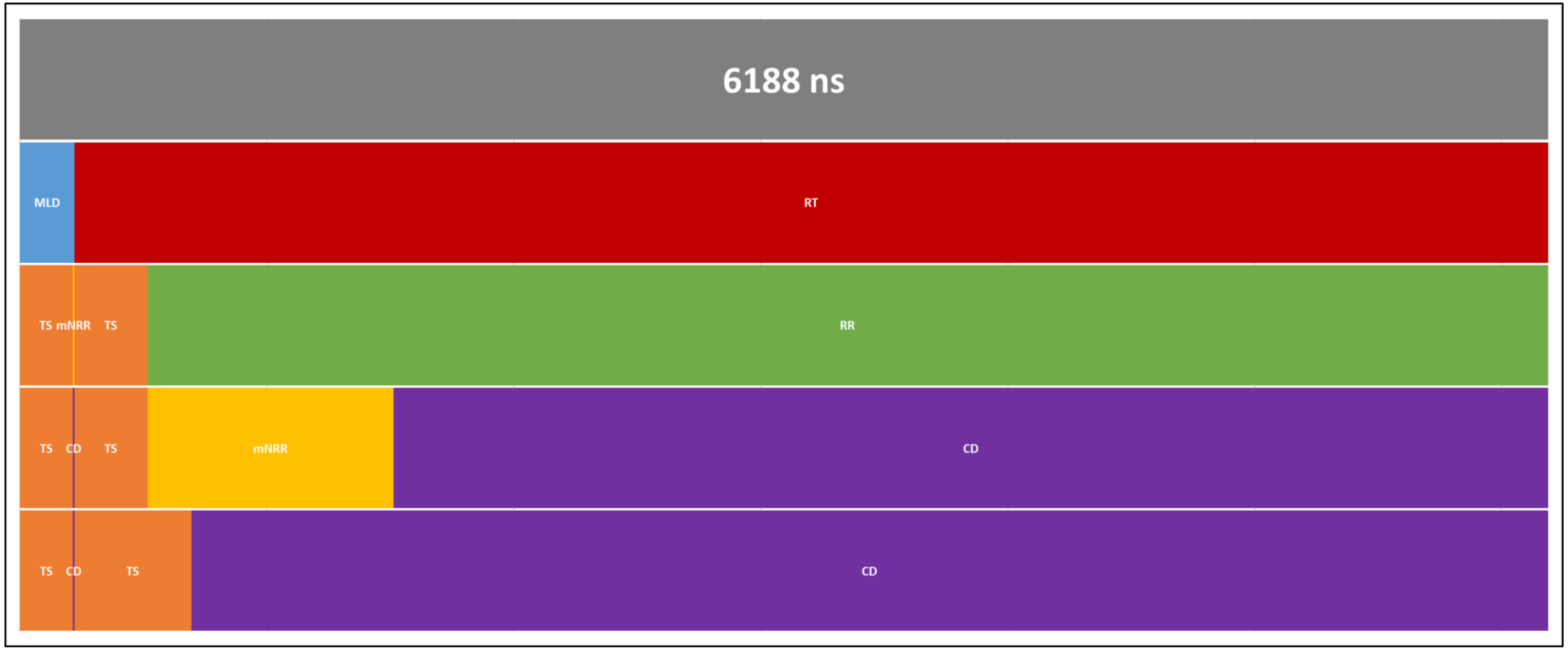



| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | -0.6 | ppm/s |
| Clock Drift Max (non-GM) | +0.6 | ppm/s |
| Clock Drift Min (non-GM) | -0.6 | ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | 1000 | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing N | 1 | |
| mNRR Smoothing M | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

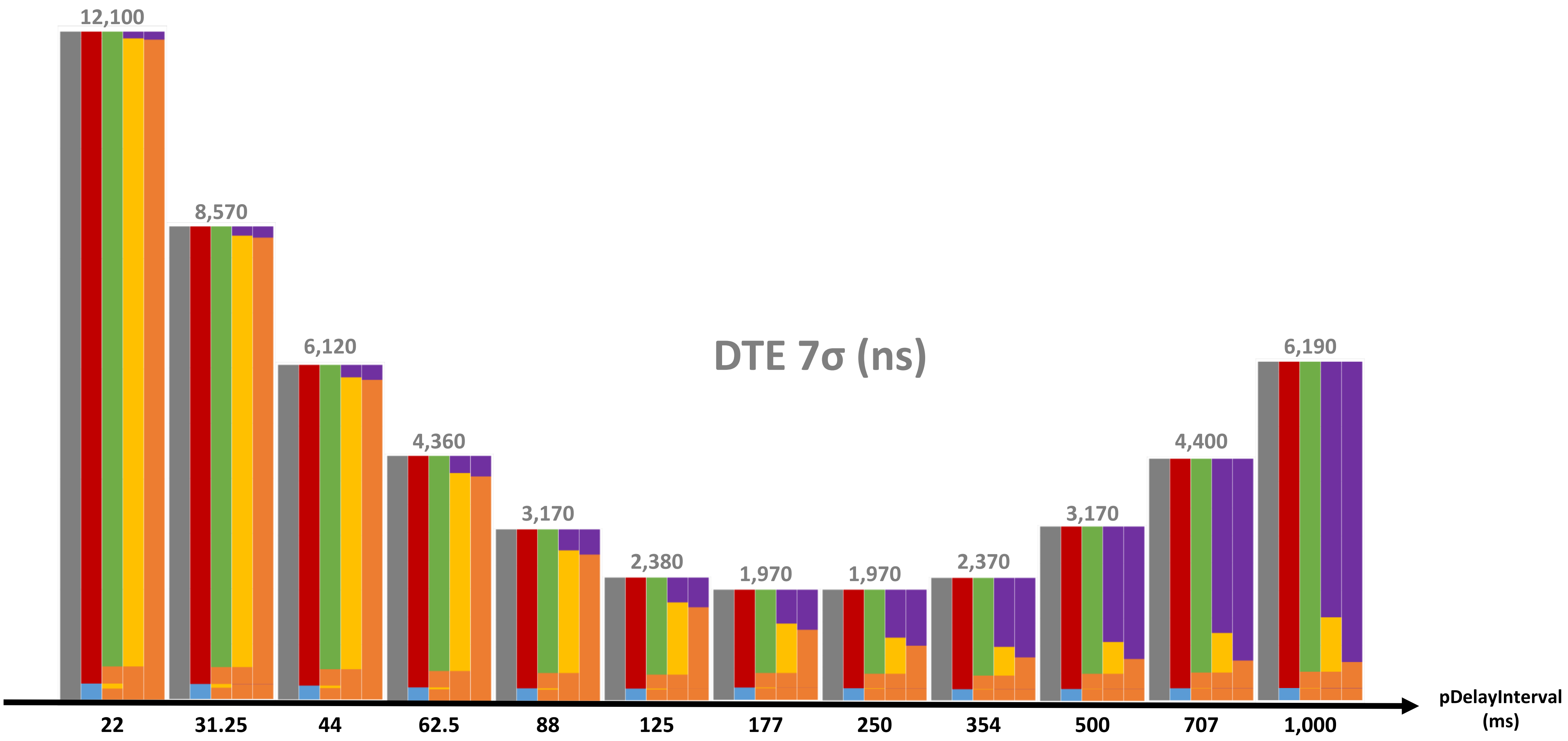
# Graphical Representation of Error Accumulation



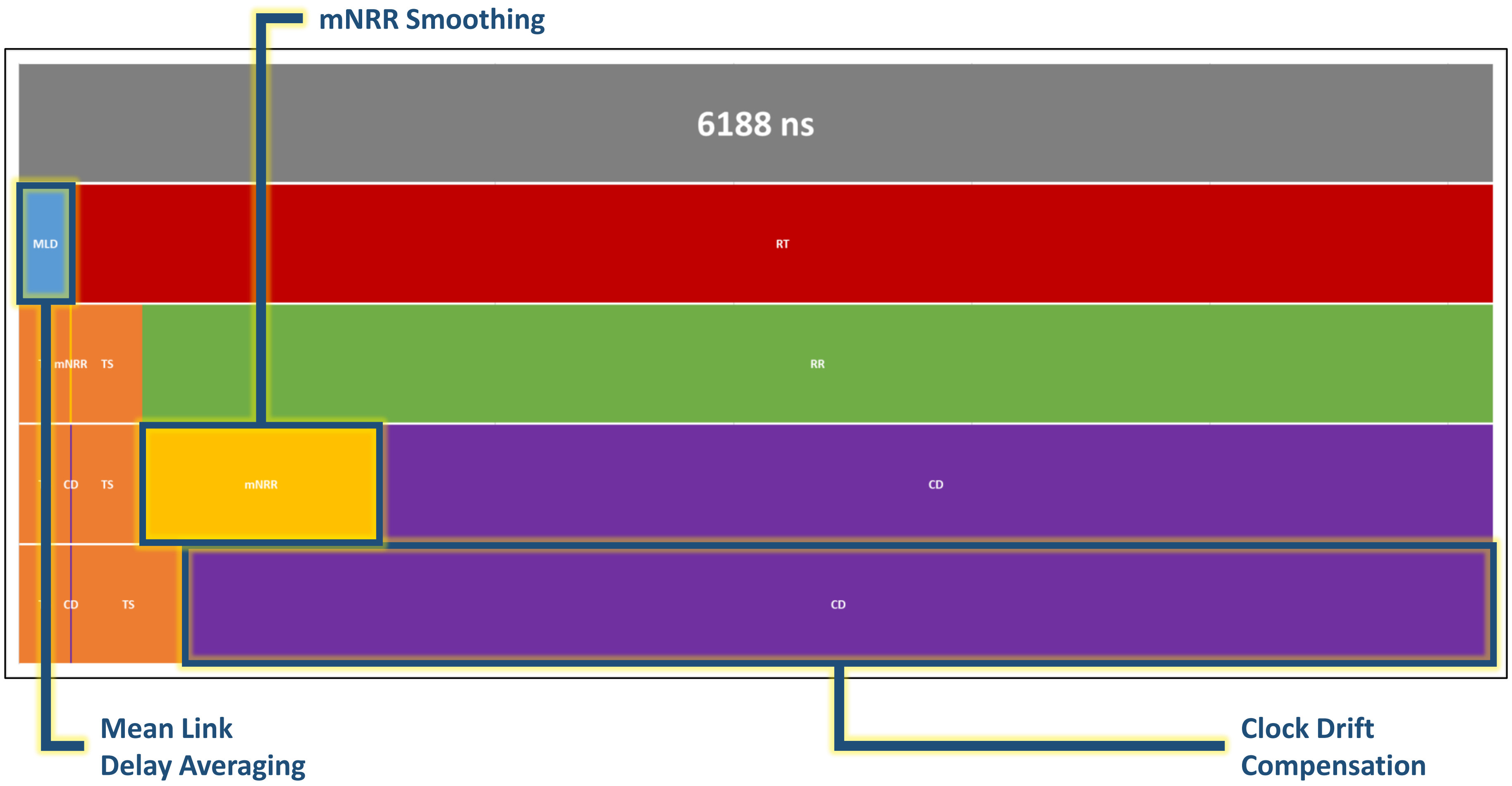




| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | -0.6 | ppm/s |
| Clock Drift Max (non-GM) | +0.6 | ppm/s |
| Clock Drift Min (non-GM) | -0.6 | ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | 1000 | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing N | 1 | |
| mNRR Smoothing M | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

# *pDelayInterval* Sensitivity Analysis

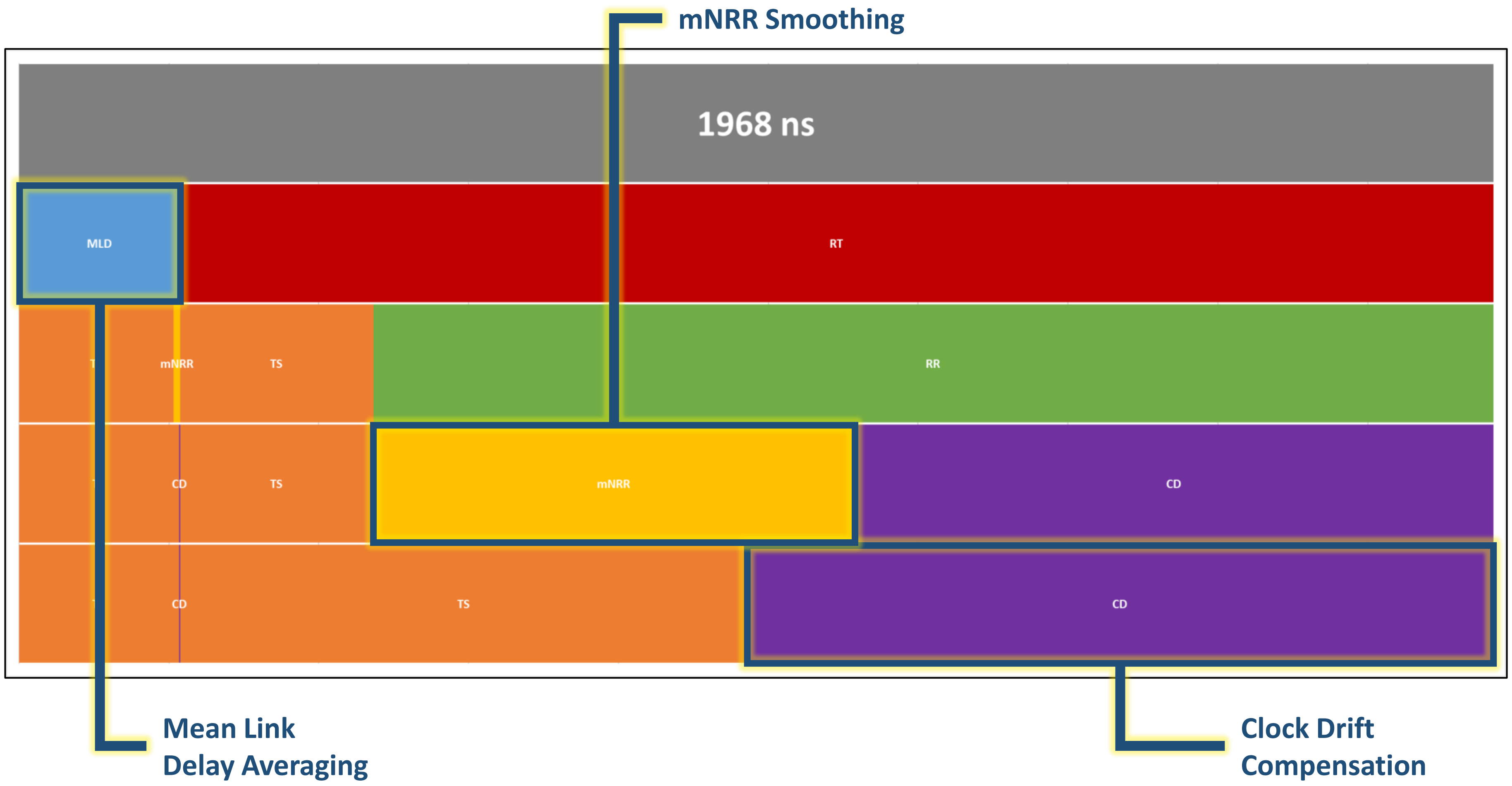


# Potential Algorithm Impact – pDelay Interval 1000ms



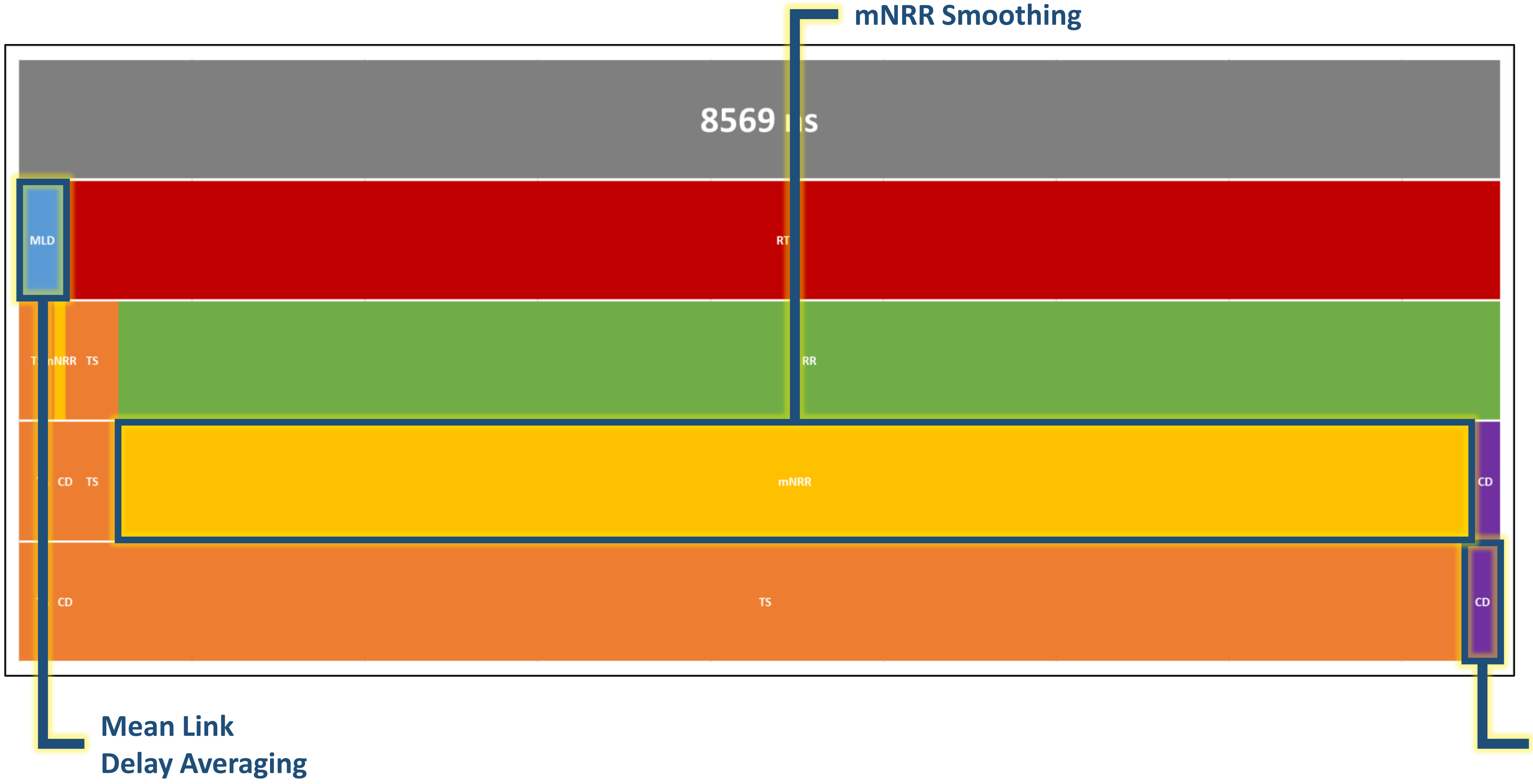

| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | -0.6 | ppm/s |
| Clock Drift Max (non-GM) | +0.6 | ppm/s |
| Clock Drift Min (non-GM) | -0.6 | ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | 1000 | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing N | 1 | |
| mNRR Smoothing M | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

# Potential Algorithm Impact – pDelay Interval 250ms




| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | -0.6 | ppm/s |
| Clock Drift Max (non-GM) | +0.6 | ppm/s |
| Clock Drift Min (non-GM) | -0.6 | ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | 250 | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing N | 1 | |
| mNRR Smoothing M | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

# Potential Algorithm Impact – pDelay Interval 31.25ms

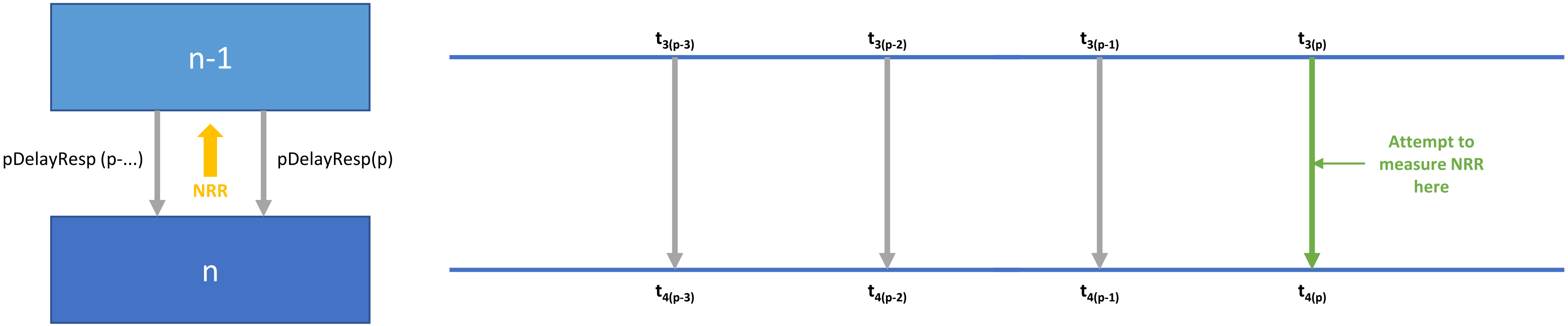



| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | -0.6 | ppm/s |
| Clock Drift Max (non-GM) | +0.6 | ppm/s |
| Clock Drift Min (non-GM) | -0.6 | ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | 31.25 | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing N | 1 | |
| mNRR Smoothing M | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

# Neighbor Rate Ratio “Smoothing”

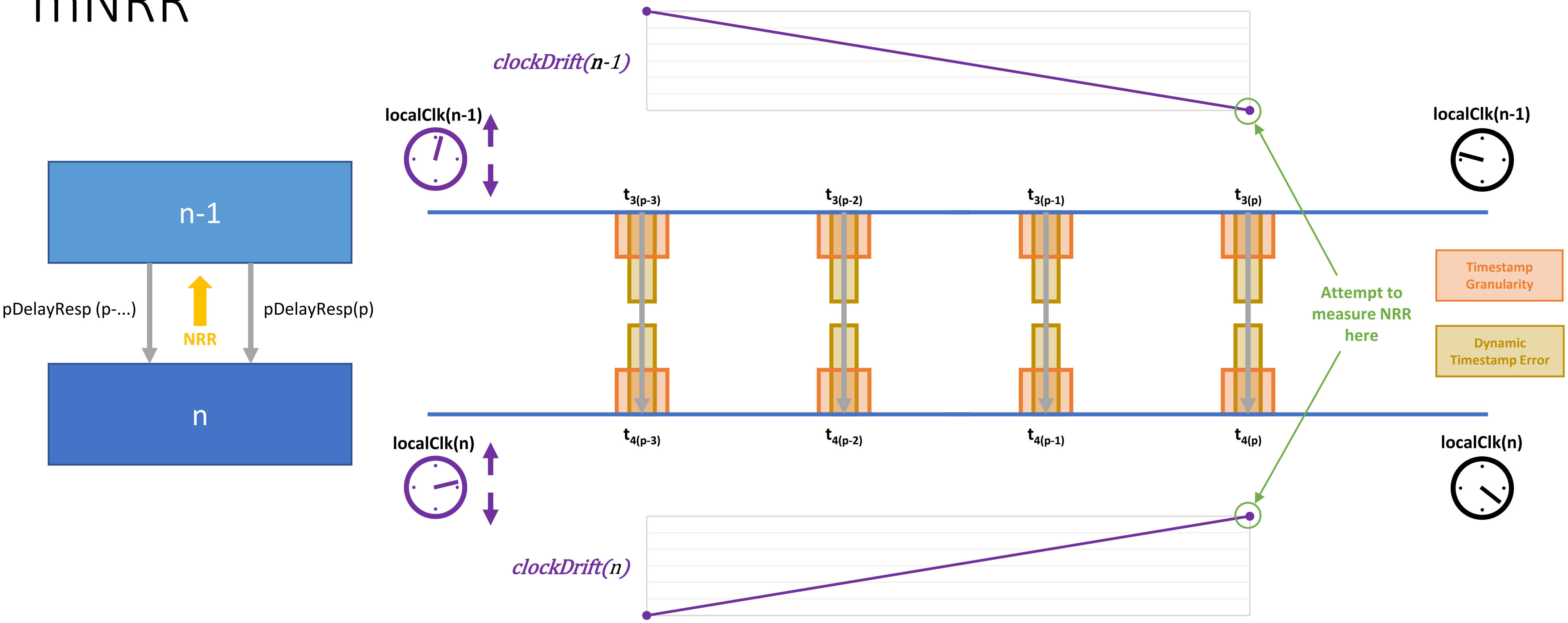
Techniques for minimising $mNRR_{error}$

# mNRR

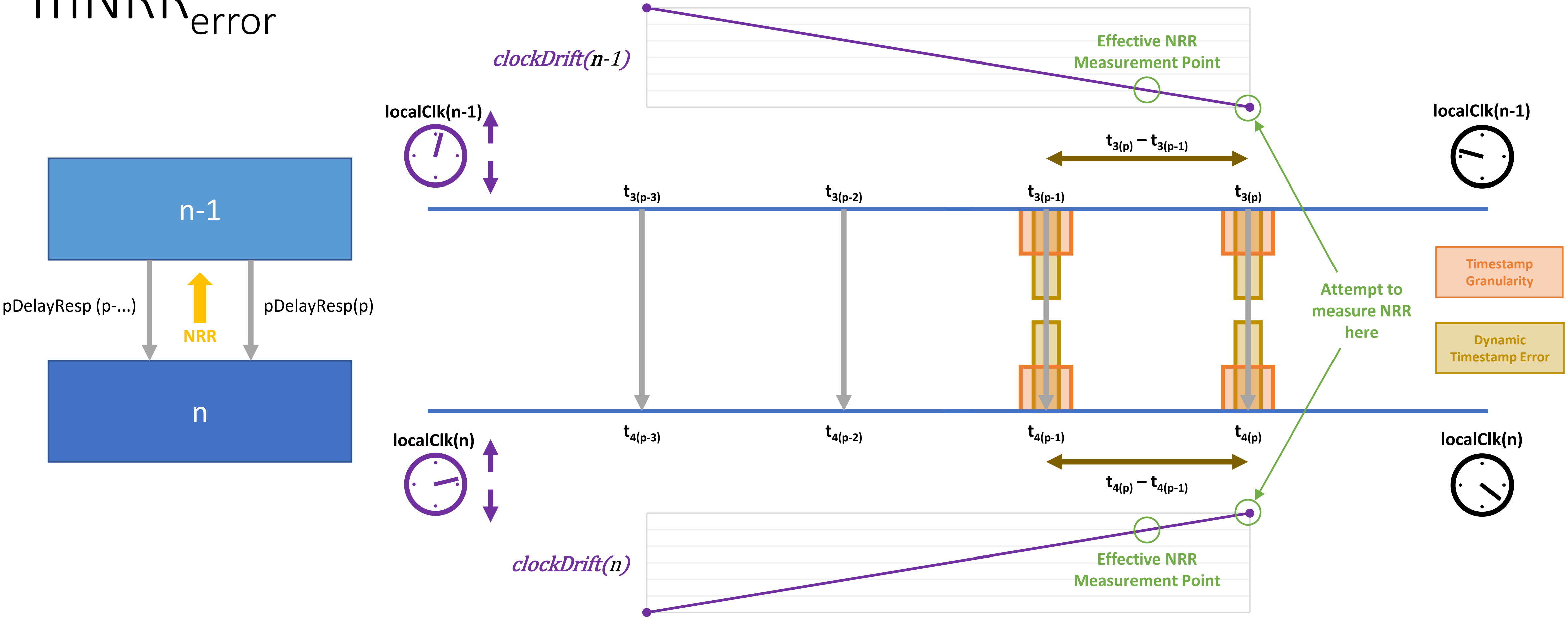


n-1
pDelayResp (p-...)
NRR
pDelayResp(p)
n
$t_{3(p-3)}$
$t_{3(p-2)}$
$t_{3(p-1)}$
$t_{3(p)}$
Attempt to measure NRR here
$t_{4(p-3)}$
$t_{4(p-2)}$
$t_{4(p-1)}$
$t_{4(p)}$

# mNRR


clockDrift(n-1)
localClk(n-1)
localClk(n-1)
n-1
pDelayResp (p-...)
NRR
pDelayResp(p)
n
t3(p-3)
t3(p-2)
t3(p-1)
t3(p)
t4(p-3)
t4(p-2)
t4(p-1)
t4(p)
Attempt to measure NRR here
Timestamp Granularity
Dynamic Timestamp Error
localClk(n)
localClk(n)
clockDrift(n)

# $mNRR_{error}$

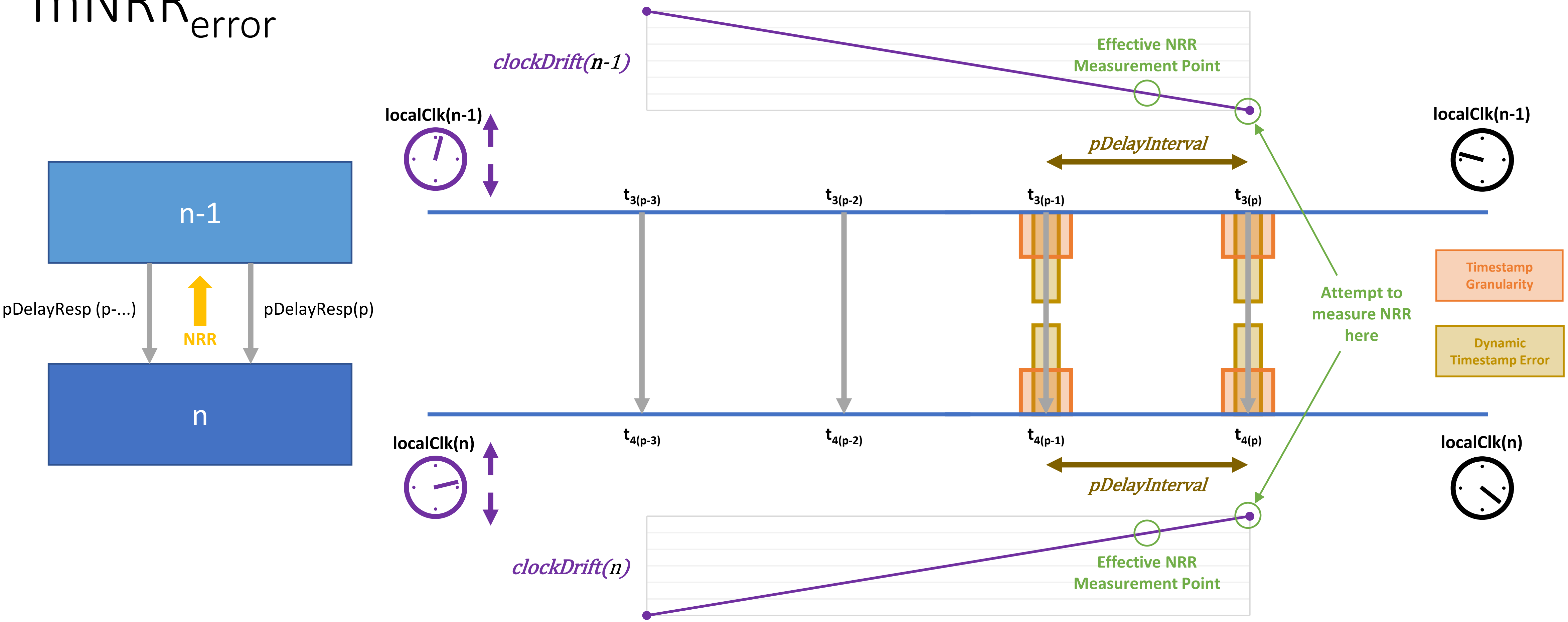



$$mNRR = \left(\frac{(t_{4(p)} - t_{4(p-1)})}{(t_{3(p)} - t_{3(p-1)})}\right)$$

$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{(t_{4(p)} - t_{4(p-1)}) \times \mathbf{10^6}}\right)$$ **ppm**

$$mNRR_{errorCD}(n) = \left(\frac{(t_{4(p)} - t_{4(p-1)})}{\mathbf{2 \times 10^9}}\right)(\boldsymbol{clockDrift}(n-1) - \boldsymbol{clockDrift}(n))$$ **ppm**

# $mNRR_{error}$

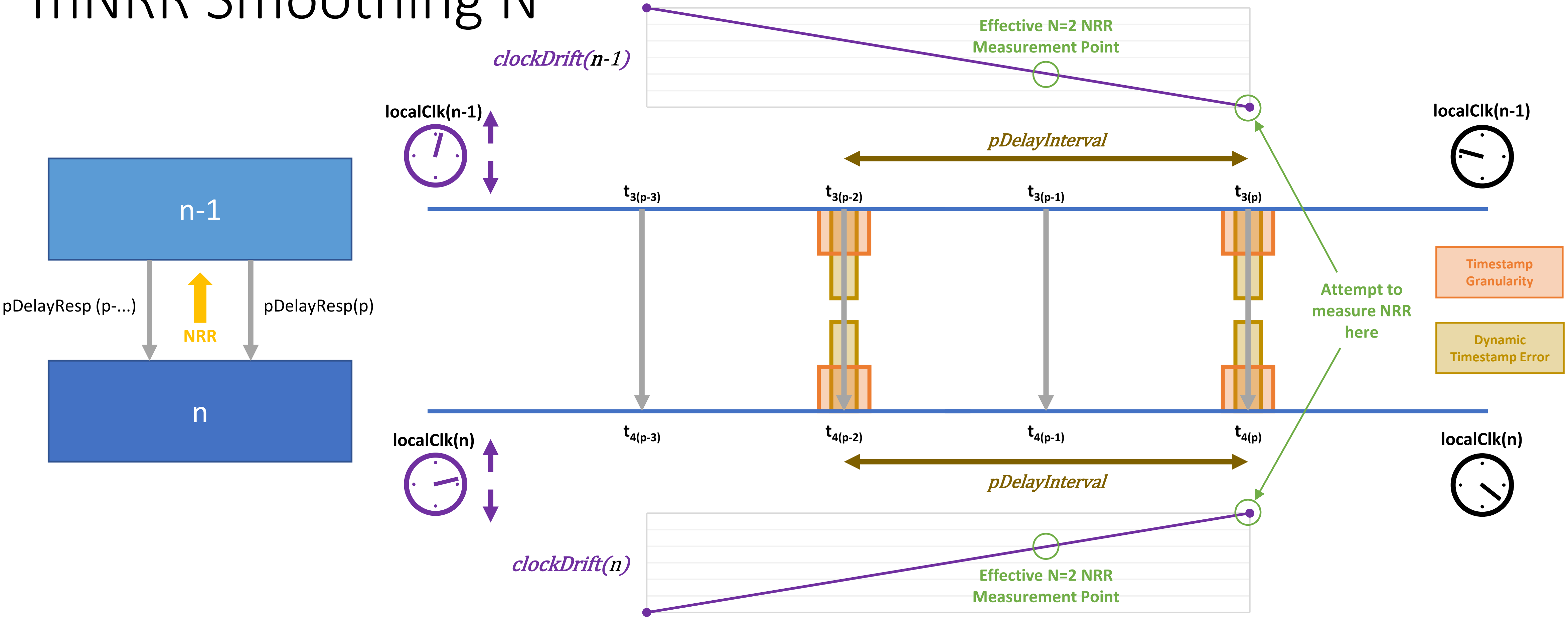



$$mNRR = \left(\frac{(t_{4(p)} - t_{4(p-1)})}{(t_{3(p)} - t_{3(p-1)})}\right)$$

$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval}\right)$$ **ppm**

$$mNRR_{errorCD}(n) = \left(\frac{pDelayInterval}{2 \times 10^3}\right)\left(clockDrift(n-1) - clockDrift(n)\right)$$ **ppm**

# mNRR Smoothing N

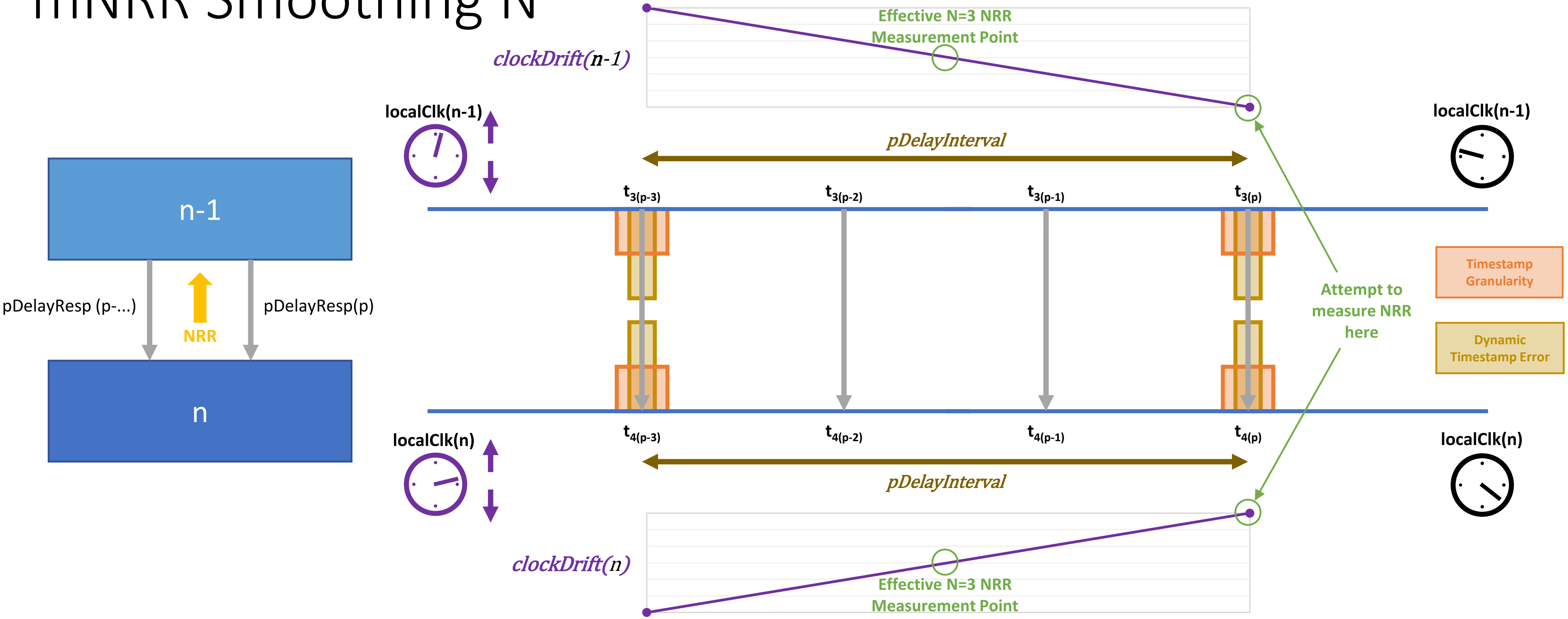



$$mNRR = \left(\frac{(t_{4(p)} - t_{4(p-1)})}{(t_{3(p)} - t_{3(p-1)})}\right)$$

$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval \times 2}\right)$$ **ppm**

$$mNRR_{errorCD}(n) = \left(\frac{pDelayInterval \times 2}{2 \times 10^3}\right)(clockDrift(n-1) - clockDrift(n))$$ **ppm**

# mNRR Smoothing N

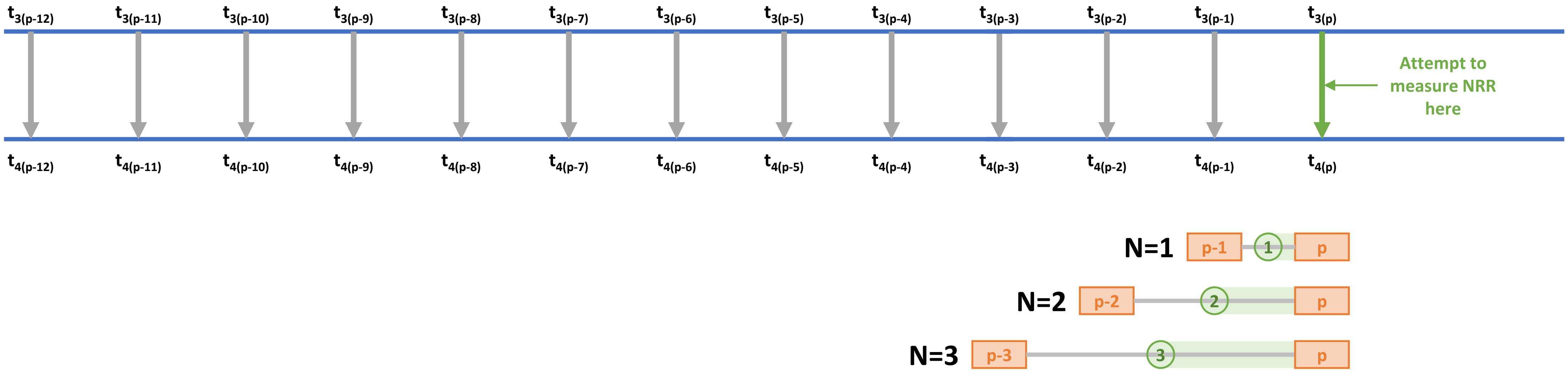



$$mNRR = \left(\frac{(t_{4(p)} - t_{4(p-1)})}{(t_{3(p)} - t_{3(p-1)})}\right)$$

$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval \times 3}\right) \quad \textbf{ppm}$$

$$mNRR_{errorCD}(n) = \left(\frac{pDelayInterval \times 3}{2 \times 10^3}\right)\left(clockDrift(n-1) - clockDrift(n)\right) \quad \textbf{ppm}$$

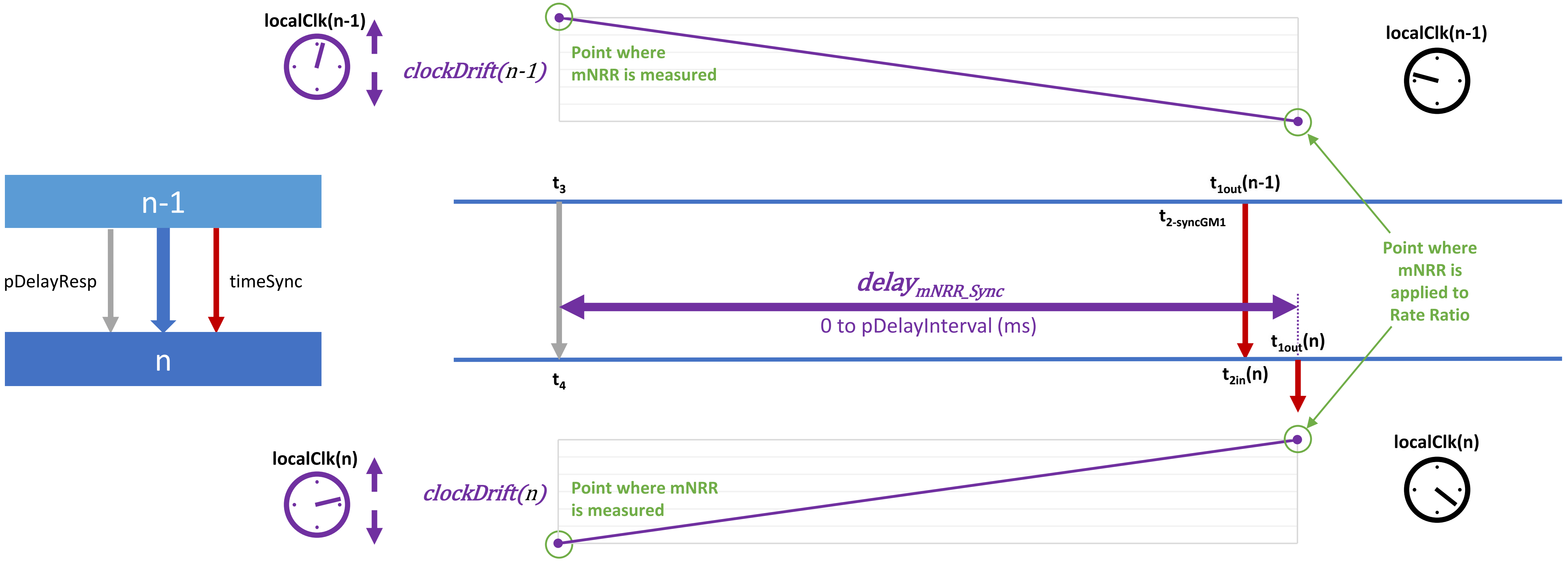
# mNRR Smoothing N



# mNRR Smoothing N

- Using the N^th previous pDelay Response timestamps...
  - Decreases the effect of Timestamp Error by a factor of N
  - Increases the effect of error due to Clock Drift by a factor of N
- Increasing N is similar to increasing pDelay Interval...
  - Decreases the effect of Timestamp Error
  - Increases the effect of error due to Clock Drift
- ...but different...
  - Much greater resolution (not limited to factors of 2)
  - Doesn't increase the direct effect of error due to Clock Drift on Rate Ratio calculation

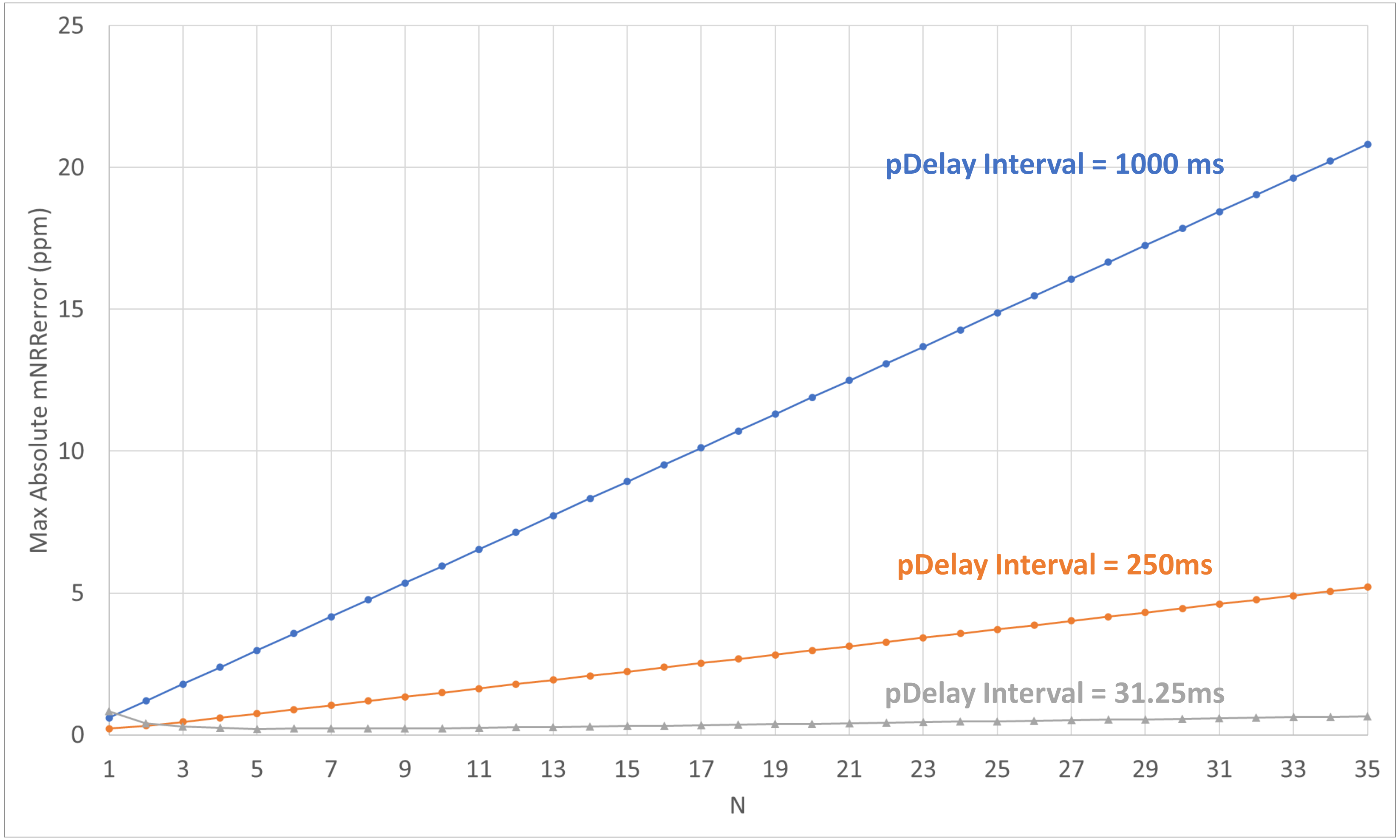
# $RR_{errorCD}$




$$RR_{errorCD}(n) = \frac{delay_{mNRR\_Sync}}{10^3}\left(clockDrift(n-1) - clockDrift(n)\right)$$

**ppm**

# Optimal Value of N

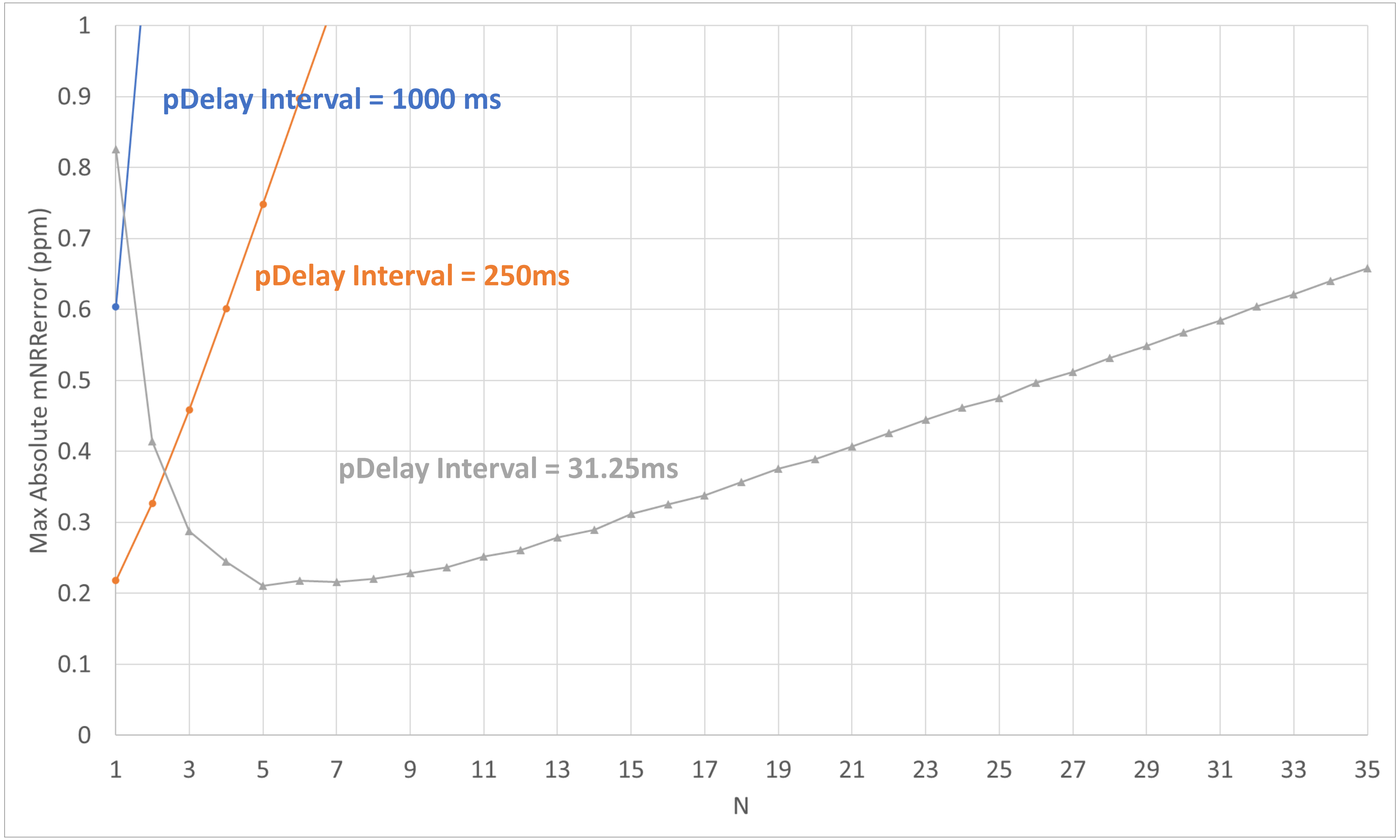



| Input Errors | | |
|---|---|---|
| Clock Drift | ±0.6 | ppm/s |
| Timestamp Granularity Error | ±4 | ns |
| Dynamic Time Stamp Error | ±4 | ns |
| **Input Parameters** | | |
| pDelay Interval | 31.25 | ms |
| **Input Correction Factors** | | |
| mNRR Smoothing M | 1 | |
| **Configuration** | | |
| Hops | 1 | |
| Runs (per value of M) | 100,000 x 10 | |

Optimal value of N is 1 unless pDelay Interval is short.

How short?...

# Optimal Value of N



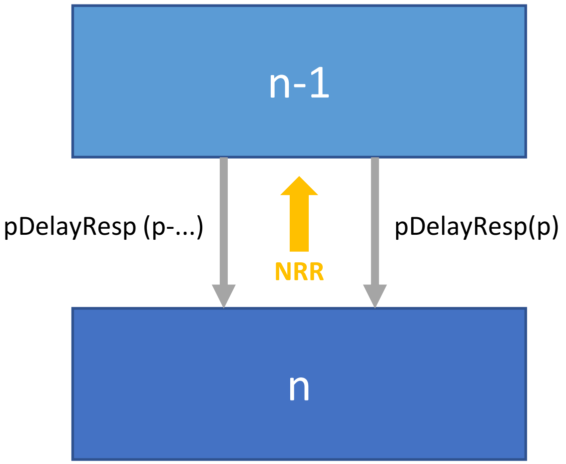

| **Input Errors** | | |
|---|---|---|
| Clock Drift | ±0.6 | ppm/s |
| Timestamp Granularity Error | ±4 | ns |
| Dynamic Time Stamp Error | ±4 | ns |
| **Input Parameters** | | |
| pDelay Interval | 31.25 | ms |
| **Input Correction Factors** | | |
| mNRR Smoothing M | 1 | |
| **Configuration** | | |
| Hops | 1 | |
| Runs (per value of M) | 100,000 x 10 | |

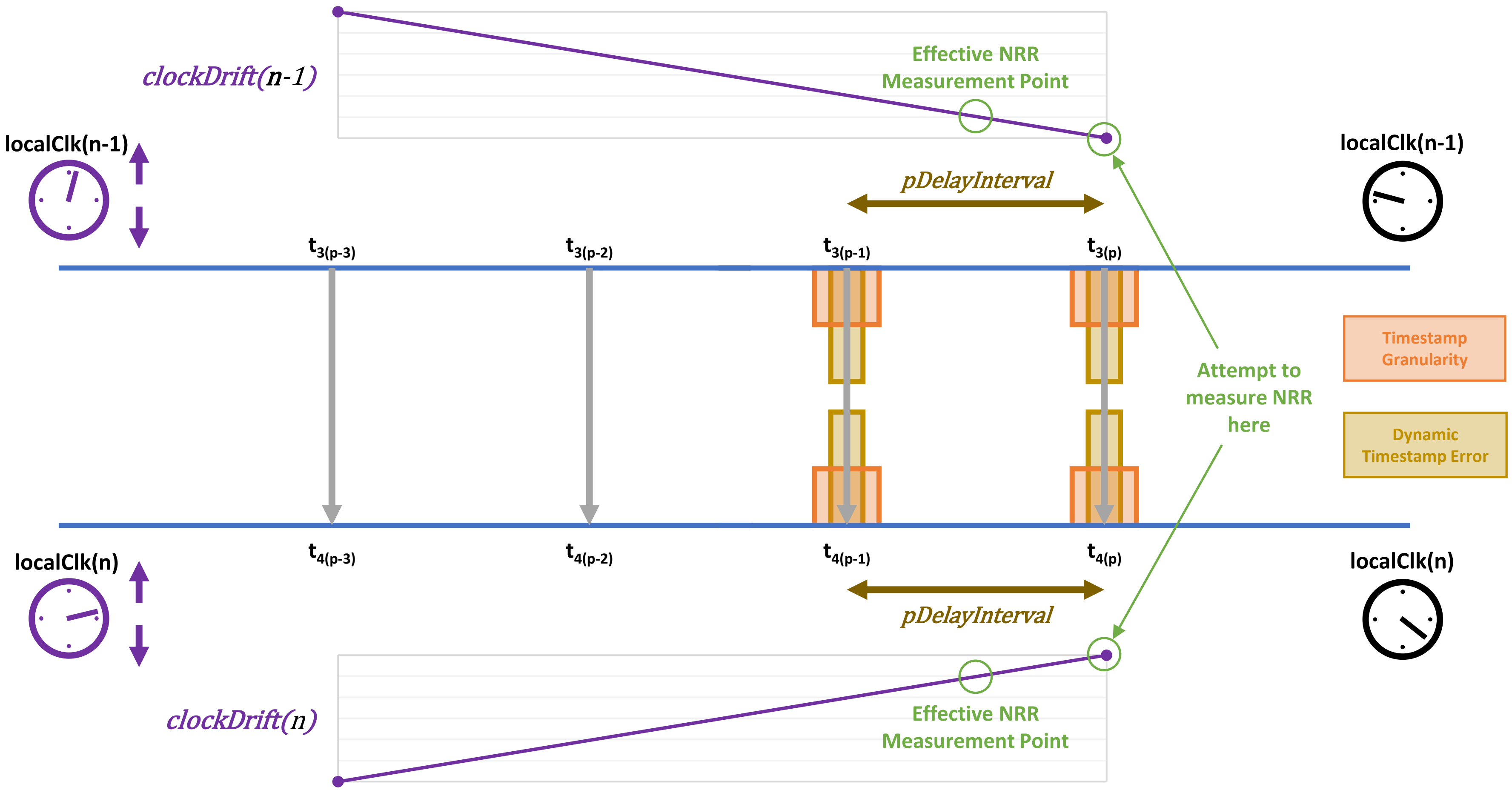
Optimal value of N for pDelay Interval **31.25ms** is **5**. (For these input errors, with no clock drift compensation.)

This is the similar to a pDelay Interval of **156.25ms**...which is impossible due to the way pDelay Interval is configured.

Best value of N for a feasible pDelay Interval is **8**...which is similar to a pDelay Interval of **250ms**.

# $mNRR_{error}$



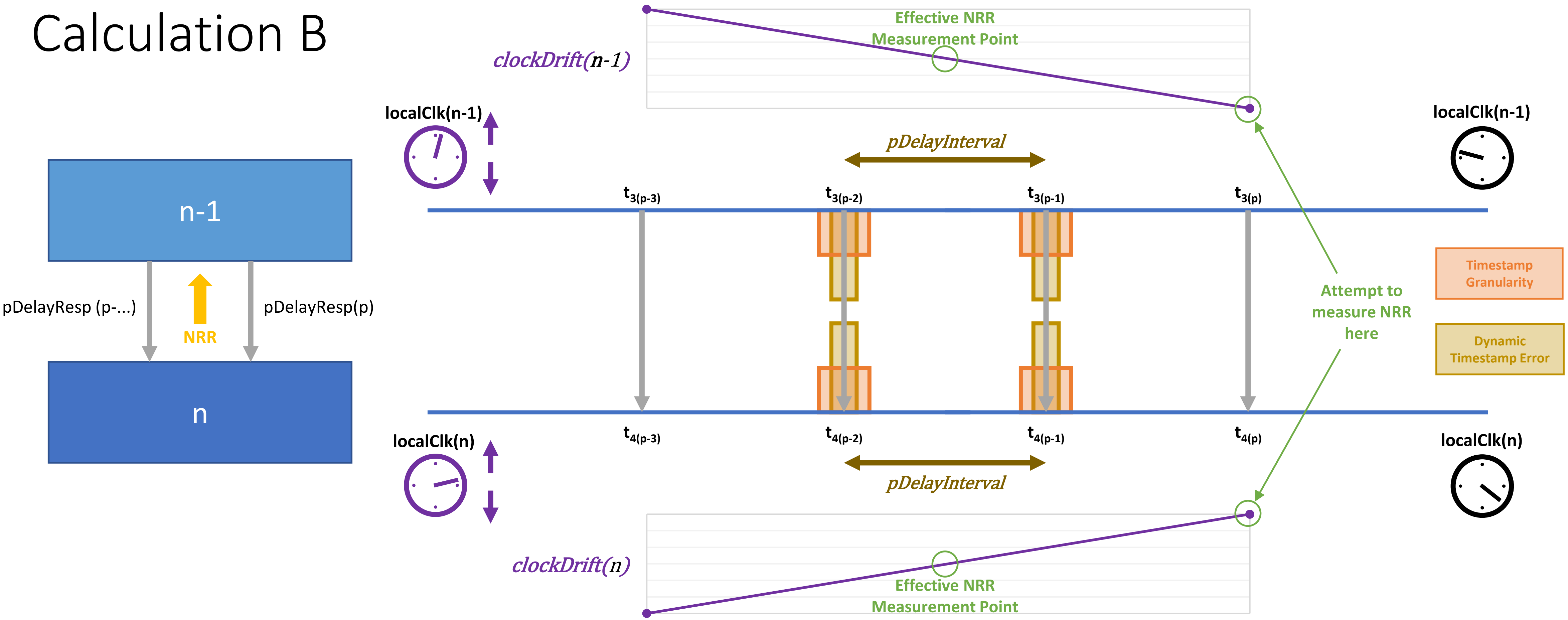




$$mNRR = \left(\frac{(t_{4(p)} - t_{4(p-1)})}{(t_{3(p)} - t_{3(p-1)})}\right)$$

$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval}\right)$$ **ppm**

$$mNRR_{errorCD}(n) = \left(\frac{pDelayInterval}{2 \times 10^3}\right)\left(clockDrift(n-1) - clockDrift(n)\right)$$ **ppm**

# mNRR Smoothing M Calculation B

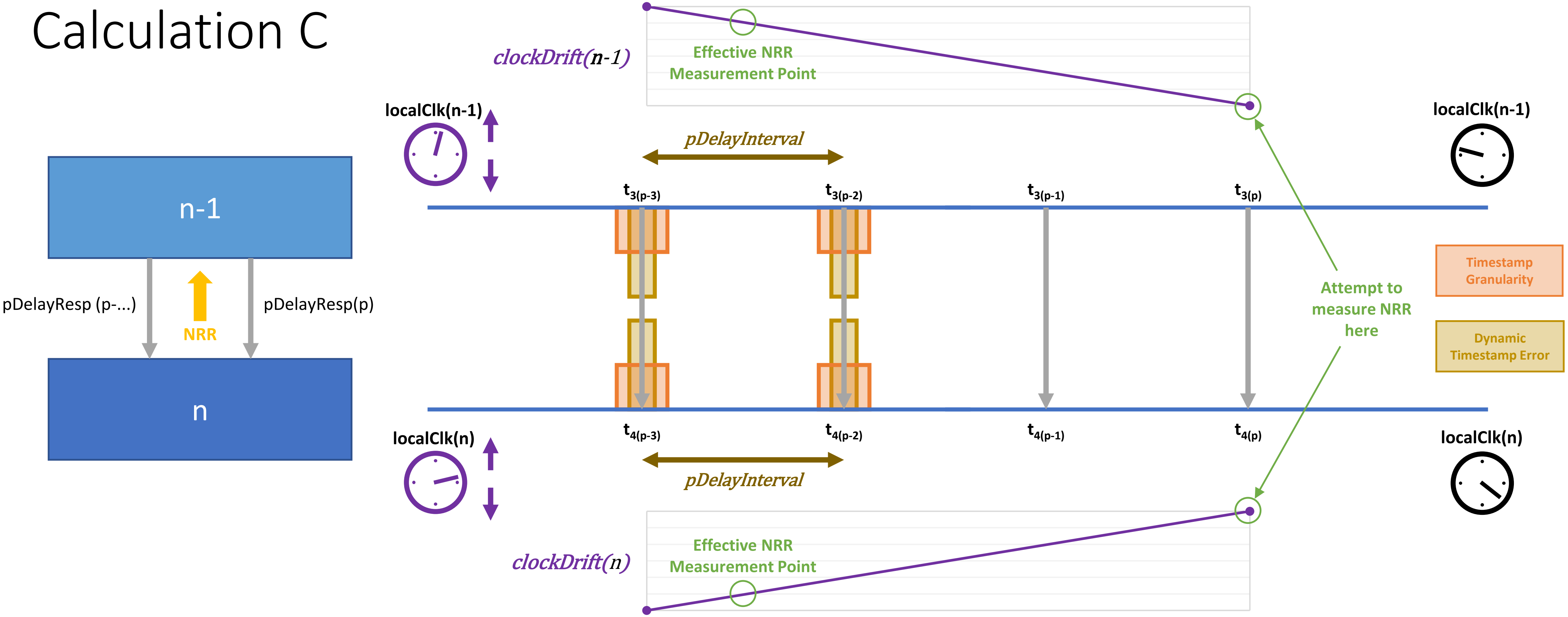



$$mNRR = \left(\frac{(t_{4_{(p)}} - t_{4_{(p-1)}})}{(t_{3_{(p)}} - t_{3_{(p-1)}})}\right)$$

$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror_{(p-1)}} - t_{4PDerror_{(p-2)}}) - (t_{3PDerror_{(p-1)}} - t_{3PDerror_{(p-2)}})}{pDelayInterval}\right)$$ **ppm**

$$mNRR_{errorCD}(n) = \left(\frac{pDelayInterval \times 3}{2 \times 10^3}\right)(clockDrift(n-1) - clockDrift(n))$$ **ppm**

# mNRR Smoothing M Calculation C




$$mNRR = \left(\frac{(t_{4(p)} - t_{4(p-1)})}{(t_{3(p)} - t_{3(p-1)})}\right)$$

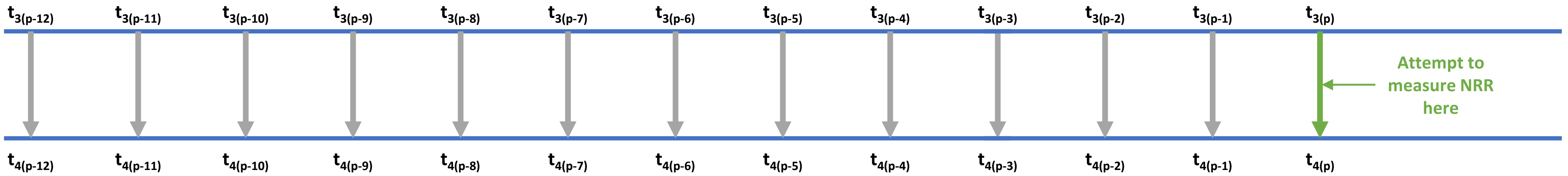
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p-2)} - t_{4PDerror(p-3)}) - (t_{3PDerror(p-2)} - t_{3PDerror(p-3)})}{pDelayInterval}\right) \quad \textbf{ppm}$$

$$mNRR_{errorCD}(n) = \left(\frac{pDelayInterval \times 5}{2 \times 10^3}\right)\left(clockDrift(n-1) - clockDrift(n)\right) \quad \textbf{ppm}$$

# mNRR Smoothing M

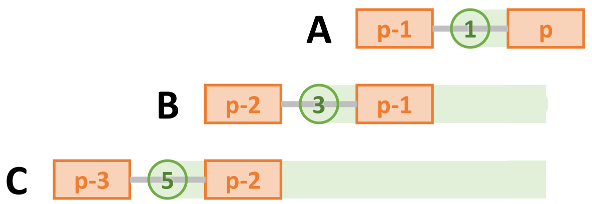
- Using the median of M previous NRR calculations...
  - Selects the NRR calucation with the median total error ($mNRR_{error}$)
    - Addition of $mNRR_{errorTS}$ from Timestamp Errors & $mNRR_{errorCD}$ from Clock Drift
- Implementing this in the Monte Carlo Analysis while keeping track of the contribution of different sources of error adds a lot of complexity
- Results are also...not intuitive
  - Vary a lot depending on selection of pDelay Interval, M & N
  - Comparison with M=1 results (i.e. no median) is...more complex
- Details on following slides...

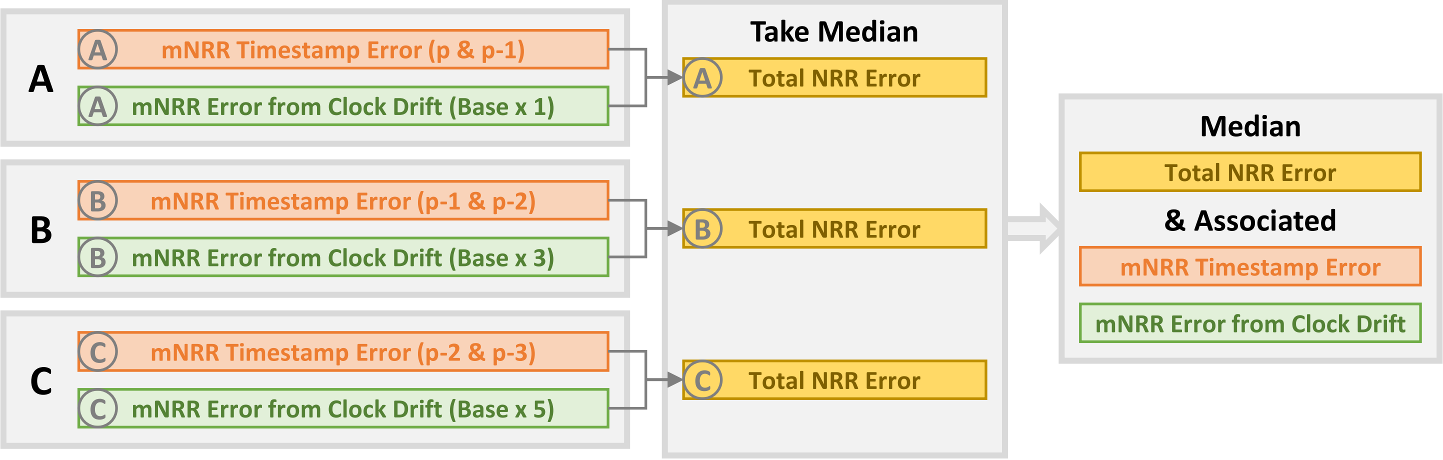
# mNRR Smoothing M – Calculations

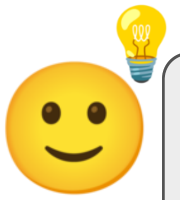



**N = 1**

**M = 3**



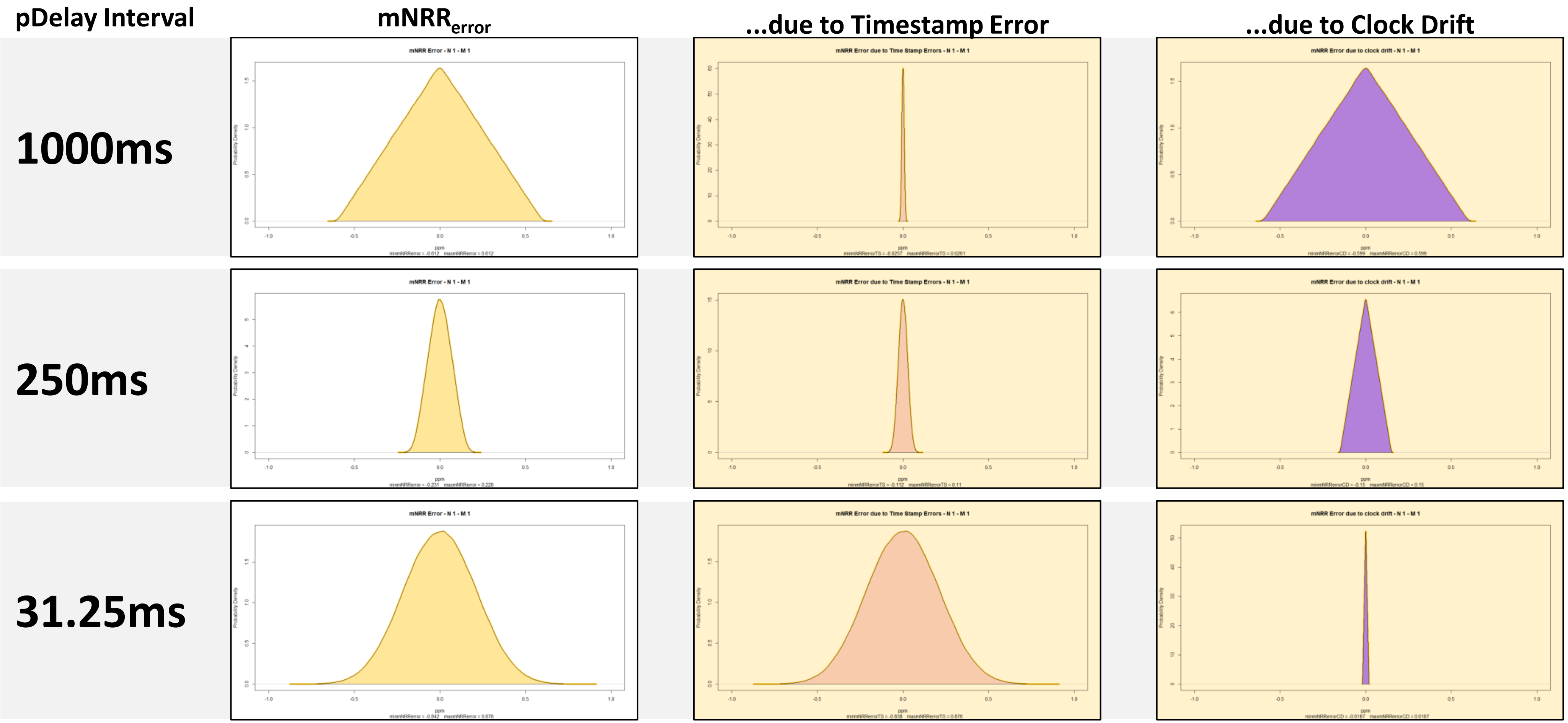



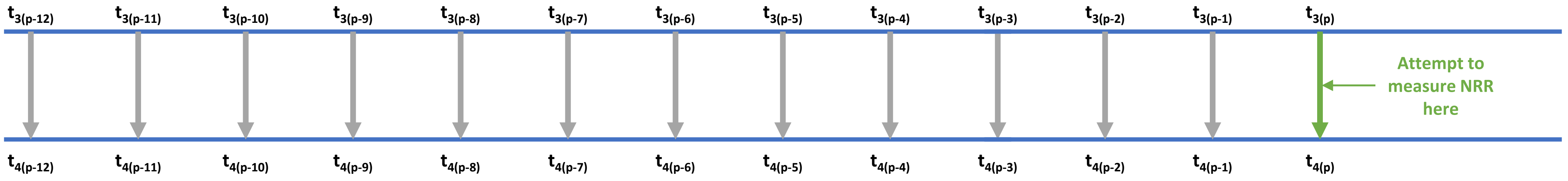
It depends on pDelay Interval! (And the values of M & N)

# Timestamp Error vs Error due to Clock Drift

(mNRR – Single Hop – No Smoothing, i.e. N=1, M=1)

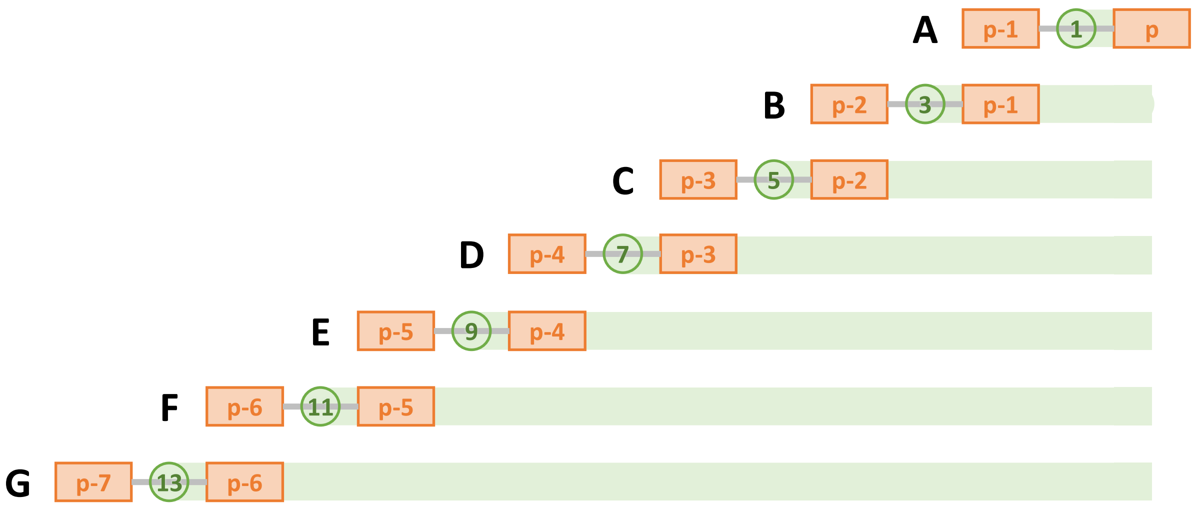


# mNRR Smoothing M & pDelay Interval



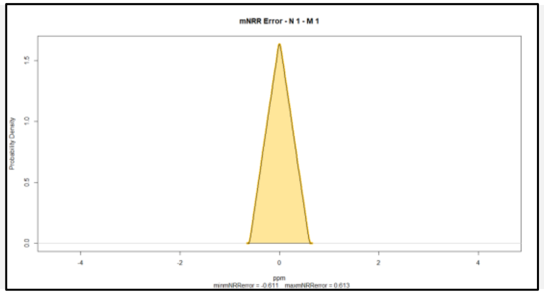
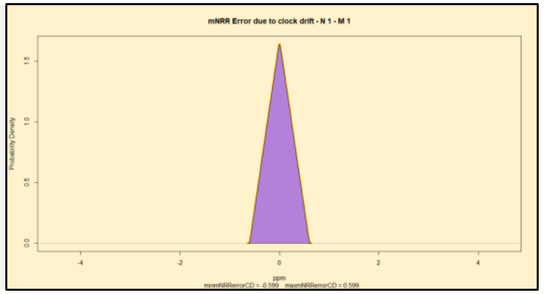
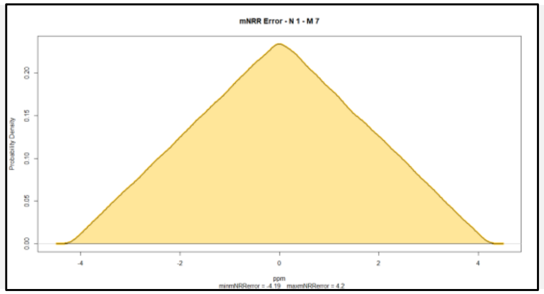
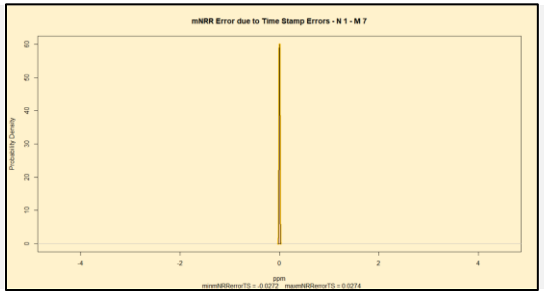
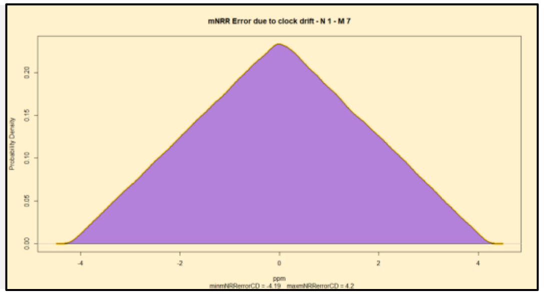

**N = 1**

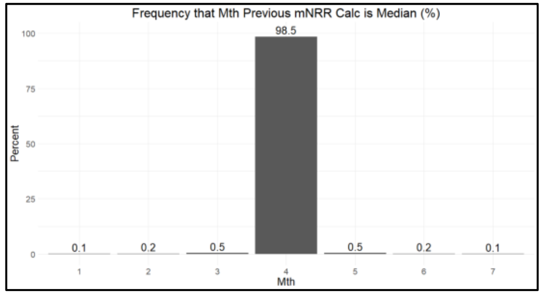
**M = 7**



# Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

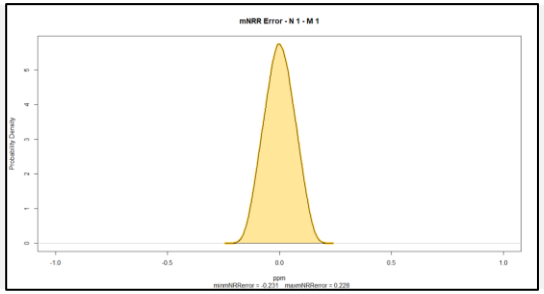
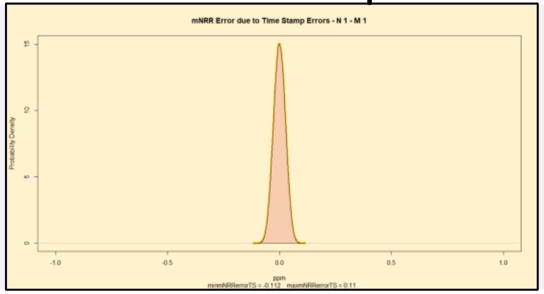
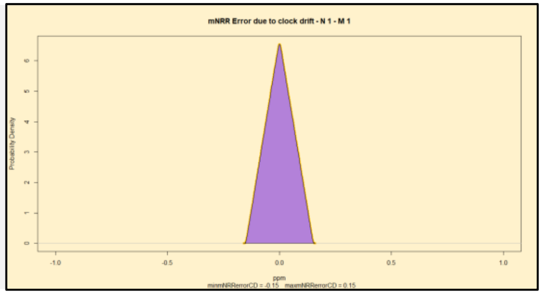
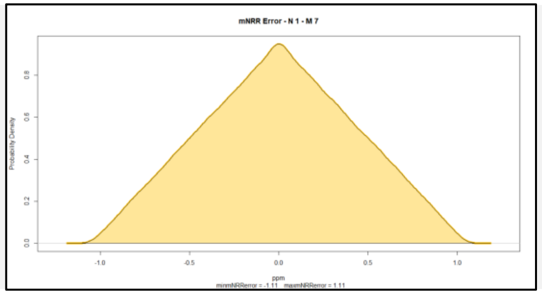
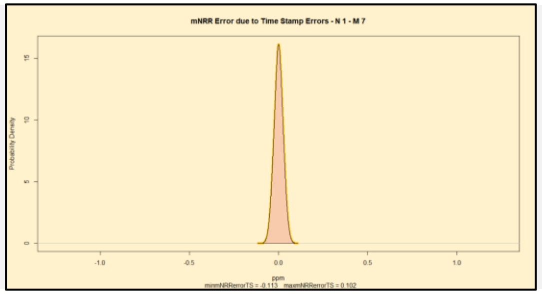
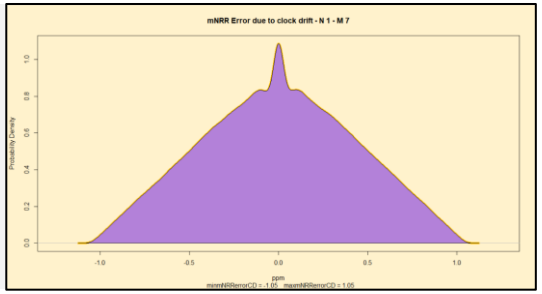
| | pDelay Interval | $mNRR_{error}$ | ...due to Timestamp Error | ...due to Clock Drift |
|---|---|---|---|---|
| **M=1** | **1000ms** |  |  |  |
| **M=7** | **1000ms** |  |  |  |

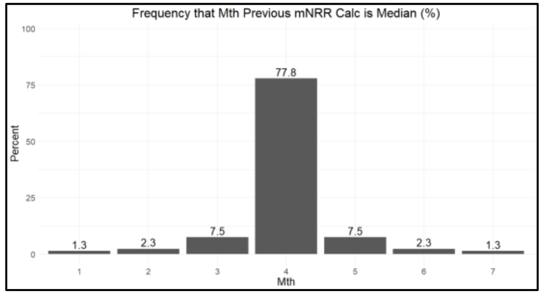



- For values of pDelay Interval where $mNRR_{error}$ due to Clock Drift dominates Timestamp error, taking the median of previous NRR calculations only increases $mNRR_{error}$.

# Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

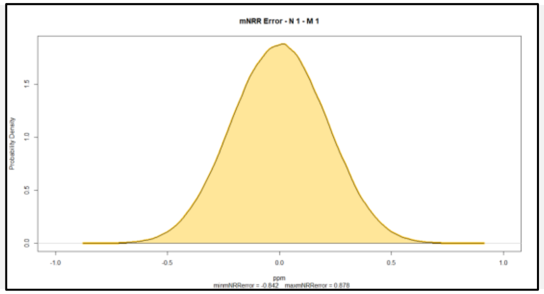
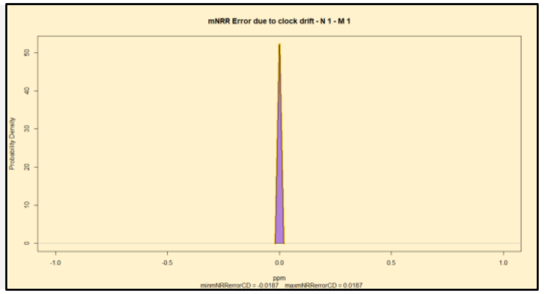
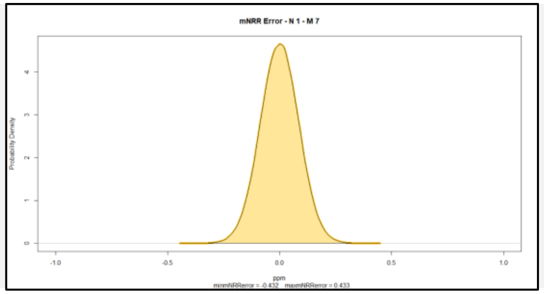
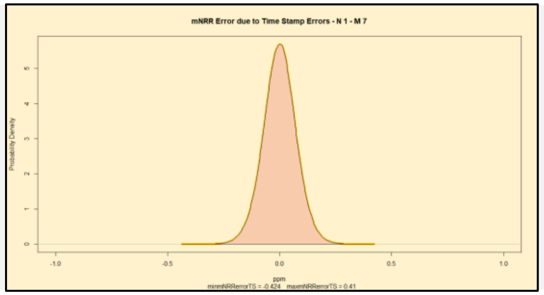
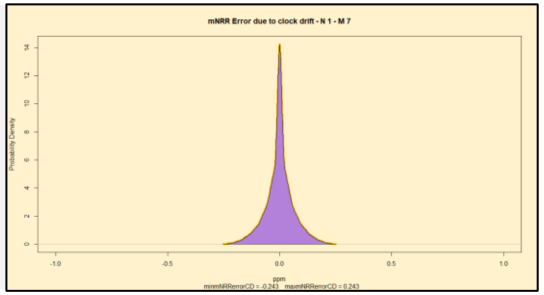
| | pDelay Interval | $mNRR_{error}$ | ...due to Timestamp Error | ...due to Clock Drift |
|---|---|---|---|---|
| **M=1** | **250ms** |  |  |  |
| **M=7** | **250ms** |  |  |  |



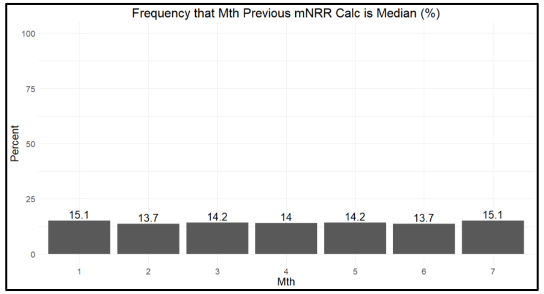

- For values of pDelay Interval where $mNRR_{error}$ due to Clock Drift and Timestamp error are of comparable magnitude, taking the median of previous NRR calculations only increases $mNRR_{error}$.

# Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

| | pDelay Interval | $mNRR_{error}$ | ...due to Timestamp Error | ...due to Clock Drift |
|---|---|---|---|---|
| **M=1** | **31.25ms** |  |  |  |
| **M=7** | **31.25ms** |  |  |  |



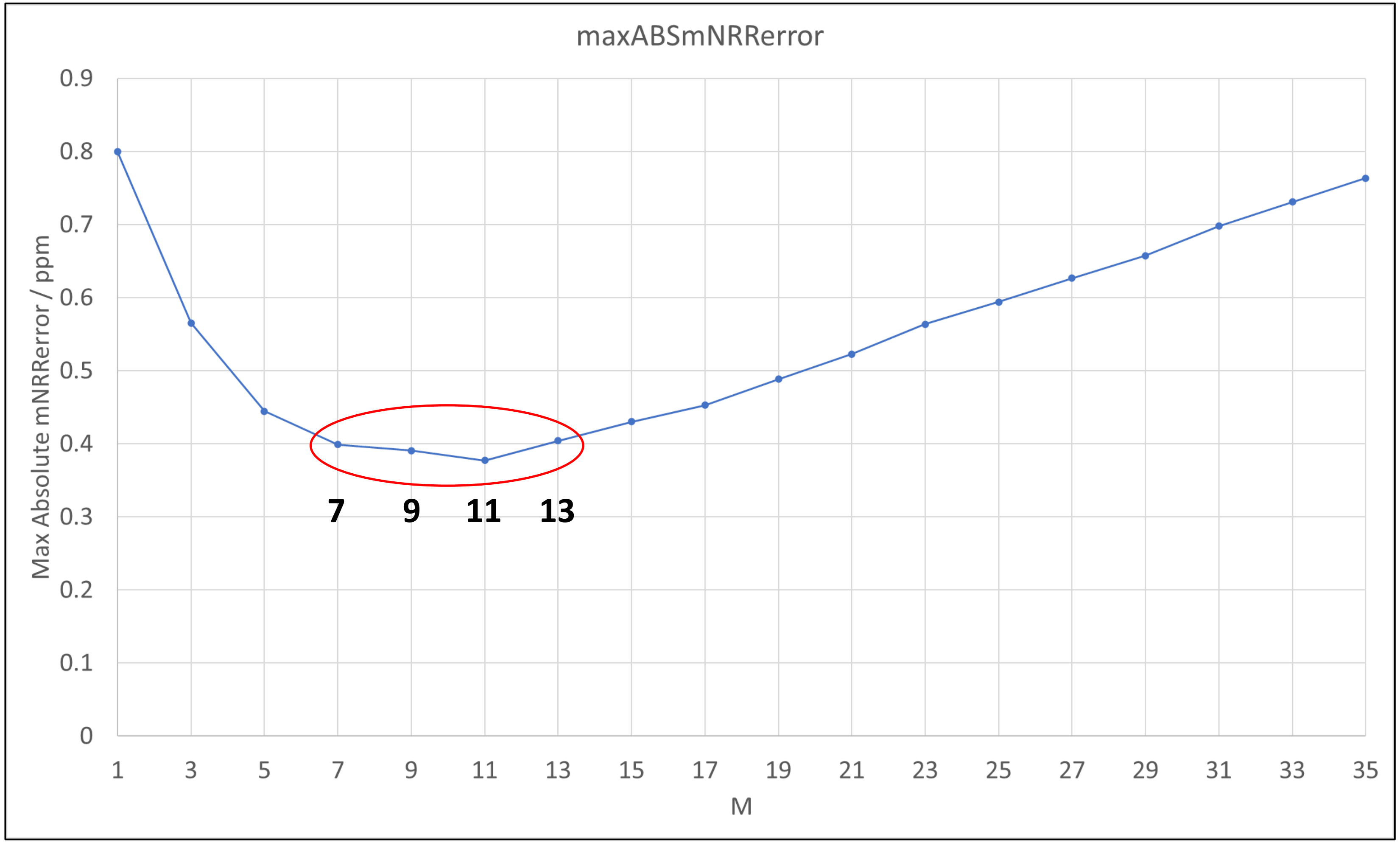



- For values of pDelay Interval where $mNRR_{error}$ due to Timestamp error dominates error due to Clock Drift, taking the median of previous NRR calculations can reduce $mNRR_{error}$.
- This raises the question: what is the optimal value of M?

# Challenges of Comparing Results

- Previously used 7σ value...but that only works for gaussian distributions...and these distributions are not gaussian.
- Could use maximum absolute value...but that turns out to be noisey
  - More runs doesn't always help; can increase the chance of finding an outlier
- Ended up using average of result of 10 simulations, each of 100,000 runs.
  - One hop only.
  - OK for comparison against each other. Not a good guide for actual maximum error in the field.
  - May not matter for analysis of DTE if that retains gaussian distribution...but that is something to keep an eye on.

# Optimal Value of M

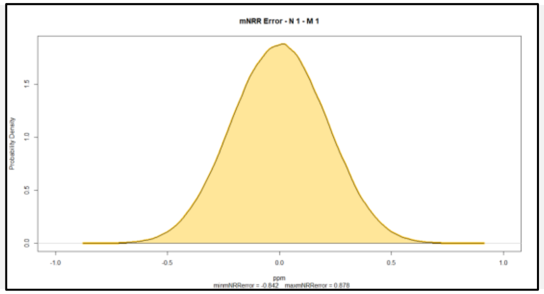
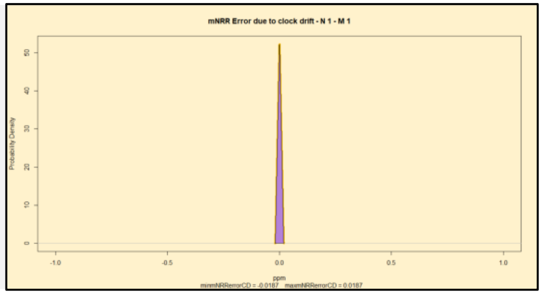
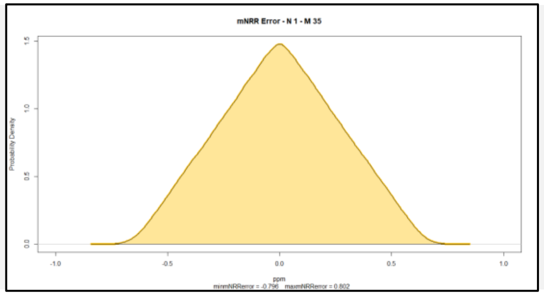
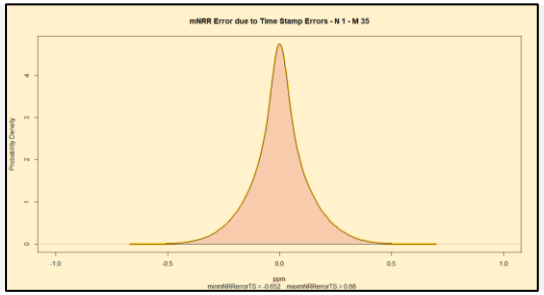
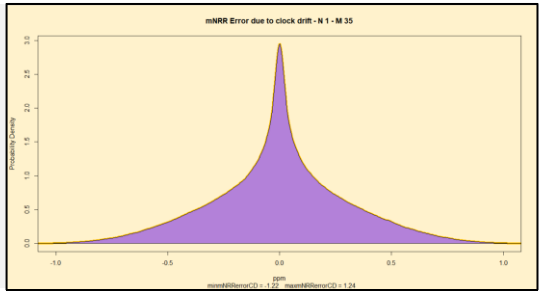


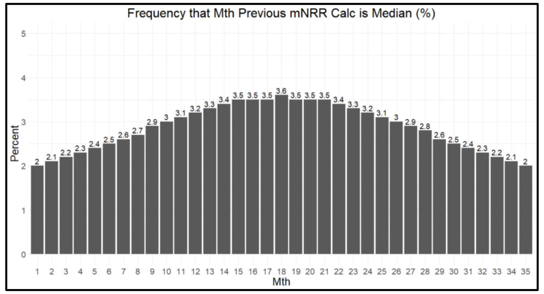

| Input Errors | | |
|---|---|---|
| Clock Drift | ±0.6 | ppm/s |
| Timestamp Granularity Error | ±4 | ns |
| Dynamic Time Stamp Error | ±4 | ns |
| **Input Parameters** | | |
| pDelay Interval | 31.25 | ms |
| **Input Correction Factors** | | |
| mNRR Smoothing N | 1 | |
| **Configuration** | | |
| Hops | 1 | |
| Runs (per value of M) | 100,000 x 10 | |

Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $mNRR_{error}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of 250ms.

# Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

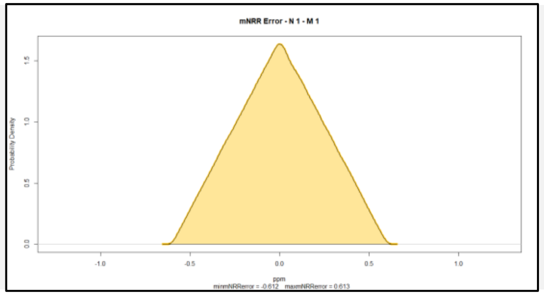
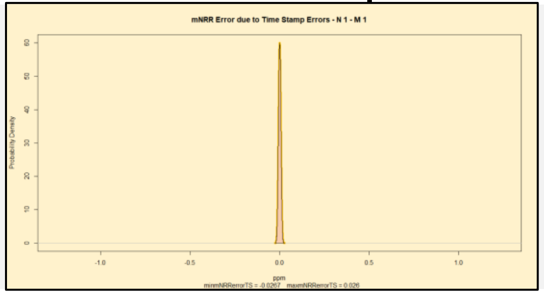
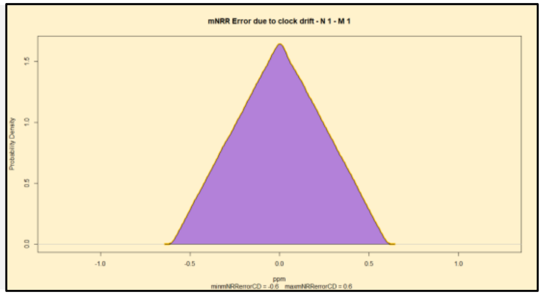
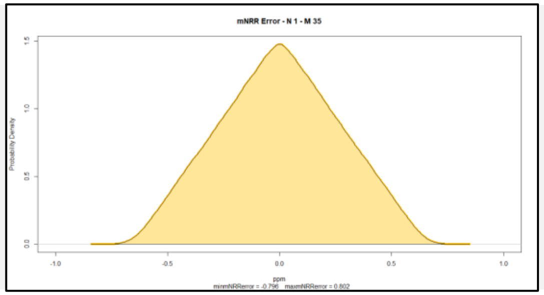
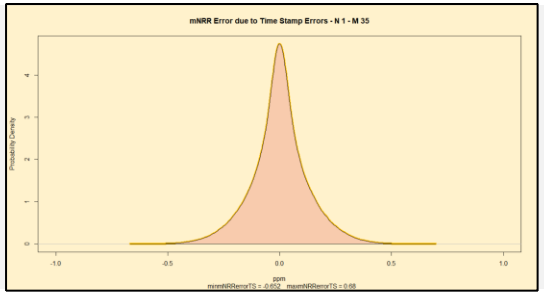
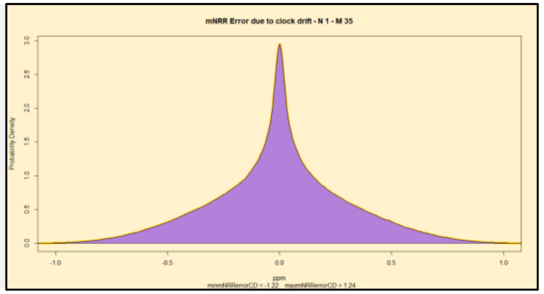
| | pDelay Interval | $mNRR_{error}$ | ...due to Timestamp Error | ...due to Clock Drift |
|---|---|---|---|---|
| **M=1** | **31.25ms** |  |  |  |
| **M=35** | **31.25ms** |  |  |  |

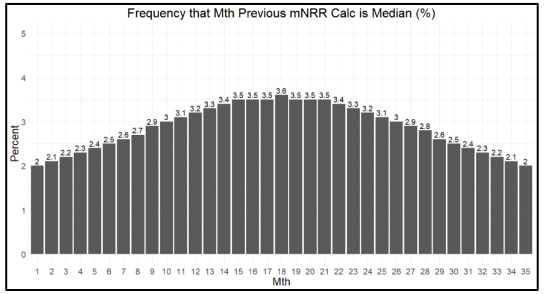



- Increasing M to 35 results in using NRR calculations from up to 1093.75ms in the past...and an $mNRR_{errror}$ distribution that is similar magnitude to pDelayInterval of 1000ms (and M=1).
- However...

# Effect of Taking Median of mNRR Calculations

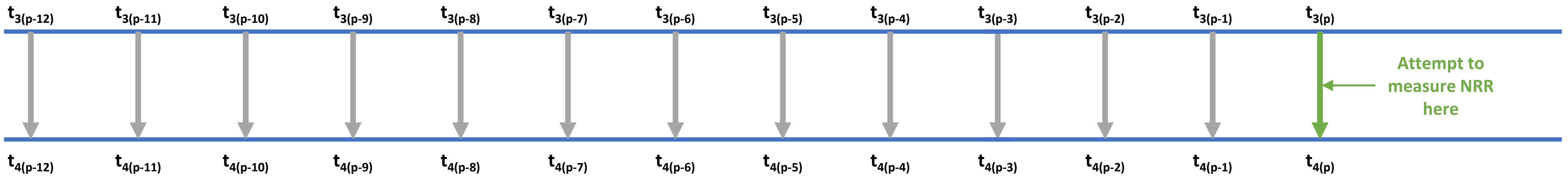
(mNRR – Single Hop – N=1)

| | pDelay Interval | $mNRR_{error}$ | ...due to Timestamp Error | ...due to Clock Drift |
|---|---|---|---|---|
| **M=1** | **1000ms** |  |  |  |
| **M=35** | **31.25ms** |  |  |  |



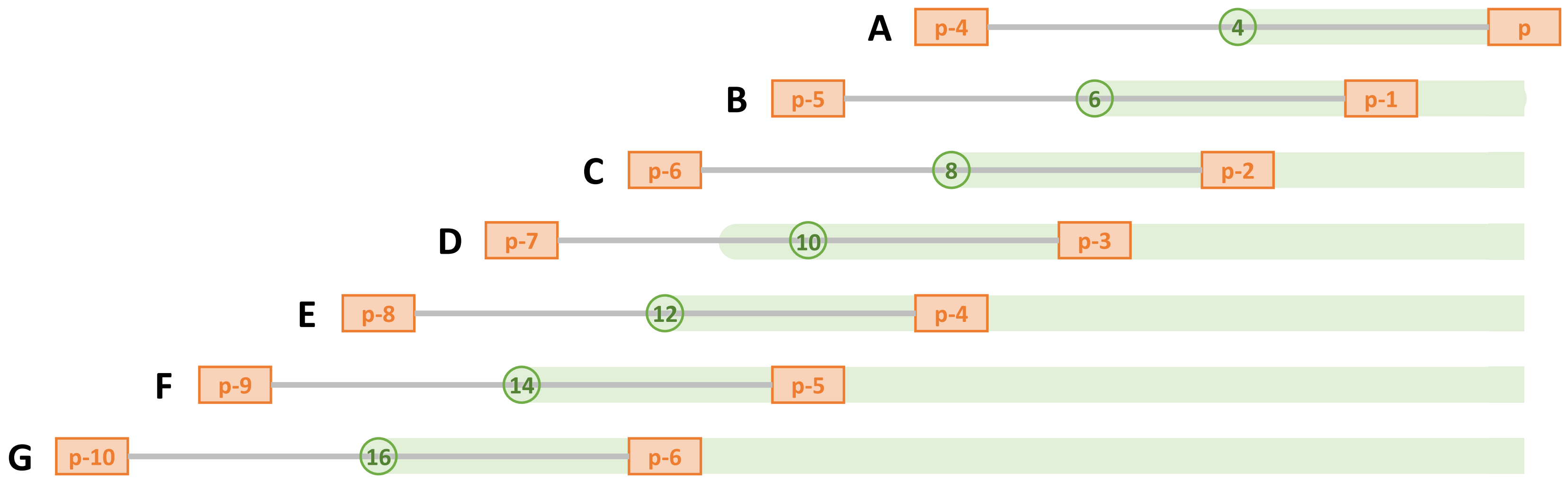

- The distribution of the underlying Timestamp errors and Clock Drift errors are very different.

# mNRR Smoothing – Combining N & M



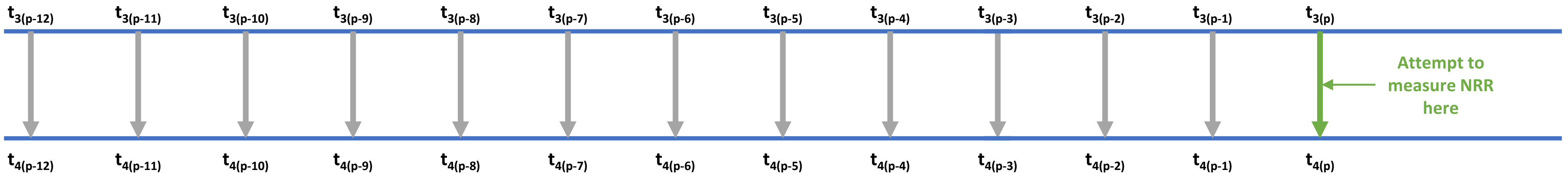

**N = 4**

**M = 7**



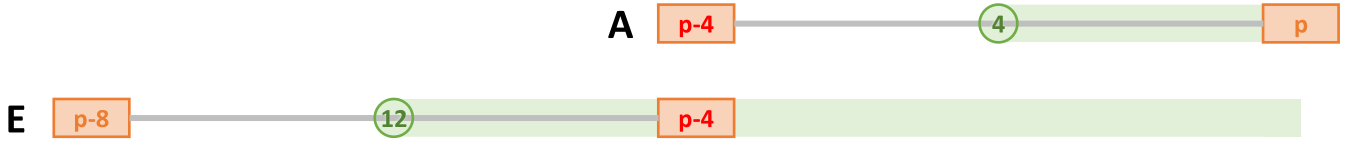

Note that the timestamps from p-4, p-5 & p-6 are used twice...which has an interesting effect...

# mNRR Smoothing – Combining N & M




**N = 4**

**M = 7**




$$mNRR_{errorTS}(A) = \left(\frac{(t_{4PDerror(p)} - \boldsymbol{t_{4PDerror(p-4)}}) - (t_{3PDerror(p)} - \boldsymbol{t_{3PDerror(p-4)}})}{pDelayInterval}\right)$$

$$mNRR_{errorTS}(E) = \left(\frac{(\boldsymbol{t_{4PDerror(p-4)}} - t_{4PDerror(p-8)}) - (\boldsymbol{t_{3PDerror(p-4)}} - t_{3PDerror(p-8)})}{pDelayInterval}\right)$$

When M>N, Timestamp some errors apply in two calculations, but with opposite signs. They don’t cancel out. The question is...does this help (because it “pushes” the two calculations to opposite sides of the “correct” value, which is more likely to be chosen at the median)...or hurt (because fewer unique timestamps are used when calculating the median)?

# M>N?

- Set Clock Drift to zero to remove its effect.
  - It will be part of any final optimisation, but the goal right now is to examine the effect (or not) of overlapping vs. non-overlapping mNRR calculation periods
- Keep M=7
- Use three combinations of pDelay Interval and N to isolate overlap vs. non-overlap, independent of increases in N acting similar to increasing pDelay Interval...and then another three without altering pDelay Interval

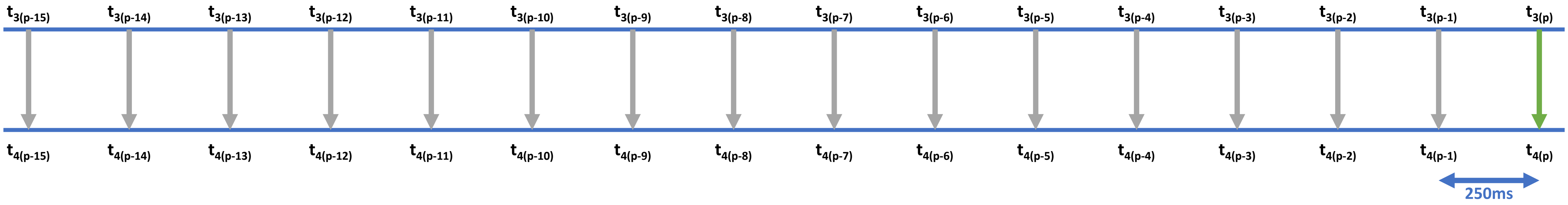
**Series 1**

| N | pDelay Interval | pDelay N x Interval |
|---|---|---|
| 1 | 250 ms | 250 ms |
| 4 | 62.5 ms | 250 ms |
| 8 | 31.25 ms | 250ms |

**Series 2**

| N | pDelay Interval | pDelay N x Interval |
|---|---|---|
| 1 | 31.25 ms | 31.25 ms |
| 4 | 31.25 ms | 62.5 ms |
| 7 | 31.25 ms | 218.75ms |

# mNRR Smoothing – Combining N & M

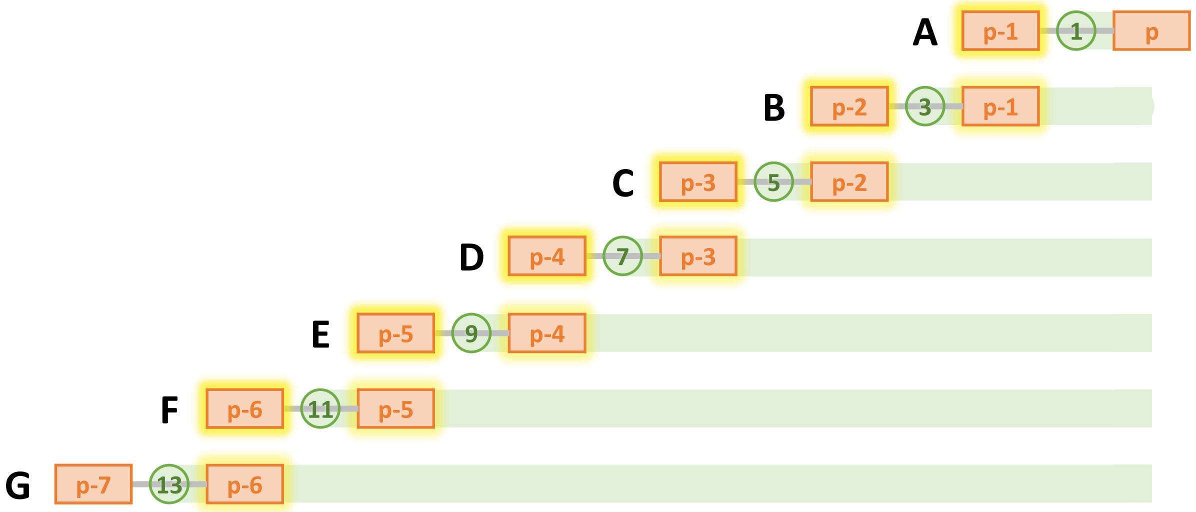



## Series 1

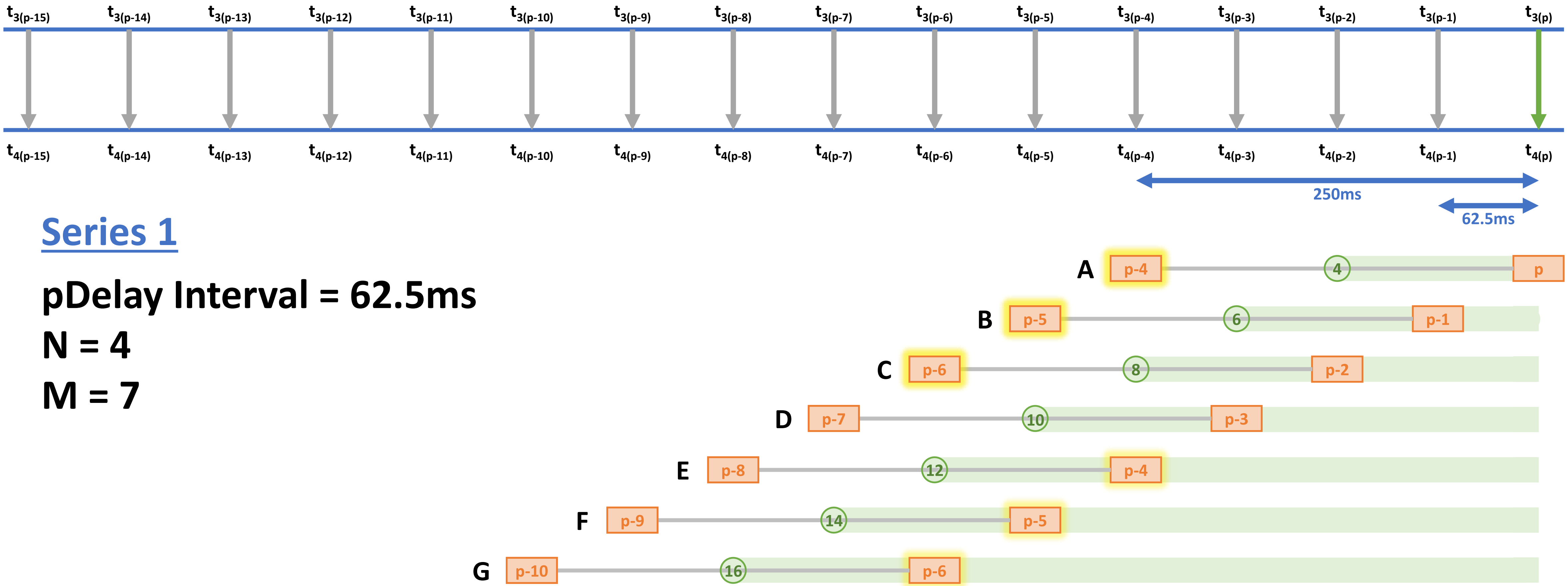
**pDelay Interval = 250ms**

**N = 1**

**M = 7**



# mNRR Smoothing – Combining N & M



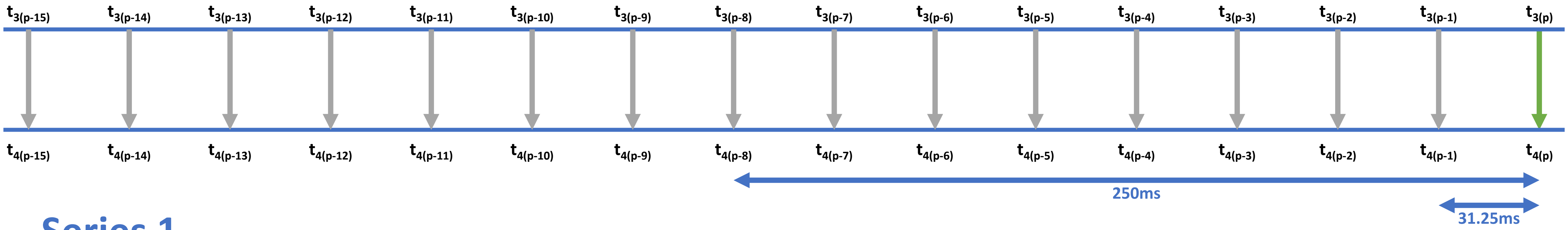

## Series 1

**pDelay Interval = 62.5ms**

**N = 4**

**M = 7**

# mNRR Smoothing – Combining N & M

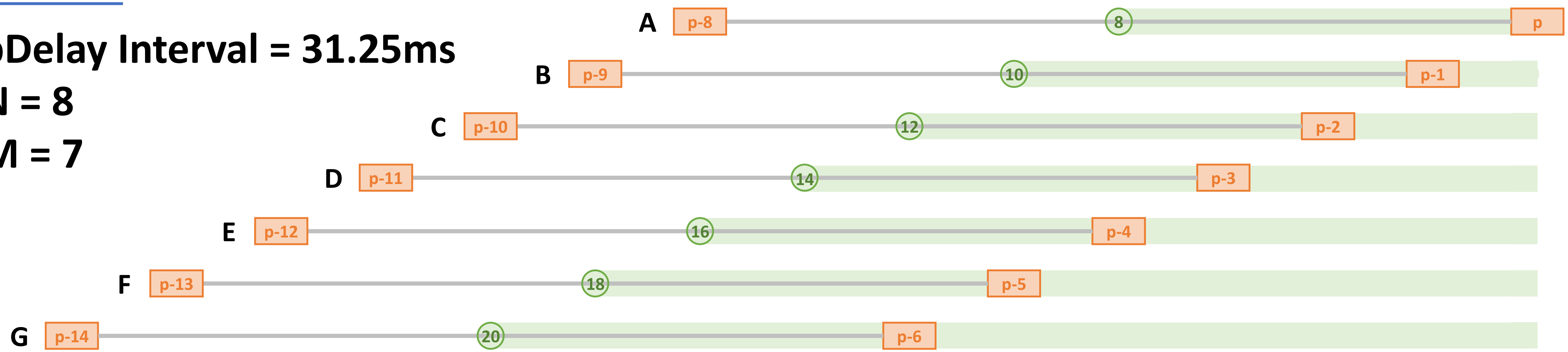



## Series 1

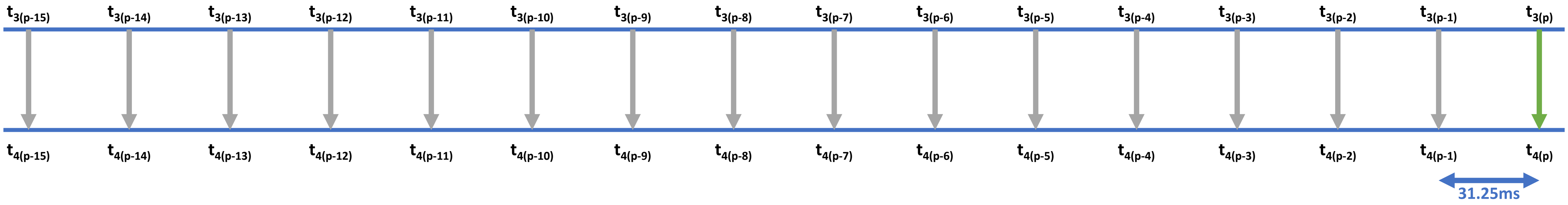
**pDelay Interval = 31.25ms**

**N = 8**

**M = 7**



# mNRR Smoothing – Combining N & M

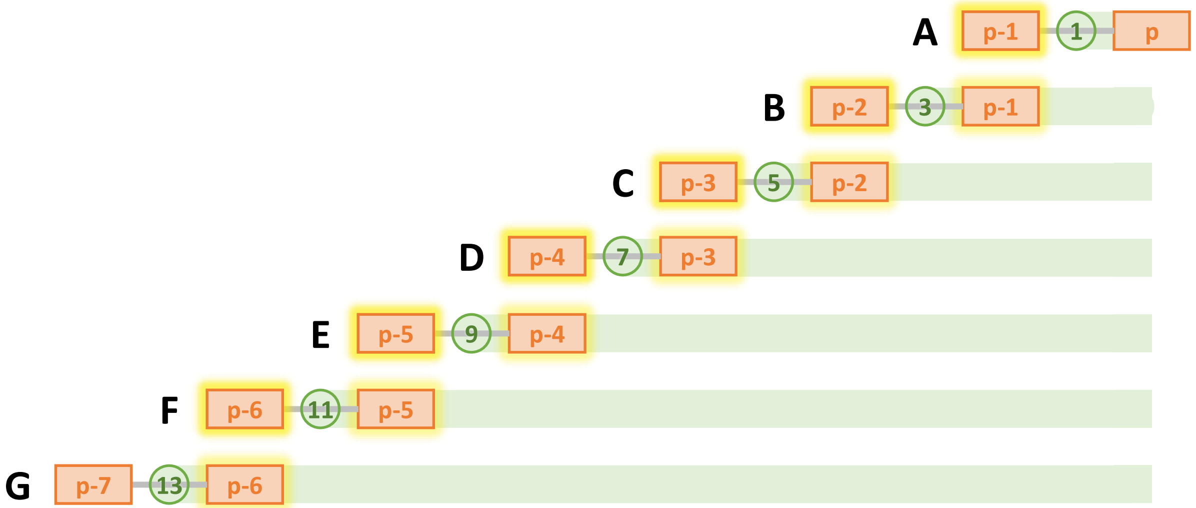



## Series 2

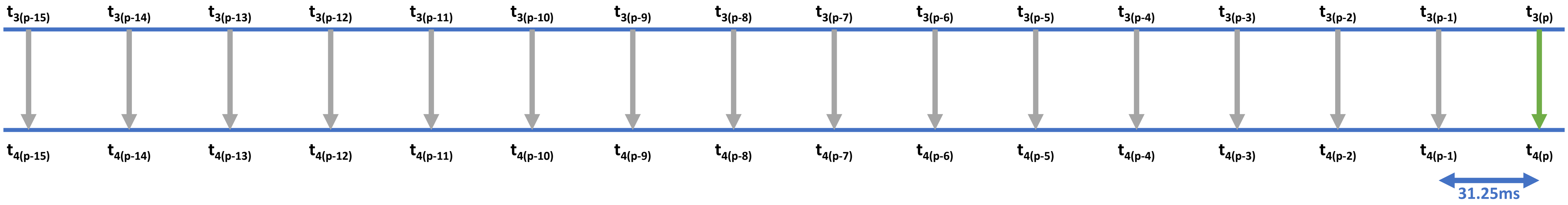
**pDelay Interval = 31.25ms**

**N = 1**

**M = 7**



# mNRR Smoothing – Combining N & M

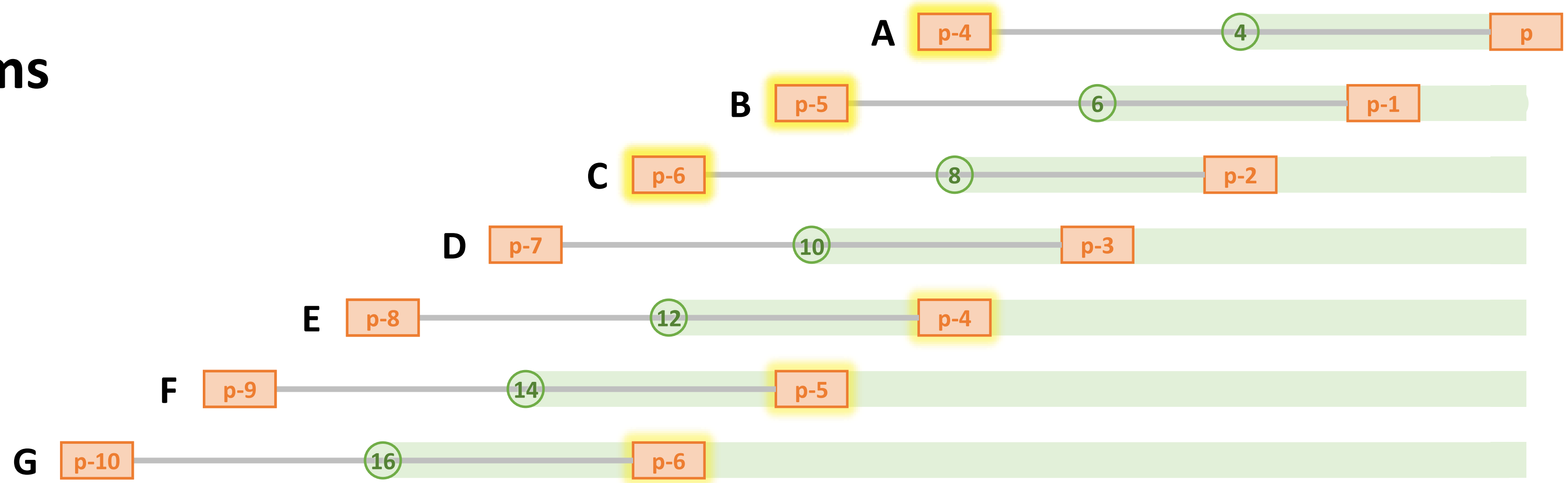



## Series 2

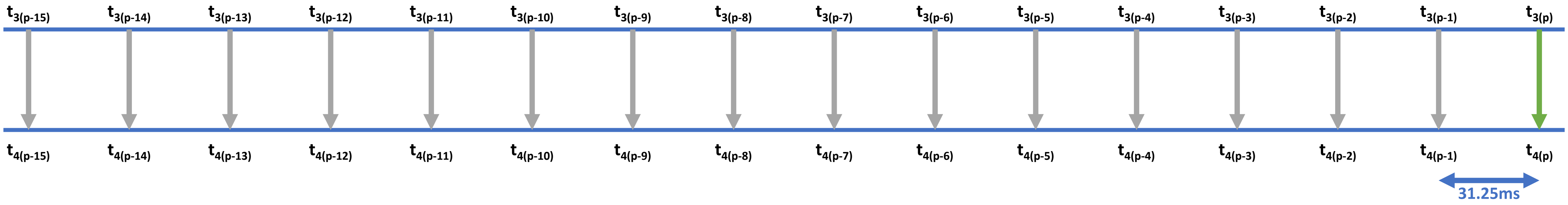
**pDelay Interval = 31.25ms**

**N = 4**

**M = 7**



# mNRR Smoothing – Combining N & M

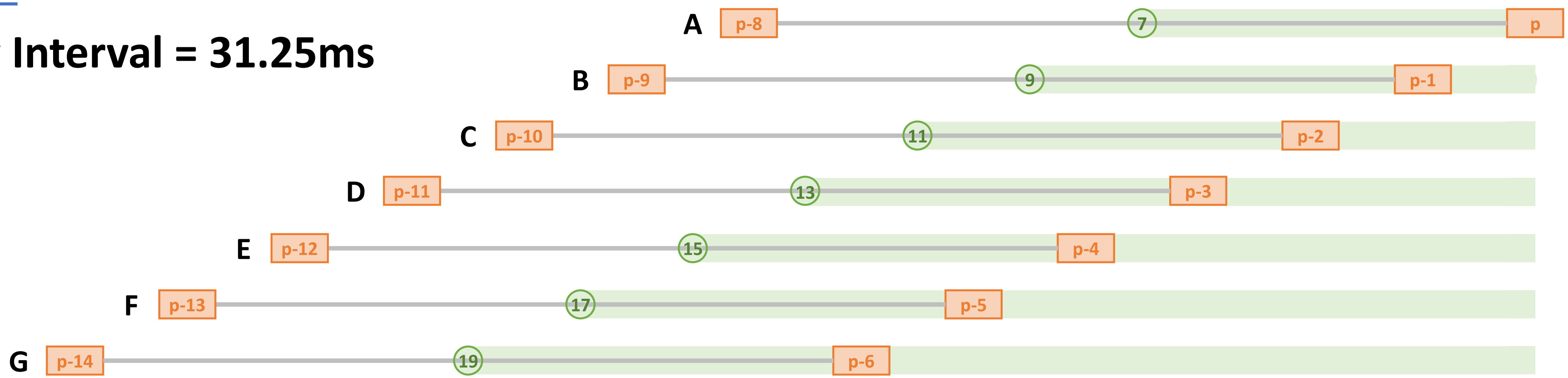



## Series 2

**pDelay Interval = 31.25ms**

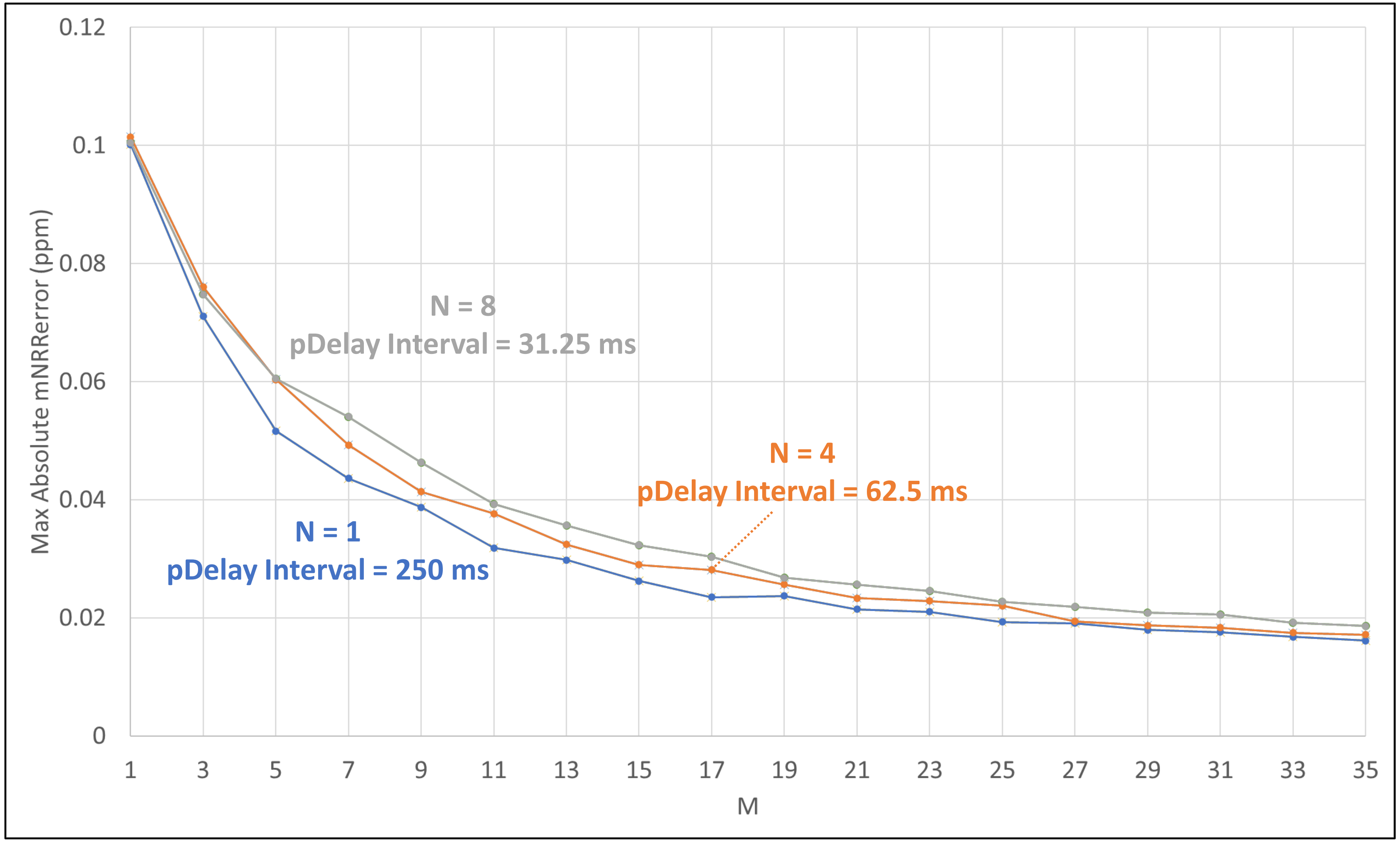
**N = 7**

**M = 7**



# Varying M & N – No Clock Drift

(N x pDelayInterval maintained at 250ms)

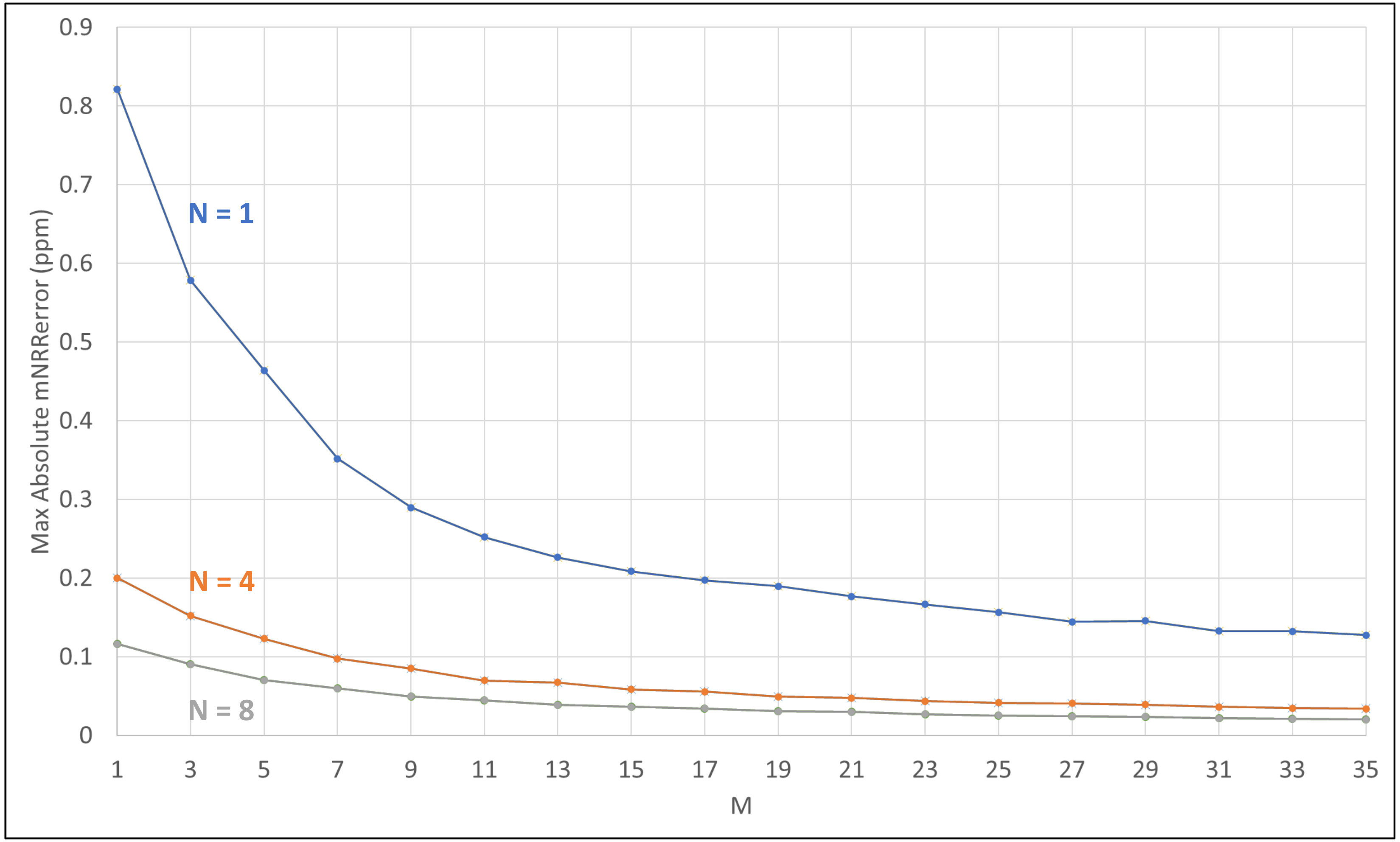



| Input Errors | | |
|---|---|---|
| Clock Drift | ±0 | ppm/s |
| Timestamp Granularity Error | ±4 | ns |
| Dynamic Time Stamp Error | ±4 | ns |
| **Configuration** | | |
| Hops | 1 | |
| Runs (per value of M) | 100,000 x 10 | |

Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $mNRR_{error}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of 250ms.

# Varying M & N – No Clock Drift

(N x pDelayInterval maintained at 250ms)

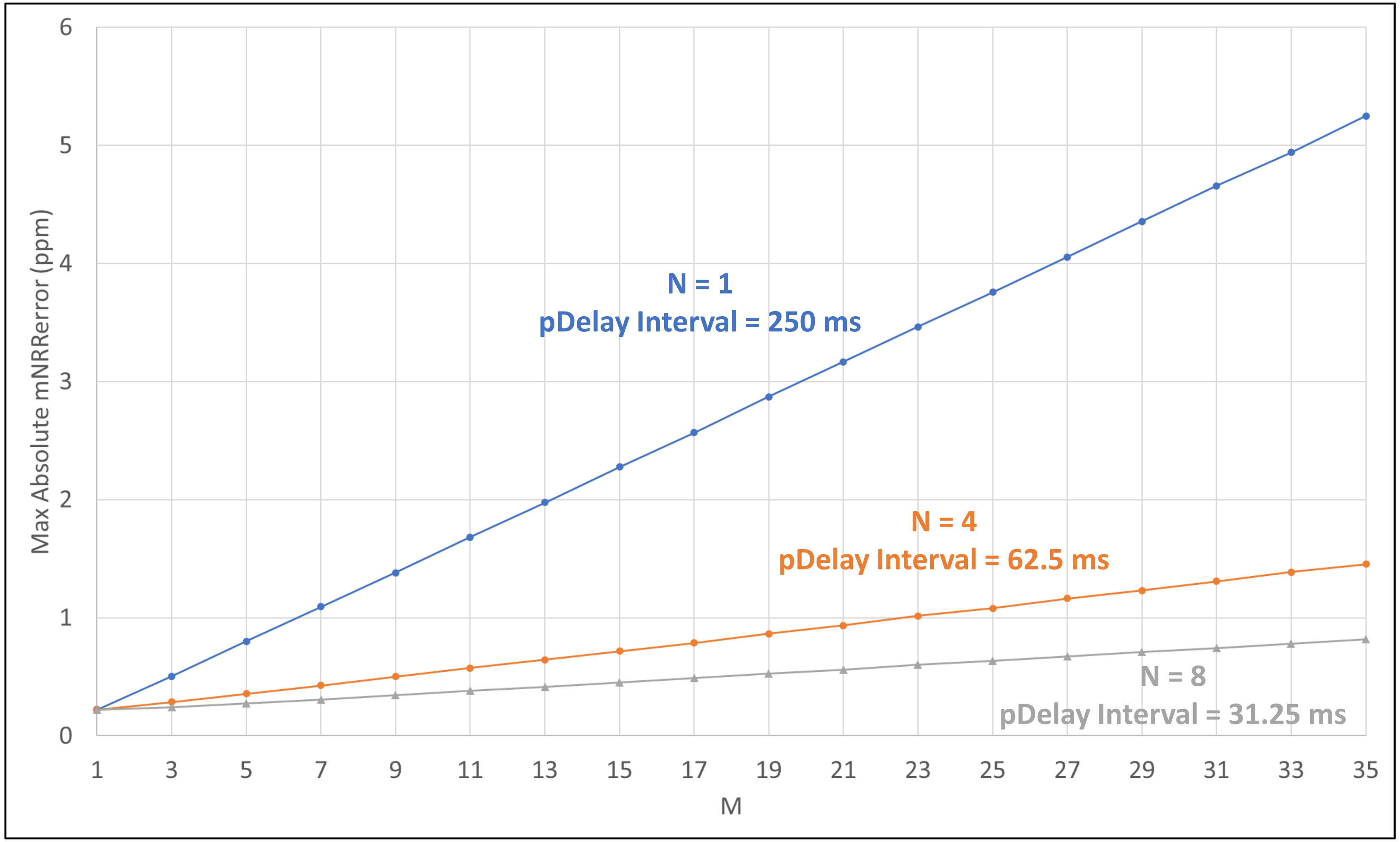



| Input Errors | | |
|---|---|---|
| Clock Drift | ±0.6 | ppm/s |
| Timestamp Granularity Error | ±4 | ns |
| Dynamic Time Stamp Error | ±4 | ns |
| **Input Parameters** | | |
| pDelay Interval | 31.25 | ms |
| **Configuration** | | |
| Hops | 1 | |
| Runs (per value of M) | 100,000 x 10 | |

Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $mNRR_{error}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of 250ms.

# Varying M & N – With Clock Drift

(N x pDelayInterval maintained at 250ms)

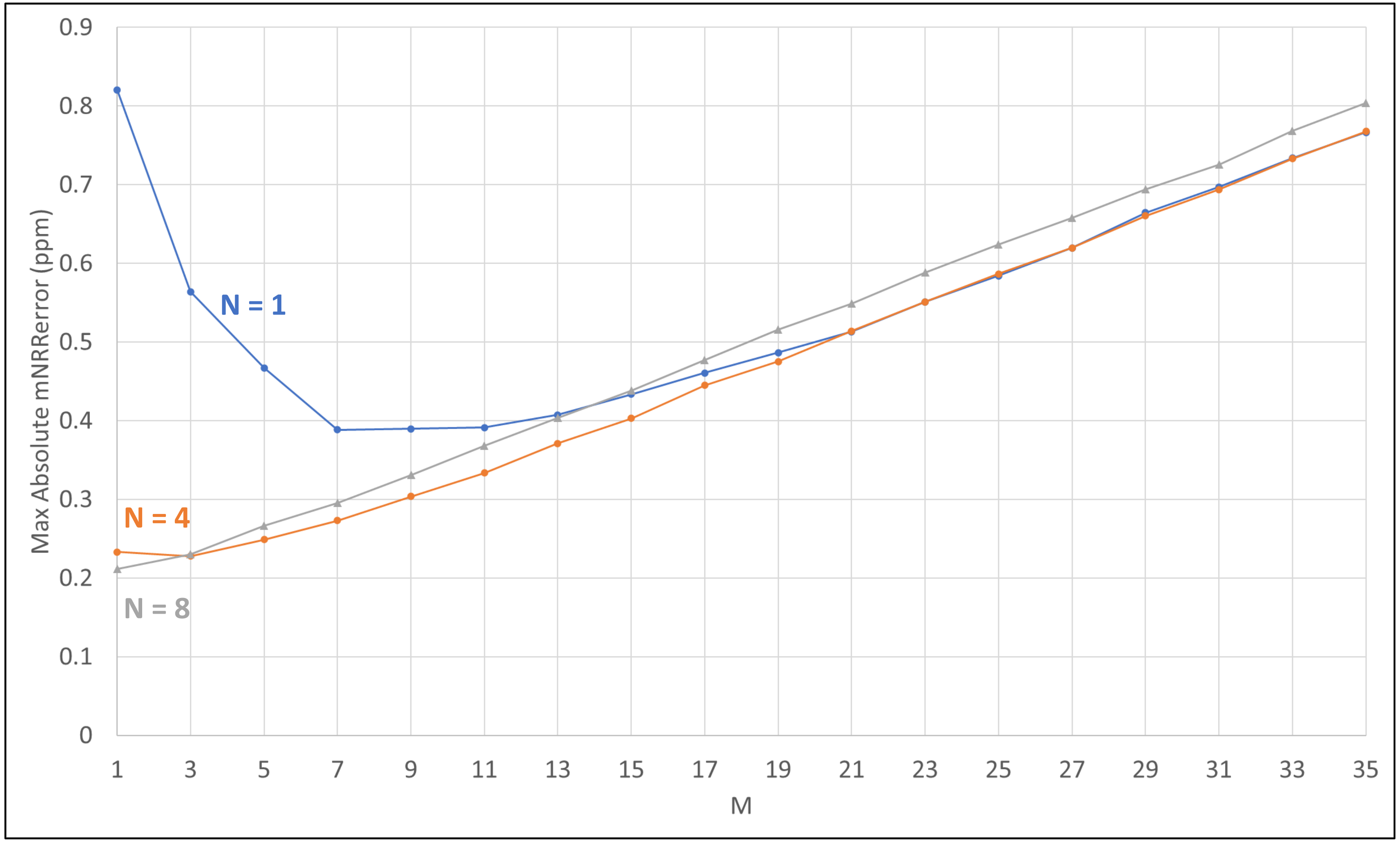



| Input Errors | | |
|---|---|---|
| Clock Drift | ±0 | ppm/s |
| Timestamp Granularity Error | ±4 | ns |
| Dynamic Time Stamp Error | ±4 | ns |
| **Configuration** | | |
| Hops | 1 | |
| Runs (per value of M) | 100,000 x 10 | |

Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $mNRR_{error}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of 250ms.

# Varying M & N – With Clock Drift

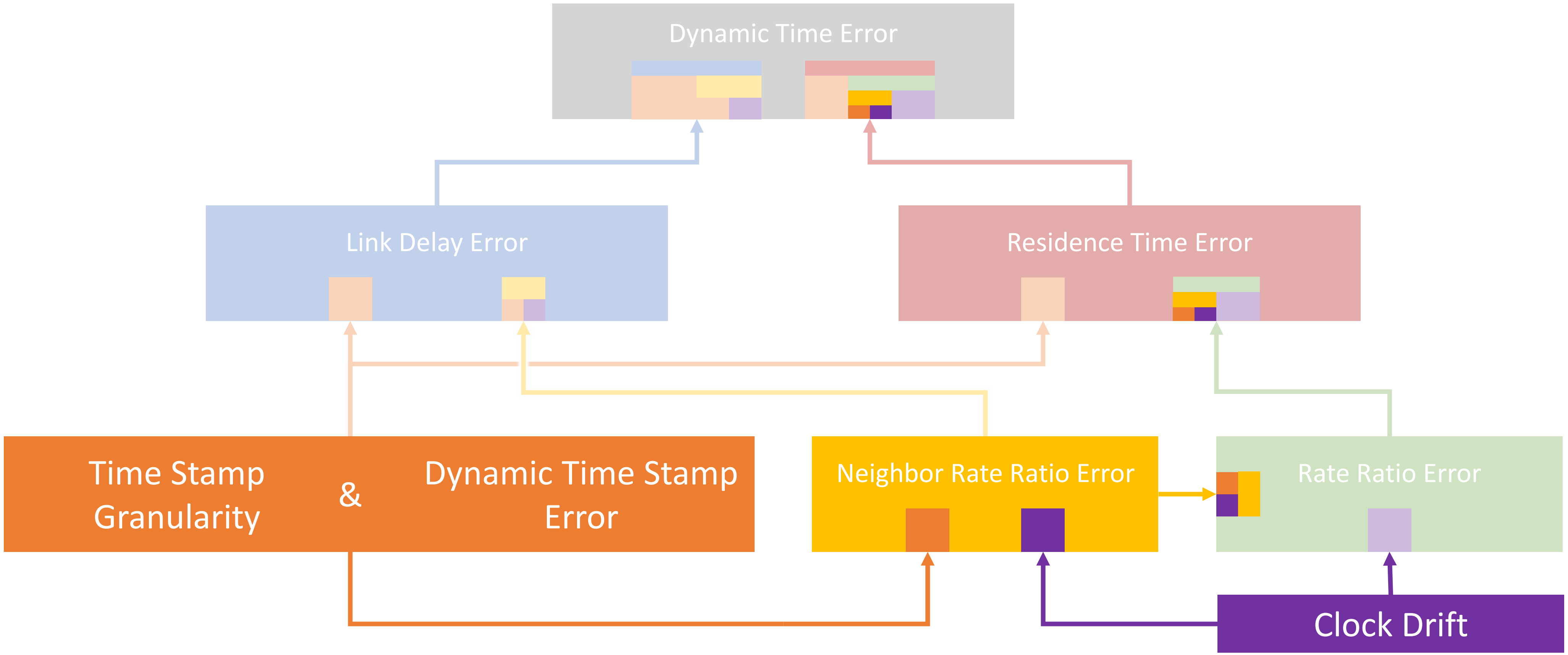
(N x pDelayInterval maintained at 250ms)




| Input Errors | | |
|---|---|---|
| Clock Drift | ±0.6 | ppm/s |
| Timestamp Granularity Error | ±4 | ns |
| Dynamic Time Stamp Error | ±4 | ns |
| **Input Parameters** | | |
| pDelay Interval | 31.25 | ms |
| **Configuration** | | |
| Hops | 1 | |
| Runs (per value of M) | 100,000 x 10 | |

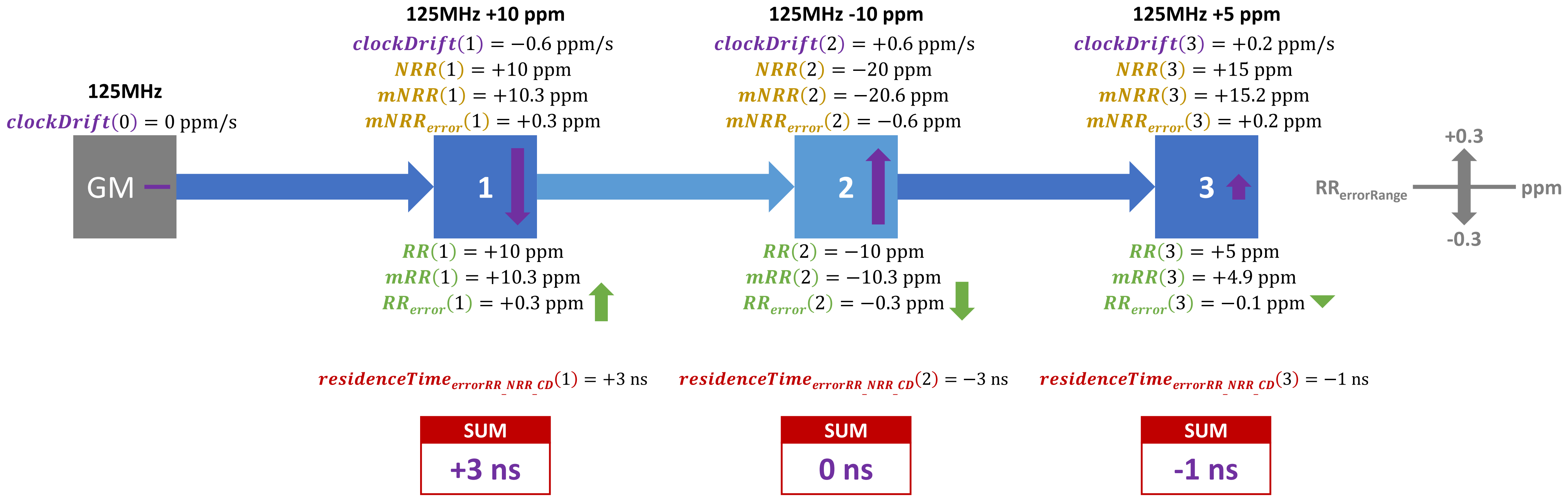
Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $mNRR_{error}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of 250ms.

# NRR Errors Accumulate in RR Error

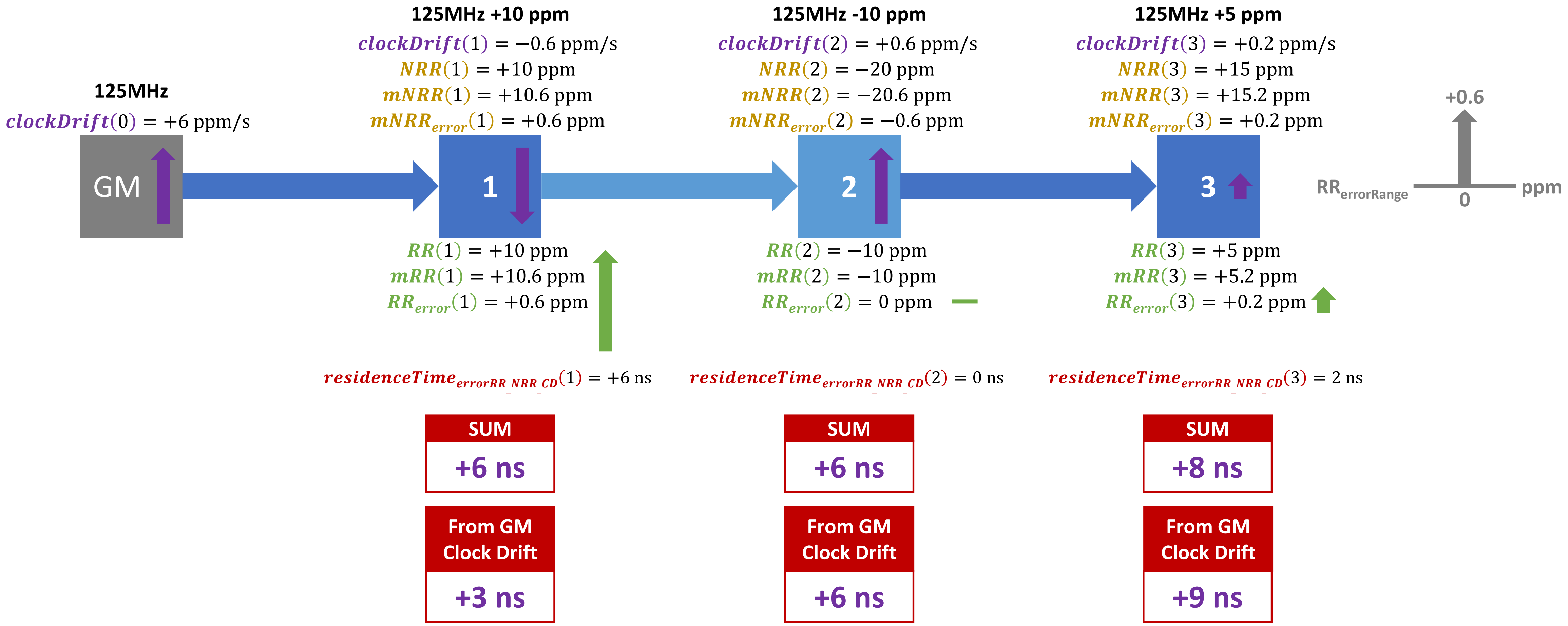



**But there is a big caveat for how NRR errors due to Clock Drift accumulate...**

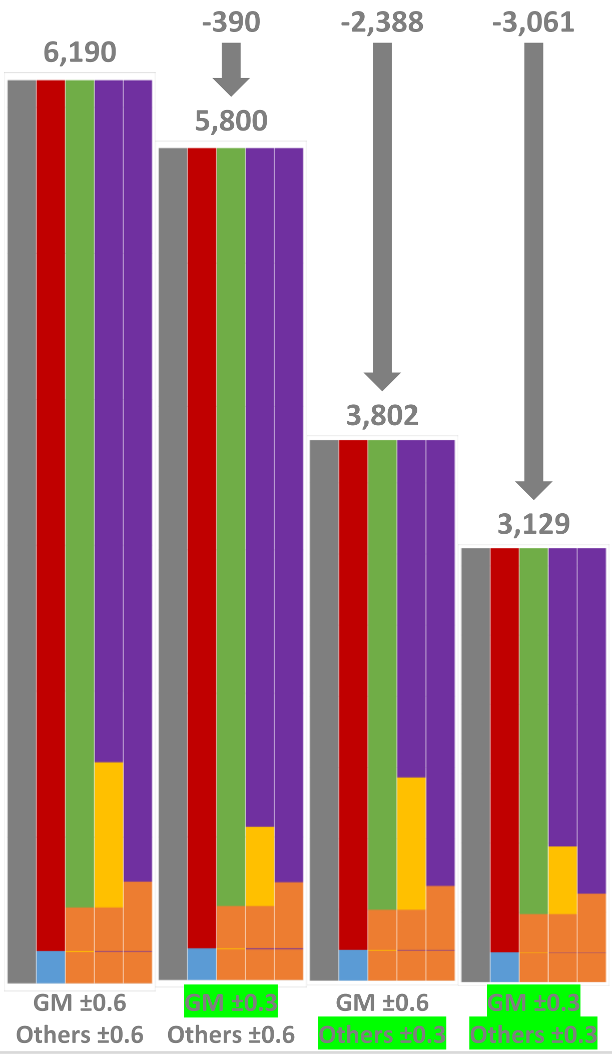
# How $residenceTime_{errorRR\_NRR\_CD}$ Accumulates



# How $residenceTime_{errorRR\_NRR\_CD}$ Accumulates



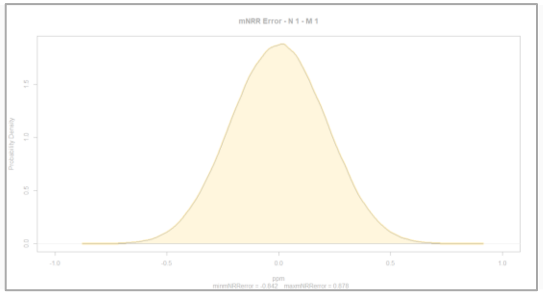
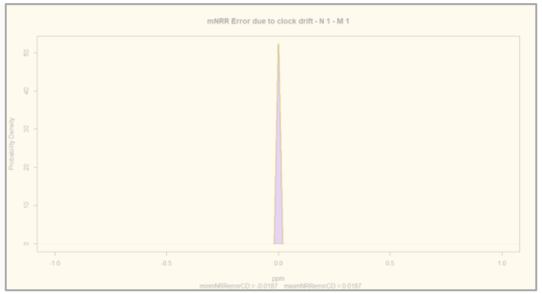
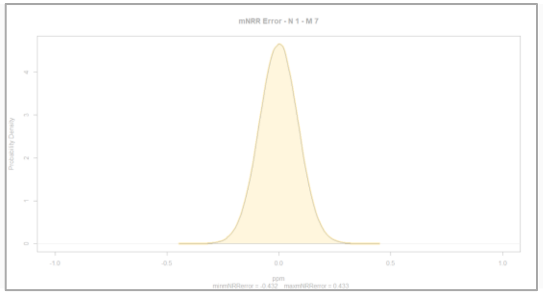
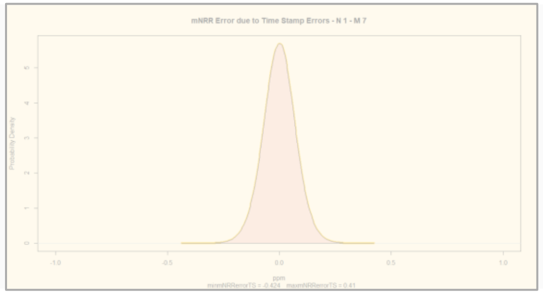
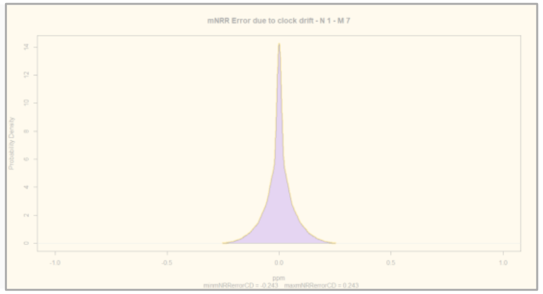
# Clock Drift Sensitivity

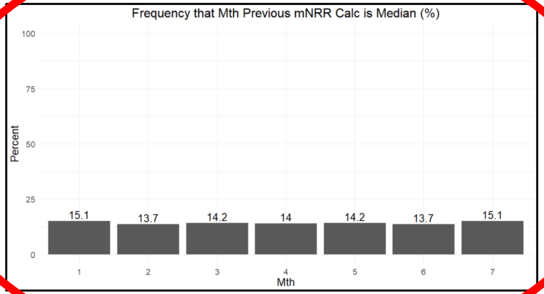



| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | Variable | ppm/s |
| GM Clock Drift Min | Variable | ppm/s |
| Clock Drift (non-GM) | Variable | ±ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | 1000 | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

# Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

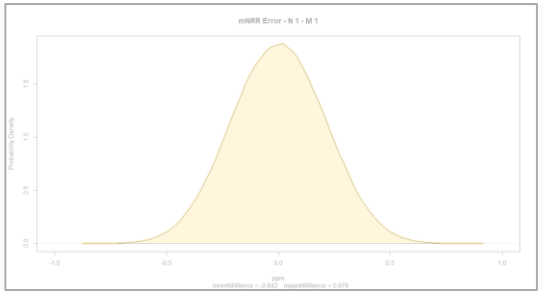
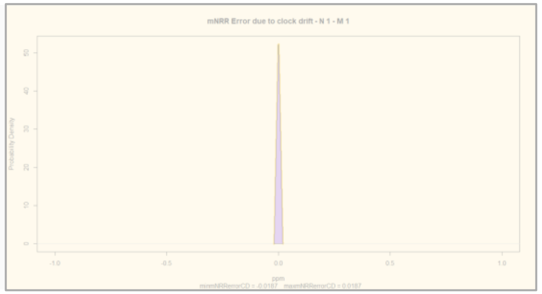
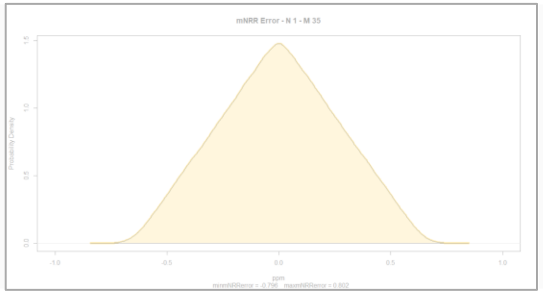
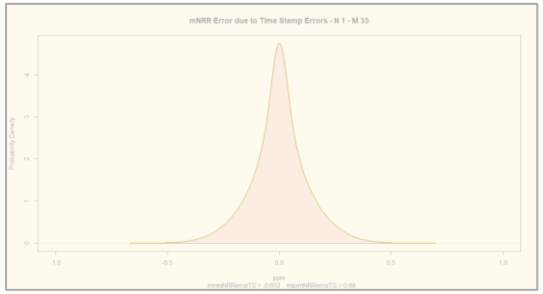
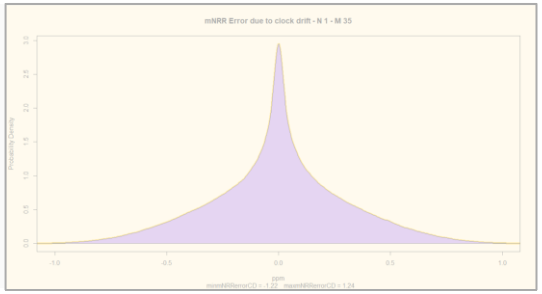
| | pDelay Interval | $mNRR_{error}$ | ...due to Timestamp Error | ...due to Clock Drift |
|---|---|---|---|---|
| **M=1** | **31.25ms** |   |   |   |
| **M=7** | **31.25ms** |   |   |   |

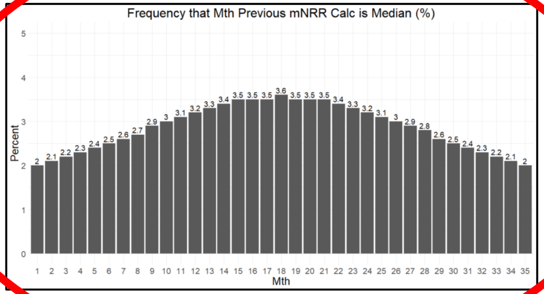



- For values of pDelay Interval where $mNRR_{error}$ due to Timestamp error dominates error due to Clock Drift, taking the median of previous NRR calculations can reduce $mNRR_{error}$.
- This raises the question: what is the optimal value of M?

# Effect of Taking Median of mNRR Calculations

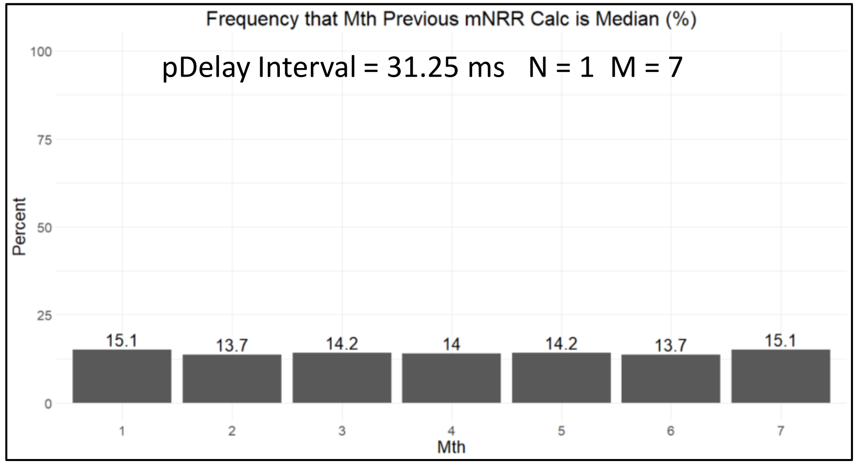
(mNRR – Single Hop – N=1)

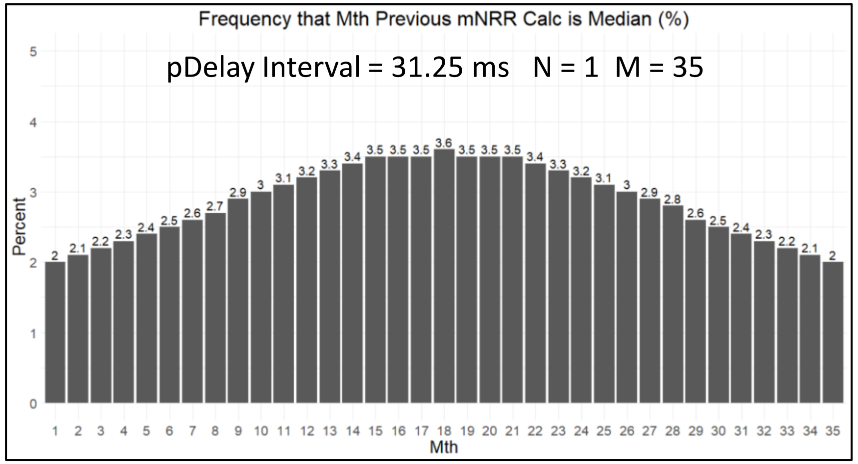
| | pDelay Interval | $mNRR_{error}$ | ...due to Timestamp Error | ...due to Clock Drift |
|---|---|---|---|---|
| M=1 | 31.25ms |  |  |  |
| M=35 | 31.25ms |  |  |  |




- Increasing M to 35 results in using NRR calculations from up to 1093.75ms in the past...and an $mNRR_{errror}$ distribution that is similar magnitude to pDelayInterval of 1000ms (and M=1).
- However...

# Effect of M on Accumulation of errors due to Clock Drift in Rate Ratio



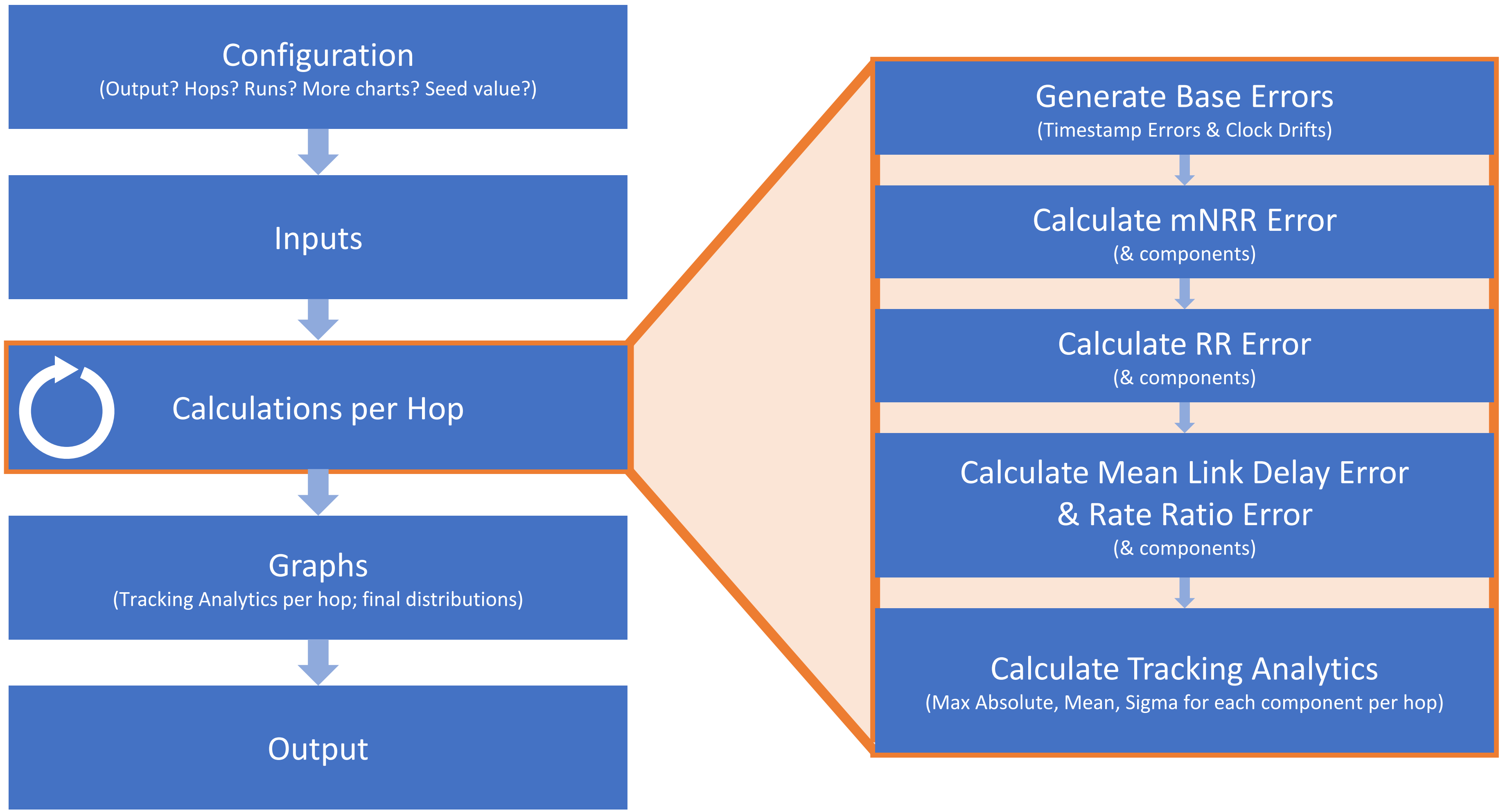




- Using the median of past NRR calculations brings with it a chance that a larger portion of the error won’t cancel out at the next node but will instead survive.
  - This portion will behave more like mNRRerror due to Clock Drift at the GM than at other nodes when M=1.

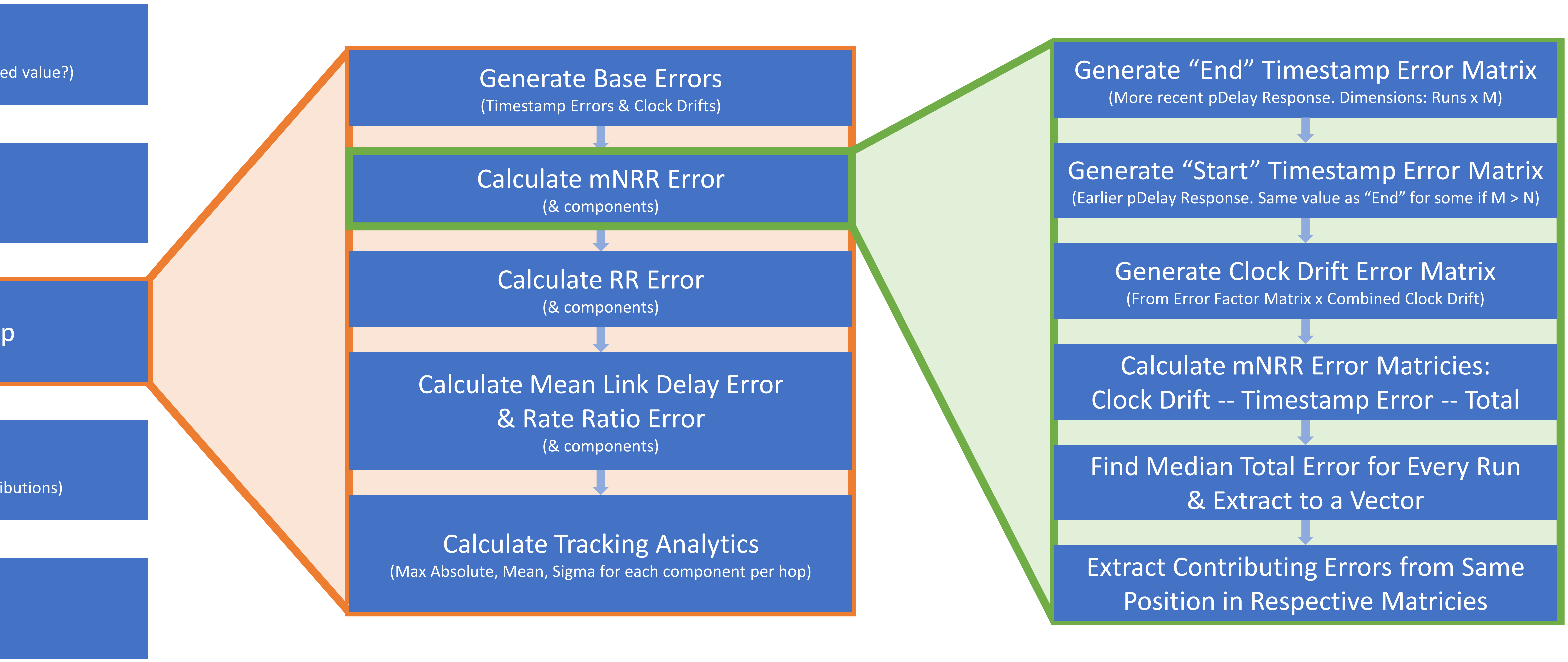
# Summary

- Using a median of past NRR calculations can reduce $mNRR_{error}$ when pDelay Interval is low.
  - This is when Timestamp Errors dominate errors due to Clock Drift
  - Effectiveness is reduced for larger pDelay Intervals and for increasing values of N, which has a similar effect.
  - If Clock Drift compensation is successful, Timestamp errors will dominate a larger values of pDelay Interval and N than otherwise.
- There are complicating factors...
  - Risks increased accumulation of errors due to Clock Drift in Rate Ratio (via NRR)
  - Can result in non-gaussian error distributions
    - Unexpected effects on how errors accumulate
    - Can't use sigma to compare magnitude of error

# RStudio Script Summary



# RStudio Script Summary



# Algorithms – Mean Link Delay

Potential algorithm for Mean Link Delay Averaging

# Mean Link Delay Averaging

- Wired connection link delay is very stable
- pDelay measurements can be noisy due to Timestamp Errors
- It should be possible to average out errors over time
  - Low bandwidth IIR filter...but need to be careful about start-up behaviour

# Mean Link Delay Averaging – Possible Algorithm

- For Xth pDelay measurement since initialisation...

$$MeanLinkDelay(1) = pDelay(1)$$

$$MeanLinkDelay(X) = \frac{\big(MeanLinkDelay(X-1) \times (X-1)\big) + pDelay(X)}{X}$$

- So for 100$^{th}$ pDelay measurement...

$$MeanLinkDelay(100) = \frac{\big(MeanLinkDelay(99) \times (99)\big) + pDelay(100)}{100}$$

- Put a cap on X? 1000?
- Reset X if pDelay deviates too much from current MeanLinkDelay?
  - Deviates too much...repeatedly?

# Algorithms – Clock Drift

Potential algorithm Clock Drift Compensation

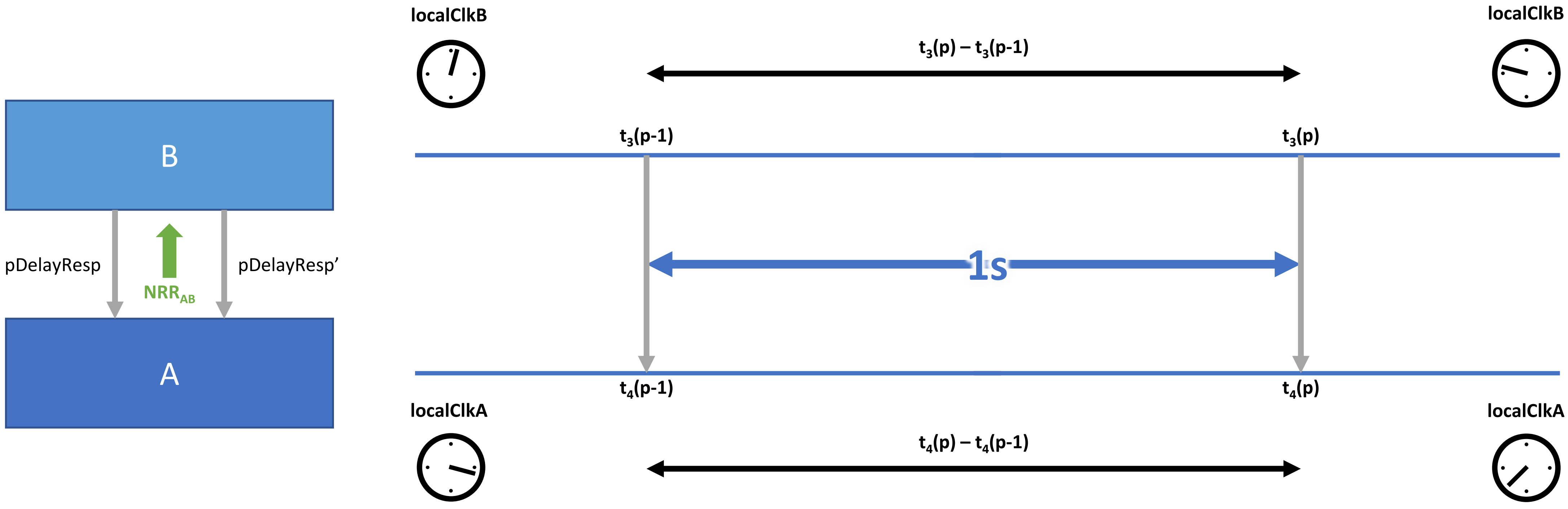
# Compensating for Clock Drift

- Very hard to measure clock drift of an individual device...
  ...but we don't care about the clock drift of an individual device
- We care about Neighbor Rate Ratio and Rate Ratio...and we are constantly measuring both
- We can use theses measurements to track clock drift between two devices and create a correction factor
- The simplest algorithm would be to assume that clock drift is linear over the period of time we are interested in, i.e. one pDelay Interval
  - Actually, up to...

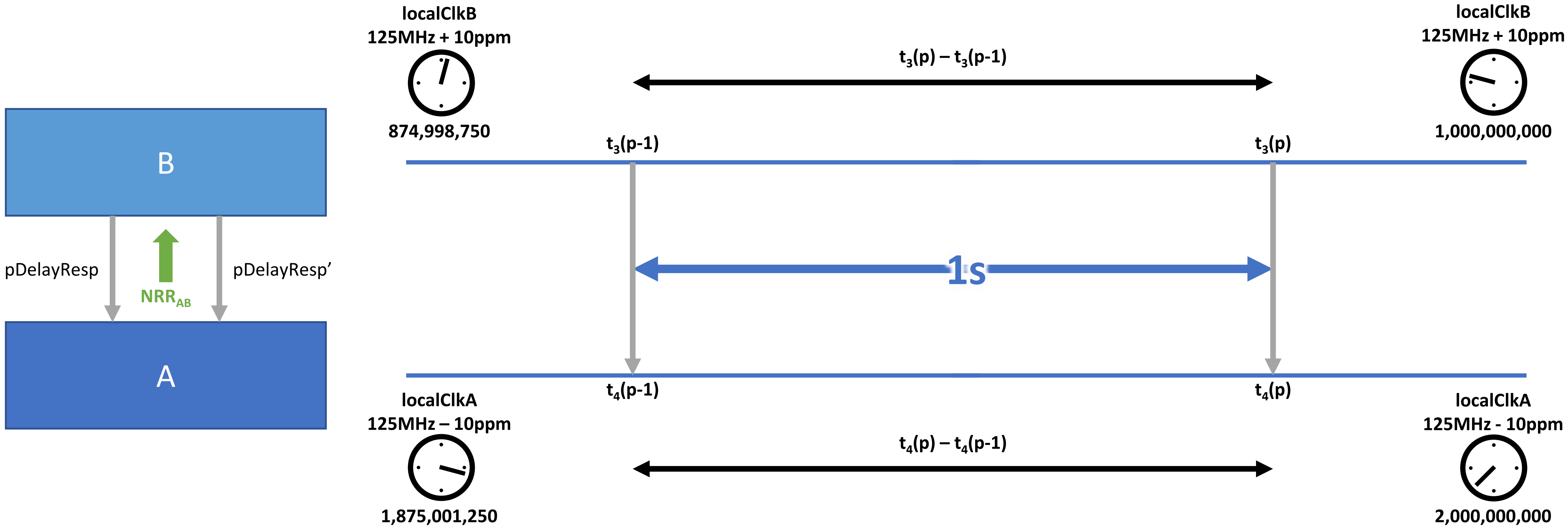
$$pDelayInterval + \frac{pDelayInterval \times (N + 2M - 1)}{2}$$

Maximum delay between attempt to measure NRR and using NRR as part of Sync messaging

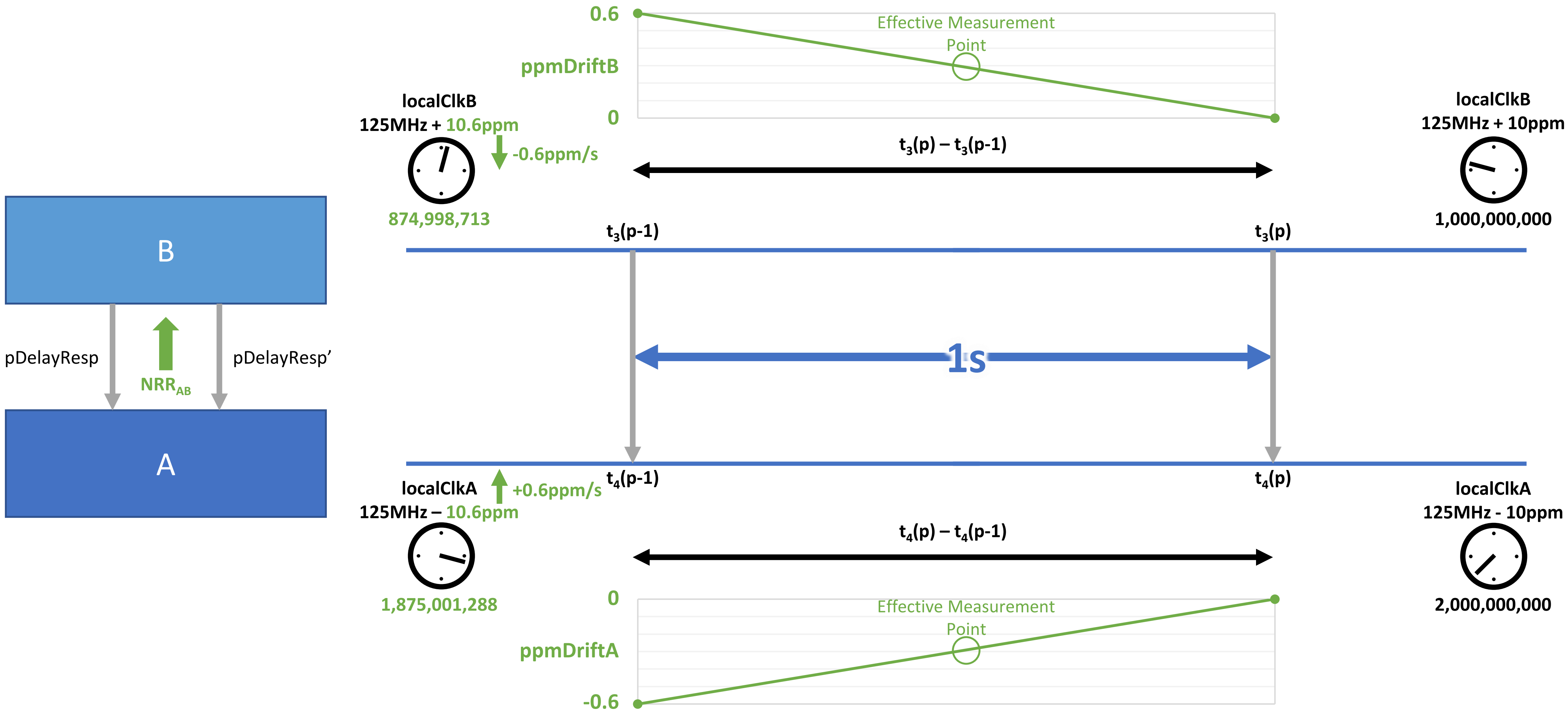
Maximum delay between attempt to measure NRR and effective measurement point



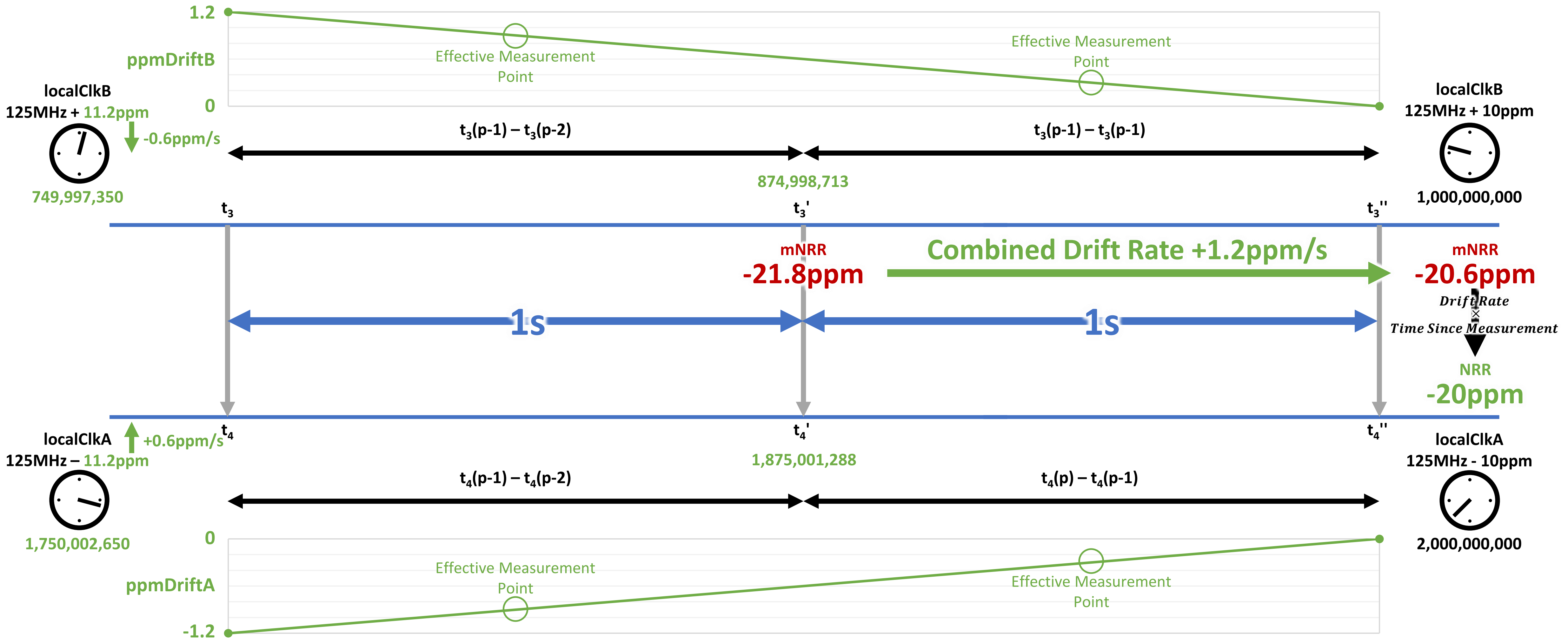
B
A
pDelayResp
pDelayResp’
NRR_AB
localClkB
localClkB
t3(p) – t3(p-1)
t3(p-1)
t3(p)
1s
t4(p-1)
t4(p)
localClkA
localClkA
t4(p) – t4(p-1)



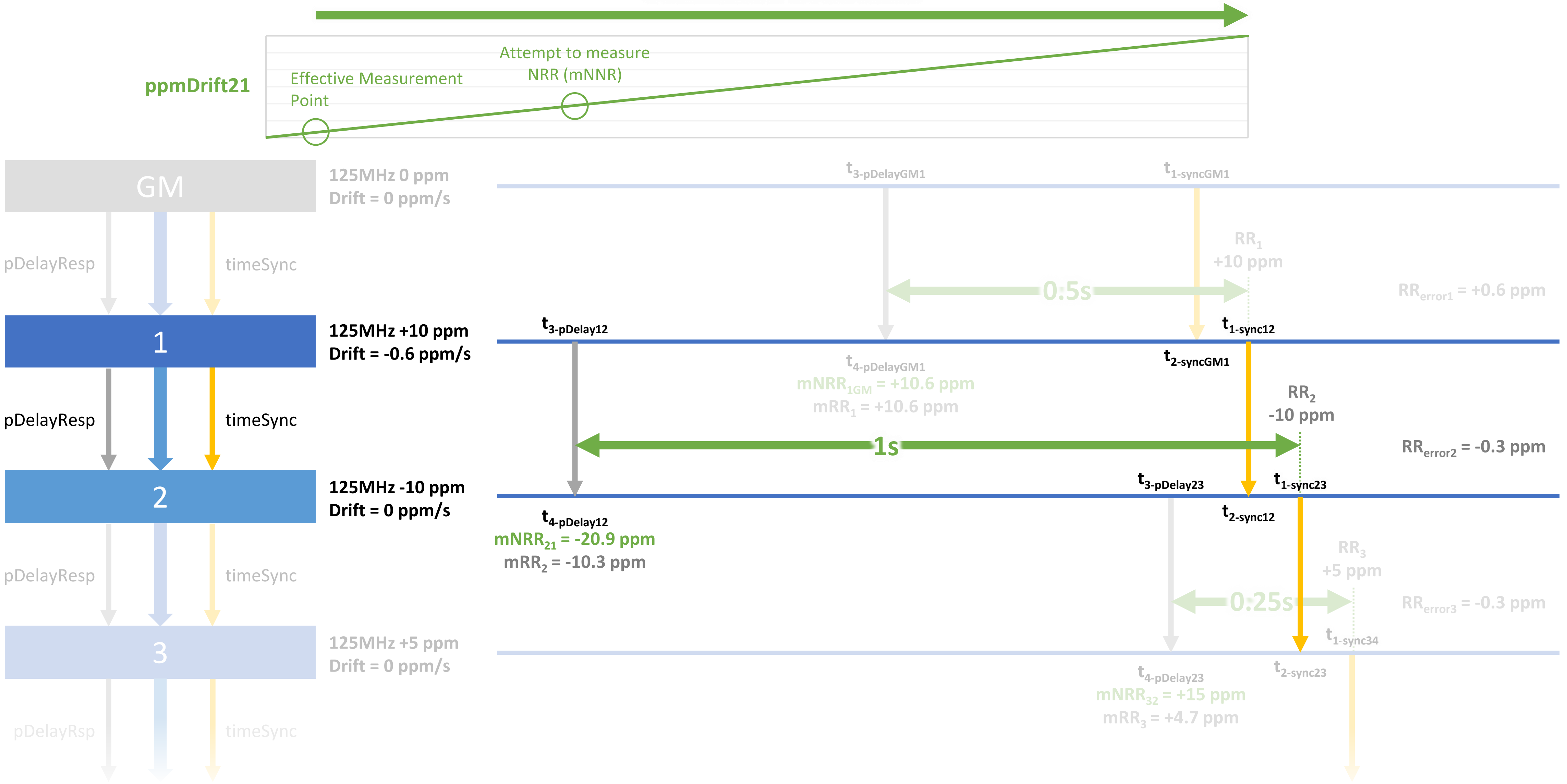

$$NRR_{AB} \text{ @ } t_4' = \frac{(t_4(p) - t_4(p\text{-}1))}{(t_3(p) - t_3(p\text{-}1))} = \frac{124{,}998{,}750}{125{,}001{,}250} \rightarrow -0.002\% = -20 \text{ ppm}$$




$$\mathbf{NRR_{AB}\ @\ t_4'} = \frac{(t_4(p) - t_4(p\text{-}1))}{(t_3(p) - t_3(p\text{-}1))} = \frac{124{,}998{,}712}{125{,}001{,}288} \rightarrow -0.00206\% = -20.6\ \text{ppm}$$


1.2
ppmDriftB
0
Effective Measurement Point
Effective Measurement Point
localClkB
125MHz + 11.2ppm
-0.6ppm/s
749,997,350
t3(p-1) – t3(p-2)
t3(p-1) – t3(p-1)
874,998,713
localClkB
125MHz + 10ppm
1,000,000,000
t3
t3'
t3''
mNRR
-21.8ppm
Combined Drift Rate +1.2ppm/s
mNRR
-20.6ppm
DriftRate
Time Since Measurement
NRR
-20ppm
1s
1s
t4
t4'
t4''
localClkA
125MHz – 11.2ppm
+0.6ppm/s
1,750,002,650
1,875,001,288
t4(p-1) – t4(p-2)
t4(p) – t4(p-1)
localClkA
125MHz - 10ppm
2,000,000,000
0
ppmDriftA
-1.2
Effective Measurement Point
Effective Measurement Point

## Combined Drift Rate



# Clock Drift Compensation – Possible Algorithm – 1

- Assume clock drifts linearly over the period of interest.
- Calculate Drift Rate

$$NRR_{driftRate} = \frac{mNRR(p) - mNRR(p-1)}{Time_{effectiveMeasure}(p) - Timee_{ffectiveMeasure}(p-1)}$$

- Most of the time this will simplify to...

$$NRR_{driftRate} = \frac{mNRR(p) - mNRR(p-1)}{N \times pDelayInterval}$$

- ...but if taking a median of past NRR calculations it will get more complicated
  - If M>N ehe effective measurement time of (p) could be earlier than (p-1)

# Clock Drift Compensation – Possible Algorithm – 2

- Apply the correction factor. Example for NRR applied during Sync messaging (ppm)...

$$NRR_{sync} = mNRR + NRR_{driftRate}\left(delay_{mNRR\_sync} + \frac{pDelayInterval \times N}{2}\right)$$

- If a median of past NRR calculations is taken...

$$NRR_{sync} = mNRR + NRR_{driftRate}\left(delay_{mNRR\_sync} + \frac{pDelayInterval \times \left(N + 2\left(M_{used}(p-1) - Mu_{sed}(p)\right) - 1\right)}{2}\right)$$

# What should go into the spec?

- Normative: clock stability?
  - Within device's operating parameters? Including not just temperature, but rate of change of temperature?
- Normative: accuracy of Correction Field?
  - Accuracy of Residence Time Measurement?
  - Accuracy of Mean Link Delay Measurement?
- Normative: accuracy of processing Rate Ratio field, i.e. calculating NRR?
  - Tolerance to Clock Drift of upstream device?
- Informative: how these requirements can be met using example algorithms?

# Next Steps

Including Validating the Monte Carlo Analysis against Time Series Simulation

# Proposed Next Steps

- Time Series Simulations to validate Monte Carlo Analysis
  - Not necessarily with values we would want to use in practice. Main point is to ensure that Monte Carlo Analysis and Time Series Simulations match.
- More Monte Carlo Analysis to develop recommendations
  - Time Series Simulations to validate
- Prepare spec contribution for March Plenary
  - Likely present to 60802 group before then to get guidance on key questions

# Key Questions

- What is the right balance between minimising error and using default parameters. For example...
  - Is a pDelay Interval of 31.25ms acceptable if it reduces error...but not by very much compared to 250ms?
- What is the right balance between minimising mNRR error and maintaining responsiveness?
  - At what point does the assumption of linearity of Clock Drift over period of interest break down? Do we need to create an analysis that includes rate of change of clock drift?
- Is alignment of pDelay messaging with Sync messaging (e.g. pDelay in 100ms before Sync) feasible?
- How realistic is it to implement some of the proposed algorithms?

# Proposed Initial Time Series Simulations

- A – P as detailed below.
- Simulations N - R are possibly the most important as they validate that the goal of 1us accuracy over 100 hops is achievable (without measures such as 1ms Residence Time; more accurate clocks; etc...)
  - Maybe run these first?
- Initially: single replication for each
- If there is time: additional replications for N & O

# Changes from Previous Simulations

- Change Dynamic Timestamp Error from...
  +8ns or -8ns with equal chance of either
  ...to...
  ±4ns (uniform linear distribution)
- Initially remove "take the median of past M calculations" step from mNRR Smoothing.
  - Retain the option to use timestamp from $N^{th}$ previous pDelayResp message when calculating mNRR (aka mNRR Smoothing)
  - Potentially add back "take the median of past M calculations" following further analysis

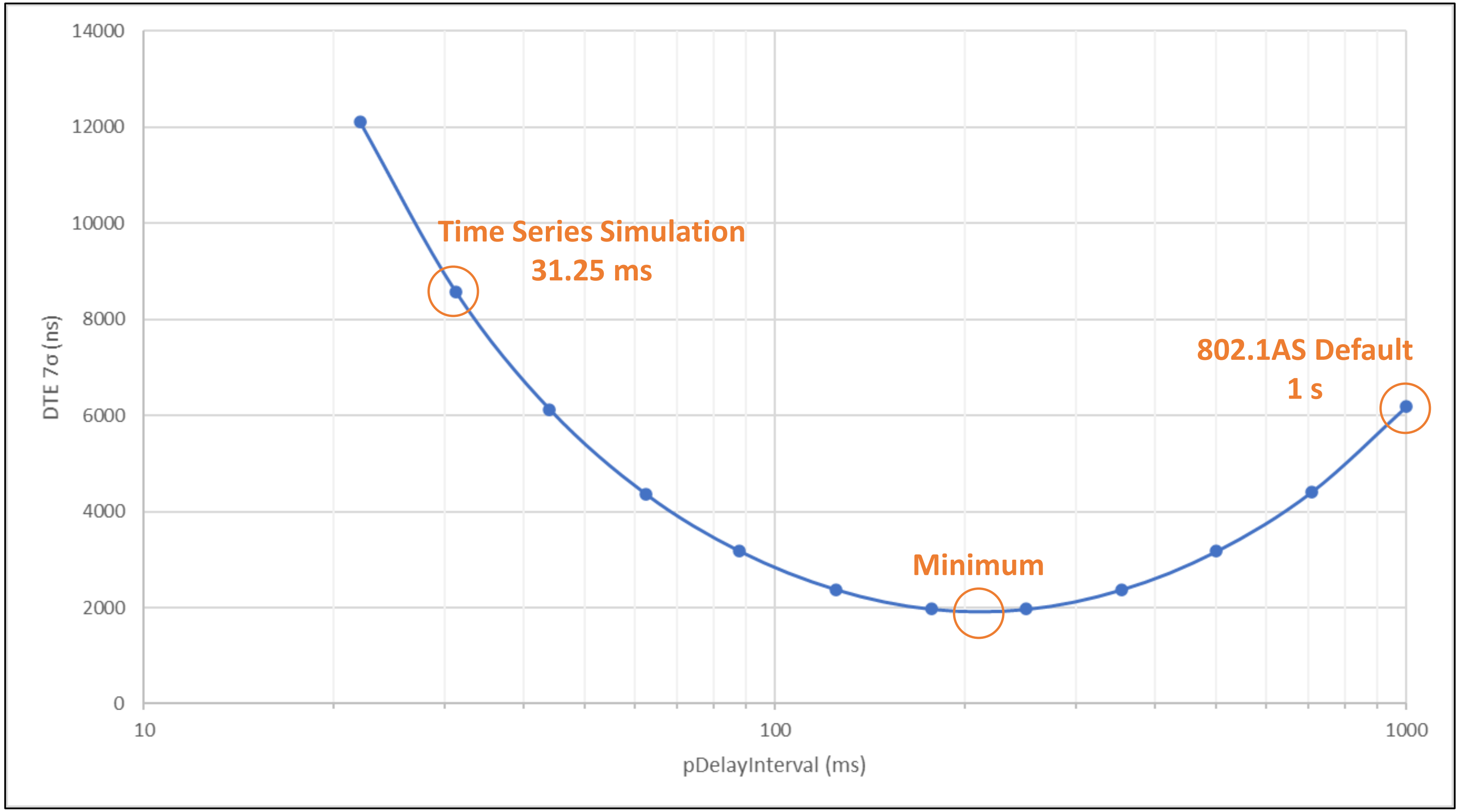
# Proposed Time Series Simulations – Details

| Experiment | Reason | Errors | | | | Parameter | Correction Factors | |
|---|---|---|---|---|---|---|---|---|
| | | GM Clock Drift (±ppm/s) | Other Clock Drift (±ppm/s) | Timestamp Granularity (±ns) | Dynamic Timestamp Error (±ns) | pDelay Interval (ms) | Mean Link Delay Averaging | mNRR Smooting Factor N |
| A | Verify optimised pDelayInterval | 0.6 | 0.6 | 4 | 4 | 1000 | Off | 1 |
| B | | | | | | 250 | | |
| C | | | | | | 31.25 | | |
| D | Verify effect of reduced Timestamp Error (high when pDelayInterval 31.25ms; low when 1000ms) | 0.6 | 0.6 | 2 | 2 | 1000 | Off | 1 |
| E | | | | | | 31.25 | | |
| F | Verify effect of reduced Timestamp Error (high when pDelayInterval 1000ms; low when 31.25ms) | 0.3 | 0.3 | 4 | 4 | 1000 | Off | 1 |
| G | | | | | | 31.25 | | |
| H | Verify Effect of mNRR Smoothing (1000ms pDelayInterval - no benefit; 250ms pDelayInteval - n = 2 is best; 31.25ms pDelayInterval - n = 15 best) | 0.6 | 0.6 | 4 | 4 | 1000 | Off | 2 |
| I | | | | | | 1000 | | 15 |
| J | | | | | | 250 | | 2 |
| K | | | | | | 250 | | 15 |
| L | | | | | | 31.25 | | 2 |
| M | | | | | | 31.25 | | 15 |

# Proposed Time Series Simulations – Details

| Experiment | Reason | Errors | | | | Parameter | Correction Factors | |
|---|---|---|---|---|---|---|---|---|
| | | GM Clock Drift (±ppm/s) | Other Clock Drift (±ppm/s) | Timestamp Granularity (±ns) | Dynamic Timestamp Error (±ns) | pDelay Interval (ms) | Mean Link Delay Averaging | mNRR Smoothing Factor N |
| N | Verify possibility of reaching goal with 90% effective Clock Drift Compensation plus Mean Link Delay Averaging & mNRR Smoothing | 0.06 | 0.06 | 4 | 4 | 250 | On | 6 |
| O | | | | | | 31.25 | | 48 |
| P | Verify possibility of reaching goal with 98% effective Clock Drift Compensation plus Mean Link Delay Averaging & mNRR Smoothing | 0.012 | 0.012 | 4 | 4 | 250 | On | 13 |
| Q | | | | | | 31.25 | | 104 |
| R | Check effect of Mean Link Delay Averaging by switching it off. (98% effective Clock Drift Compensation plus mNRR Smoothing) | 0.012 | 0.012 | 4 | 4 | 250 | Off | 13 |

# A, B & C: Validate *pDelayInterval* Sensitivity Analysis



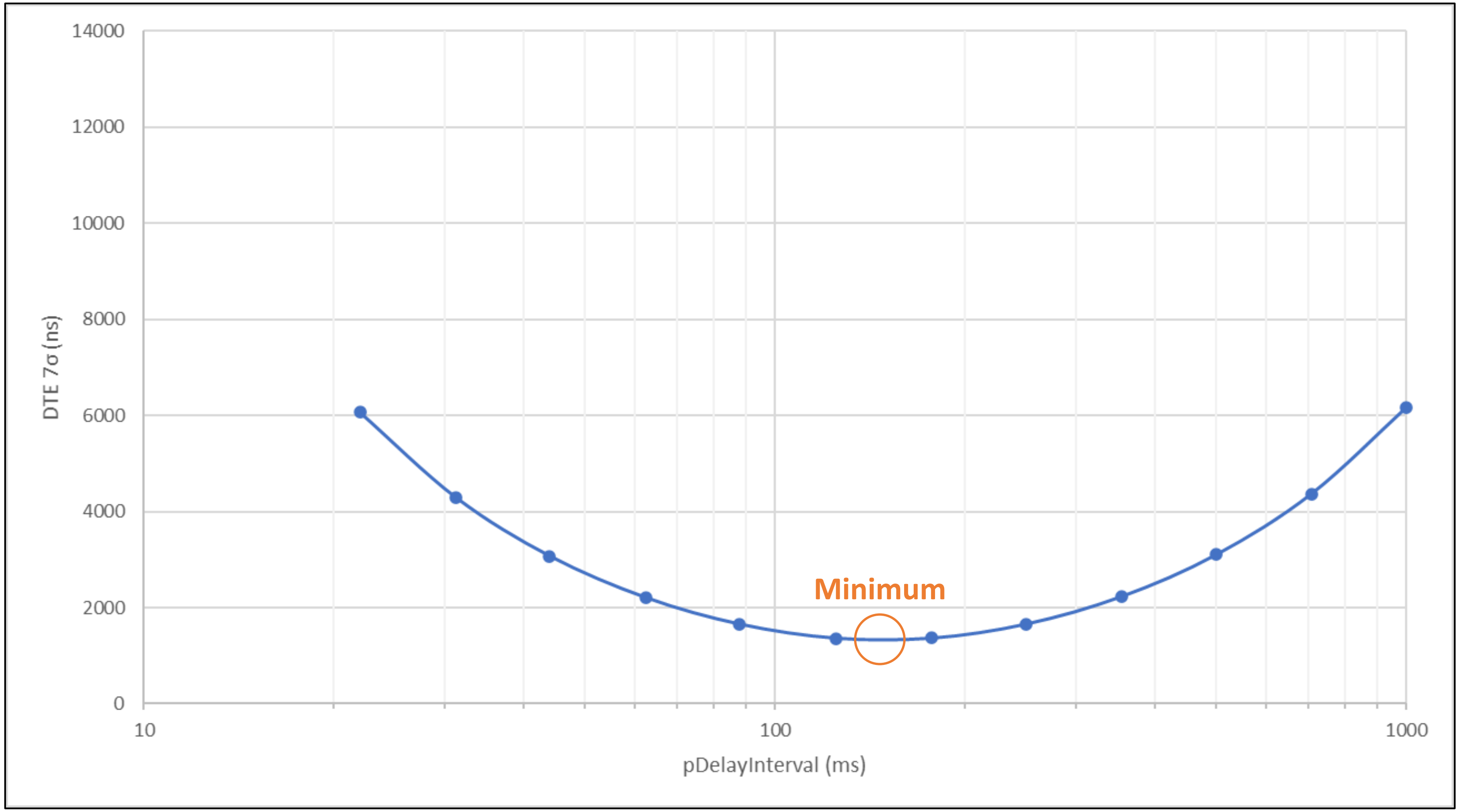

| | | |
|---|---|---|
| **Input Errors** | | |
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | +0.6 | ppm/s |
| Clock Drift (non-GM) | 0.6 | ±ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | Variable | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

Vary This

**Minimum is at approx 210 ms…**

**…for this set of parameters.**

# D & E: Validate *pDelayInterval* Sensitivity Analysis with Lower Timestamp Error

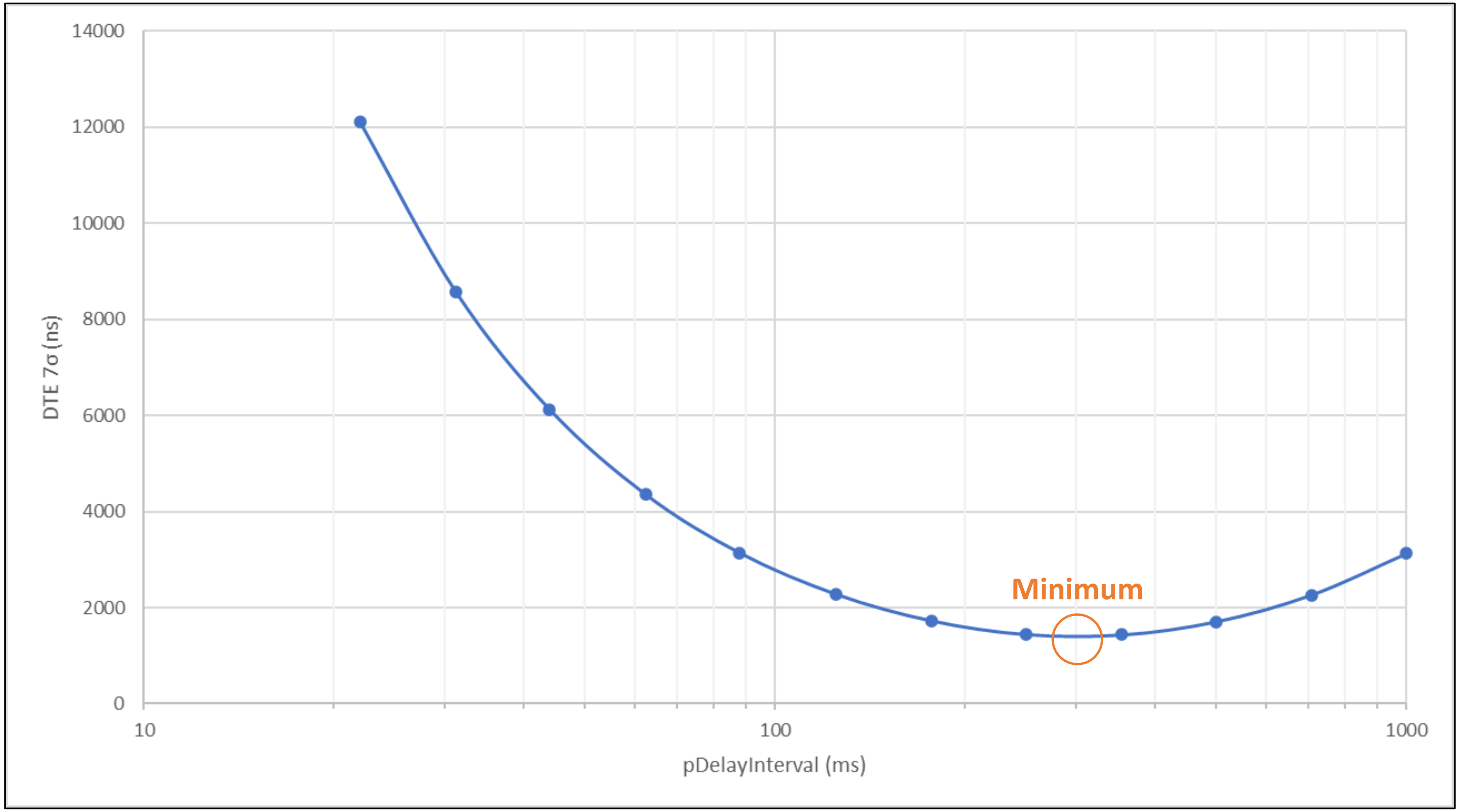



| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | +0.6 | ppm/s |
| Clock Drift (non-GM) | 0.6 | ±ppm/s |
| Timestamp Granularity TX | 2 | ±ns |
| Timestamp Granularity RX | 2 | ±ns |
| Dynamic Time Stamp Error TX | 2 | ±ns |
| Dynamic Time Stamp Error RX | 2 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | Variable | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

**Minimum is at approx 150ms…**

**…for this set of parameters.**

# F & G: Validate *pDelayInterval* Sensitivity Analysis with Lower Clock Drift

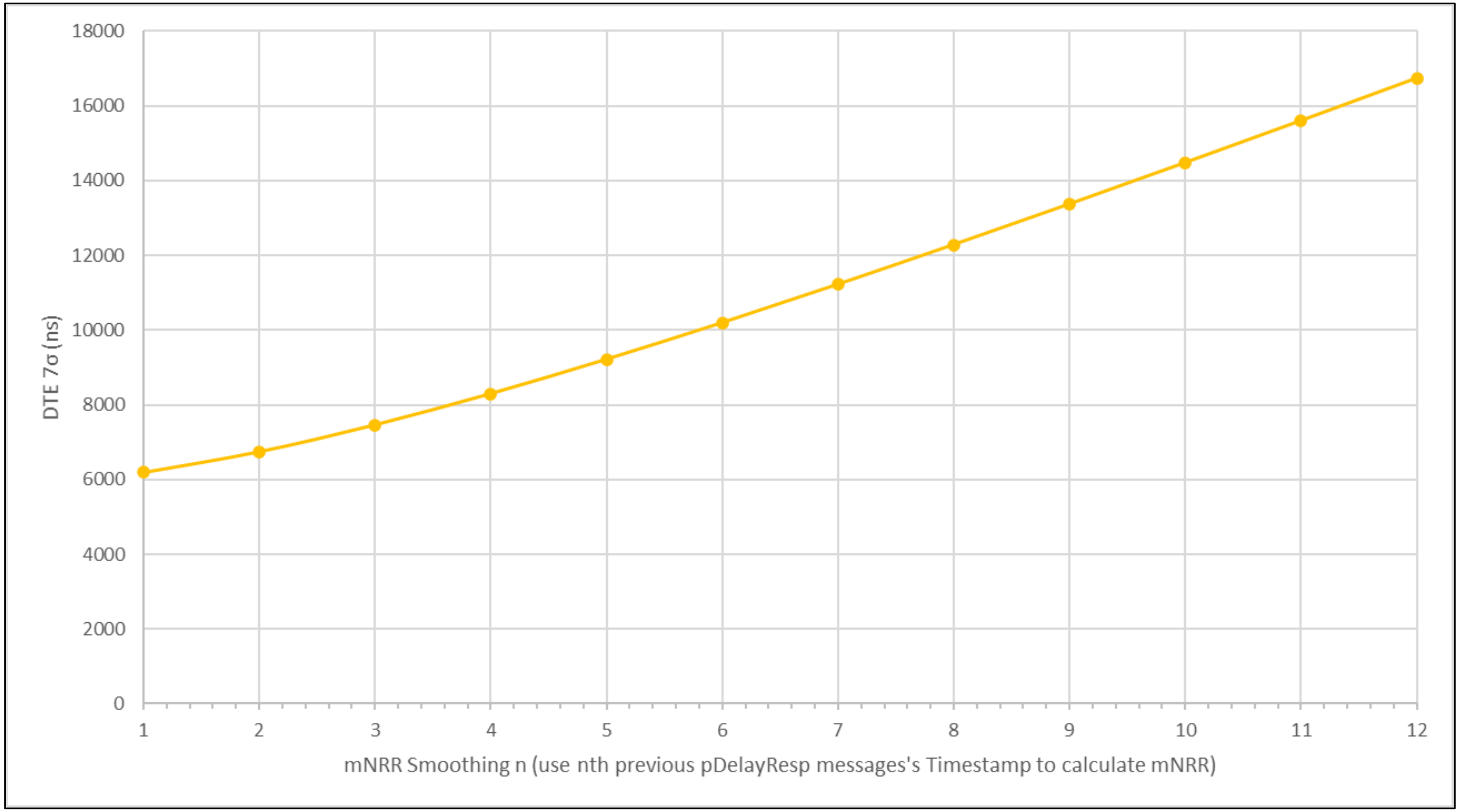



| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.3 | ppm/s |
| GM Clock Drift Min | +0.3 | ppm/s |
| Clock Drift (non-GM) | 0.3 | ±ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | Variable | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing | 1 | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

**Minimum is at approx 300ms…**

**…for this set of parameters.**

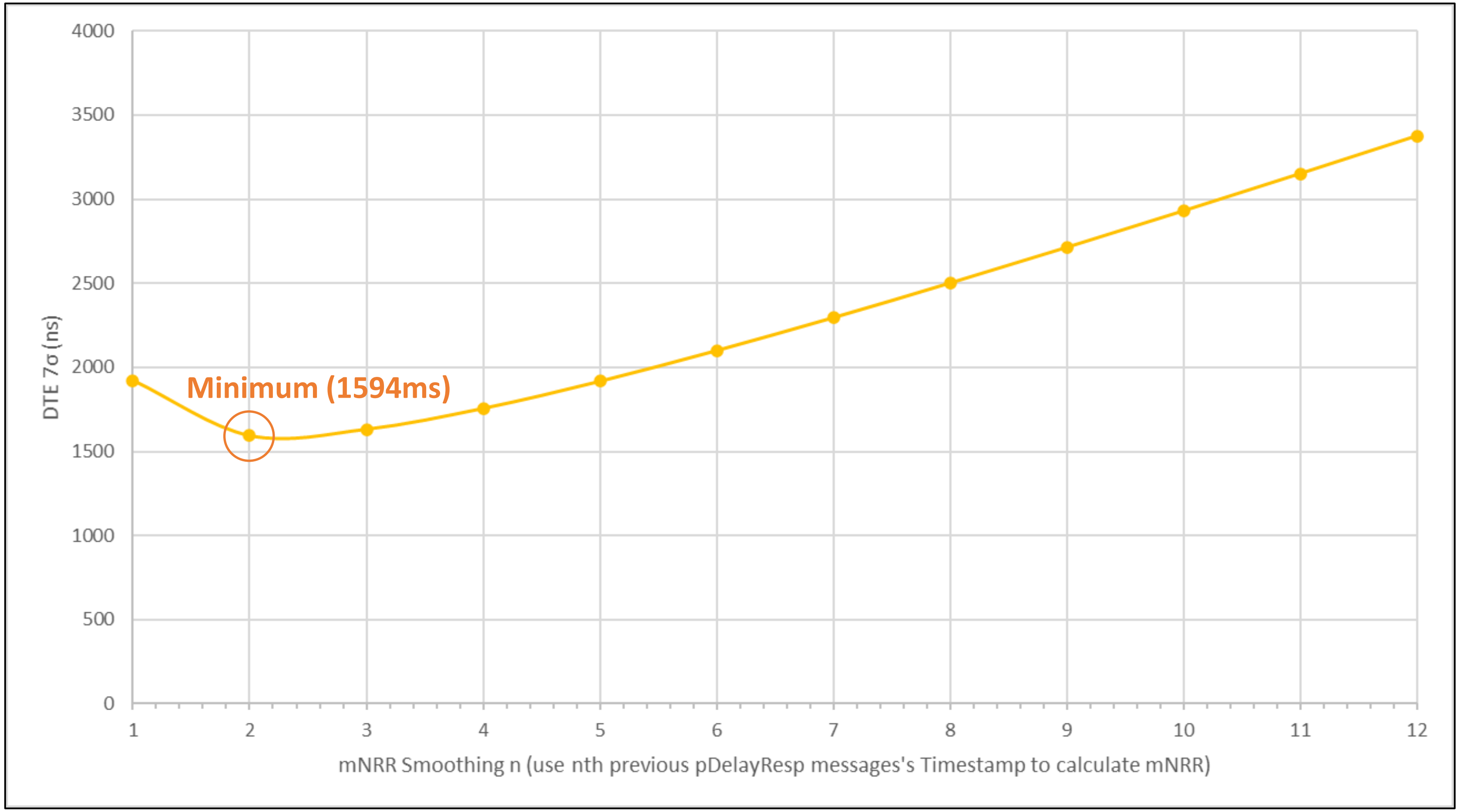
# H & I: Validate mNRR Smoothing – 1s pDelay Interval




| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | -0.6 | ppm/s |
| Clock Drift (non-GM) | 0.6 | ±ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | **1000** | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing | Variable | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

**mNRR smoothing doesn't help when $RT_{errorRR\_NRR}$ is already dominated by errors due to Clock Drift ($RT_{errorRR\_NRR\_CD}$)**

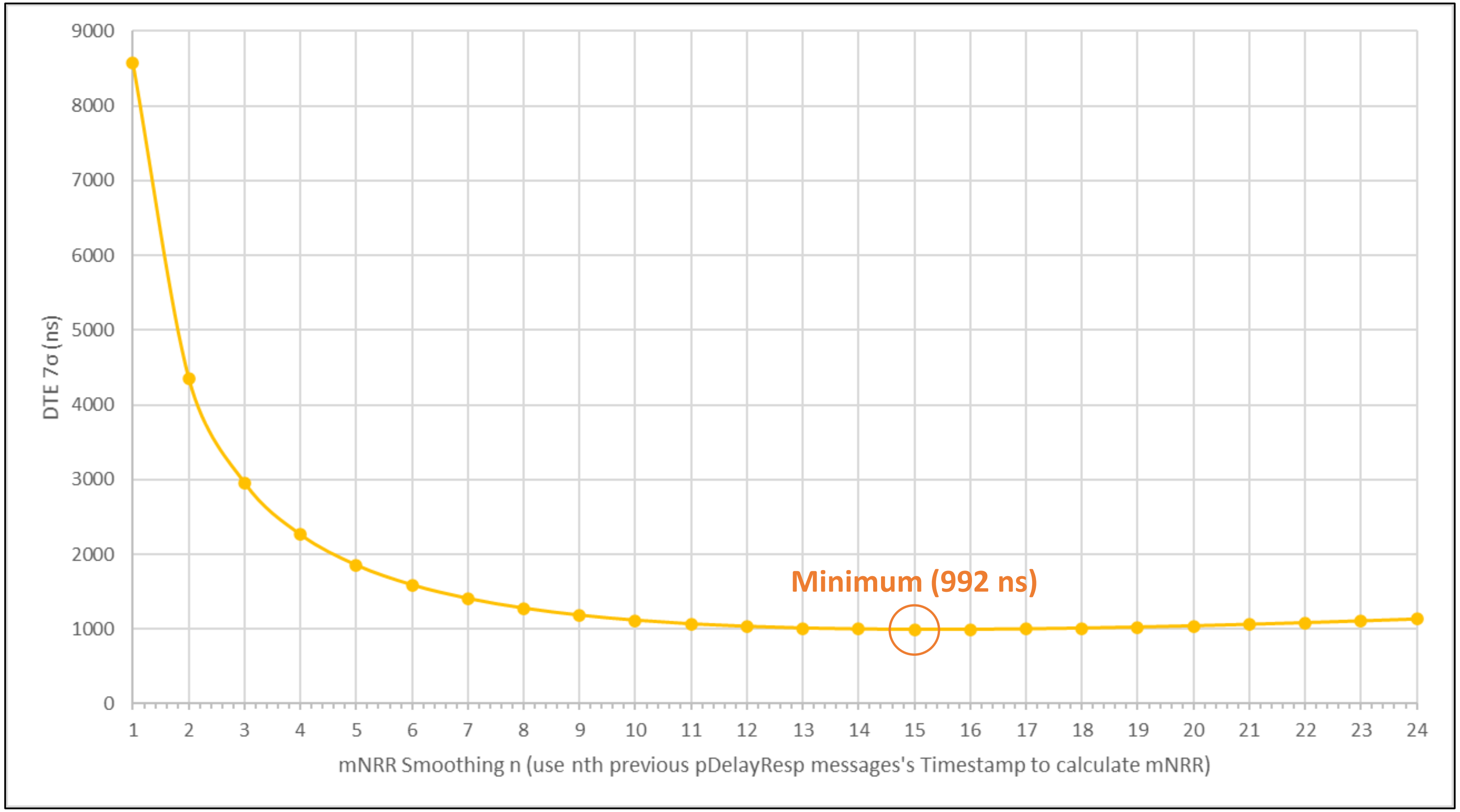
# J & K: Validate mNRR Smoothing – 250ms pDelay Interval




| | | |
|---|---|---|
| **Input Errors** | | |
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | -0.6 | ppm/s |
| Clock Drift (non-GM) | 0.6 | ±ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | **200** | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing | Variable | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

**mNRR smoothing may help a bit when *pDelayInterval* is optimised (for these parameters at least) for low values of n, but as n rises the errors due to clock drift soon outweigh the benefits of reduced Timestamp-related errors**

# L & M: Validate mNRR Smoothing – 31.25ms pDelay Interval




| Input Errors | | |
|---|---|---|
| GM Clock Drift Max | +0.6 | ppm/s |
| GM Clock Drift Min | -0.6 | ppm/s |
| Clock Drift (non-GM) | 0.6 | ±ppm/s |
| Timestamp Granularity TX | 4 | ±ns |
| Timestamp Granularity RX | 4 | ±ns |
| Dynamic Time Stamp Error TX | 4 | ±ns |
| Dynamic Time Stamp Error RX | 4 | ±ns |
| **Input Parameters** | | |
| pDelay Interval | **31.25** | ms |
| pDelay Response Time | 10 | ms |
| residenceTime | 10 | ms |
| **Input Correction Factors** | | |
| Mean Link Delay | 0 | % |
| Drift Rate | 0 | % |
| pDelayResponse → Sync | 0 | % |
| mNRR Smoothing | Variable | |
| **Configuration** | | |
| Hops | 100 | |
| Runs | 100,000 | |

**mNRR Smoothing helps a lot when *pDelayInterval* is short and Timestamp-related errors dominate $RT_{errorRR\_NRR}$. The short pDelayInterval limits $RT_{errorRR\_CD}$ (which would rise if pDelay were increased).**

# Thank you!

# Back Up Material

# Clock Drift Sensitivity