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.

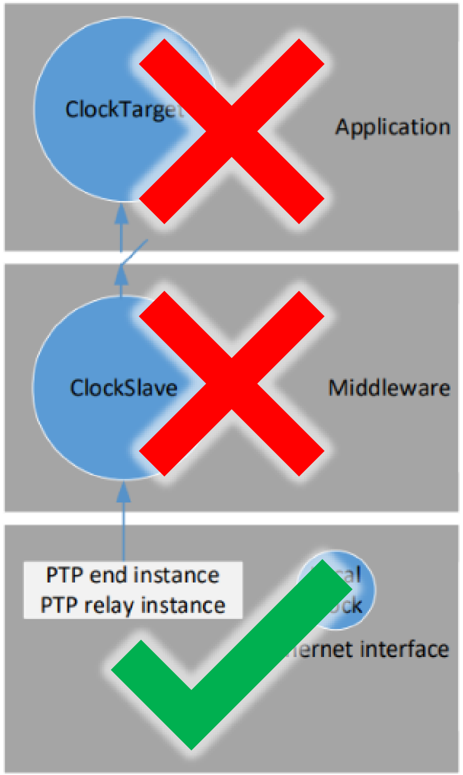
Background & Recap – 3

- 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

v2 Addition

Clarification: Scope of Monte Carlo Analysis

- The analysis models errors and how they arise and interact, not the underlying entities that experience or generate the errors.
  - There is only Clock Drift...there are no Clocks
  - There is only Timestamp Error...there are no Timestamps
  - There is only Dynamic Time Error...there is no modelling of Time
- The analysis only covers errors associated with pDelay and Sync messaging...which are based on the Local Clock.
  - It models each node, including the GM, as a single clock...or rather, errors associated with a single clock.
  - No Global Time. No Working Clock. No ClockTarget or ClockSlave (only the lowest box from Guenter's presentation*)
    - Additional modelling may be required for errors associated with these elements. But, if the basic mechanism can't achieve the goal, these elements aren't going to improve the situation.



* https://www.ieee802.org/1/files/public/docs2021/60802-Steindl-ClockTarget-and-ClockSource-1121-v05.pdf

v2 Addition

Clarification: Scope of Monte Carlo Analysis

- The RStudio script combines ppm errors via addition...which introduces an error in the error...but the error in the error is swamped by other errors.
  - Errors in ppm are ratios and, to be accurate, should be multiplied.
  - But...if the ppm errors are small (one or two digits)...the inaccuracy from addition isn’t significant.
    - 20 ppm + 30 ppm = 50 ppm
    - 20 ppm x 30ppm = 50.0006 ppm
- The trade-off is worth it for reduced runtime.
  - Multiplication is more expensive, computationally, than addition...especially when using double precision floating point numbers.

v2 Addition

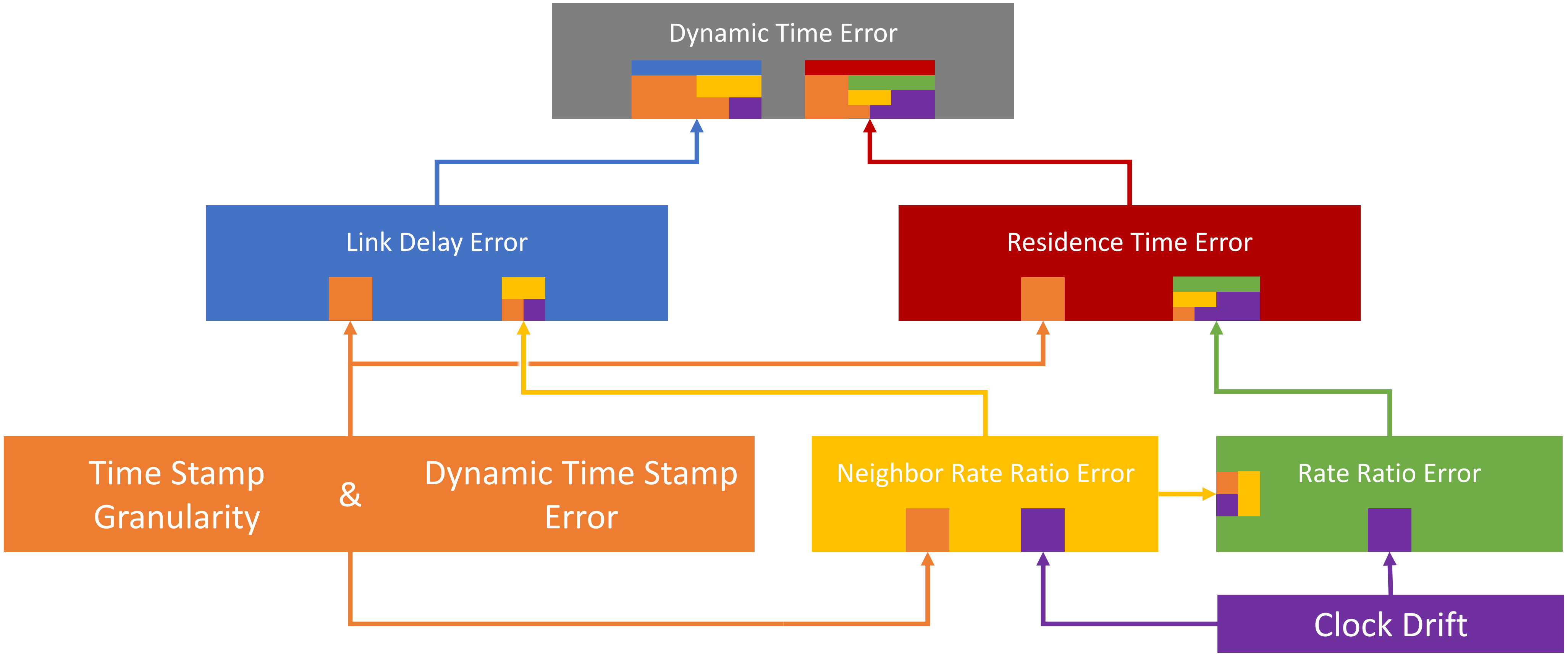
Planned Improvement

- Analysis in this and previous presentations modelled DTE at the point the Sync message arrived at the End Station (hop 100).
- There is additional error as the GM and Local Clock drift with respect to each other prior to arrival of the next Sync message.
  - Modelled as follows (when hop = hops, i.e. for the final hop only)...

$$DTE_{endStationError} = \frac{syncInterval \times RRer_{ror}(hop)}{1000} \quad \textbf{ns}$$

  - Note: first time syncInterval has been used in this approach to error analysis
- This error will be added to future analyses (again: but not this one)
- Not expected to be significant relative to overall DTE

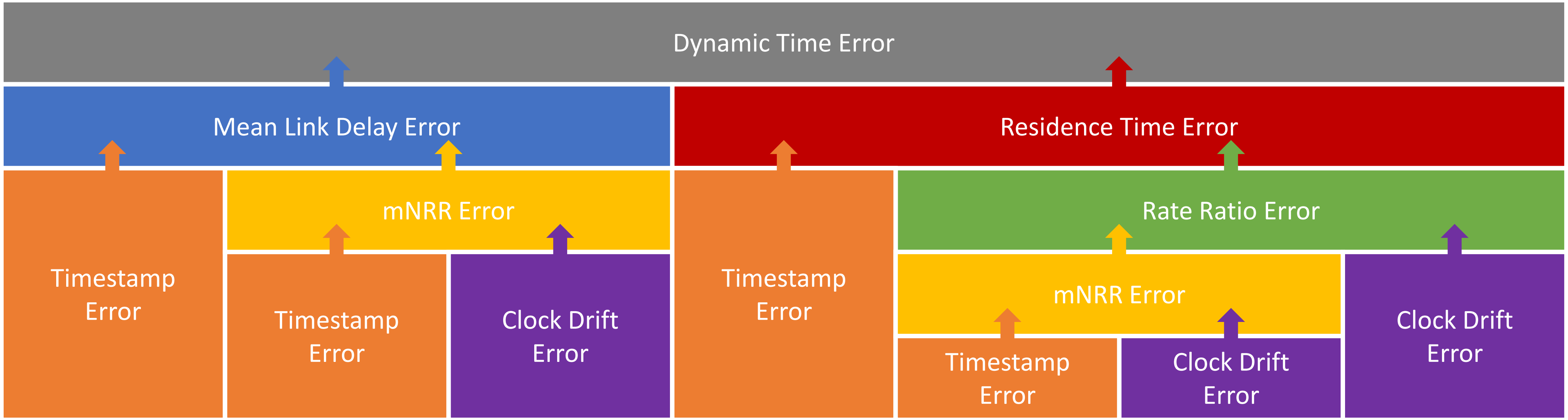
Time Sync – How Errors Add Up



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

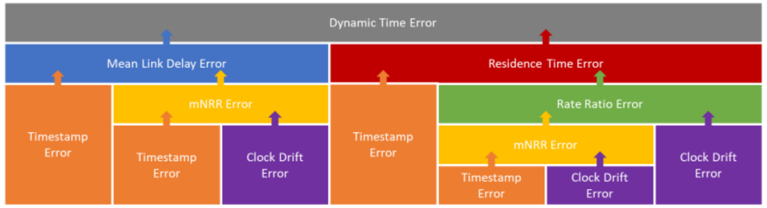
Graphical Representation of Error Accumulation

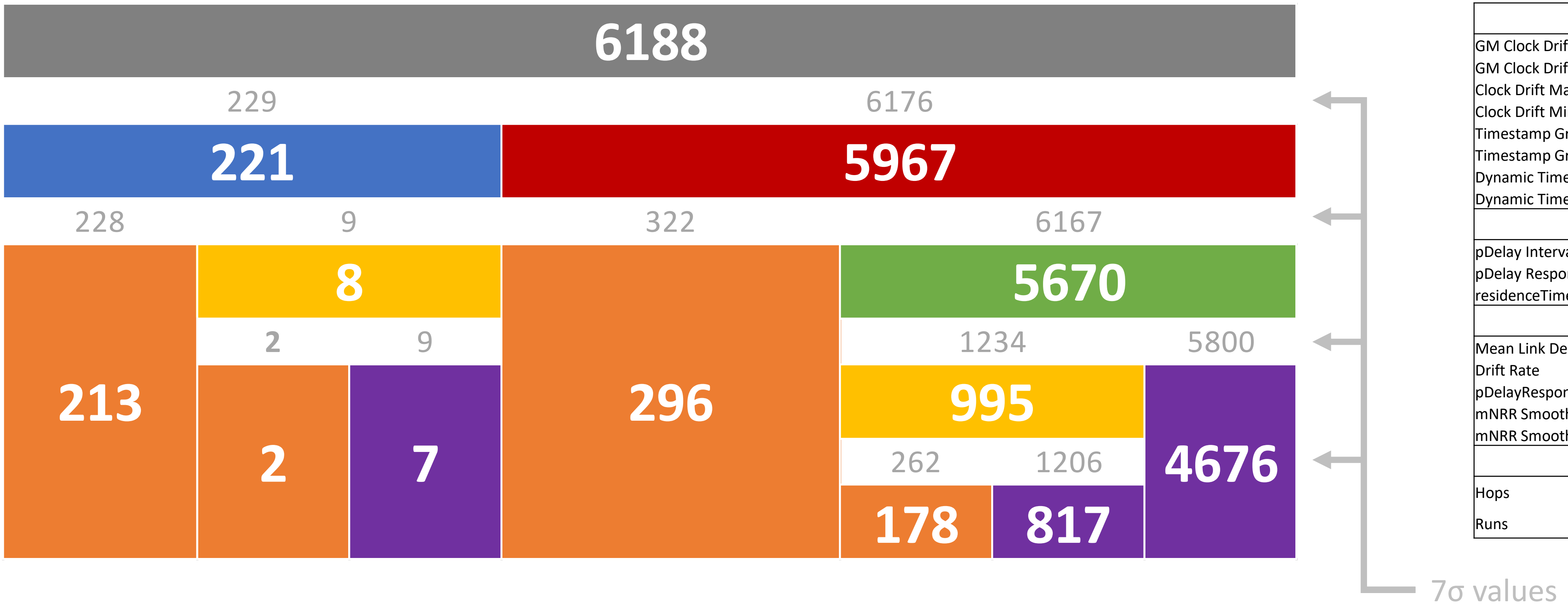
- Each error breaks down into two parts



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

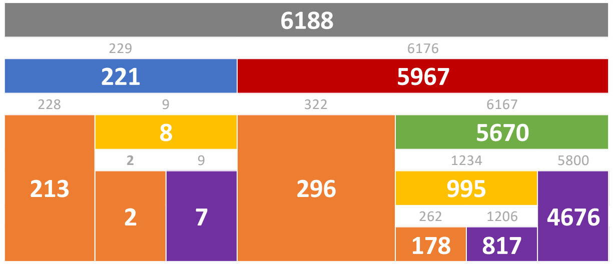
Graphical Representation of Error Accumulation

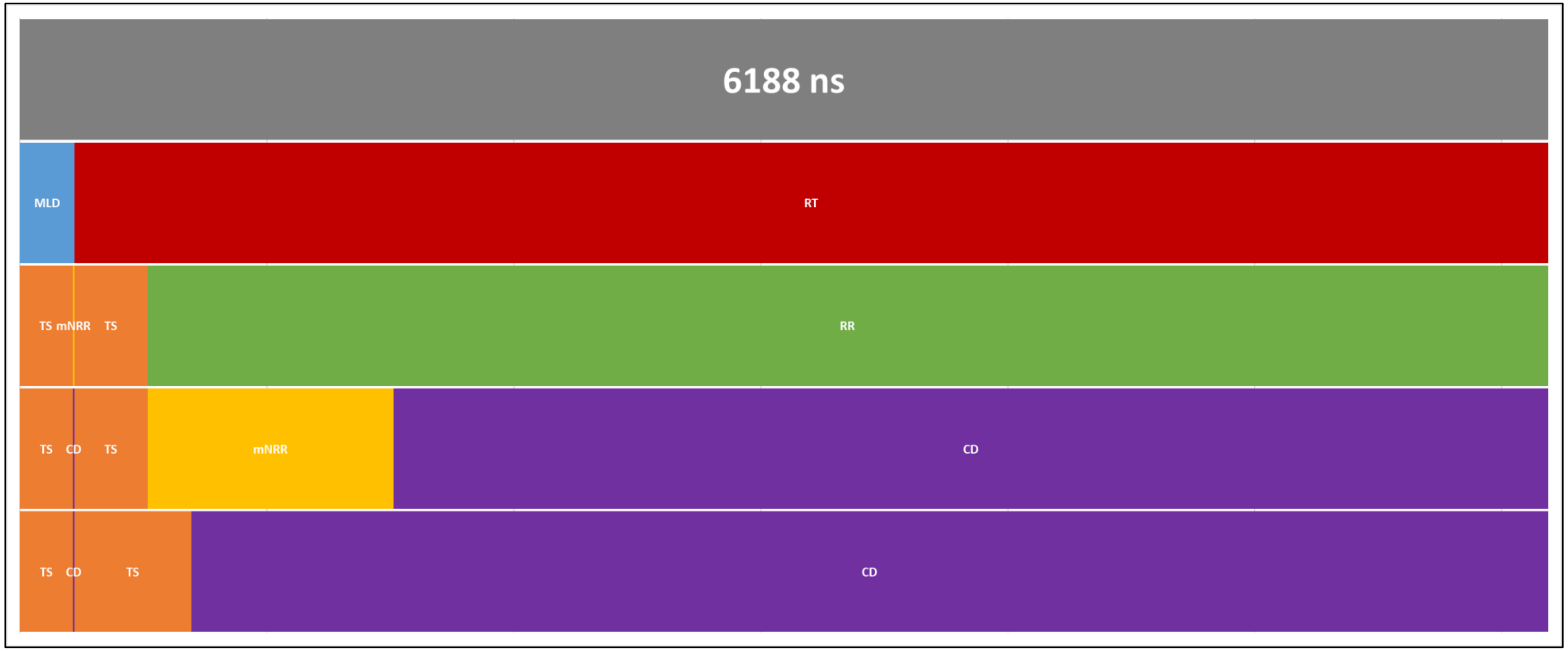




Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	-0.6	ppm/s
Clock Drift Max (non-GM)	+0.6	ppm/s
Clock Drift Min (non-GM)	-0.6	ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay 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	

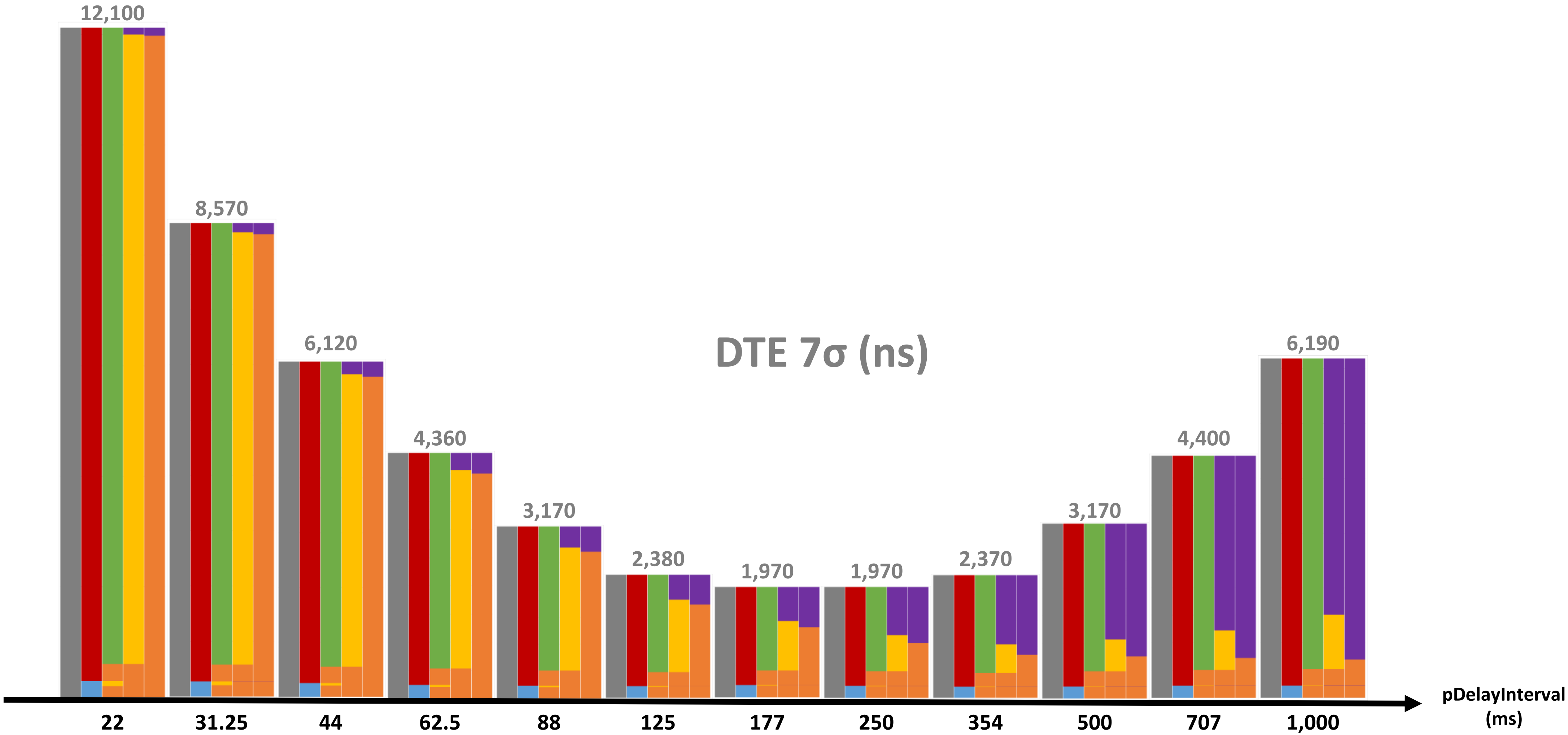
Graphical Representation of Error Accumulation



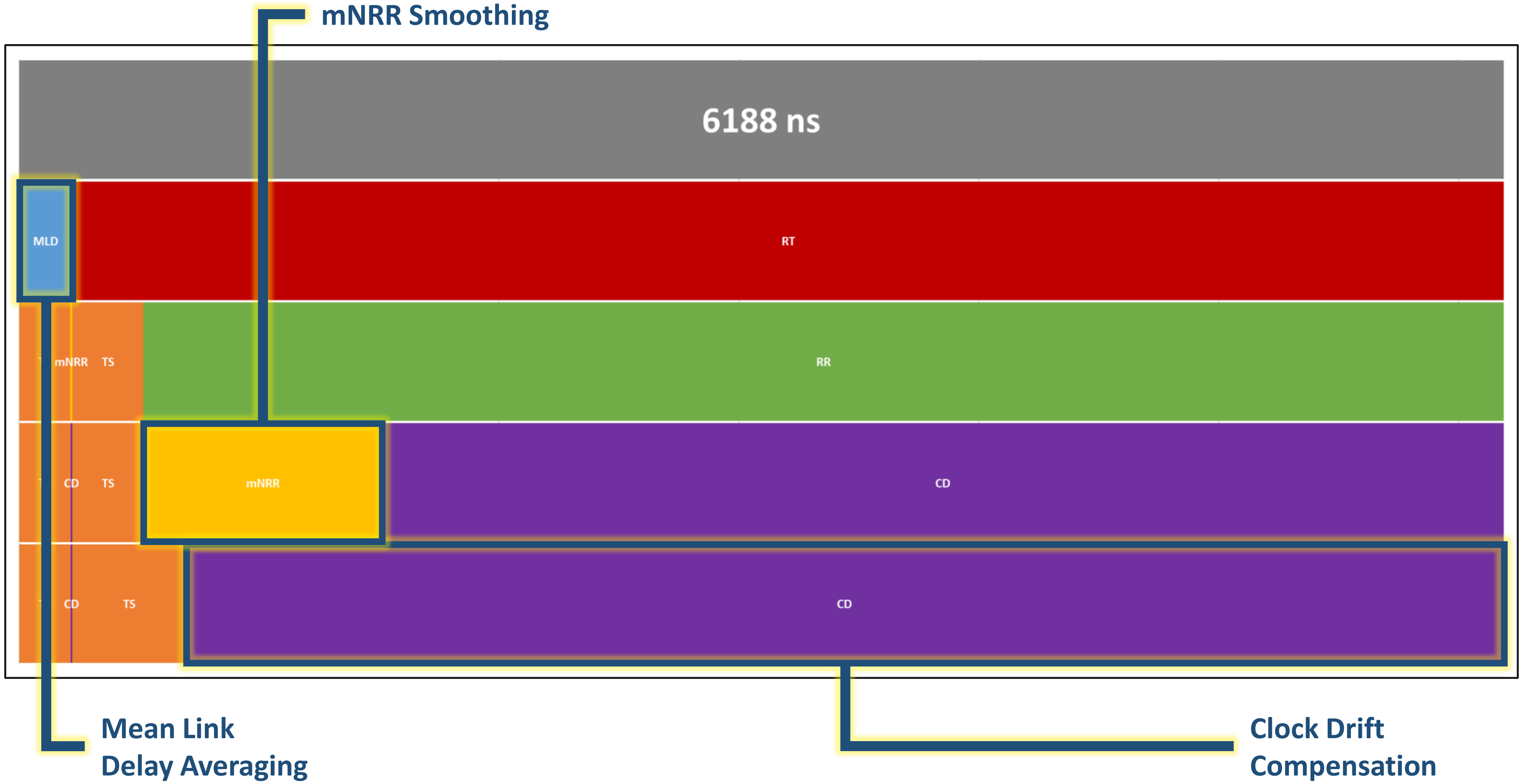


Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	-0.6	ppm/s
Clock Drift Max (non-GM)	+0.6	ppm/s
Clock Drift Min (non-GM)	-0.6	ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay 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



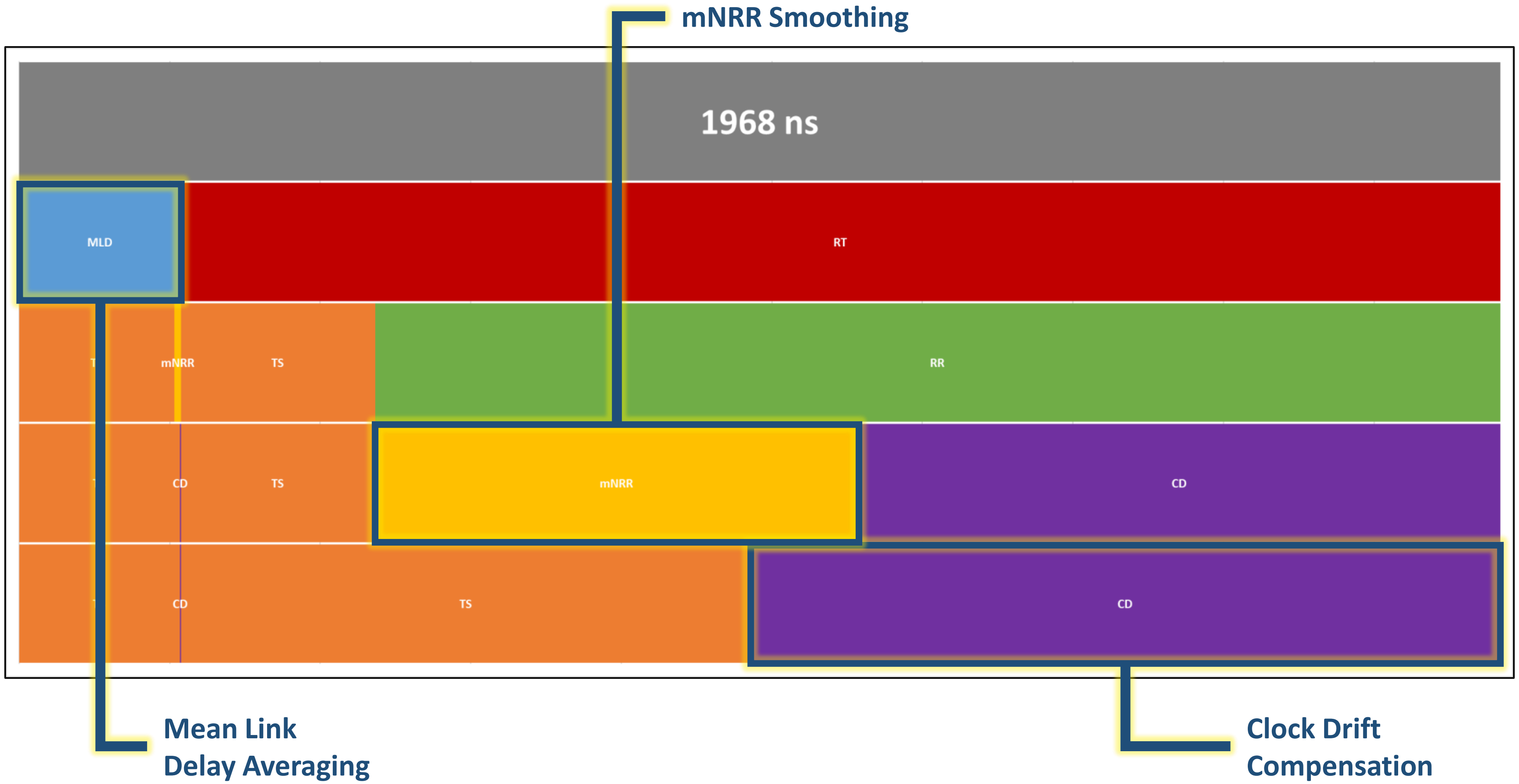
mNRR Smoothing

Mean Link Delay Averaging

Clock Drift Compensation

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



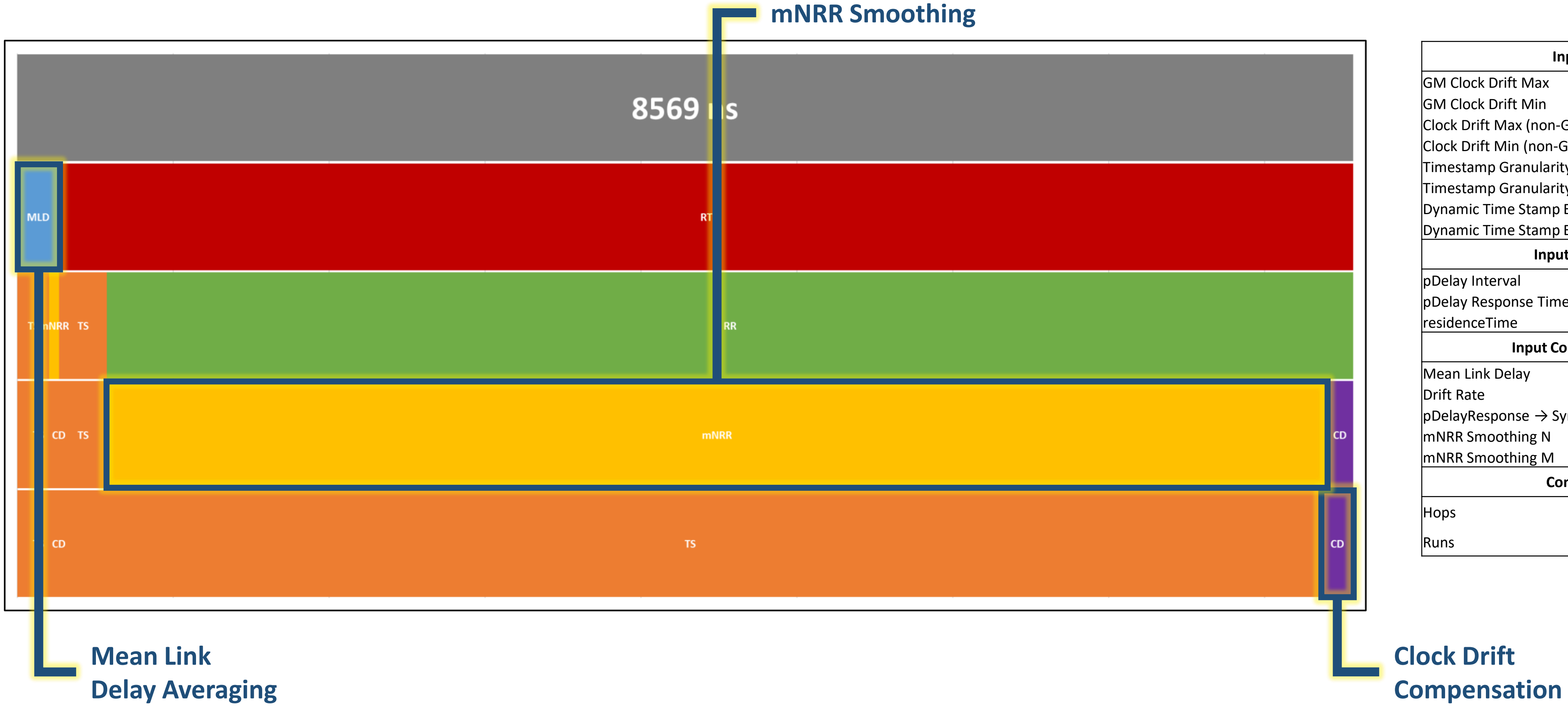
mNRR Smoothing

Mean Link Delay Averaging

Clock Drift Compensation

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



mNRR Smoothing

Mean Link Delay Averaging

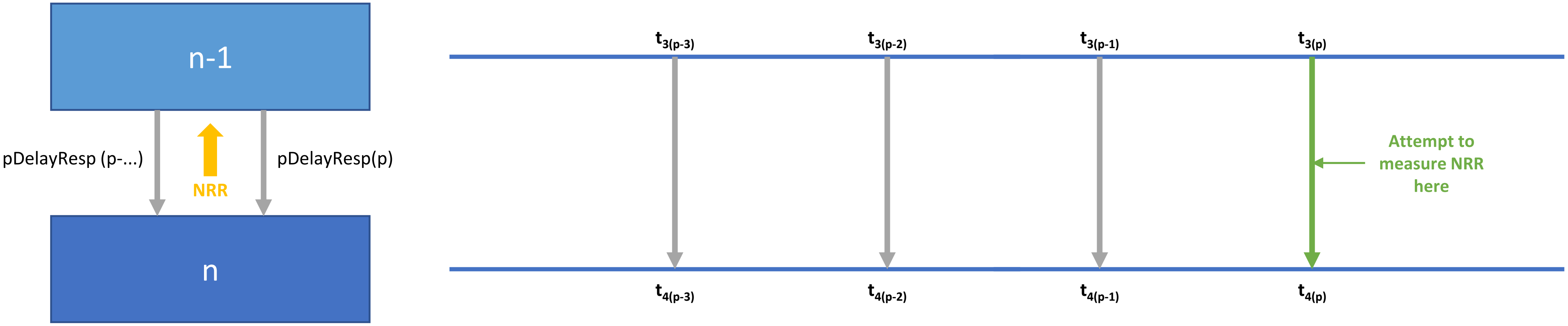
Clock Drift Compensation

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	

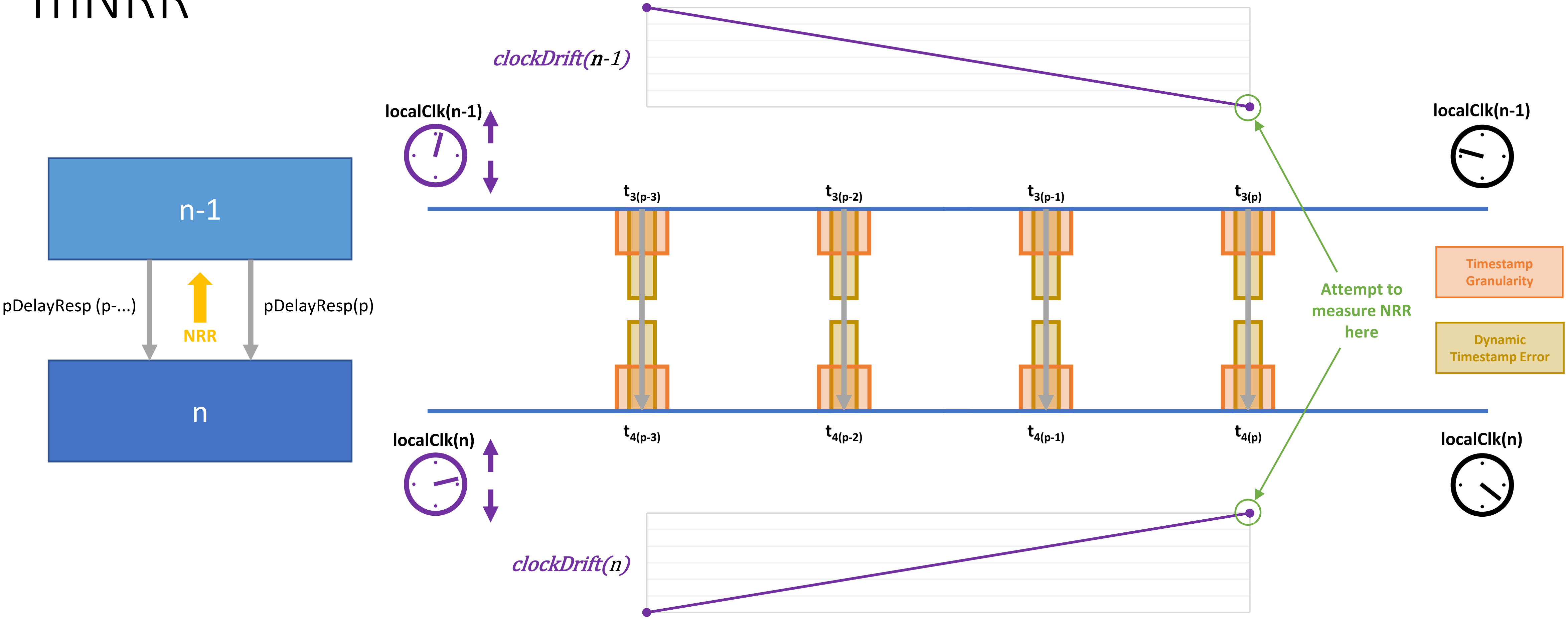
Neighbor Rate Ratio “Smoothing”

Techniques for minimising $\text{mNRR}_{\text{error}}$

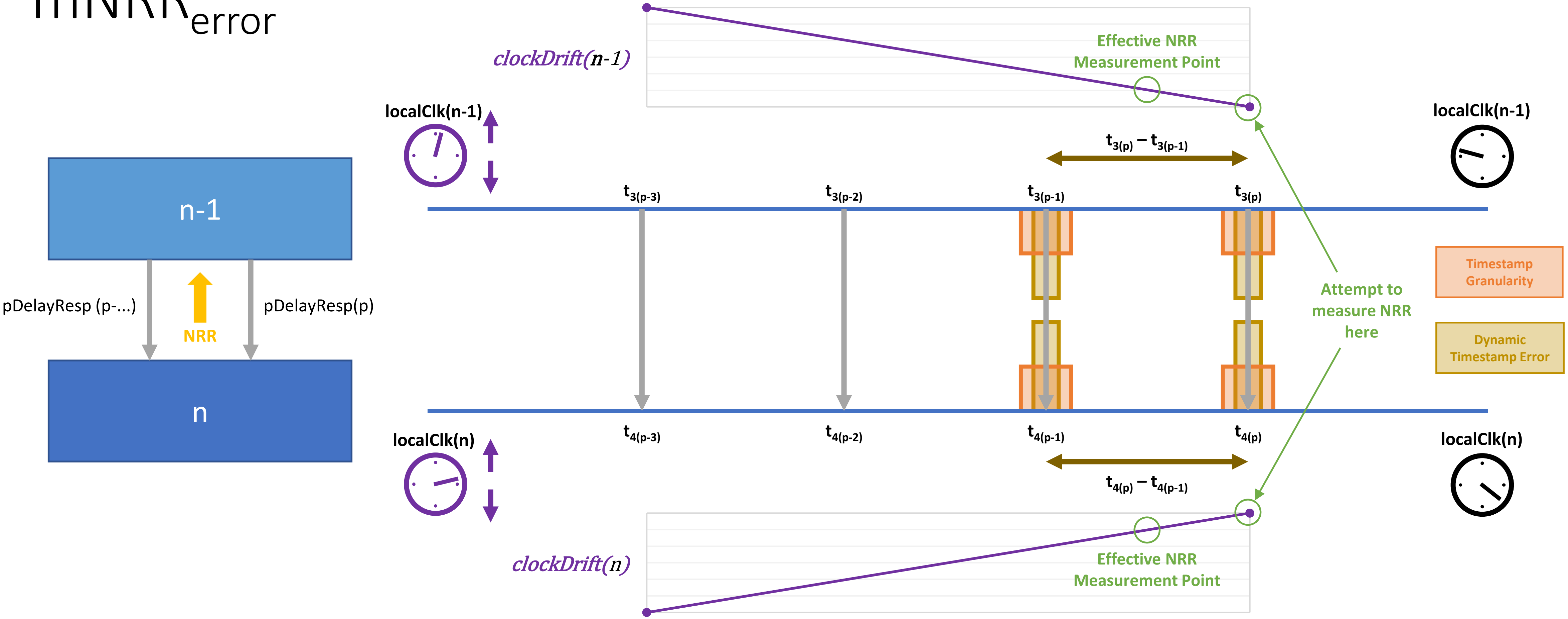
mNRR


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
$t_{3(p-3)}$
$t_{3(p-2)}$
$t_{3(p-1)}$
$t_{3(p)}$
$t_{4(p-3)}$
$t_{4(p-2)}$
$t_{4(p-1)}$
$t_{4(p)}$
Attempt to measure NRR here
Timestamp Granularity
Dynamic Timestamp Error
localClk(n)
localClk(n)
clockDrift(n)

mNRR$_{error}$



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

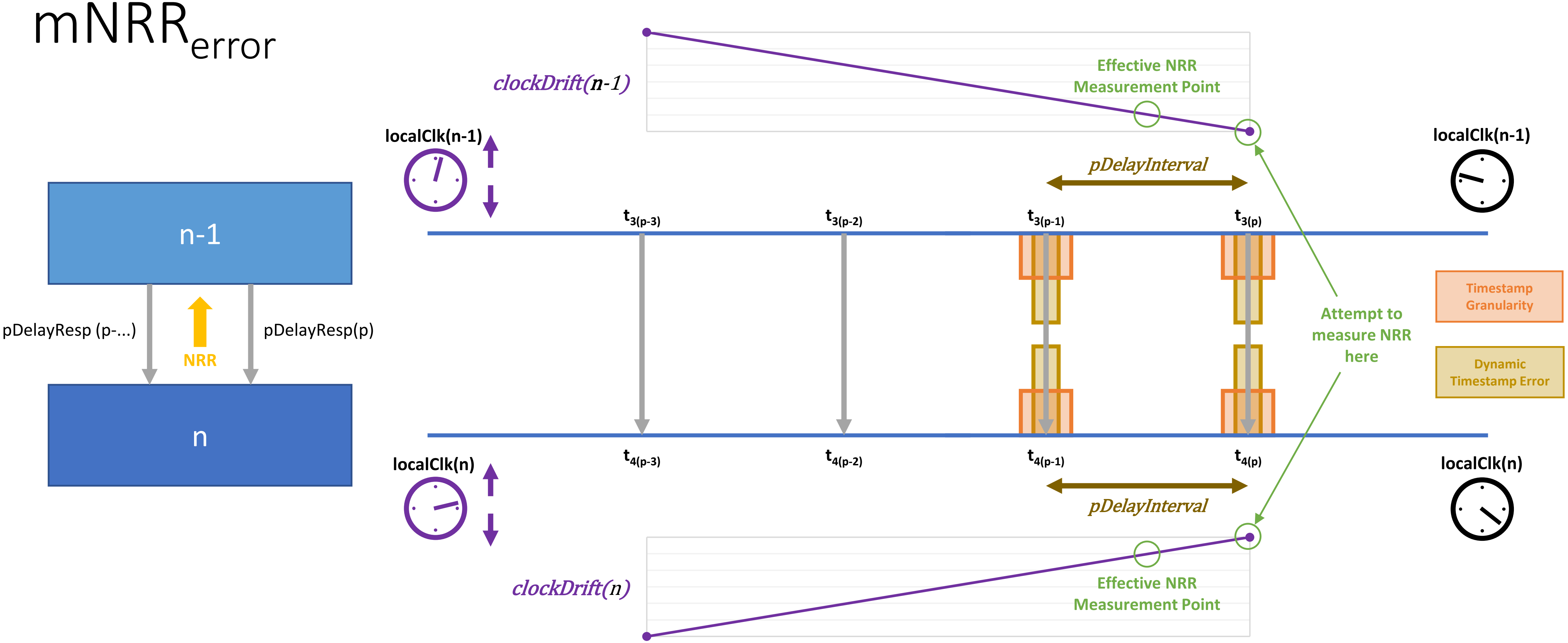
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{(t_{4(p)} - t_{4(p-1)}) \times \mathbf{10^6}}\right)$$

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$_{error}$

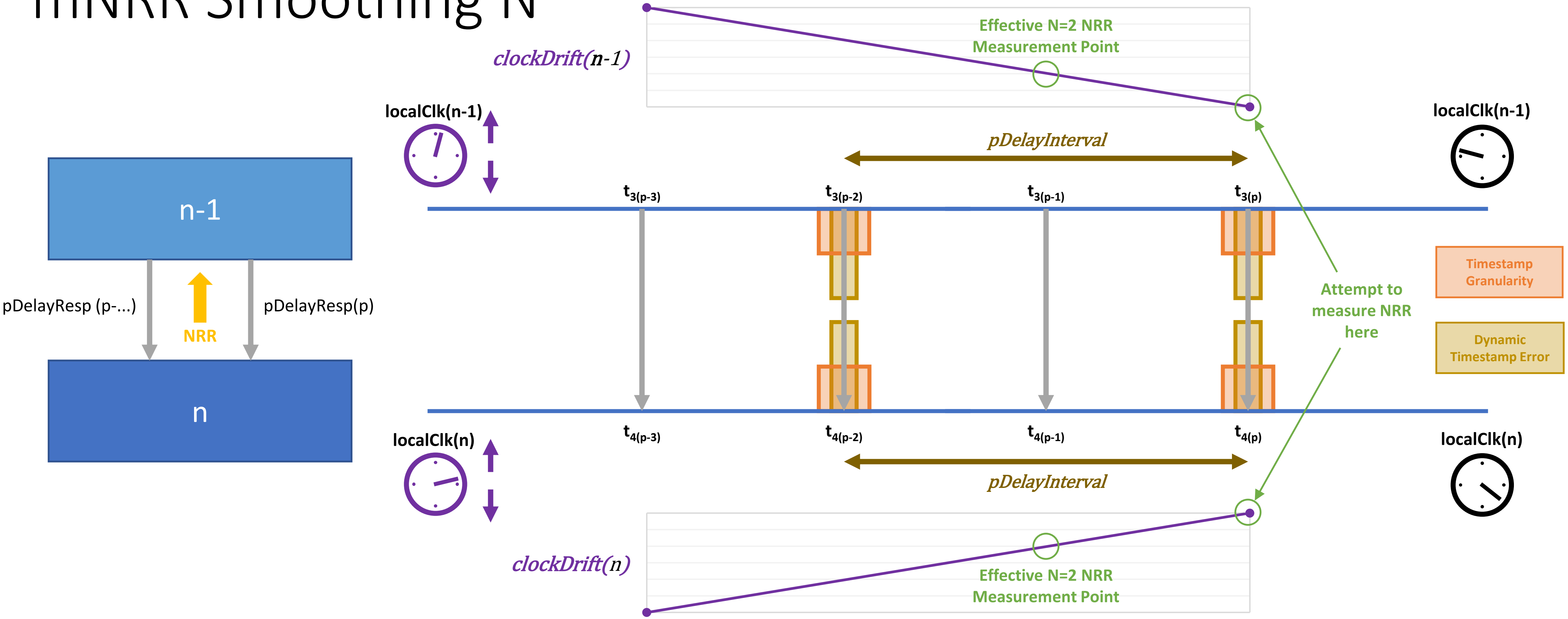


$$mNRR = \left(\frac{\left(t_{4(p)} - t_{4(p-1)}\right)}{\left(t_{3(p)} - t_{3(p-1)}\right)}\right)$$

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

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

mNRR Smoothing N



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

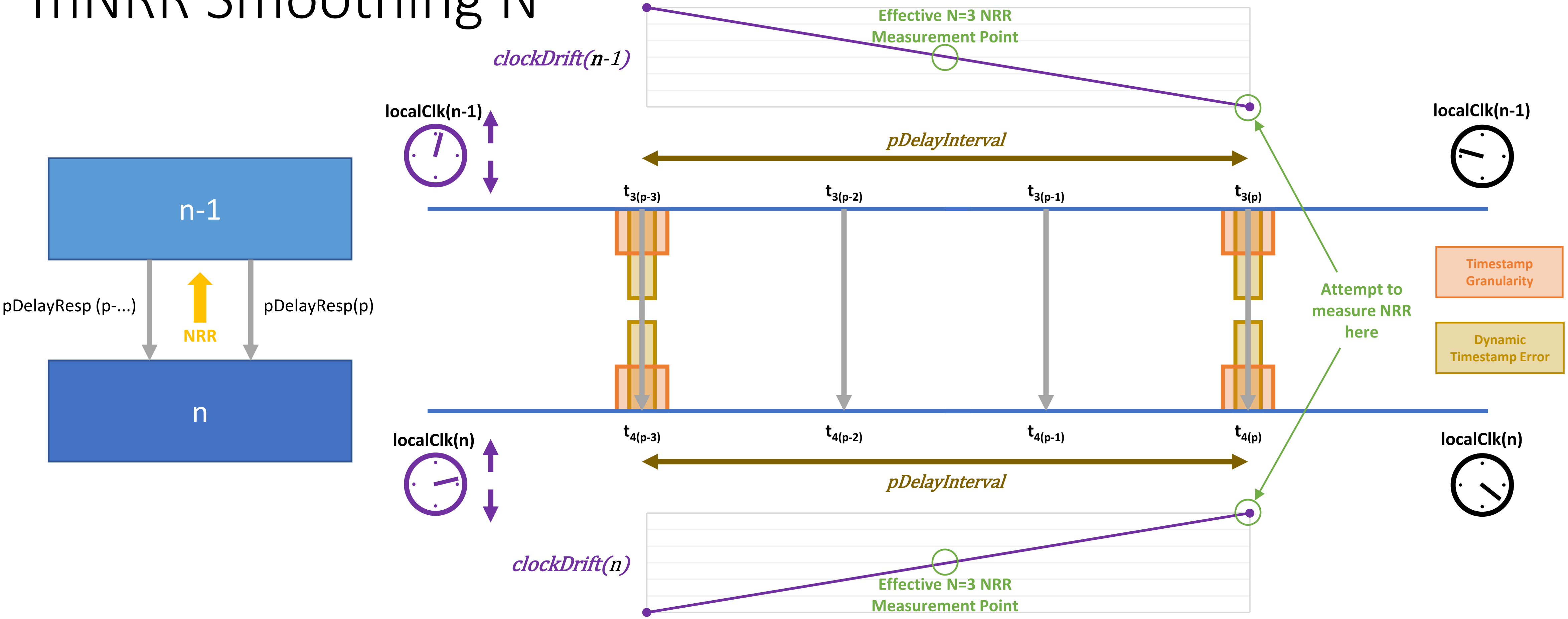
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval \times 2}\right)$$

ppm

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

ppm

mNRR Smoothing N

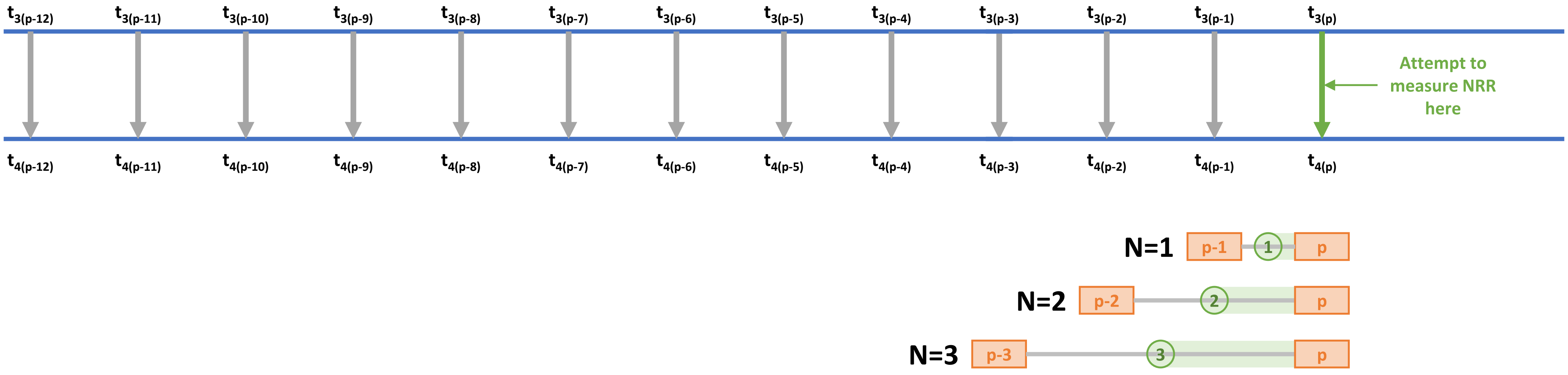


$$mNRR = \left(\frac{(t_{4(p)} - t_{4(p-1)})}{(t_{3(p)} - t_{3(p-1)})}\right)$$

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

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

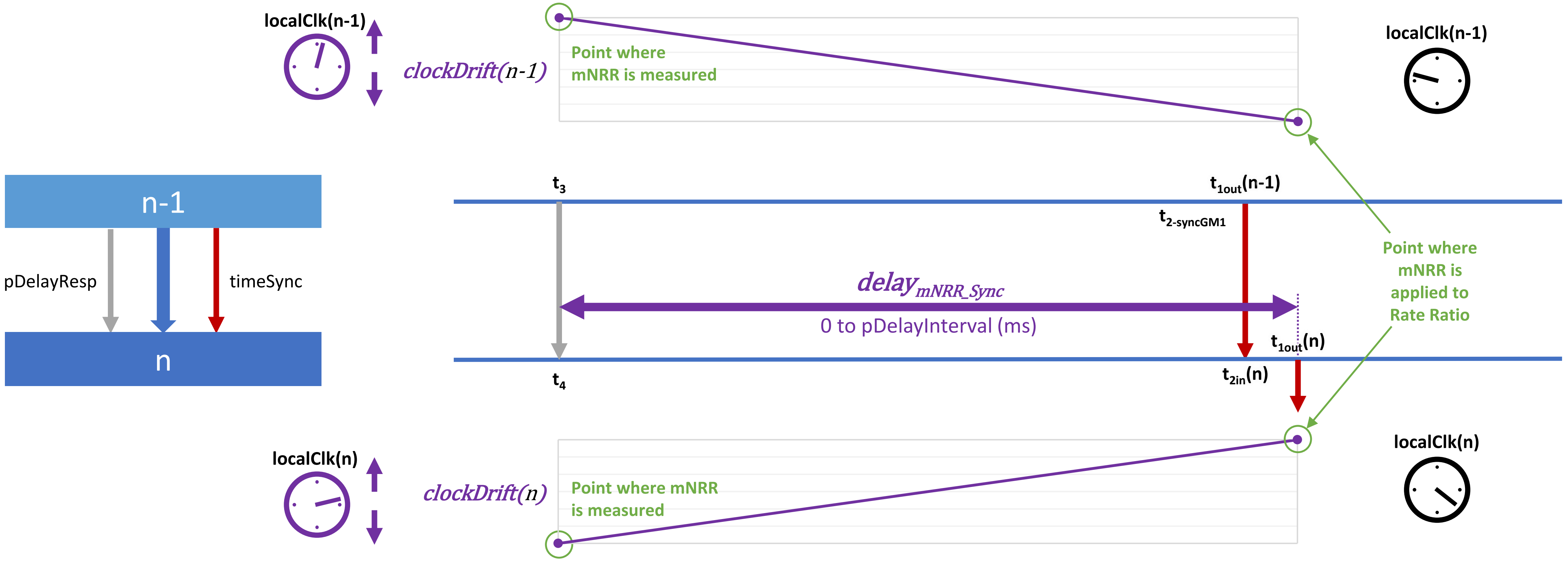
mNRR Smoothing N



mNRR Smoothing N

- Using the Nth 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

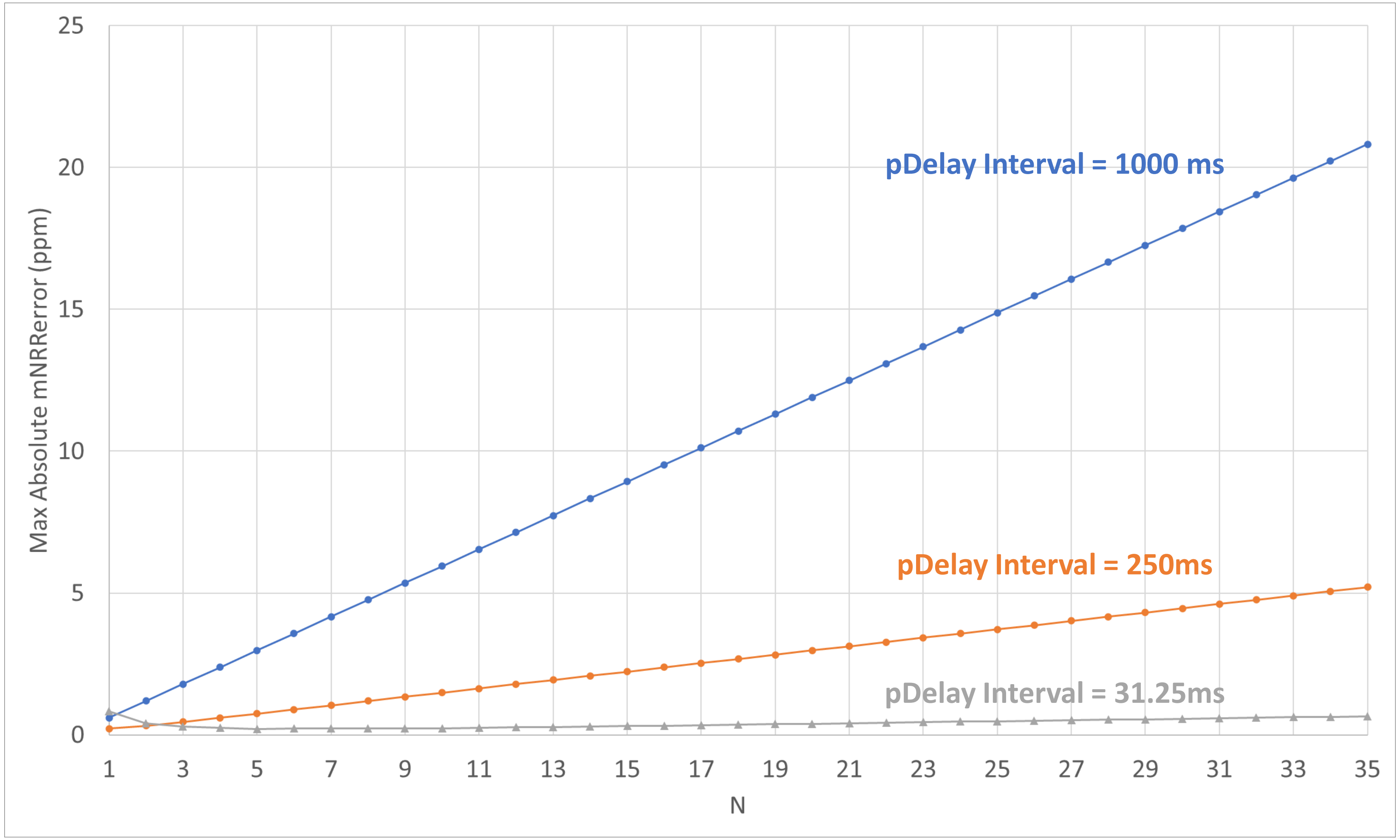
$RR_{errorCD}$



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

ppm

Optimal Value of N

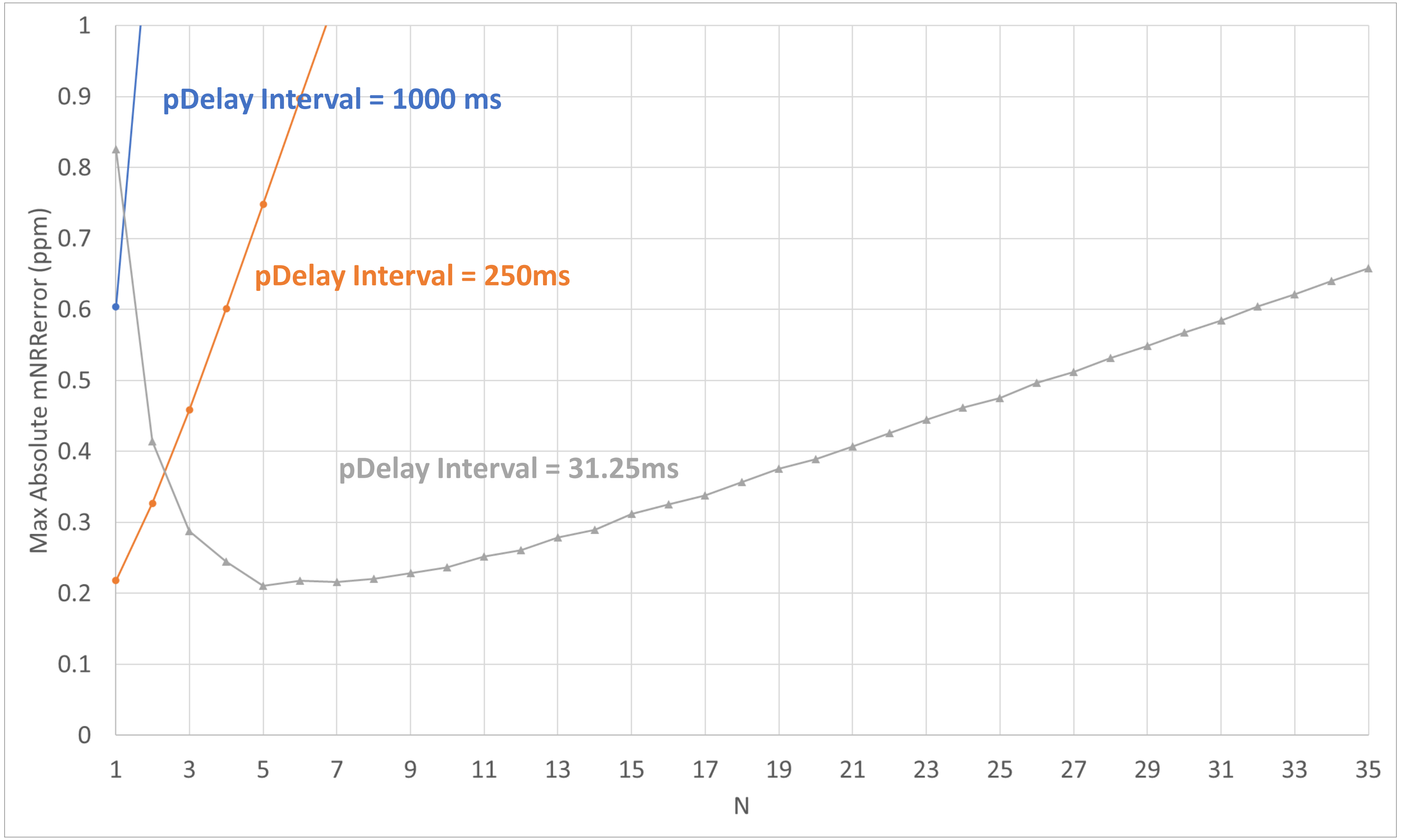


Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Input Correction Factors		
mNRR Smoothing M	1	
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

How short?...

Optimal Value of N



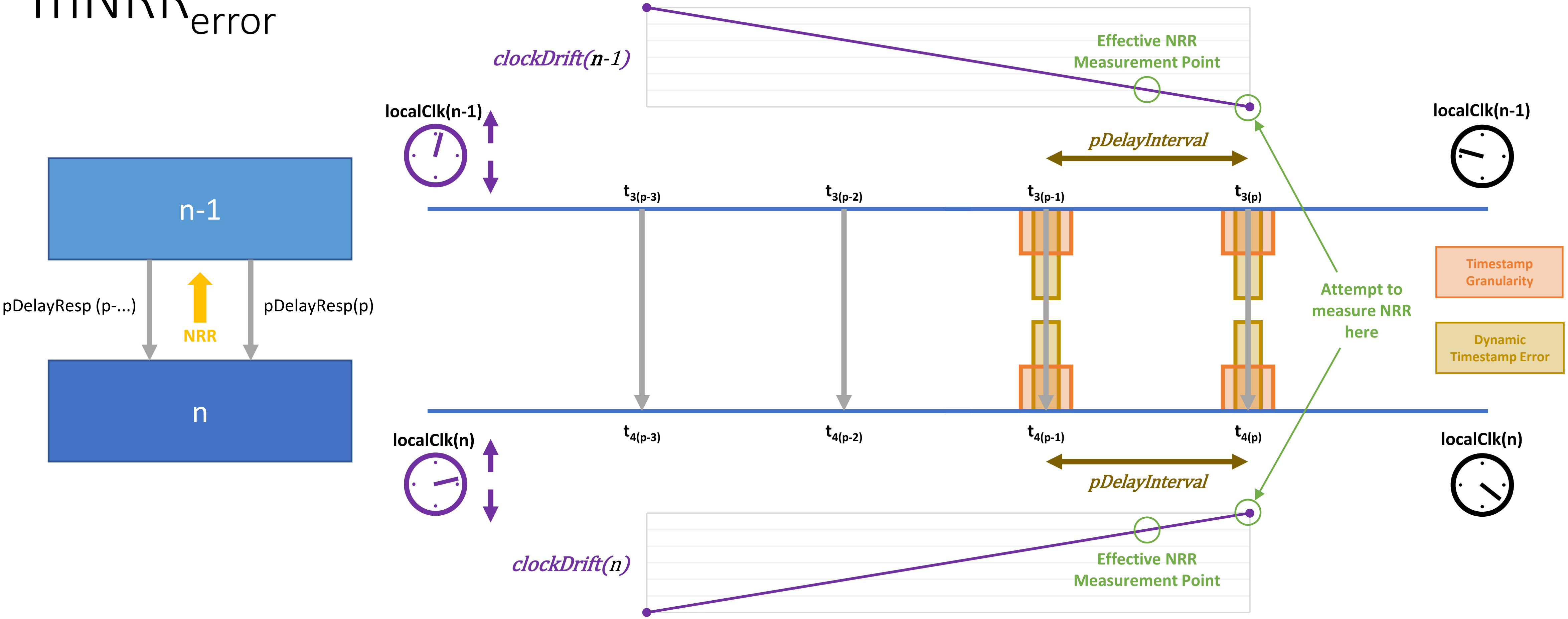
Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Input Correction Factors		
mNRR Smoothing M	1	
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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$_{error}$

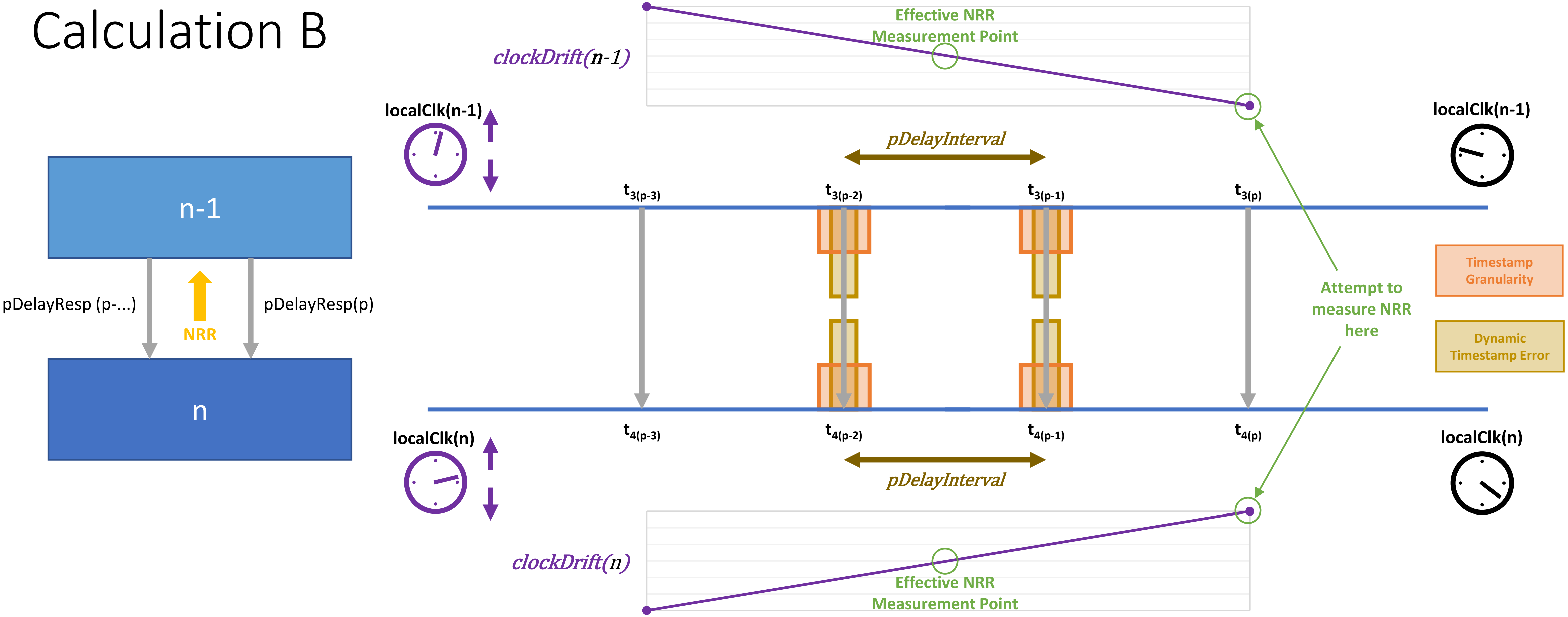


$$mNRR = \left(\frac{\left(t_{4(p)} - t_{4(p-1)}\right)}{\left(t_{3(p)} - t_{3(p-1)}\right)}\right)$$

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

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

mNRR Smoothing M Calculation B



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

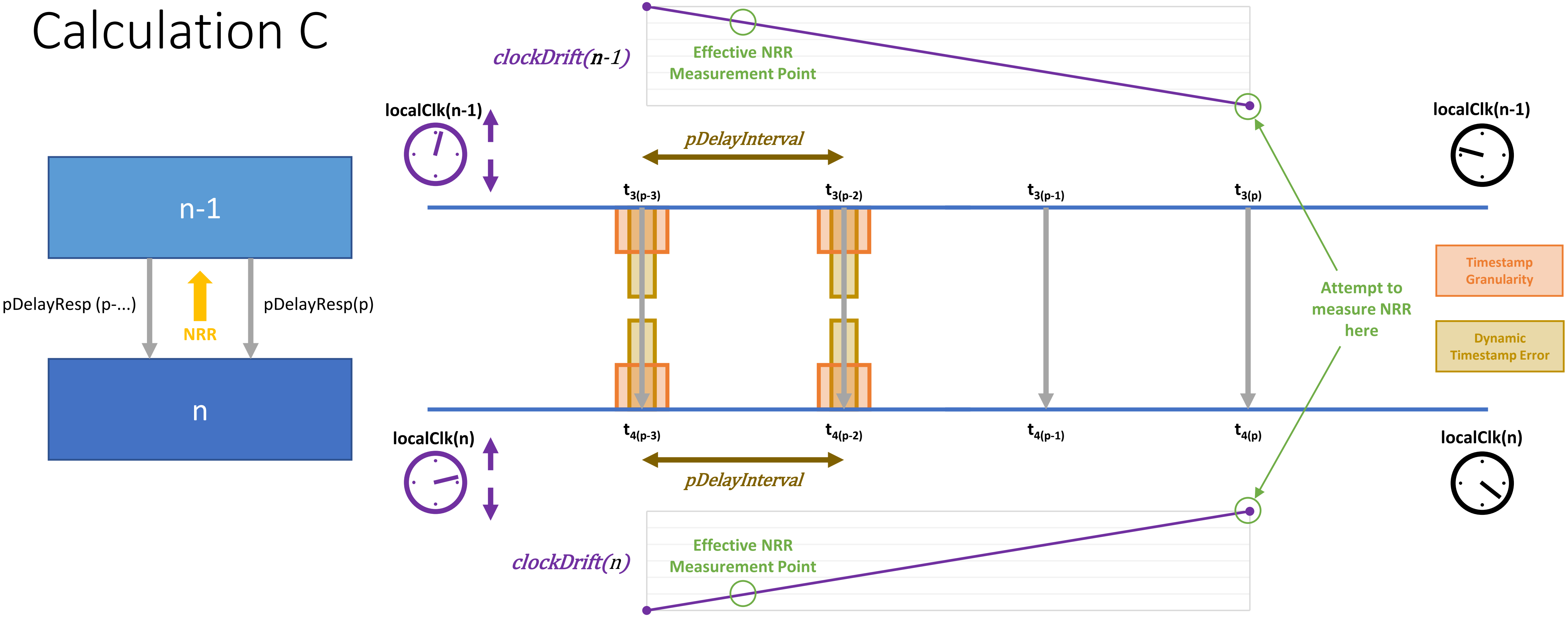
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror_{(p-1)}} - t_{4PDerror_{(p-2)}}) - (t_{3PDerror_{(p-1)}} - t_{3PDerror_{(p-2)}})}{pDelayInterval}\right)$$

ppm

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

ppm

mNRR Smoothing M
Calculation C



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

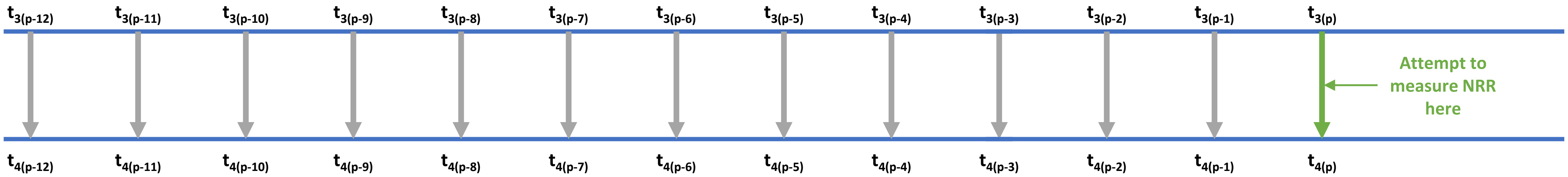
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p-2)} - t_{4PDerror(p-3)}) - (t_{3PDerror(p-2)} - t_{3PDerror(p-3)})}{pDelayInterval}\right)$$ **ppm**

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

mNRR Smoothing M

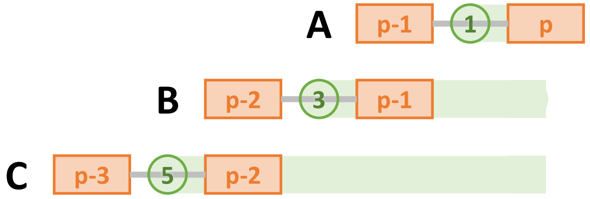
- Using the median of M previous NRR calculations...
    - Selects the NRR calucation with the median total error ($mNRR_{error}$)
        - Addition of $mNRR_{errorTS}$ from Timestamp Errors & $mNRR_{errorCD}$ from Clock Drift
- Implementing this in the Monte Carlo Analysis while keeping track of the 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...

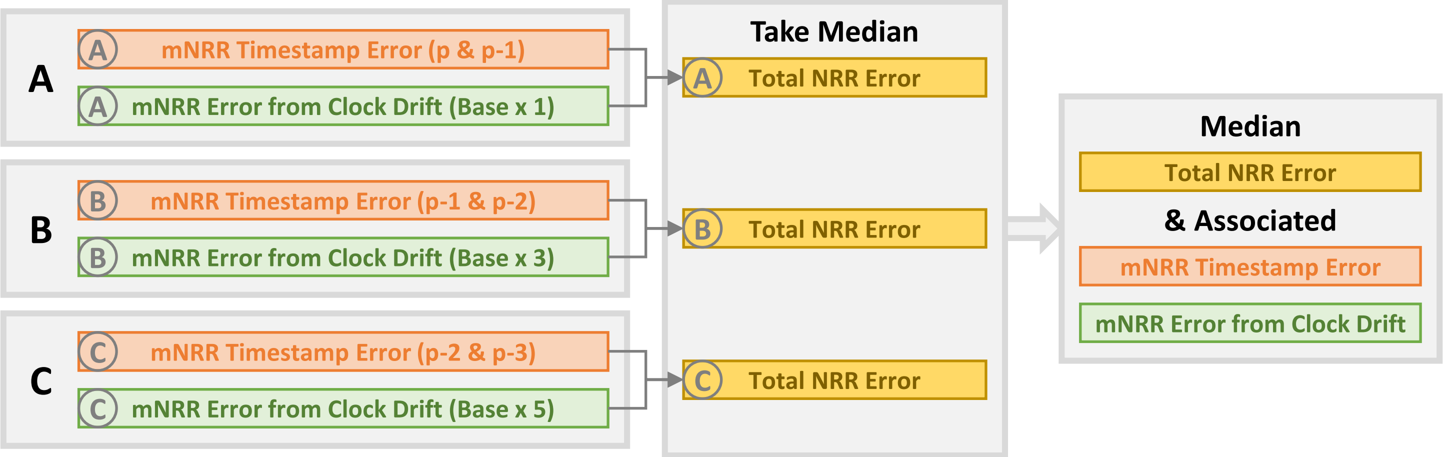
mNRR Smoothing M – Calculations



N = 1

M = 3





A
- (A) mNRR Timestamp Error (p & p-1)
- (A) mNRR Error from Clock Drift (Base x 1)

B
- (B) mNRR Timestamp Error (p-1 & p-2)
- (B) mNRR Error from Clock Drift (Base x 3)

C
- (C) mNRR Timestamp Error (p-2 & p-3)
- (C) mNRR Error from Clock Drift (Base x 5)

Take Median
- (A) Total NRR Error
- (B) Total NRR Error
- (C) Total NRR Error

Median
- Total NRR Error

& Associated
- mNRR Timestamp Error
- mNRR Error from Clock Drift

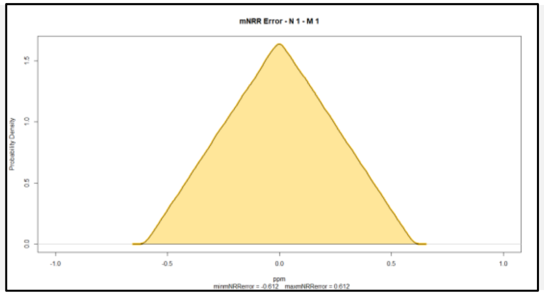
It depends on pDelay Interval! (And the values of M & N)

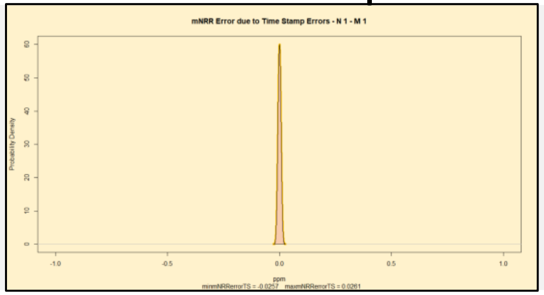
Timestamp Error vs Error due to Clock Drift

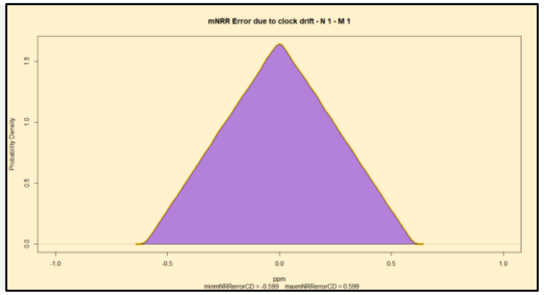
(mNRR – Single Hop – No Smoothing, i.e. N=1, M=1)

pDelay Interval | **$\text{mNRR}_{\text{error}}$** | **...due to Timestamp Error** | **...due to Clock Drift**

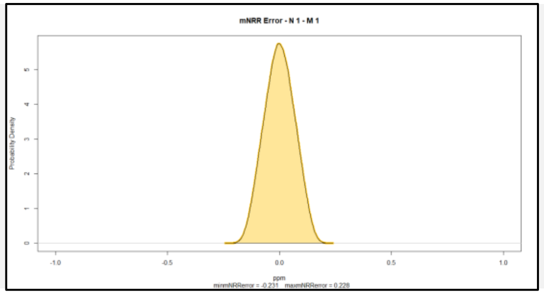
1000ms

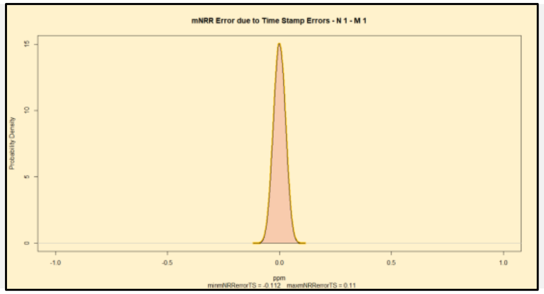


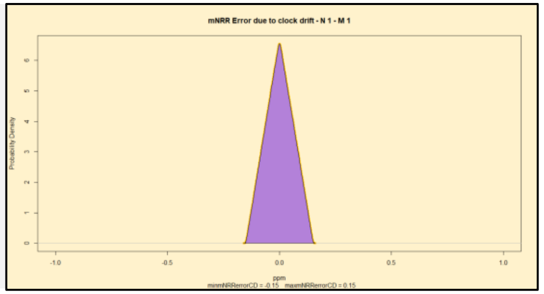




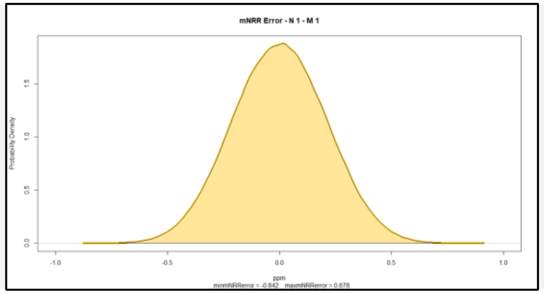
250ms

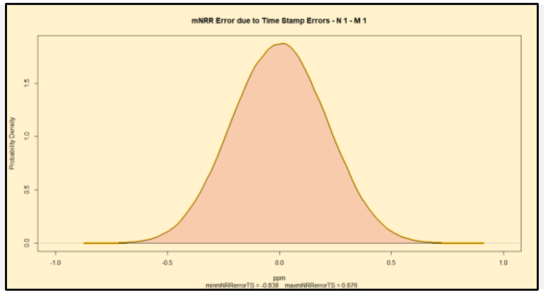


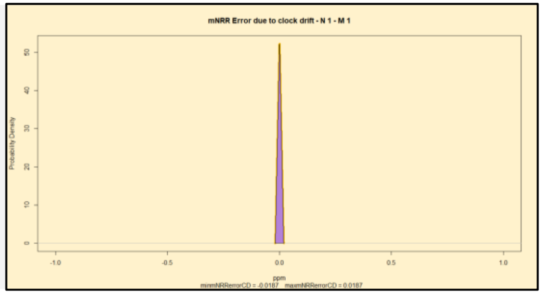




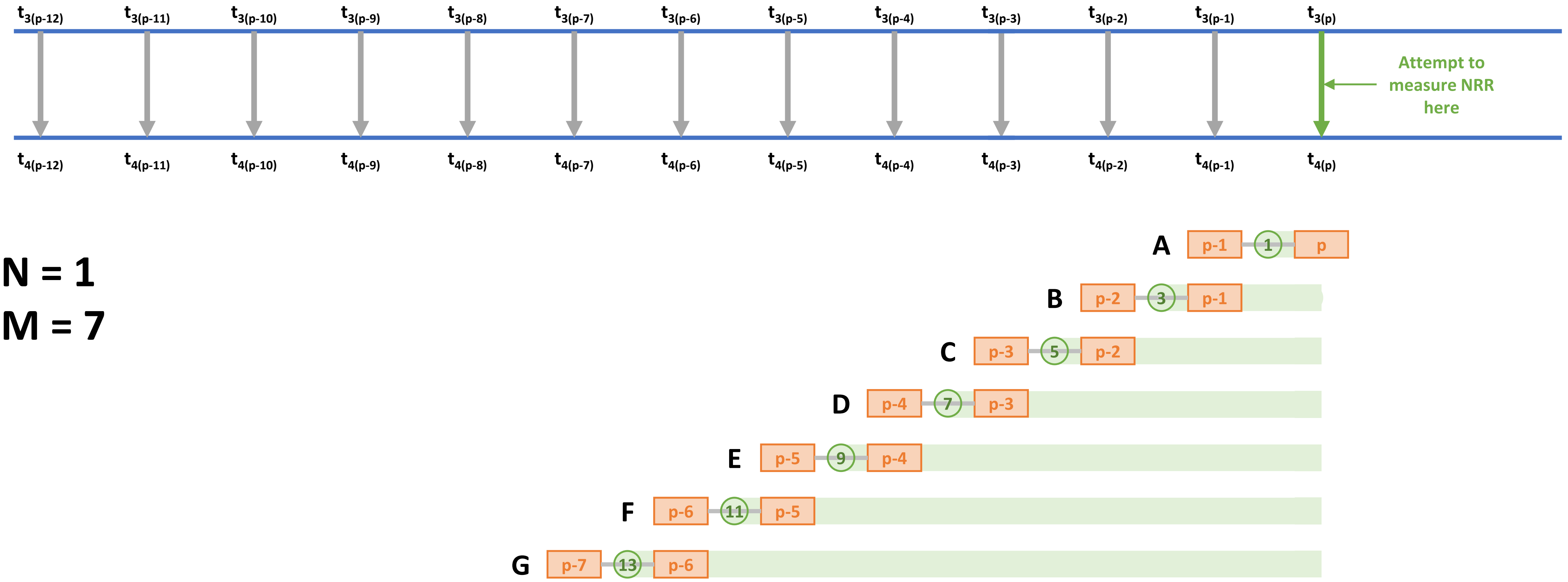
31.25ms







mNRR Smoothing M & pDelay Interval



N = 1

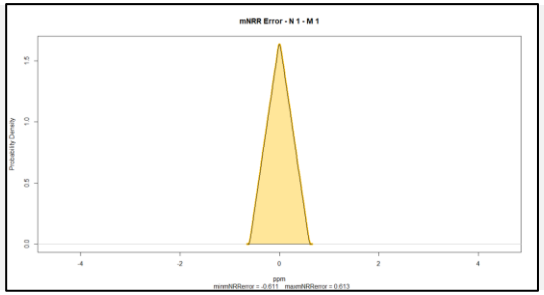
M = 7

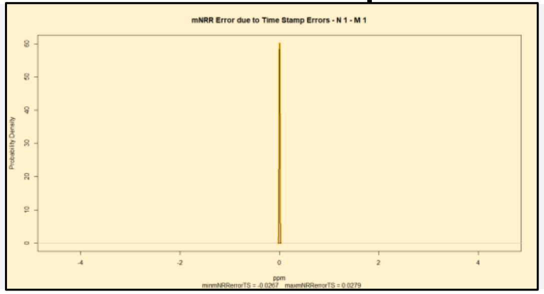
Effect of Taking Median of mNRR Calculations

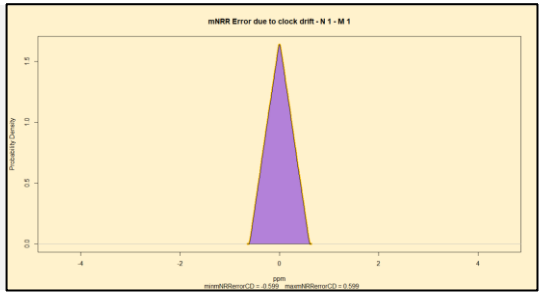
(mNRR – Single Hop – N=1)

pDelay Interval **$\text{mNRR}_{\text{error}}$** **...due to Timestamp Error** **...due to Clock Drift**

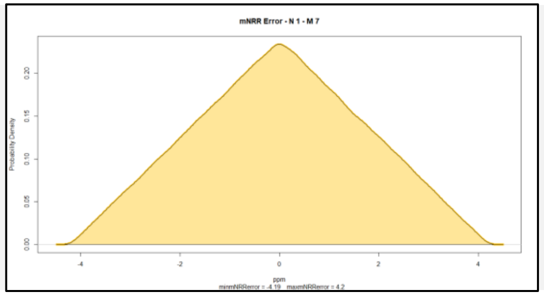
M=1 **1000ms**

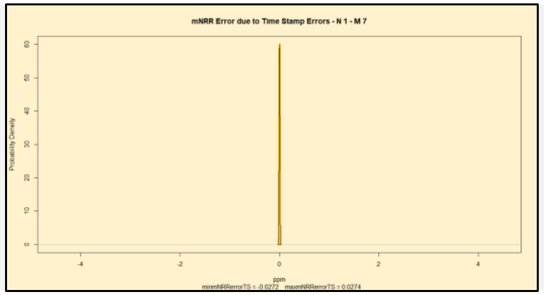


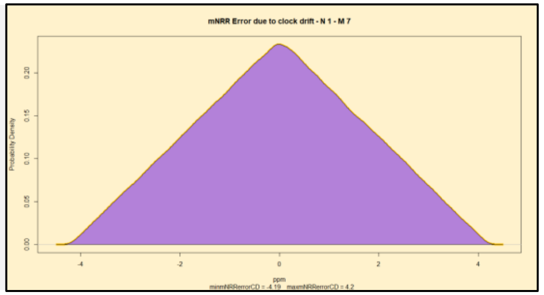


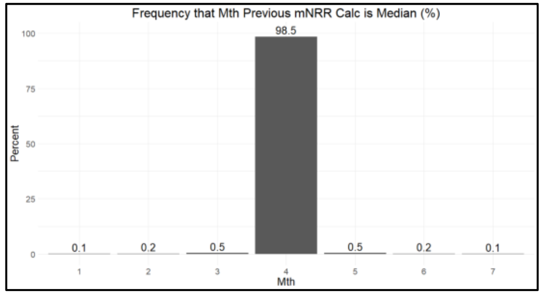


M=7 **1000ms**





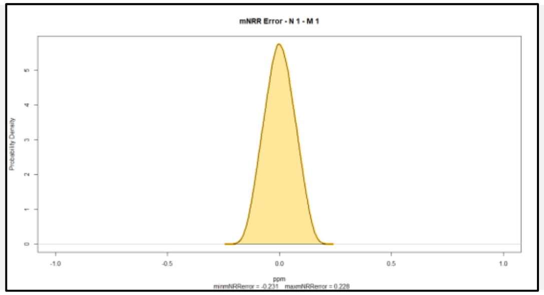
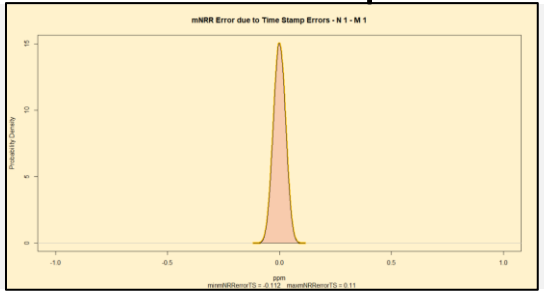
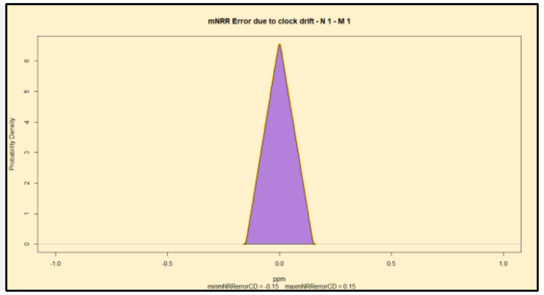
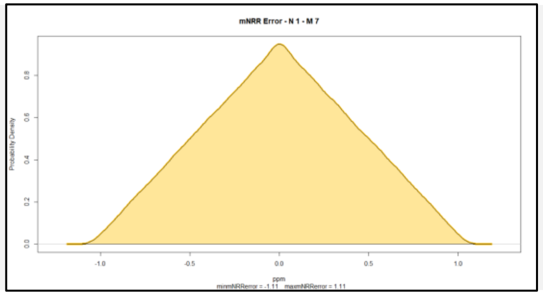
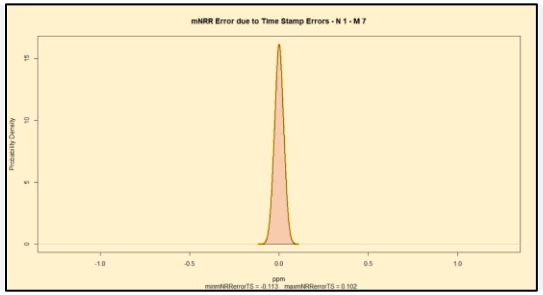
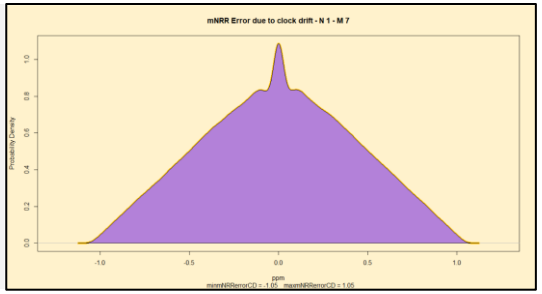


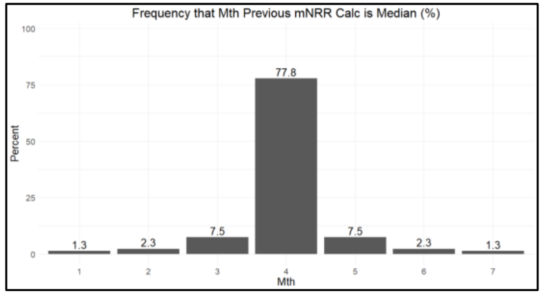


- For values of pDelay Interval where $\text{mNRR}_{\text{error}}$ due to Clock Drift dominates Timestamp error, taking the median of previous NRR calculations only increases $\text{mNRR}_{\text{error}}$.

Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

	pDelay Interval	**$\text{mNRR}_{\text{error}}$**	**...due to Timestamp Error**	**...due to Clock Drift**
M=1	**250ms**			
M=7	**250ms**			



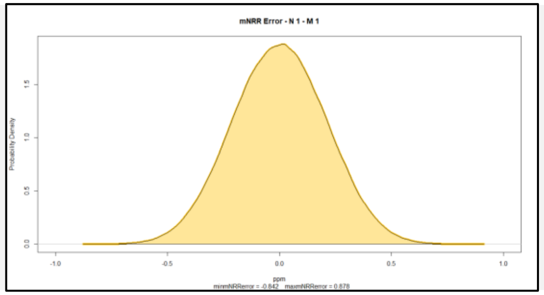
- For values of pDelay Interval where $\text{mNRR}_{\text{error}}$ due to Clock Drift and Timestamp error are of comparable magnitude, taking the median of previous NRR calculations only increases $\text{mNRR}_{\text{error}}$.

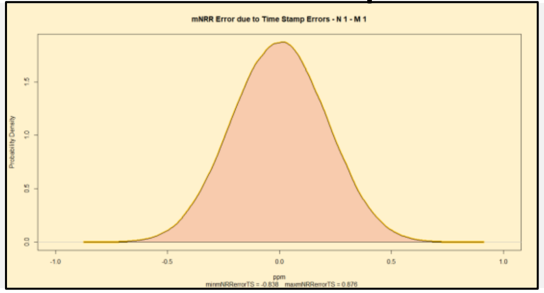
Effect of Taking Median of mNRR Calculations

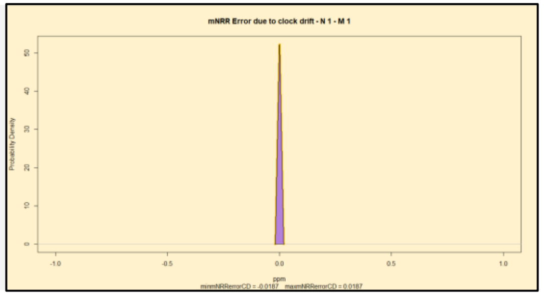
(mNRR – Single Hop – N=1)

pDelay Interval | **$mNRR_{error}$** | **...due to Timestamp Error** | **...due to Clock Drift**

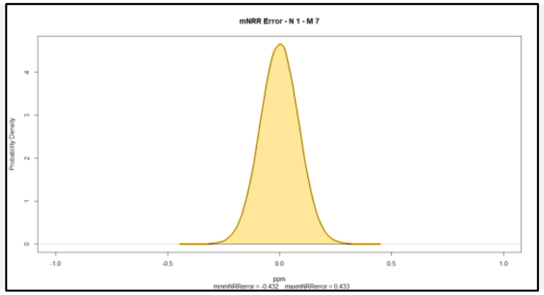
M=1 **31.25ms**

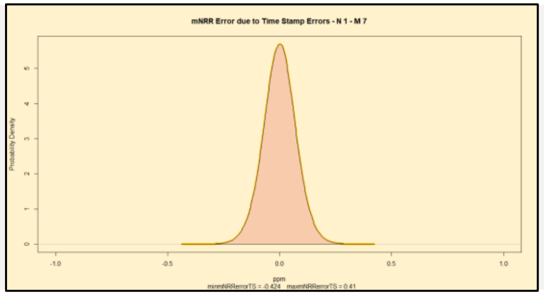


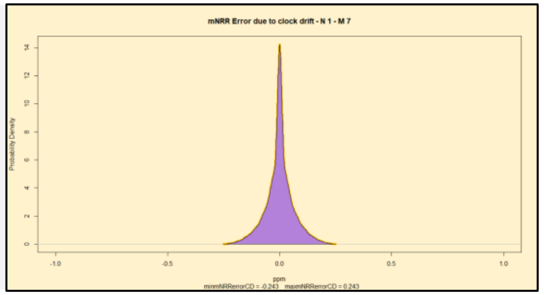


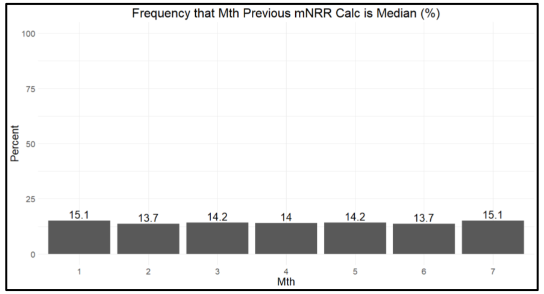


M=7 **31.25ms**







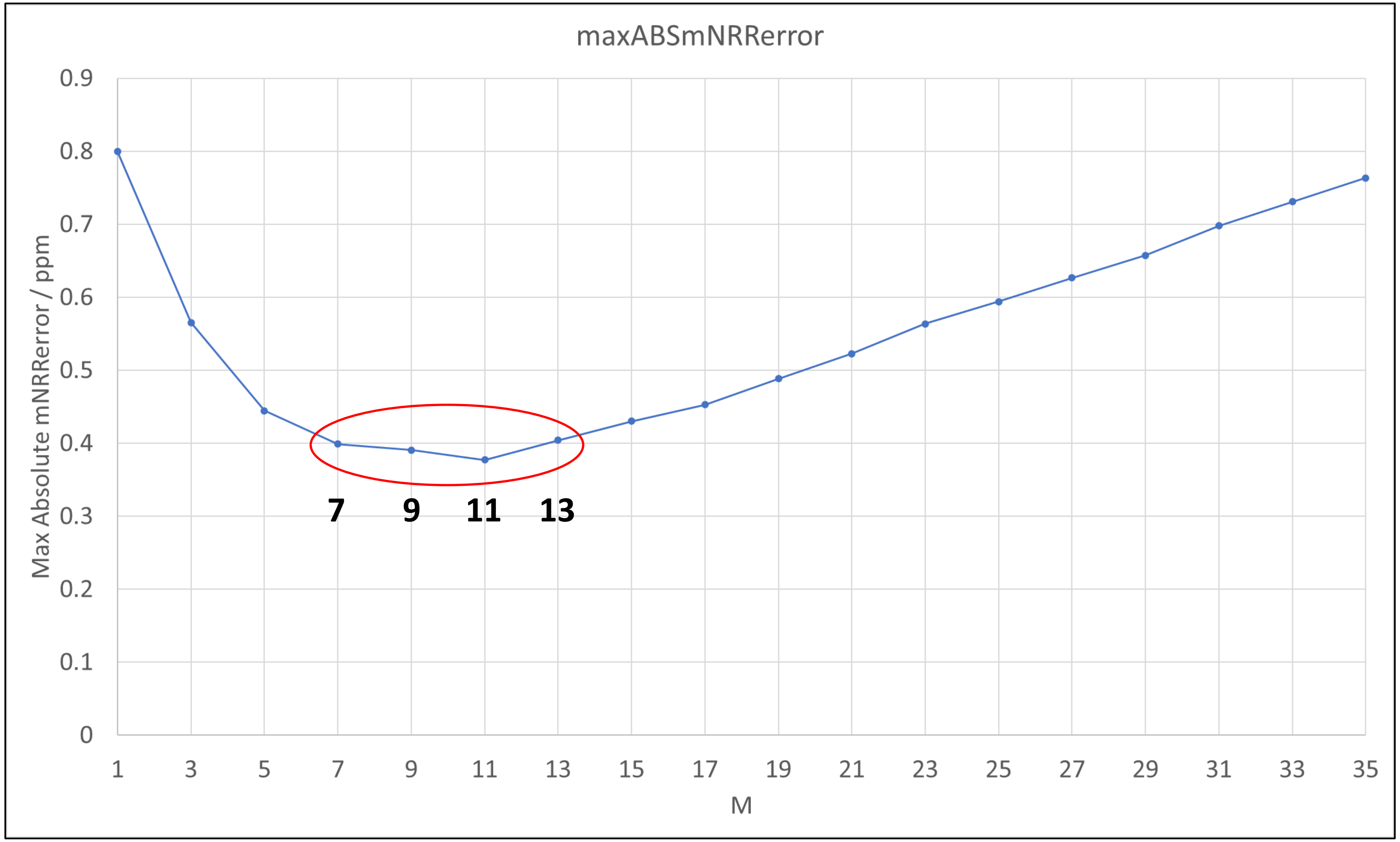


- For values of pDelay Interval where $mNRR_{error}$ due to Timestamp error dominates error due to Clock Drift, taking the median of previous NRR calculations can reduce $mNRR_{error}$.
- This raises the question: what is the optimal value of M?

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	

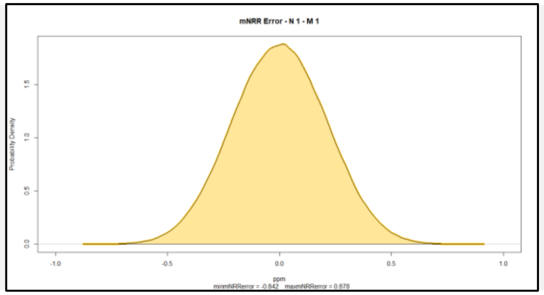
Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $\text{mNRR}_{\text{error}}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of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

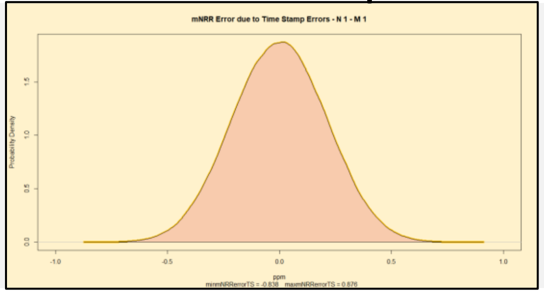
Effect of Taking Median of mNRR Calculations

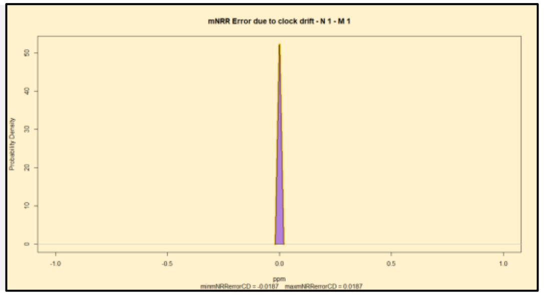
(mNRR – Single Hop – N=1)

pDelay Interval | **$\text{mNRR}_{\text{error}}$** | **...due to Timestamp Error** | **...due to Clock Drift**

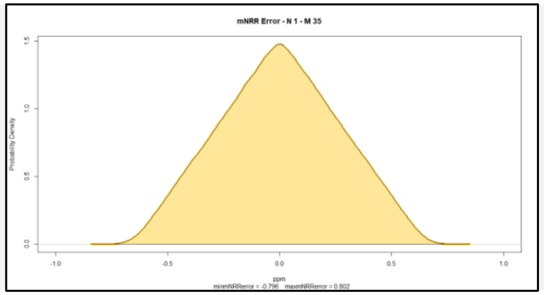
M=1 **31.25ms**

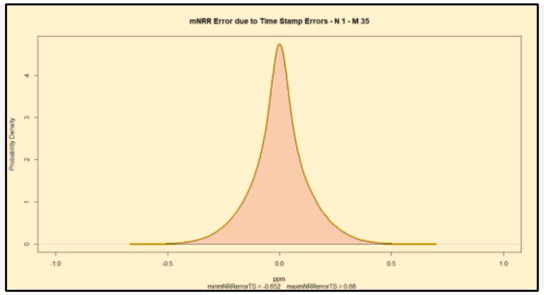


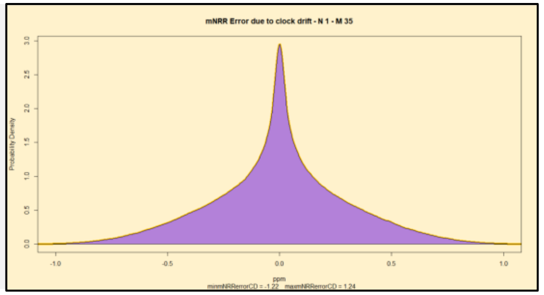


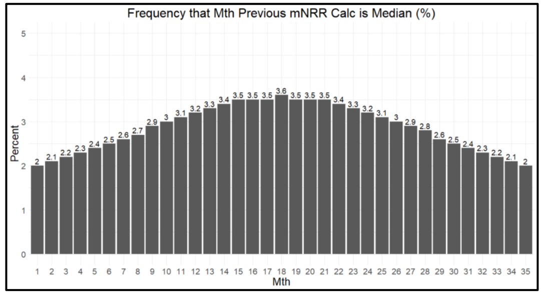


M=35 **31.25ms**





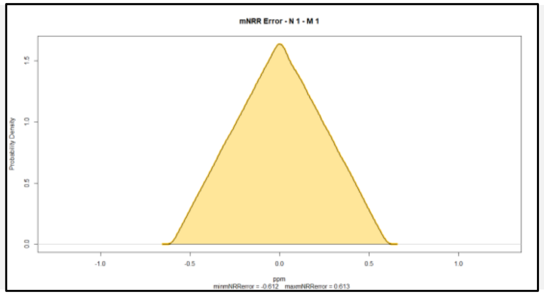
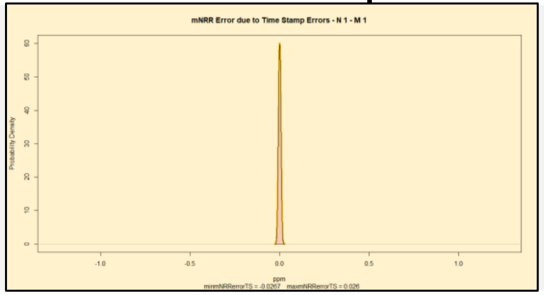
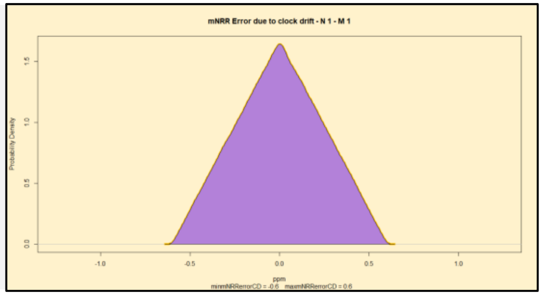
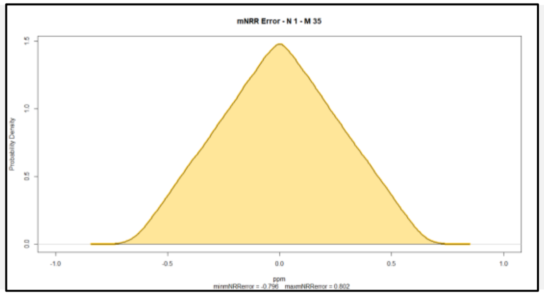
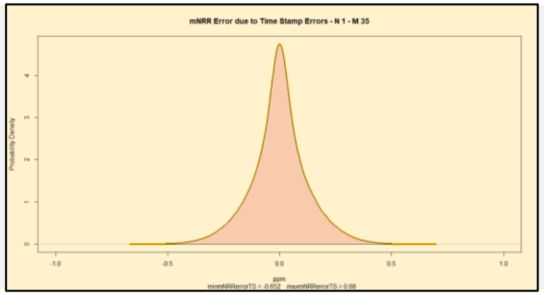
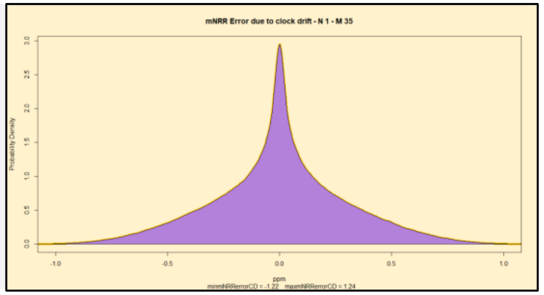


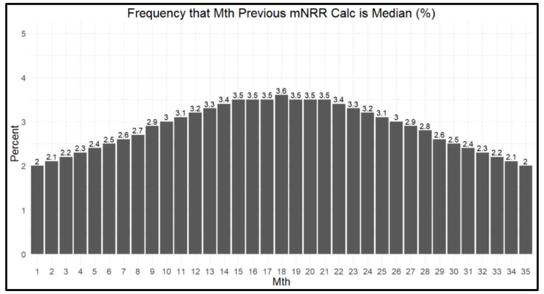


- Increasing M to 35 results in using NRR calculations from up to 1093.75ms in the past...and an $\text{mNRR}_{\text{errror}}$ distribution that is similar magnitude to pDelayInterval of 1000ms (and M=1).
- However...

Effect of Taking Median of mNRR Calculations

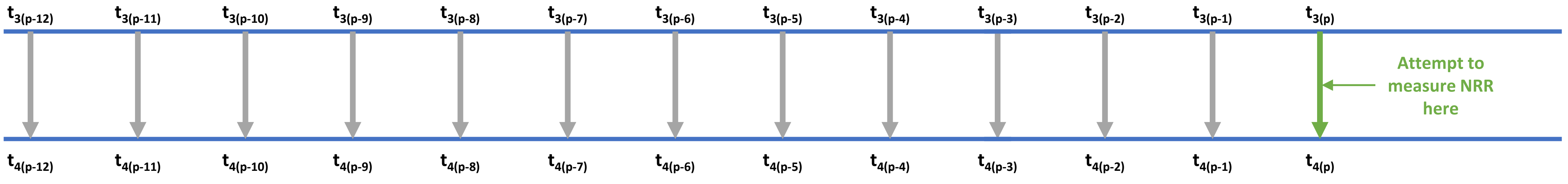
(mNRR – Single Hop – N=1)

	pDelay Interval	$\text{mNRR}_{\text{error}}$	...due to Timestamp Error	...due to Clock Drift
M=1	**1000ms**			
M=35	**31.25ms**			



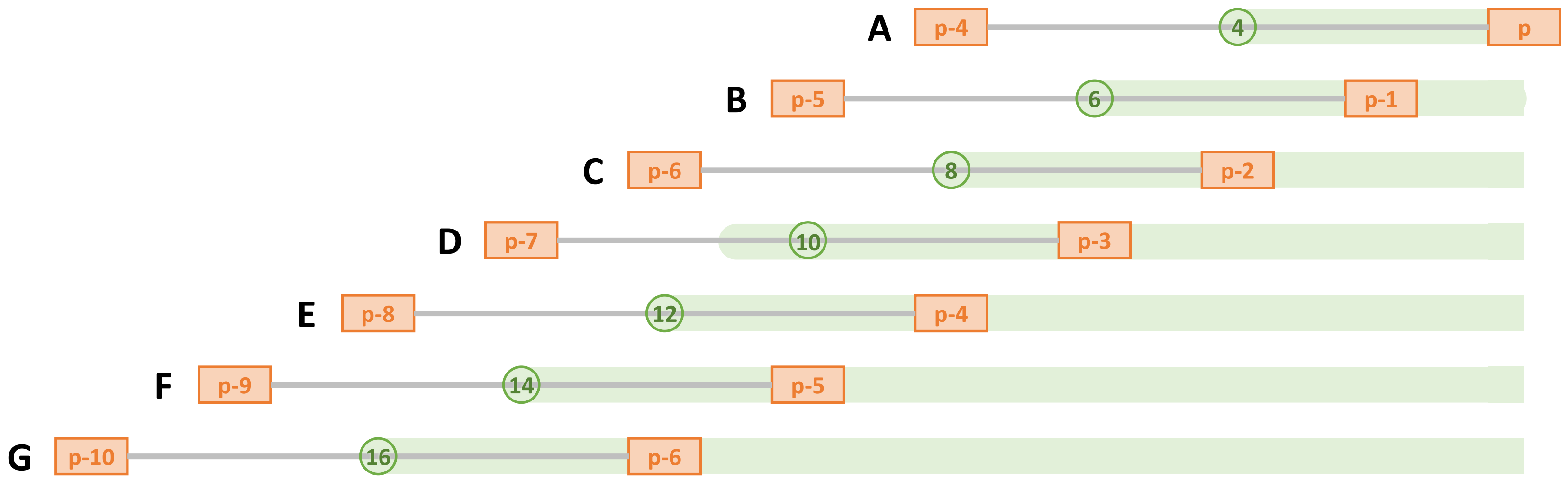
- The distribution of the underlying Timestamp errors and Clock Drift errors are very different.

mNRR Smoothing – Combining N & M



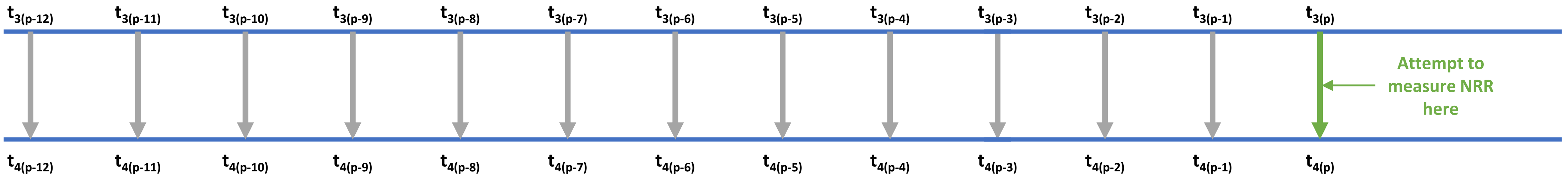
N = 4

M = 7



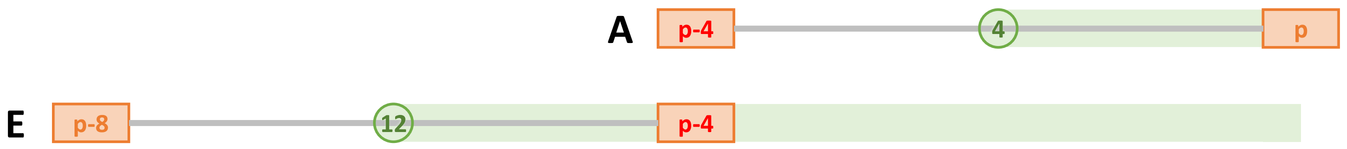
Note that the timestamps from p-4, p-5 & p-6 are used twice...which has an interesting effect...

mNRR Smoothing – Combining N & M



N = 4

M = 7



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

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

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

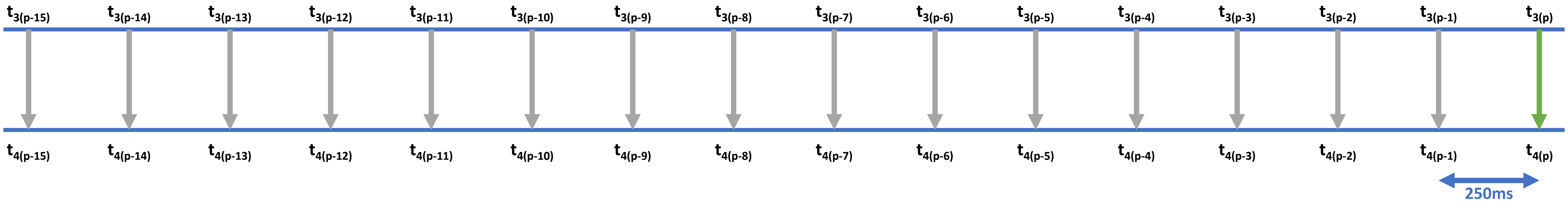
Series 1

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

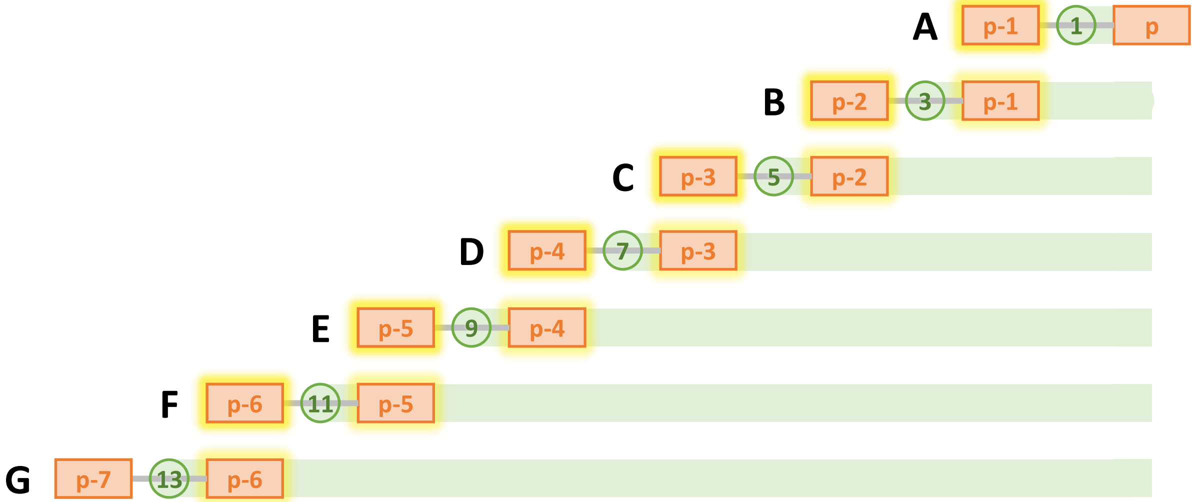


Series 1

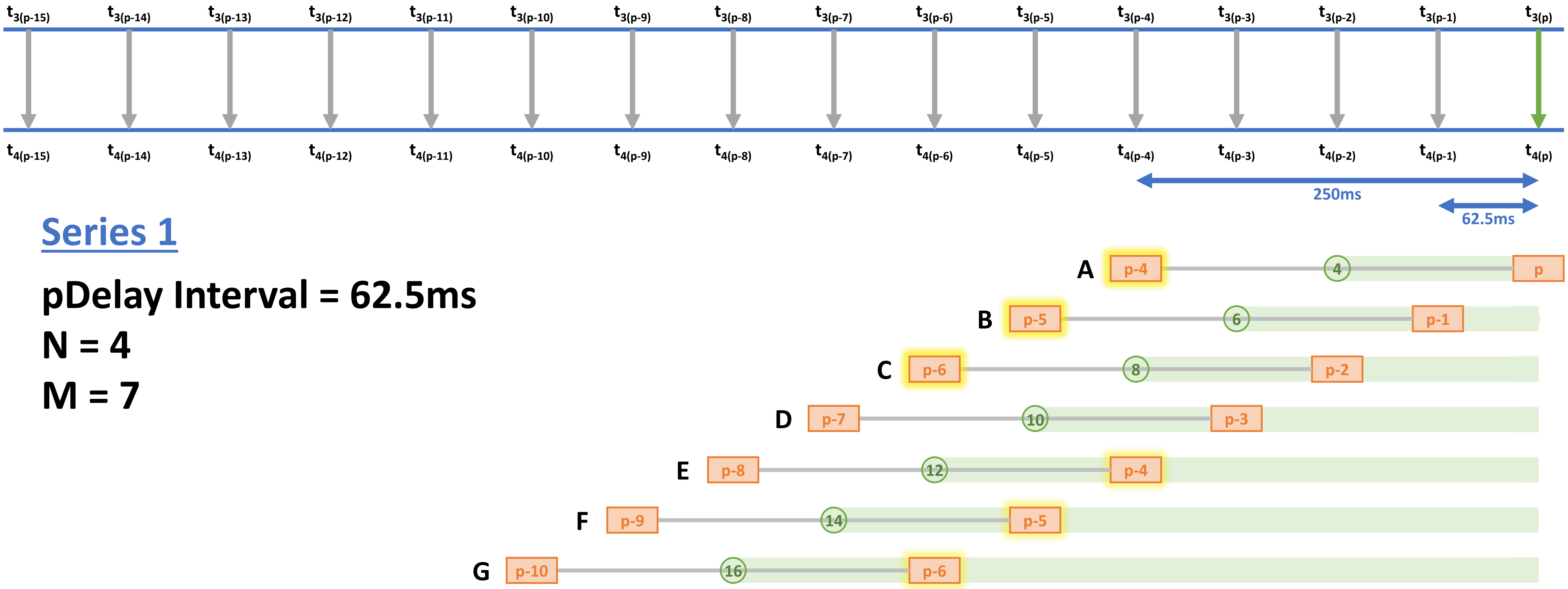
pDelay Interval = 250ms

N = 1

M = 7



mNRR Smoothing – Combining N & M



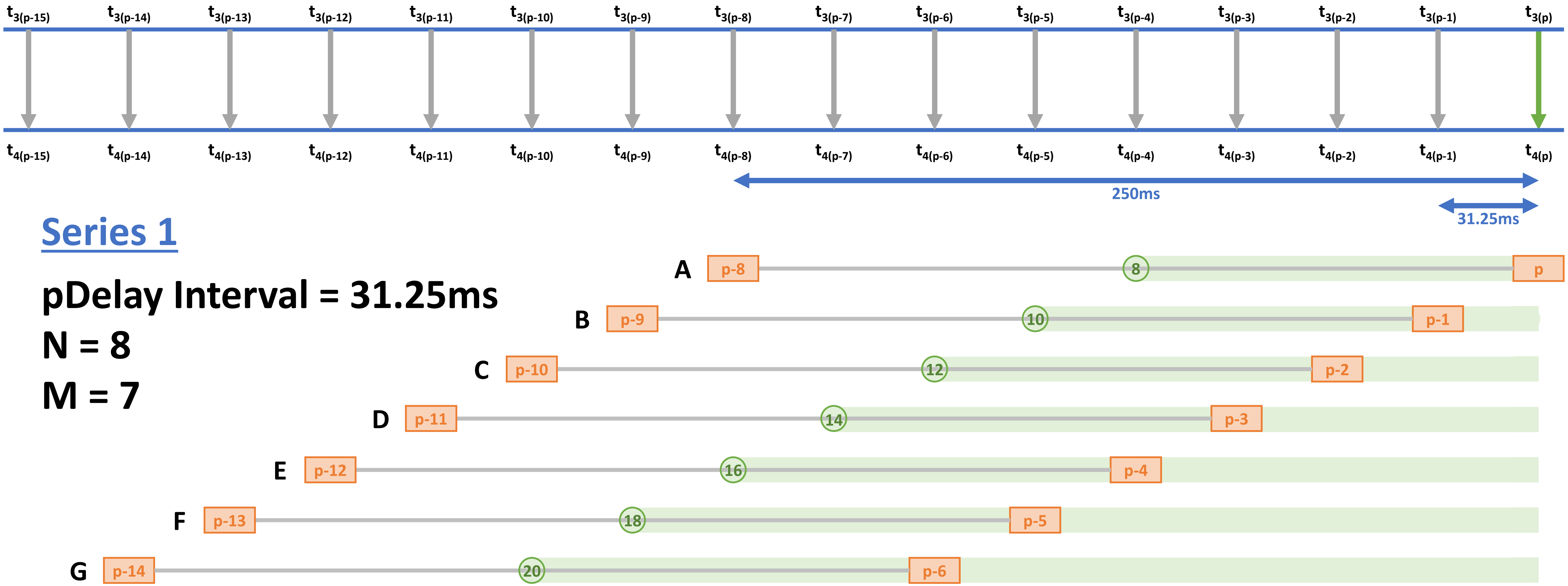
Series 1

pDelay Interval = 62.5ms

N = 4

M = 7

mNRR Smoothing – Combining N & M



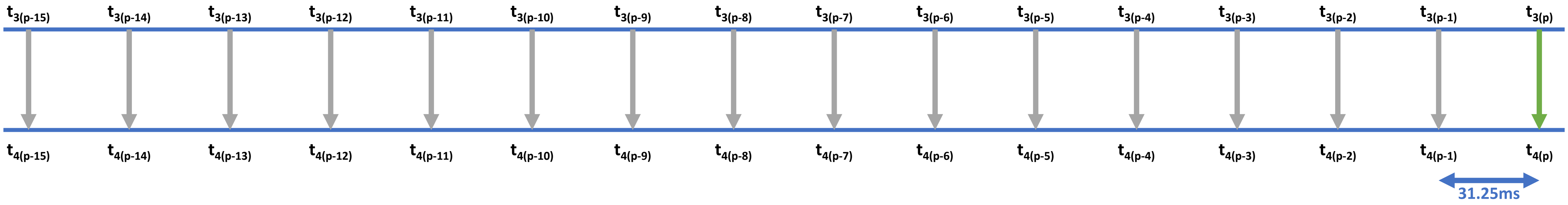
Series 1

pDelay Interval = 31.25ms

N = 8

M = 7

mNRR Smoothing – Combining N & M

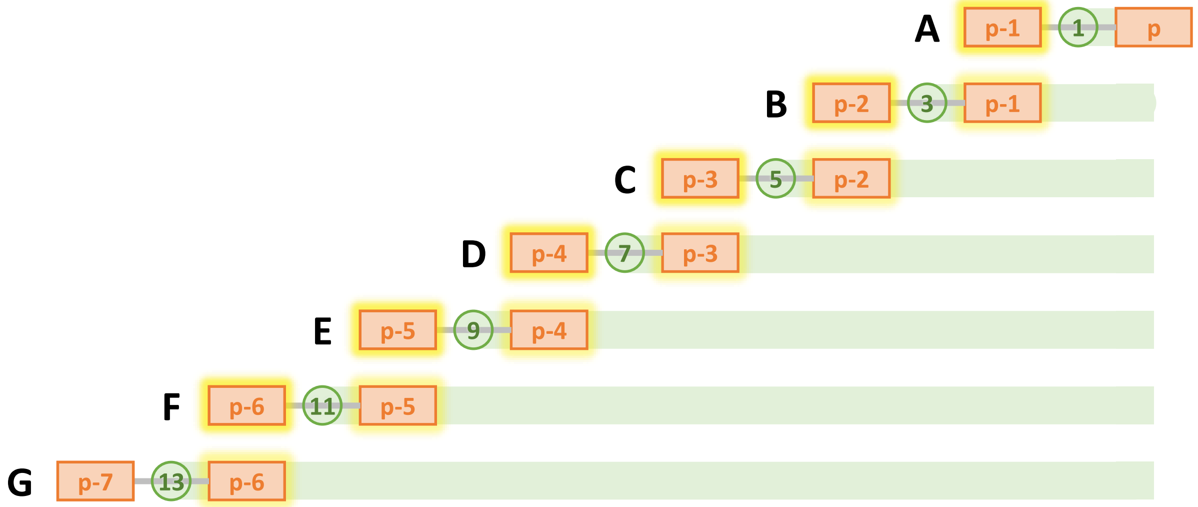


Series 2

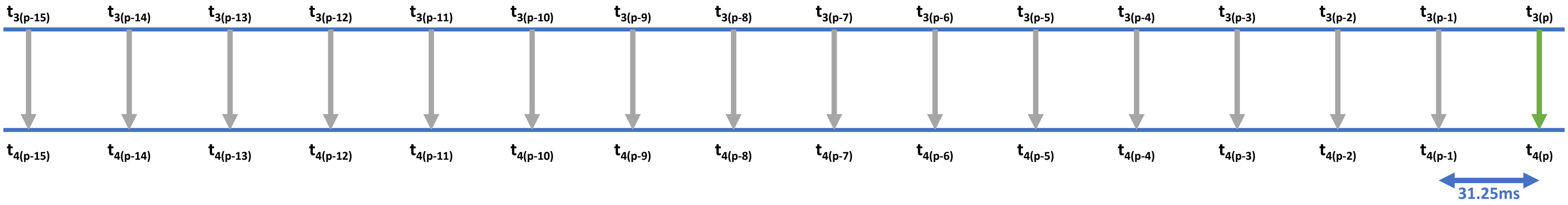
pDelay Interval = 31.25ms

N = 1

M = 7



mNRR Smoothing – Combining N & M

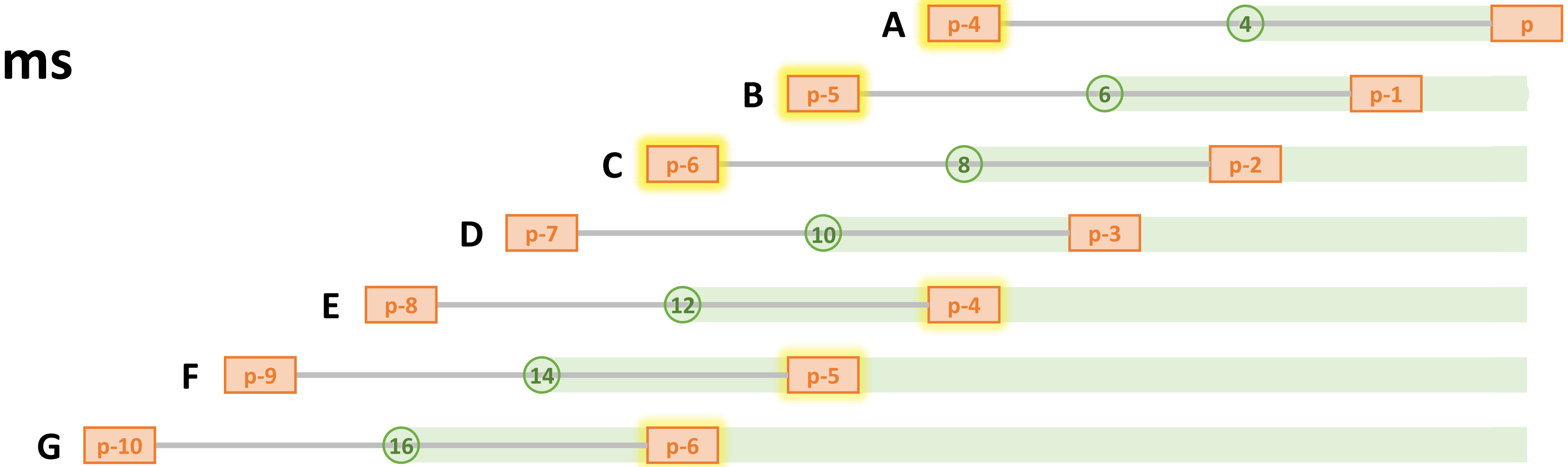


Series 2

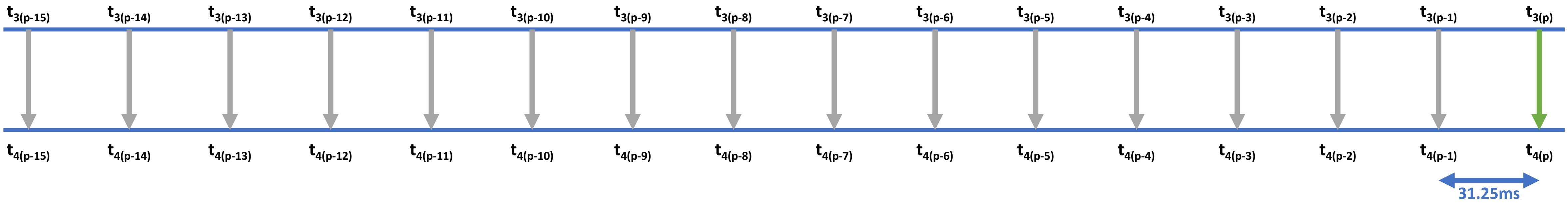
pDelay Interval = 31.25ms

N = 4

M = 7



mNRR Smoothing – Combining N & M

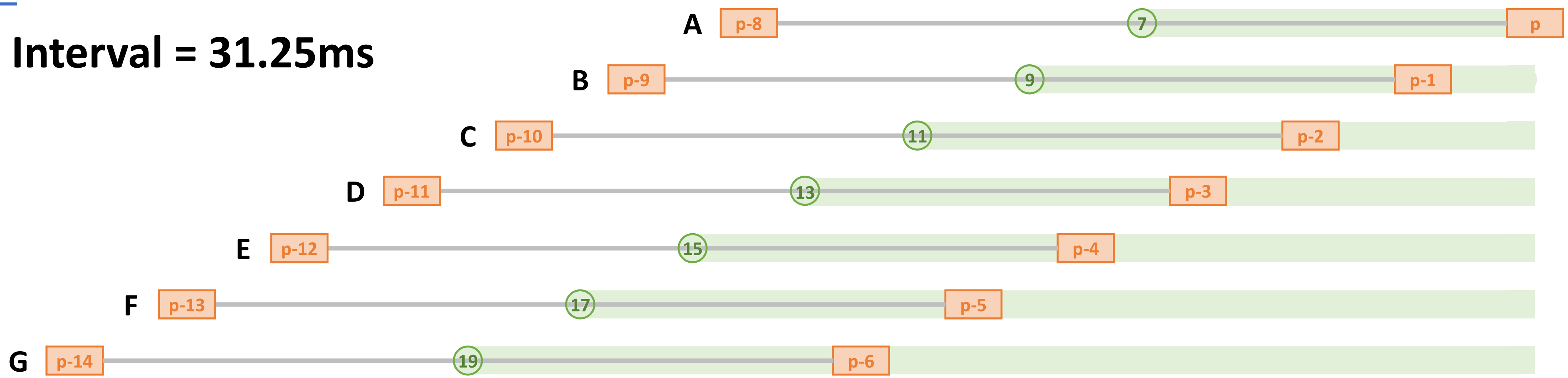


Series 2

pDelay Interval = 31.25ms

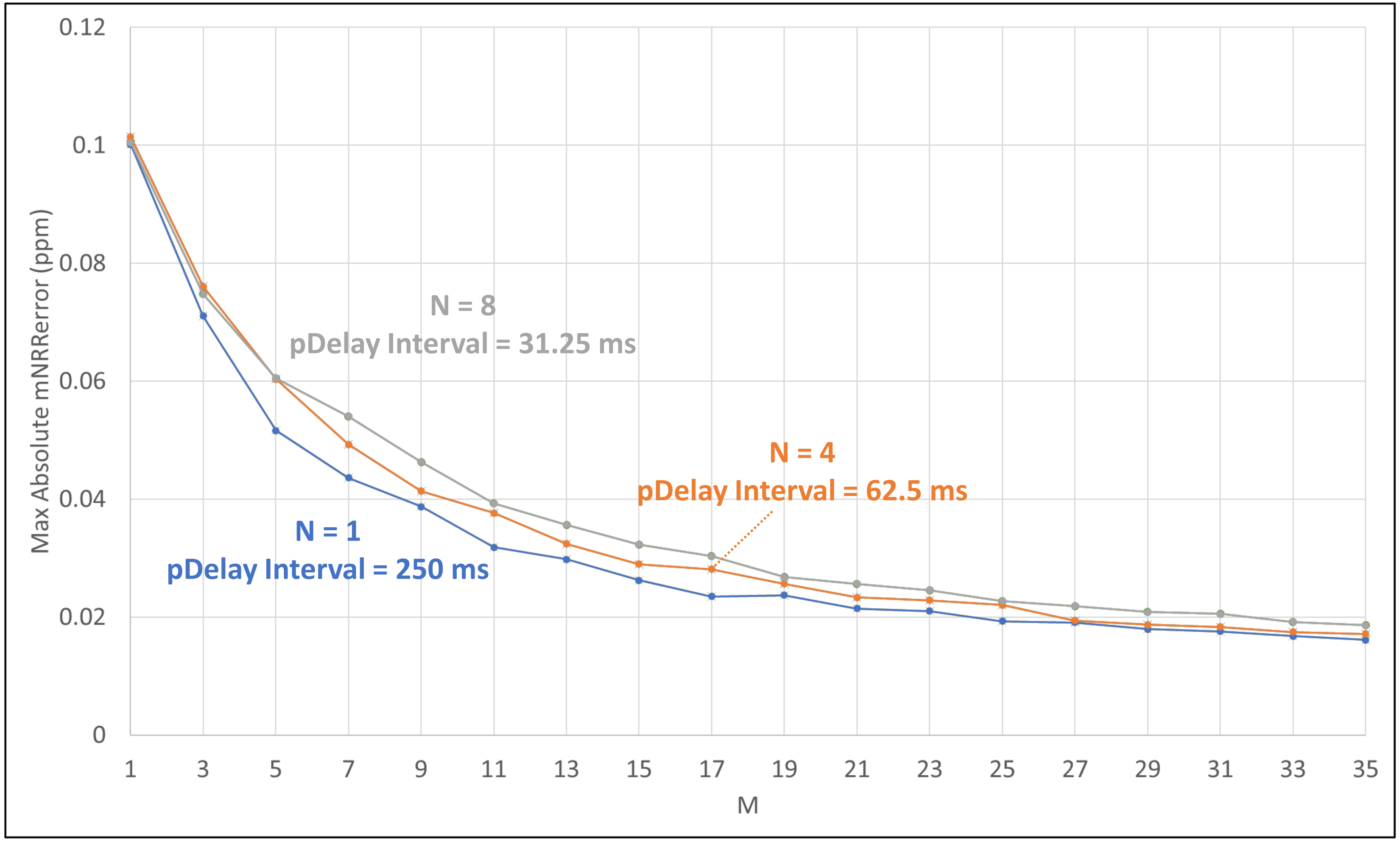
N = 7

M = 7



Varying M & N – No Clock Drift

(N x pDelayInterval maintained at 250ms)

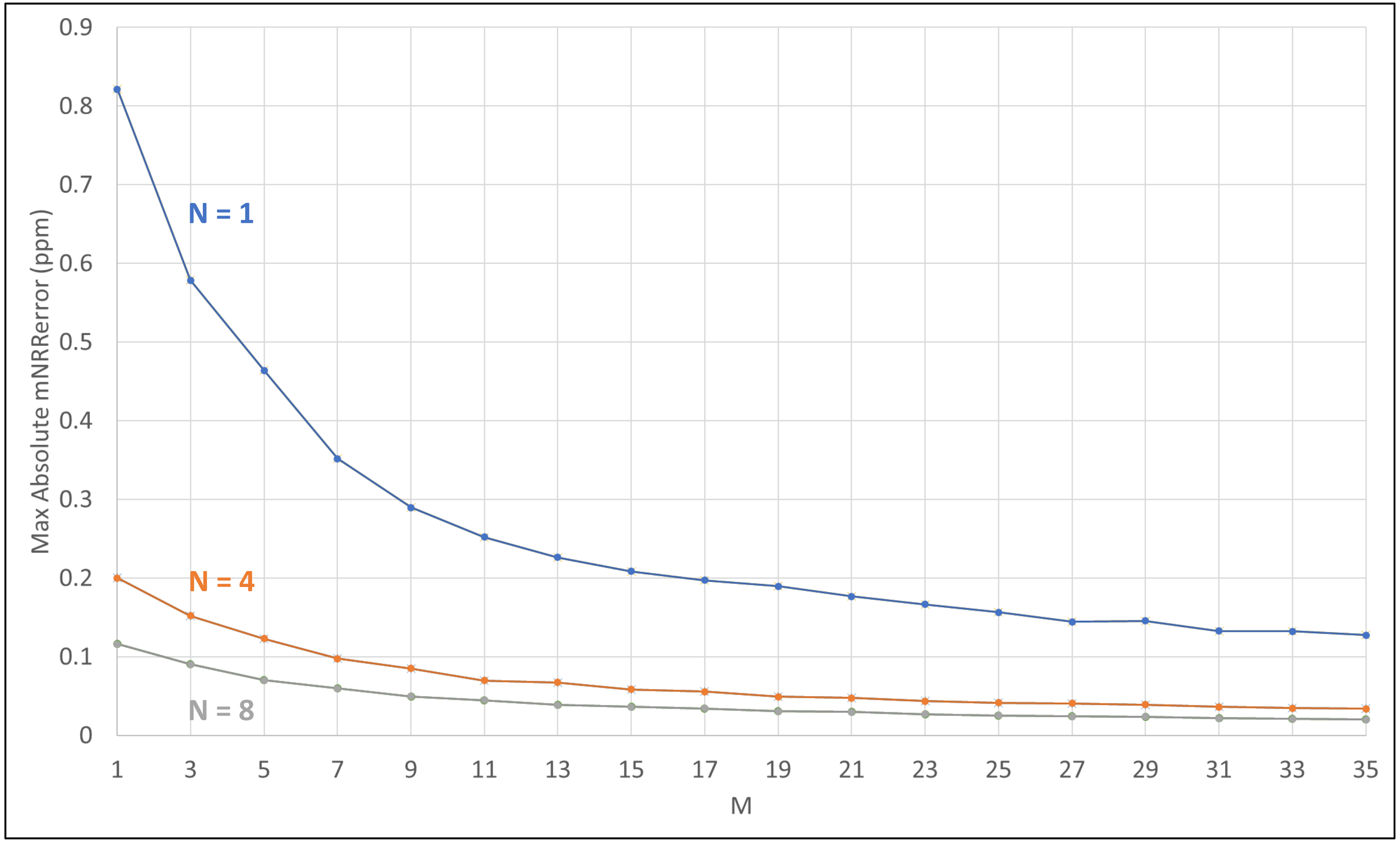


Input Errors		
Clock Drift	±0	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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)

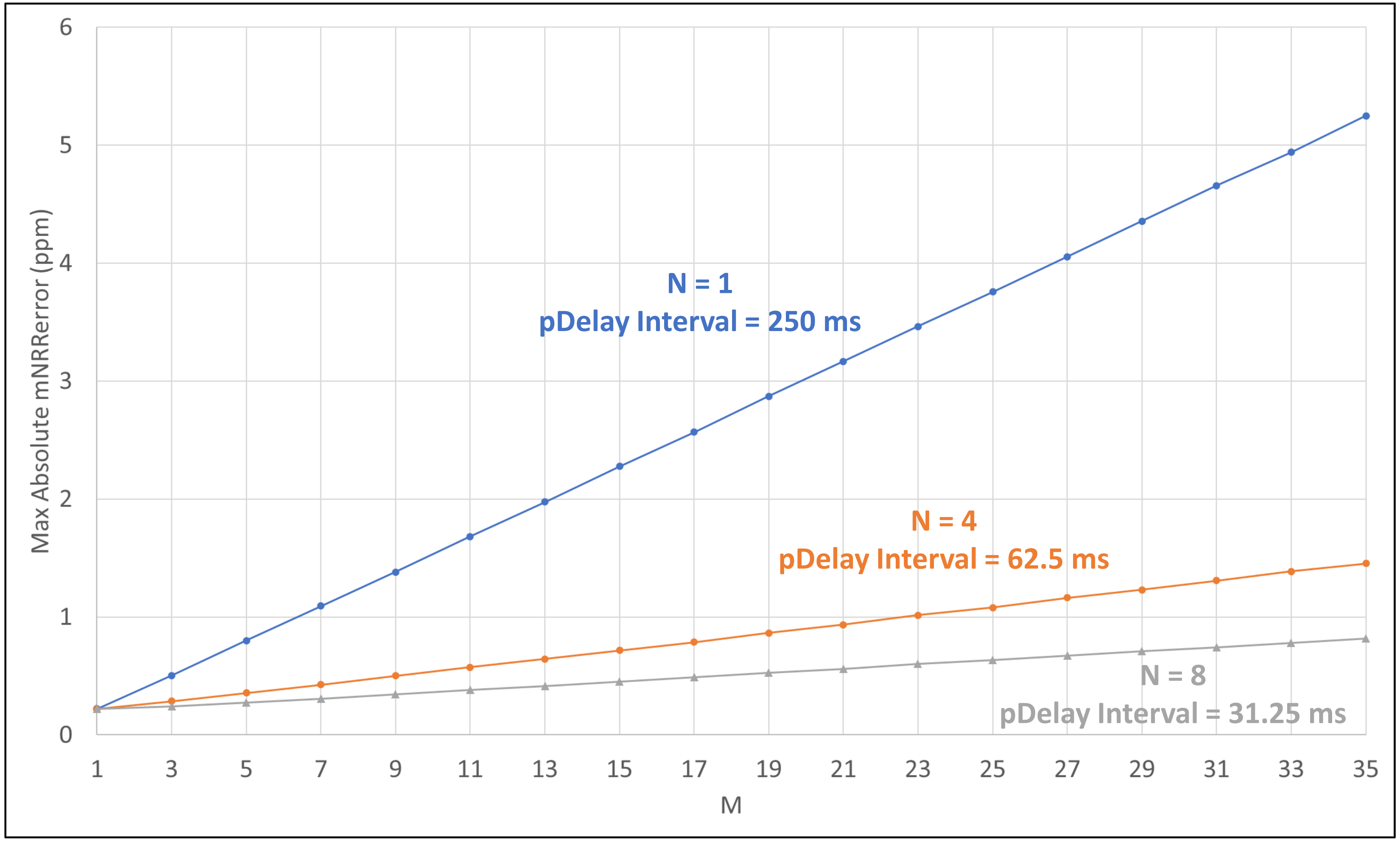


Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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)

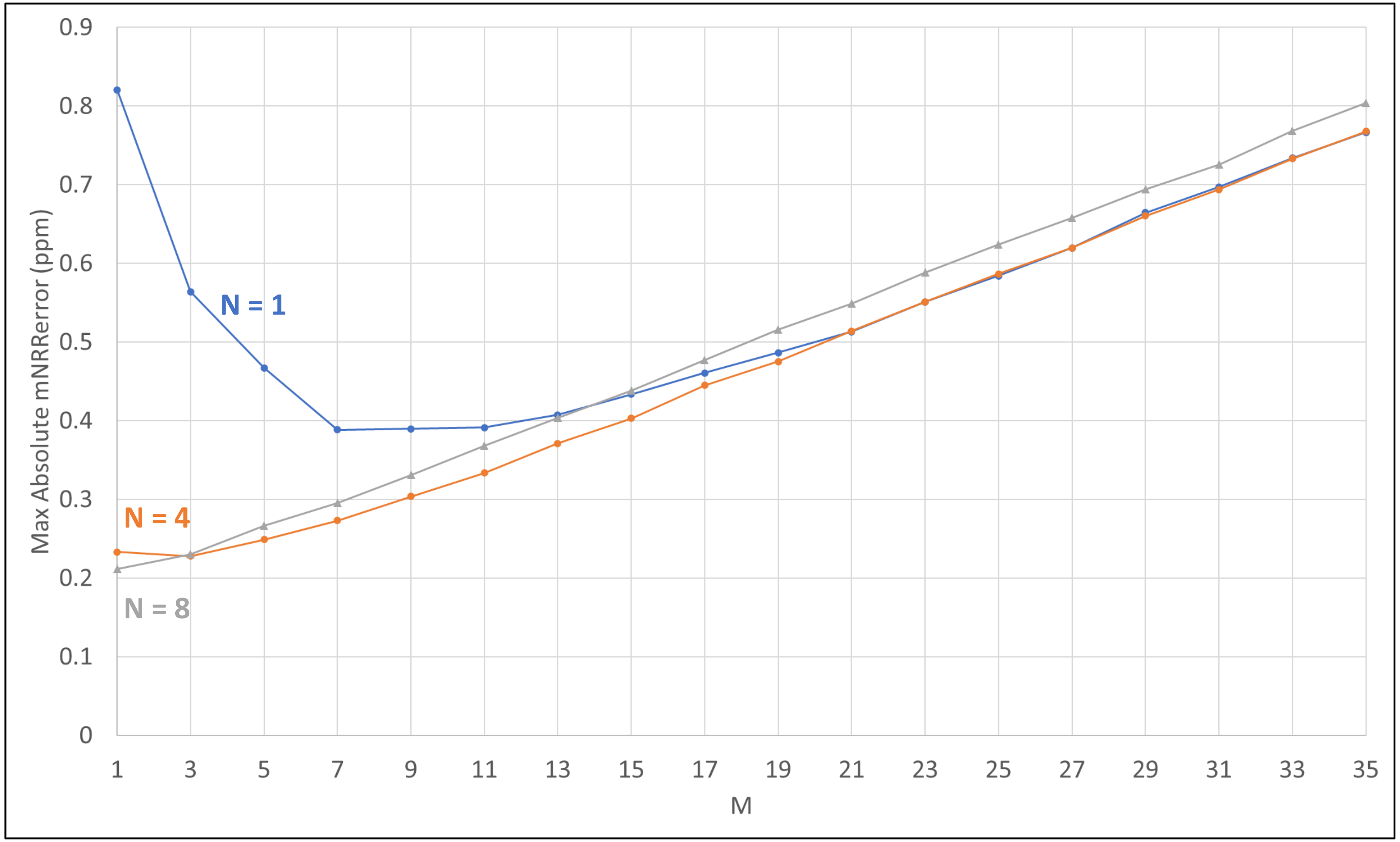


Input Errors		
Clock Drift	±0	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

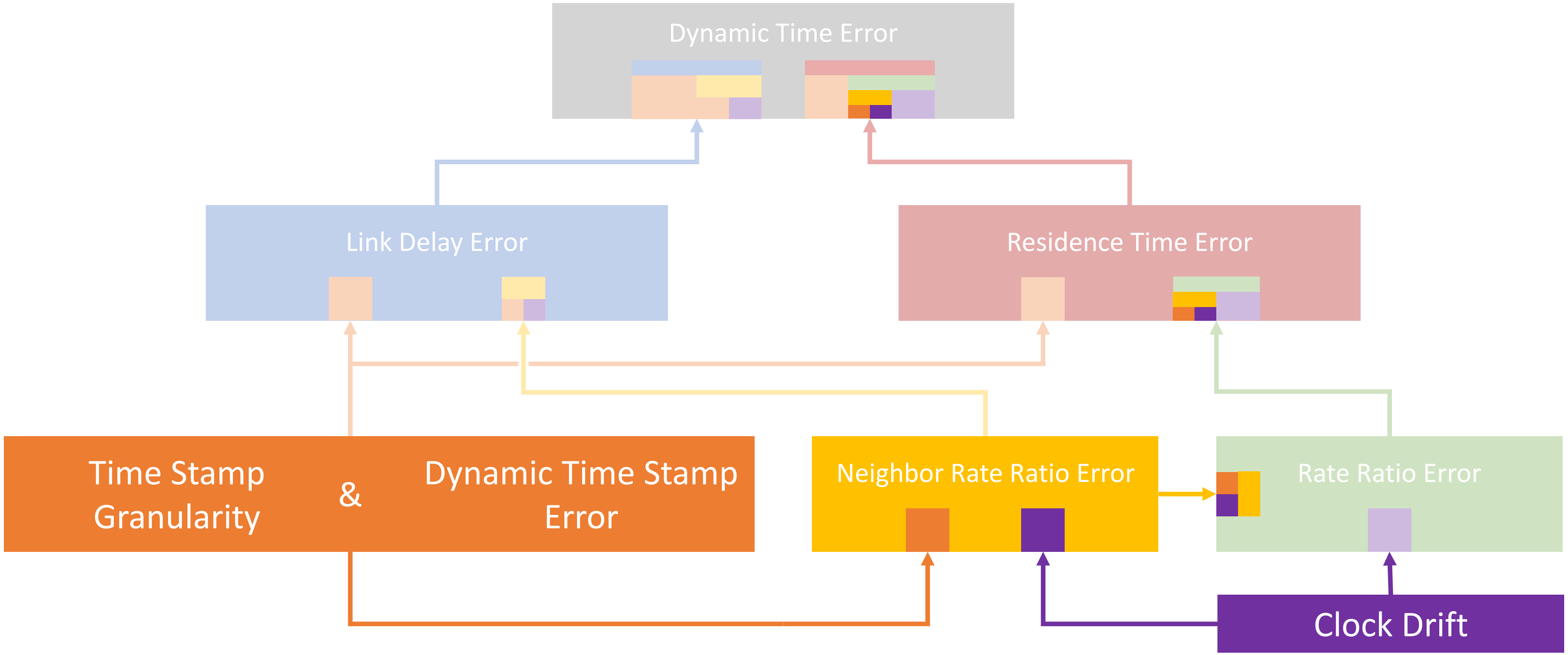
(N x pDelayInterval maintained at 250ms)



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	

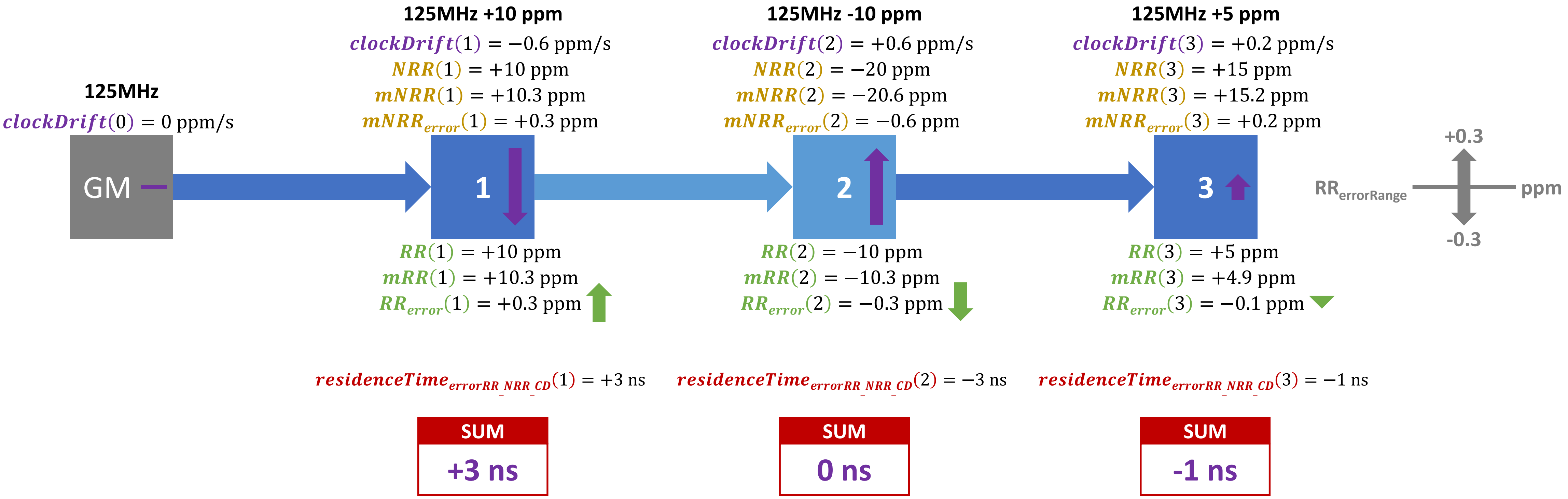
Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $\text{mNRR}_{\text{error}}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of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



125MHz

$clockDrift(0) = 0$ ppm/s

125MHz +10 ppm

$clockDrift(1) = -0.6$ ppm/s
$NRR(1) = +10$ ppm
$mNRR(1) = +10.3$ ppm
$mNRR_{error}(1) = +0.3$ ppm

$RR(1) = +10$ ppm
$mRR(1) = +10.3$ ppm
$RR_{error}(1) = +0.3$ ppm

$residenceTime_{errorRR\_NRR\_CD}(1) = +3$ ns

SUM
+3 ns

125MHz -10 ppm

$clockDrift(2) = +0.6$ ppm/s
$NRR(2) = -20$ ppm
$mNRR(2) = -20.6$ ppm
$mNRR_{error}(2) = -0.6$ ppm

$RR(2) = -10$ ppm
$mRR(2) = -10.3$ ppm
$RR_{error}(2) = -0.3$ ppm

$residenceTime_{errorRR\_NRR\_CD}(2) = -3$ ns

SUM
0 ns

125MHz +5 ppm

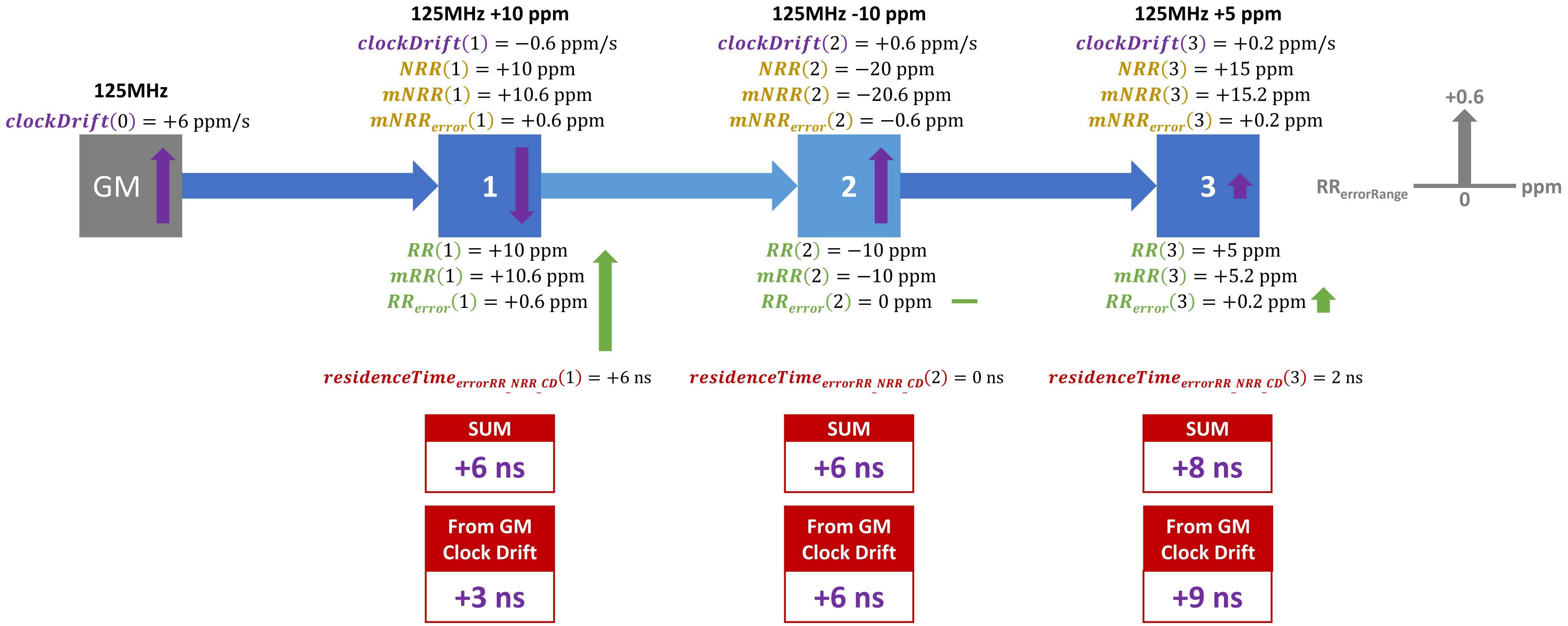
$clockDrift(3) = +0.2$ ppm/s
$NRR(3) = +15$ ppm
$mNRR(3) = +15.2$ ppm
$mNRR_{error}(3) = +0.2$ ppm

$RR(3) = +5$ ppm
$mRR(3) = +4.9$ ppm
$RR_{error}(3) = -0.1$ ppm

$residenceTime_{errorRR\_NRR\_CD}(3) = -1$ ns

SUM
-1 ns

How $\mathit{residenceTime}_{\mathit{errorRR\_NRR\_CD}}$ Accumulates



125MHz

$\mathit{clockDrift}(0) = +6\ \text{ppm/s}$

125MHz +10 ppm

$\mathit{clockDrift}(1) = -0.6\ \text{ppm/s}$

$\mathit{NRR}(1) = +10\ \text{ppm}$

$\mathit{mNRR}(1) = +10.6\ \text{ppm}$

$\mathit{mNRR}_{error}(1) = +0.6\ \text{ppm}$

$\mathit{RR}(1) = +10\ \text{ppm}$

$\mathit{mRR}(1) = +10.6\ \text{ppm}$

$\mathit{RR}_{error}(1) = +0.6\ \text{ppm}$

$\mathit{residenceTime}_{\mathit{errorRR\_NRR\_CD}}(1) = +6\ \text{ns}$

SUM
+6 ns

From GM Clock Drift
+3 ns

125MHz -10 ppm

$\mathit{clockDrift}(2) = +0.6\ \text{ppm/s}$

$\mathit{NRR}(2) = -20\ \text{ppm}$

$\mathit{mNRR}(2) = -20.6\ \text{ppm}$

$\mathit{mNRR}_{error}(2) = -0.6\ \text{ppm}$

$\mathit{RR}(2) = -10\ \text{ppm}$

$\mathit{mRR}(2) = -10\ \text{ppm}$

$\mathit{RR}_{error}(2) = 0\ \text{ppm}$

$\mathit{residenceTime}_{\mathit{errorRR\_NRR\_CD}}(2) = 0\ \text{ns}$

SUM
+6 ns

From GM Clock Drift
+6 ns

125MHz +5 ppm

$\mathit{clockDrift}(3) = +0.2\ \text{ppm/s}$

$\mathit{NRR}(3) = +15\ \text{ppm}$

$\mathit{mNRR}(3) = +15.2\ \text{ppm}$

$\mathit{mNRR}_{error}(3) = +0.2\ \text{ppm}$

$\mathit{RR}(3) = +5\ \text{ppm}$

$\mathit{mRR}(3) = +5.2\ \text{ppm}$

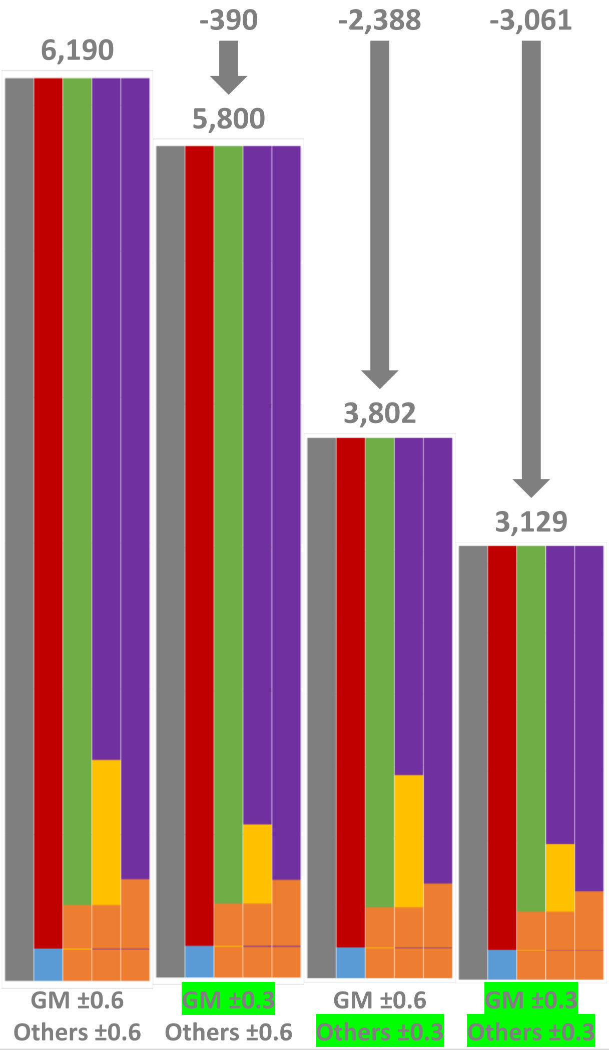
$\mathit{RR}_{error}(3) = +0.2\ \text{ppm}$

$\mathit{residenceTime}_{\mathit{errorRR\_NRR\_CD}}(3) = 2\ \text{ns}$

SUM
+8 ns

From GM Clock Drift
+9 ns

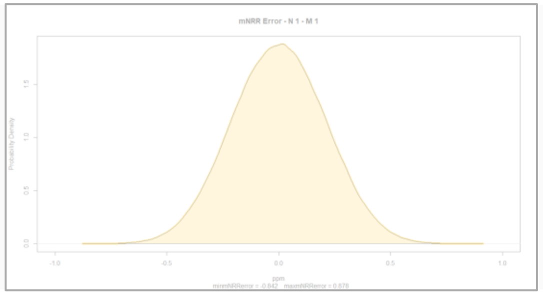
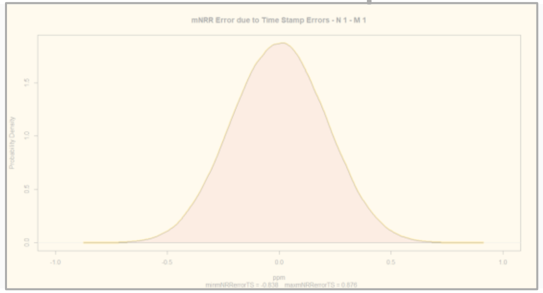
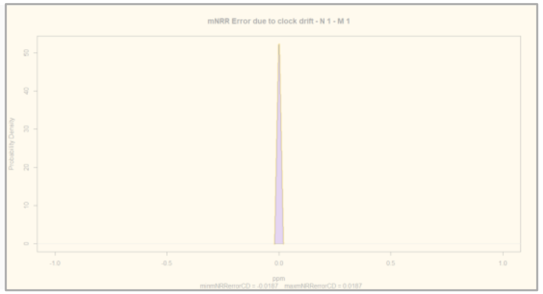
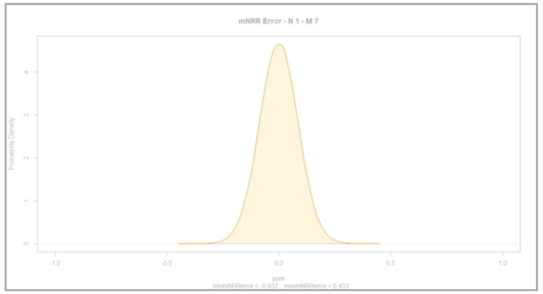
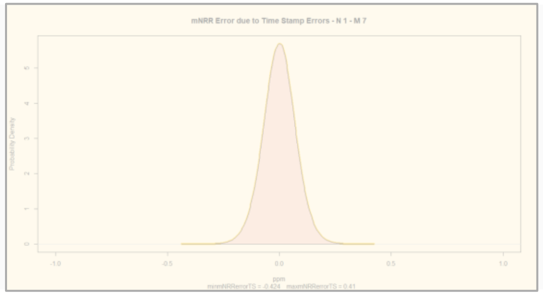
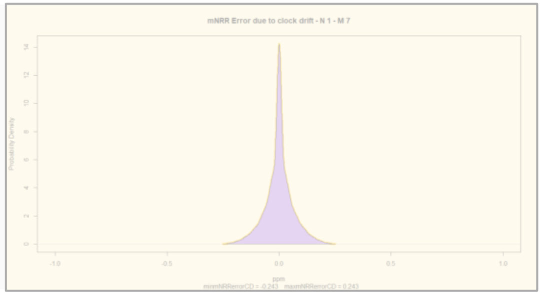
Clock Drift Sensitivity

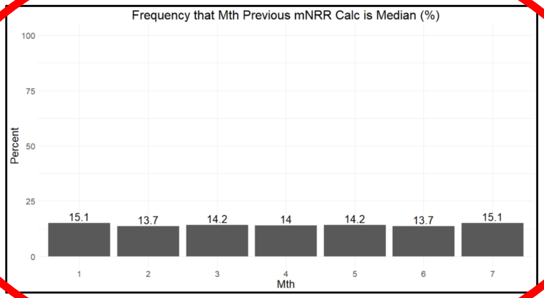


Input Errors		
GM Clock Drift Max	Variable	ppm/s
GM Clock Drift Min	Variable	ppm/s
Clock Drift (non-GM)	Variable	±ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay Interval	1000	ms
pDelay Response 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)

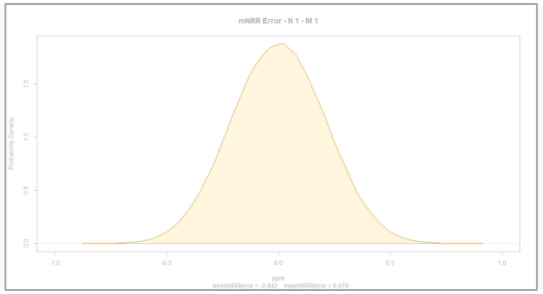
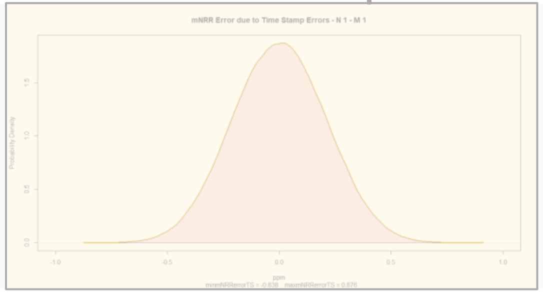
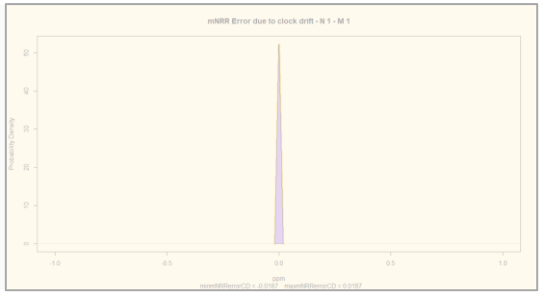
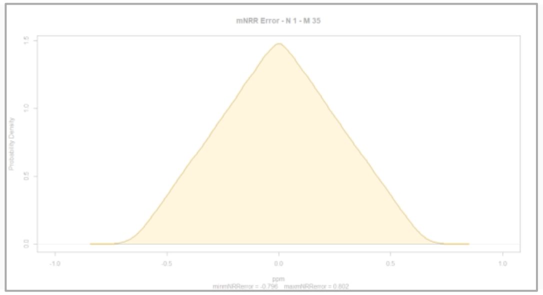
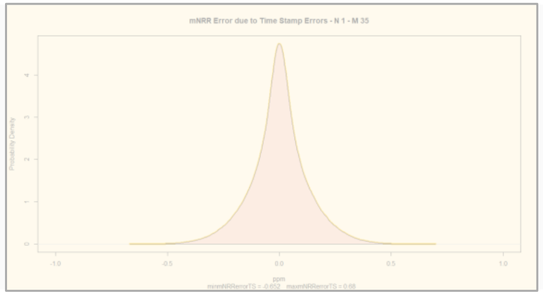
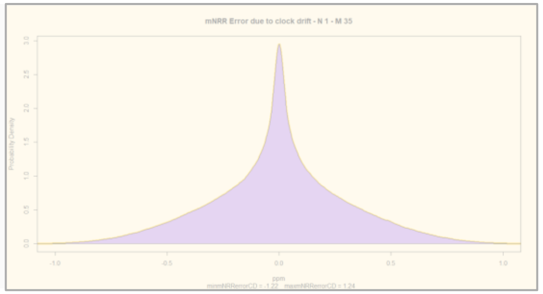
	pDelay Interval	$mNRR_{error}$	...due to Timestamp Error	...due to Clock Drift
M=1	**31.25ms**			
M=7	**31.25ms**			

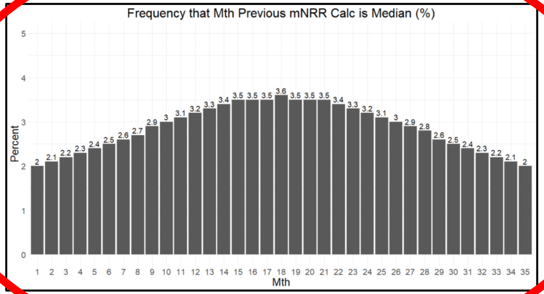


- For values of pDelay Interval where $mNRR_{error}$ due to Timestamp error dominates error due to Clock Drift, taking the median of previous NRR calculations can reduce $mNRR_{error}$.
- This raises the question: what is the optimal value of M?

Effect of Taking Median of mNRR Calculations

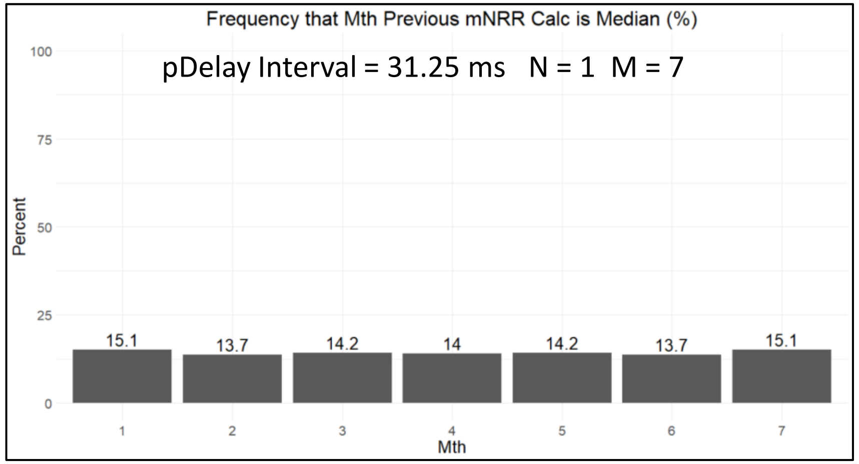
(mNRR – Single Hop – N=1)

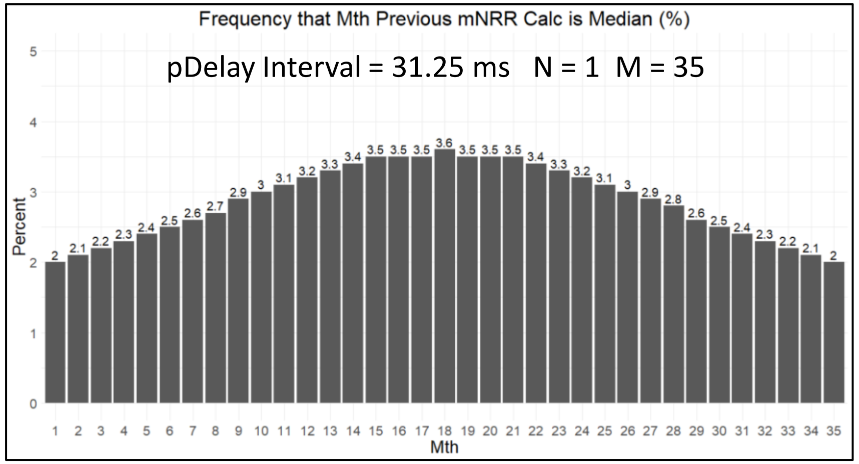
	pDelay Interval	$mNRR_{error}$	...due to Timestamp Error	...due to Clock Drift
M=1	31.25ms			
M=35	31.25ms			



- 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



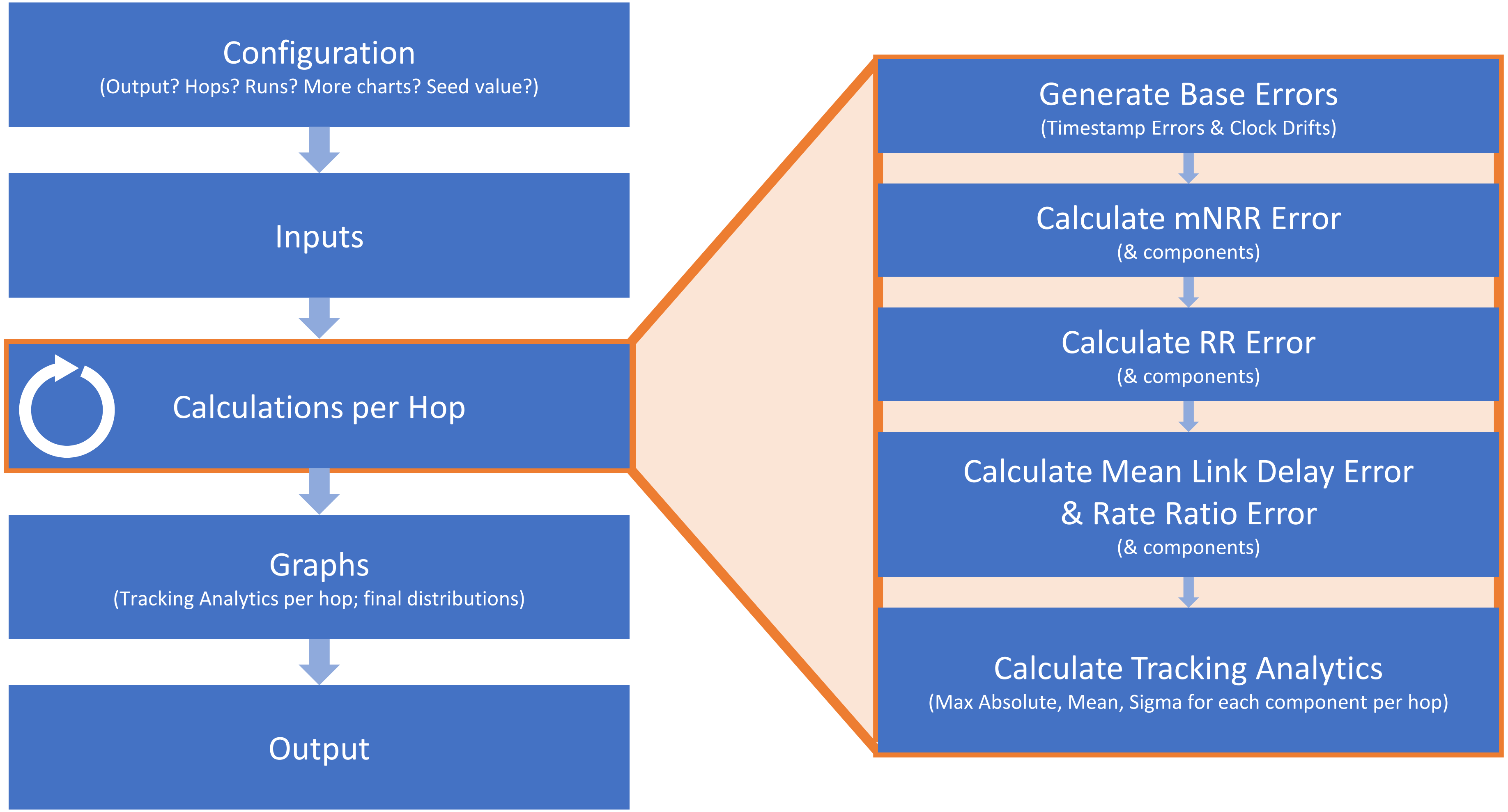


- Using the median of past NRR calculations brings with it a chance that a larger portion of the error won’t cancel out at the next node but will instead survive.
  - This portion will behave more like mNRRerror due to Clock Drift at the GM than at other nodes when M=1.

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



Configuration
(Output? Hops? Runs? More charts? Seed value?)

Inputs

Calculations per Hop

Graphs
(Tracking Analytics per hop; final distributions)

Output

Generate Base Errors
(Timestamp Errors & Clock Drifts)

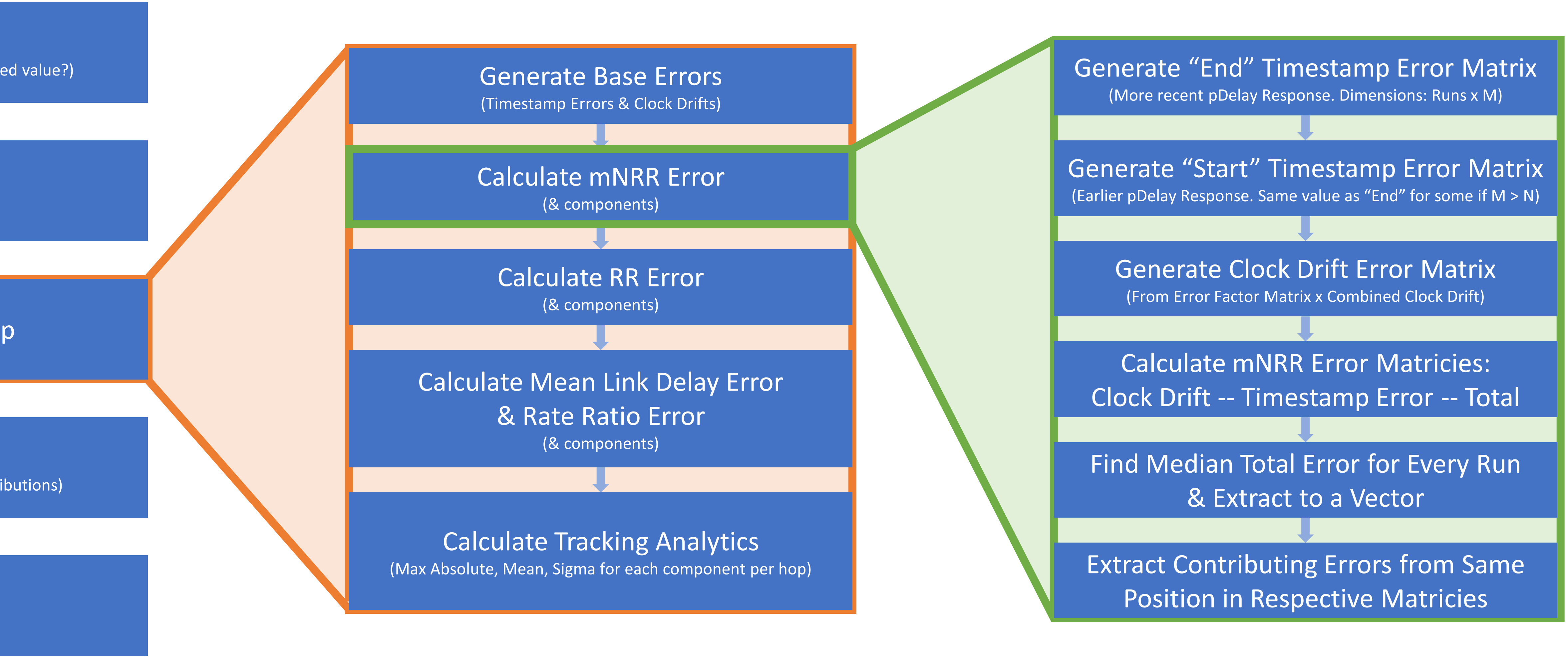
Calculate mNRR Error
(& components)

Calculate RR Error
(& components)

Calculate Mean Link Delay Error
& Rate Ratio Error
(& components)

Calculate Tracking Analytics
(Max Absolute, Mean, Sigma for each component per hop)

RStudio Script Summary



ed value?)

p

ibutions)

Generate Base Errors
(Timestamp Errors & Clock Drifts)

Calculate mNRR Error
(& components)

Calculate RR Error
(& components)

Calculate Mean Link Delay Error
& Rate Ratio Error
(& components)

Calculate Tracking Analytics
(Max Absolute, Mean, Sigma for each component per hop)

Generate "End" Timestamp Error Matrix
(More recent pDelay Response. Dimensions: Runs x M)

Generate "Start" Timestamp Error Matrix
(Earlier pDelay Response. Same value as "End" for some if M > N)

Generate Clock Drift Error Matrix
(From Error Factor Matrix x Combined Clock Drift)

Calculate mNRR Error Matricies:
Clock Drift -- Timestamp Error -- Total

Find Median Total Error for Every Run
& Extract to a Vector

Extract Contributing Errors from Same
Position in Respective Matricies

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...

$$if\ p \leq 1000, \mathrm{F} = \mathrm{X}$$

$$if\ p > 1000, \mathrm{F} = 1000$$

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

$$MeanLinkDelay(p) = \frac{\left(MeanLinkDelay(p-1) \times (F-1)\right) + pDelay(X)}{F}$$

- So, for example...

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

$$MeanLinkDelay(10500) = \frac{\left(MeanLinkDelay(10499) \times (999)\right) + pDelay(10500)}{1000}$$

- Is 1000 the right cap for F? Do we need a cap at all?
- Reset F if pDelay deviates too much from current MeanLinkDelay?
  - Deviates too much...repeatedly?

Algorithms – Clock Drift

Potential algorithm Clock Drift Compensation

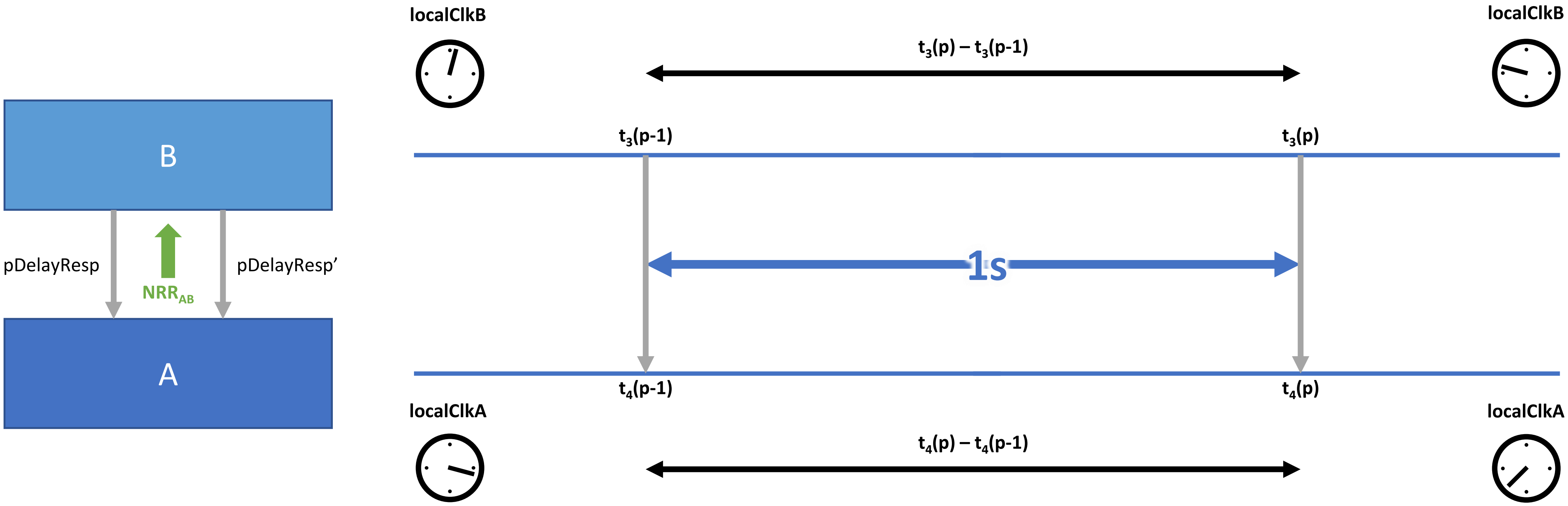
Compensating for Clock Drift

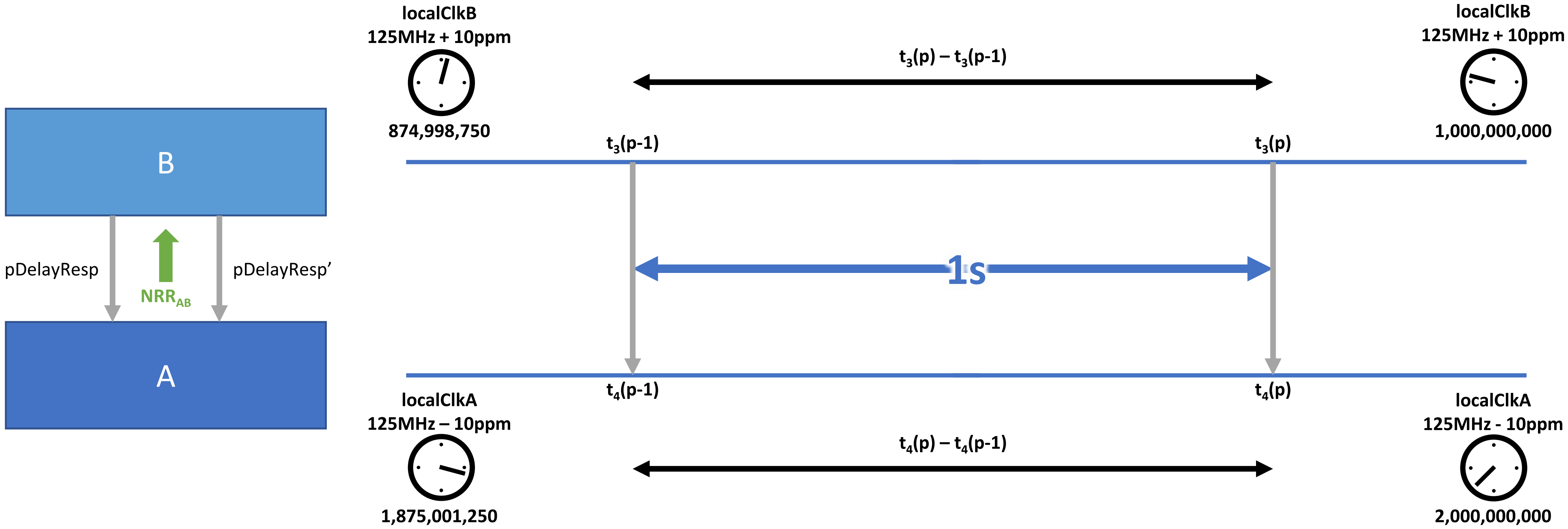
- Very hard to measure clock drift of an individual device...
  ...but we don’t care about the clock drift of an individual device
- We care about Neighbor Rate Ratio and Rate Ratio...and we are constantly measuring both
- We can use theses measurements to track clock drift between two devices and create 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...

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

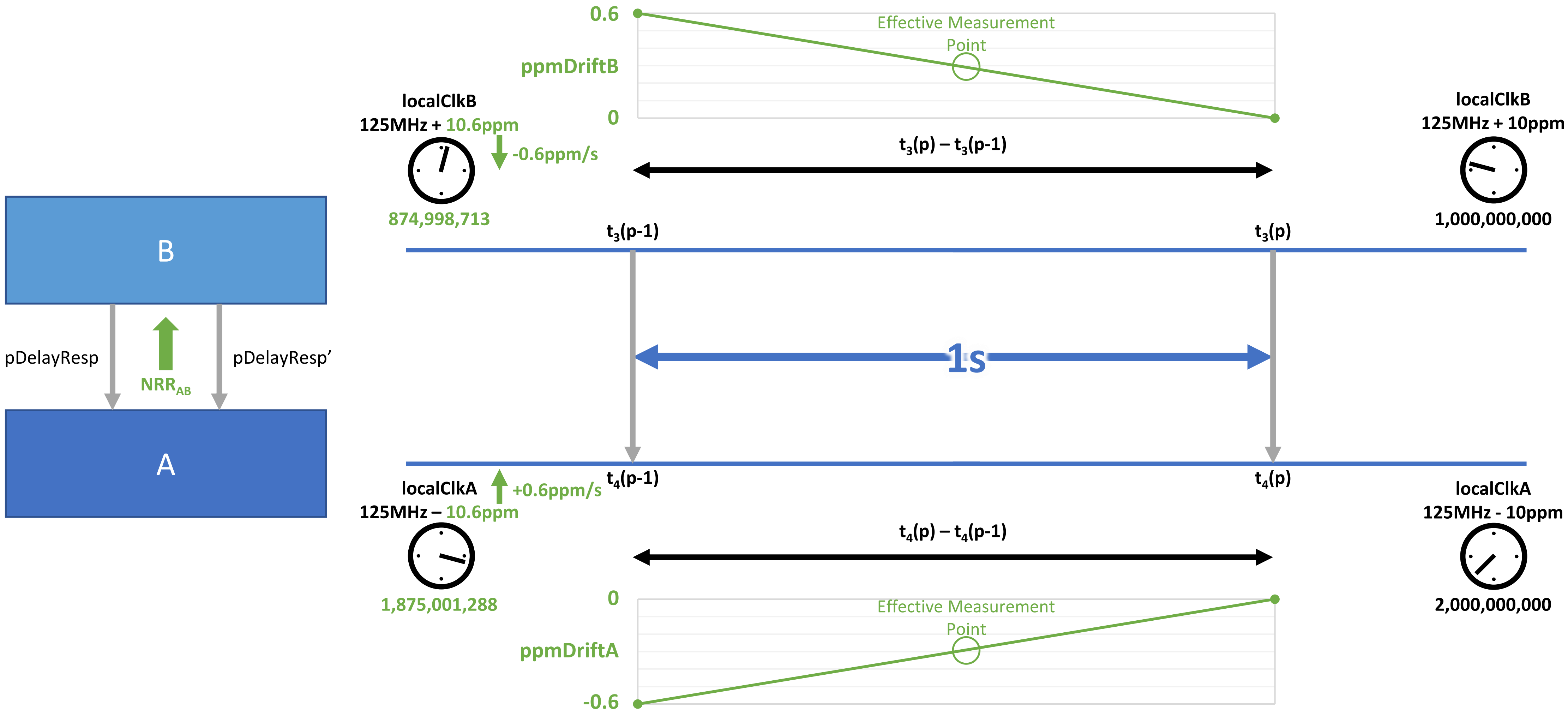
Maximum delay between attempt to measure NRR and using NRR as part of Sync messaging

Maximum delay between attempt to measure NRR and effective measurement point

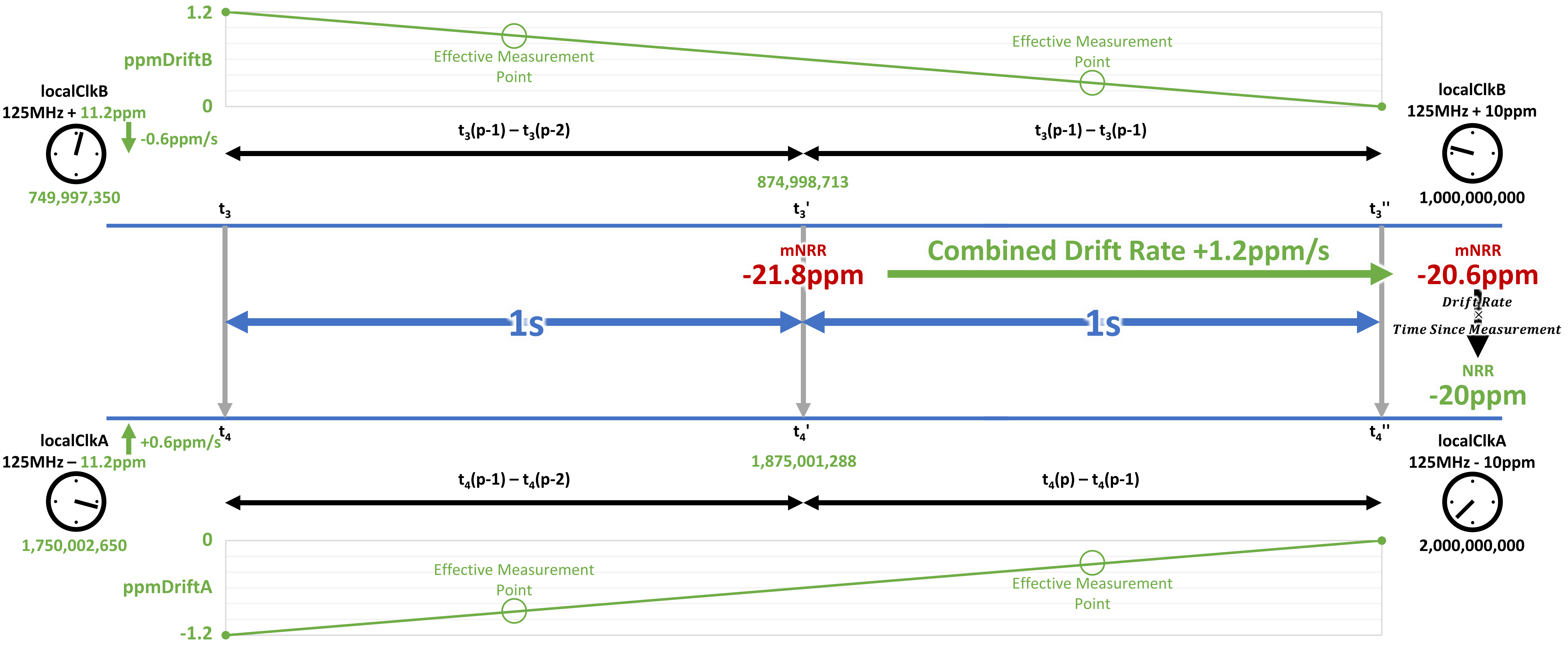

localClkB
localClkB
$t_3(p) – t_3(p-1)$
B
$t_3(p-1)$
$t_3(p)$
pDelayResp
pDelayResp’
NRR_{AB}
1s
A
$t_4(p-1)$
$t_4(p)$
localClkA
localClkA
$t_4(p) – t_4(p-1)$



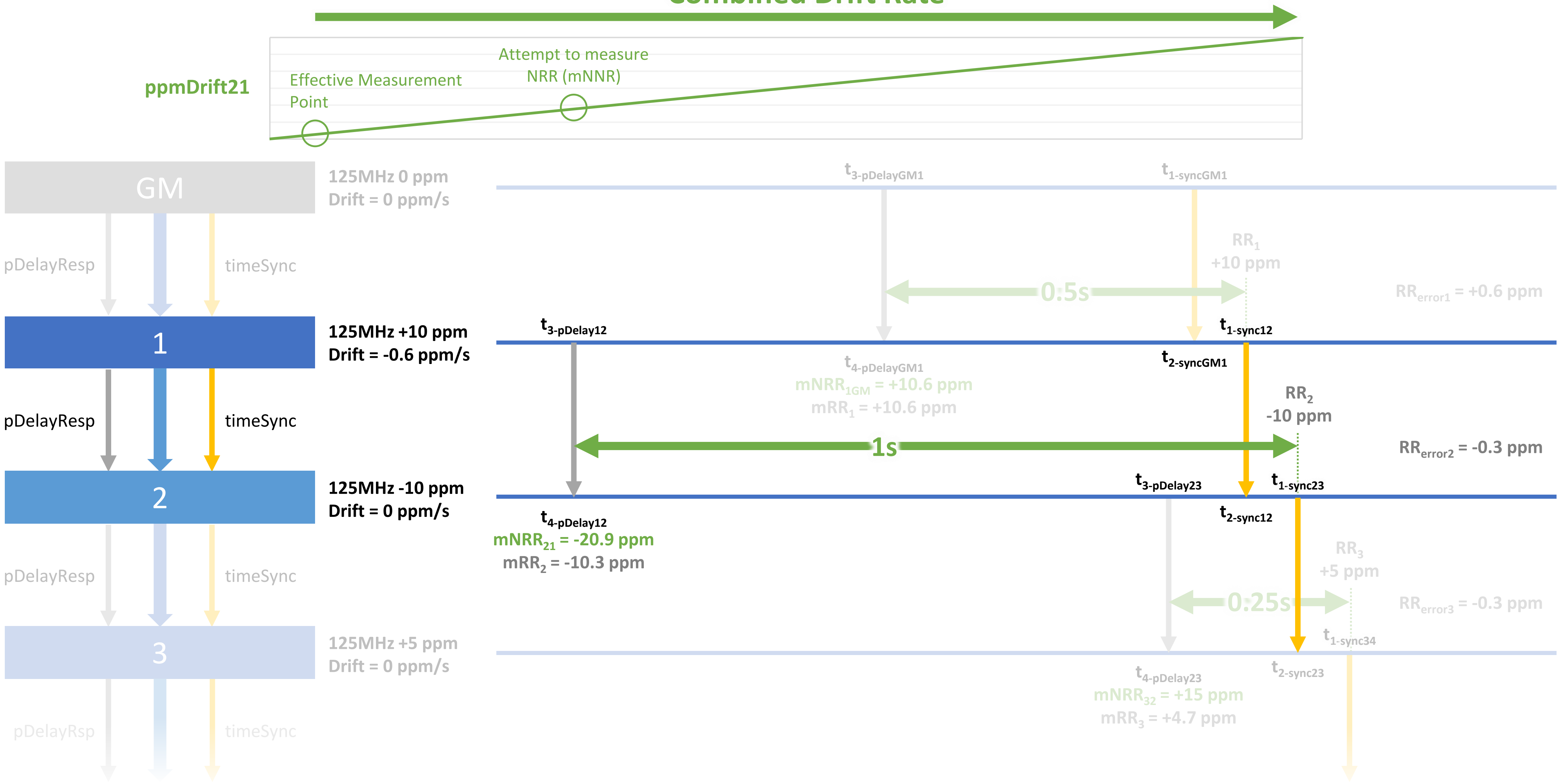
$$NRR_{AB} \text{ @ } t_4' = \frac{(t_4(p) - t_4(p\text{-}1))}{(t_3(p) - t_3(p\text{-}1))} = \frac{124{,}998{,}750}{125{,}001{,}250} \rightarrow -0.002\% = -20 \text{ ppm}$$



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


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
Drift Rate
×
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



ppmDrift21

Effective Measurement Point

Attempt to measure NRR (mNNR)

GM

125MHz 0 ppm
Drift = 0 ppm/s

pDelayResp

timeSync

1

125MHz +10 ppm
Drift = -0.6 ppm/s

pDelayResp

timeSync

2

125MHz -10 ppm
Drift = 0 ppm/s

pDelayResp

timeSync

3

125MHz +5 ppm
Drift = 0 ppm/s

pDelayRsp

timeSync

$t_{3\text{-pDelayGM1}}$

$t_{1\text{-syncGM1}}$

RR_1
+10 ppm

0.5s

RR_{error1} = +0.6 ppm

$t_{3\text{-pDelay12}}$

$t_{1\text{-sync12}}$

$t_{4\text{-pDelayGM1}}$
$mNRR_{1GM}$ = +10.6 ppm
mRR_1 = +10.6 ppm

$t_{2\text{-syncGM1}}$

RR_2
-10 ppm

1s

RR_{error2} = -0.3 ppm

$t_{3\text{-pDelay23}}$

$t_{1\text{-sync23}}$

$t_{4\text{-pDelay12}}$
$mNRR_{21}$ = -20.9 ppm
mRR_2 = -10.3 ppm

$t_{2\text{-sync12}}$

RR_3
+5 ppm

0.25s

RR_{error3} = -0.3 ppm

$t_{1\text{-sync34}}$

$t_{4\text{-pDelay23}}$
$mNRR_{32}$ = +15 ppm
mRR_3 = +4.7 ppm

$t_{2\text{-sync23}}$

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 the 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)...

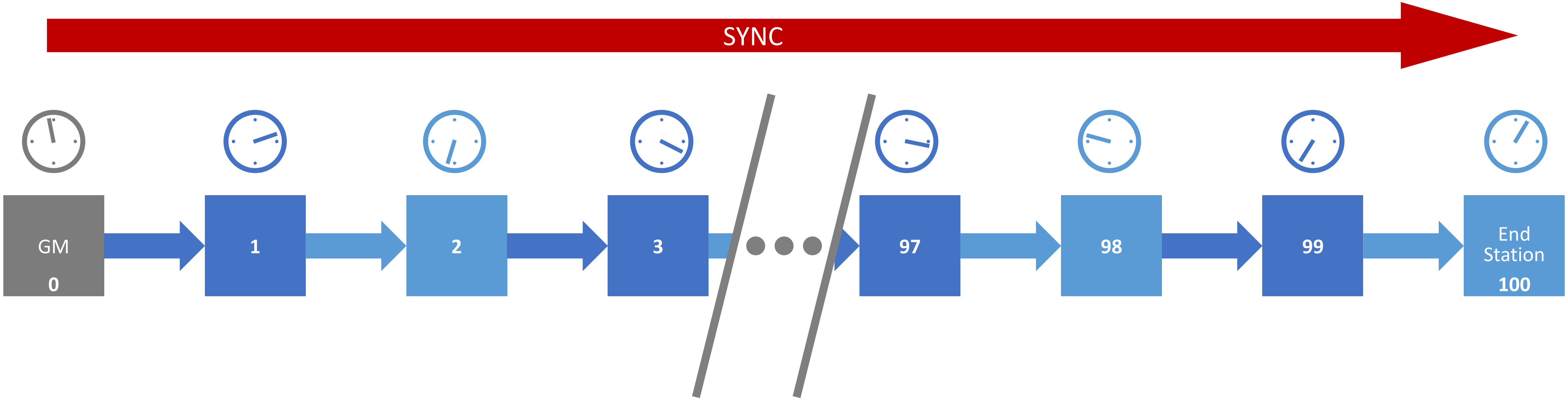
$$NRR_{sync} = mNRR + NRR_{driftRate}\left(delay_{mNRR\_sync} + \frac{pDelayInterval \times N}{2}\right)$$

- 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?

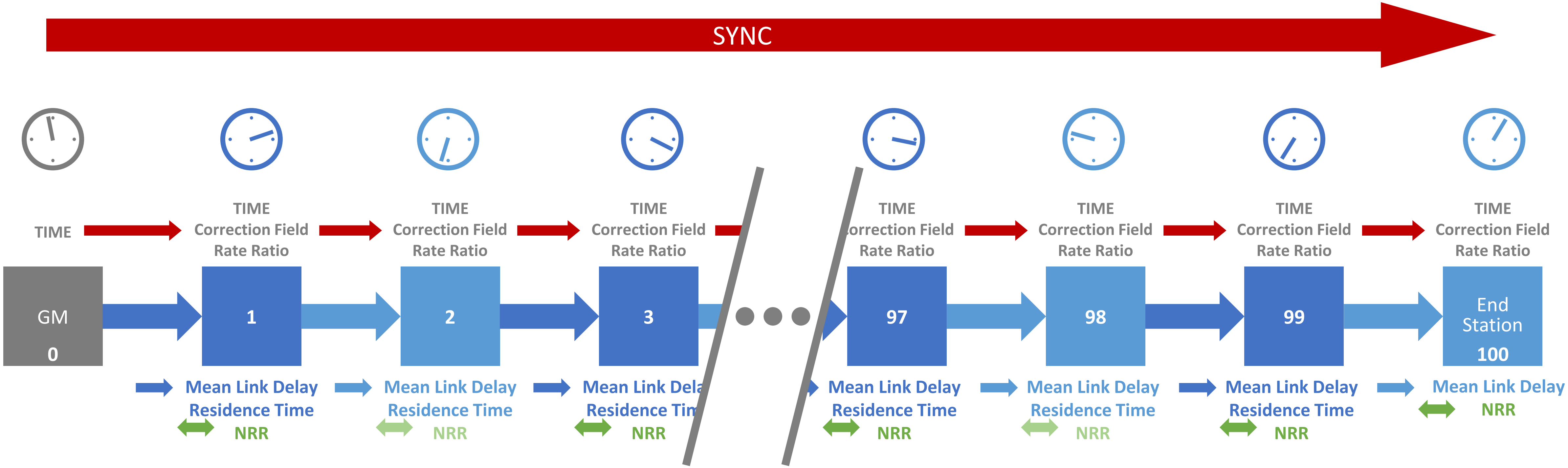
Time Sync



How long did that message take to get to me?

How much faster or slower is my clock running than the Grand Master’s clock?

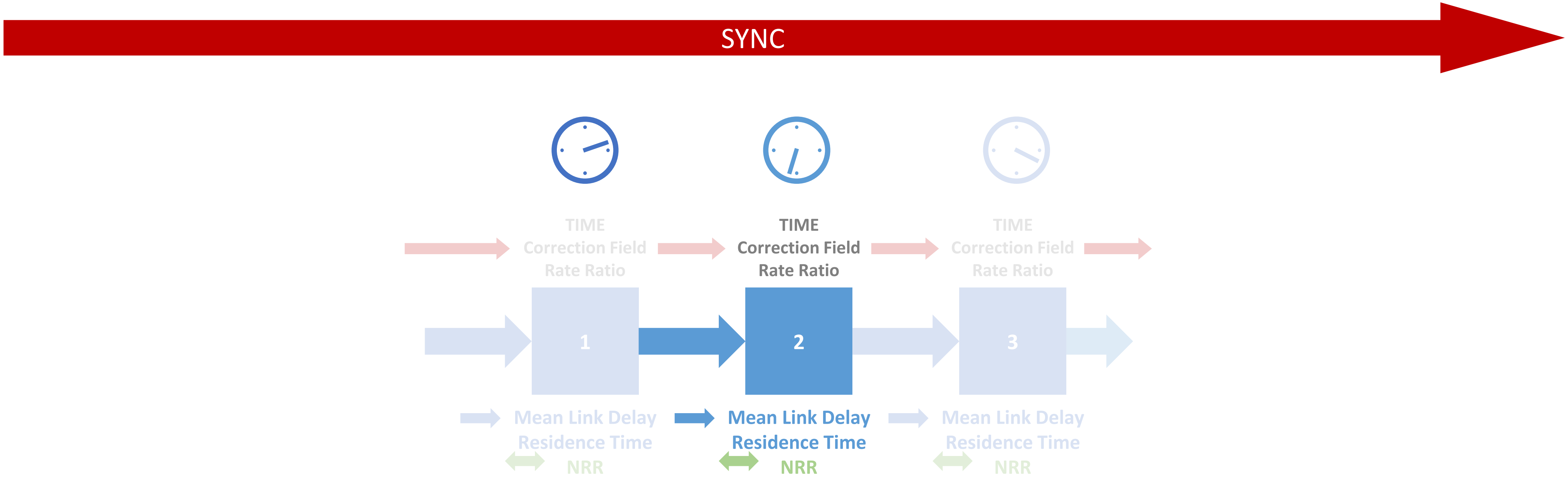
Time Sync



Correction Field = Sum of all Path Delays (pDelay) & Residence Times

Rate Ratio (RR of Local Clock to GM) = Sum of all Neighbor Rate Ratios (if NRR in ppm)

Time Sync – Normative Requirements



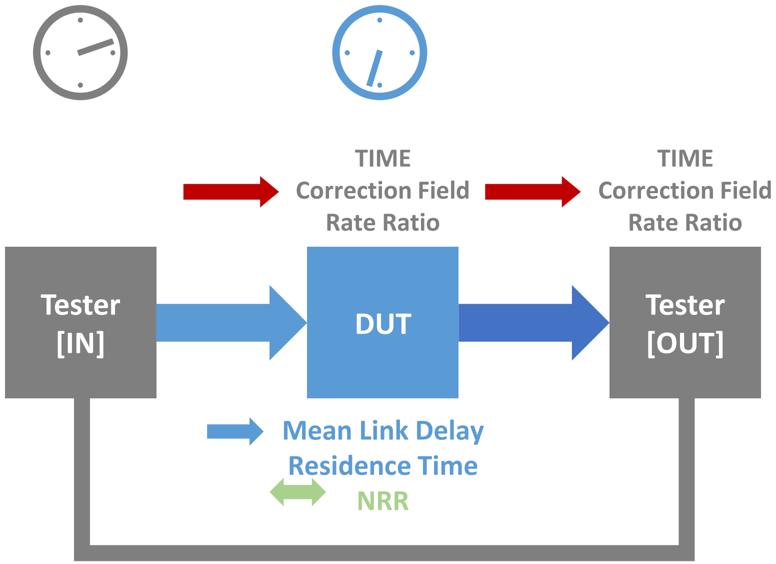
Each node must not add more error into the Correction Field and Rate Ratio of a Sync message than the system can handle.

Error is added into Rate Ratio via NRR, which is affected by Clock Drift of the upstream device.

What should go into the spec?

- **Normative**: clock stability (ppm/s)
  - Within device’s operating parameters. Including not just temperature, but rate of change of temperature.
- **Normative**: accuracy of processing Correction Field. Includes efficiency of algorithms.
  - While upstream clock drifts.
- **Normative**: accuracy of processing Rate Ratio field, i.e. calculating NRR. Includes efficiency of algorithms.
  - While upstream clock drifts.
- **Informative**: how these requirements can be met using example algorithms.
  - Do no proscribe algorithms. Implementation is hard to verify; we ultimately don’t care about the specific algorithm but rather then end effect; and it may be an area for innovation and differentiation.

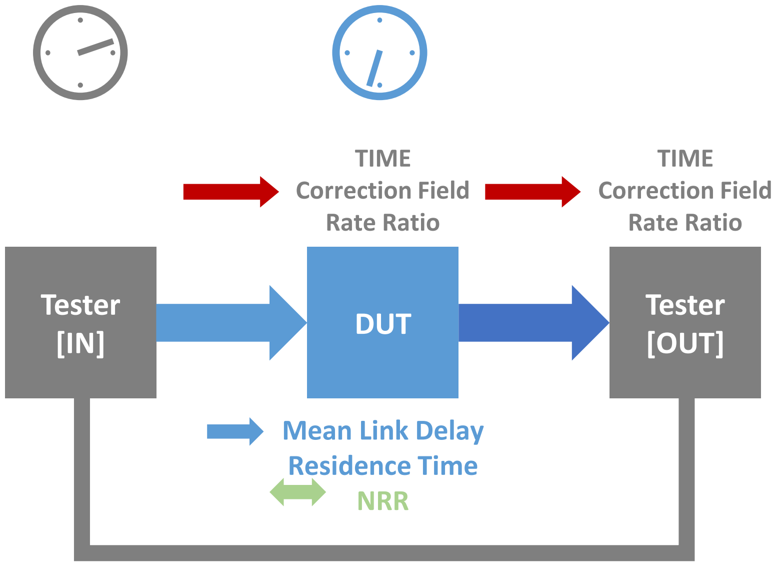
Testing Clock Stability



Stable Tester Clock. Vary temperature of DUT according to declared capabilities.

DUT’s output Rate Ratio must remain within normative limits.

Testing Accuracy of Correction Field and Rate Ratio Processing



Testers measure Mean Link Delay & Residence Time;
Tester[IN] alters its clock (for DUT’s NRR measurement) over time
according to worst case normative Clock Drift limits.

DUT’s output Correction Field & Rate Ratio must remain within Normative Limits.

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 (detailed below) has already been run for one replication
- B – E as detailed below.
  - Initially: single replication for each
  - If there is time: multiple replications for A to E
- Additional simulations to match optimised settings from further Monte Carlo analysis

Changes from Previous Simulations

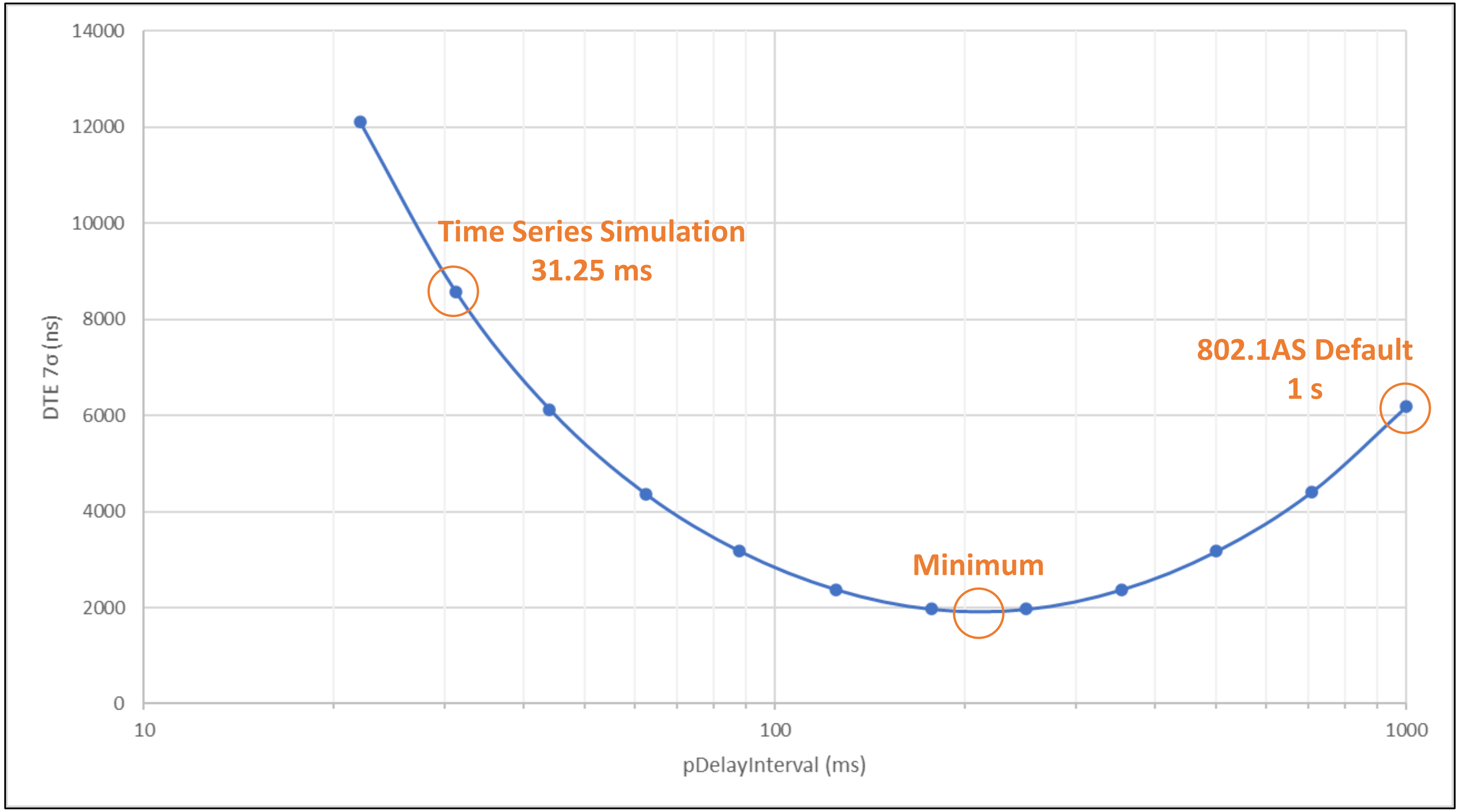
- Change Dynamic Timestamp Error from...
  +8ns or -8ns with equal chance of either
  ...to...
  ±4ns (uniform linear distribution)
- Change Residence Time and pDelay Turnaround Time from 1ms to 10ms
- Initially remove “take the median of past M calculations” step from mNRR Smoothing.
  - Retain the 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

		Errors			Parameter			Correction Factors	
Experiment	**Reason**	**Clock Drift Model – 40°C ↔ +85°C Hold for 5min at Each** (Each node's position in cycle distributed at random across 100% of Cycle)	**Timestamp Granularity (±ns)**	**Dynamic Timestamp Error (±ns)**	**pDelay Interval (ms)**	**Residence Time (ms)**	**pDelay Turnaround Time (ms)**	**Mean Link Delay Averaging**	**mNRR Smoothing Factor N**
A	Baseline with previous assumptions (Simulation 1 from previous results*)	Ramp Rate 25°C / min Cycle of 20 mins	4	8 (0.5 probability either way)	31.25	1	1	Off	1
B	Verify optimised pDelayInterval		4	4	1000	10	10		
C					250	10	10		
D					31.25	10	10		
E	Verify effect of reduced Timestamp Error (reduced DTE when pDelay Interval is low, i.e. 31.25ms)		2	2	31.25	10	10		
F	Verify effect of reduced Clock Drift (reduced DTE when pDelay Interval is high, i.e. 1000ms)	Ramp Rate 12.5°C / min Cycle of 30 mins	4	4	1000	10	10		

Timestamp Granularity and Dynamic Timestamp Error are uniform distributions
Sync Interval: 125ms
* 2021-04 - 60802-garner-new-simulation-results-new-freq-stab-model-0421-v02.pdf

B, C & D: Validate *pDelayInterval* Sensitivity Analysis



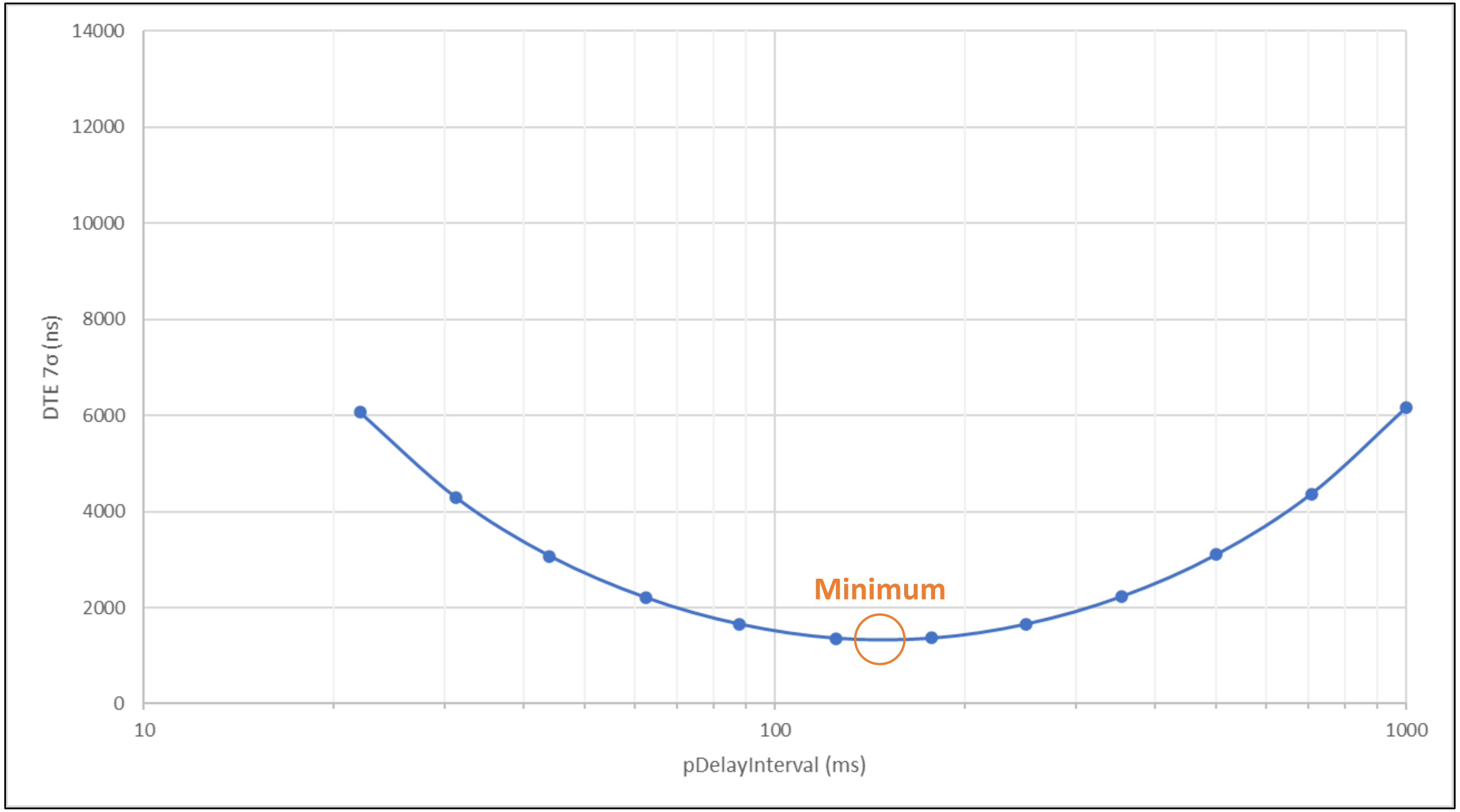
Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	+0.6	ppm/s
Clock Drift (non-GM)	0.6	±ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay Interval	Variable	ms
pDelay Response Time	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.

E: Validate *pDelayInterval* Sensitivity Analysis with Lower Timestamp Error

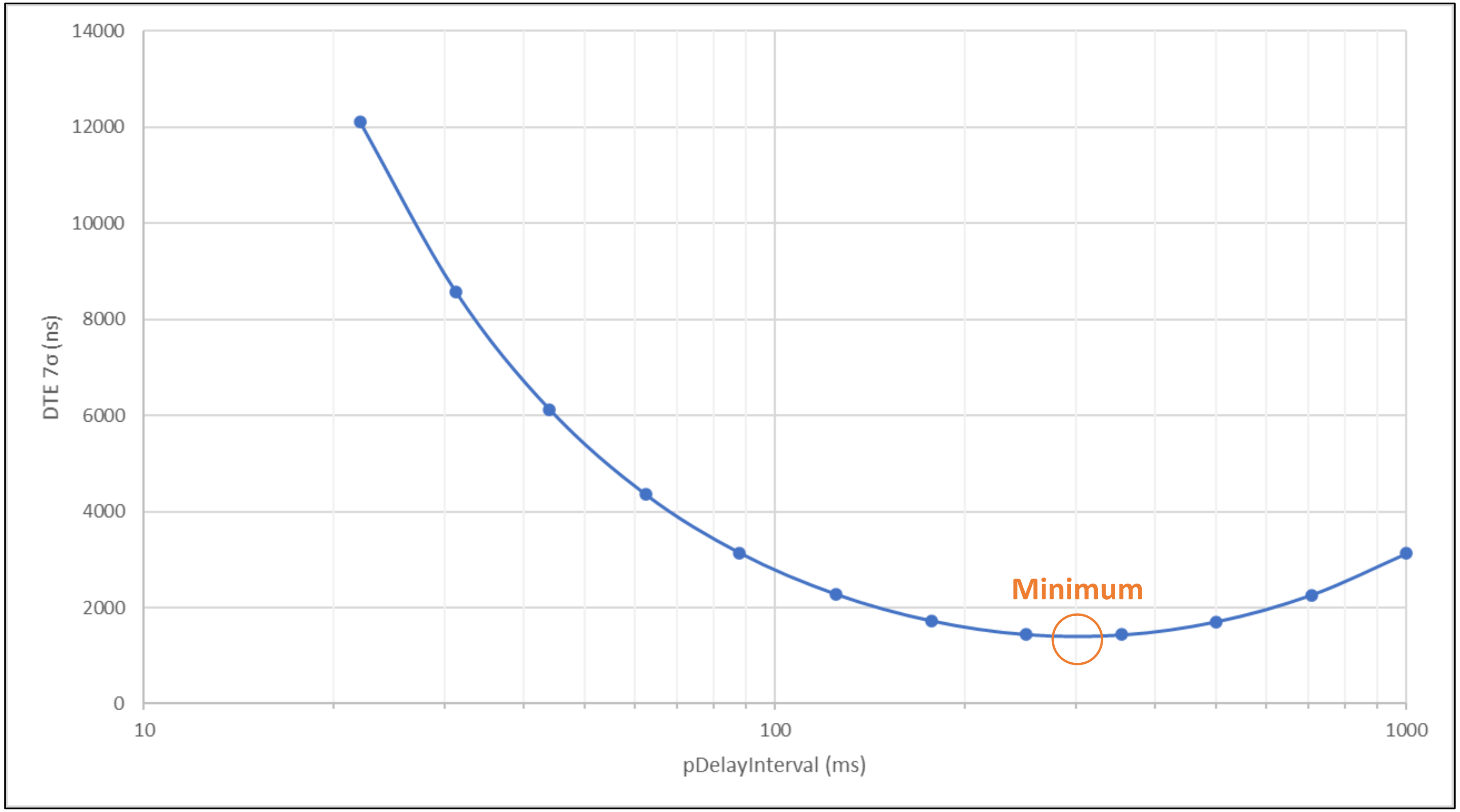


Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	+0.6	ppm/s
Clock Drift (non-GM)	0.6	±ppm/s
Timestamp Granularity TX	2	±ns
Timestamp Granularity RX	2	±ns
Dynamic Time Stamp Error TX	2	±ns
Dynamic Time Stamp Error RX	2	±ns
Input Parameters		
pDelay Interval	Variable	ms
pDelay Response Time	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: 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	

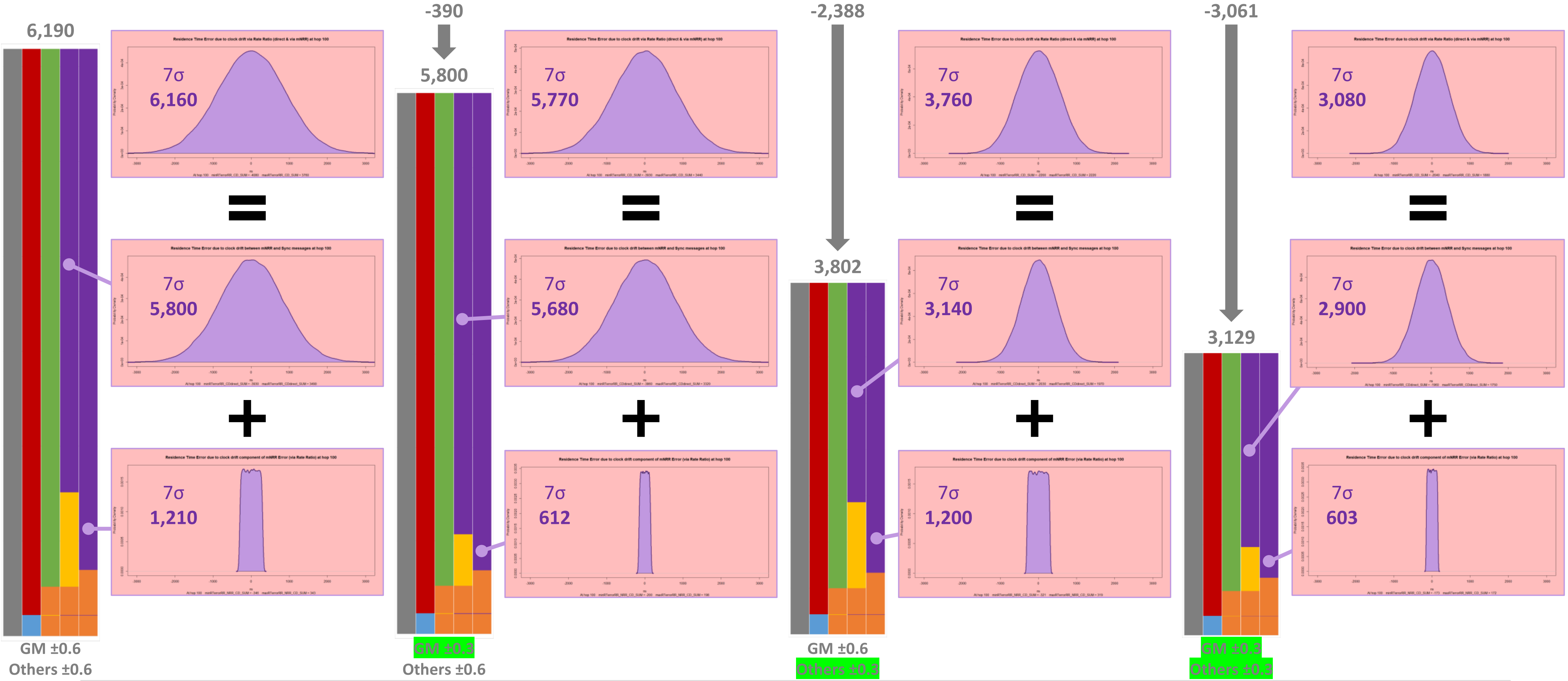
Minimum is at approx 300ms…

…for this set of parameters.

Thank you!

Back Up Material

Clock Drift Sensitivity



6,190

7σ 6,160 = 7σ 5,800 + 7σ 1,210

GM ±0.6
Others ±0.6

-390

5,800

7σ 5,770 = 7σ 5,680 + 7σ 612

GM ±0.3
Others ±0.6

-2,388

3,802

7σ 3,760 = 7σ 3,140 + 7σ 1,200

GM ±0.6
Others ±0.3

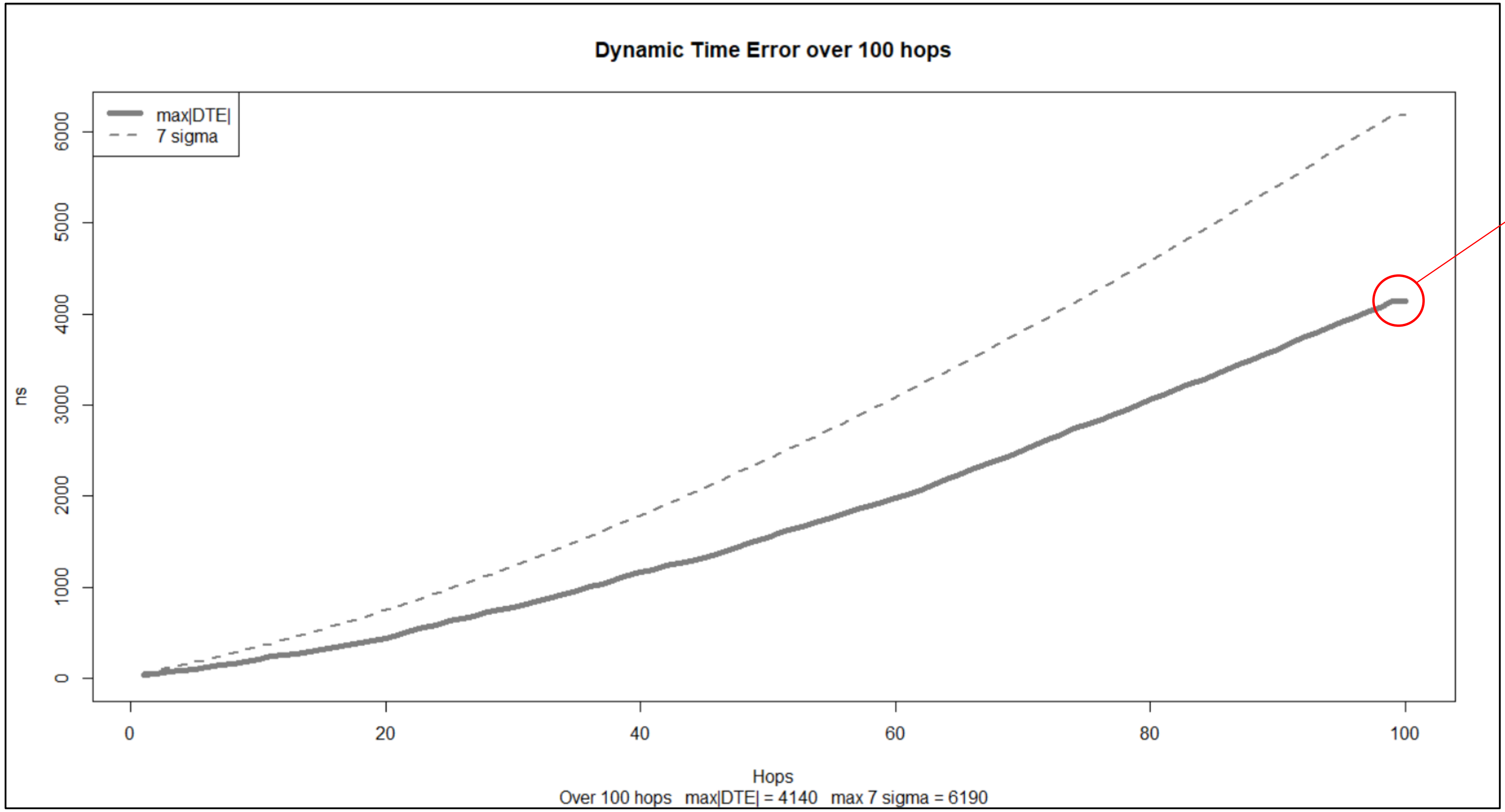
-3,061

3,129

7σ 3,080 = 7σ 2,900 + 7σ 603

GM ±0.3
Others ±0.3

DTE Over 100 Hops – Impact of End Station Error



Addition of End Station Error to analysis is expected to remove the “flattening out” of the DTE curve during for the final hop...at least for pDelay Intervals where errors due to clock drift dominate.