



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

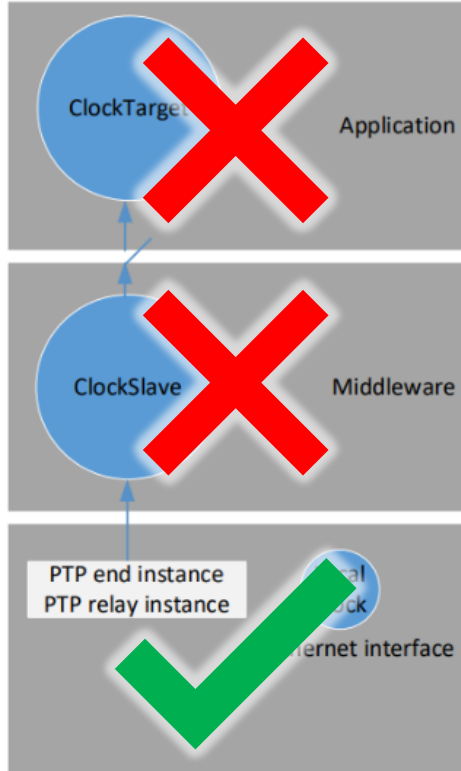
- A second aspect of mNRR Smoothing was judged to be ineffective: use of a median of past NRR calculations.
 - This is **not** the case.
- The plan after the last presentation was for a small group to collaborate on deciding how to validate the Monte Carlo Analysis against the Time Series Simulation
 - 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

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 every 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>

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.
 - $20 \text{ ppm} + 30 \text{ ppm} = 50 \text{ ppm}$
 - $20 \text{ ppm} \times 30 \text{ ppm} = 50.0006 \text{ ppm}$
- The trade-off is worth it for reduced runtime.
 - Multiplication is more expensive, computationally, than addition...especially when using double precision floating point numbers.

Planned Improvement

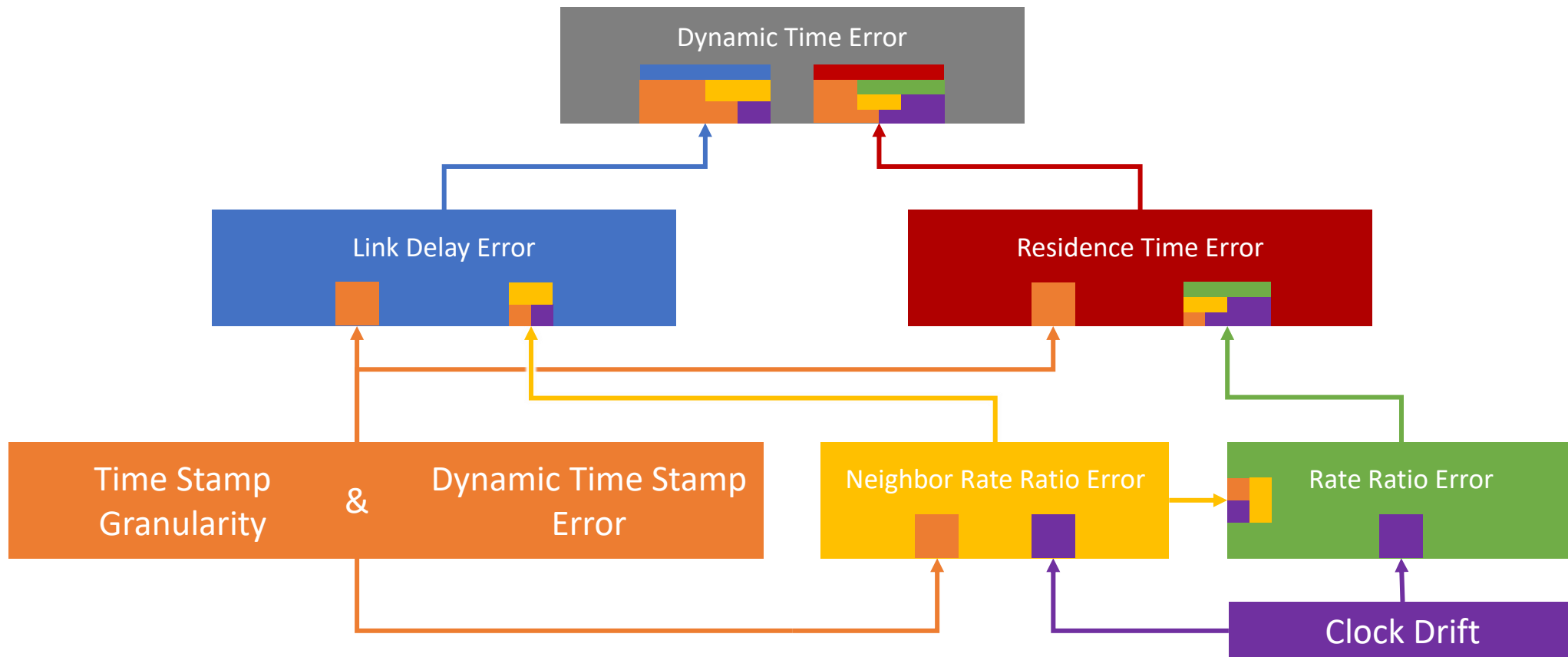
- Analysis in this and previous presentations modelled DTE at the point the Sync message arrived at the End Station (hop 100).
- There is additional error as the GM and Local Clock drift with respect to each other prior to arrival of the next Sync message.
 - Modelled as follows (when hop = hops, i.e.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

$$DTE_{endStationError} = \frac{syncInterval \times RRe_{rror}(hop)}{1000}$$

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

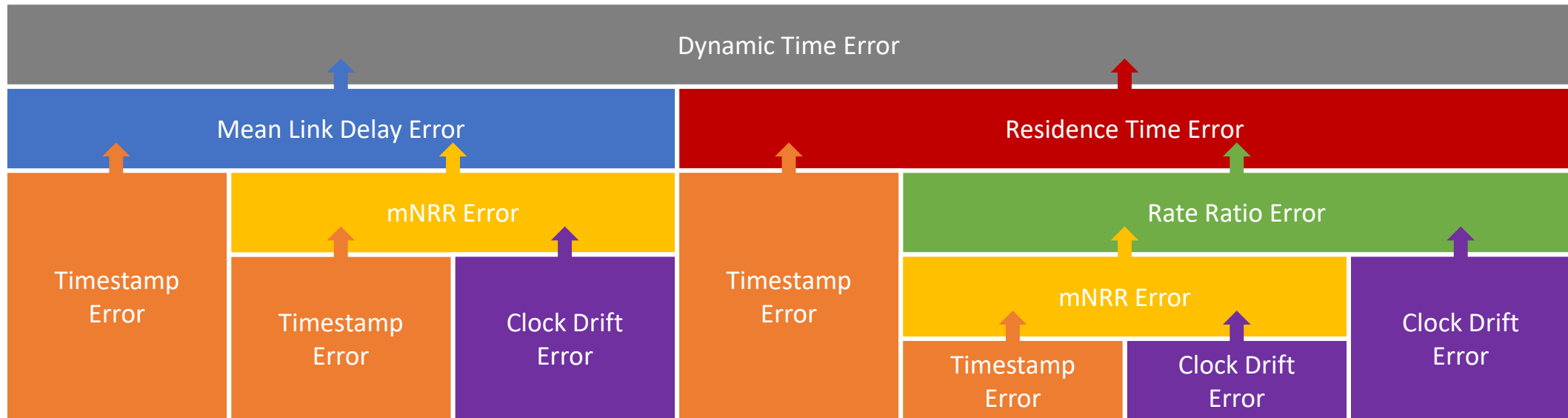
Time Sync – How Errors Add Up



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

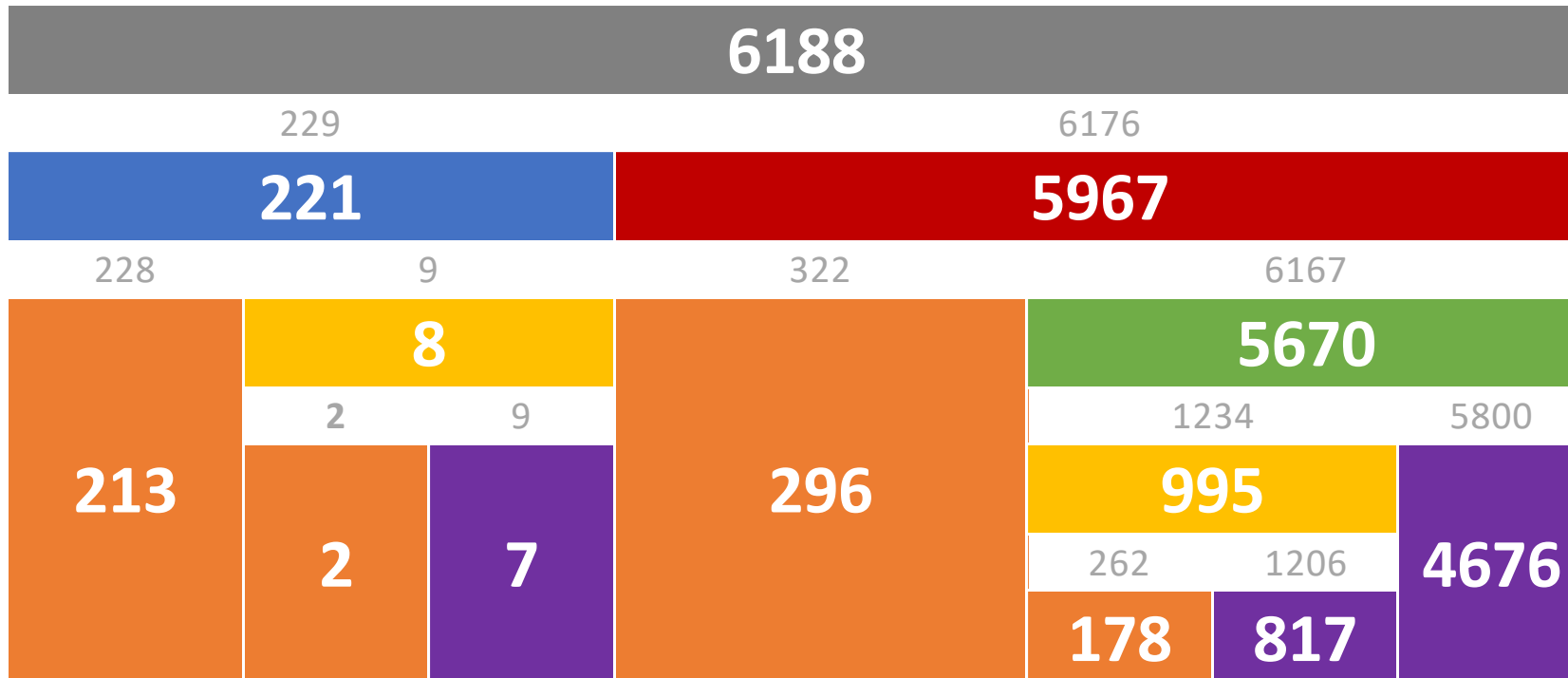
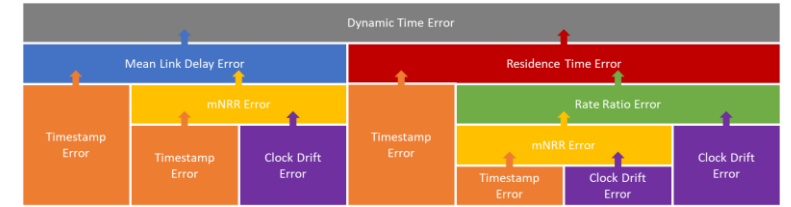
Graphical Representation of Error Accumulation

- Each error breaks down into two parts



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

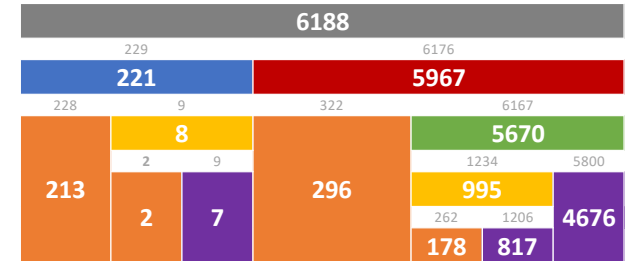
Graphical Representation of Error Accumulation



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	

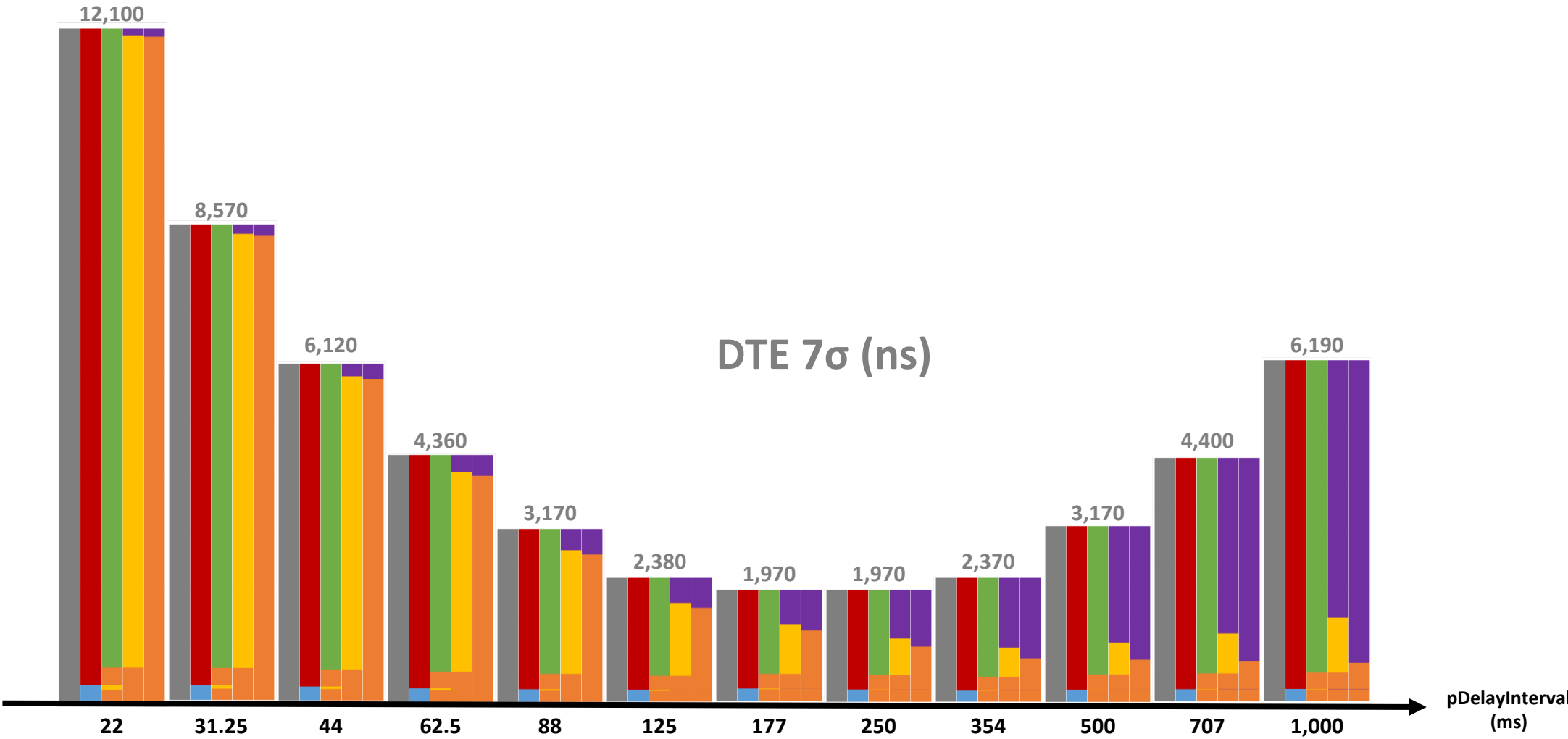
7σ values

Graphical Representation of Error Accumulation

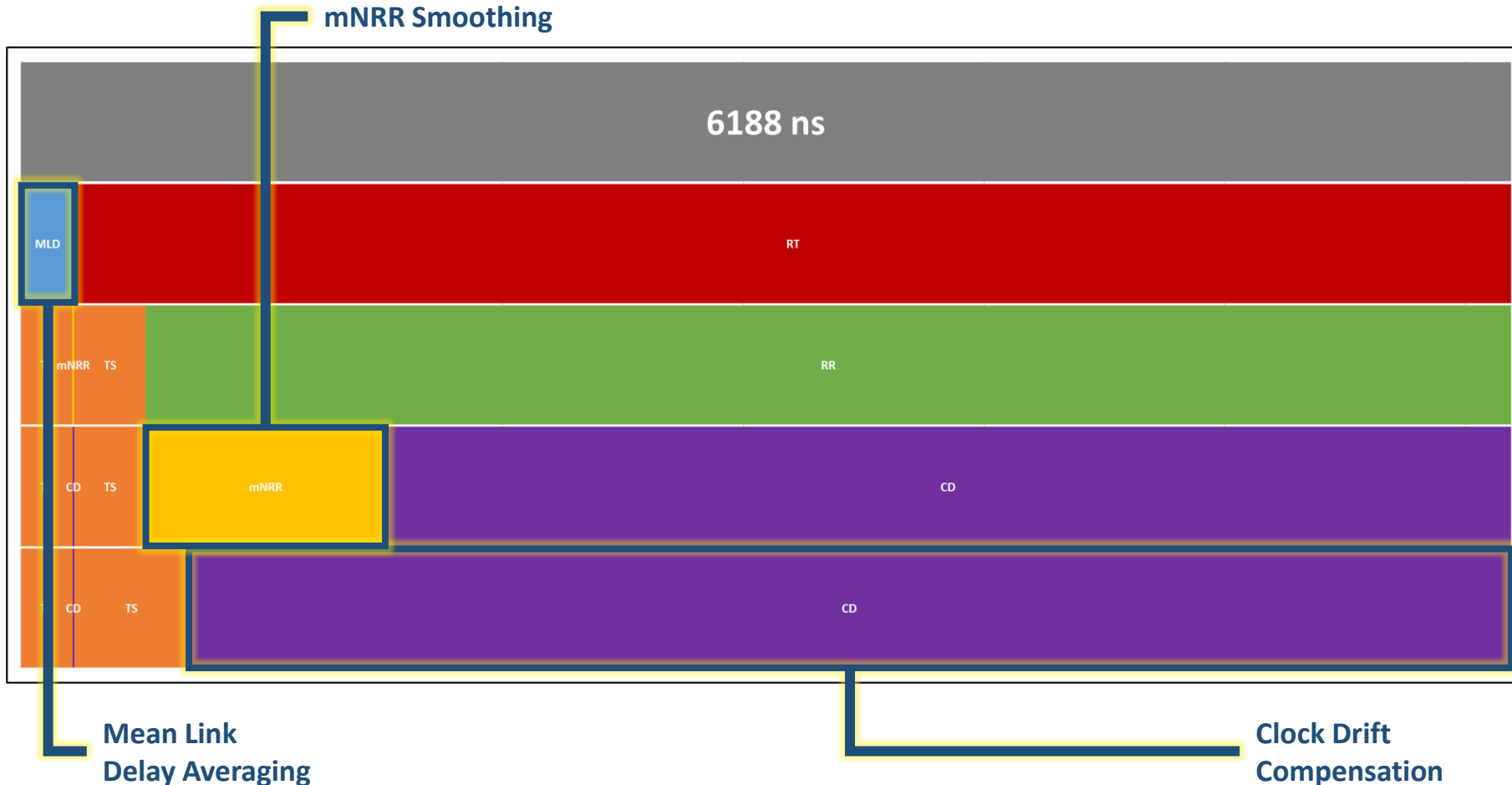


Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	-0.6	ppm/s
Clock Drift Max (non-GM)	+0.6	ppm/s
Clock Drift Min (non-GM)	-0.6	ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay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

pDelayInterval Sensitivity Analysis

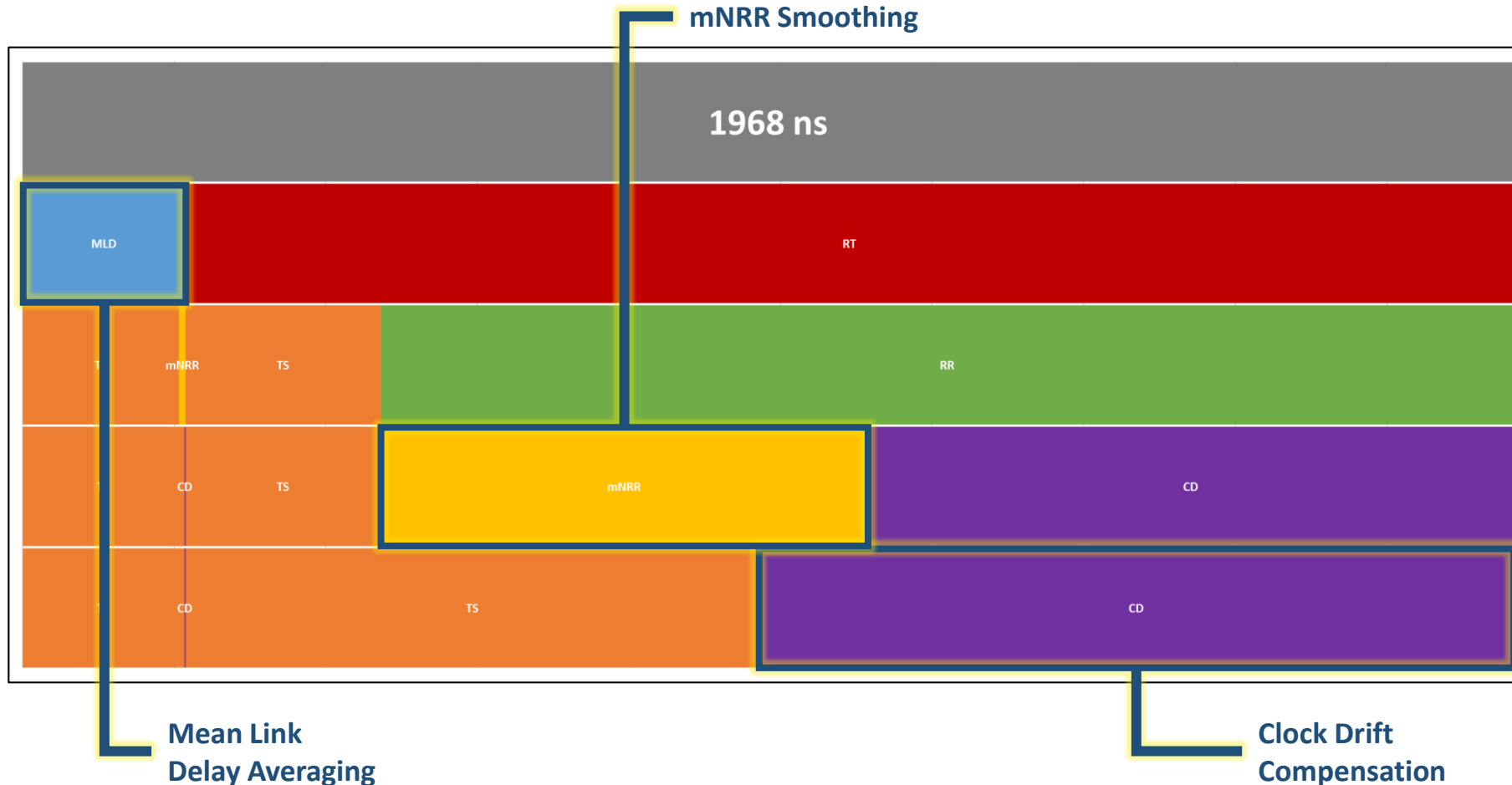


Potential Algorithm Impact – pDelay Interval 1000ms



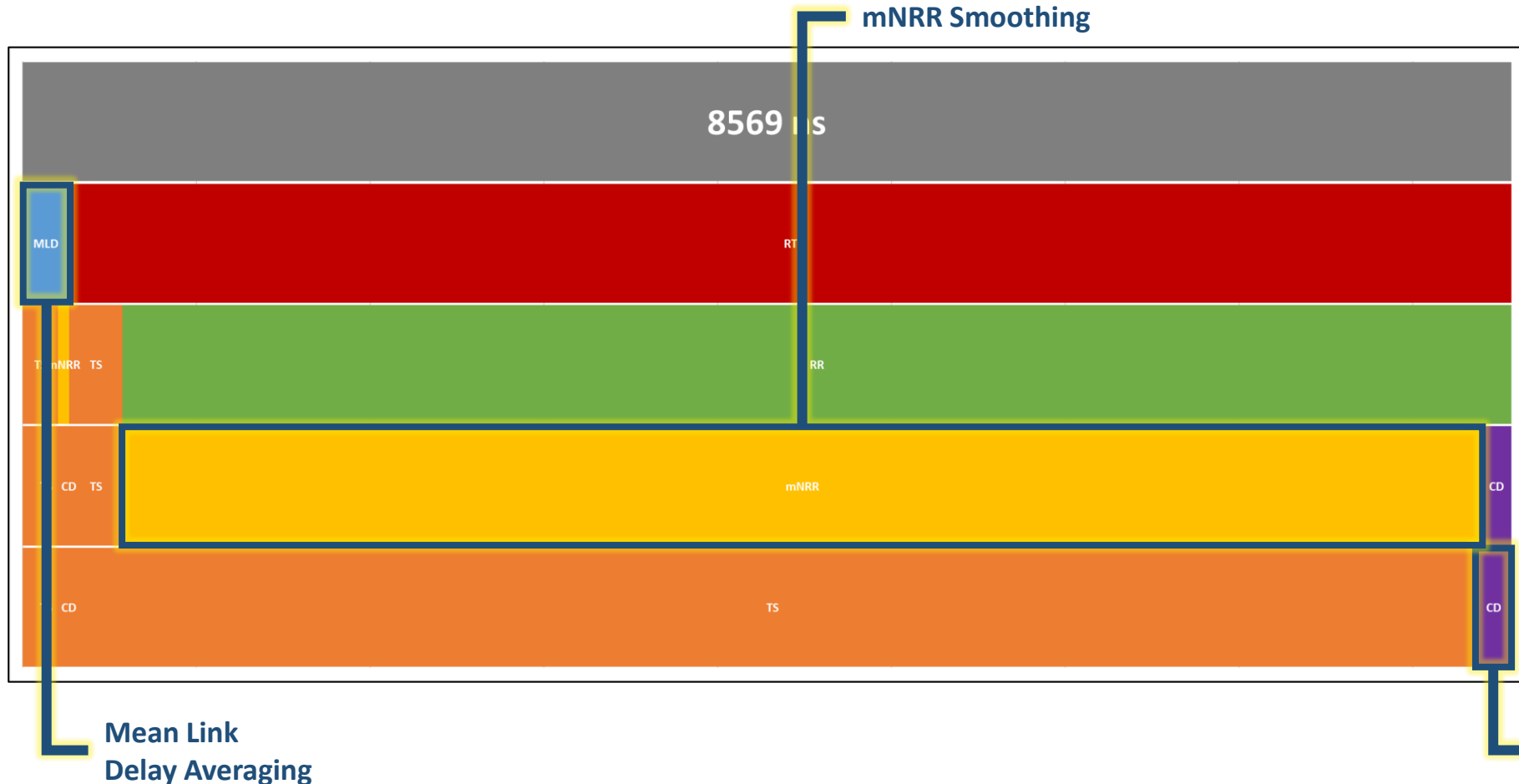
Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	-0.6	ppm/s
Clock Drift Max (non-GM)	+0.6	ppm/s
Clock Drift Min (non-GM)	-0.6	ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay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

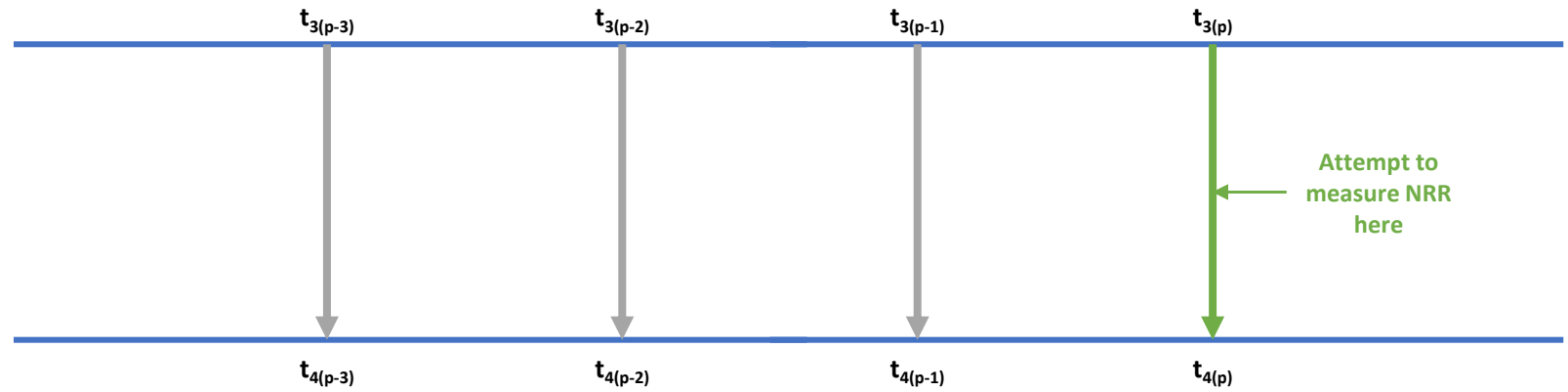
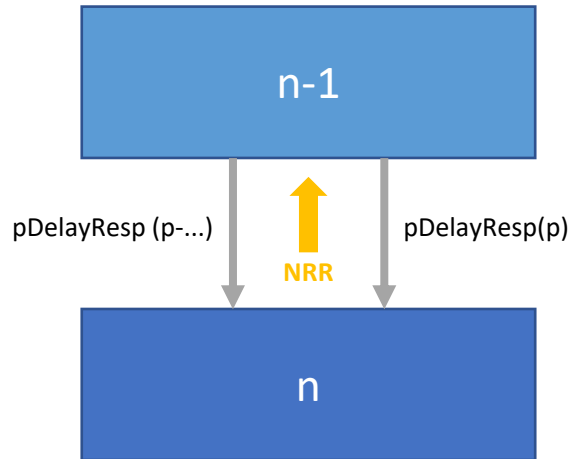


Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	-0.6	ppm/s
Clock Drift Max (non-GM)	+0.6	ppm/s
Clock Drift Min (non-GM)	-0.6	ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay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

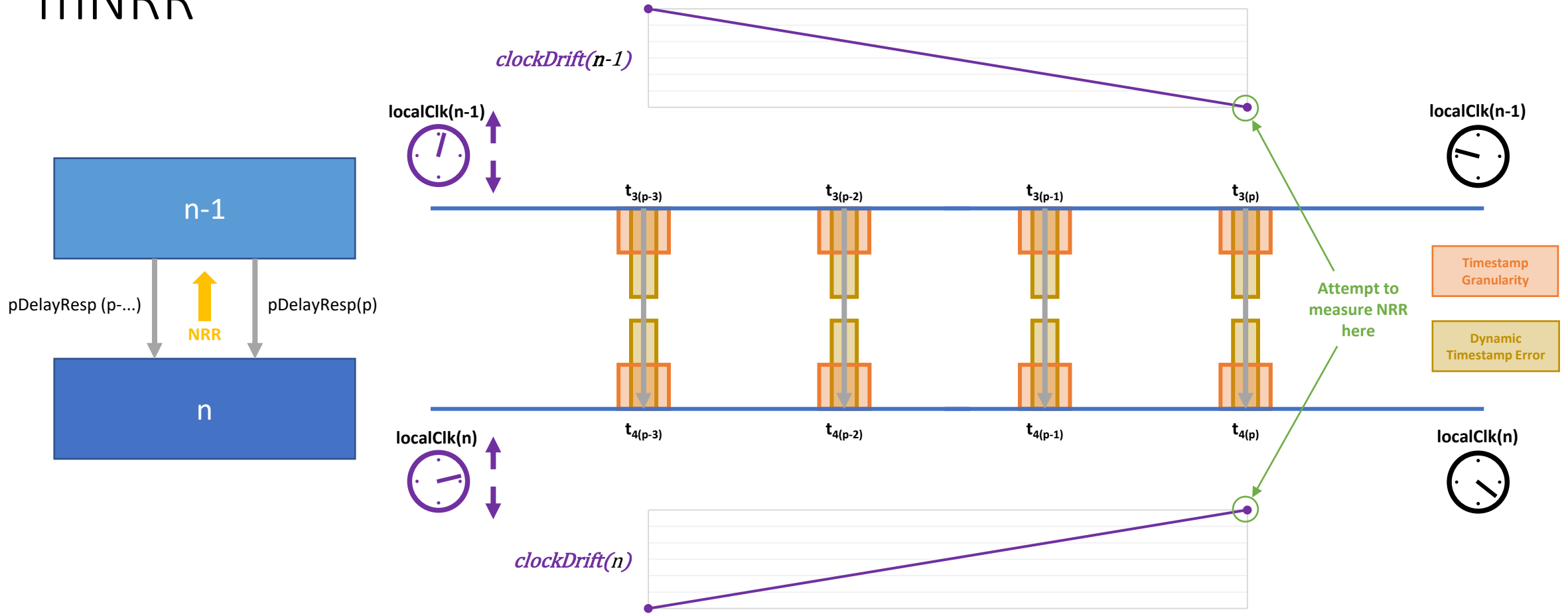
Neighbor Rate Ratio “Smoothing”

Techniques for minimising $mNRR_{\text{error}}$

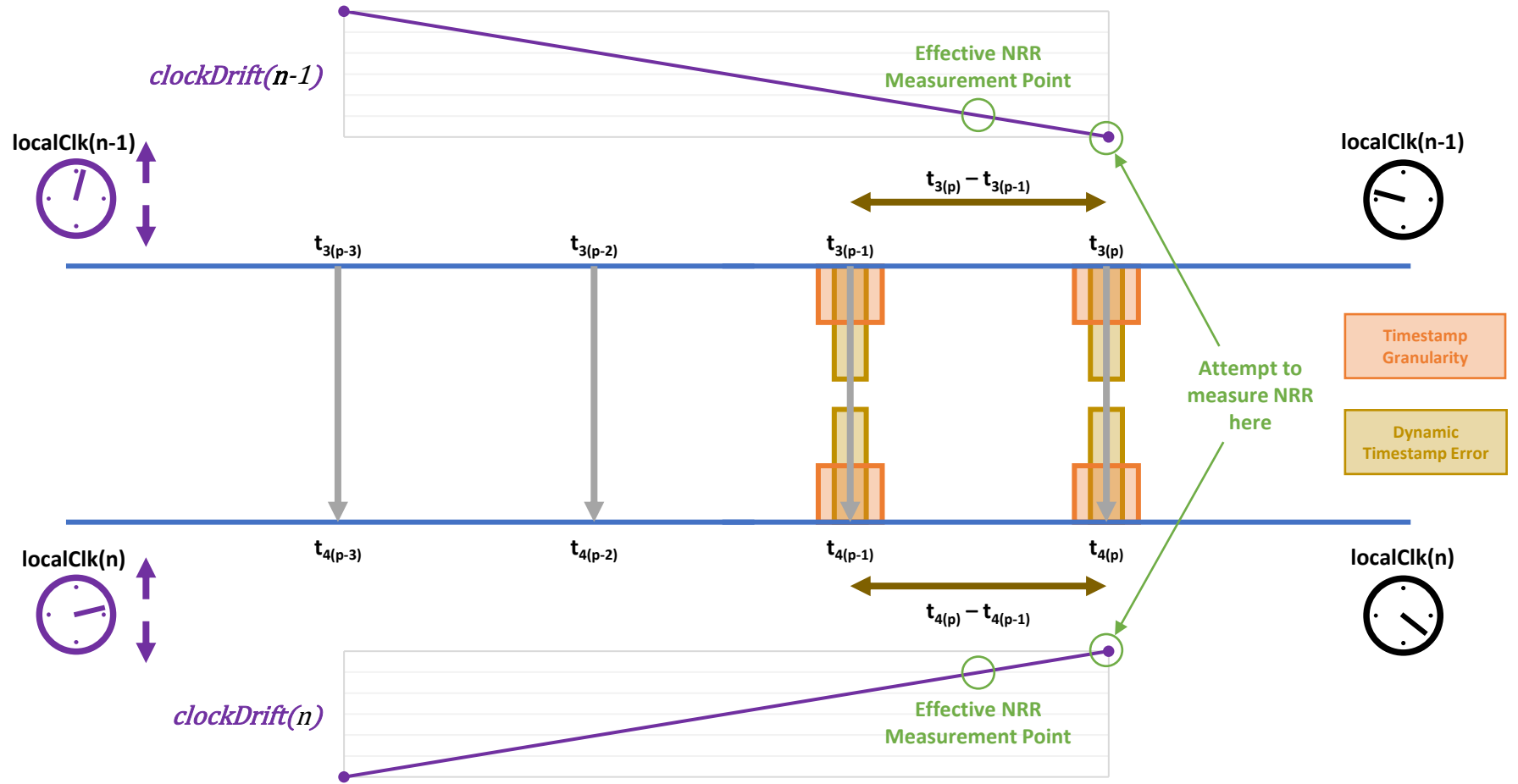
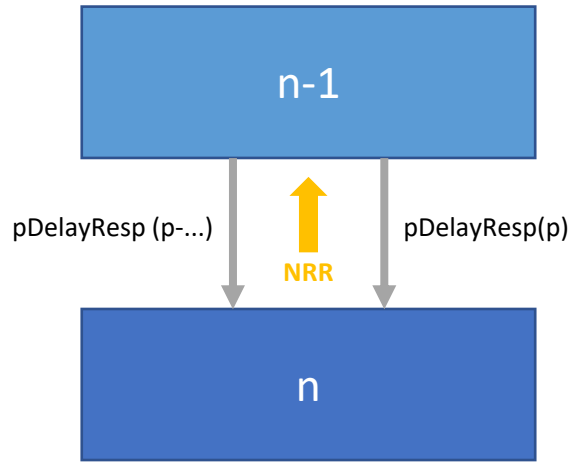
mNRR



mNRR



mNRR_{error}



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

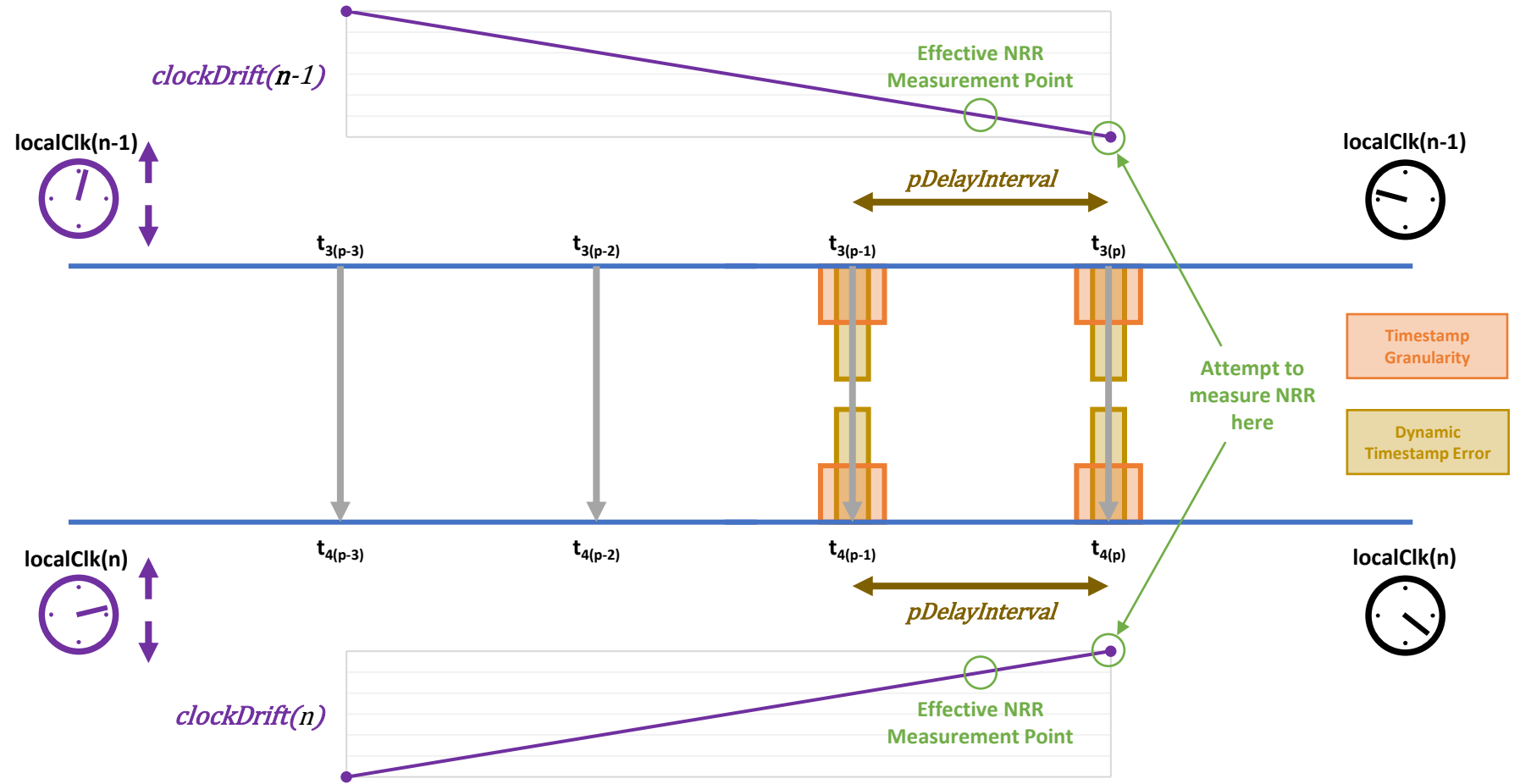
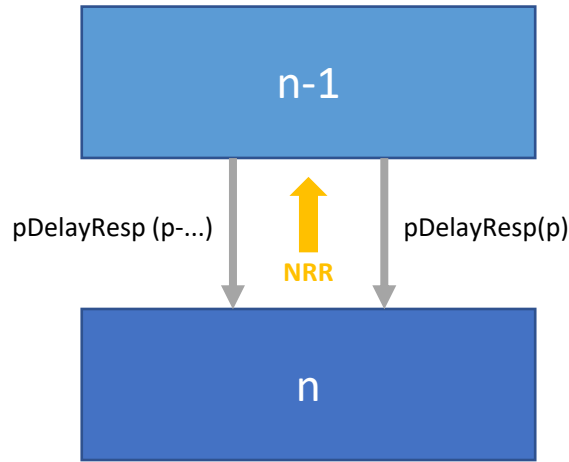
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{(t_{4(p)} - t_{4(p-1)}) \times 10^6} \right)$$

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

ppm

ppm

mNRR_{error}



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

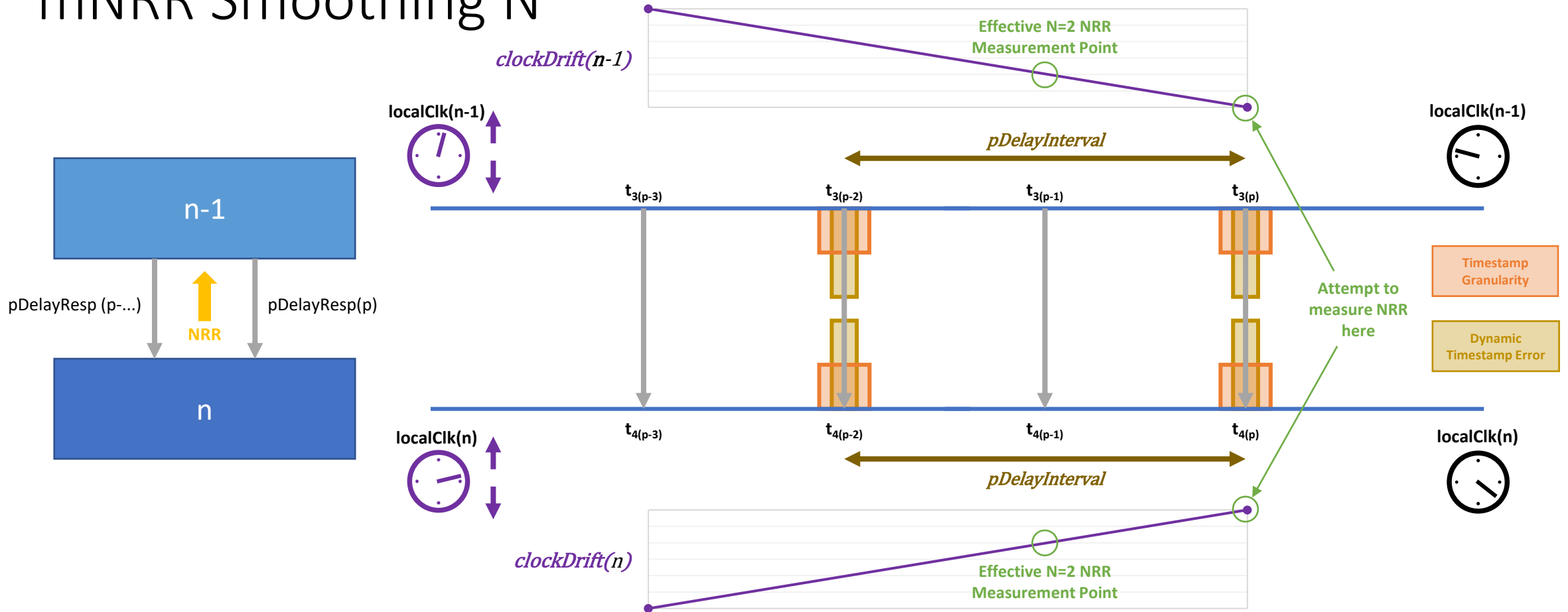
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval} \right)$$

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

ppm

ppm

mNRR Smoothing N



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

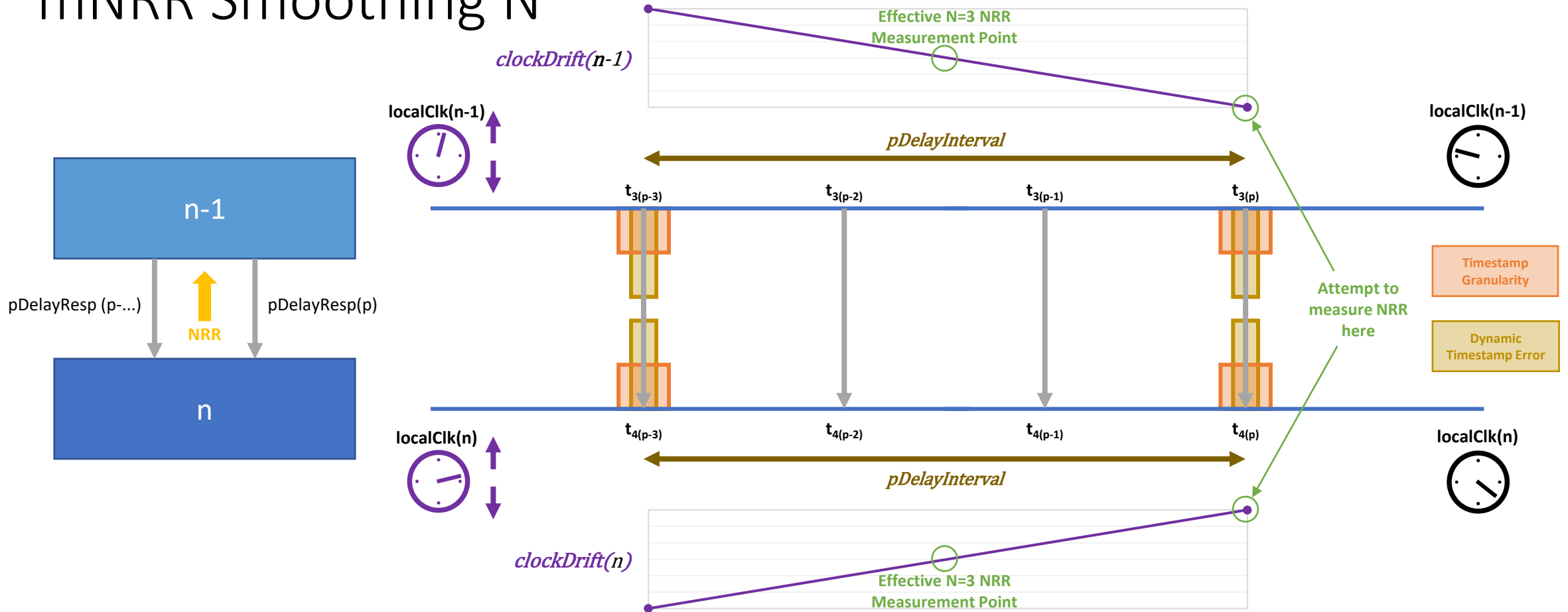
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval \times 2} \right)$$

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

ppm

ppm

mNRR Smoothing N



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

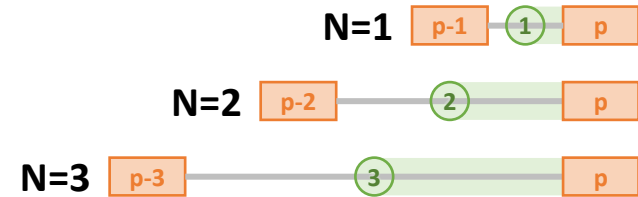
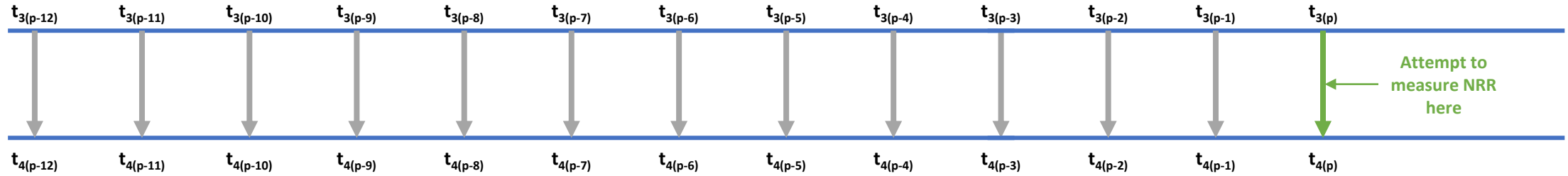
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval \times 3} \right)$$

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

ppm

ppm

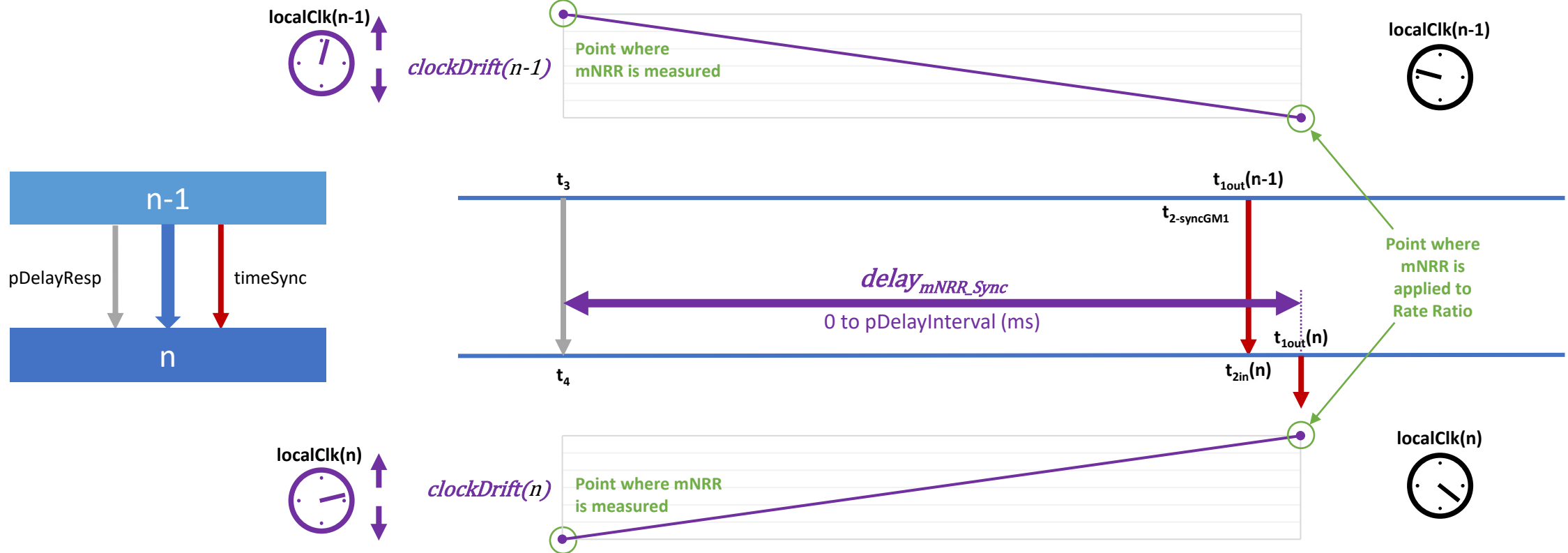
mNRR Smoothing N



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

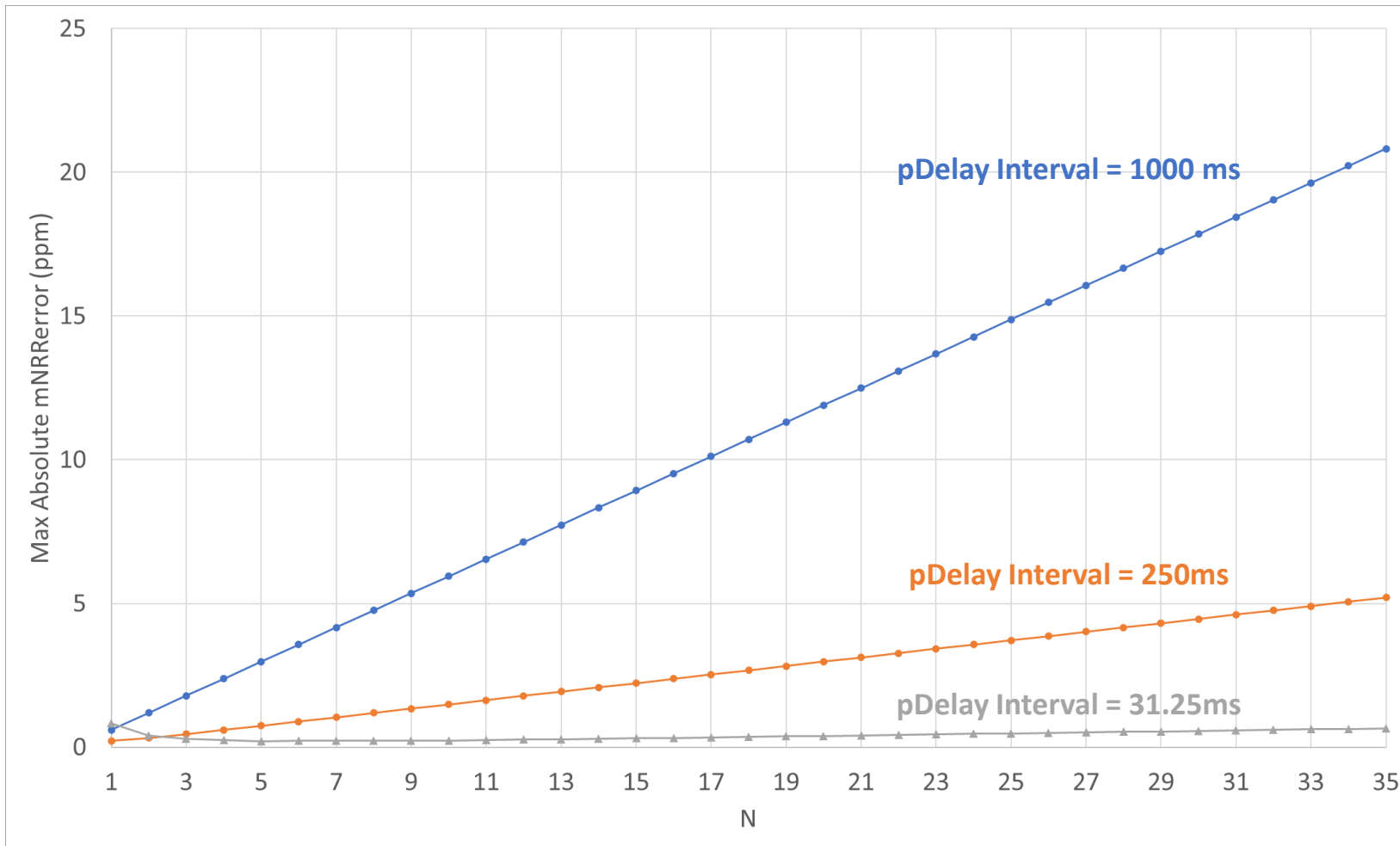
$RR_{errorCD}$



$$RR_{errorCD}(n) = \frac{delay_{mNRR_Sync}}{10^3} (clockDrift(n-1) - clockDrift(n))$$

ppm

Optimal Value of N

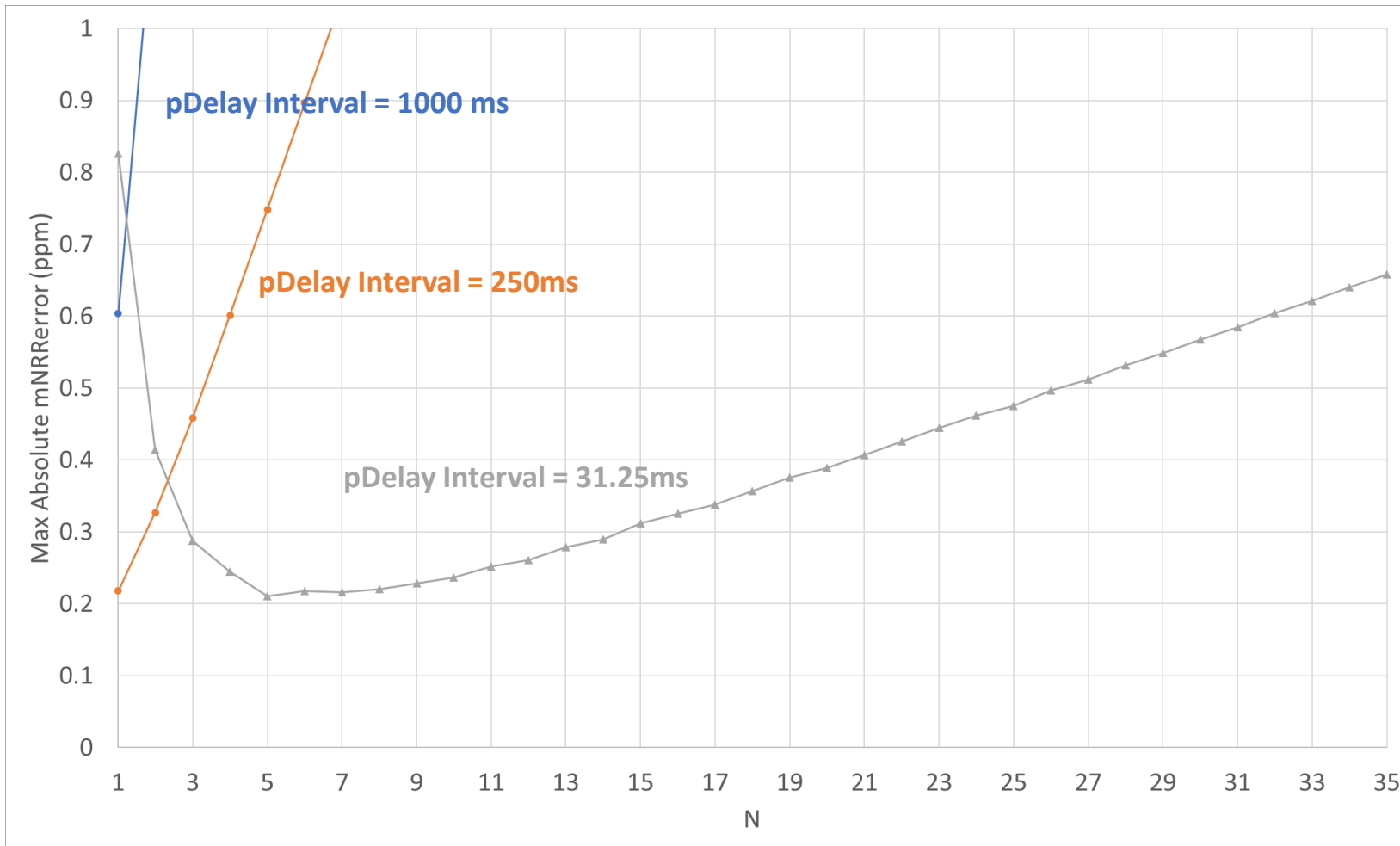


Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Input Correction Factors		
mNRR Smoothing M	1	
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

How short?...

Optimal Value of N



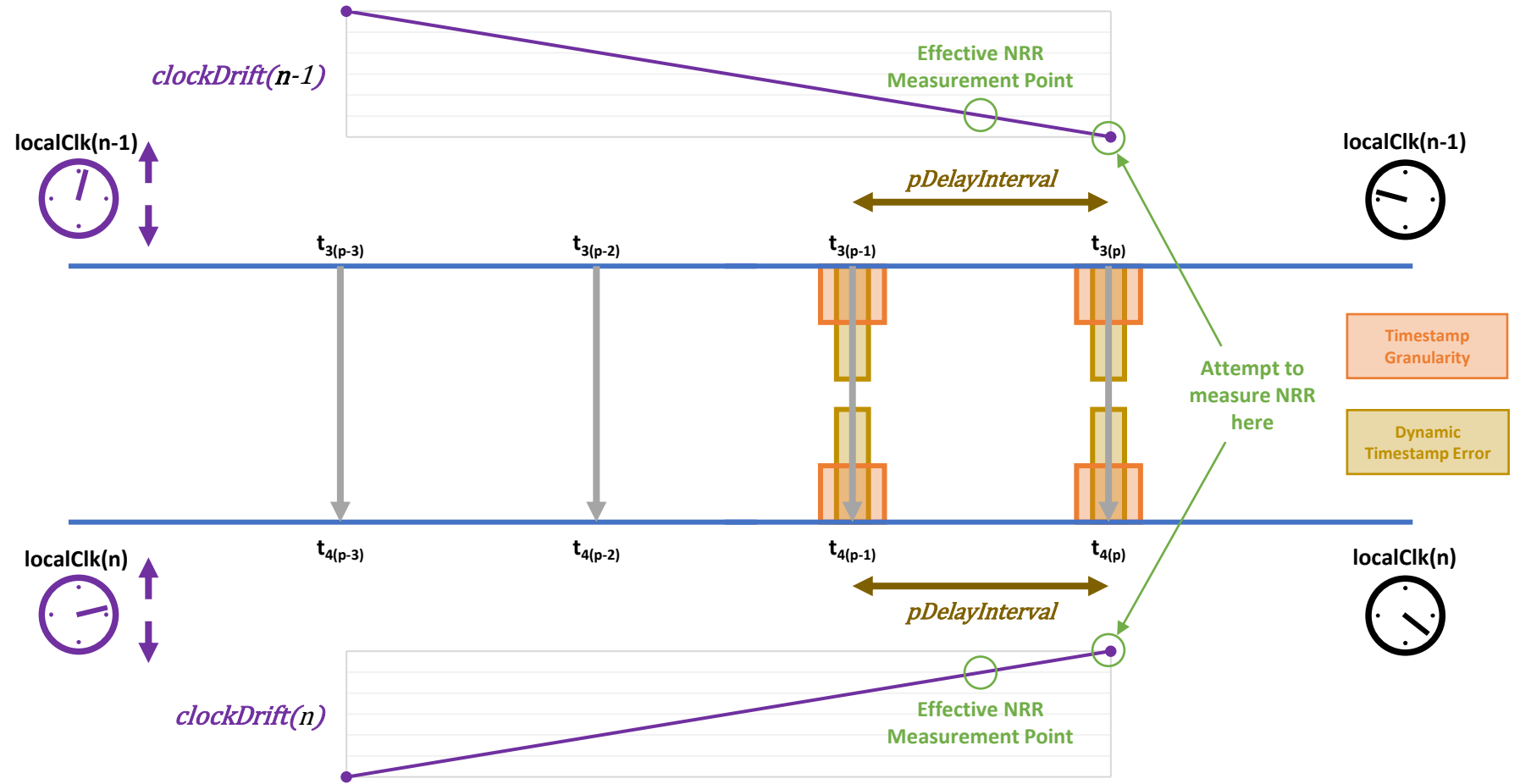
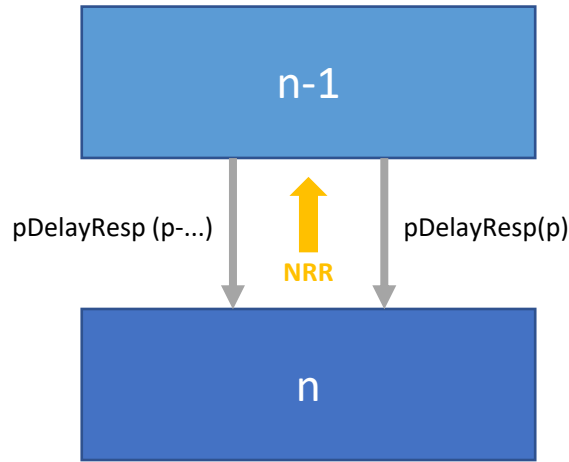
Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Input Correction Factors		
mNRR Smoothing M	1	
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

This is 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}



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

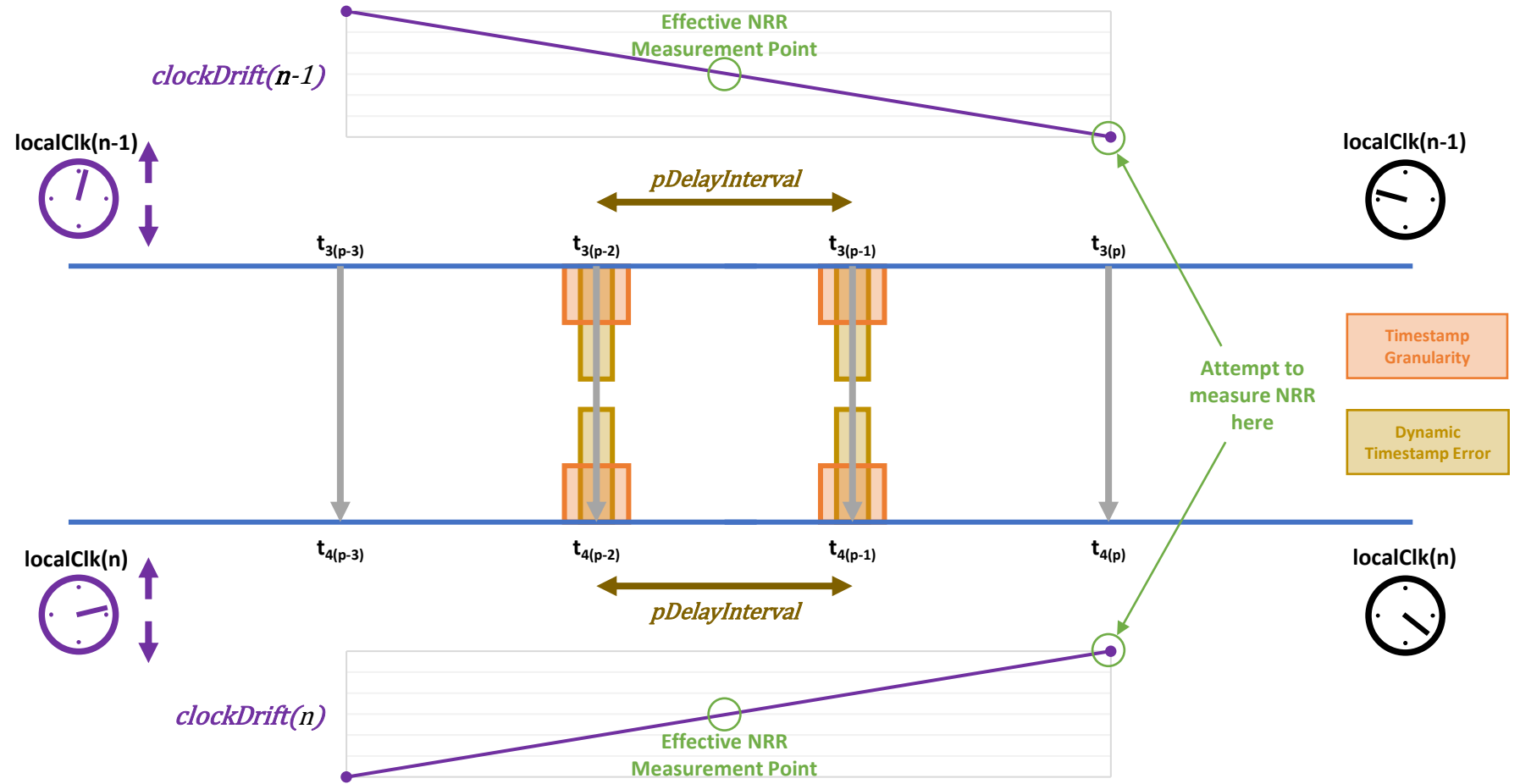
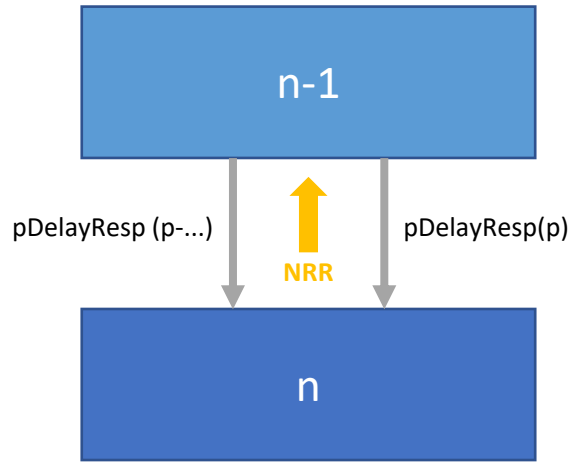
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p)} - t_{4PDerror(p-1)}) - (t_{3PDerror(p)} - t_{3PDerror(p-1)})}{pDelayInterval} \right)$$

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

ppm

ppm

mNRR Smoothing M Calculation B



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

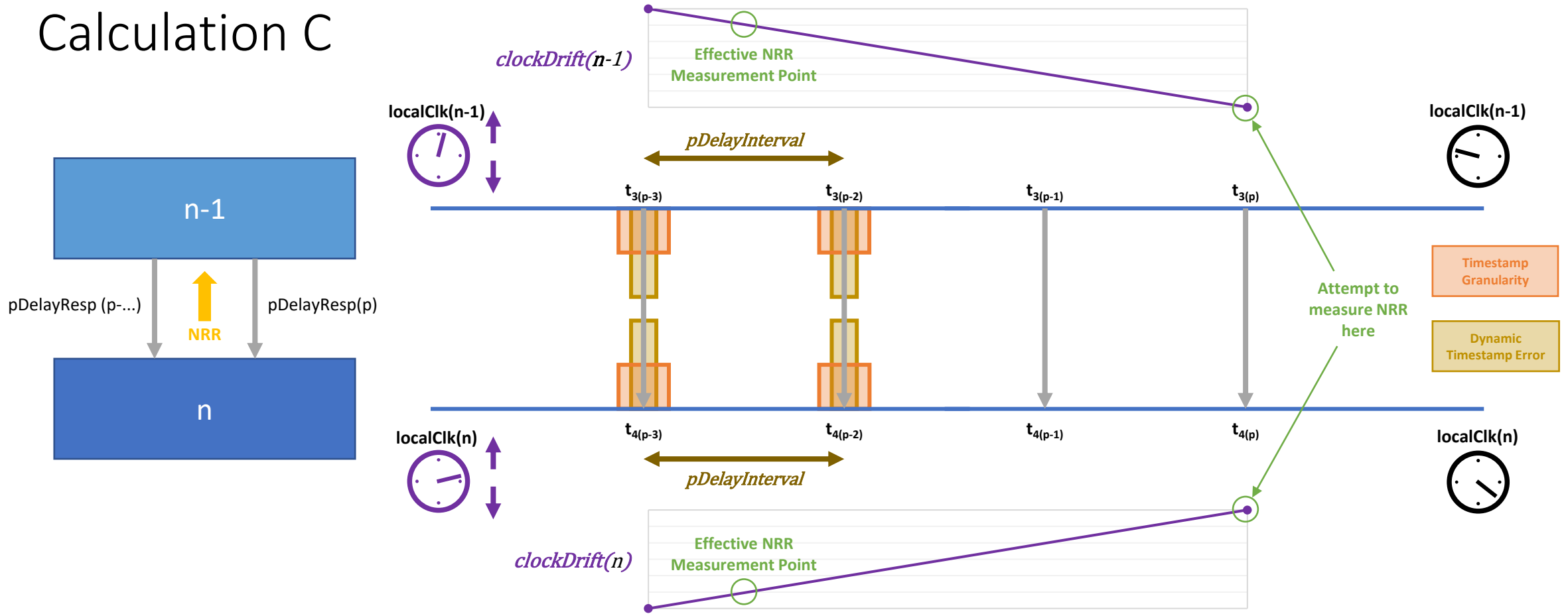
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror(p-1)} - t_{4PDerror(p-2)}) - (t_{3PDerror(p-1)} - t_{3PDerror(p-2)})}{pDelayInterval} \right)$$

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

ppm

ppm

mNRR Smoothing M Calculation C



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

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

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

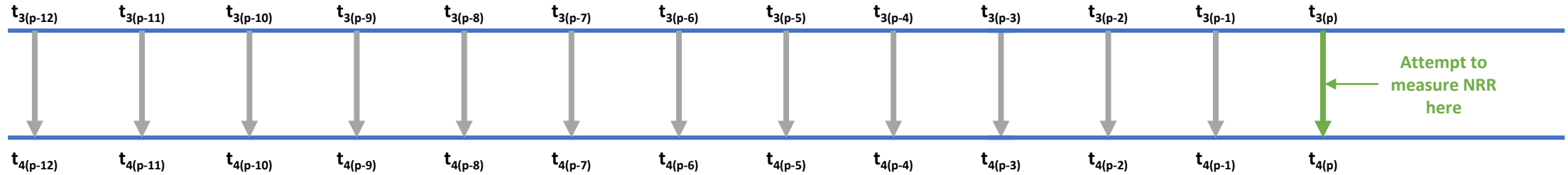
ppm

ppm

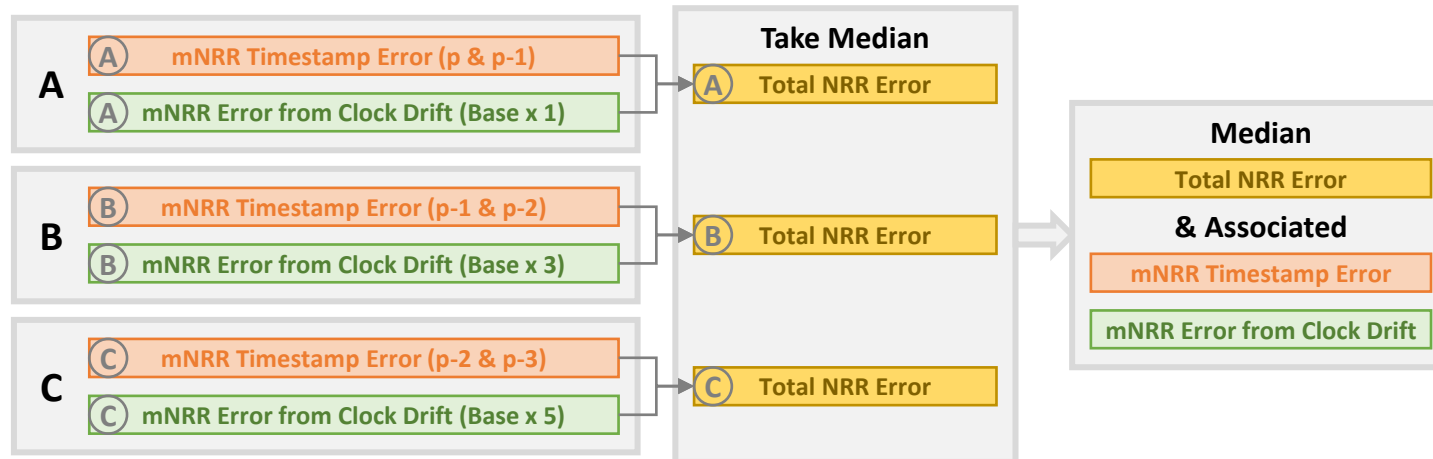
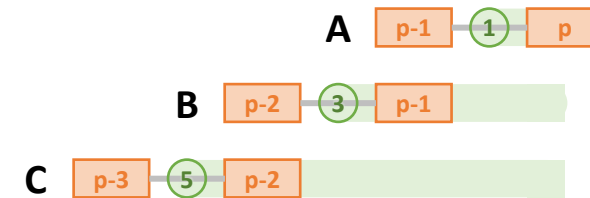
mNRR Smoothing M

- Using the median of M previous NRR calculations...
 - Selects the NRR calculation 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



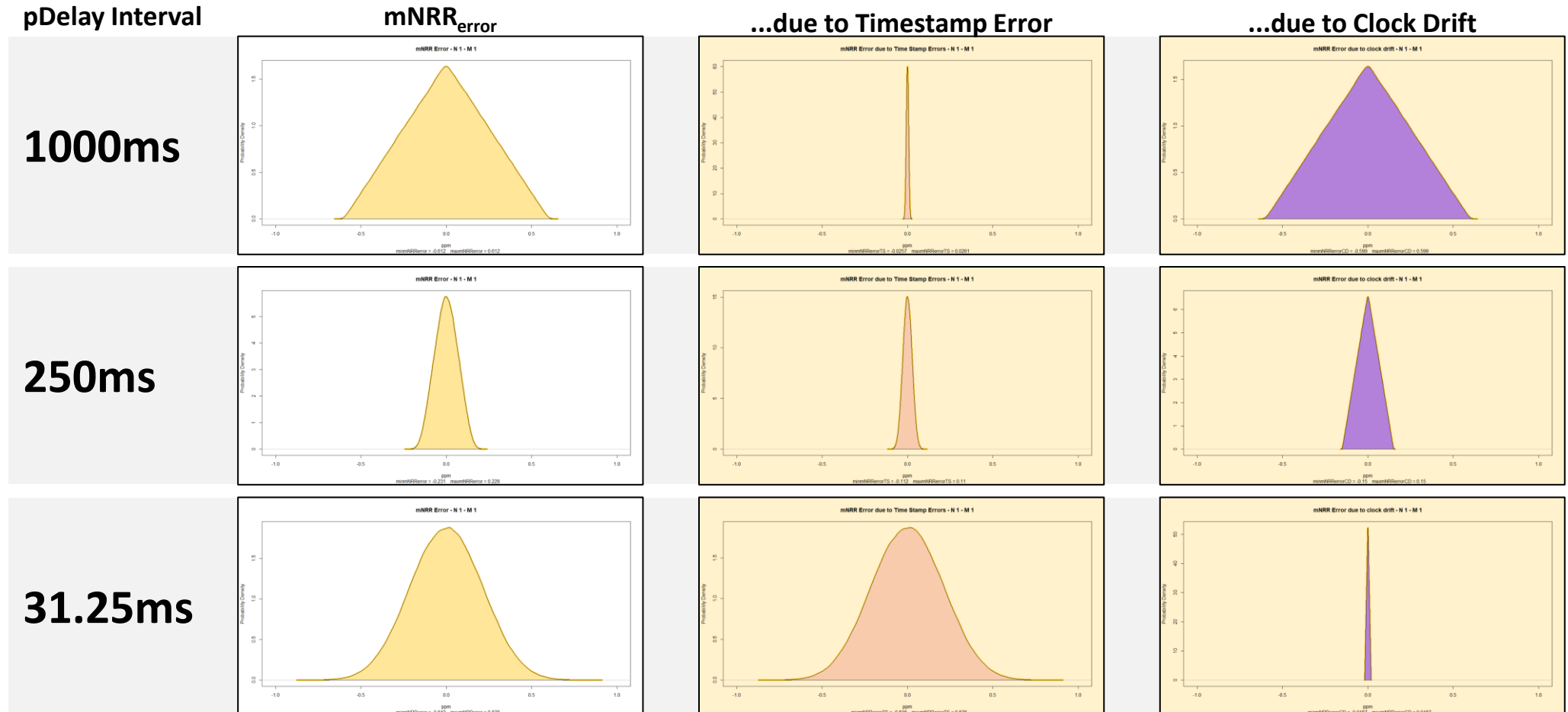
N = 1
M = 3



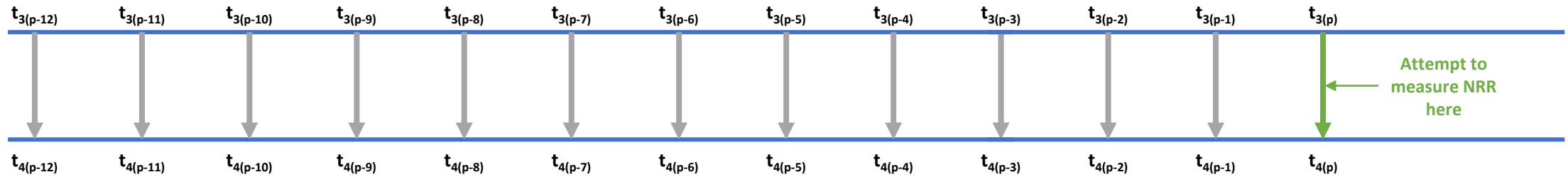
It depends on pDelay Interval!
(And the values of M & N)

Timestamp Error vs Error due to Clock Drift

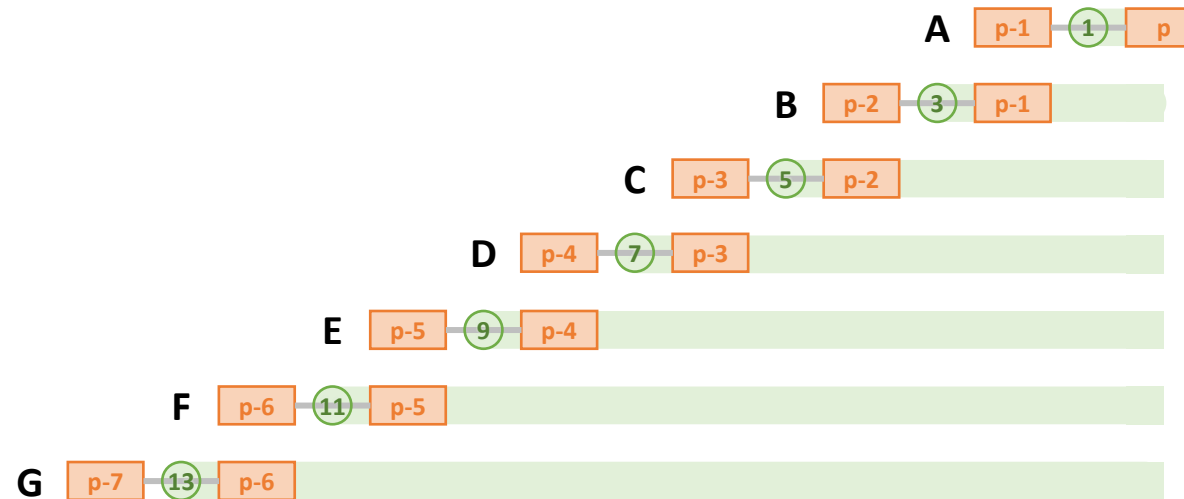
(mNRR – Single Hop – No Smoothing, i.e. N=1, M=1)



mNRR Smoothing M & pDelay Interval

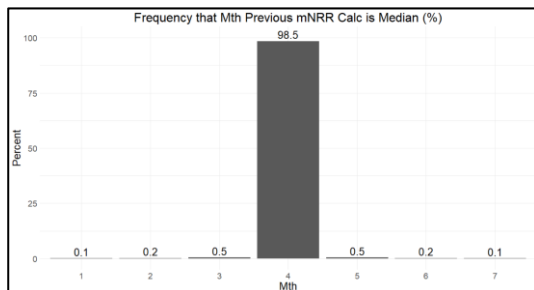
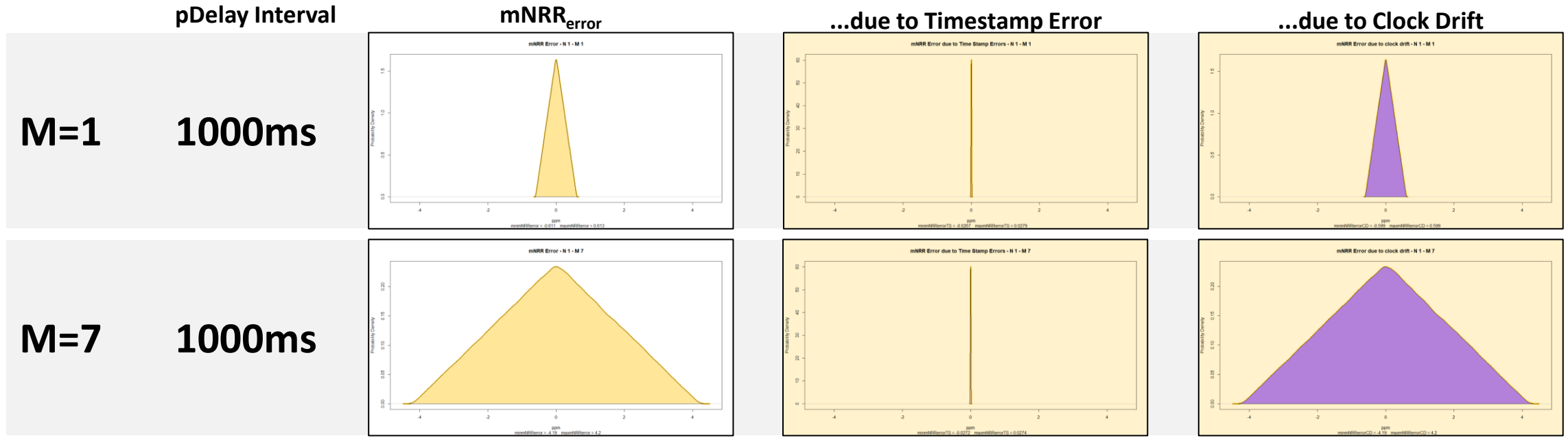


N = 1
M = 7



Effect of Taking Median of mNRR Calculations

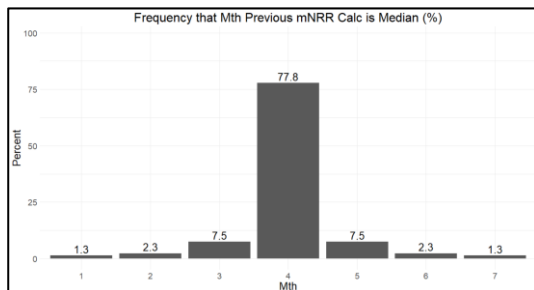
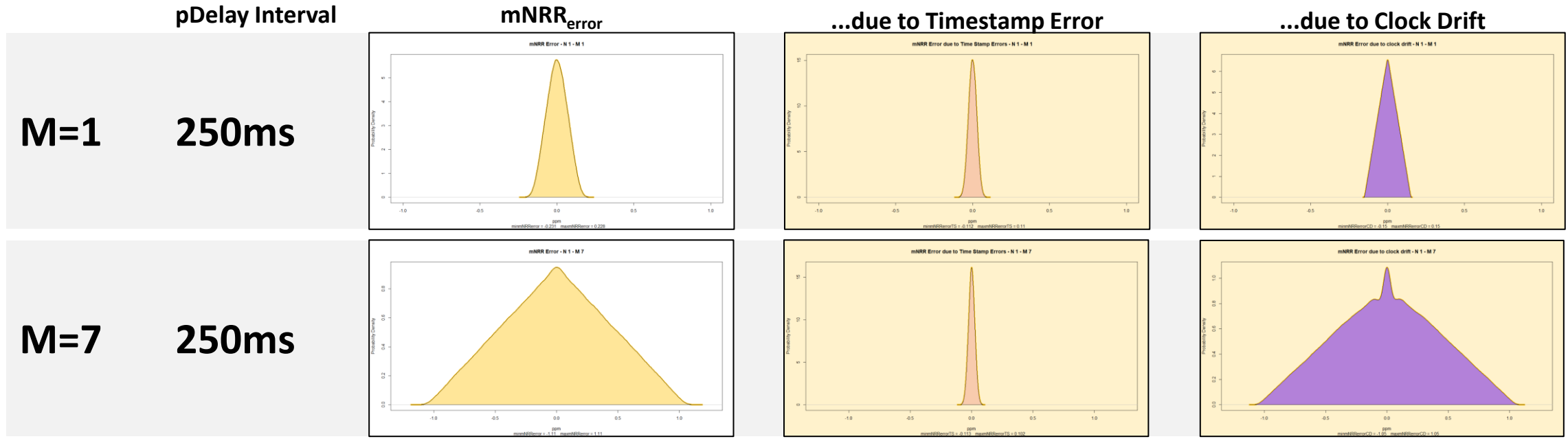
(mNRR – Single Hop – N=1)



- 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

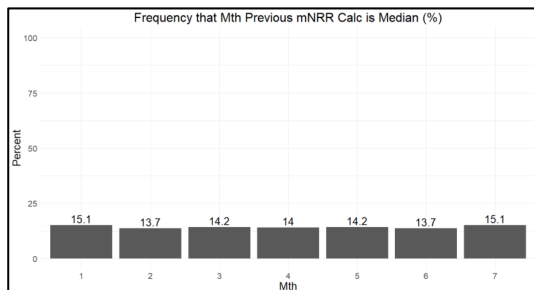
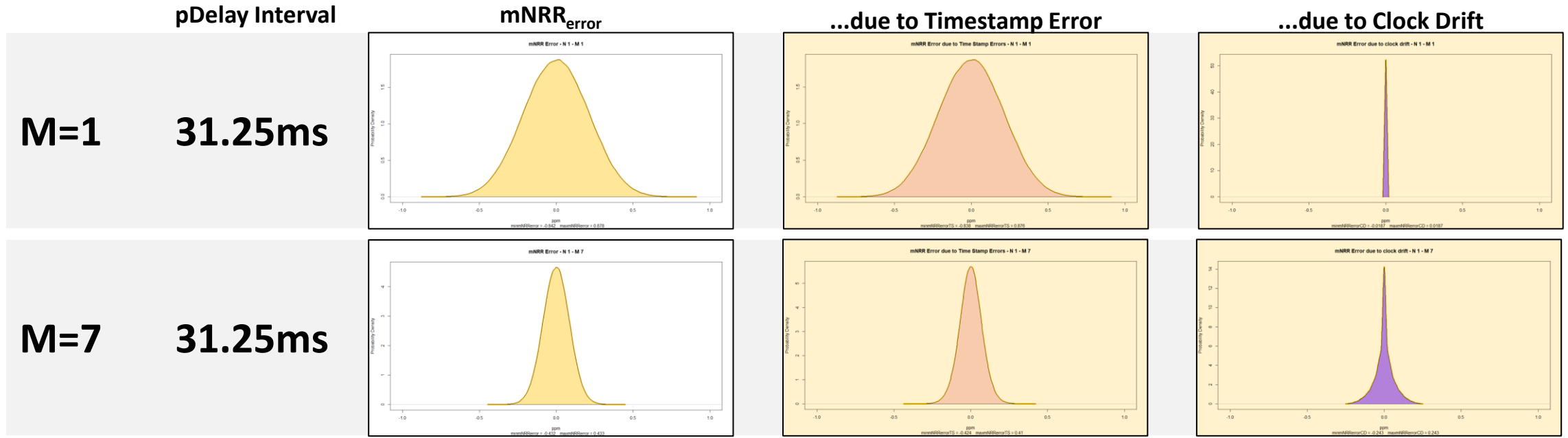
(mNRR – Single Hop – N=1)



- 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)

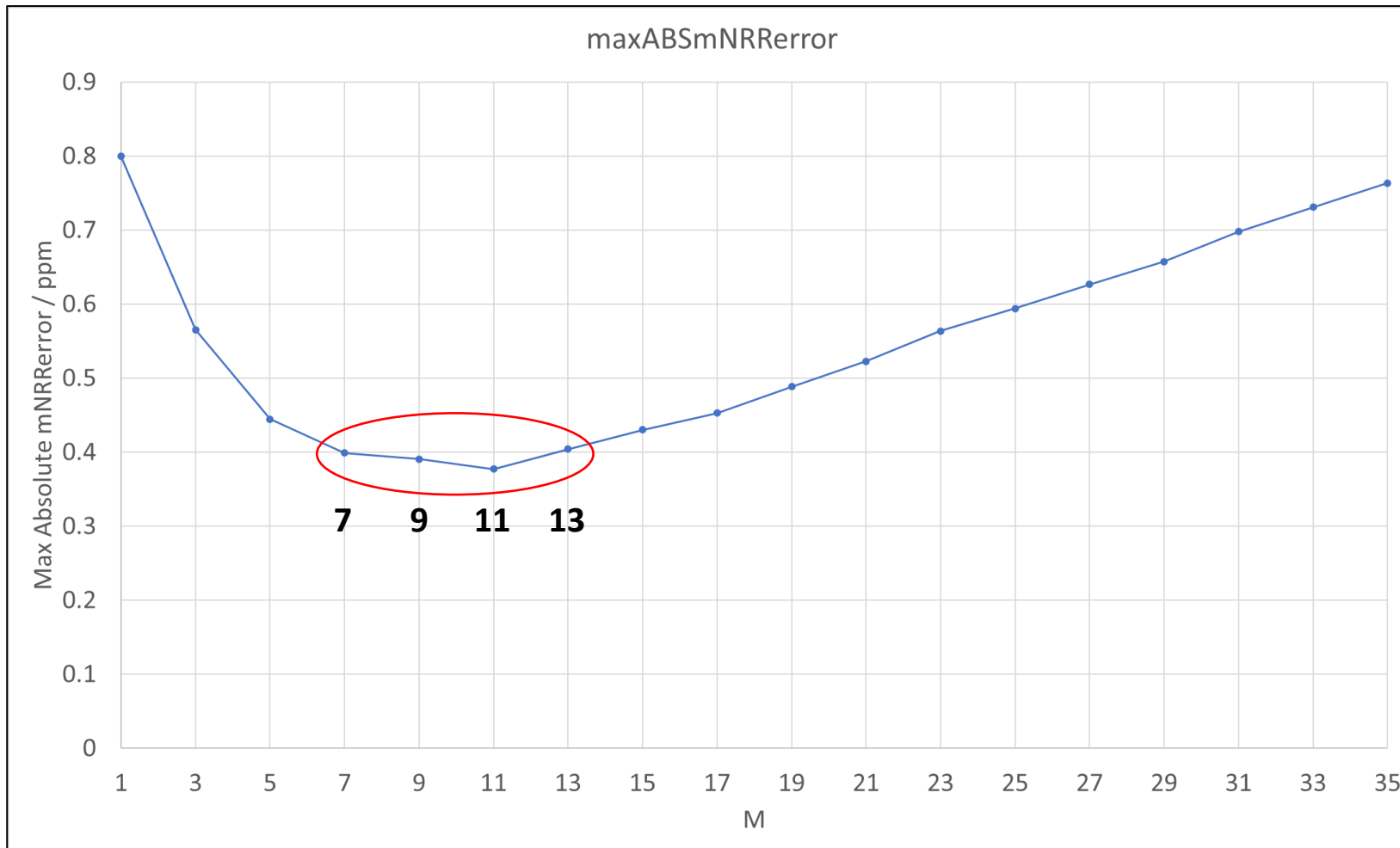


- For values of pDelay Interval where mNRR_{error} due to Timestamp error dominates error due to Clock Drift, taking the median of previous NRR calculations can reduce mNRR_{error}
- This raises the question: what is the optimal value of M?

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 noisy
 - 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

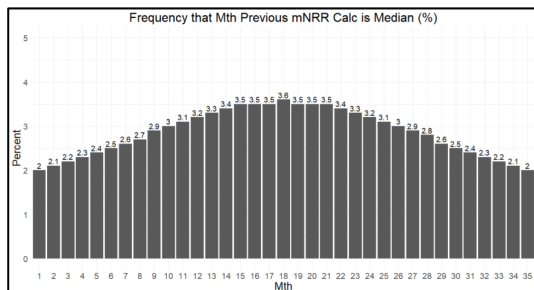
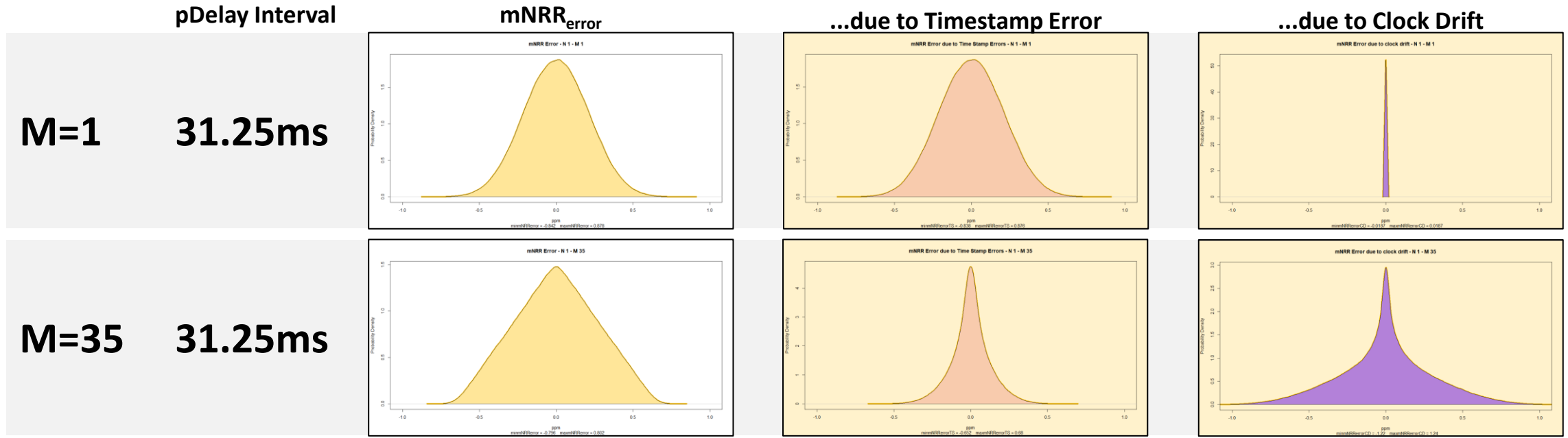


Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Input Correction Factors		
mNRR Smoothing N	1	
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

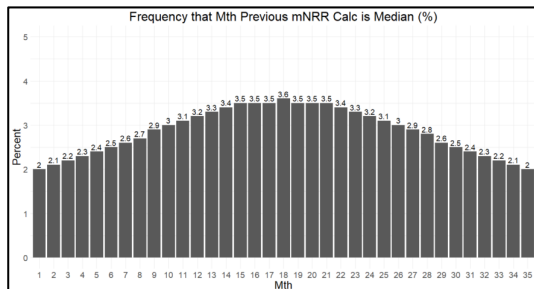
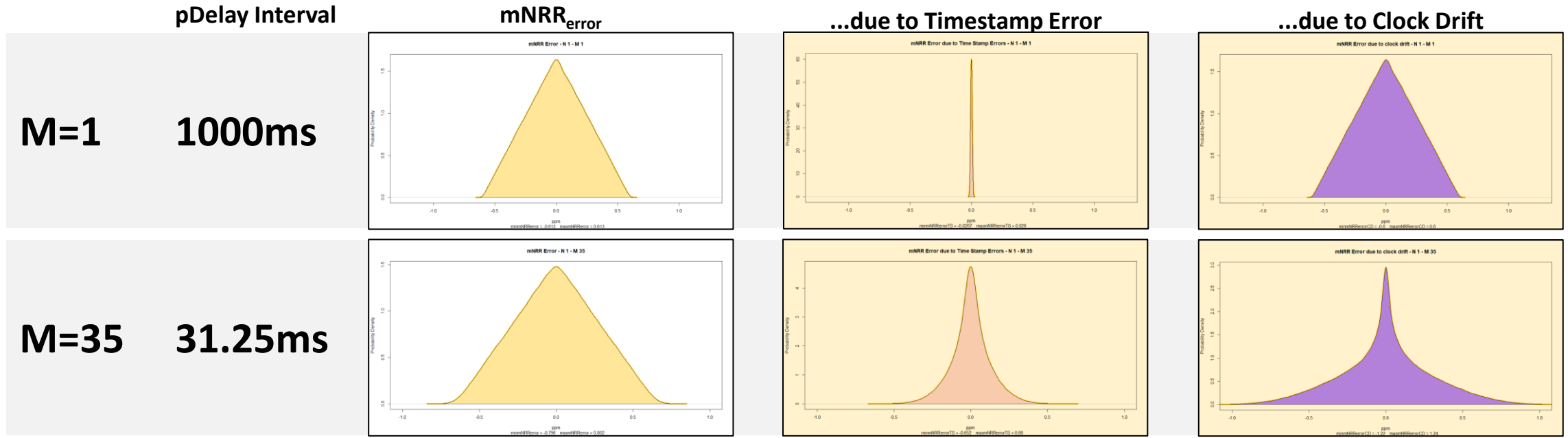
(mNRR – Single Hop – N=1)



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

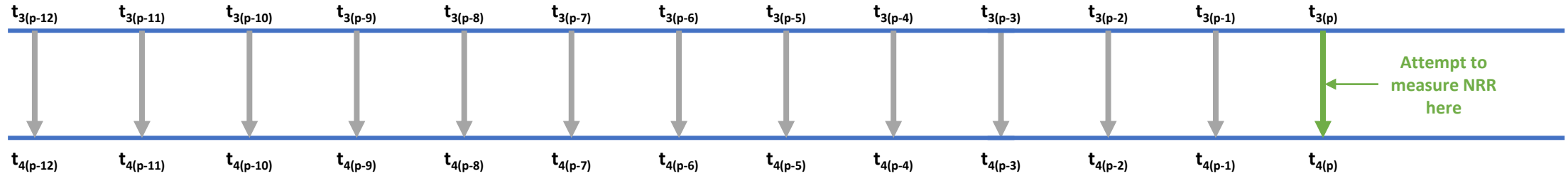
Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

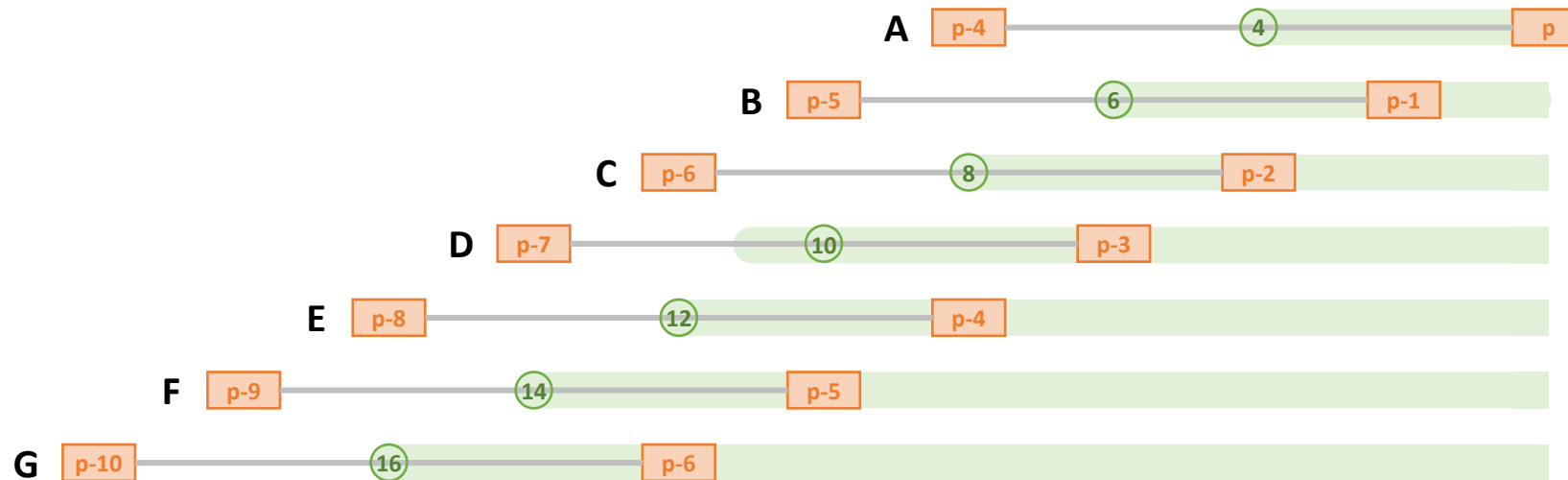


- The distribution of the underlying Timestamp errors and Clock Drift errors are very different.

mNRR Smoothing – Combining N & M

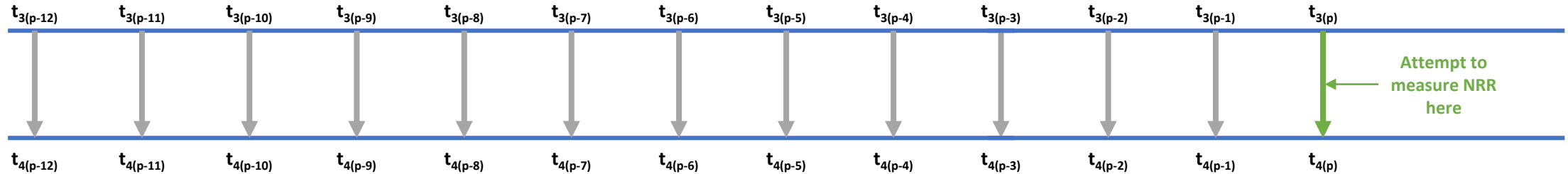


N = 4
M = 7

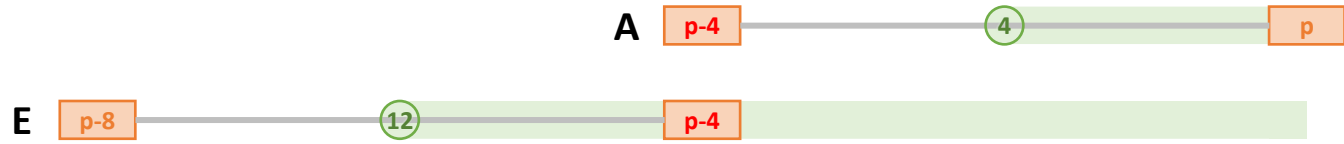


Note that the timestamps from p-4, p-5 & p-6 are used twice...which has an interesting effect...

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

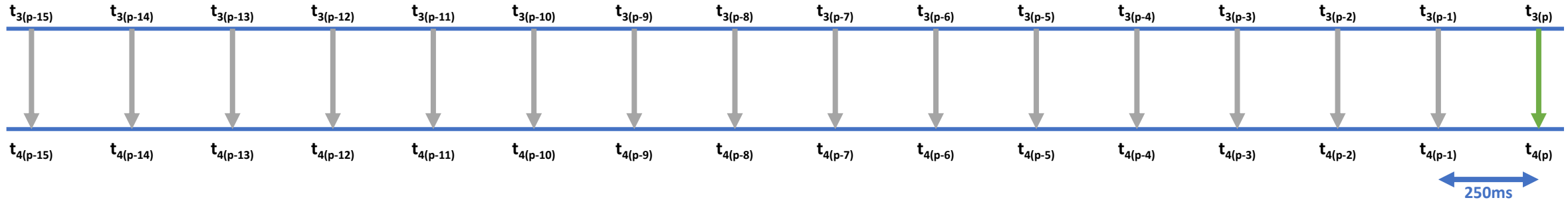
Series 1

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

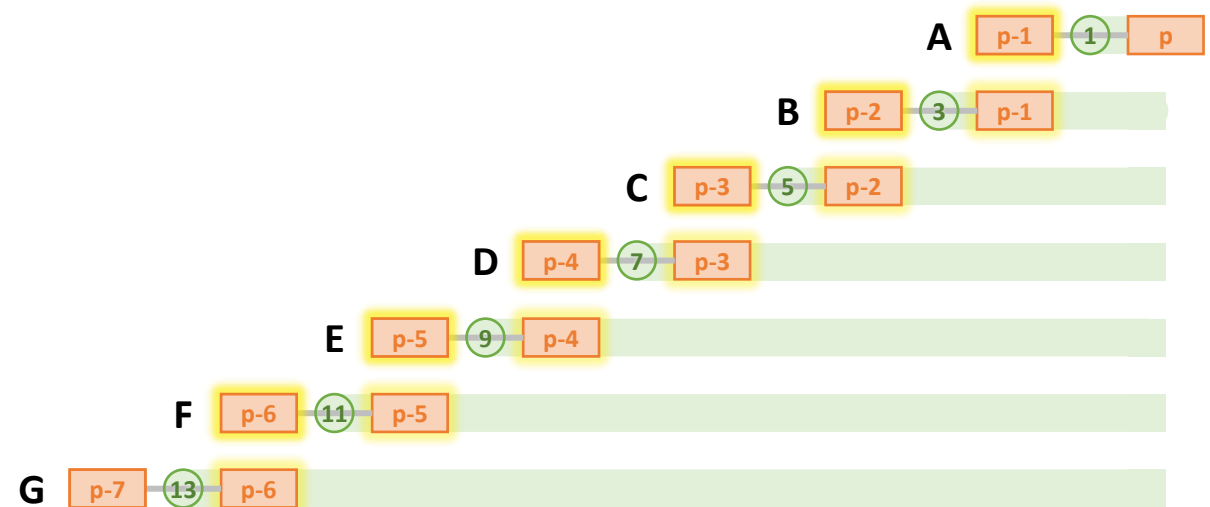


Series 1

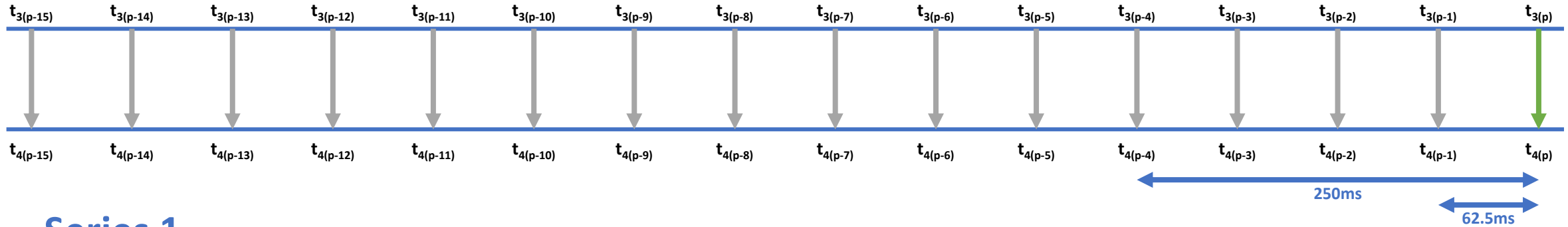
pDelay Interval = 250ms

N = 1

M = 7



mNRR Smoothing – Combining N & M

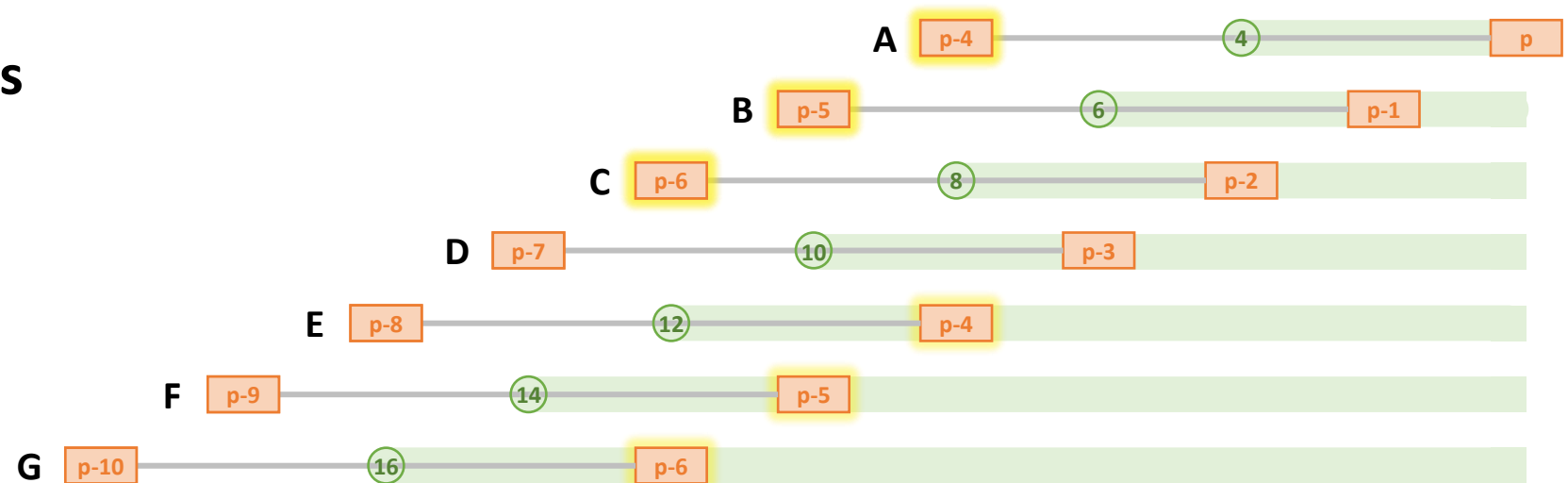


Series 1

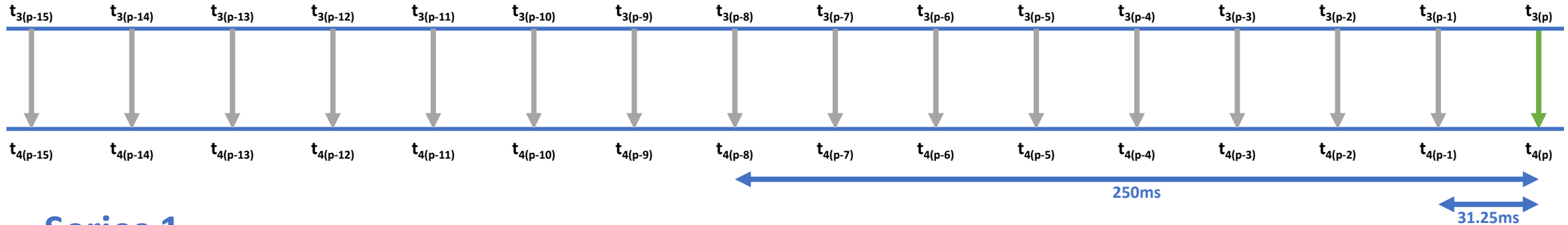
pDelay Interval = 62.5ms

N = 4

M = 7



mNRR Smoothing – Combining N & M

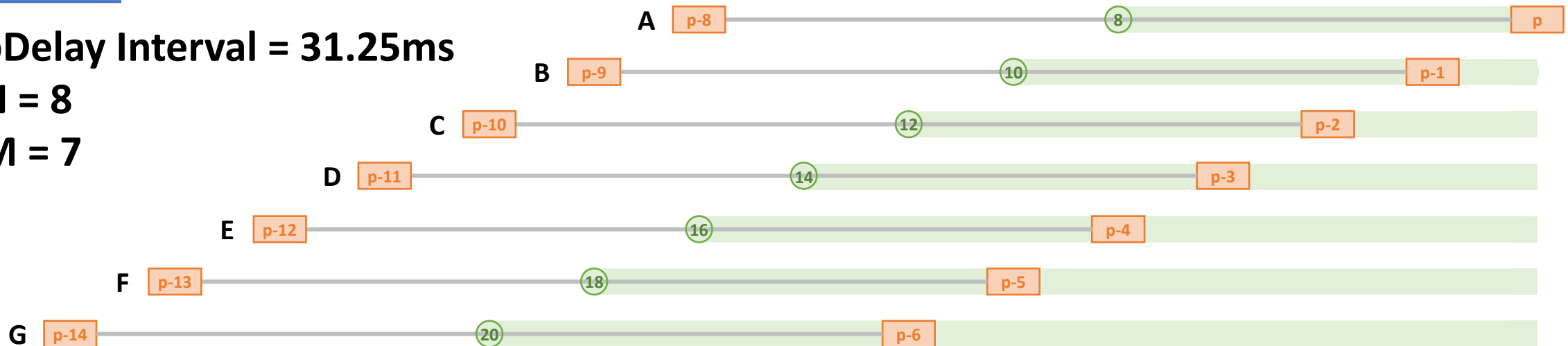


Series 1

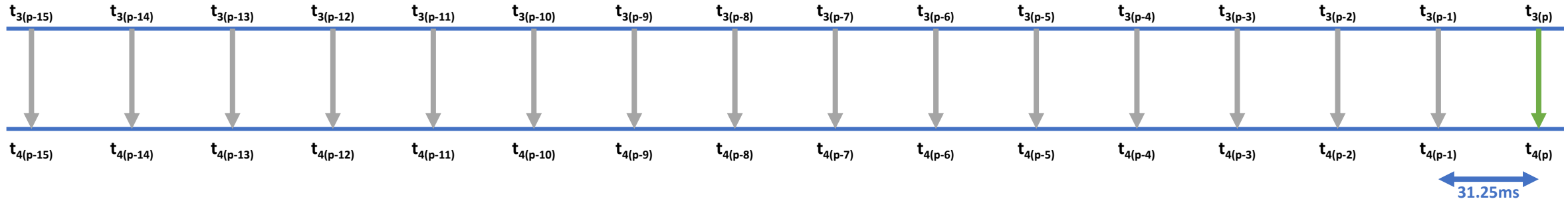
pDelay Interval = 31.25ms

N = 8

M = 7



mNRR Smoothing – Combining N & M

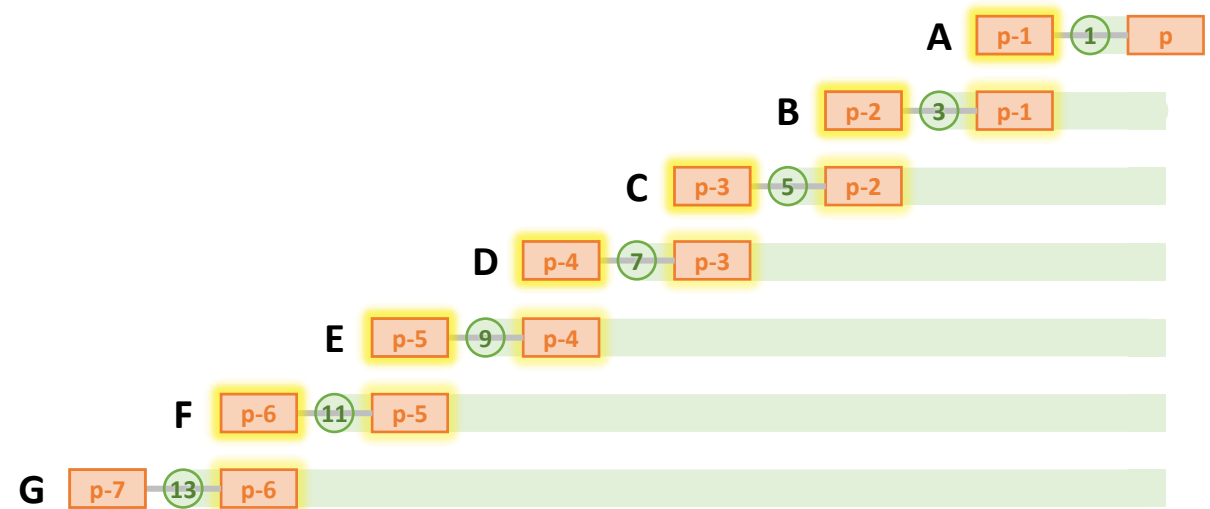


Series 2

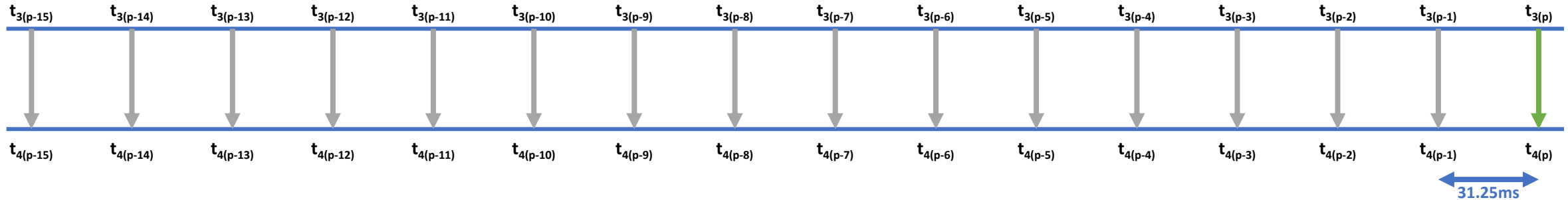
pDelay Interval = 31.25ms

N = 1

M = 7



mNRR Smoothing – Combining N & M

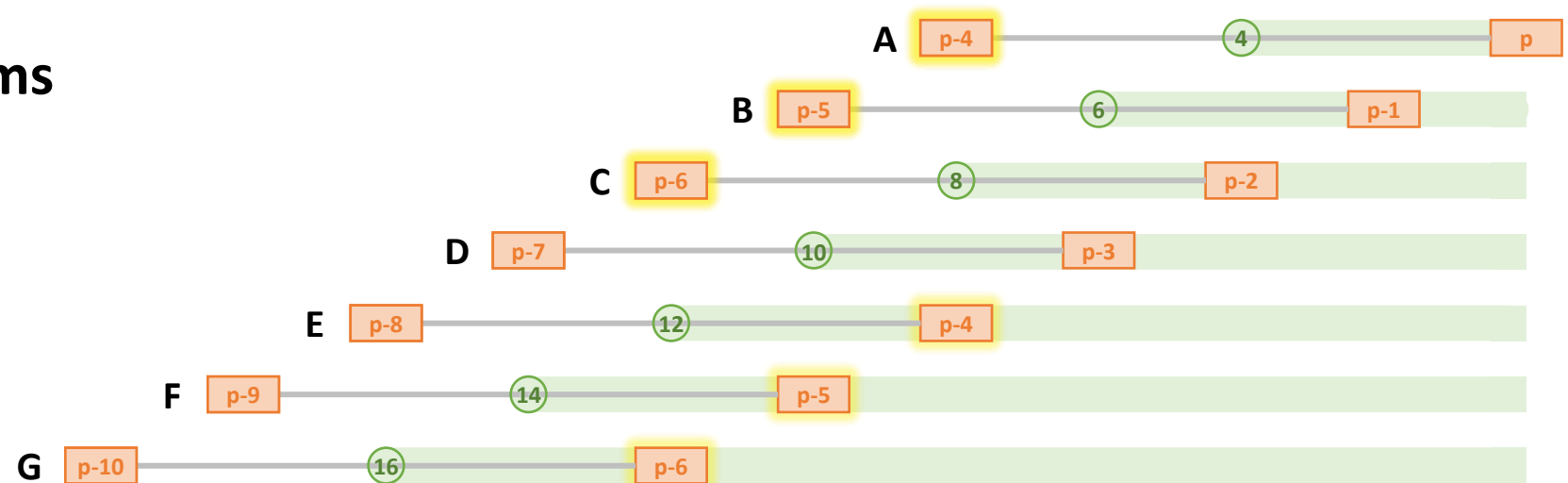


Series 2

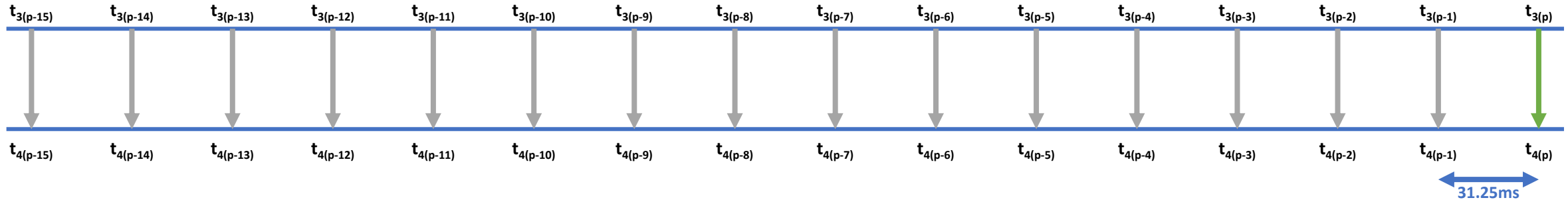
pDelay Interval = 31.25ms

N = 4

M = 7



mNRR Smoothing – Combining N & M

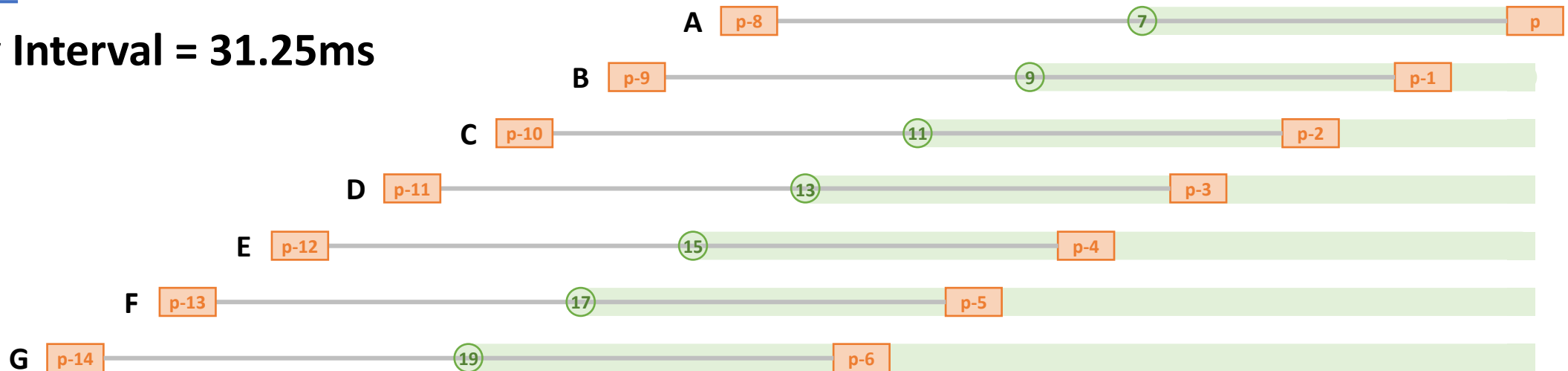


Series 2

pDelay Interval = 31.25ms

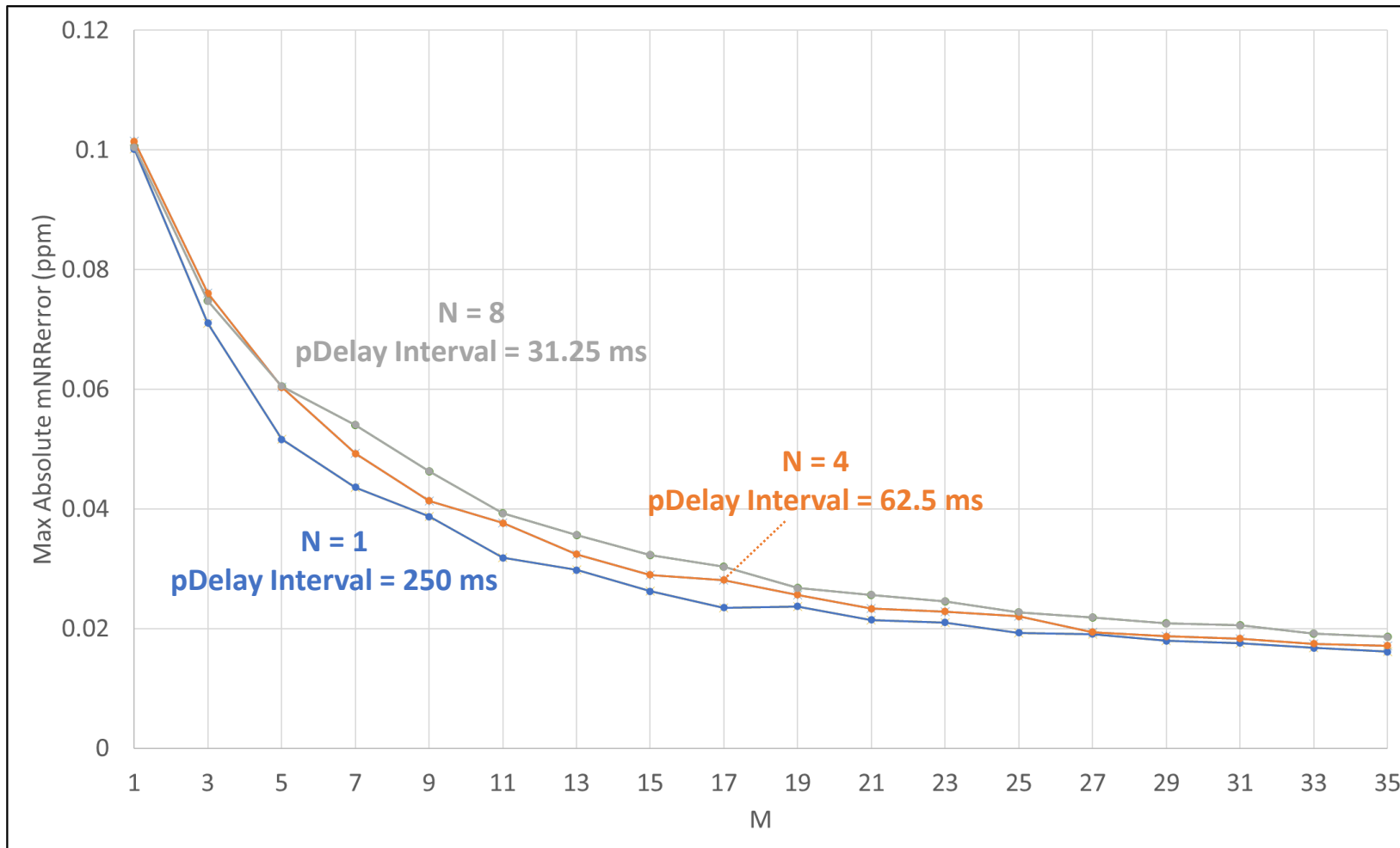
N = 7

M = 7



Varying M & N – No Clock Drift

(N x pDelayInterval maintained at 250ms)

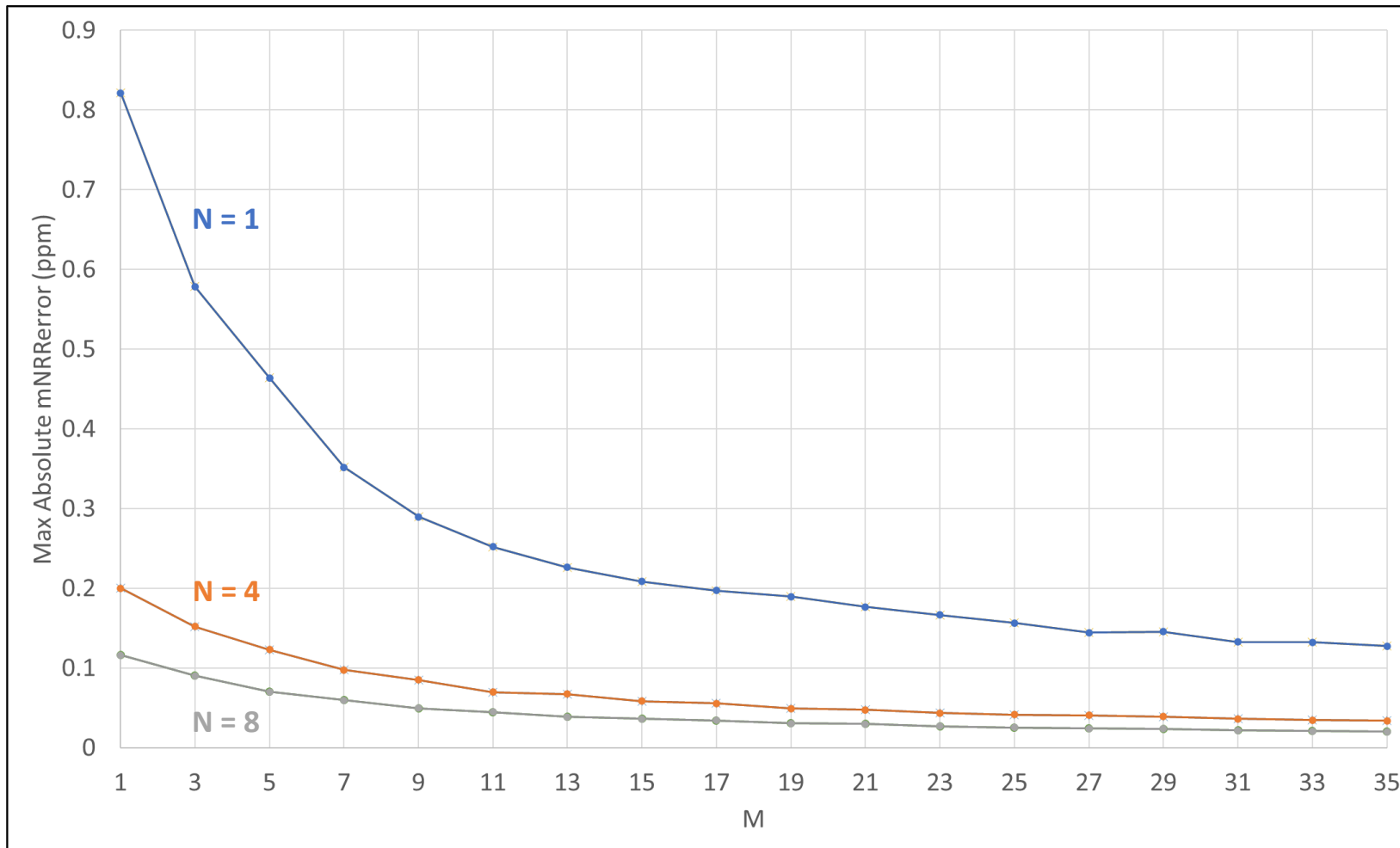


Input Errors		
Clock Drift	±0	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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)

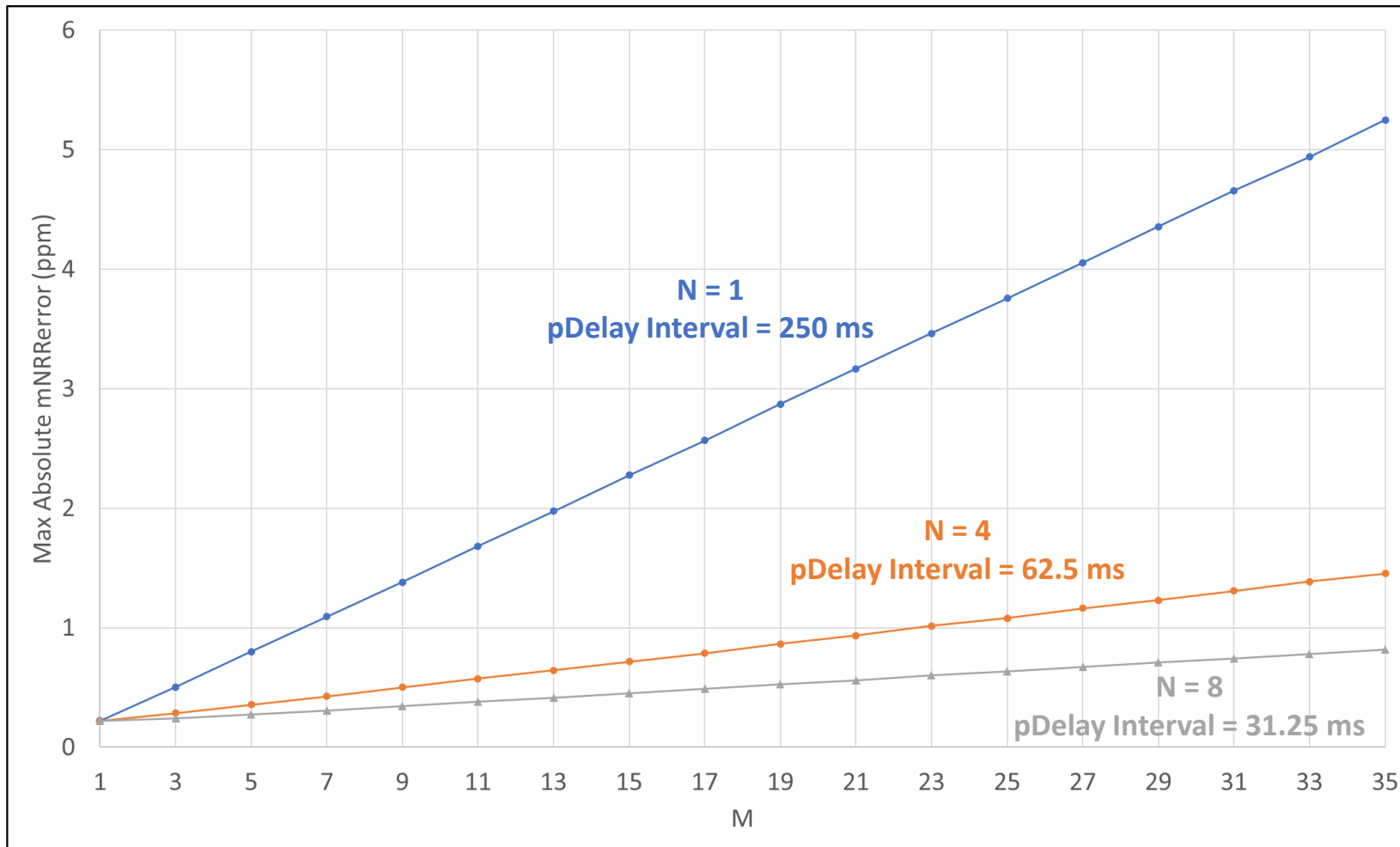


Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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)

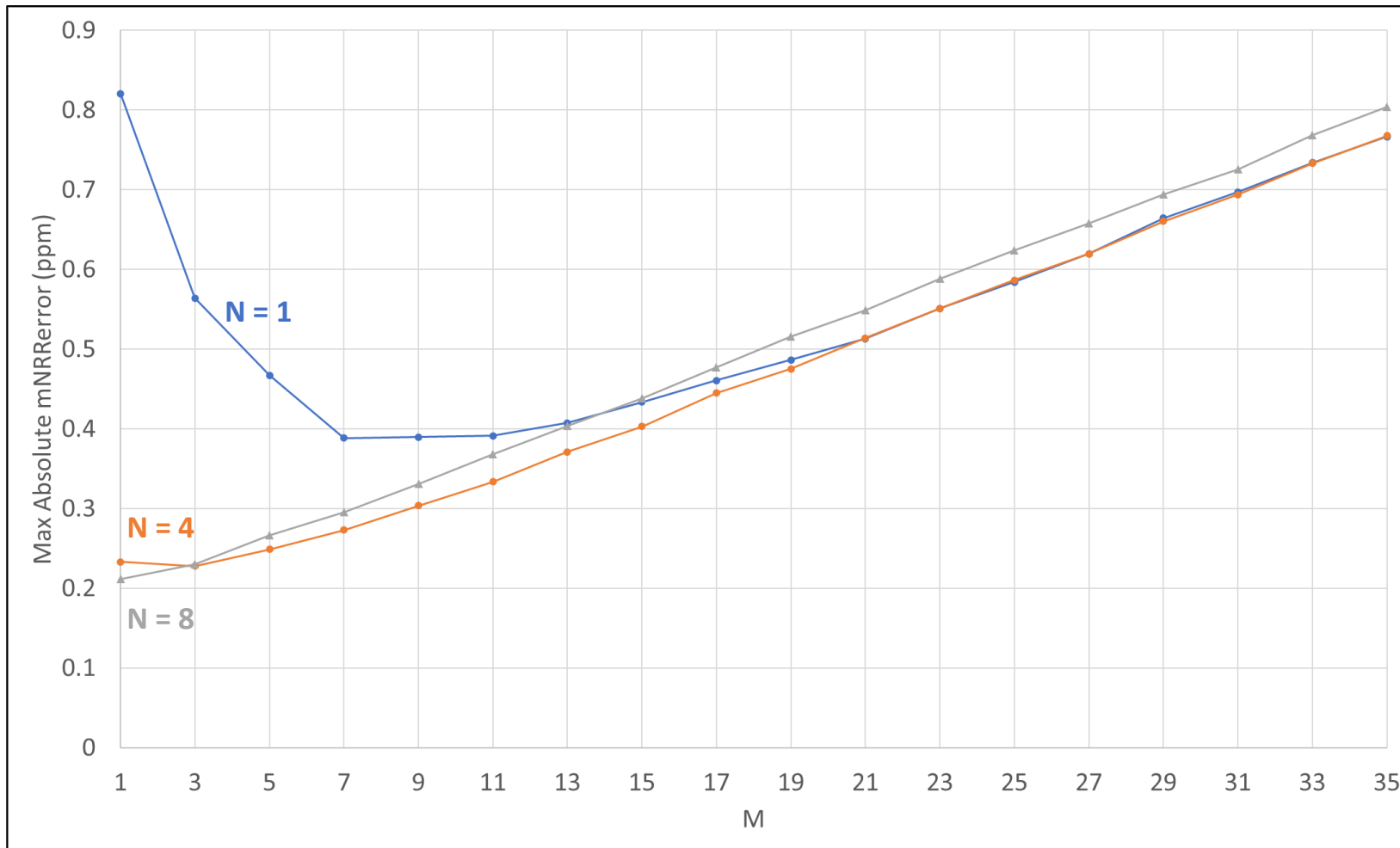


Input Errors		
Clock Drift	±0	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

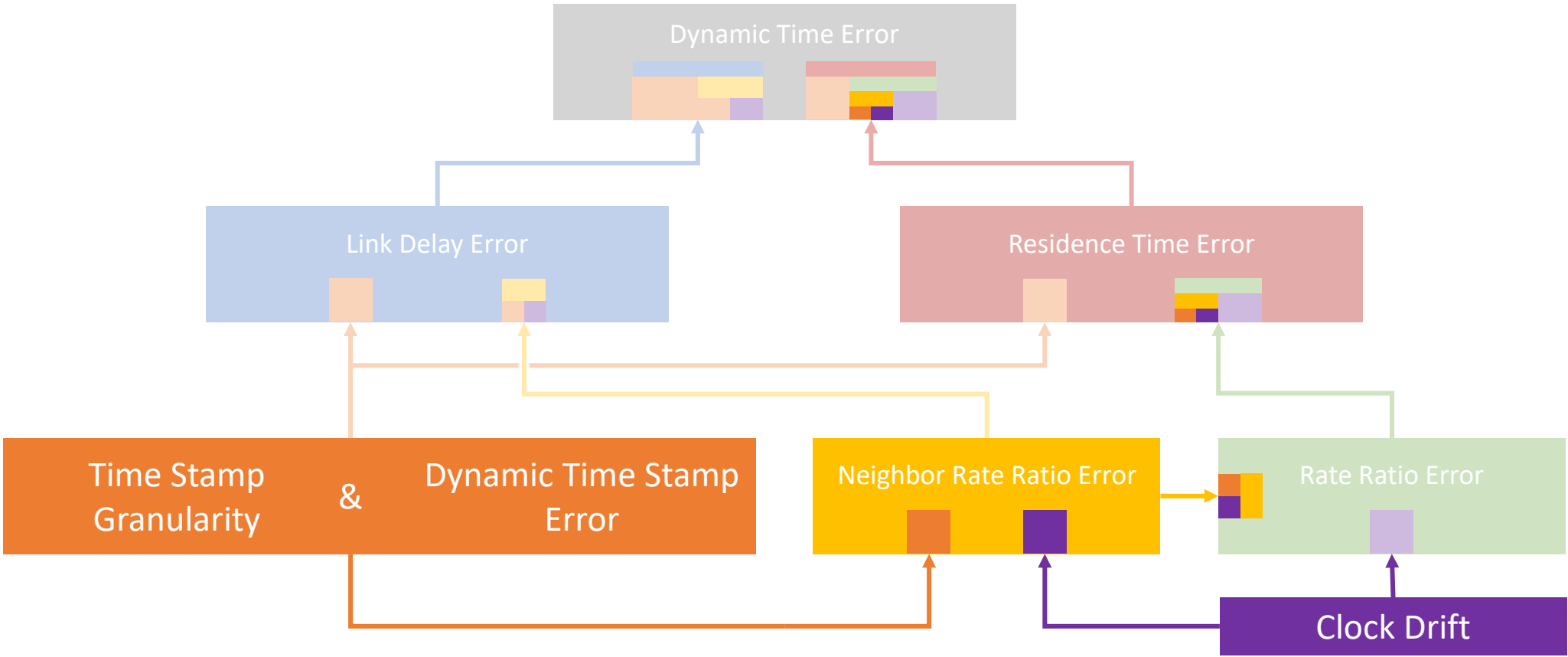
(N x pDelayInterval maintained at 250ms)



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	

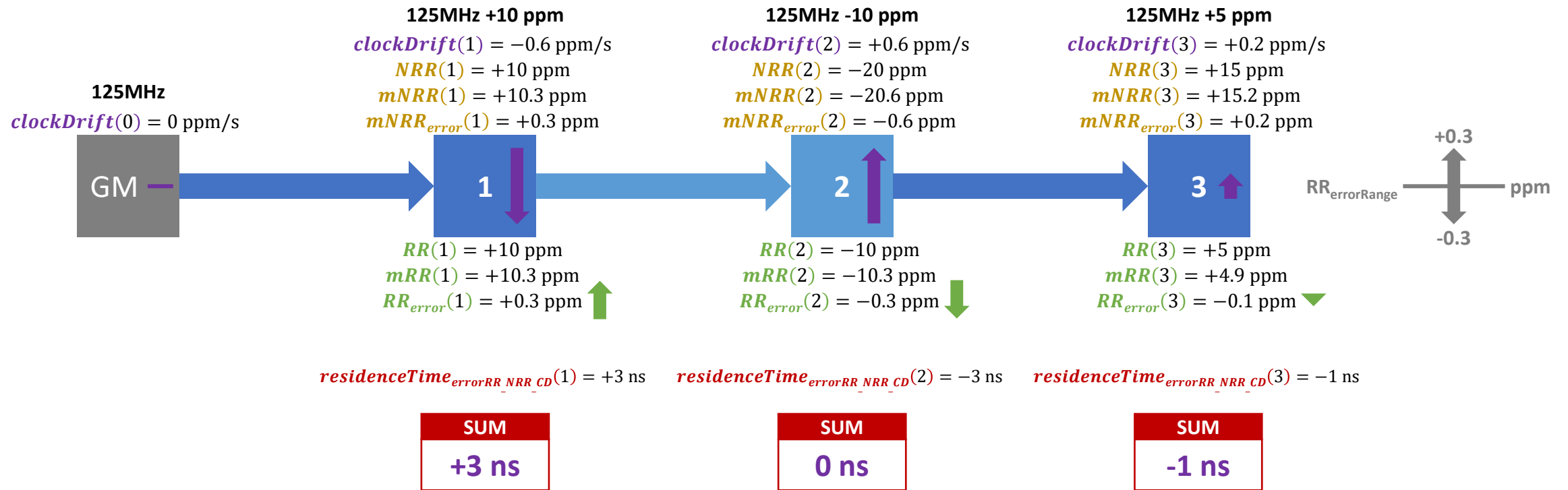
Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $mNRR_{error}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of 250ms.

NRR Errors Accumulate in RR Error

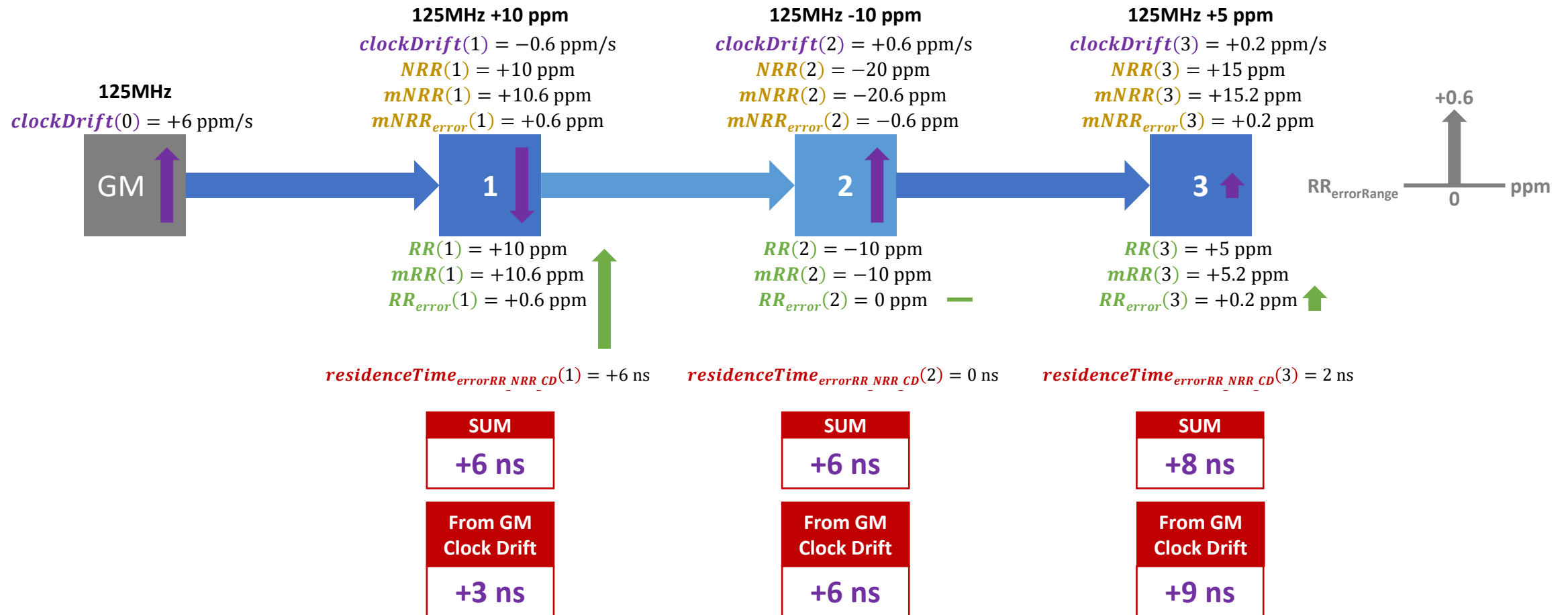


But there is a big caveat for how NRR errors due to Clock Drift accumulate...

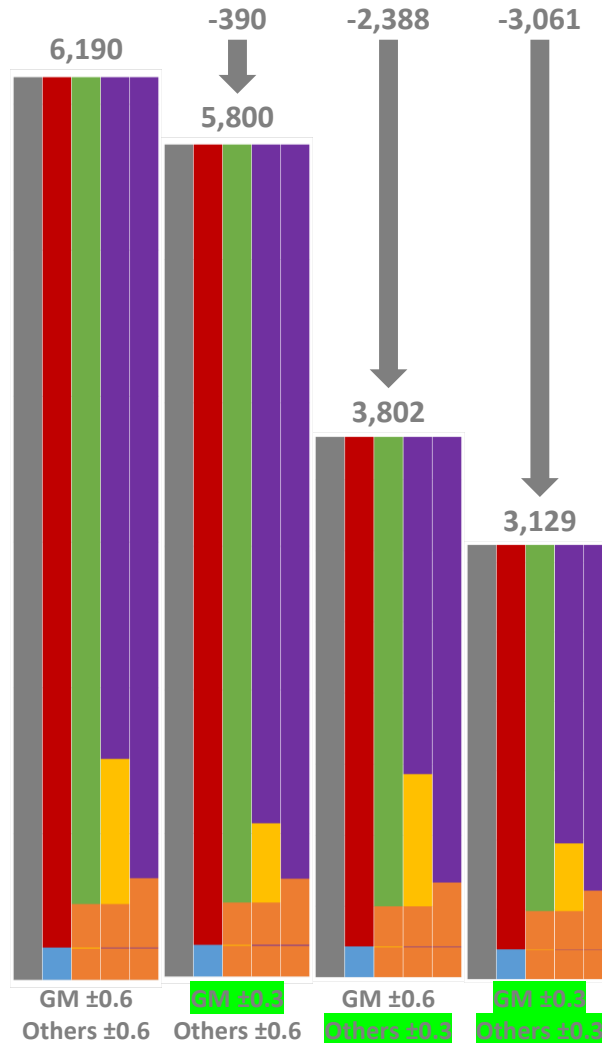
How *residenceTime_{errorRR_NRR_CD}* Accumulates



How *residenceTime_{errorRR_NRR_CD}* Accumulates



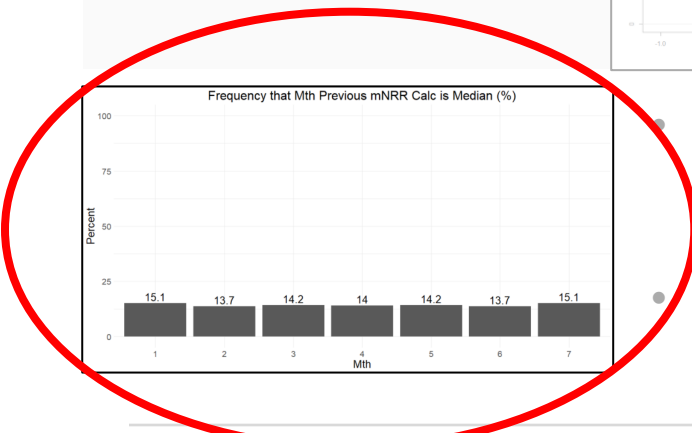
Clock Drift Sensitivity



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

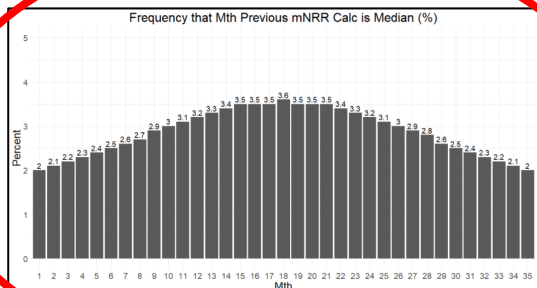
(mNRR – Single Hop – N=1)



- 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?

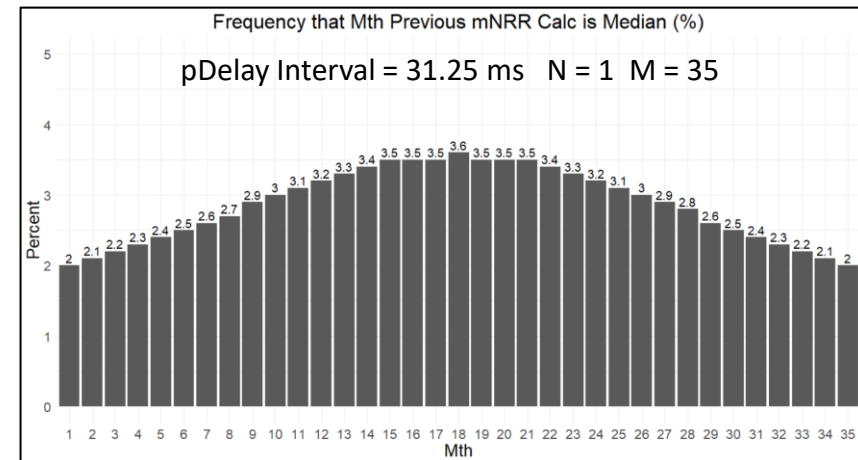
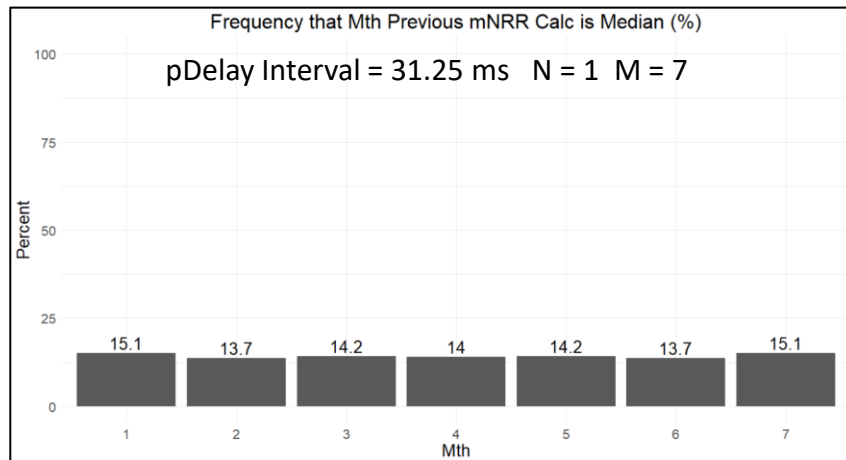
Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)



- Increasing M to 35 results in using NRR calculations from up to 1093.75ms in the past...and an $mNRR_{error}$ 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

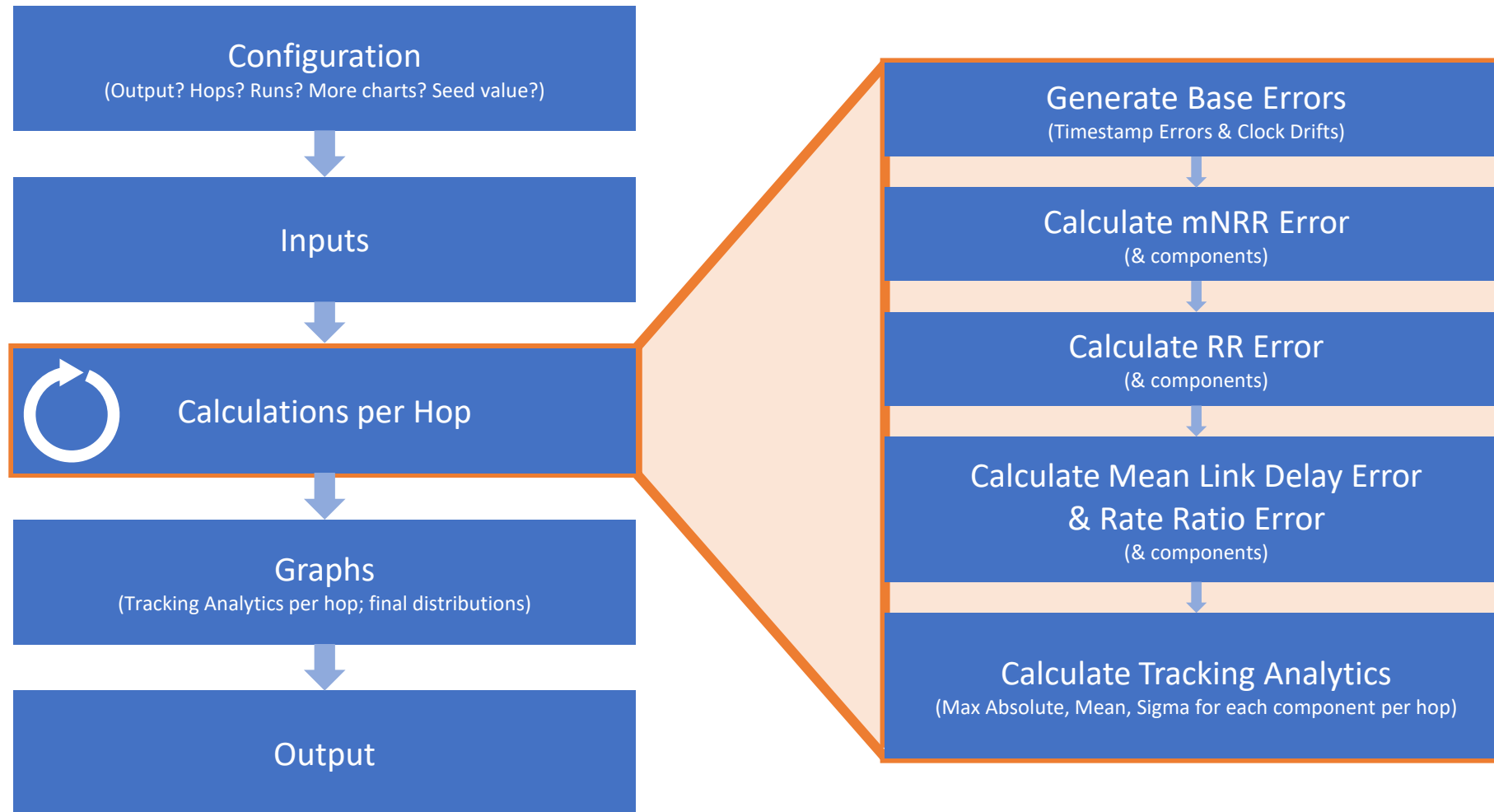


- Using the median of past NRR calculations brings with it a chance that a larger portion of the error won't cancel out at the next node but will instead survive.
 - This portion will behave more like mNRRerror due to Clock Drift at the GM than at other nodes when M=1.

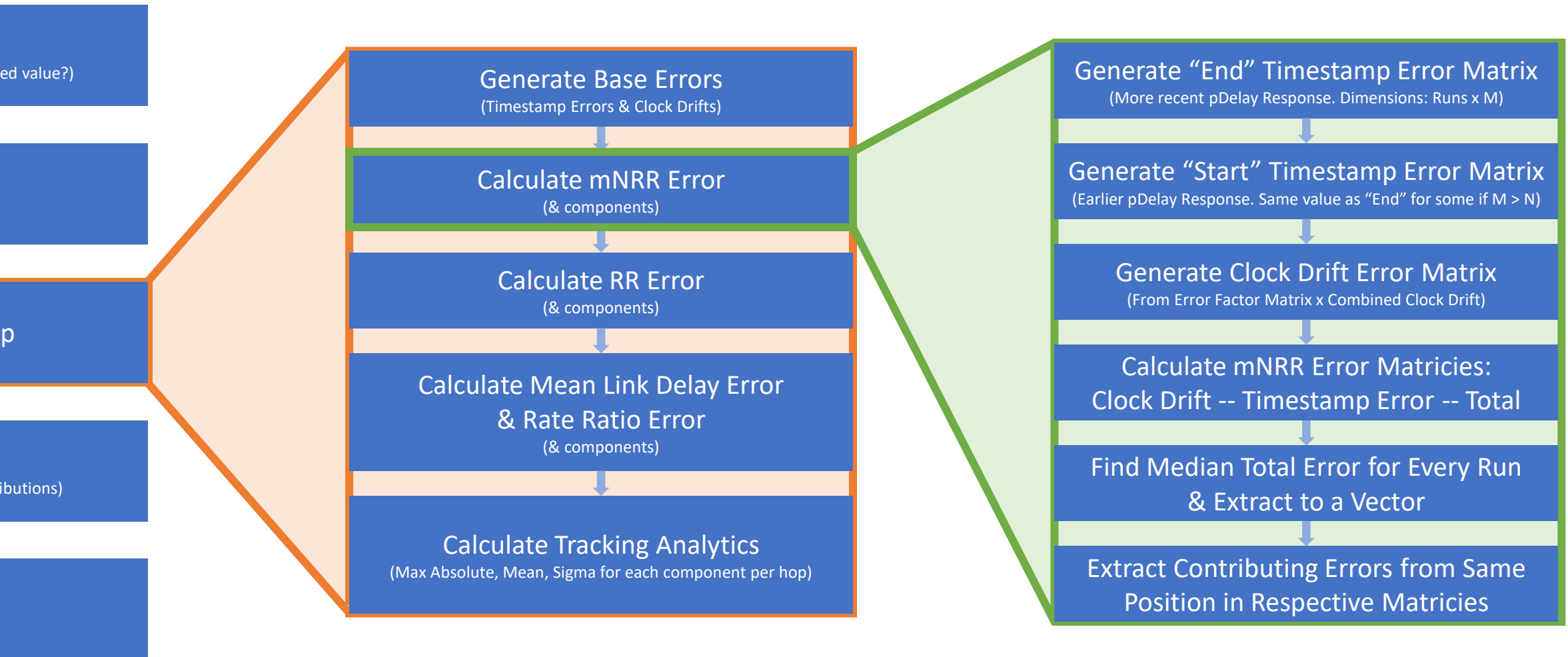
Summary

- Using a median of past NRR calculations can reduce $mNRR_{\text{error}}$ 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...

if $X \leq 1000, F = X$

$$\text{MeanLinkDelay}(1) = p\text{Delay}(1)$$

if $X > 1000, F = 1000$

$$\text{MeanLinkDelay}(X) = \frac{(\text{MeanLinkDelay}(X - 1) \times (F - 1)) + p\text{Delay}(X)}{F}$$

- So, for example...

$$\text{MeanLinkDelay}(100) = \frac{(\text{MeanLinkDelay}(99) \times (99)) + p\text{Delay}(100)}{100}$$

$$\text{MeanLinkDelay}(10500) = \frac{(\text{MeanLinkDelay}(10499) \times (999)) + p\text{Delay}(10500)}{1000}$$

- 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

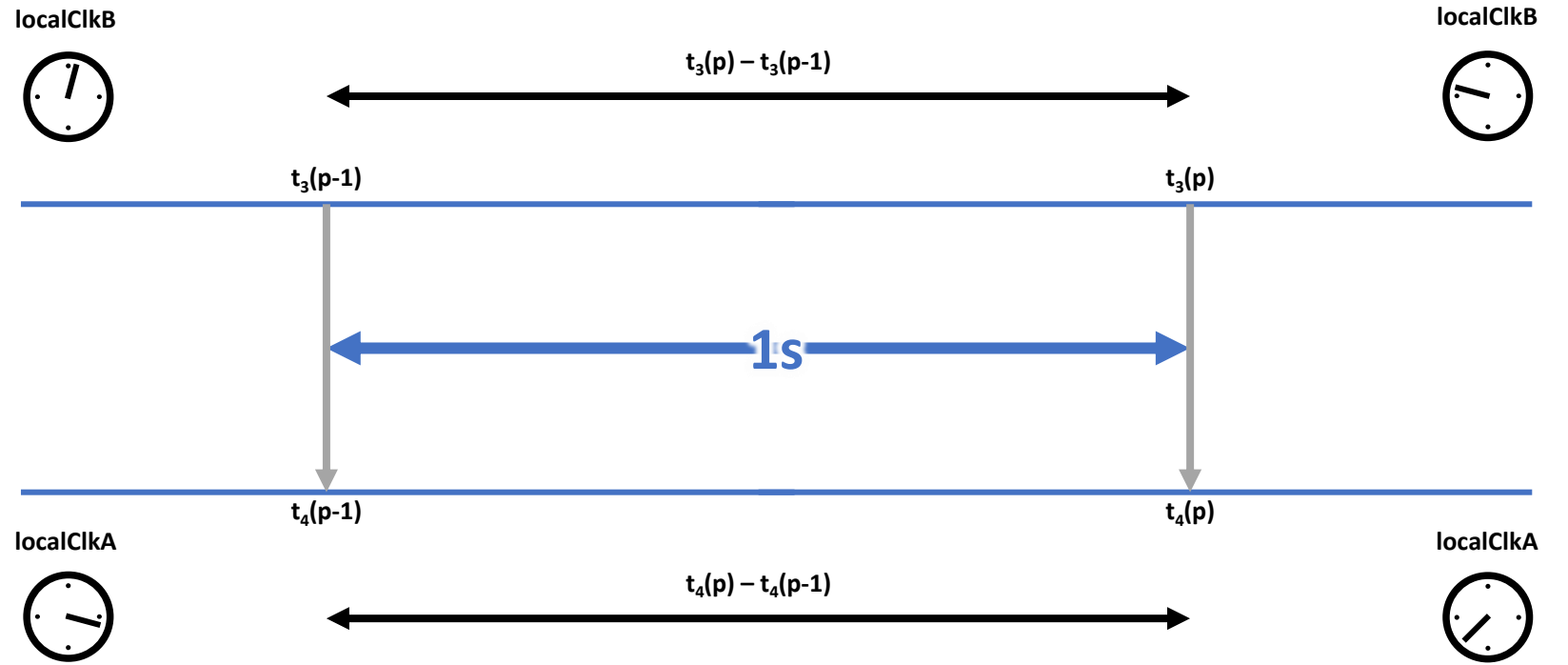
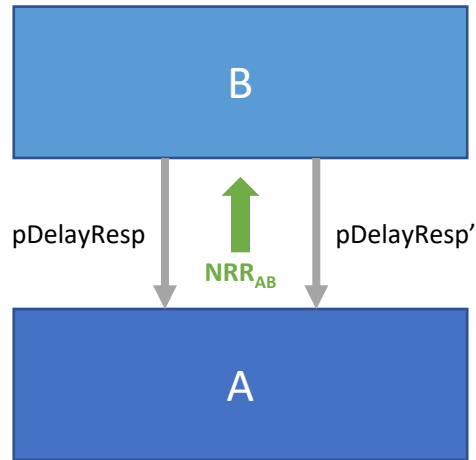
Compensating for Clock Drift

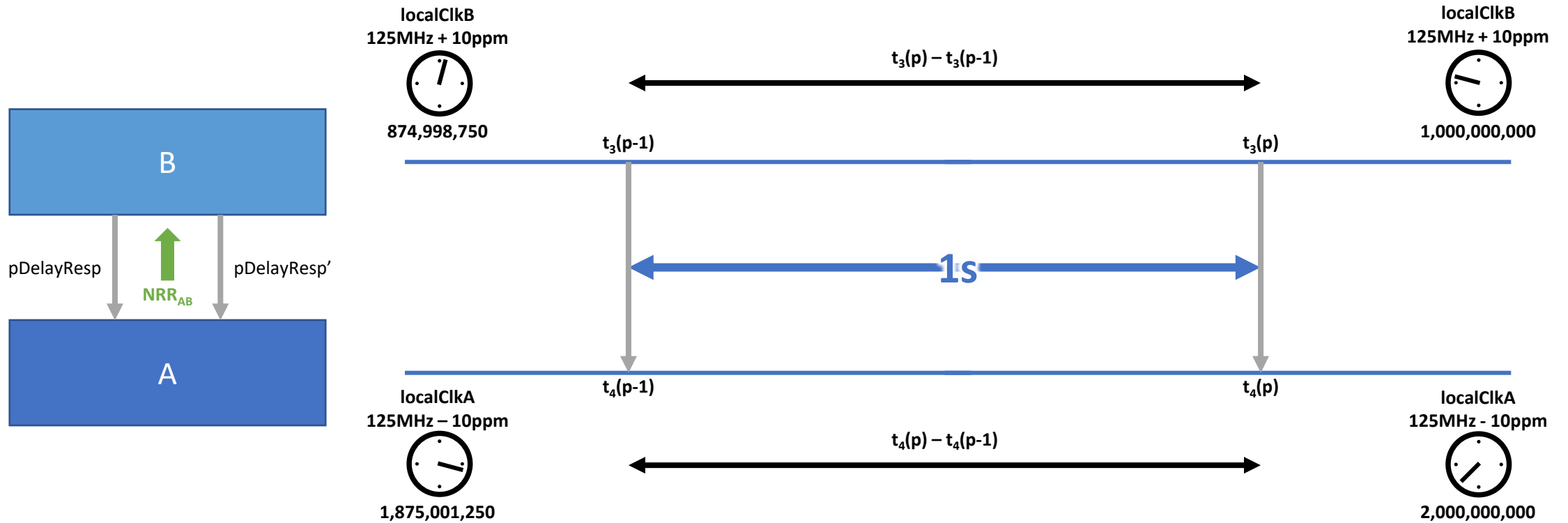
- Very hard to measure clock drift of an individual device...
...but we don't care about the clock drift of an individual device
- We care about Neighbor Rate Ratio and Rate Ratio...and we are constantly measuring both
- We can use these measurements to track clock drift between two devices and create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

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

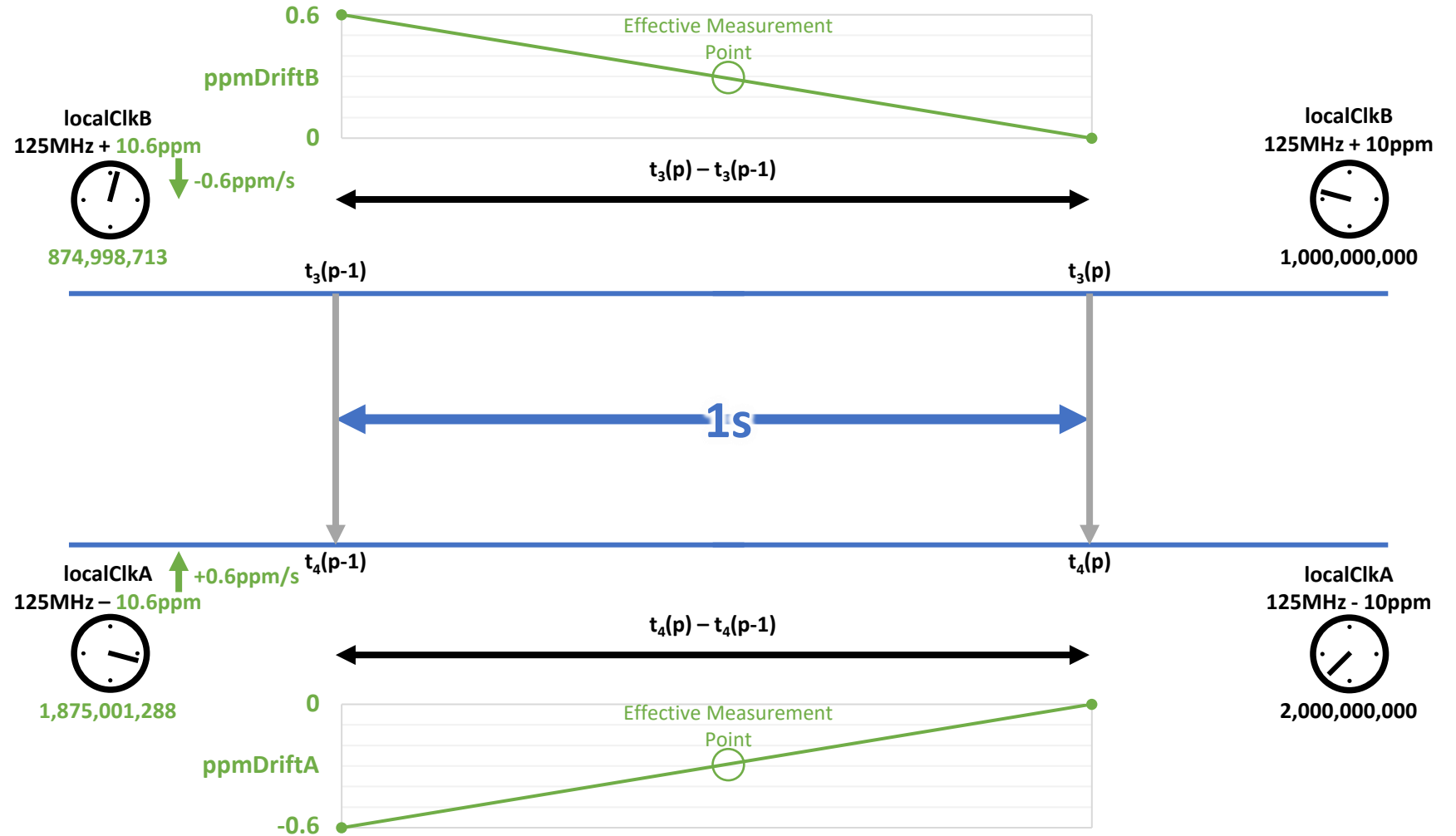
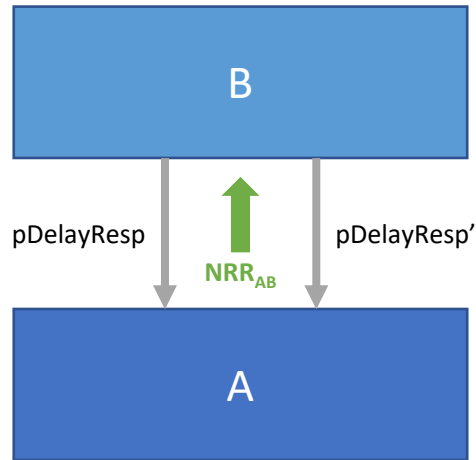
Maximum delay between attempt to measure NRR and using NRR as part of Sync messaging

Maximum delay between attempt to measure NRR and effective measurement point

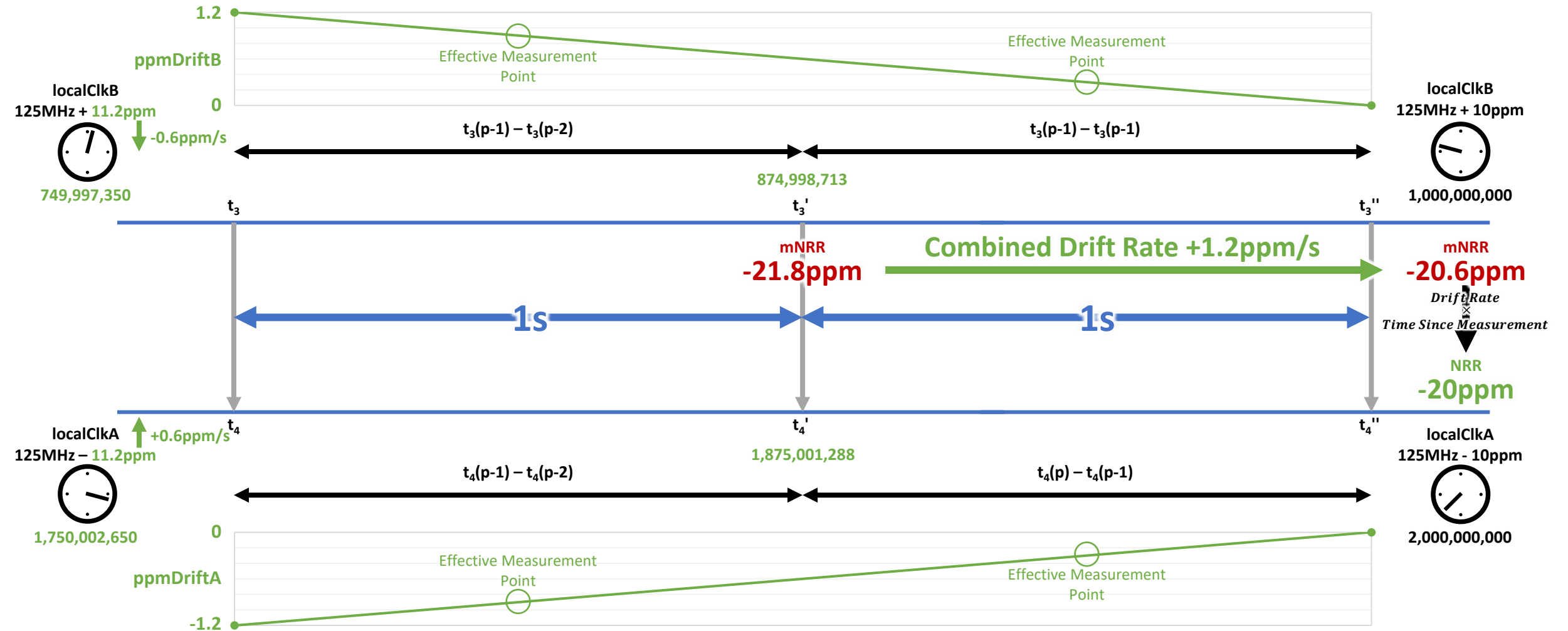




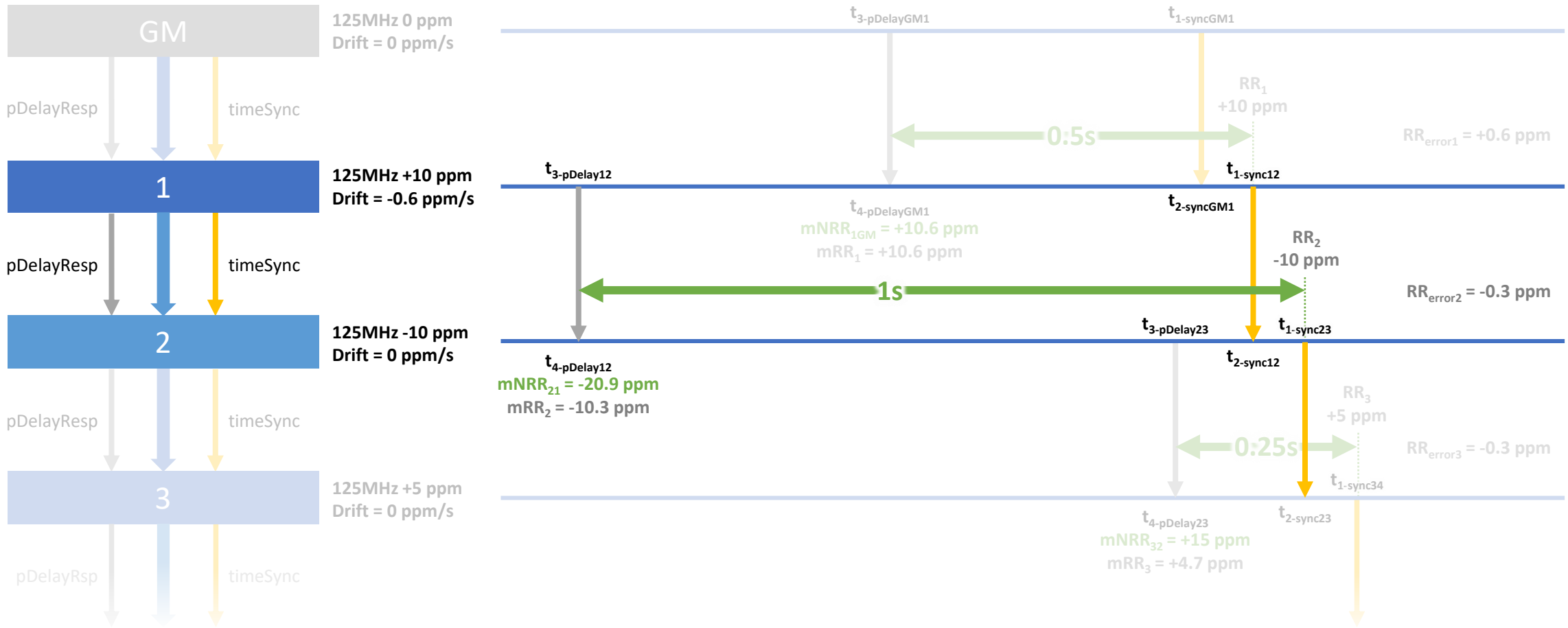
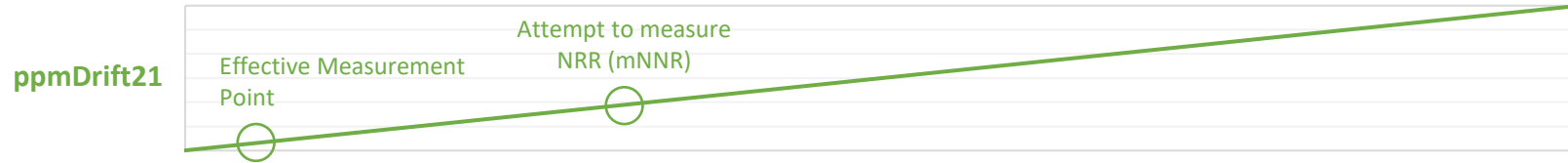
$$NRR_{AB} @ t_4' = \frac{(t_4(p) - t_4(p-1))}{(t_3(p) - t_3(p-1))} = \frac{124,998,750}{125,001,250} \rightarrow -0.002\% = -20 \text{ ppm}$$



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



Combined Drift Rate



Clock Drift Compensation – Possible Algorithm – 1

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

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

- 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)...

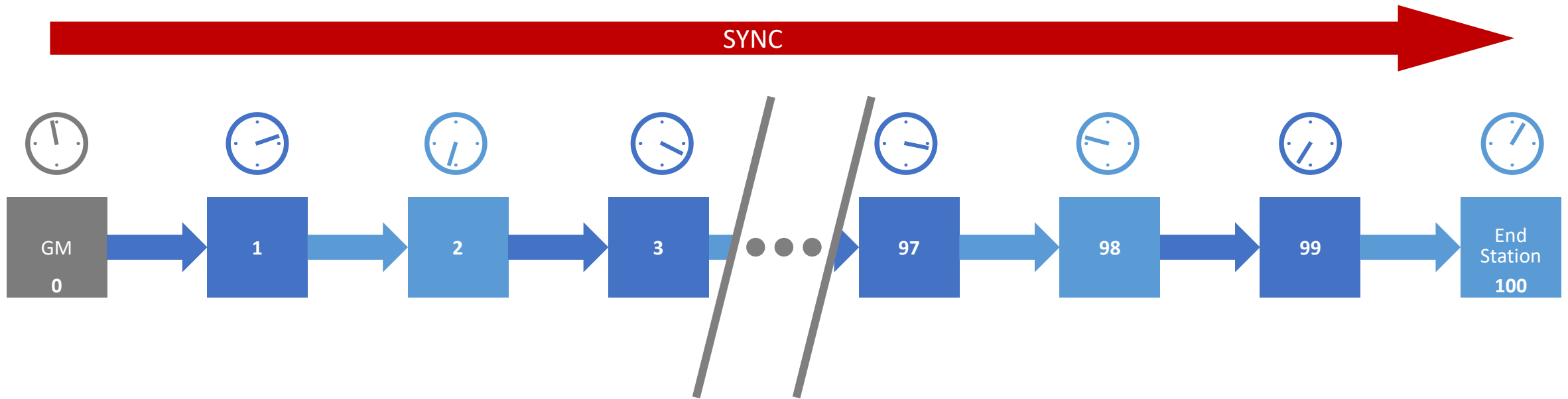
$$NRR_{sync} = mNRR + NRR_{driftRate} \left(delay_{mNRR_sync} + \frac{pDelayInterval \times N}{2} \right)$$

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

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

What should go into the spec?

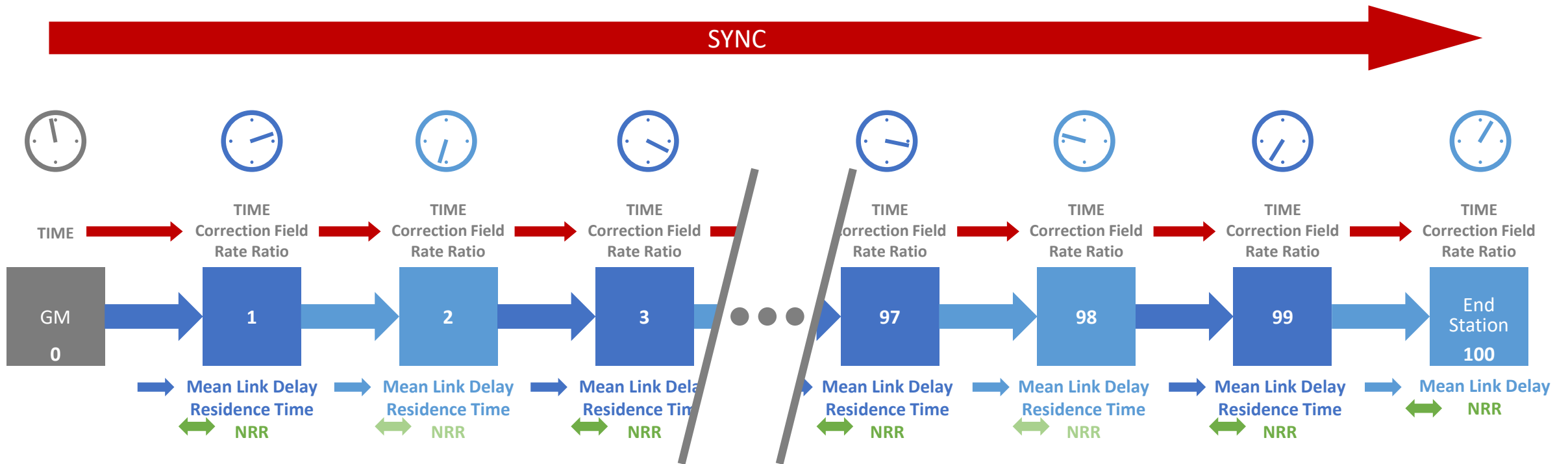
Time Sync



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?

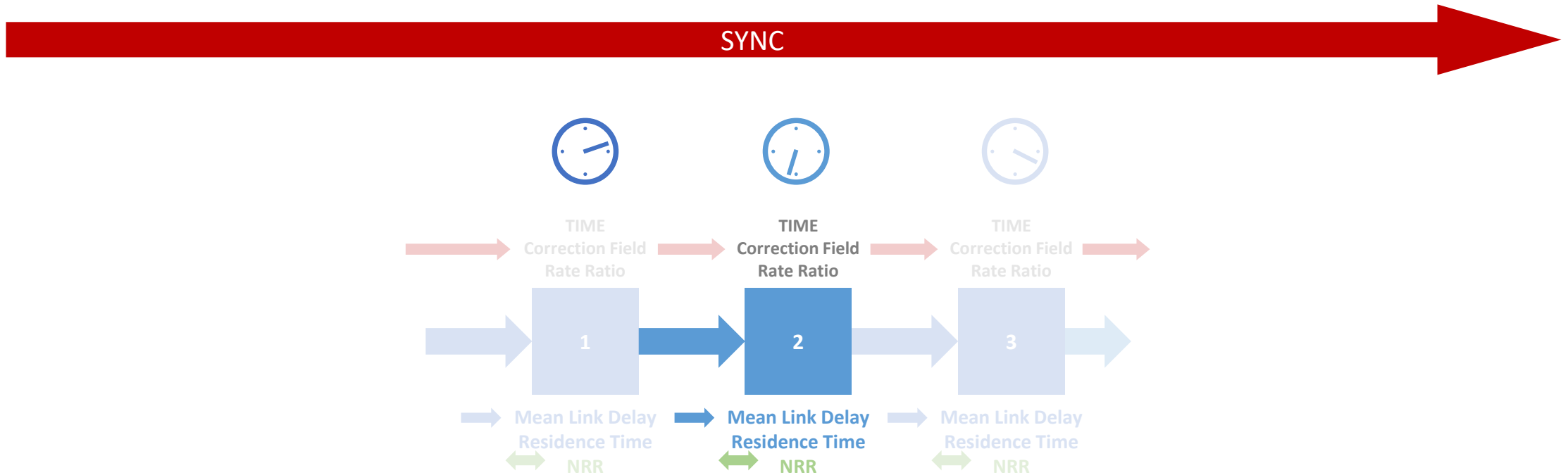
Time Sync



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



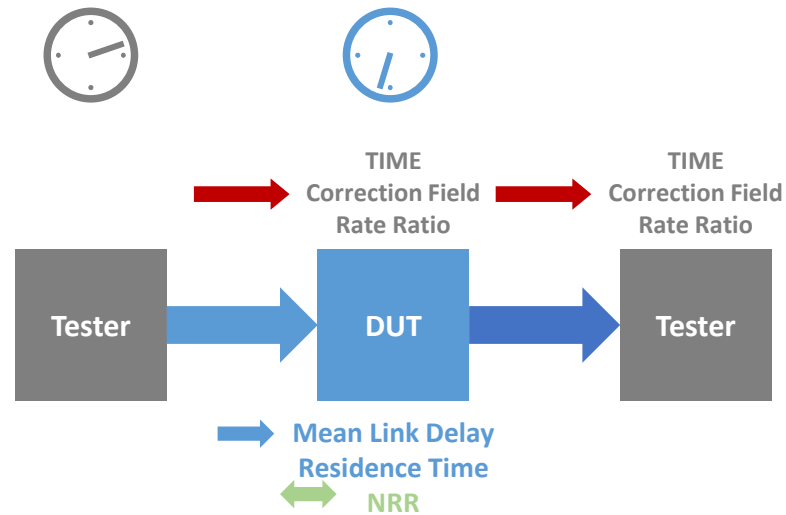
Each node must not add more error into the Correction Field and Rate Ratio of a Sync message than the system can handle.

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 not proscribe algorithms. Implementation is hard to verify; we ultimately don't care about the specific algorithm but rather the end effect; and it may be an area for innovation and differentiation.

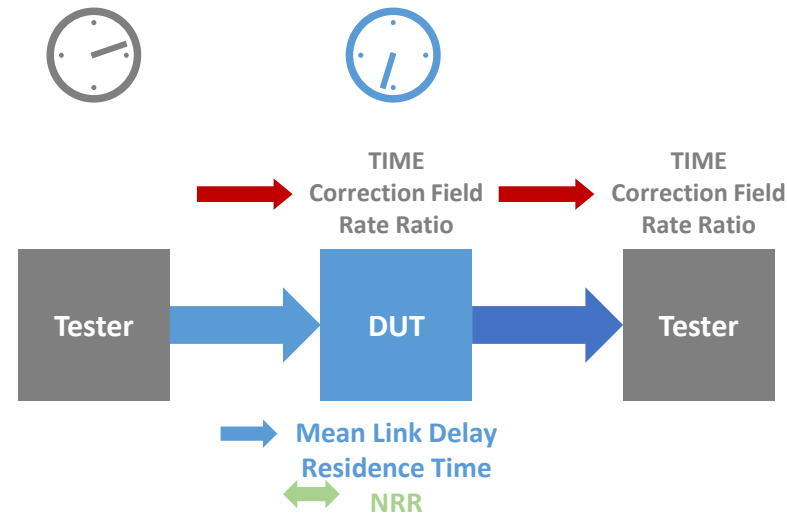
Testing Clock Stability



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



Tester measures Mean Link Delay & Residence Time;
alters Rate Ratio over time according to 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 – 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...
 ± 4 ns (uniform linear distribution)
- Initially remove “take the median of past M calculations” step from mNRR Smoothing.
 - Retain the option to use timestamp from Nth previous pDelayResp message when calculating mNRR (aka mNRR Smoothing)
 - Potentially add back “take the median of past M calculations” following further analysis

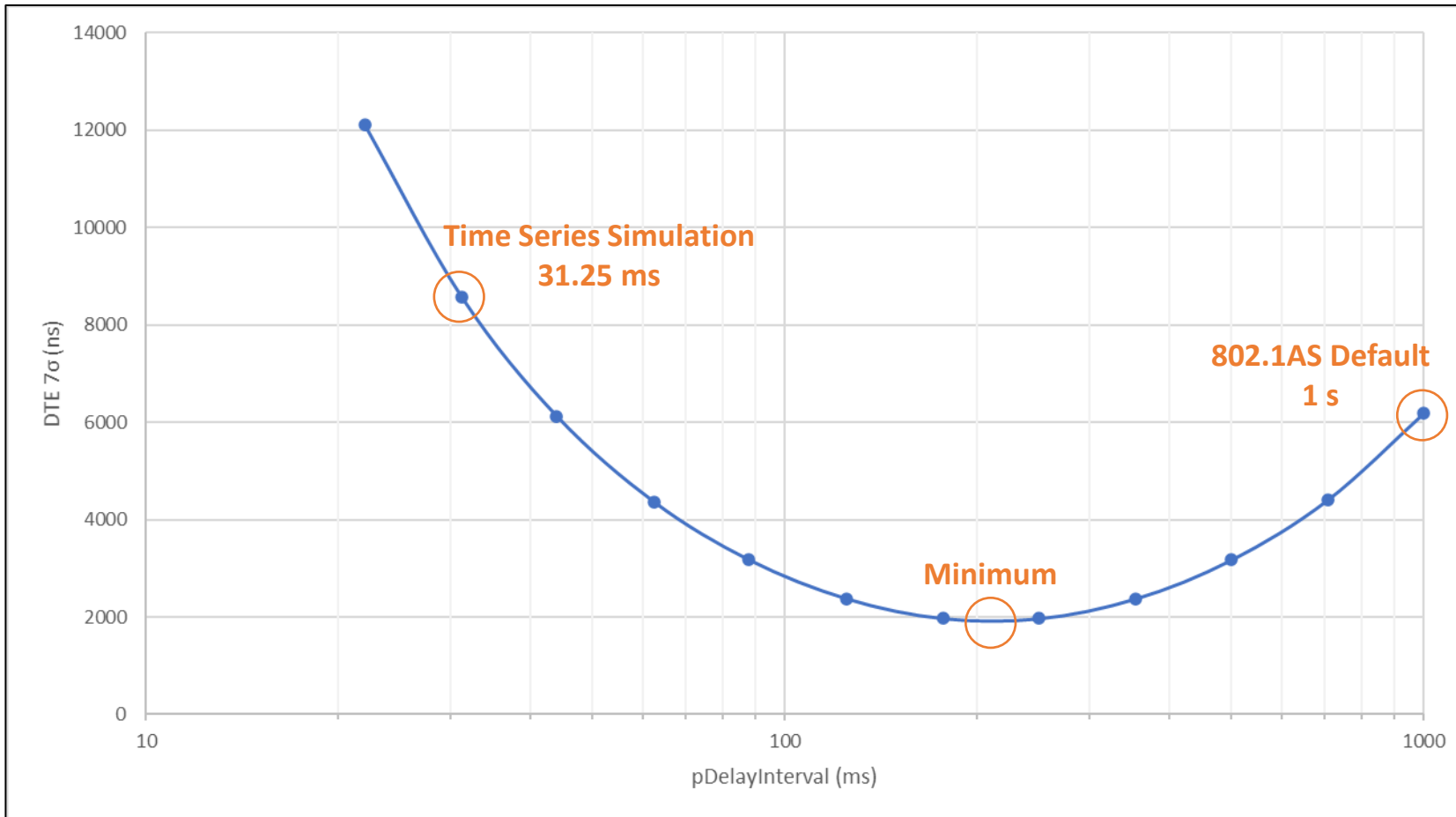
Proposed Time Series Simulations – Details

Experiment	Reason	Errors				Parameter	Correction Factors	
		GM Clock Drift (\pm ppm/s)	Other Clock Drift (\pm ppm/s)	Timestamp Granularity (\pm ns)	Dynamic Timestamp Error (\pm ns)	pDelay Interval (ms)	Mean Link Delay Averaging	mNRR Smoothing 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 pDelayInterval - 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 (\pm ppm/s)	Other Clock Drift (\pm ppm/s)	Timestamp Granularity (\pm ns)	Dynamic Timestamp Error (\pm ns)	pDelay Interval (ms)	Mean Link Delay Averaging	mNRR Smoothing Factor N
N	Verify possibility of reaching goal with 90% effective Clock Drift	0.06	0.06	4	4	250	On	6
O	Compenstation plus Mean Link Delay Averaging & mNRR Smoothing					31.25		48
P	Verify possibility of reaching goal with 98% effective Clock Drift	0.012	0.012	4	4	250	On	13
Q	Compenstation plus Mean Link Delay Averaging & mNRR Smoothing					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



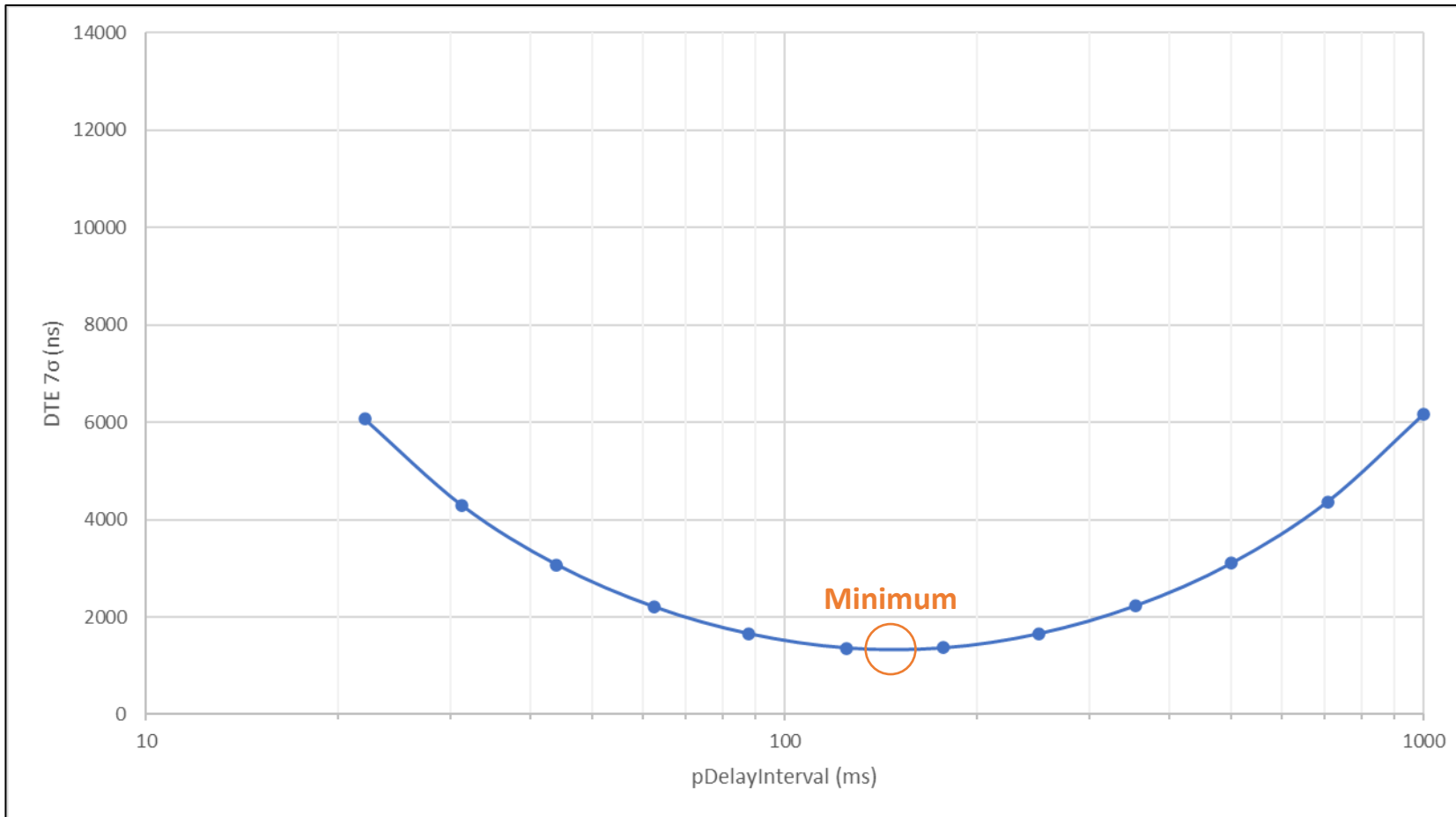
Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	+0.6	ppm/s
Clock Drift (non-GM)	0.6	±ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay Interval	Variable	ms
pDelay Response Time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

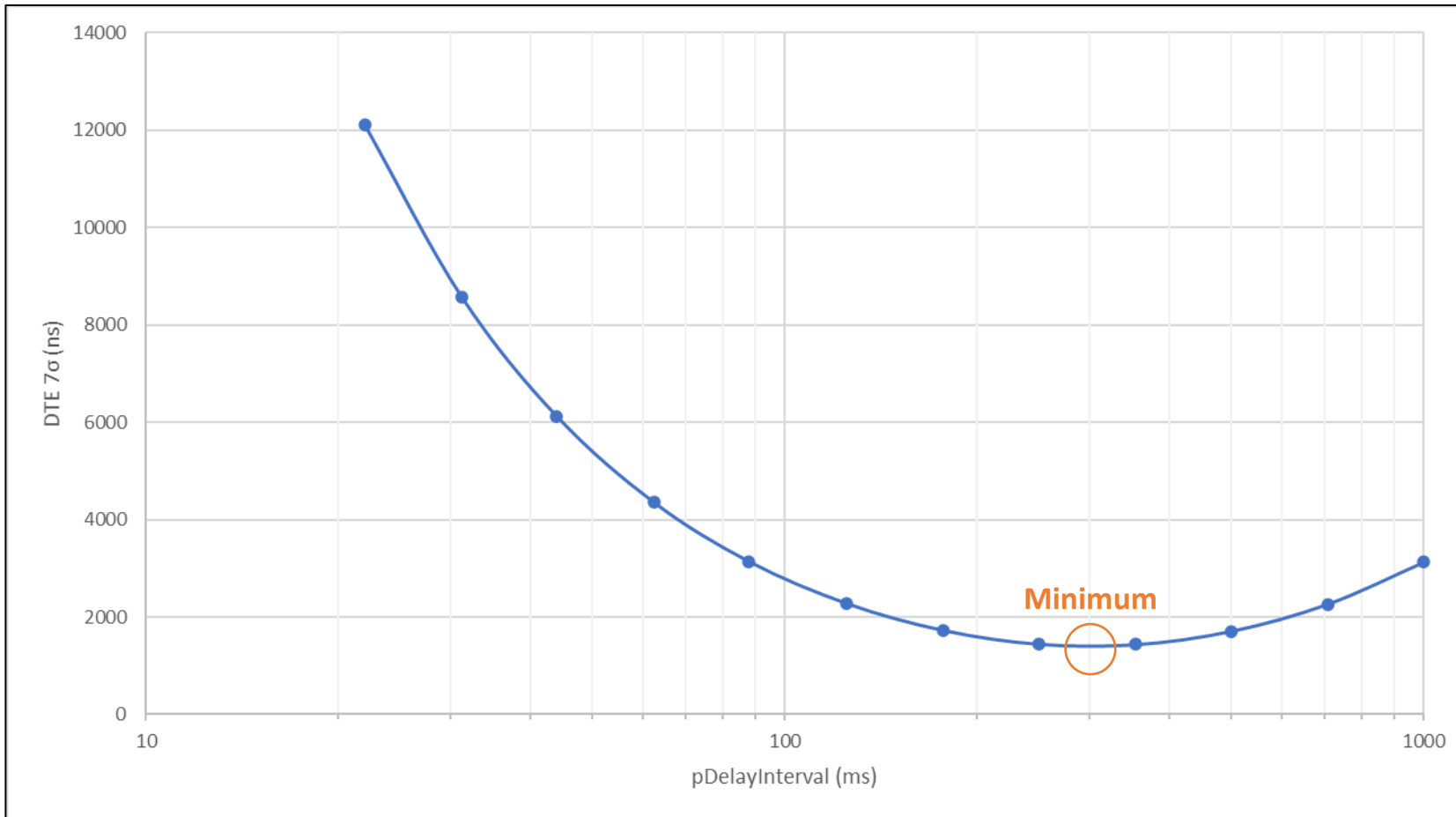


Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	+0.6	ppm/s
Clock Drift (non-GM)	0.6	±ppm/s
Timestamp Granularity TX	2	±ns
Timestamp Granularity RX	2	±ns
Dynamic Time Stamp Error TX	2	±ns
Dynamic Time Stamp Error RX	2	±ns
Input Parameters		
pDelay Interval	Variable	ms
pDelay Response Time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

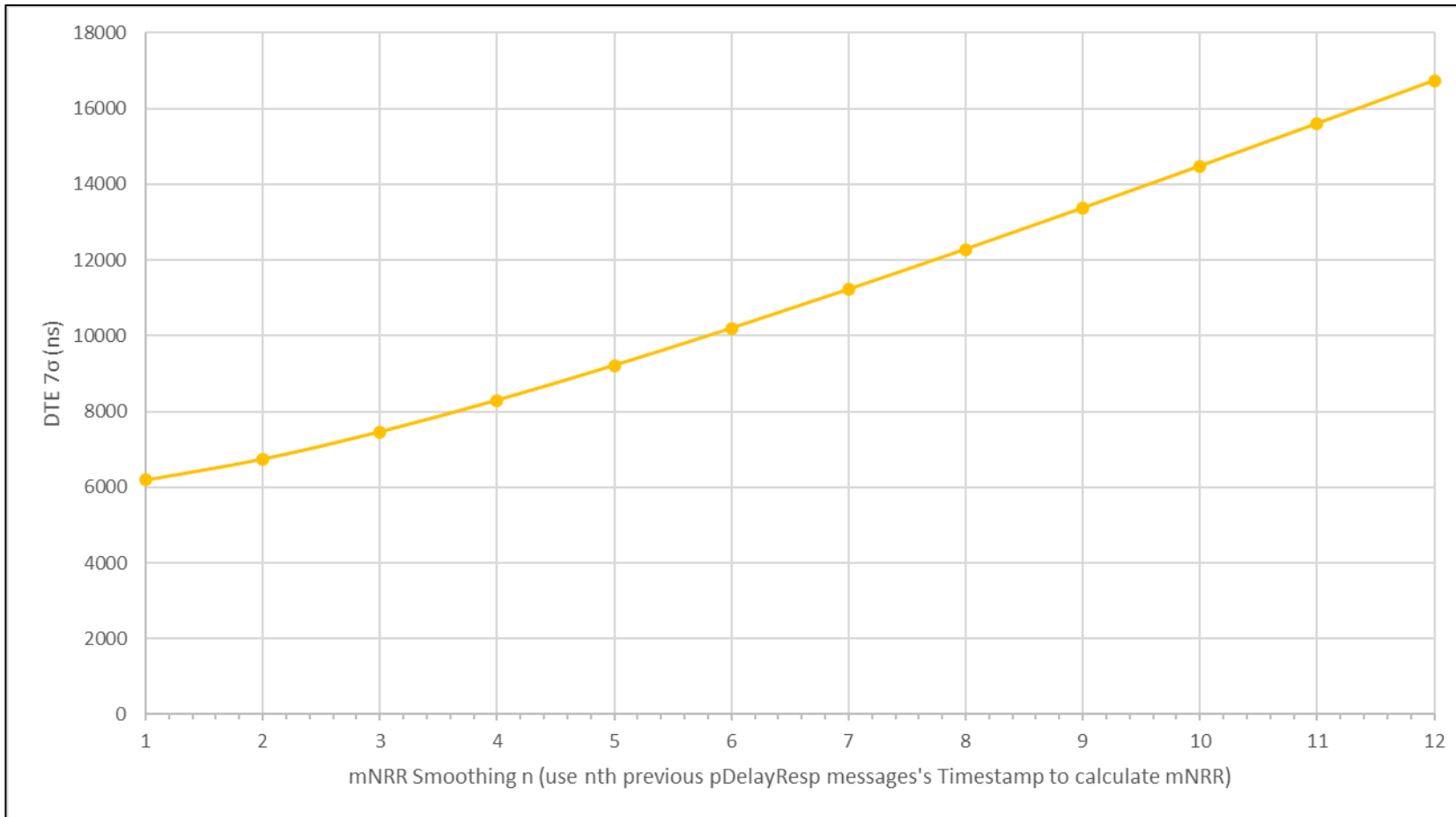


Input Errors		
GM Clock Drift Max	+0.3	ppm/s
GM Clock Drift Min	+0.3	ppm/s
Clock Drift (non-GM)	0.3	±ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay Interval	Variable	ms
pDelay Response Time	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.

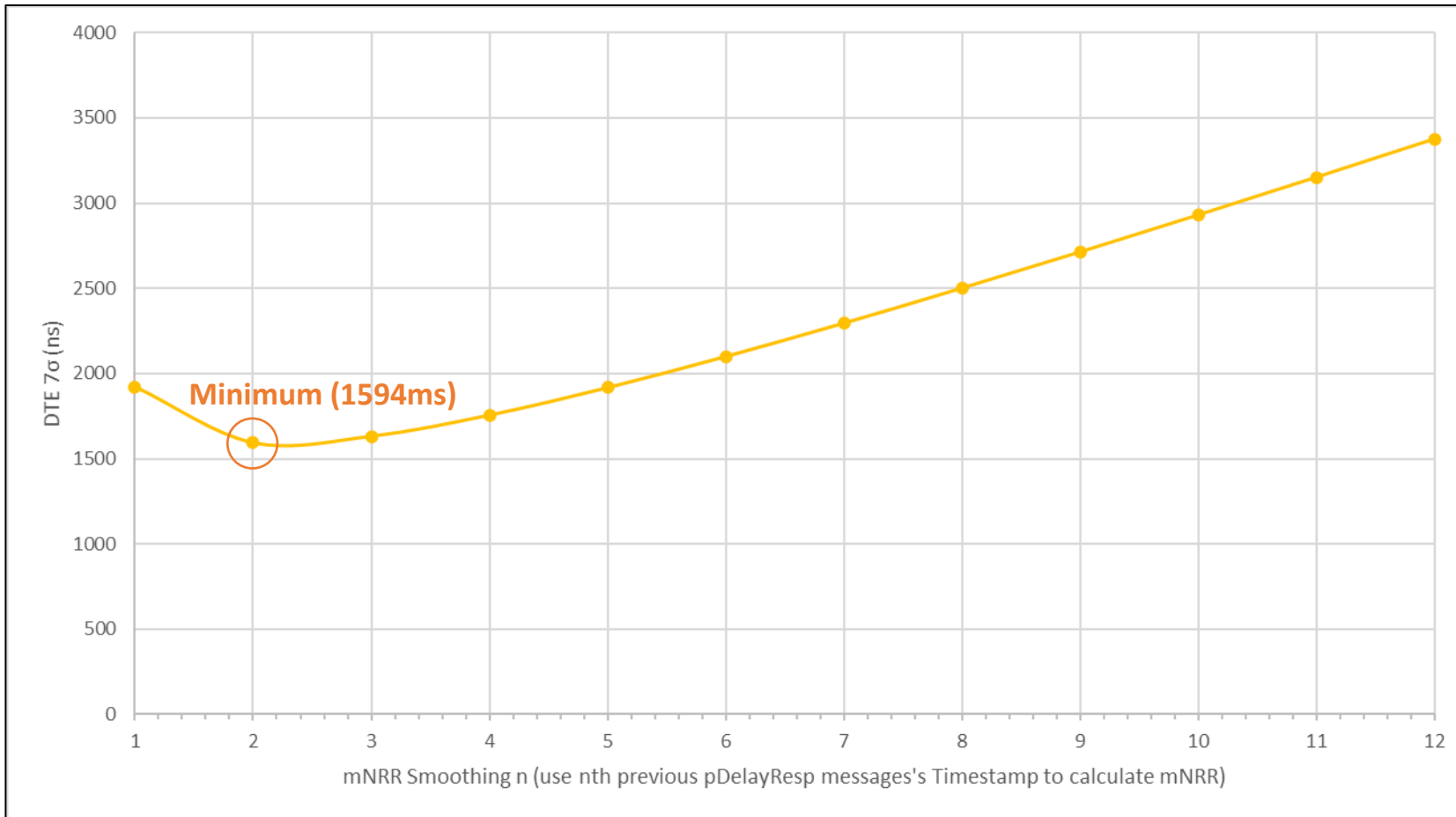
H & I: Validate mNRR Smoothing – 1s pDelay Interval



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 ($RT_{errorRR_NRR_CD}$)

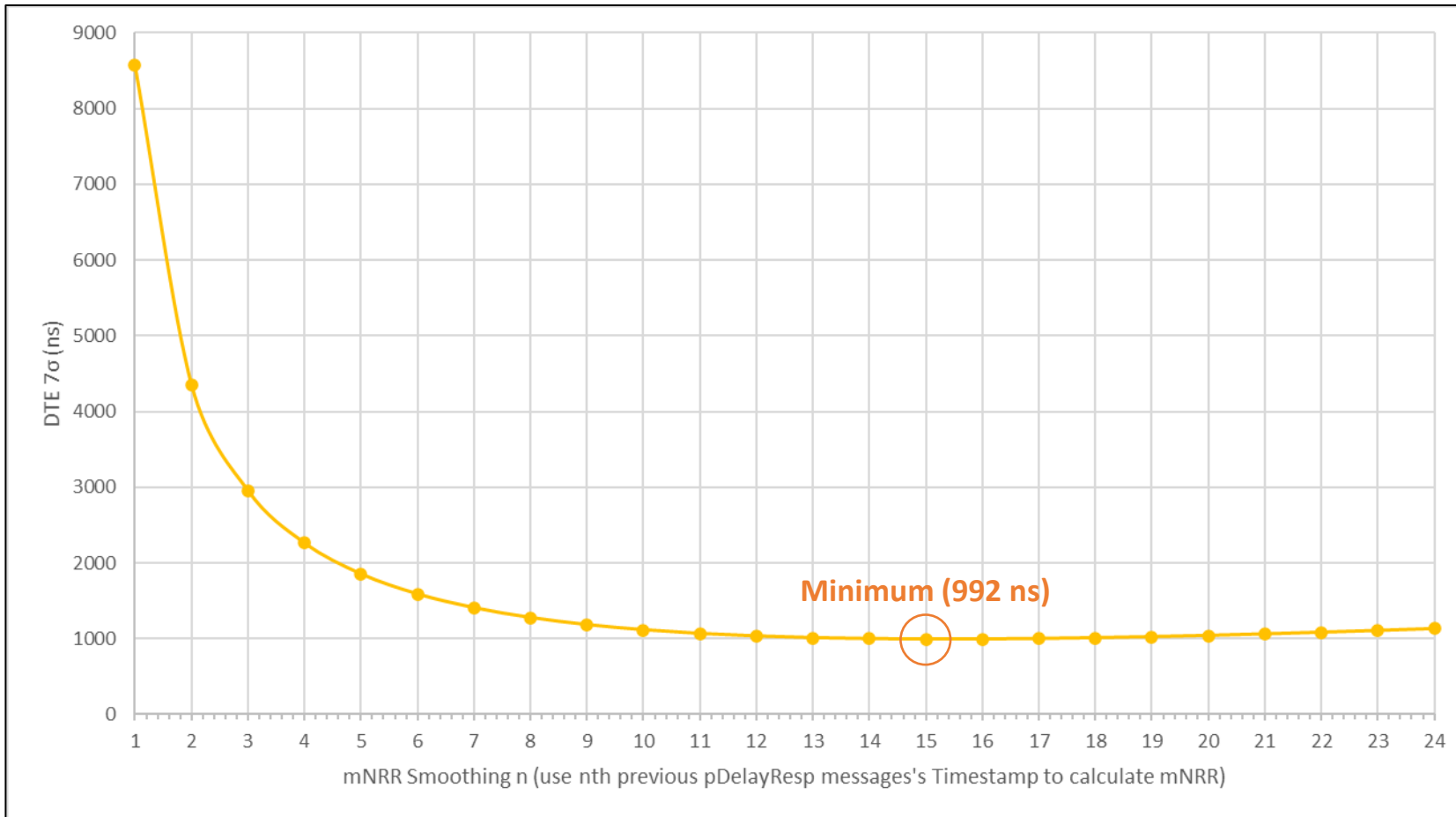
J & K: Validate mNRR Smoothing – 250ms pDelay Interval



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



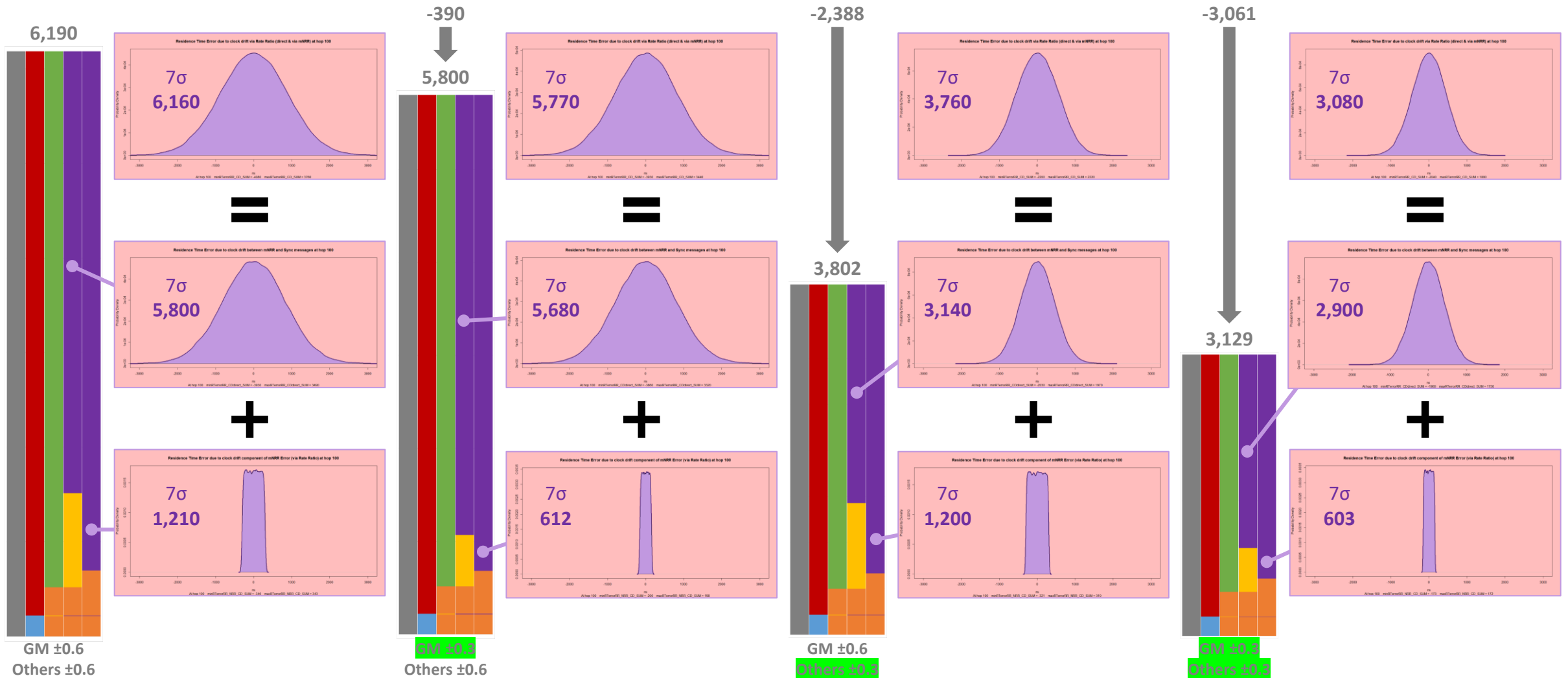
Input Errors		
GM Clock Drift Max	+0.6	ppm/s
GM Clock Drift Min	-0.6	ppm/s
Clock Drift (non-GM)	0.6	±ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay Interval	31.25	ms
pDelay Response Time	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).

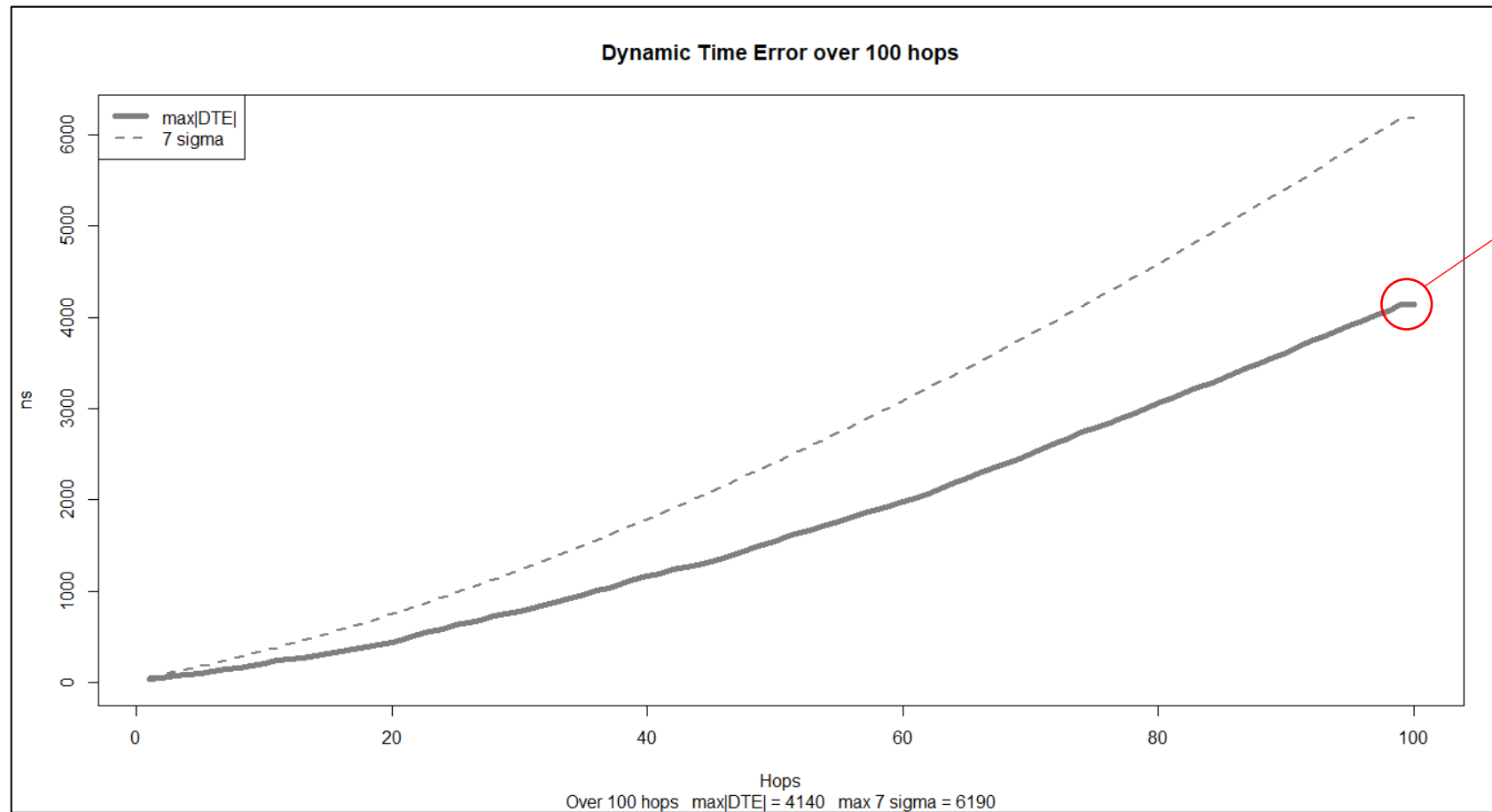
Thank you!

Back Up Material

Clock Drift Sensitivity



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.