60802 Dynamic Time Sync Error – Error Model & Monte Carlo Method Analysis

David McCall & Kevin Stanton (Intel)

March IEEE 802 Plenary – 802.1 TSN – IEC/IEEE 60802

Abstract

- The Monte Carlo Analysis approach to modelling Dynamic Time Error (DTE) across long chains of networked devices was developed to assist the IEC/IEEE 60802 group meet the target of 1us Time Sync Error across 100 hops.
- Previous presentations...
  - 60802-McCall-et-al-Time-Sync-Error-Model-0921-v03.pdf
  - 60802-McCall-Stanton-Time-Sync-Error-Model-and-Analysis-2021-11-v02.pdf
  - 60802-McCall-Stanton-Time-Sync-Error-Model-and-Analysis-0222-v03.pdf
  - 60802-McCall-Stanton-Time-Sync-Error-Model-and-Analysis-0322-v01.pdf
  - 60802-McCall-Time-Sync-Monte-Carlo-Results-for-Time-Series-Comparison-0322-v01.pdf
- In this contribution we:
  - Provide an updated on the Monte Carlo Analysis: the overall approach; the error algorithms; algorithmic compensation factors; and RStudio script made since November 2021.

Content

- Overview
  - Goals; Approach; Scope; Trade-offs
- Which Errors are Modelled & How They Add Up
- Error Algorithms (including Error Correction Factors)
- Potential Error Correction Algorithms
- RStudio Script Overview
- Data Output & Analysis

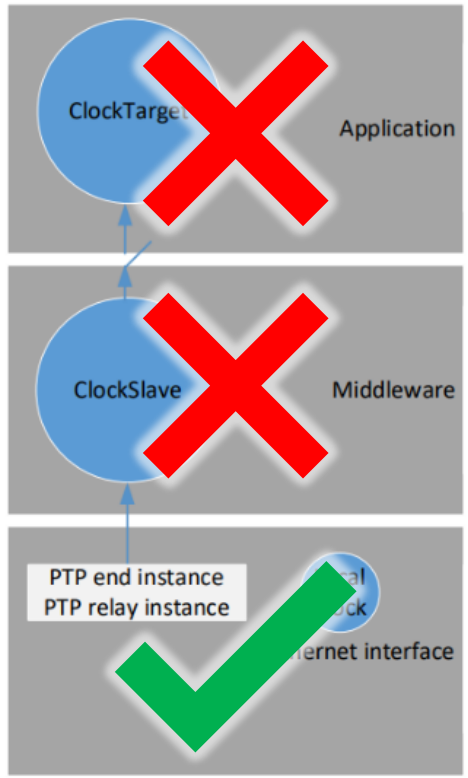
Overview

Goals of Monte Carlo Analysis

- The Monte Carlo analysis is intended as an addition to the toolbox, not an alternative to Time Series simulation.
- It provides…
  - The ability to iterate much faster
  - Greater insight into the source of errors and how they accumulate
  - Greater confidence that when selecting parameters to achieve a desired goal
  - Input into future Time Series simulations

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. **It is not a Time Series analysis.**
  - 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, plus the time between Sync messages.
  - It models each node, including the GM, as a single clock...or rather, errors associated with a single clock (for non-GM nodes: Local 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.

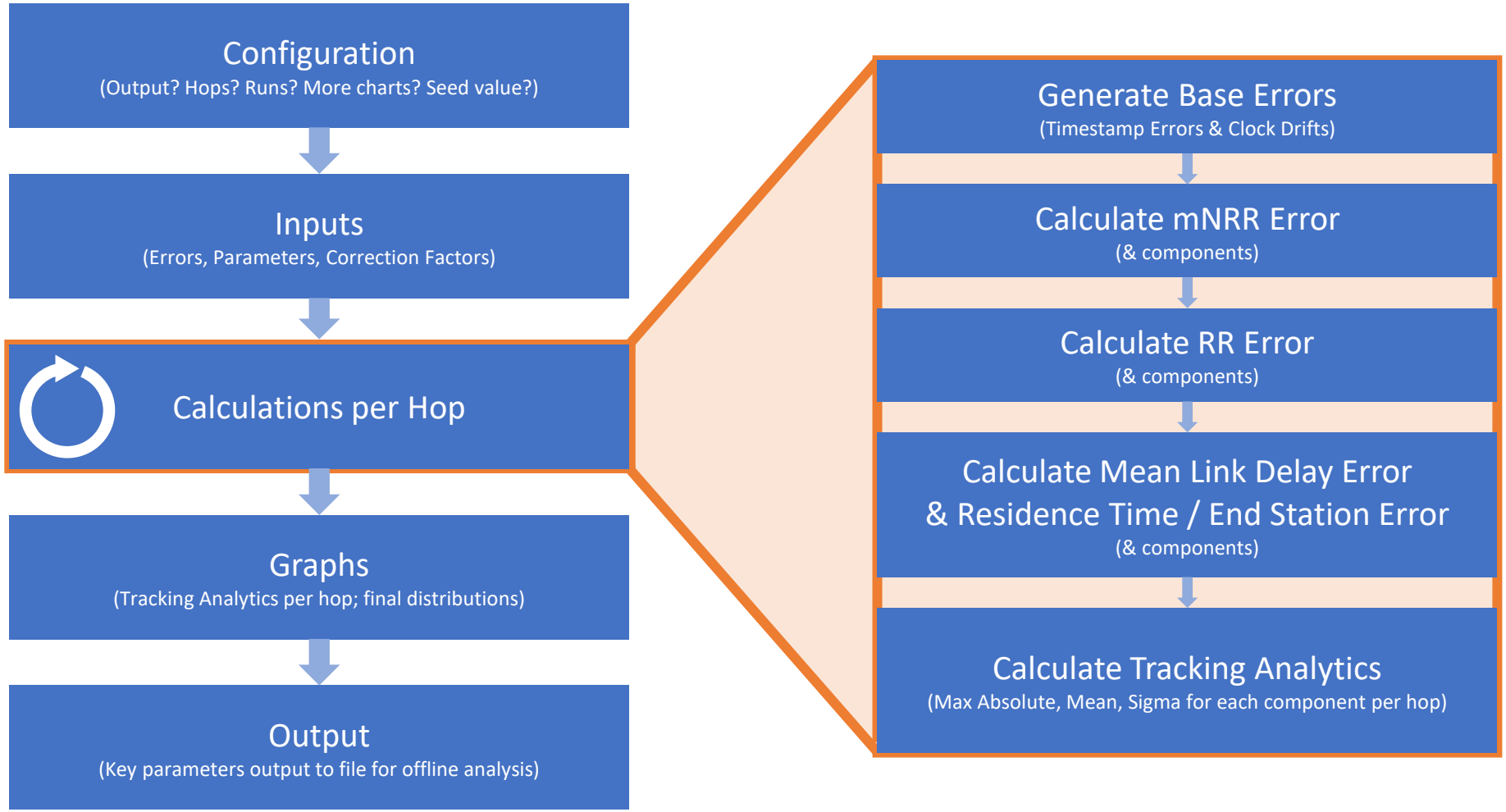


* https://www.ieee802.org/1/files/public/docs2021/60802-Steindl-ClockTarget-and-ClockSource-1121-v05.pdf

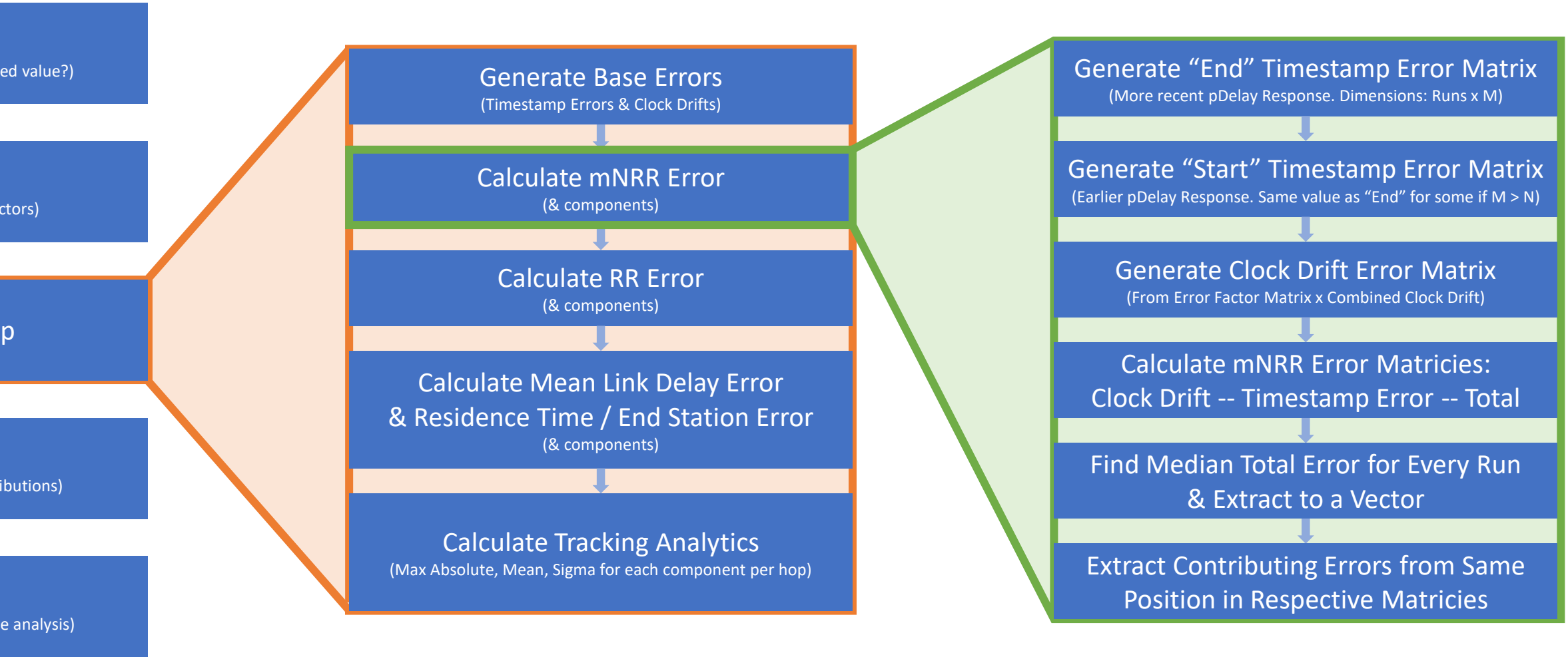
Approach

- The script carries out a number of Runs in parallel
  - Typically 100,000 to 10,000,000
- Each Run can be thought of as modelling the **major components** of **Dynamic Time Error** associated with a **single Sync message** as it passes from the GM, along a chain of bridges (typically nodes/hops 1 to 99) to an End Station (typically node/hop 100). This includes...
  - Errors associated with the measurement of NRR, which takes place at each node prior to Sync message processing, including errors due to the passage of time between measurement of NRR and Sync message processing
  - Errors that occur at the End Station before the arrival of the next Sync message
    - This is the only node where this factor is calculated; it effectively replaces the Residence Time calculation at the End Station (the Sync message is not passed on, so there is no Residence Time)
- Each Run is independent
- The script calculates all the errors for all the Runs at hop 1 before proceeding to hop 2, etc...

RStudio Script Summary



RStudio Script Summary – Median of NRR



Tradeoffs: Adding ppm

- 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).

Tradeoffs: Simplifications

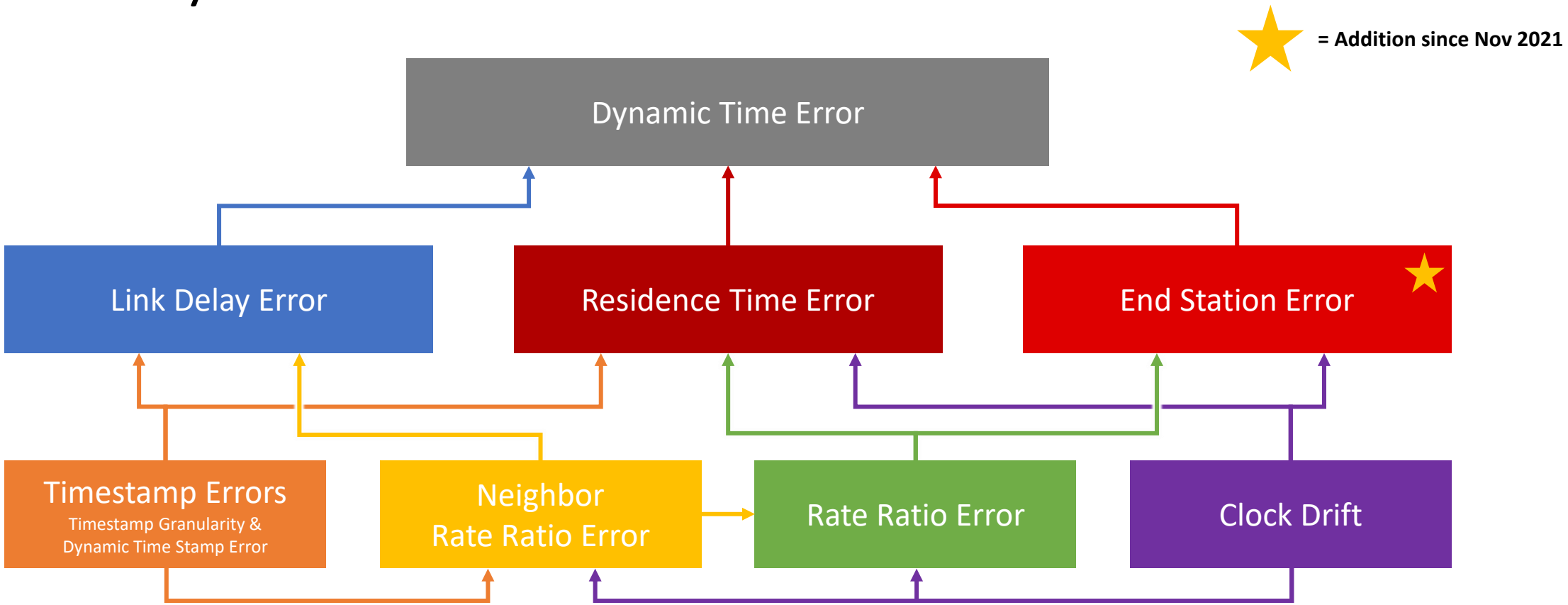
- Mean Link Delay error due to Rate Ratio error is ignored
  - During Sync message processing at each node, Residence Time and Mean Link Delay are added together then multipled by Rate Ratio prior to addition to the Correction Field. Errors in Rate Ratio creates errors in the Correction Field.
  - However, Link Delay is typically <100ns; often <10ns. Residence Time is typically 10ms, i.e. 4 or 5 orders of magnitude larger.
  - Modelling the component of Link Delay Error due to Rate Ratio would also make the calculation of components of Dynamic Time Error much more complicated (and time consuming) for little benefit.
- Clock Drift is assumed to be linear
  - Although the model is not a time series simulation, the magnitude of some errors is linked to various time intervals and clock drift during them.
  - The model assumes that clock drift is linear (i.e. constant ppm/s) over the period of interest.
  - The "period of interest" varies based various factors
    - pDelayInterval and mNRRsmoothing → amount of time before Sync messaging
    - residenceTime → amount of time during Sync messaging
    - syncInterval → amount of time after Sync messaging
  - Typically "period of interest" is 1s to 5s, over which time an assumption of clock drift linearity is a reasonable approximation
  - Non-linearities will cause deviations vs. the model, but provided they aren't too large and/or present at too many nodes along the chain, the model will still be a good approximation

Tradeoffs: Correction Factors

- The model includes three correction factors to address specific sources of error...
  - $\text{MeanLinkDelay}_{\text{errorCorrection}}$
  - $\text{NRRdriftRate}_{\text{errorCorrection}}$
  - $\text{RRdriftRate}_{\text{errorCorrection}}$
- In a real system, or a Time Series Simulation, these would be algorithms that process multiple inputs over a period of time.
- This model isn't a Time Series Simulation, so it employs correction factors that approximate the effectiveness of a algorithm via a simple "percentage effective".
  - 90% effective means that 10% of the targeted error remains
- Potential algorithms to match each of the correction factors are described later in this presentation
  - The practical effectiveness of the algorithms, i.e. what percentage of the targeted error they can actually eliminate, must be determined via other means.

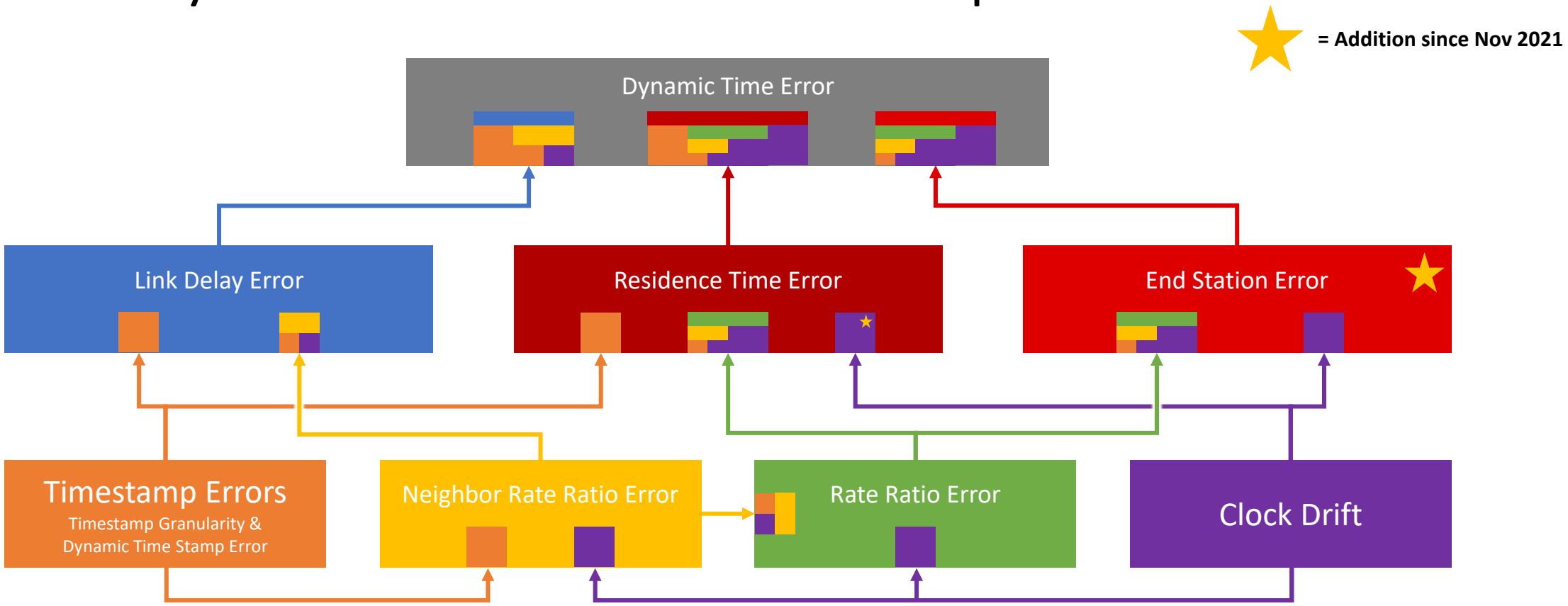
Which Errors to Model & How They Add Up

Time Sync – Errors to Model



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

Time Sync – How Errors Add Up



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

Monte Carlo Method Analysis

Overview of the modelling approach and analysis tool built in RStudio

Monte Carlo Method Analysis

- Inputs
- Errors, Formulae & Observations
- R & RStudio

Input Errors

Error	Distribution	Default Min	Default Max	Equation	Unit
$clockDrift_{GM}$ ($ClockDrift_{GMmin}$ & $ClockDrift_{GMmax}$)	Uniform	-1.5	+1.5	$clockDrift_{GM} \sim U(clockDrift_{GMmin}, clockDrift_{GMmax})$	ppm/s
$clockDrift$ ($ClockDrift_{min}$ & $ClockDrift_{max}$)	Uniform	-1.5	+1.5	$clockDrift \sim U(-clockDrift, +clockDrift)$	ppm/s
$TSGE_{TX}$	Uniform	-4	+4	$TSGE_{TX} \sim U(-TSGETX, +TSGETX)$	ns
$TSGE_{RX}$	Uniform	-4	+4	$TSGE_{RX} \sim U(-TSGERX, +TSGERX)$	ns
$DTSE_{TX}$	Uniform	-4	+4	$DTSE_{TX} \sim U(-DTSETX, +DTSETX)$	ns
$DTSE_{RX}$	Uniform	-4	+4	$DTSE_{RX} \sim U(-DTSERX, +DTSERX)$	ns

Formulae below take into account the unit of the input value.
Formulae in the main section of the September presentation did not.

Input Parameters

Error	Default Value	Unit
pDelayInterval	1,000	ms
syncInterval	125	ms
pDelayTurnaround	10	ms
residenceTime	10	ms

Input Correction Factor

Error	Default Value	Unit
$meanLinkDelay_{errorCorrection}$	0	Value (0-1)
$driftRateNRR_{errorCorrection}$	0	Value (0-1)
$driftRateRR_{errorCorrection}$	0	Value (0-1)
$pDelayRespSync_{correction}$	0	Value (0-1)
$mNRRsmoothingN$	1	Number (1+)
$mNRRsmoothingM$	1	Odd Number (1+)

All formulae in this analysis are ultimately composed of one of these inputs:

Input Errors (Clock Drift or Timestamp Errors)

Input Parameters

Input Correction Factors

Input Correction Factors

- This is not a time series analysis, so it's not practical to model the exact mechanism that might be used to implement compensation for errors.
- Three are modelled as an "effectiveness factor", e.g. a value of 0.75 (75%) means that only 25% of the relevant error will show up in DTE
  - Averaging of Mean Link Delay over time → $meanLinkDelay_{errorCorrection}$
  - Measuring and then compensating for Clock Drift → $driftRateNRR_{errorCorrection}$ & $driftRateRR_{errorCorrection}$
  - Aligning pDelay Response messages so they occur close to Sync message arrival → $pDelayRespSync_{correction}$

Input Correction Factors

- Using timestamps of the Nth prior pDelayResp messages to reduce $mNRR_{error}$ is modelled using the ***mNRRsmoothingN*** factor, which is a whole number with minimum 1.
  - The timestamp of the Nth pDelayResp message prior to the most recent one is used to calculate mNRR.
    - A value of 1 indicates the 1st message prior to the most recent one, i.e. no "smoothing".
  - See the section on $mNRR_{error}$ for more detail.
- Taking a median of the previous M prior NRR calculations to reduce $mNRR_{error}$ is modelled using the ***mNRRsmoothingM*** factor, which is an odd number with minimum 1
  - Values for NRR are calculated following the arrival of pDelayResp messages. The median results from the previous M calculations is used as mNRR.
    - A value of 1 indicates no median is taken, i.e. no "smoothing".
  - Which value ends up as the median has knock on effects for other errors, e.g. the further back in time the calculation took place, the greater the potential drift between clocks and consequent error from that factor
  - See the section on $mNRR_{error}$ for more detail.

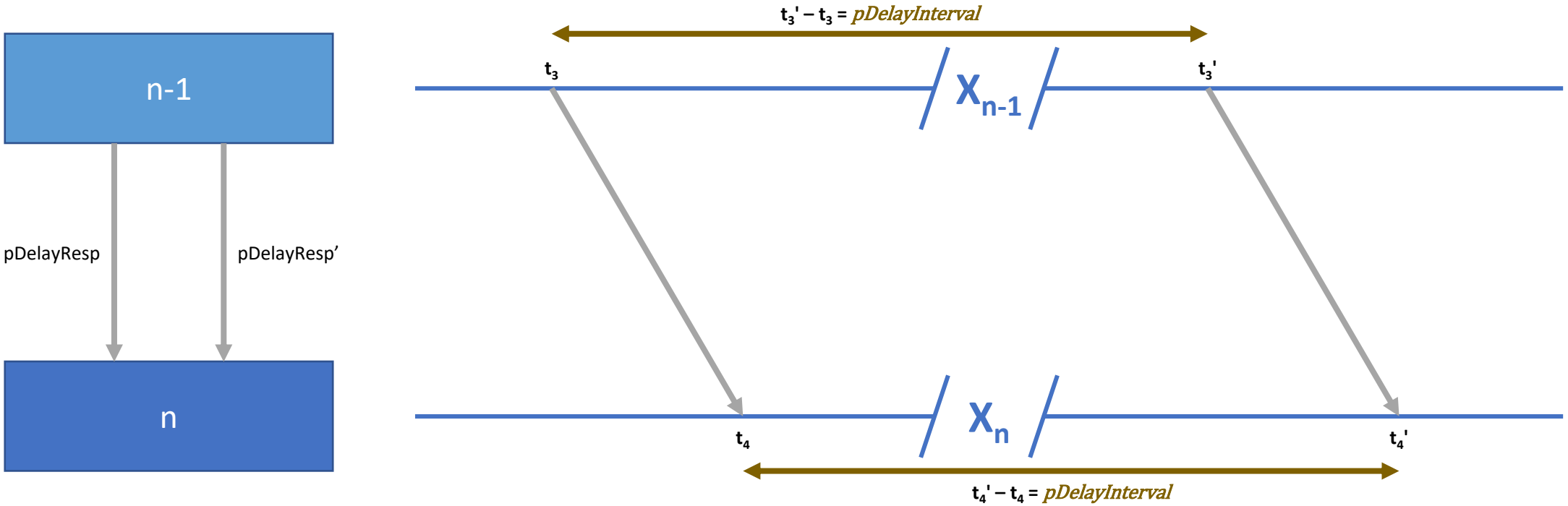
Error & Error Components – Formatting

Item	Examples using Residence Time	Residence Time errors due to...
Element	$residenceTime$	
Error in that Element	$residenceTime_{error}$	
Component of that error due to another Element	$residenceTime_{errorTS}$ $residenceTime_{errorTSdirect}$ $residenceTime_{errorCD}$ $residenceTime_{errorRR}$	All Timestamp Errors Direct Timestamp Errors All Clock Drift Errors Rate Ratio Errors
Next level down... Component of a component	$residenceTime_{errorRR\_TS}$ $residenceTime_{errorRR\_CDdirect}$ $residenceTime_{errorRR\_NRR}$	All Timestamp components of Rate Ratio Errors Direct Clock Drift component of Rate Ratio Errors NRR error component of Rate Ratio Errors
Next level down... Component of a component of a component	$residenceTime_{errorRR\_NRR\_TS}$ $residenceTime_{errorRR\_NRR\_CD}$	Timestamp component of NRR via Rate Ratio Errors Clock Drift component of NRR via Rate Ratio Errors

mNRR Error

Except taking a median of past NRR calculations, which is considered in the next section.

Measured Neighbor Rate Ratio (mNRR)

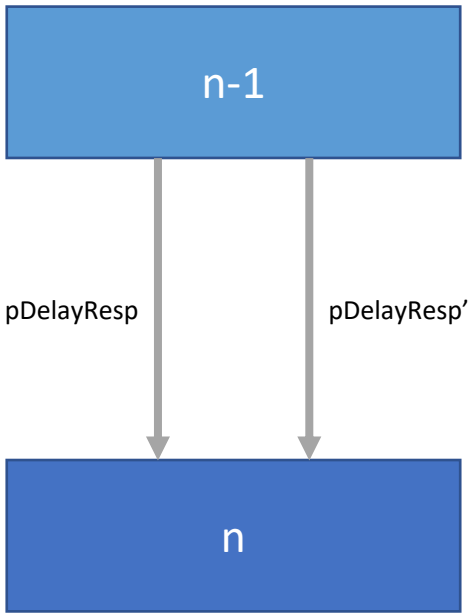


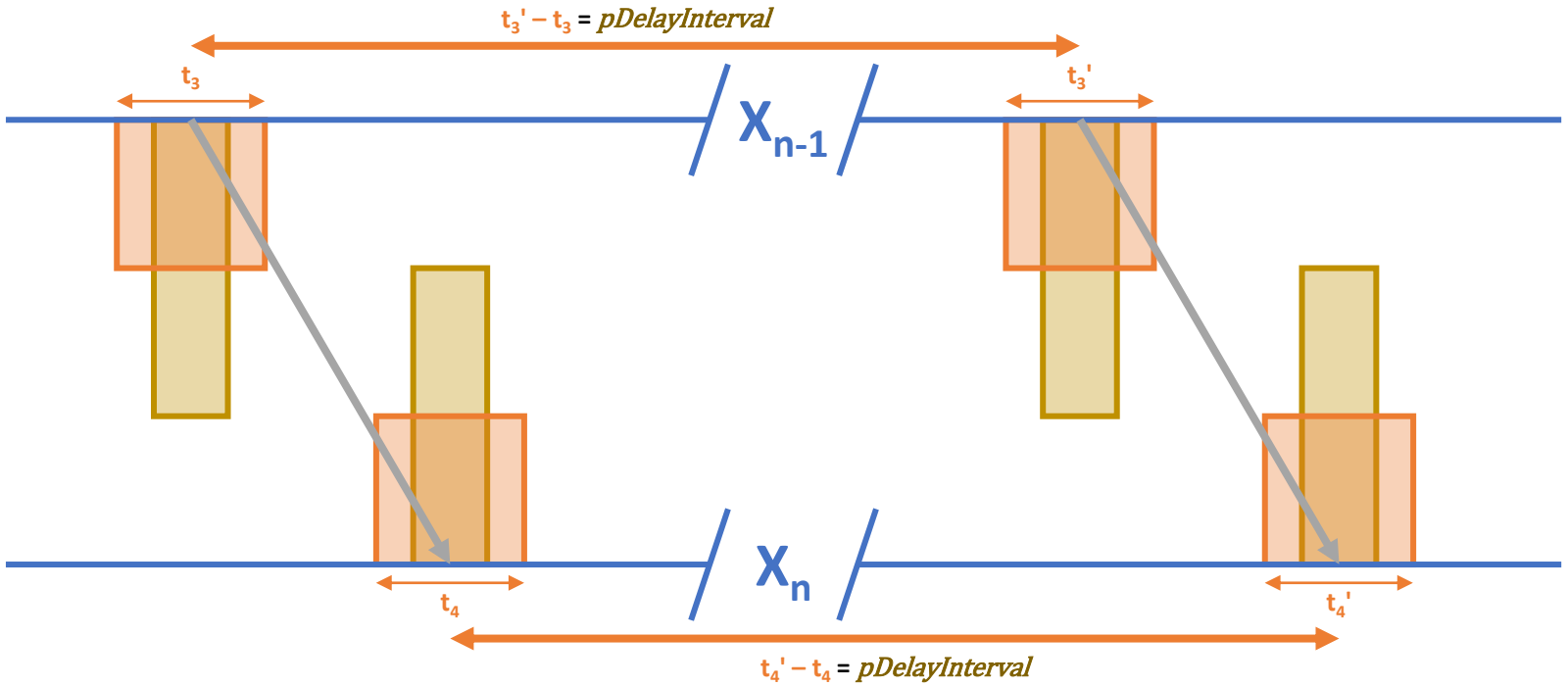
$$mNRR = \left(\frac{(t_4' - t_4)}{(t_3' - t_3)}\right)$$

$$mNRR_{error} = mNRRerrorT_S + mNRRerrorC_D$$

ppm

$mNRR_{errorTS}$

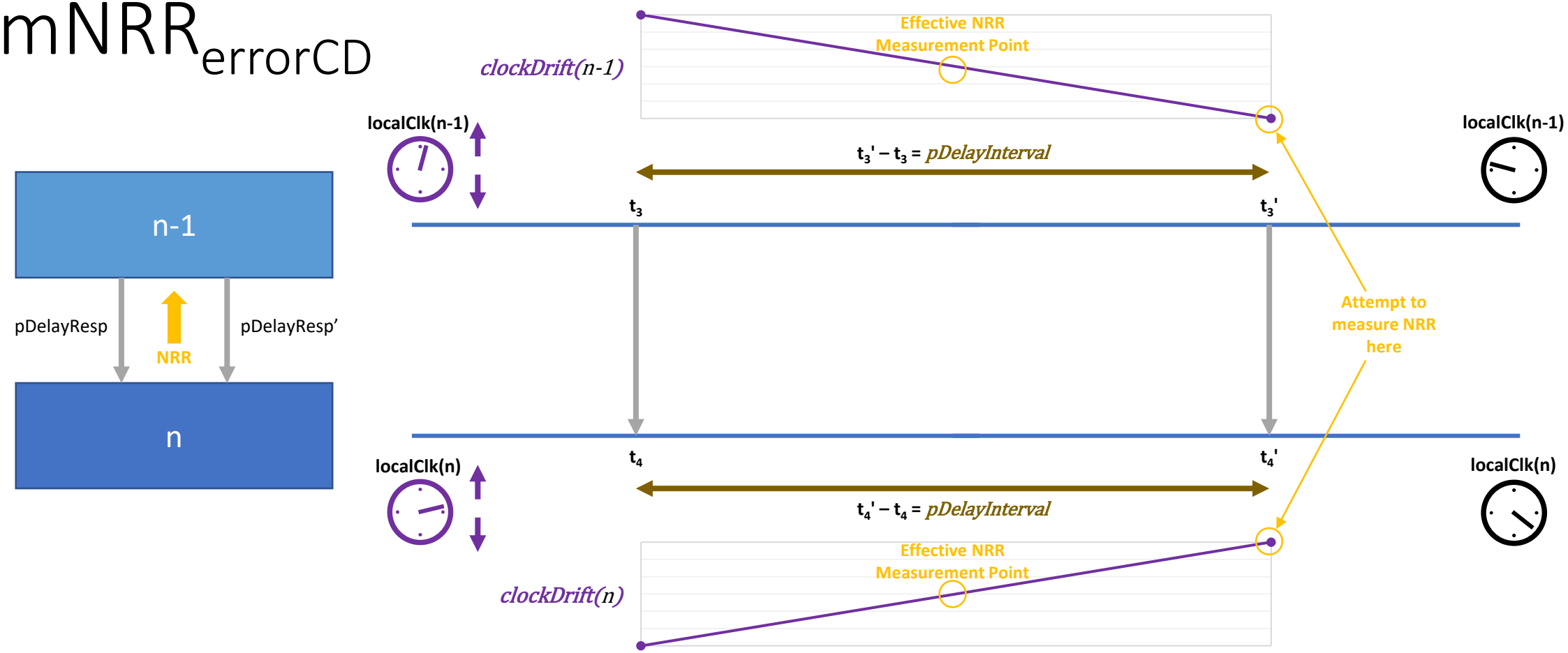




$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror} - t_{4PDerrorPrevious}) - (t_{3PDerror} - t_{3PDerrorPrevious})}{\boldsymbol{pDelayInterval} \times \boldsymbol{mNRRsmoothingN}} \right)$$

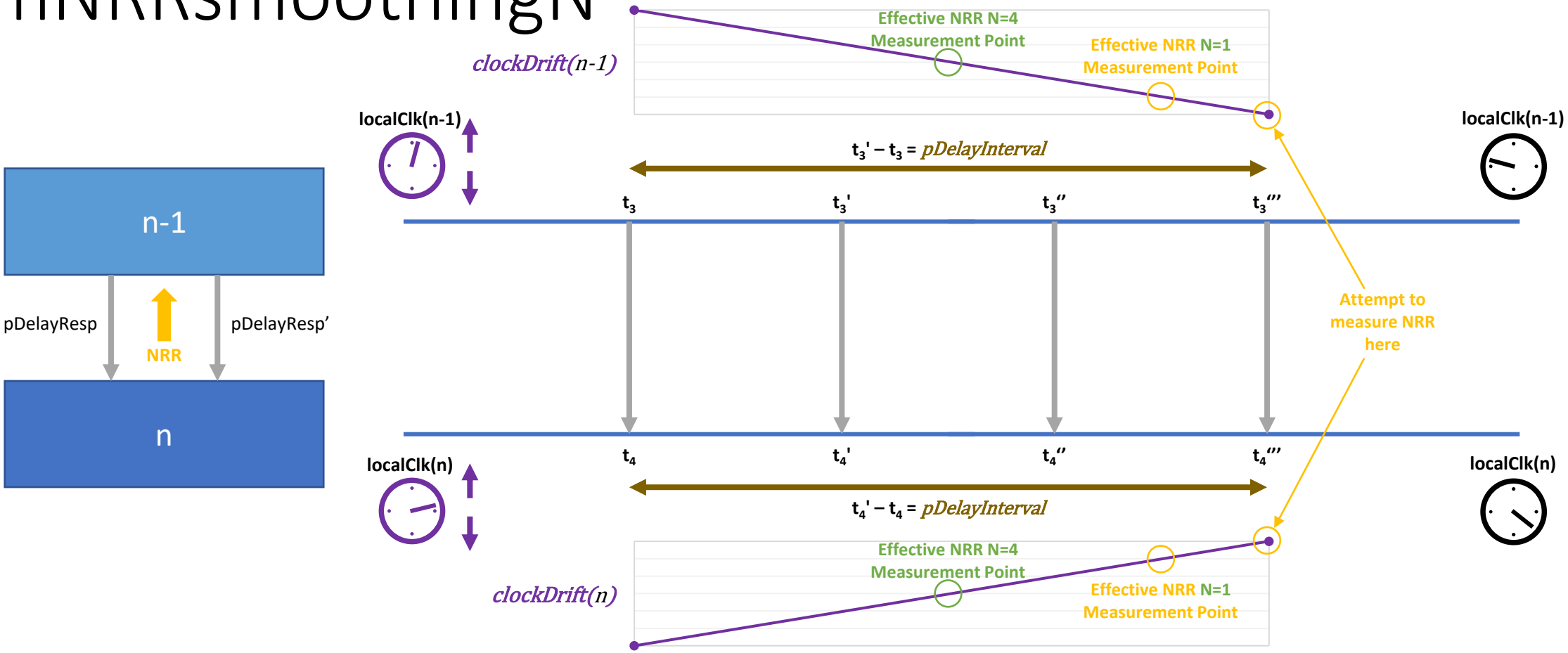
ppm

$mNRR_{errorCD}$



$$mNRR_{errorDrift}(n) = (1 - \boldsymbol{driftRate}_{errorCorrection}) \times \boldsymbol{mNRRsmoothingN} \times \left(\frac{\boldsymbol{pDelayInterval}}{\boldsymbol{2 \times 10^3}}\right)(\boldsymbol{clockDrift}(n-1) - \boldsymbol{clockDrift}(n))$$ **ppm**

mNRRsmoothingN



$$mNRR_{errorDrift}(n) = (1 - \boldsymbol{driftRate}_{errorCorrection}) \times \boldsymbol{mNRRsmoothingN} \times \left(\frac{\boldsymbol{pDelayInterval}}{\mathbf{2 \times 10^3}}\right)(\boldsymbol{clockDrift}(n-1) - \boldsymbol{clockDrift}(n))$$ **ppm**

Formulae – $mNRR_{error}$

$$mNRR_{error} = mNRR_{errorCD} + mNRRer_{rorTS}$$ **ppm**

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

$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror} - t_{4PDerrorPrevious}) - (t_{3PDerror} - t_{3PDerrorPrevious})}{\boldsymbol{pDelayInterval} \times \boldsymbol{mNRRsmoothingN}}\right)$$ **ppm**

$$t_{3PDerrorPrevious} = \boldsymbol{TSGE_{TX}} + \boldsymbol{DTSE_{TX}} \qquad t_{4PDerrorPrevious} = \boldsymbol{TSGE_{RX}} + \boldsymbol{DTSE_{RX}}$$ **ns**

The factors $t_{3PDerror}$ and $t_{4PDerror}$ are the same as in the $pDelay_{error}$ calculation.
(Same value. No need for new random numbers, as the same pDelayResp message is used to measure NRR.)

Observations of $mNRR_{error}$ Behaviour

- Errors in mNRR do not accumulate along the chain of nodes
- An mNRR error due to clock drift at one node will tend to be reversed at the next node.

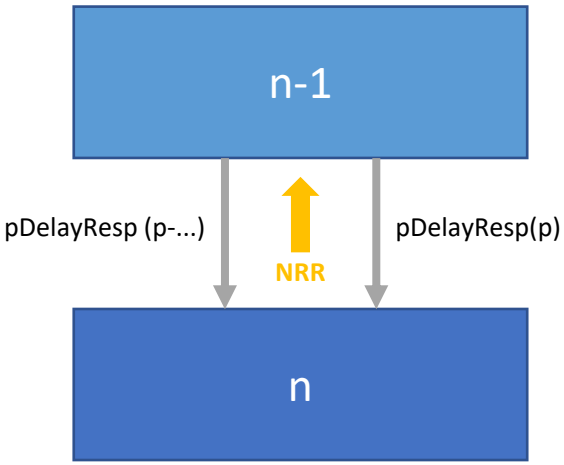
$$mNRR_{errorCD}(n) = \left(\frac{\boldsymbol{pDelayInterval}}{\mathbf{2 \times 10^3}}\right)\left(\boldsymbol{clockDrift}(n-1) - \boldsymbol{clockDrift}(n)\right) \qquad mNRR_{errorCD}(n+1) = \left(\frac{\boldsymbol{pDelayInterval}}{\mathbf{2 \times 10^3}}\right)\left(\boldsymbol{clockDrift}(n) - \boldsymbol{clockDrift}(n+1)\right)$$

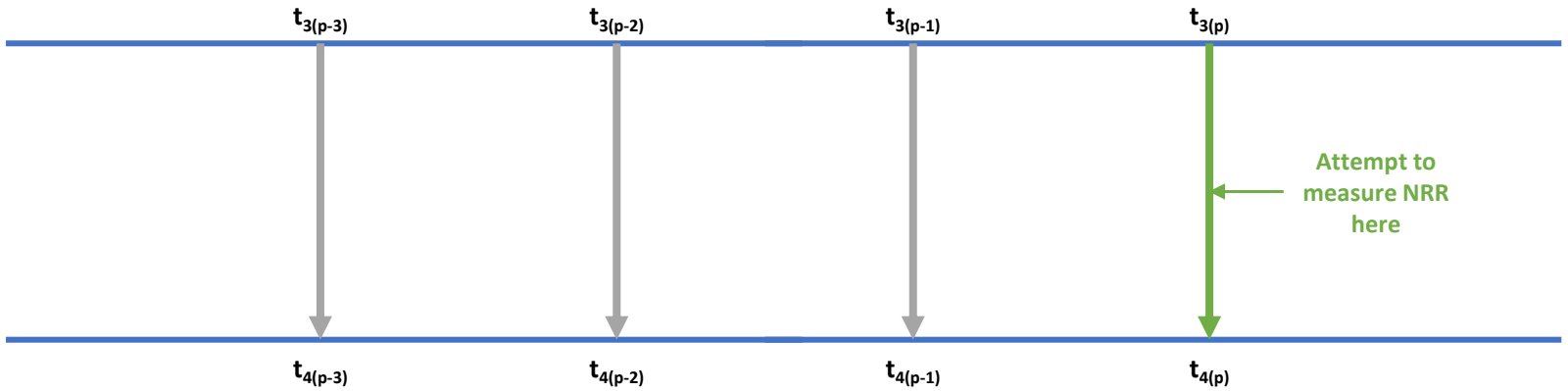
  - This does not apply for mNRR errors due to clock drift at the GM
- mNRR errors due to Timestamp errors are independent of each other. DTE errors at one node due to this component will not tend to be reversed at the next node.
- mNRR errors due to clock drift are modelled as a combination of two uniform distributions (clock drifts at n and n-1) and will have a simple probability distribution
  - Errors due to Timestamp errors are modelled as combinations of eight uniform distributions and will be more complex.

mNRRsmoothing

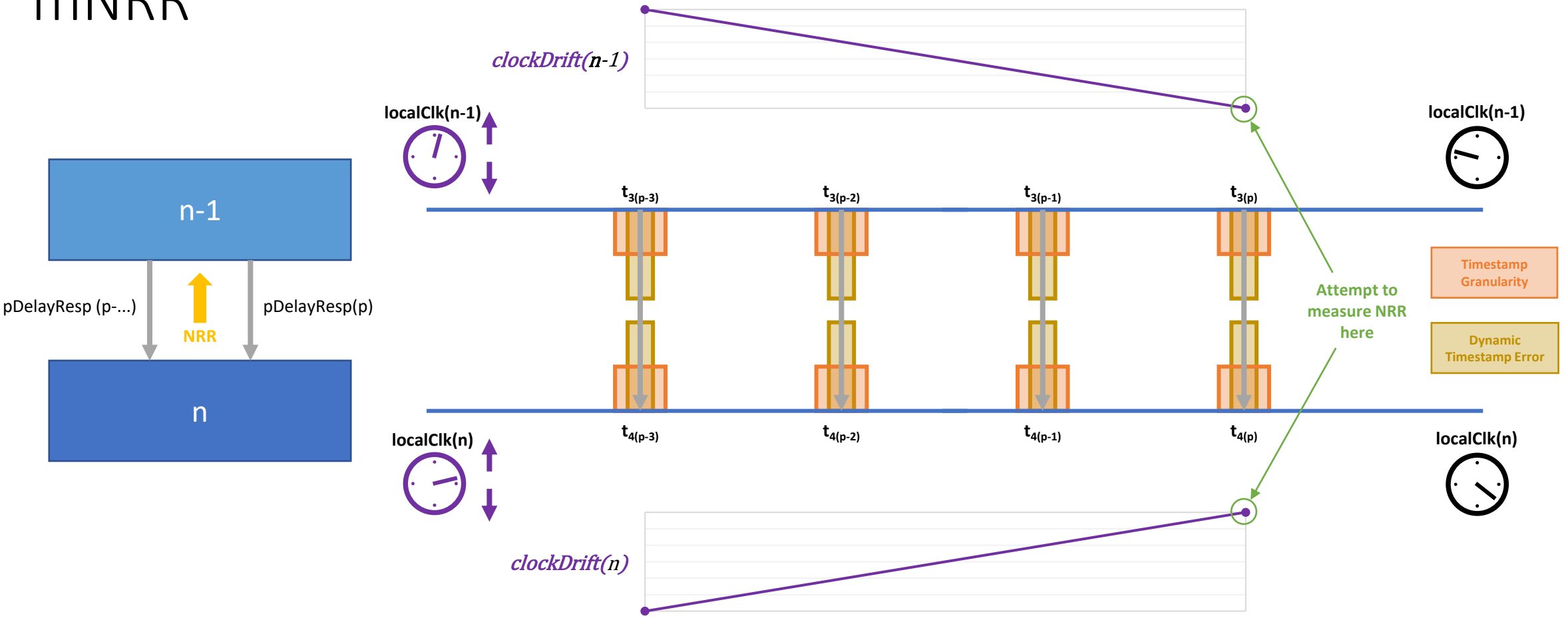
Taking a median of past NRR calculations

mNRR

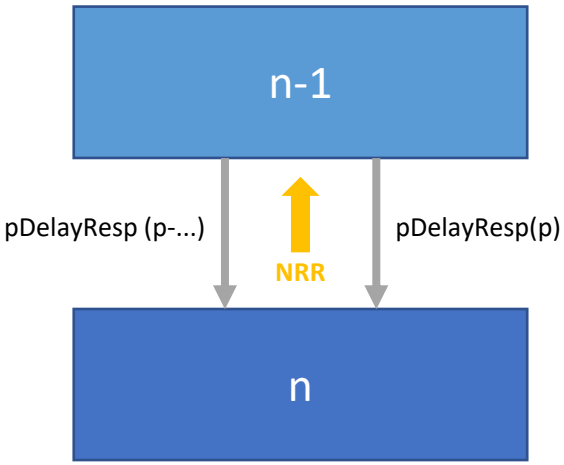

n-1
pDelayResp (p-...)
NRR
pDelayResp(p)
n

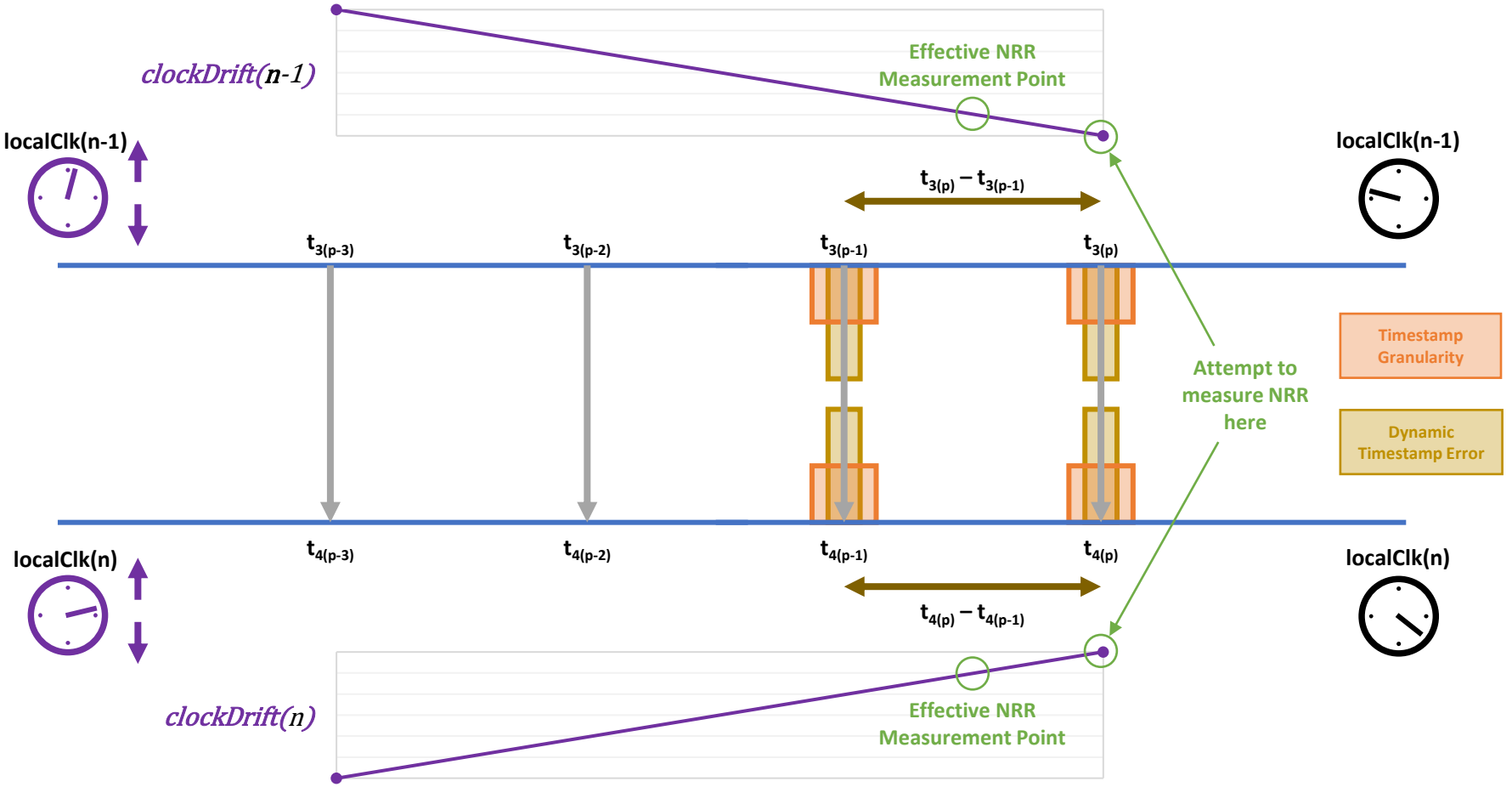

$t_{3(p-3)}$
$t_{3(p-2)}$
$t_{3(p-1)}$
$t_{3(p)}$
Attempt to measure NRR here
$t_{4(p-3)}$
$t_{4(p-2)}$
$t_{4(p-1)}$
$t_{4(p)}$

mNRR


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

$mNRR_{error}$



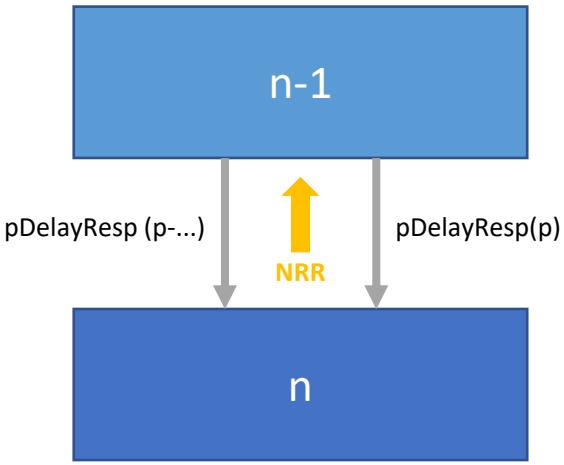


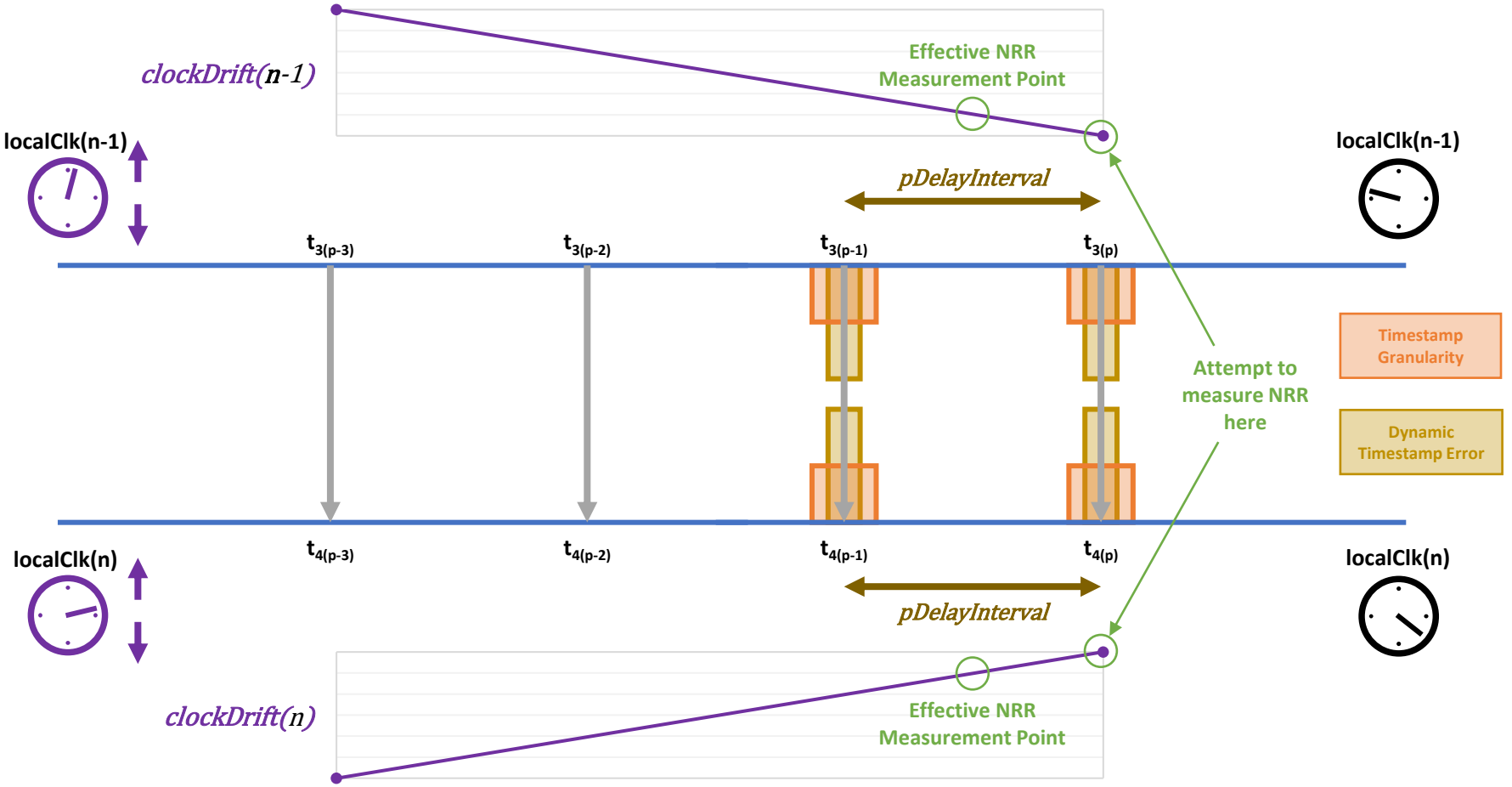
$$mNRR = \left(\frac{(t_{4_{(p)}} - t_{4_{(p-1)}})}{(t_{3_{(p)}} - t_{3_{(p-1)}})}\right)$$

$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror_{(p)}} - t_{4PDerror_{(p-1)}}) - (t_{3PDerror_{(p)}} - t_{3PDerror_{(p-1)}})}{(t_{4_{(p)}} - t_{4_{(p-1)}}) \times 10^6}\right) \quad \textbf{ppm}$$

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

$mNRR_{error}$





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

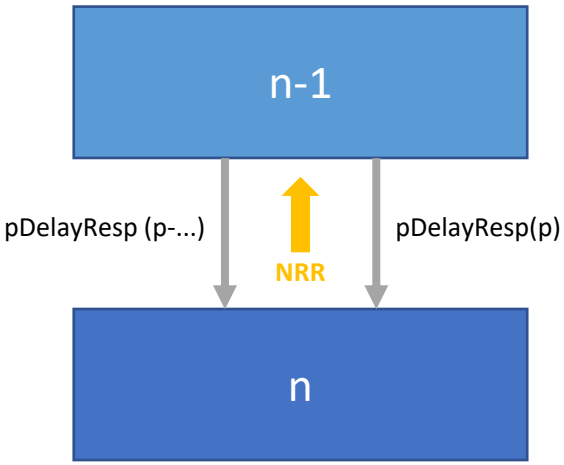
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror_{(p)}} - t_{4PDerror_{(p-1)}}) - (t_{3PDerror_{(p)}} - t_{3PDerror_{(p-1)}})}{pDelayInterval}\right)$$

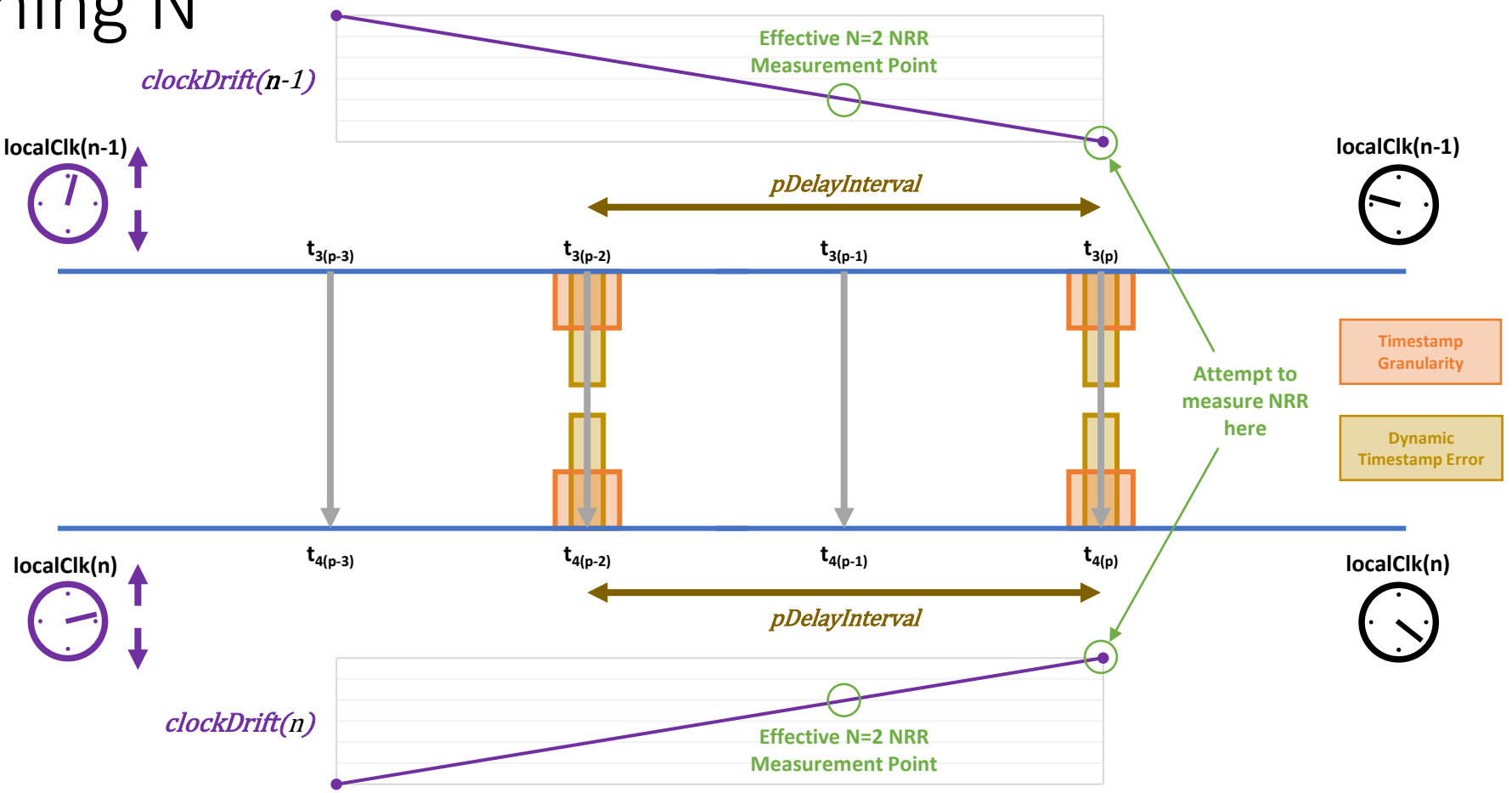
ppm

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

ppm

mNRR Smoothing N



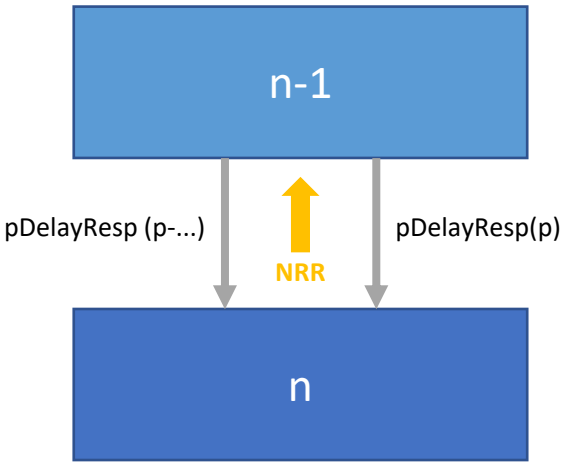


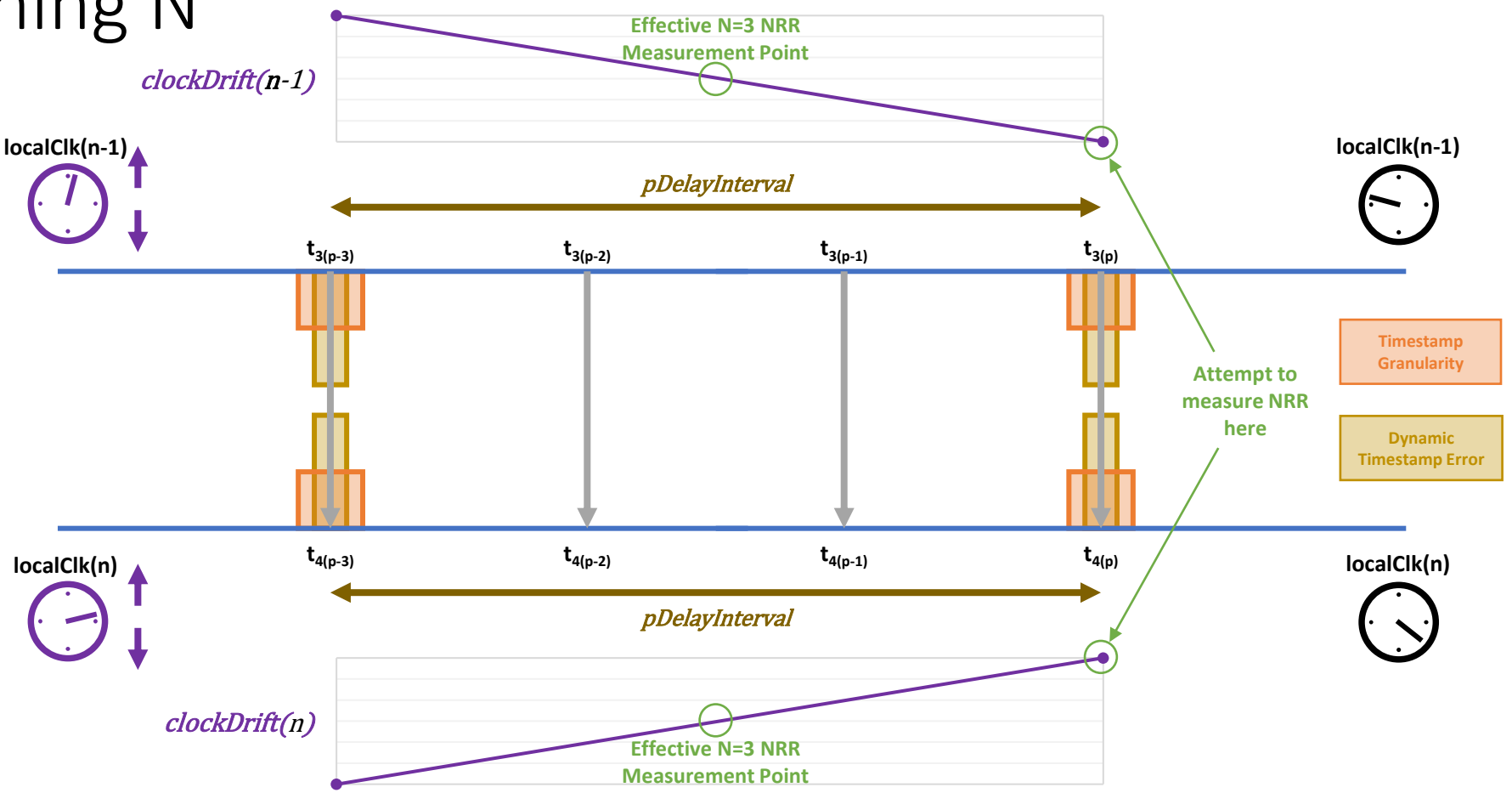
$$mNRR = \left(\frac{(t_{4_{(p)}} - t_{4_{(p-1)}})}{(t_{3_{(p)}} - t_{3_{(p-1)}})}\right)$$

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

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

mNRR Smoothing N





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

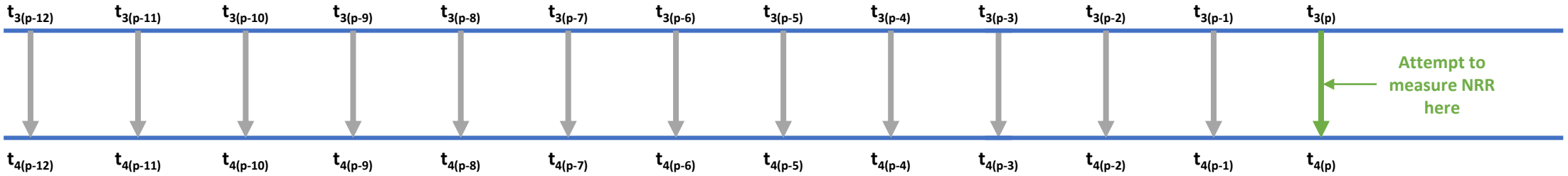
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror_{(p)}} - t_{4PDerror_{(p-1)}}) - (t_{3PDerror_{(p)}} - t_{3PDerror_{(p-1)}})}{pDelayInterval \times 3}\right)$$

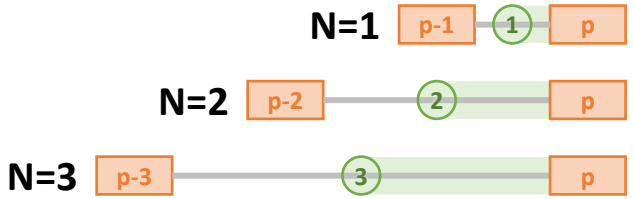
ppm

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

ppm

mNRR Smoothing N

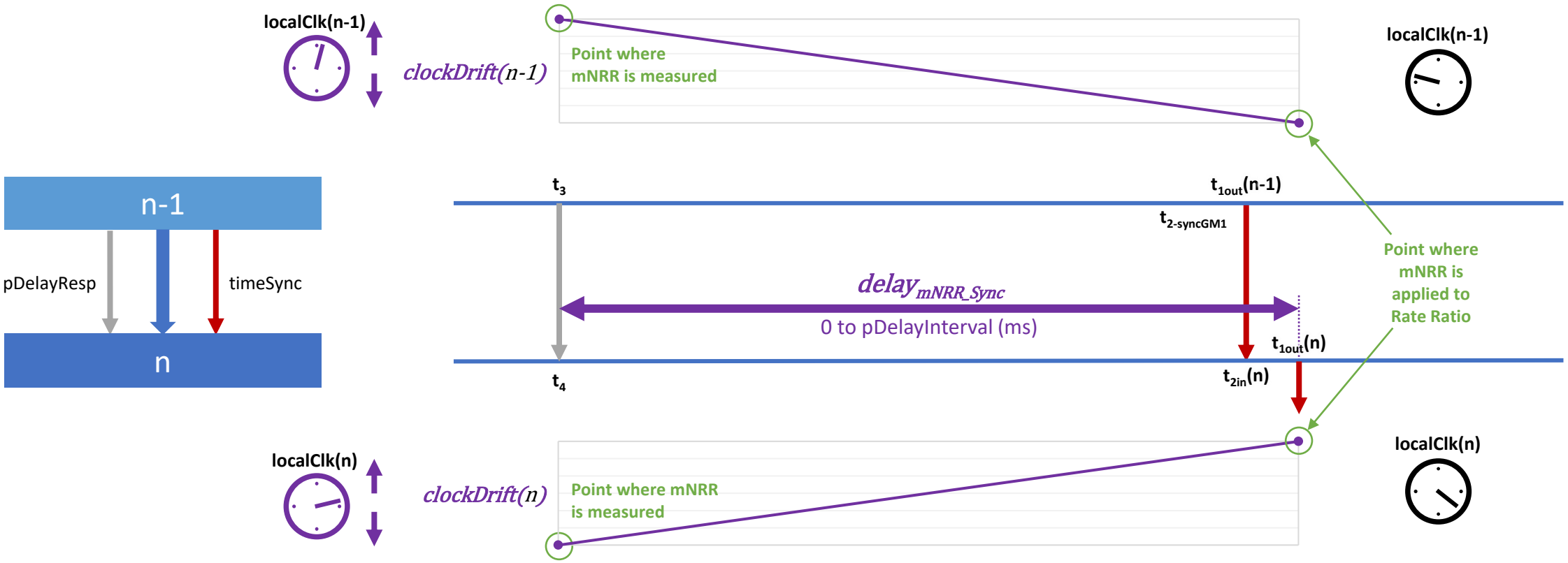




mNRR Smoothing N

- 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

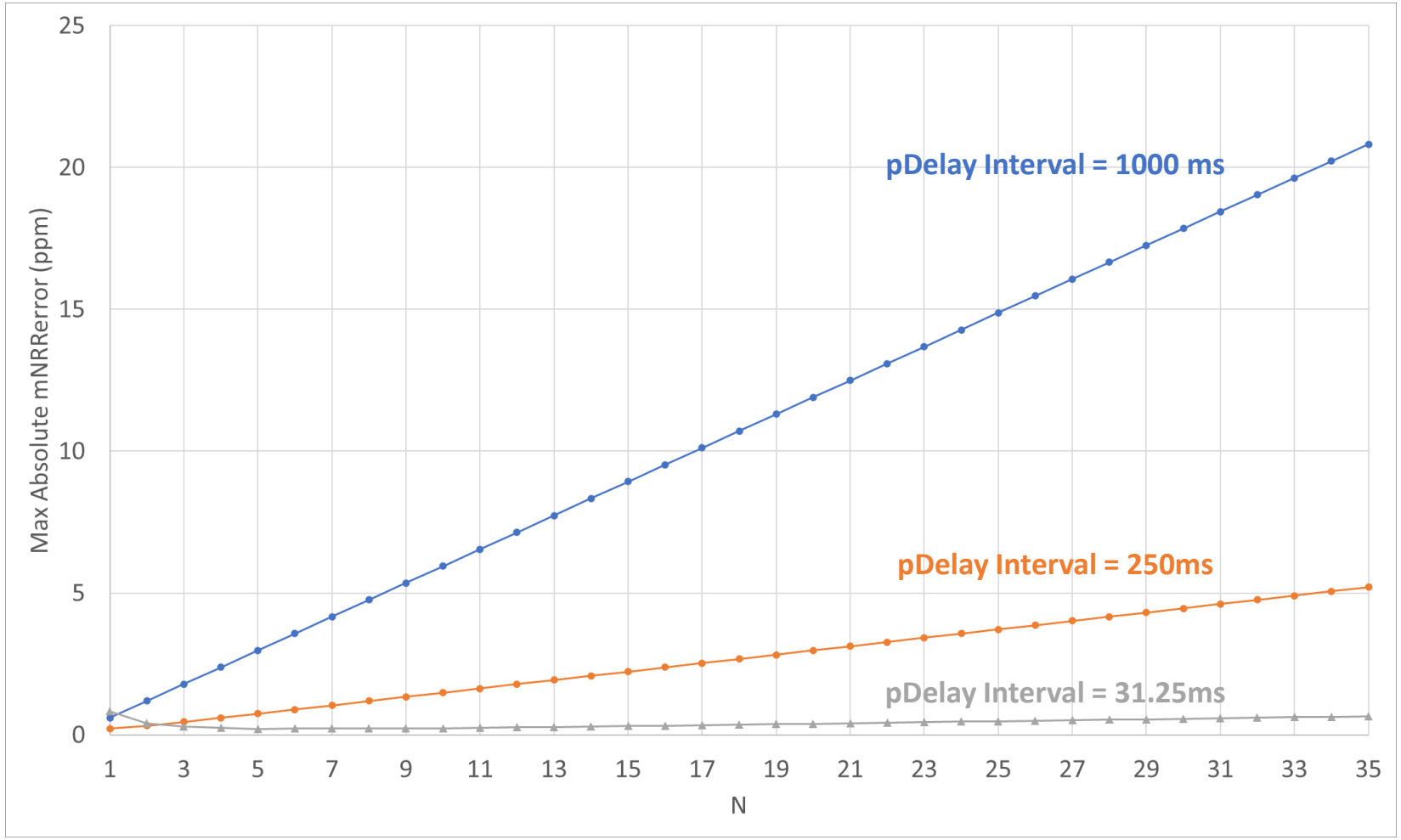
$RR_{errorCD}$



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

ppm

Optimal Value of N

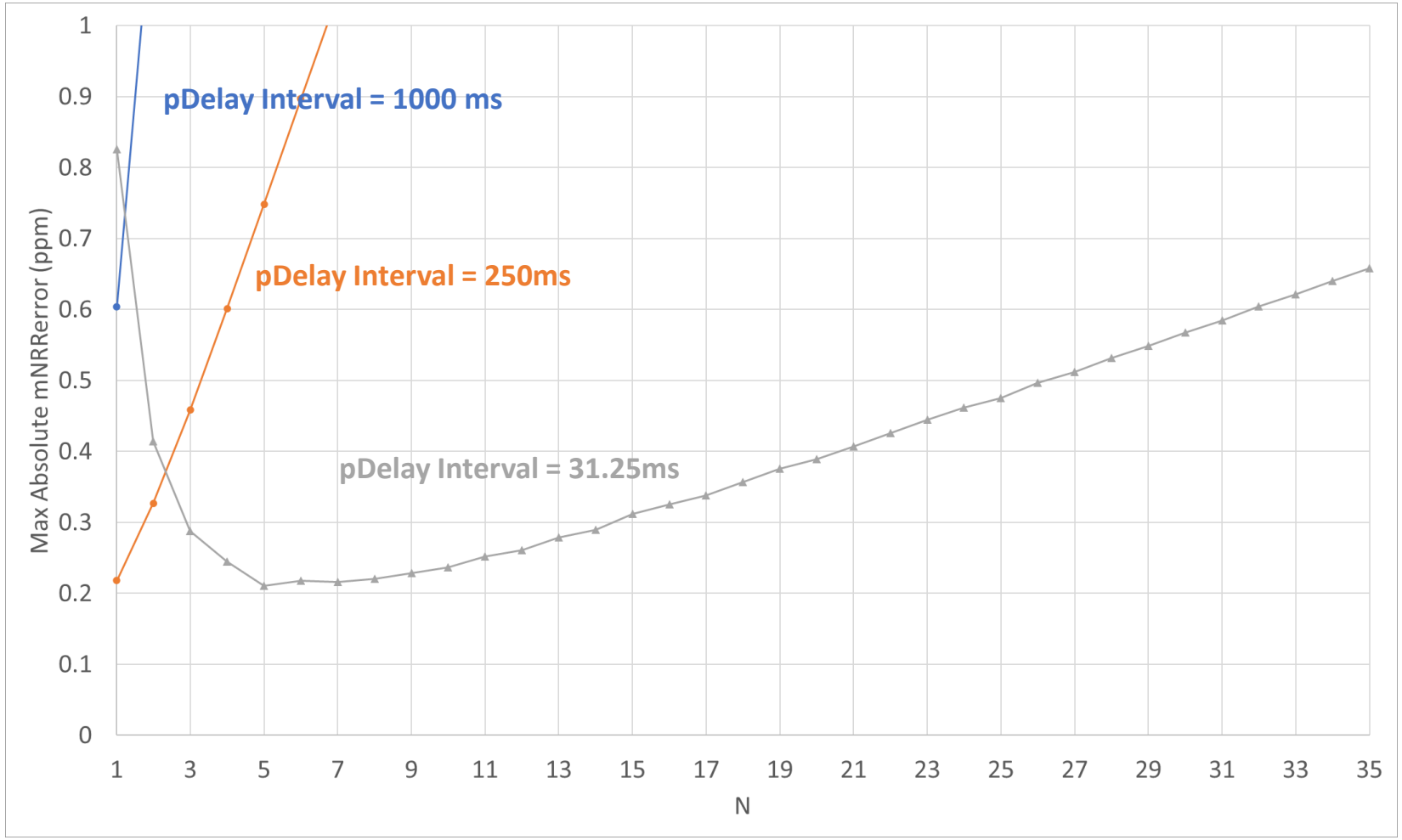


Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Input Correction Factors		
mNRR Smoothing M	1	
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

How short?...

Optimal Value of N



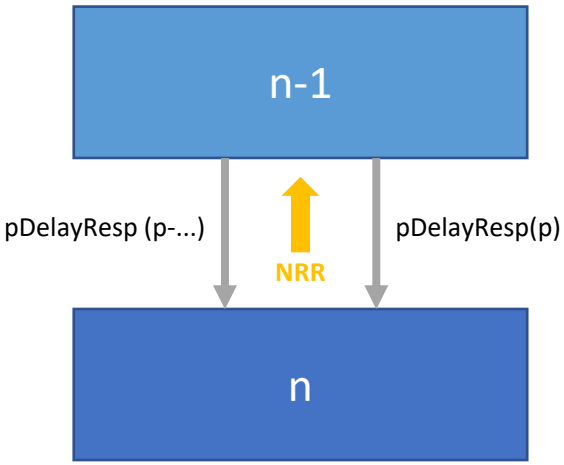
Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Input Correction Factors		
mNRR Smoothing M	1	
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

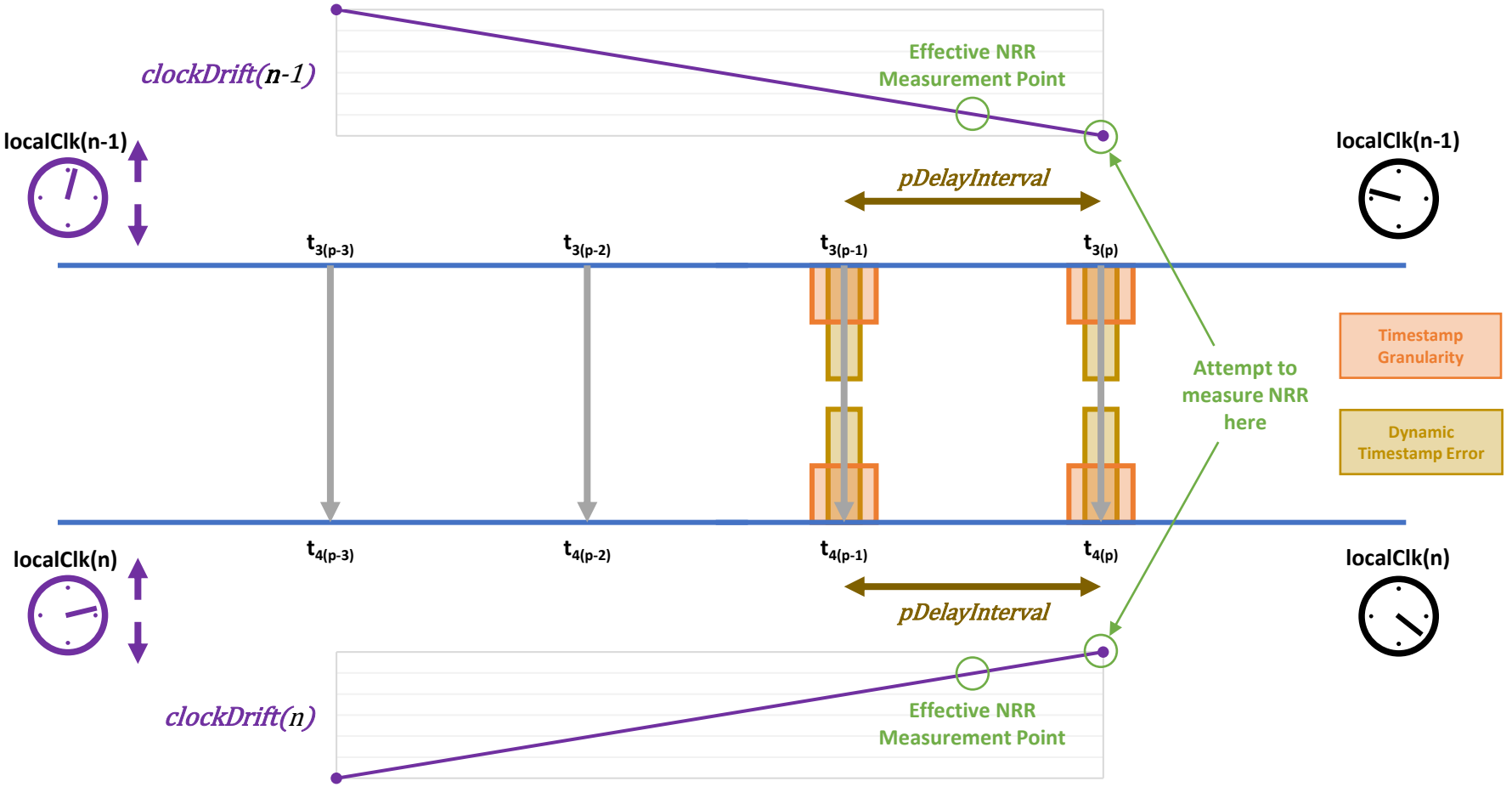
Optimal value of N for pDelay Interval **31.25ms** is **5**. (For these input errors, with no clock drift compensation.)

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**.

$mNRR_{error}$





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

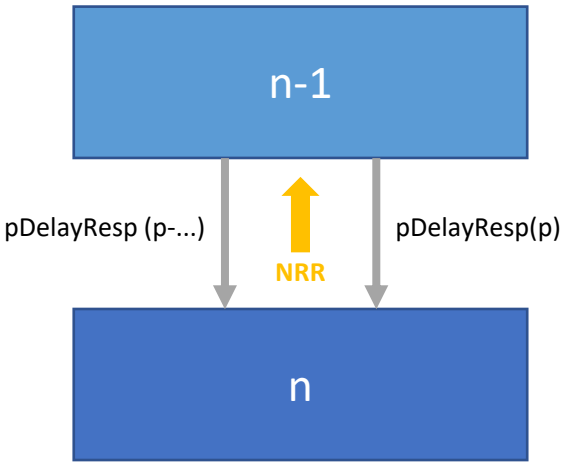
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror_{(p)}} - t_{4PDerror_{(p-1)}}) - (t_{3PDerror_{(p)}} - t_{3PDerror_{(p-1)}})}{pDelayInterval}\right)$$

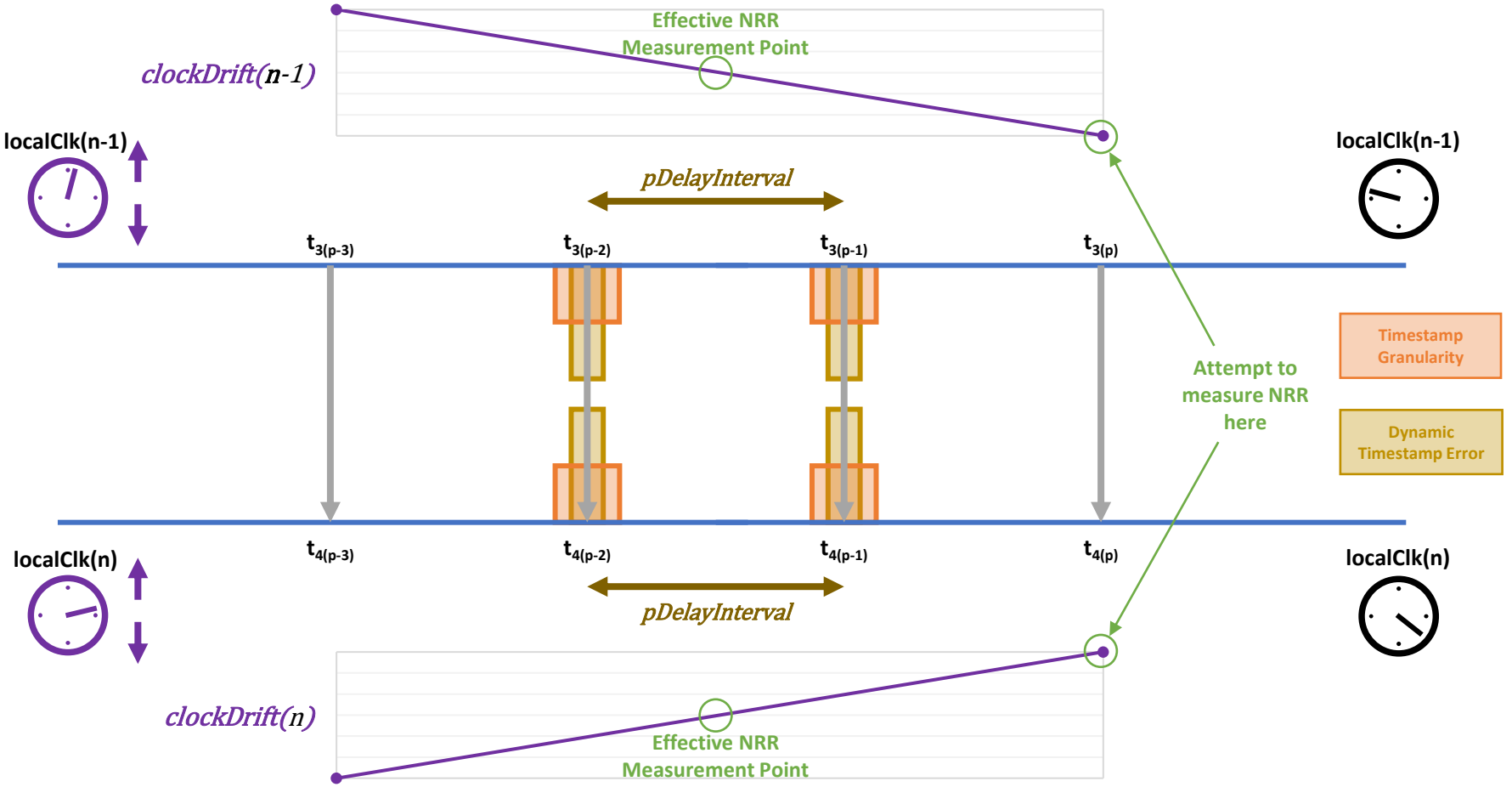
ppm

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

ppm

mNRR Smoothing M Calculation B





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

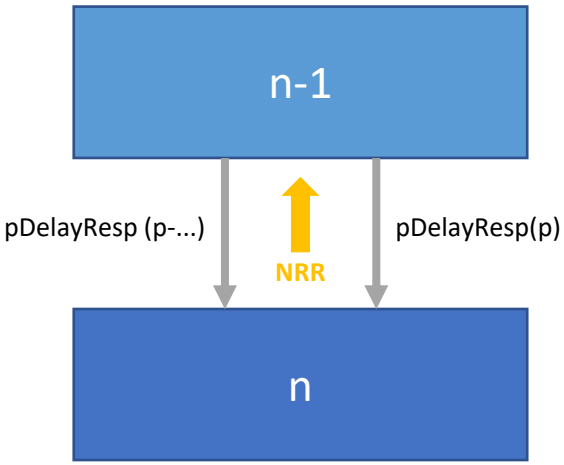
$$mNRR_{errorTS} = \left(\frac{(t_{4PDerror_{(p-1)}} - t_{4PDerror_{(p-2)}}) - (t_{3PDerror_{(p-1)}} - t_{3PDerror_{(p-2)}})}{pDelayInterval}\right)$$

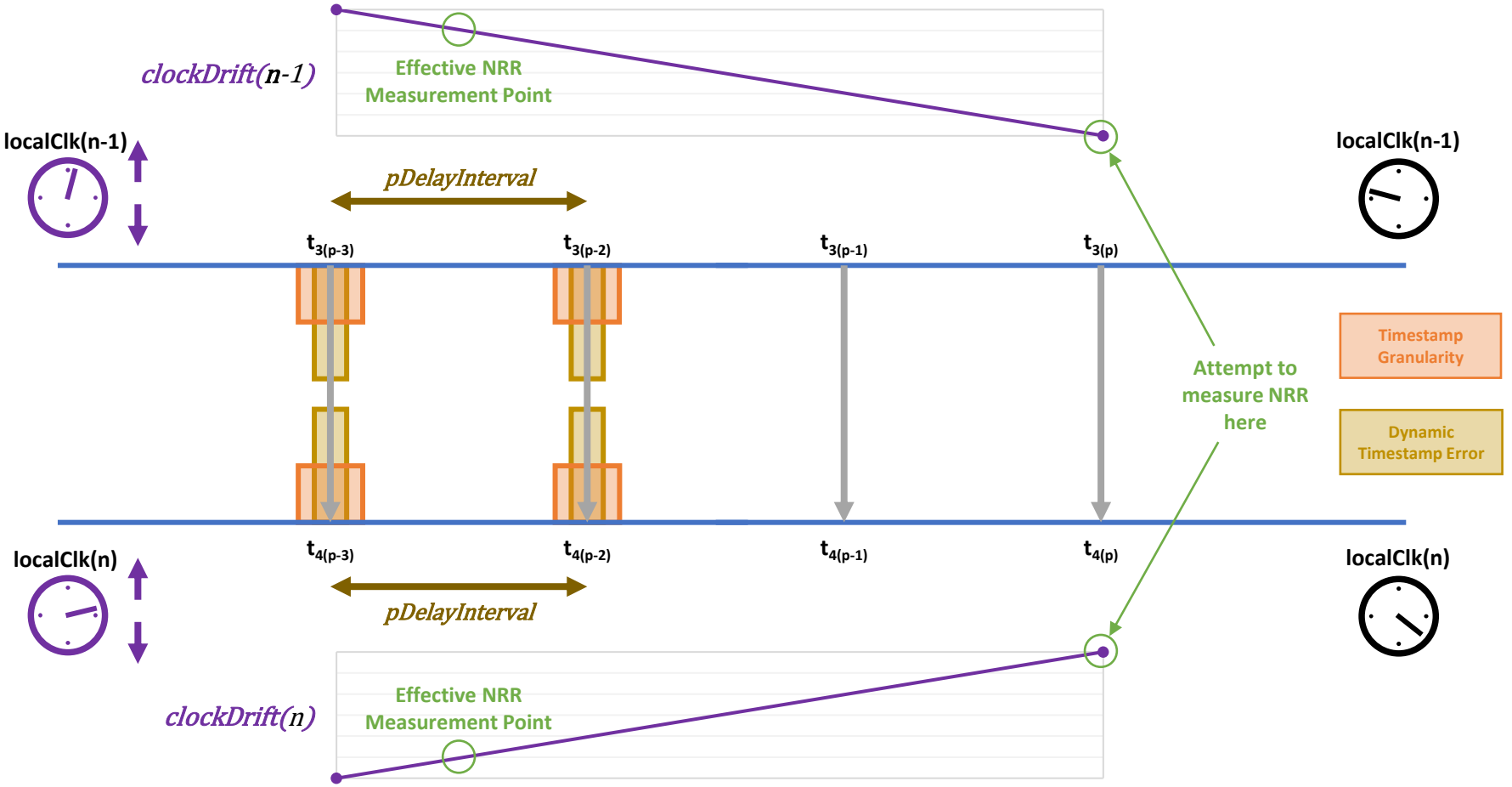
ppm

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

ppm

mNRR Smoothing M Calculation C





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

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

ppm

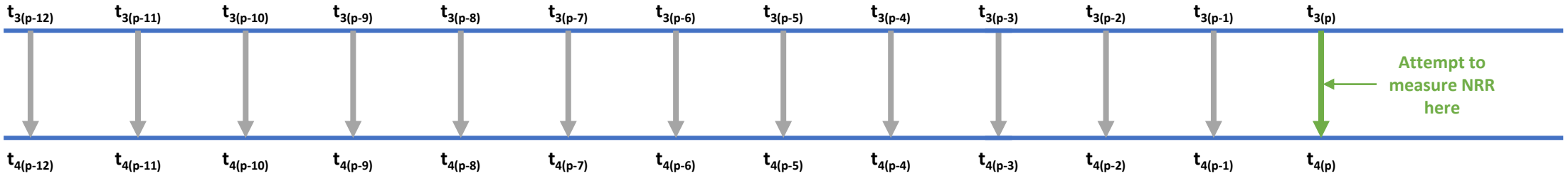
$$mNRR_{errorCD}(n) = \left(\frac{pDelayInterval \times 5}{2 \times 10^3}\right)(clockDrift(n-1) - clockDrift(n))$$

ppm

mNRR Smoothing M

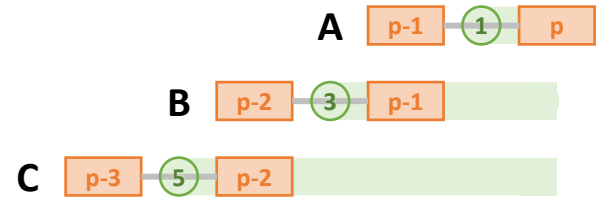
- Using the median of M previous NRR calculations...
  - Selects the NRR calucation with the median total error ($\boldsymbol{mNRR_{error}}$)
    - Addition of $\boldsymbol{mNRR_{errorTS}}$ from Timestamp Errors & $\boldsymbol{mNRR_{errorCD}}$ from Clock Drift
- Implementing this in the Monte Carlo Analysis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

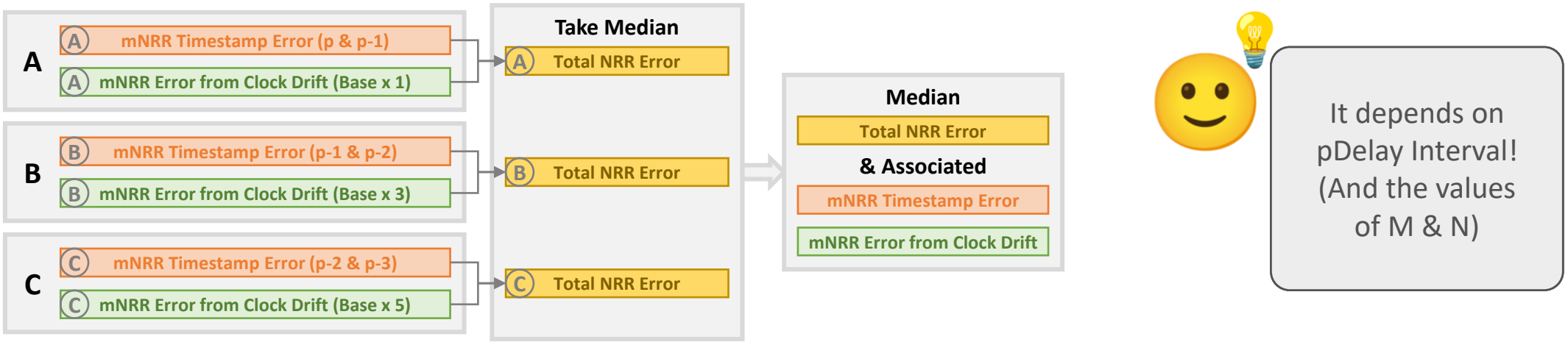
mNRR Smoothing M – Calculations



N = 1

M = 3

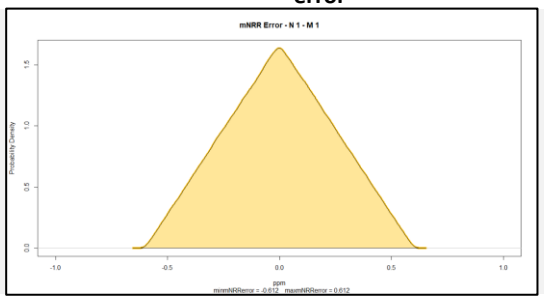
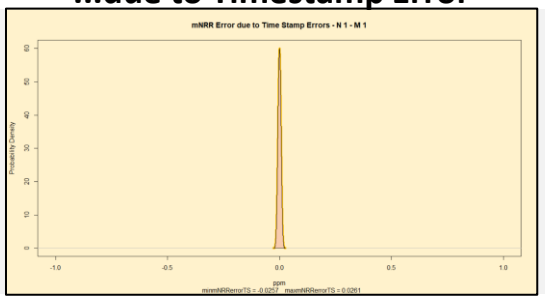
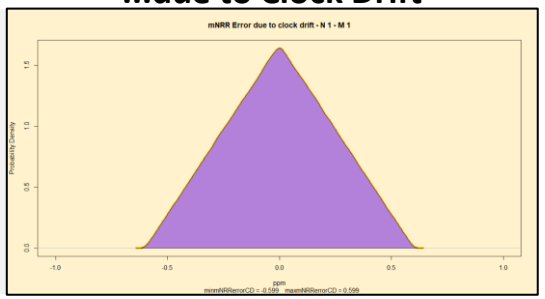
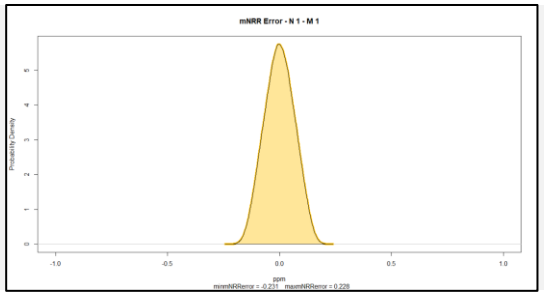
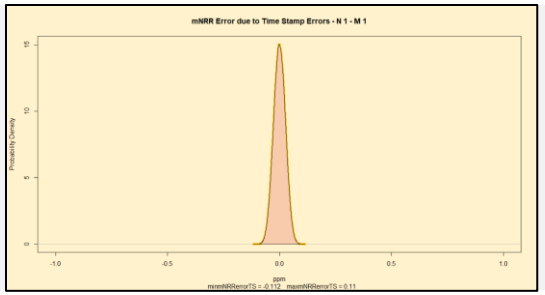
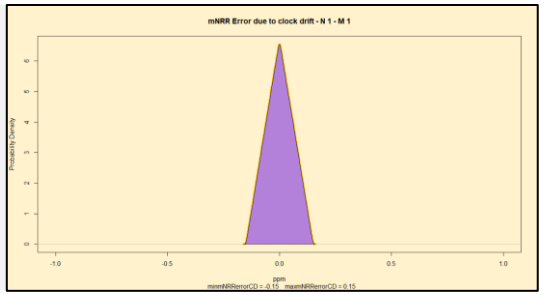
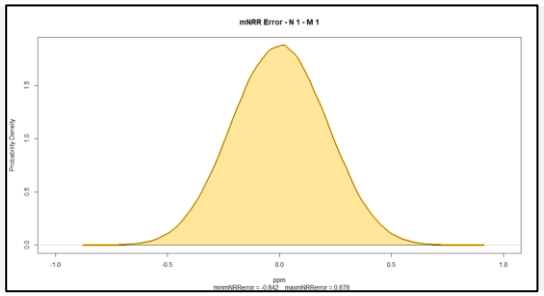
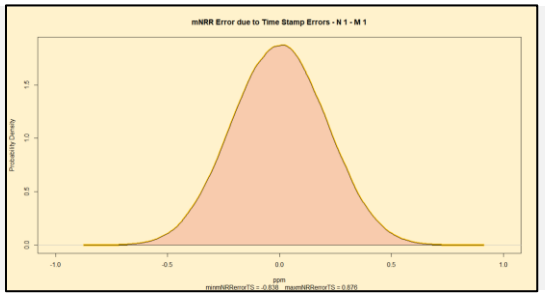
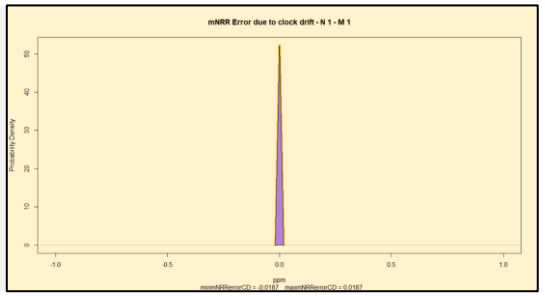




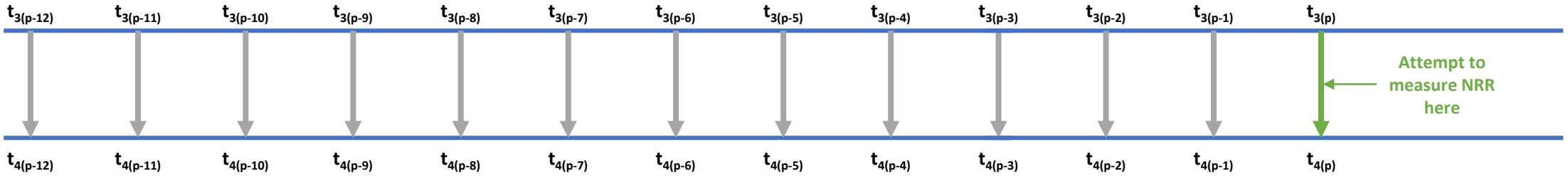
It depends on pDelay Interval! (And the values of M & N)

Timestamp Error vs Error due to Clock Drift

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

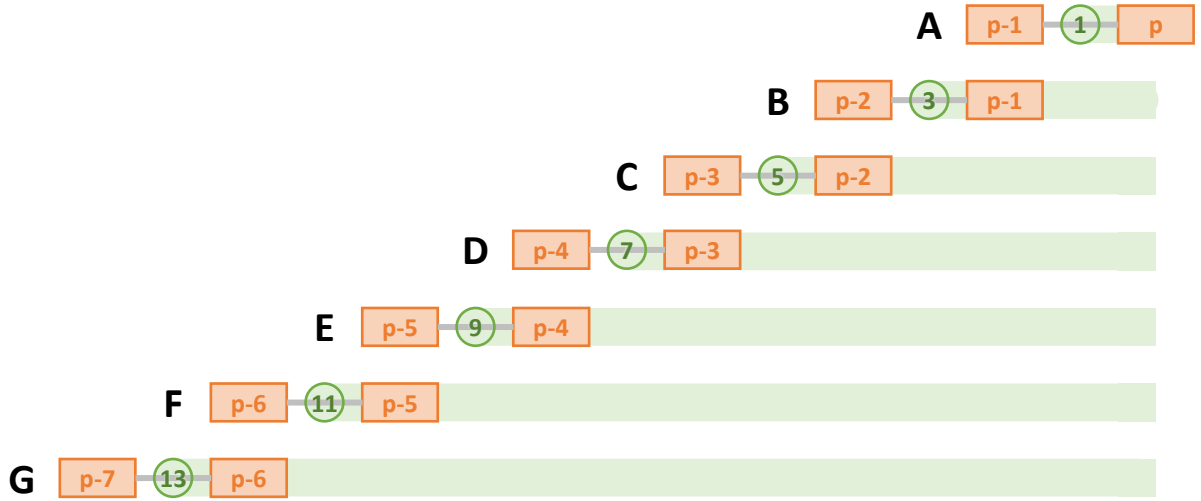
pDelay Interval	$mNRR_{error}$	...due to Timestamp Error	...due to Clock Drift
1000ms			
250ms			
31.25ms			

mNRR Smoothing M & pDelay Interval



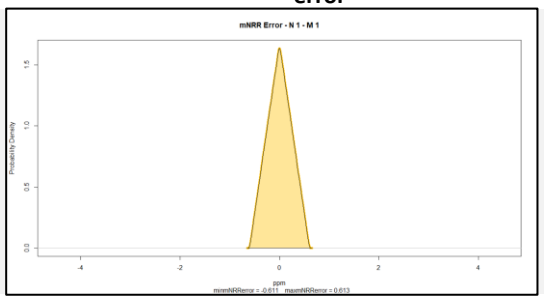
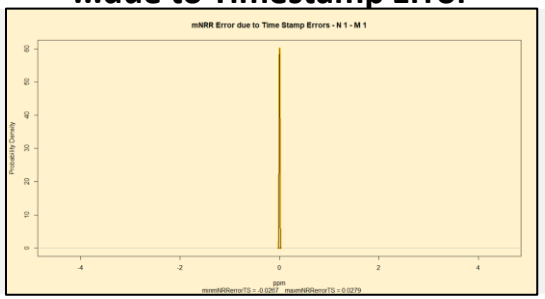
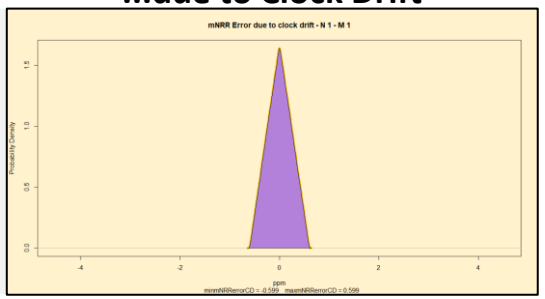
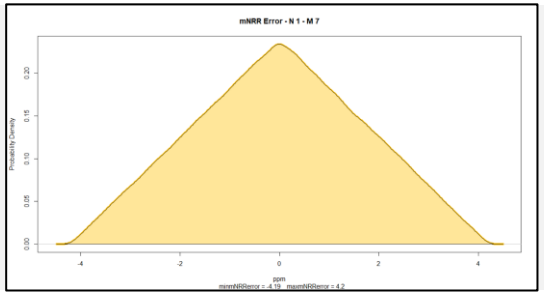
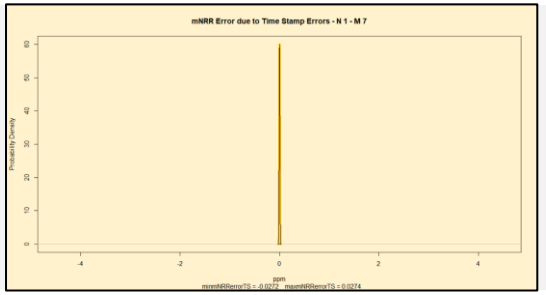
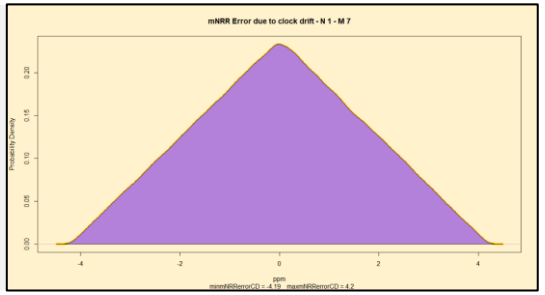
N = 1

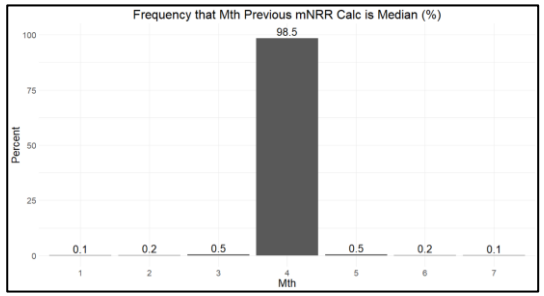
M = 7



Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

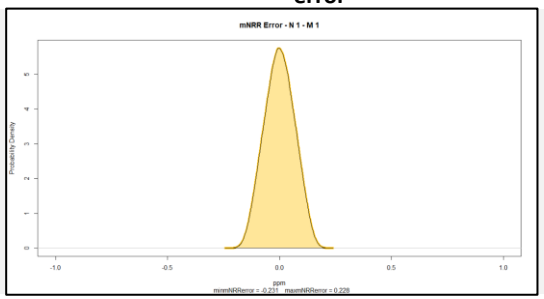
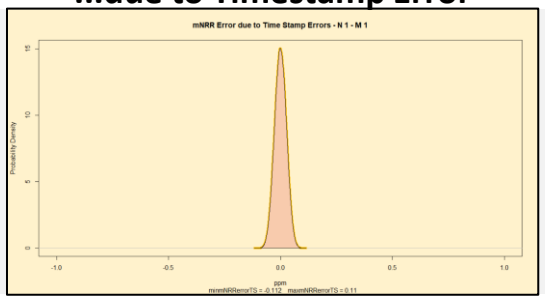
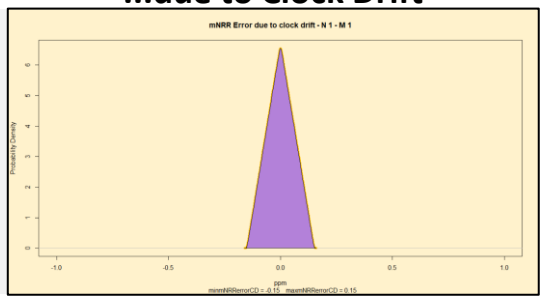
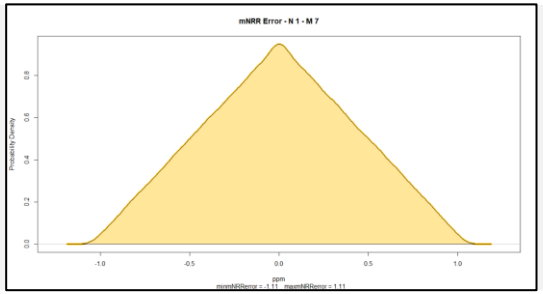
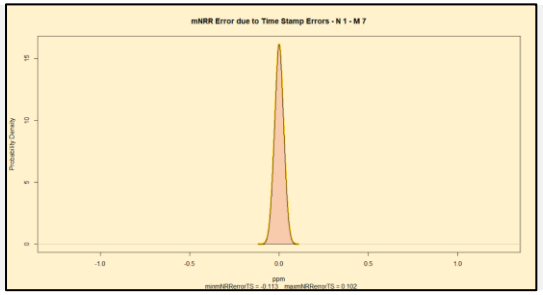
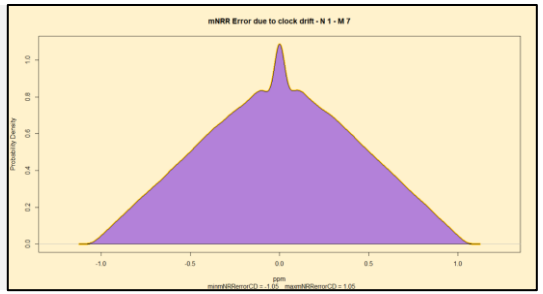
	pDelay Interval	$mNRR_{error}$	...due to Timestamp Error	...due to Clock Drift
M=1	**1000ms**			
M=7	**1000ms**			

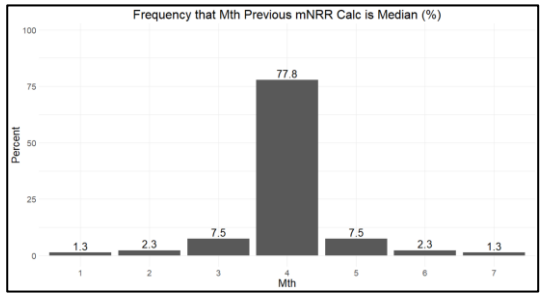


- For values of pDelay Interval where $mNRR_{error}$ due to Clock Drift dominates Timestamp error, taking the median of previous NRR calculations only increases $mNRR_{error}$.

Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

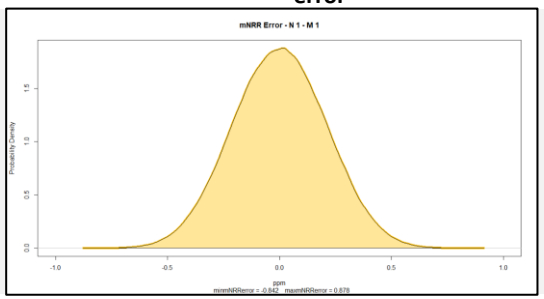
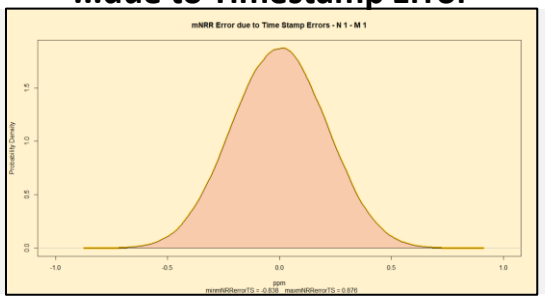
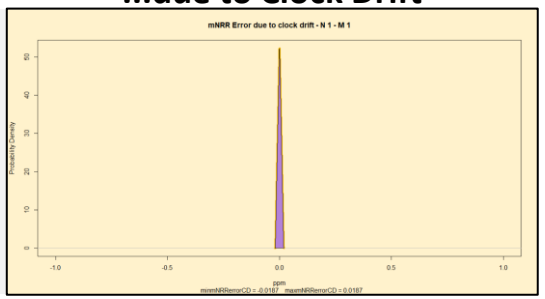
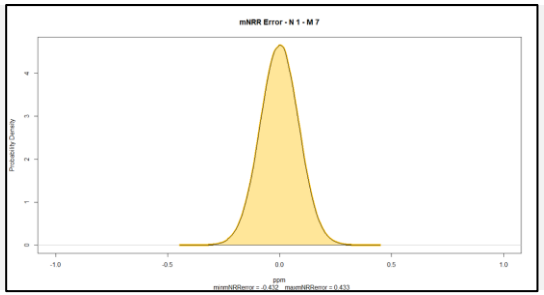
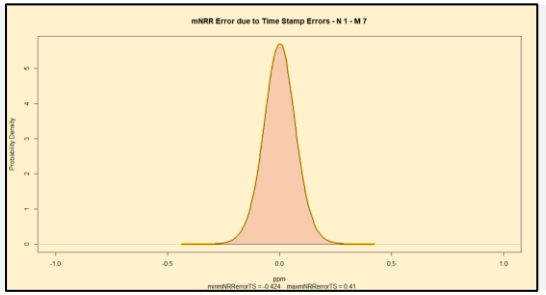
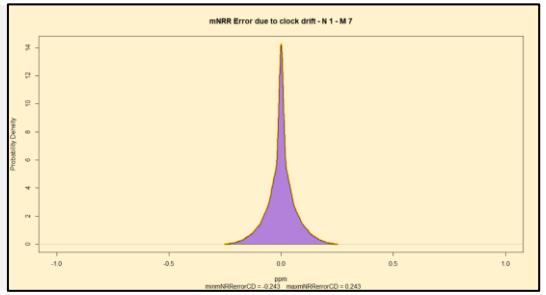
	pDelay Interval	$mNRR_{error}$	...due to Timestamp Error	...due to Clock Drift
M=1	**250ms**			
M=7	**250ms**			

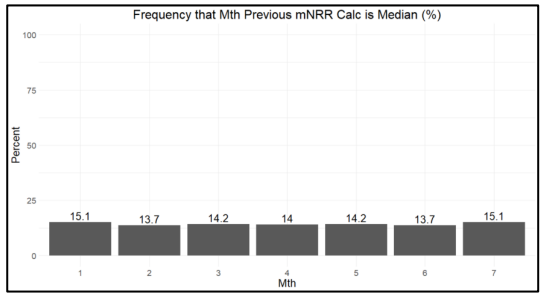


- For values of pDelay Interval where $mNRR_{error}$ due to Clock Drift and Timestamp error are of comparable magnitude, taking the median of previous NRR calculations only increases $mNRR_{error}$.

Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

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

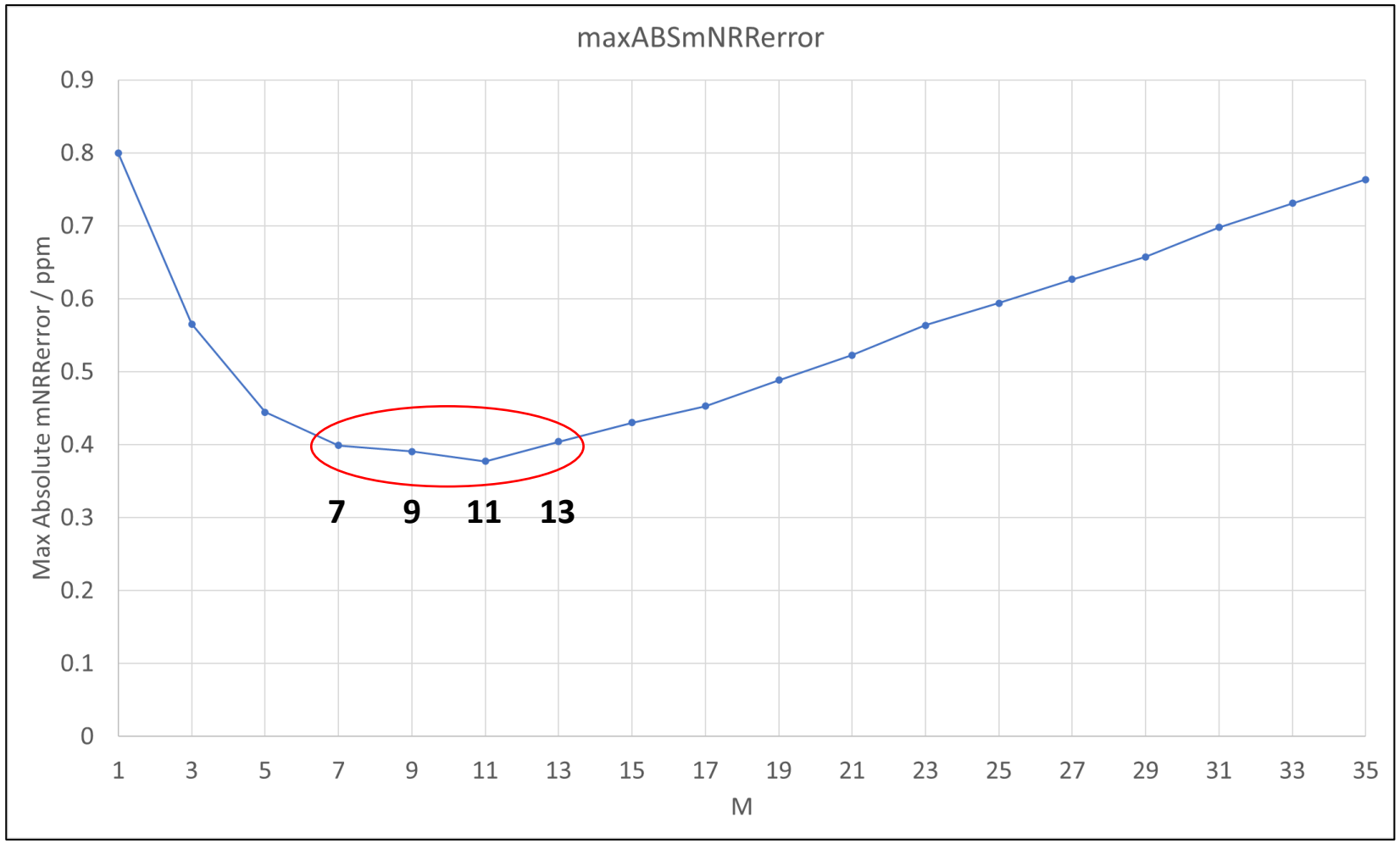


- For values of pDelay Interval where $mNRR_{error}$ due to Timestamp error dominates error due to Clock Drift, taking the median of previous NRR calculations can reduce $mNRR_{error}$.
- This raises the question: what is the optimal value of M?

Challenges of Comparing Results

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

Optimal Value of M

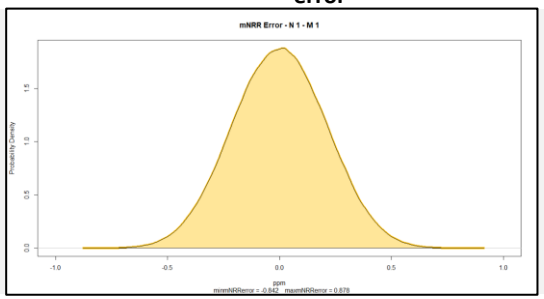
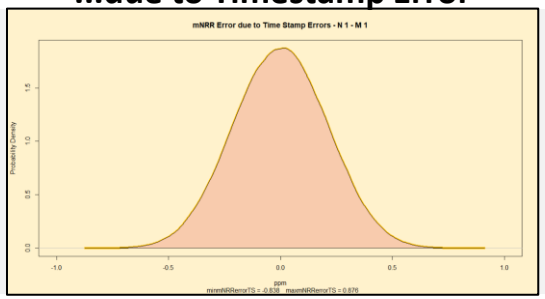
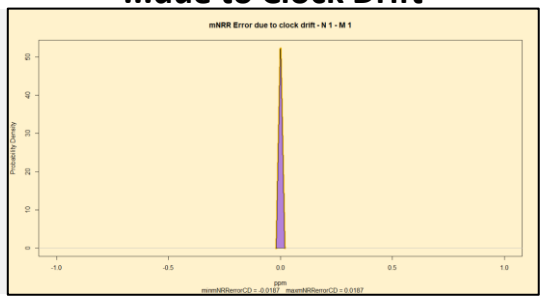
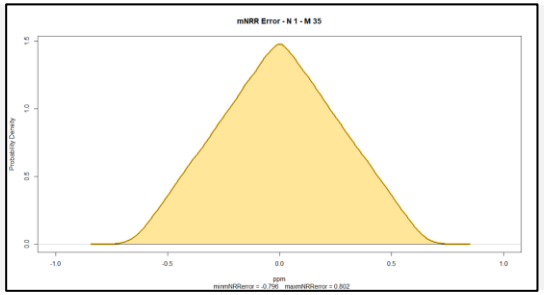
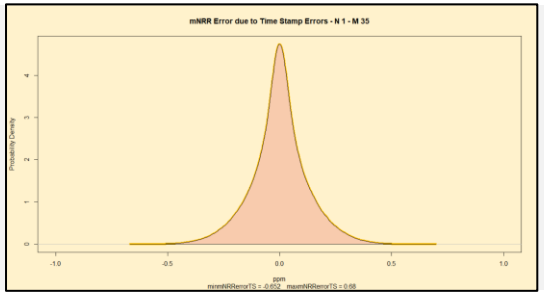
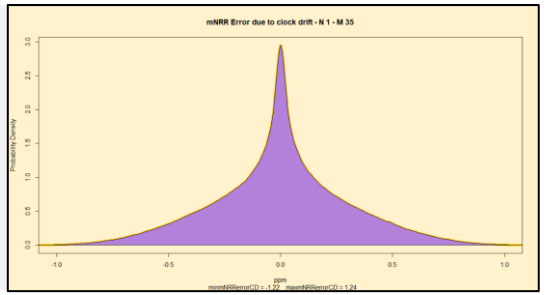


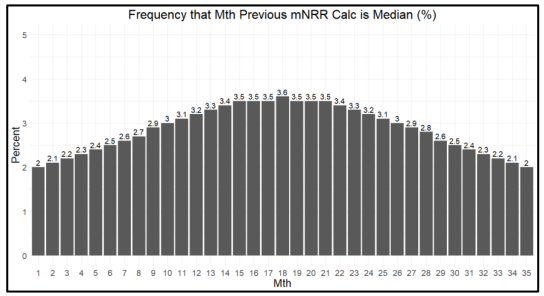
Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Input Correction Factors		
mNRR Smoothing N	1	
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

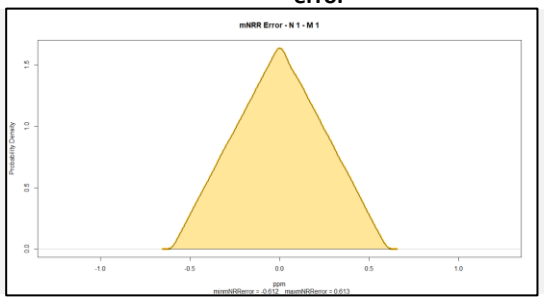
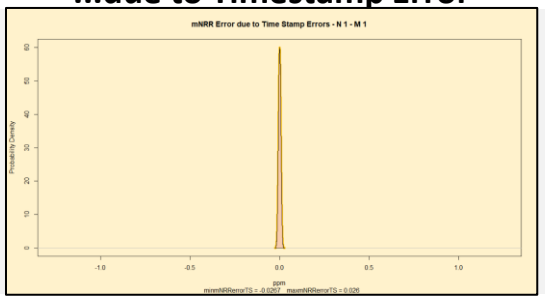
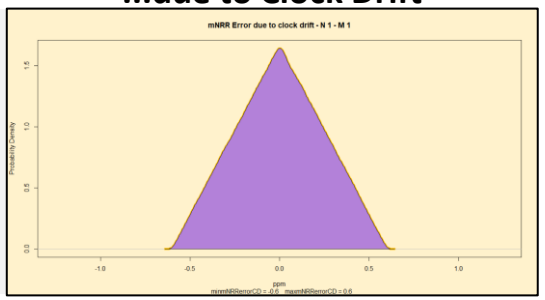
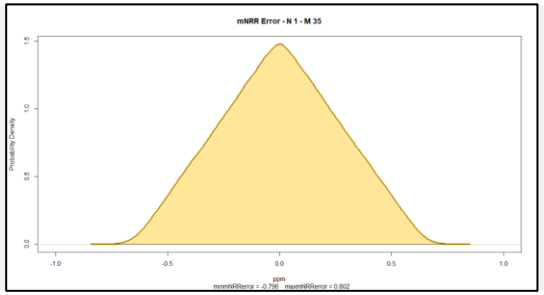
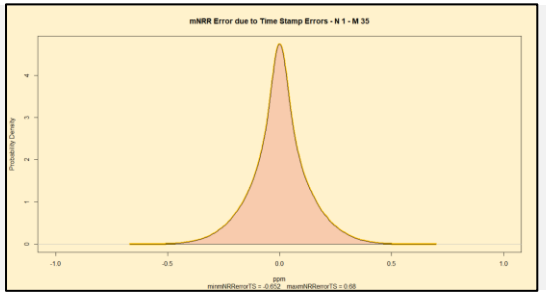
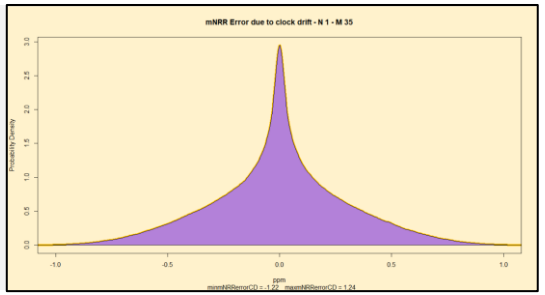
	pDelay Interval	$mNRR_{error}$	...due to Timestamp Error	...due to Clock Drift
M=1	**31.25ms**			
M=35	**31.25ms**			

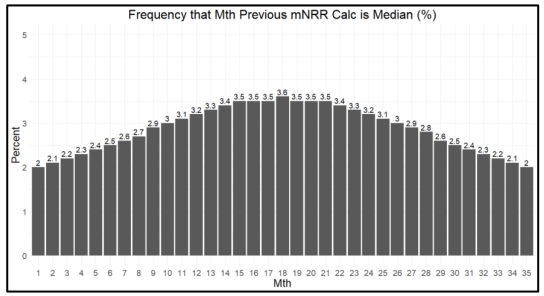


- Increasing M to 35 results in using NRR calculations from up to 1093.75ms in the past...and an $mNRR_{errror}$ distribution that is similar magnitude to pDelayInterval of 1000ms (and M=1).
- However...

Effect of Taking Median of mNRR Calculations

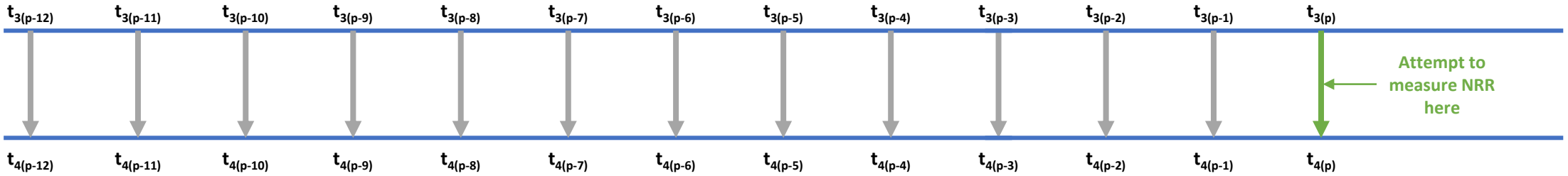
(mNRR – Single Hop – N=1)

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

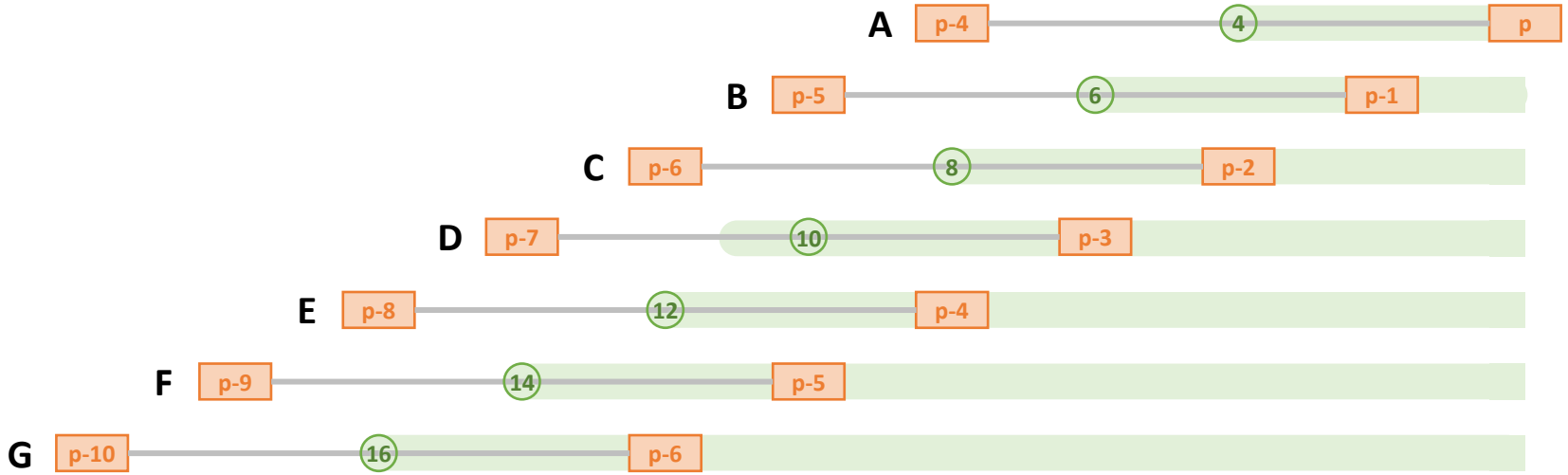


- The distribution of the underlying Timestamp errors and Clock Drift errors are very different.

mNRR Smoothing – Combining N & M

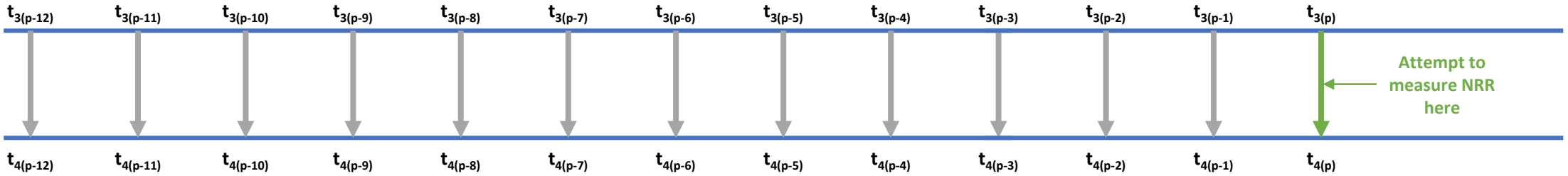


N = 4
M = 7



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

mNRR Smoothing – Combining N & M



N = 4
M = 7

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

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

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

M>N?

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

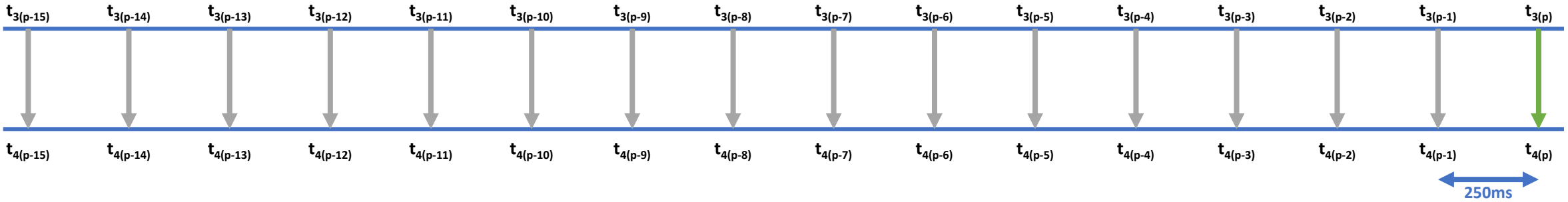
Series 1

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

Series 2

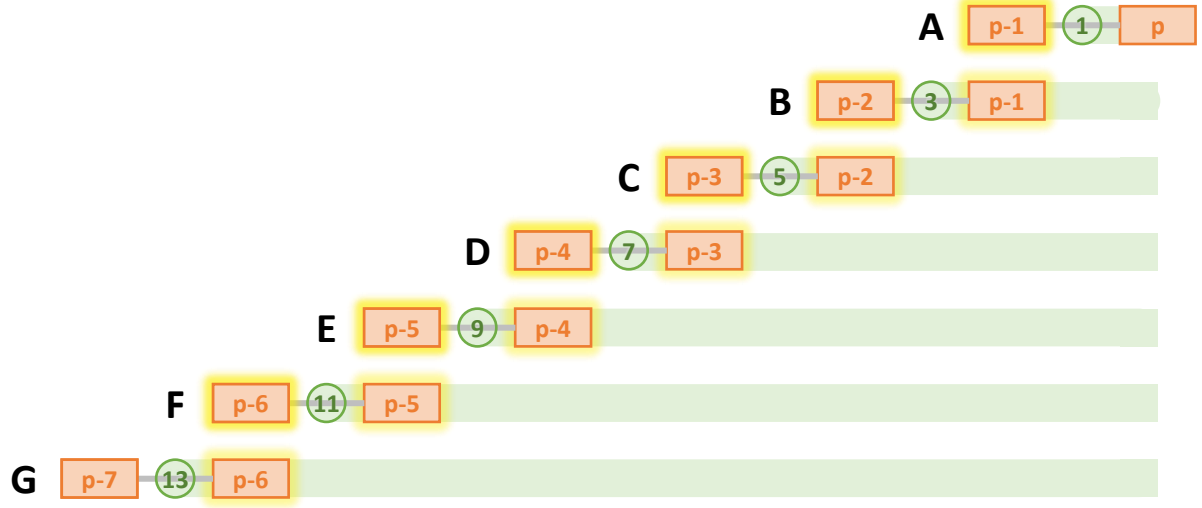
N	pDelay Interval	pDelay N x Interval
1	31.25 ms	31.25 ms
4	31.25 ms	62.5 ms
7	31.25 ms	218.75ms

mNRR Smoothing – Combining N & M

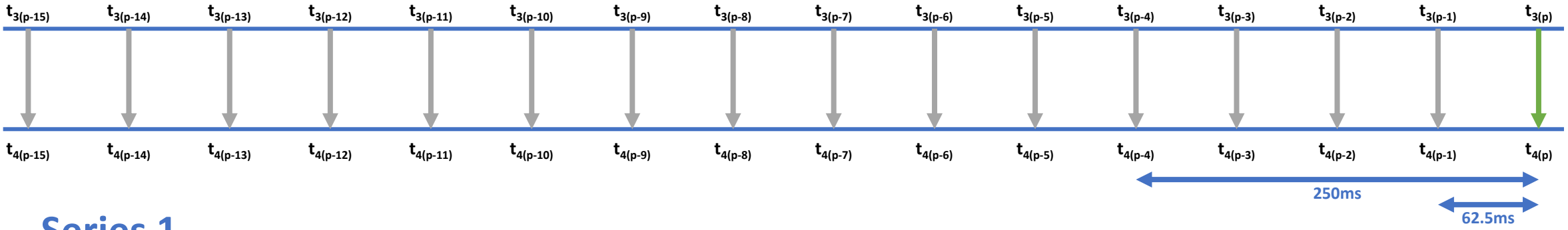


Series 1

pDelay Interval = 250ms
N = 1
M = 7

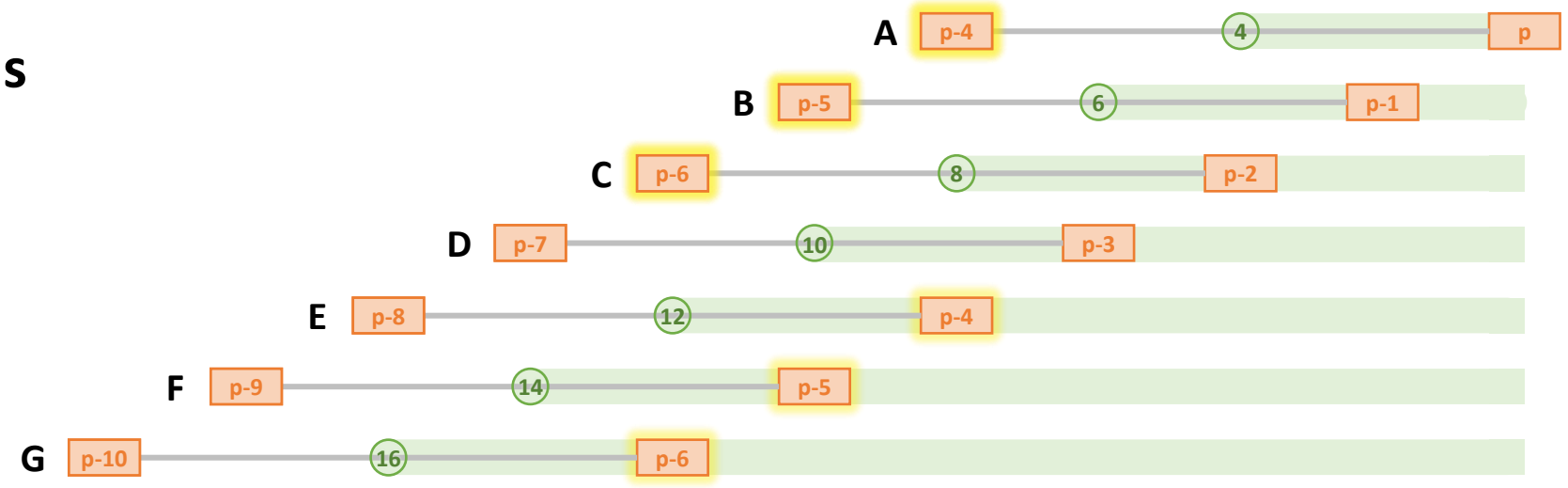


mNRR Smoothing – Combining N & M

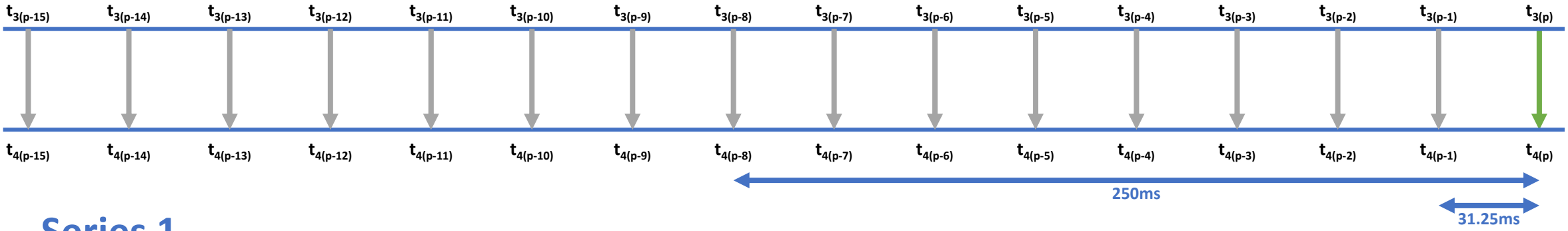


Series 1

pDelay Interval = 62.5ms
N = 4
M = 7



mNRR Smoothing – Combining N & M

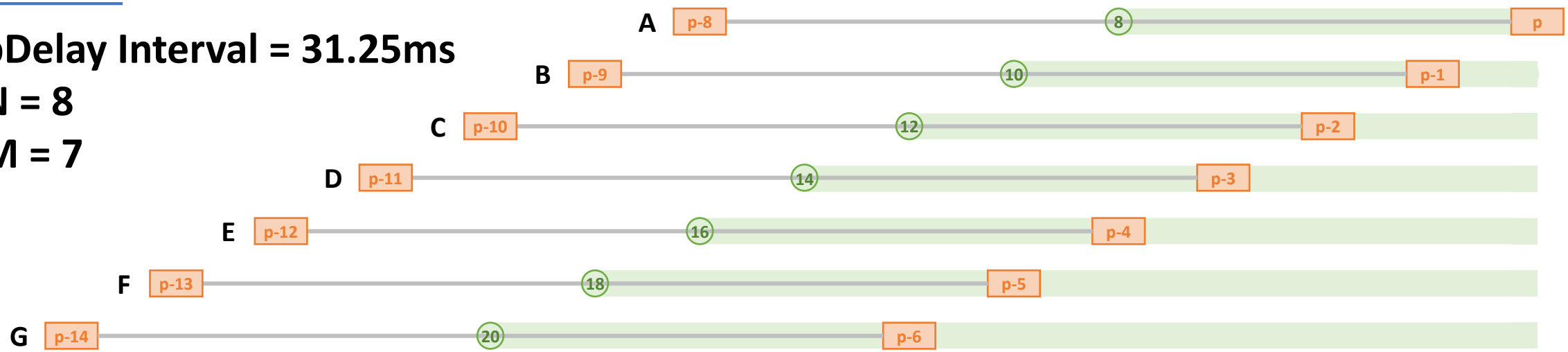


Series 1

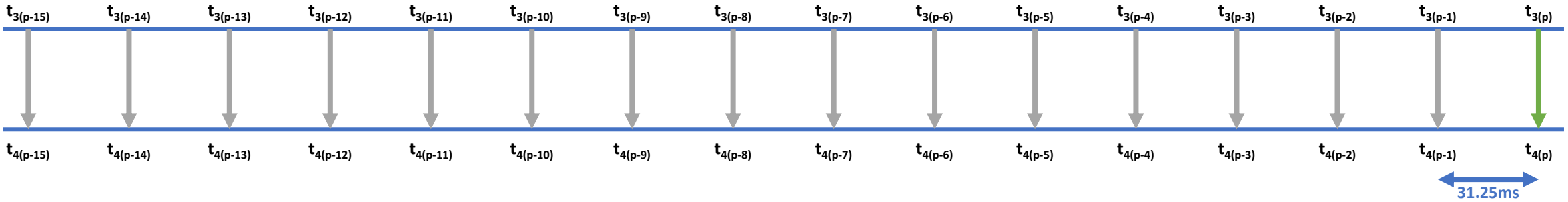
pDelay Interval = 31.25ms

N = 8

M = 7

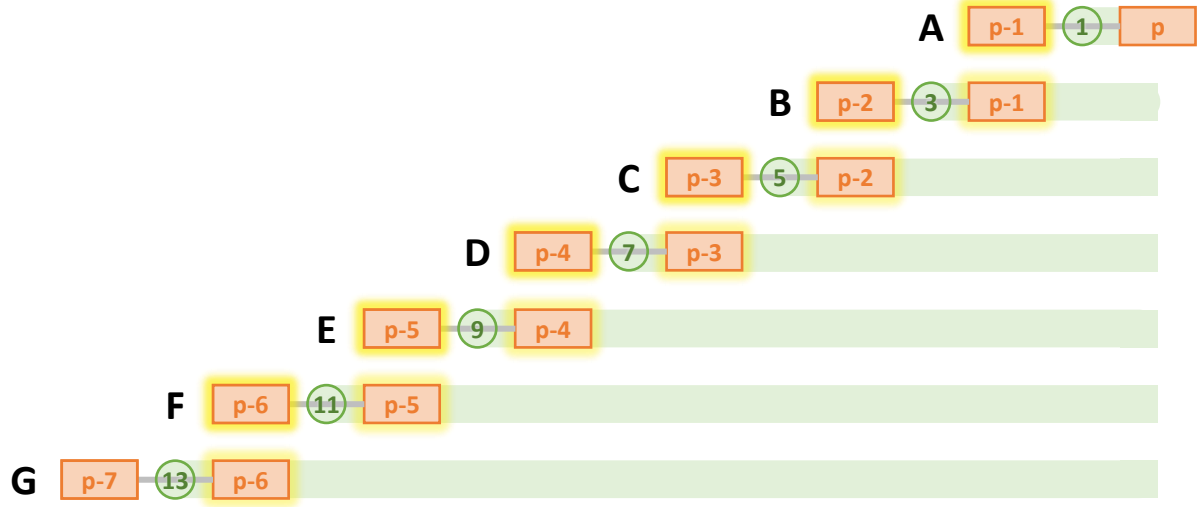


mNRR Smoothing – Combining N & M

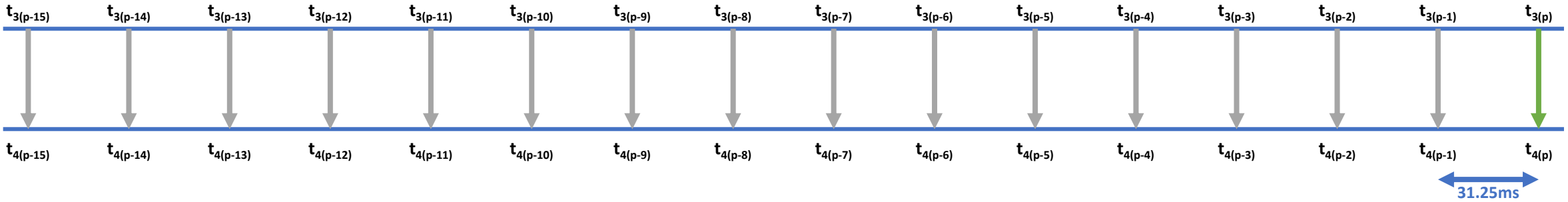


Series 2

pDelay Interval = 31.25ms
N = 1
M = 7

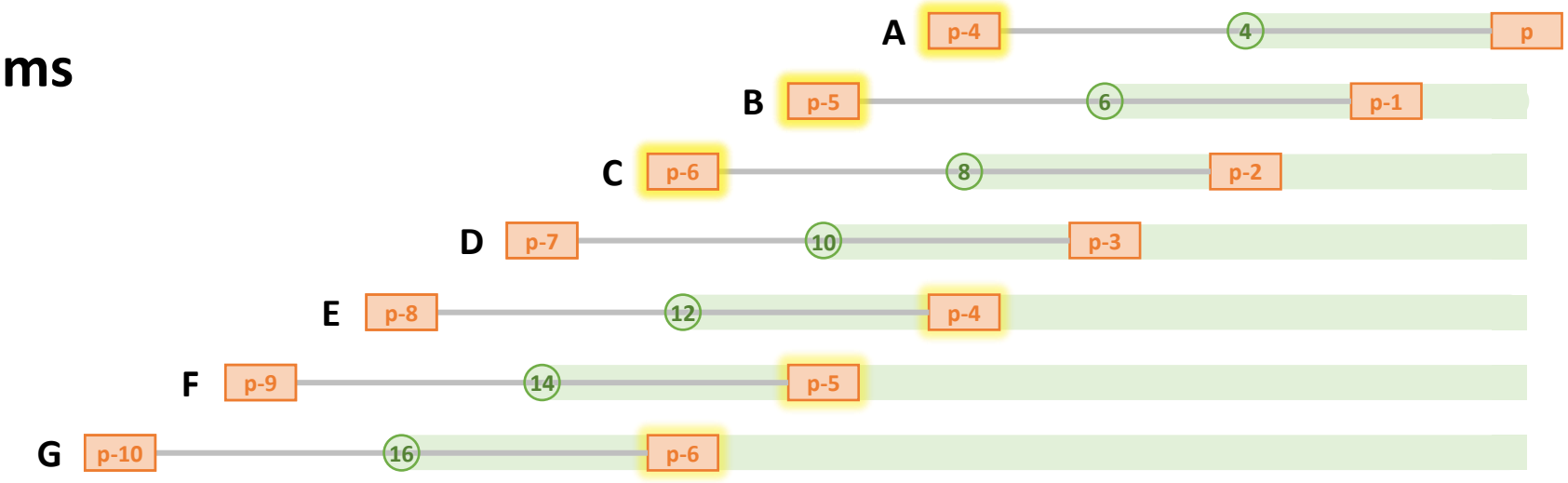


mNRR Smoothing – Combining N & M

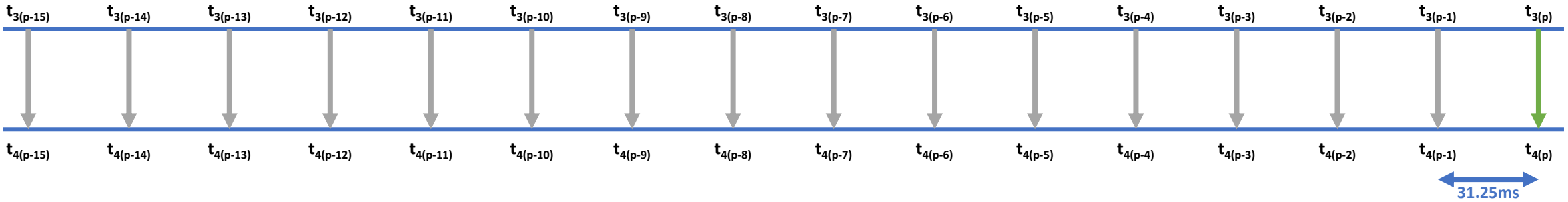


Series 2

pDelay Interval = 31.25ms
N = 4
M = 7

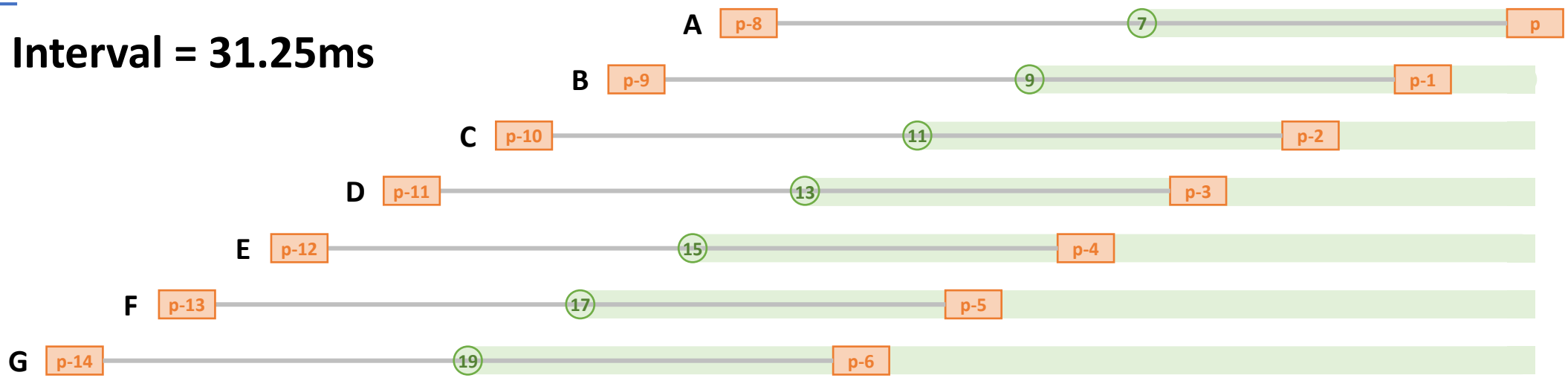


mNRR Smoothing – Combining N & M



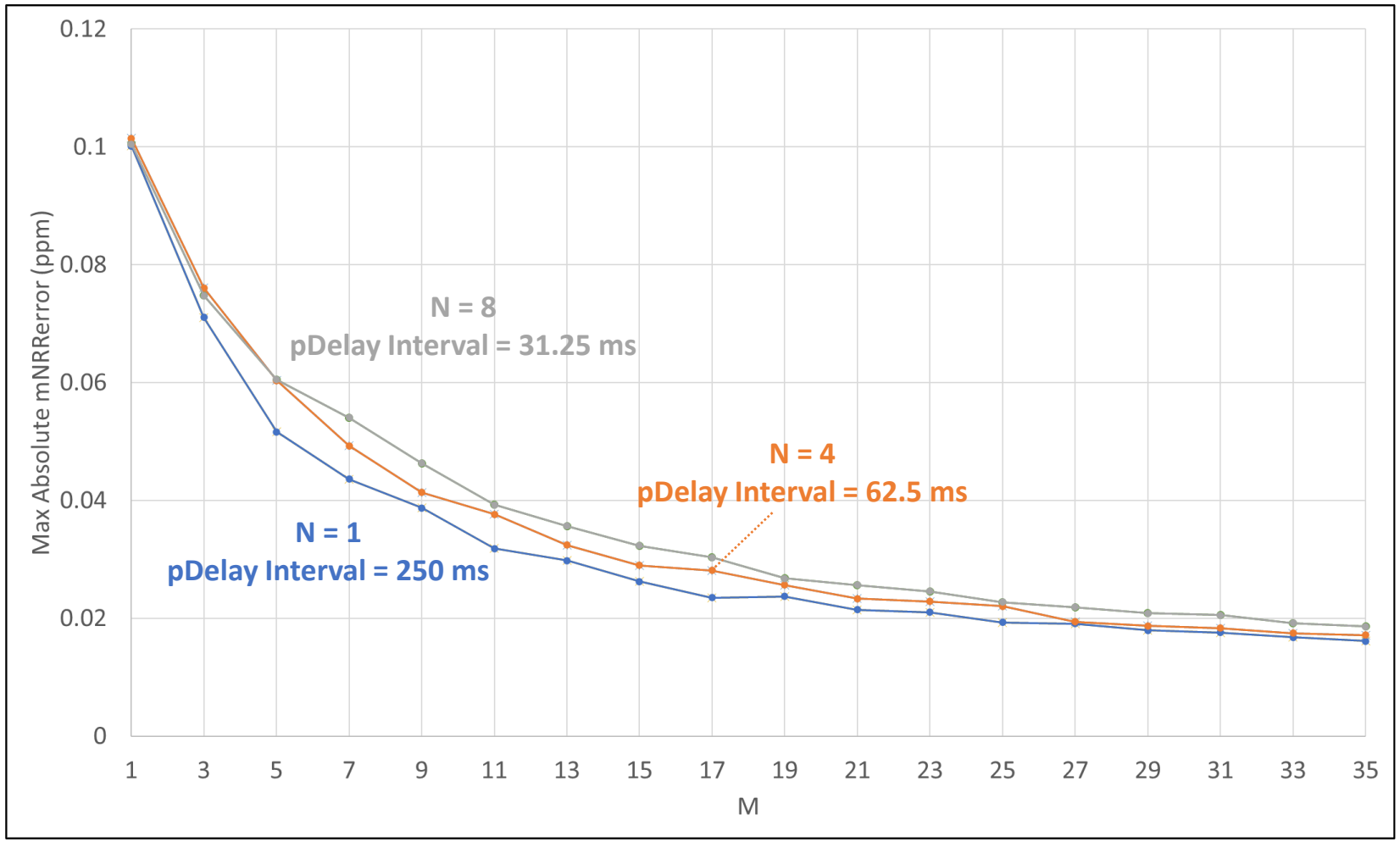
Series 2

pDelay Interval = 31.25ms
N = 7
M = 7



Varying M & N – No Clock Drift

(N x pDelayInterval maintained at 250ms)

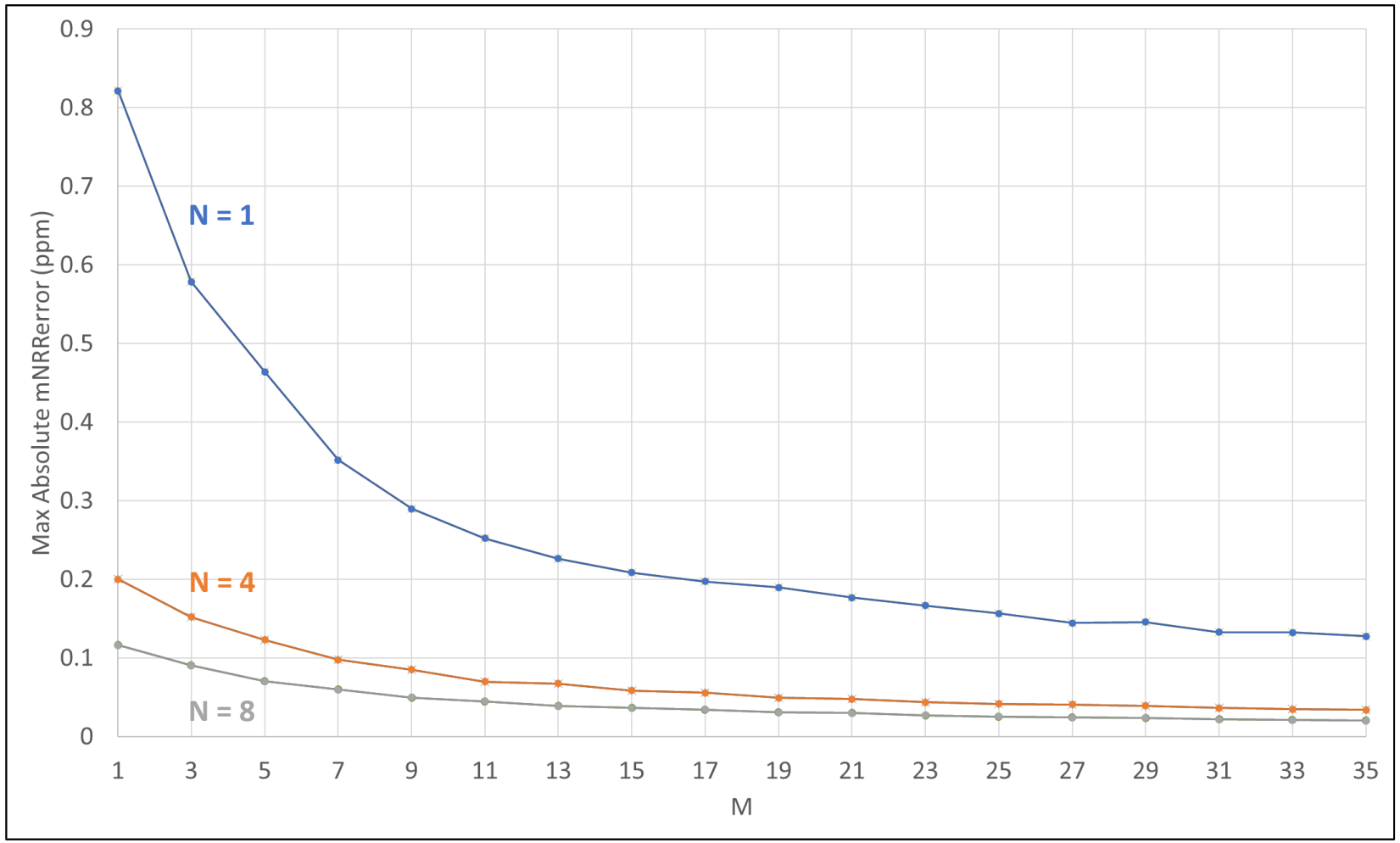


Input Errors		
Clock Drift	±0	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

Varying M & N – No Clock Drift

(N x pDelayInterval maintained at 250ms)

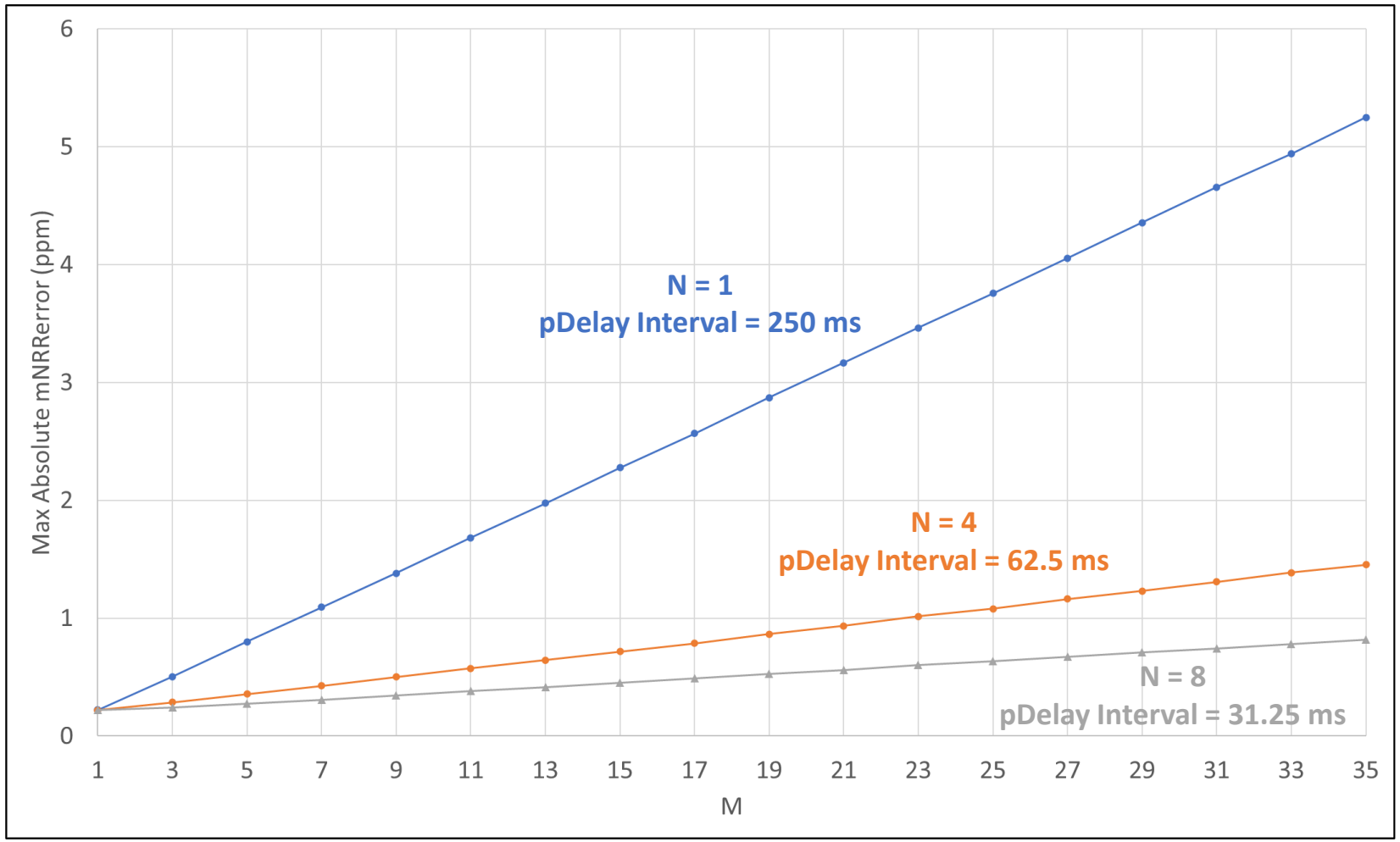


Input Errors		
Clock Drift	±0.6	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Input Parameters		
pDelay Interval	31.25	ms
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

Varying M & N – With Clock Drift

(N x pDelayInterval maintained at 250ms)

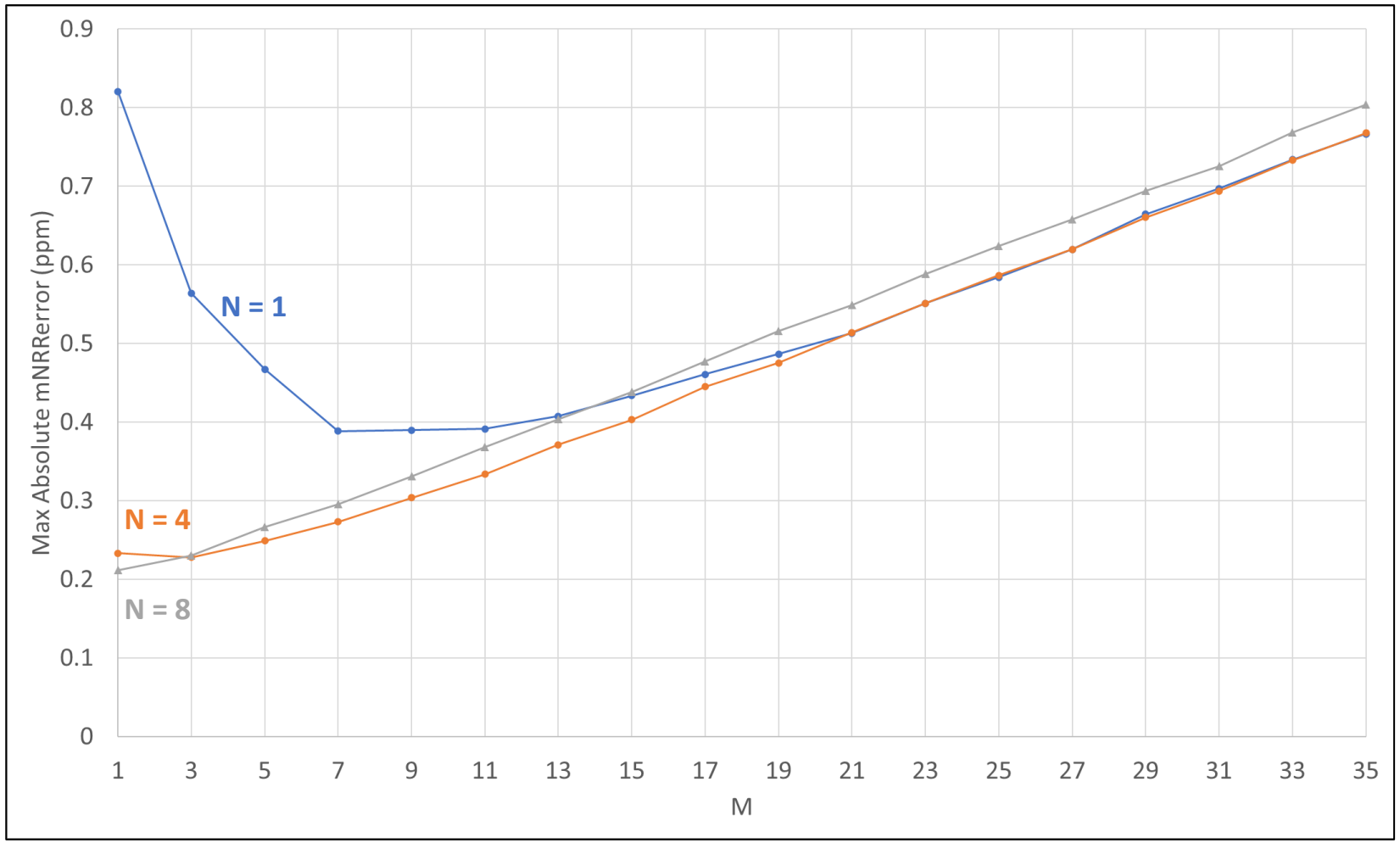


Input Errors		
Clock Drift	±0	ppm/s
Timestamp Granularity Error	±4	ns
Dynamic Time Stamp Error	±4	ns
Configuration		
Hops	1	
Runs (per value of M)	100,000 x 10	

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

Varying M & N – With Clock Drift

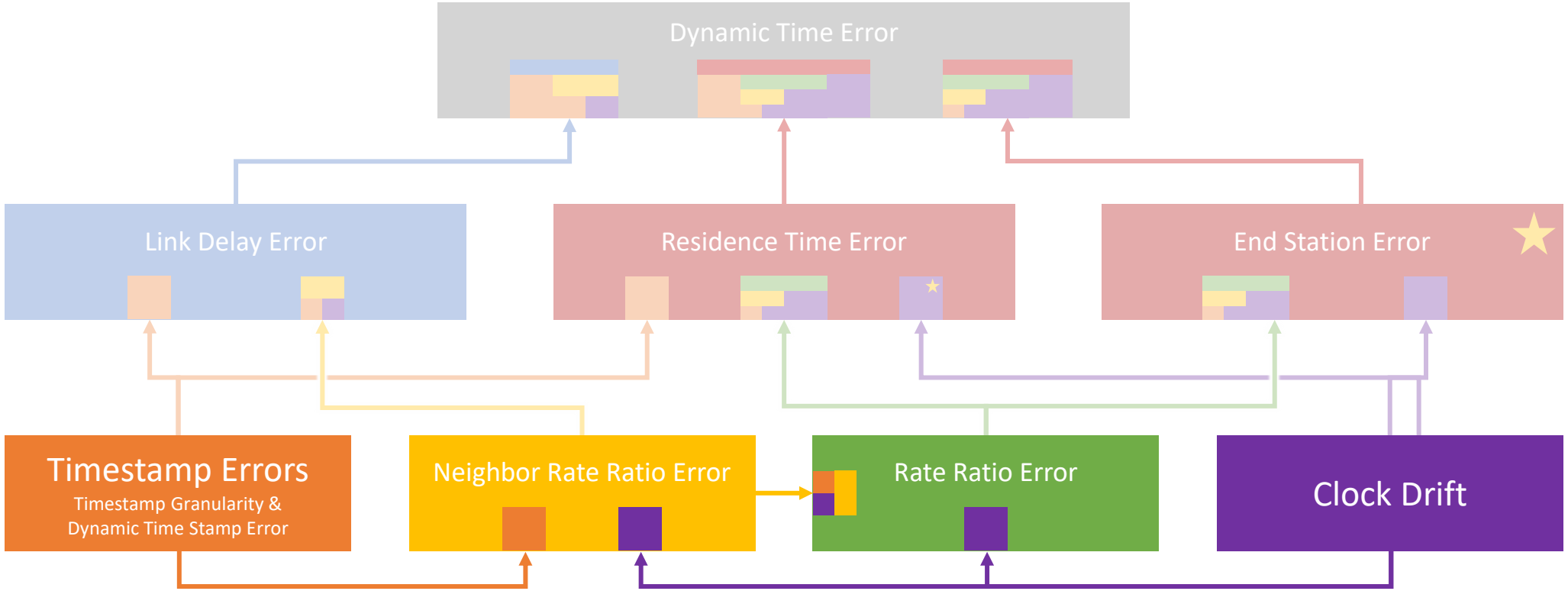
(N x pDelayInterval maintained at 250ms)



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

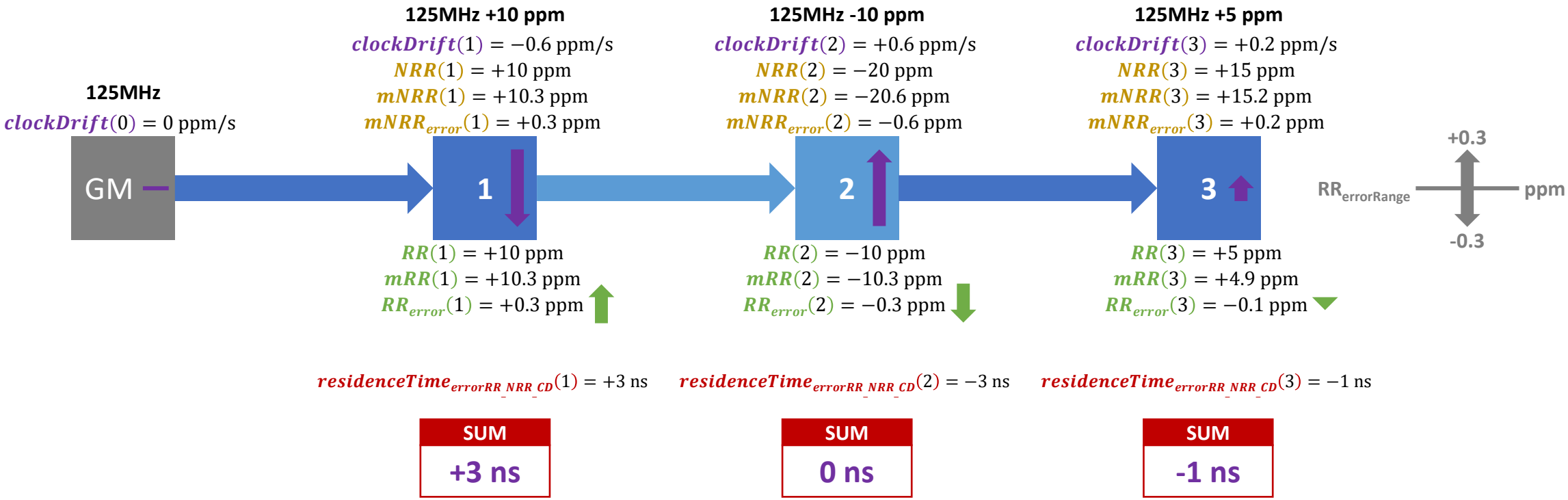
Optimal values of M (for N=1, pDelay Interval 31.25ms) are **between 7 & 13**. (Optimal for $mNRR_{error}$, not necessarily DTE.) That translates to using NRR calculations from up to 218.75ms and 406.25ms in the past...which is a similar magnitude as the optimal pDelay Interval value of 250ms.

NRR Errors Accumulate in RR Error

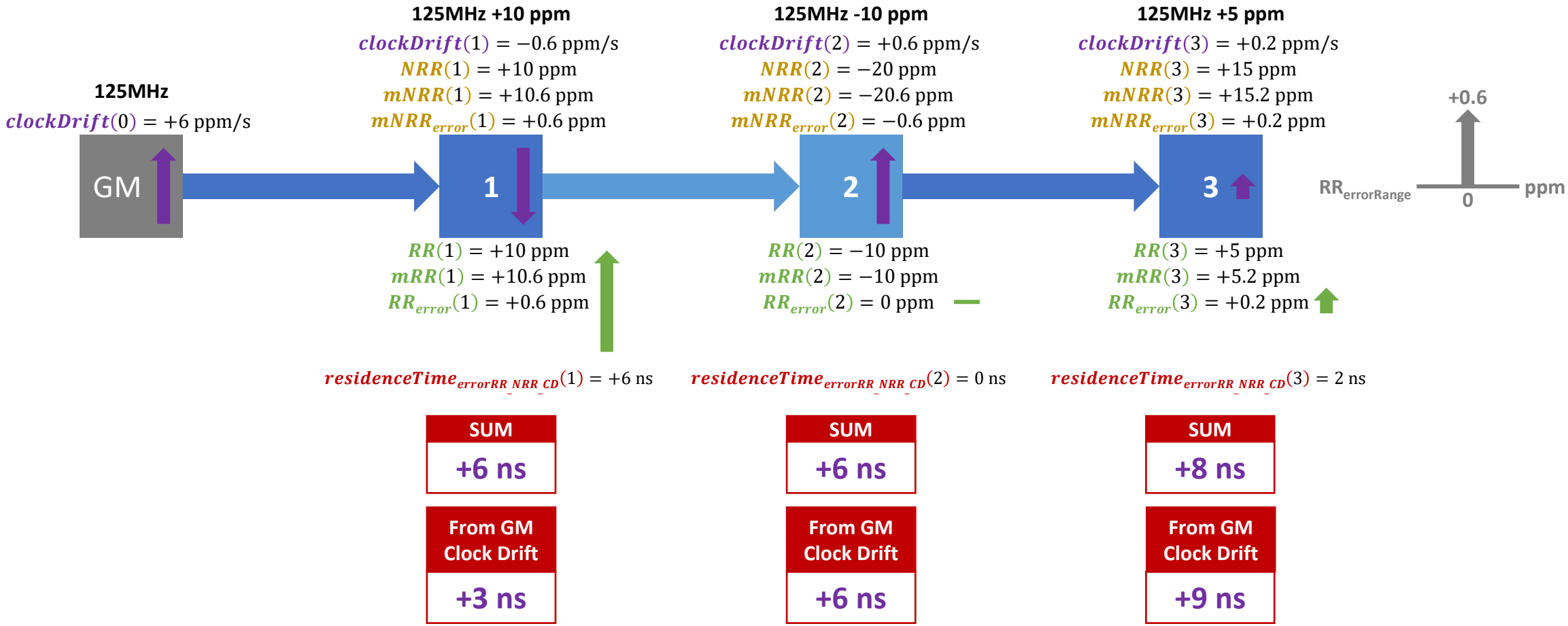


But there is a big caveat for how NRR errors due to Clock Drift accumulate...

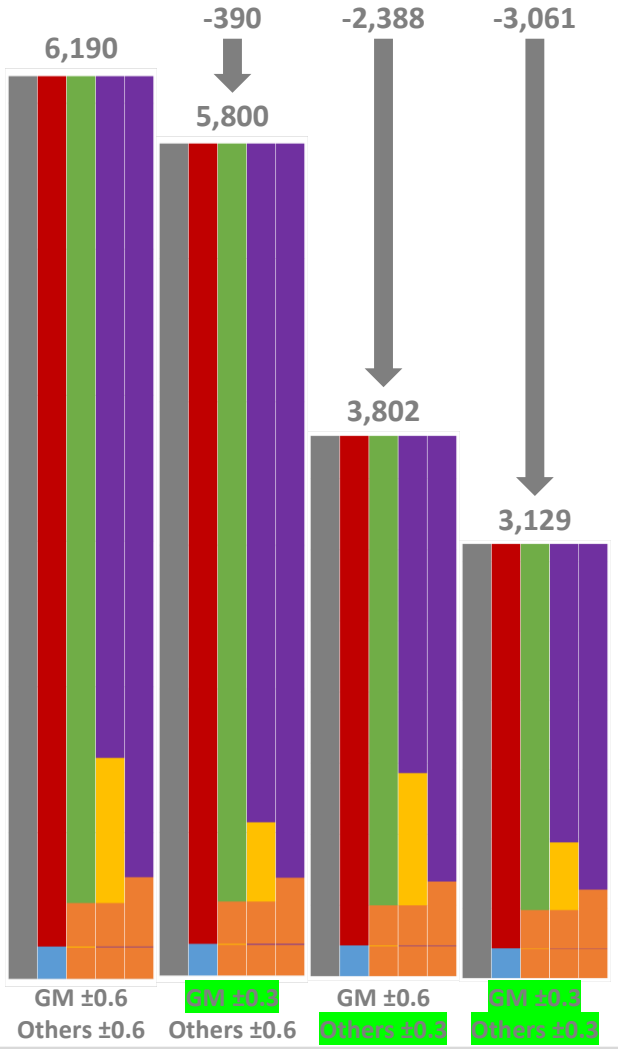
How *residenceTime*$_{errorRR\_NRR\_CD}$ Accumulates



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



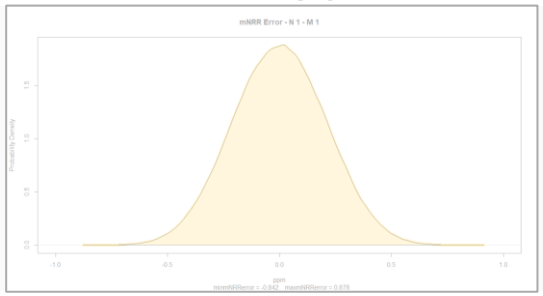
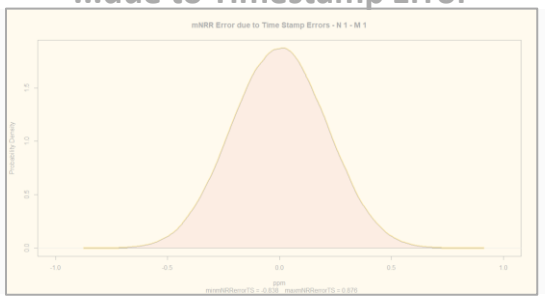
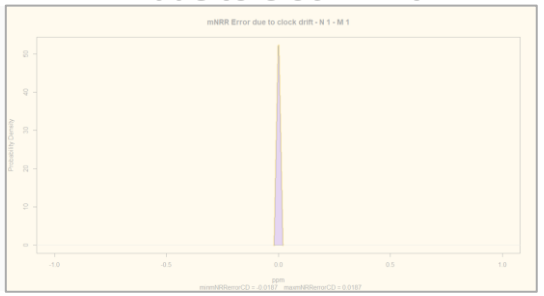
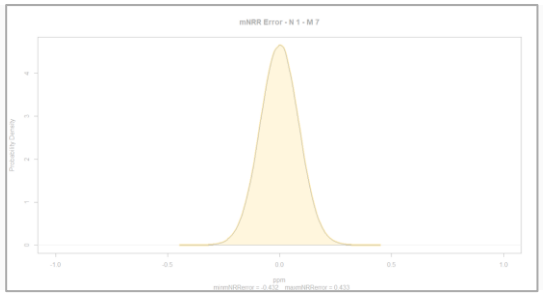
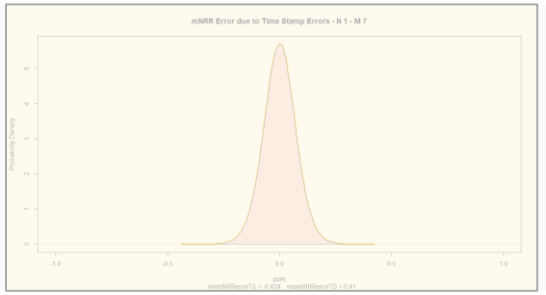
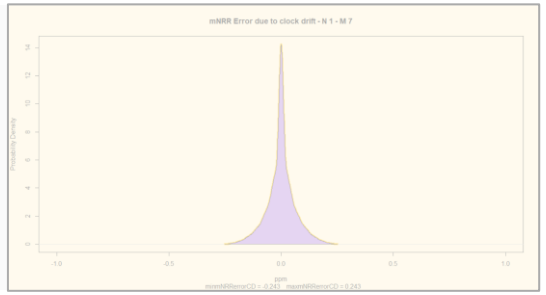
Clock Drift Sensitivity

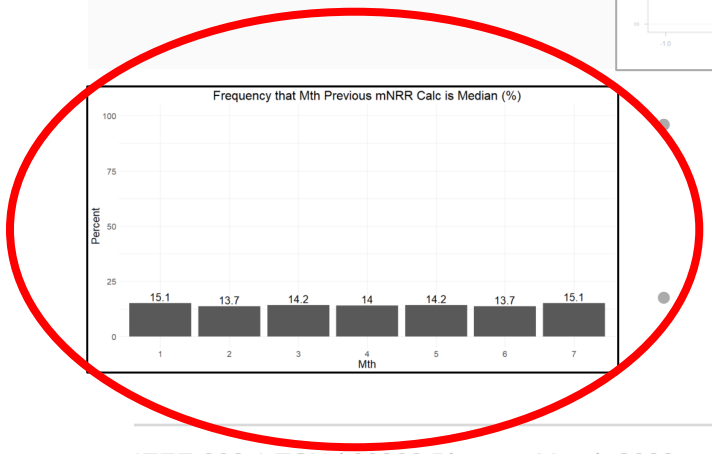


Input Errors		
GM Clock Drift Max	Variable	ppm/s
GM Clock Drift Min	Variable	ppm/s
Clock Drift (non-GM)	Variable	±ppm/s
Timestamp Granularity TX	4	±ns
Timestamp Granularity RX	4	±ns
Dynamic Time Stamp Error TX	4	±ns
Dynamic Time Stamp Error RX	4	±ns
Input Parameters		
pDelay Interval	1000	ms
pDelay Response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

Effect of Taking Median of mNRR Calculations

(mNRR – Single Hop – N=1)

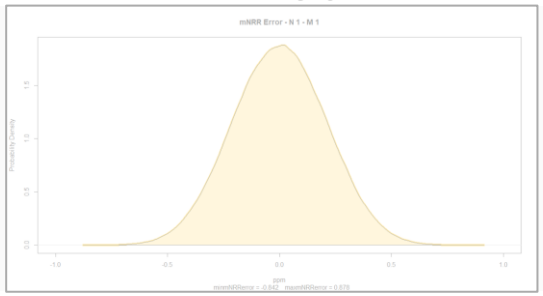
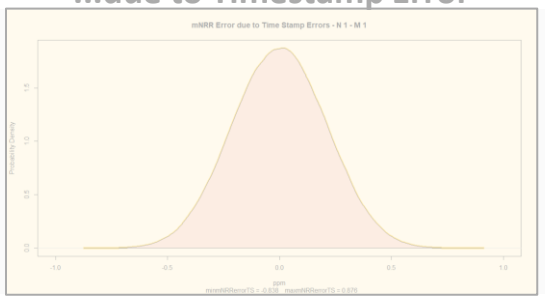
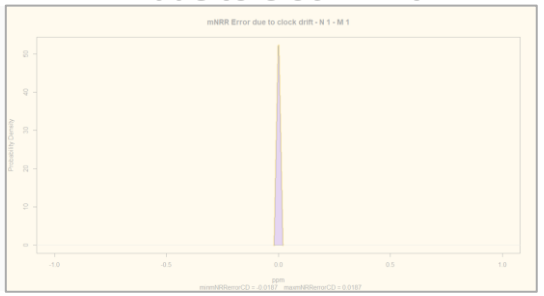
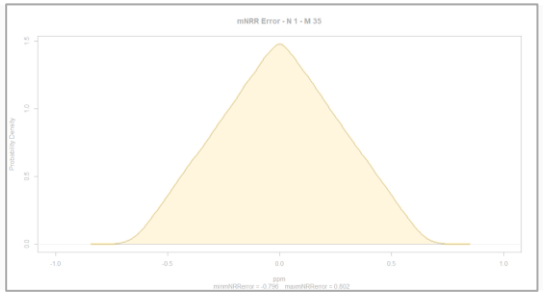
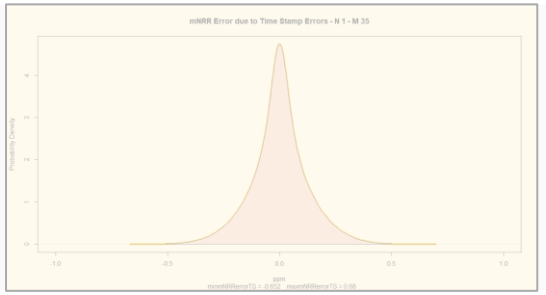
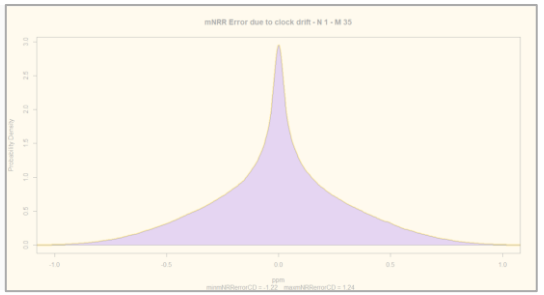
	pDelay Interval	$mNRR_{error}$	...due to Timestamp Error	...due to Clock Drift
M=1	31.25ms			
M=7	31.25ms			

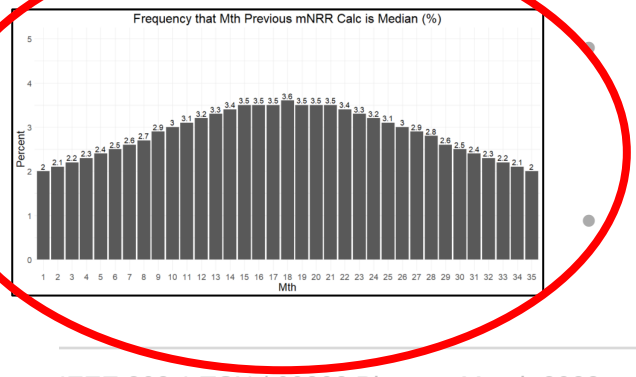


- For values of pDelay Interval where $mNRR_{error}$ due to Timestamp error dominates error due to Clock Drift, taking the median of previous NRR calculations can reduce $mNRR_{error}$.
- This raises the question: what is the optimal value of M?

Effect of Taking Median of mNRR Calculations

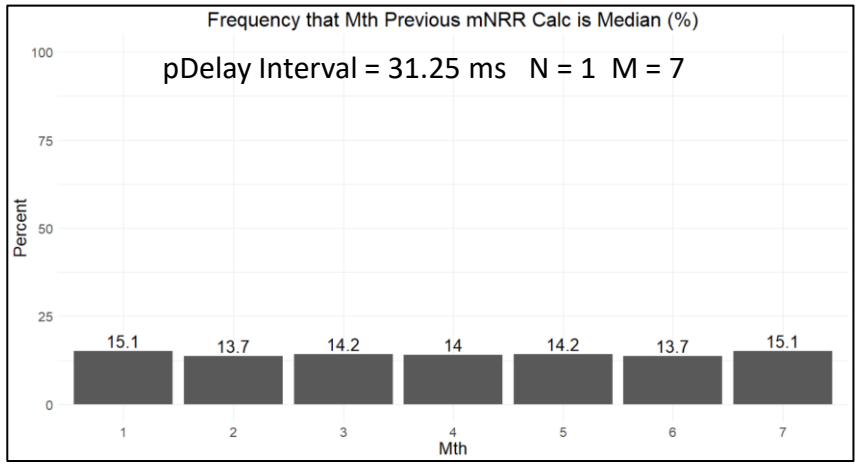
(mNRR – Single Hop – N=1)

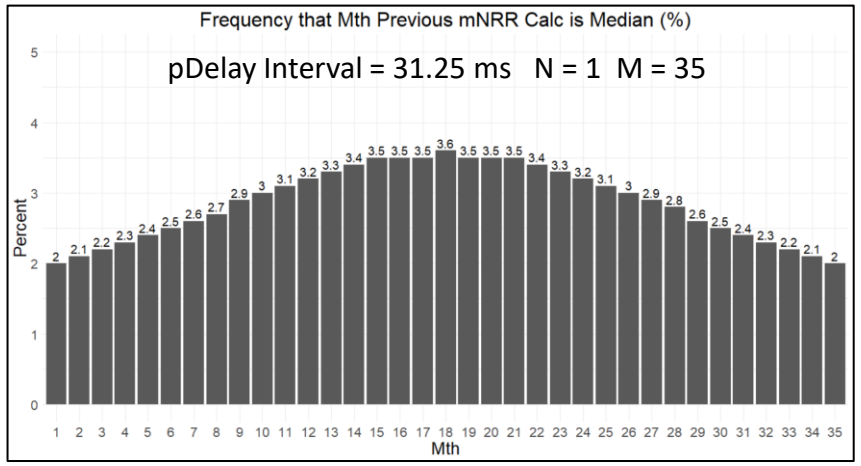
	pDelay Interval	$mNRR_{error}$	...due to Timestamp Error	...due to Clock Drift
M=1	31.25ms			
M=35	31.25ms			



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

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





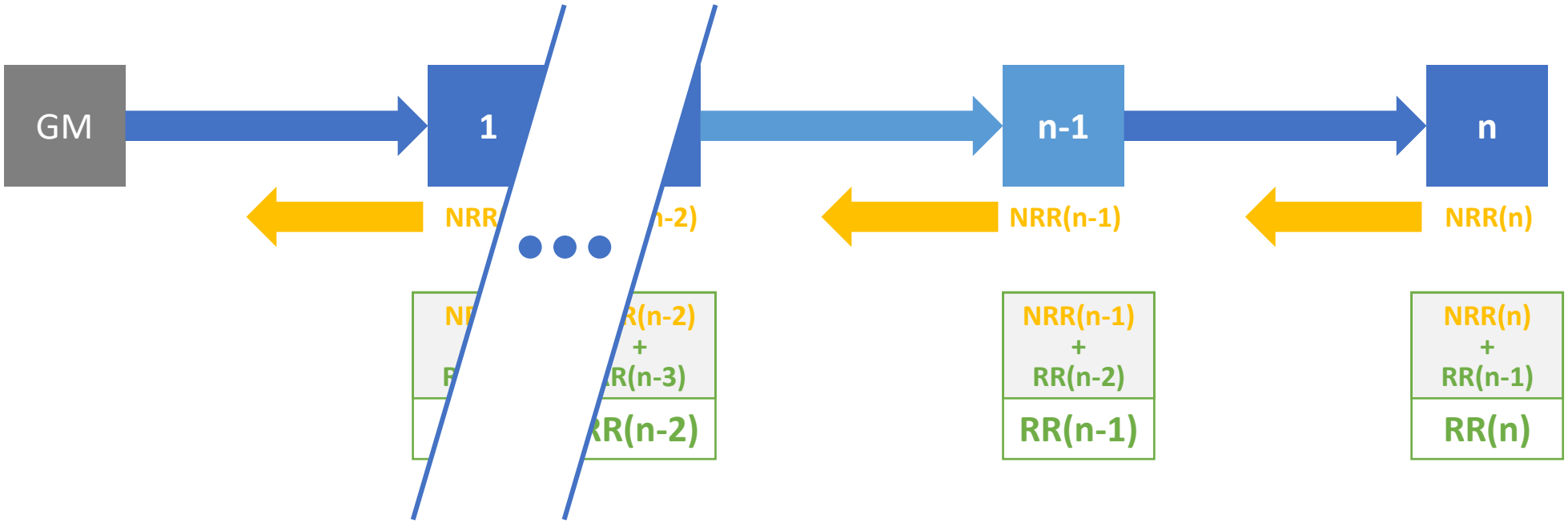
- Using the median of past NRR calculations brings with it a chance that a larger portion of the error won't cancel out at the next node but will instead survive.
  - This portion will behave more like mNRRerror due to Clock Drift at the GM than at other nodes when M=1.

Summary

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

Rate Ratio Error

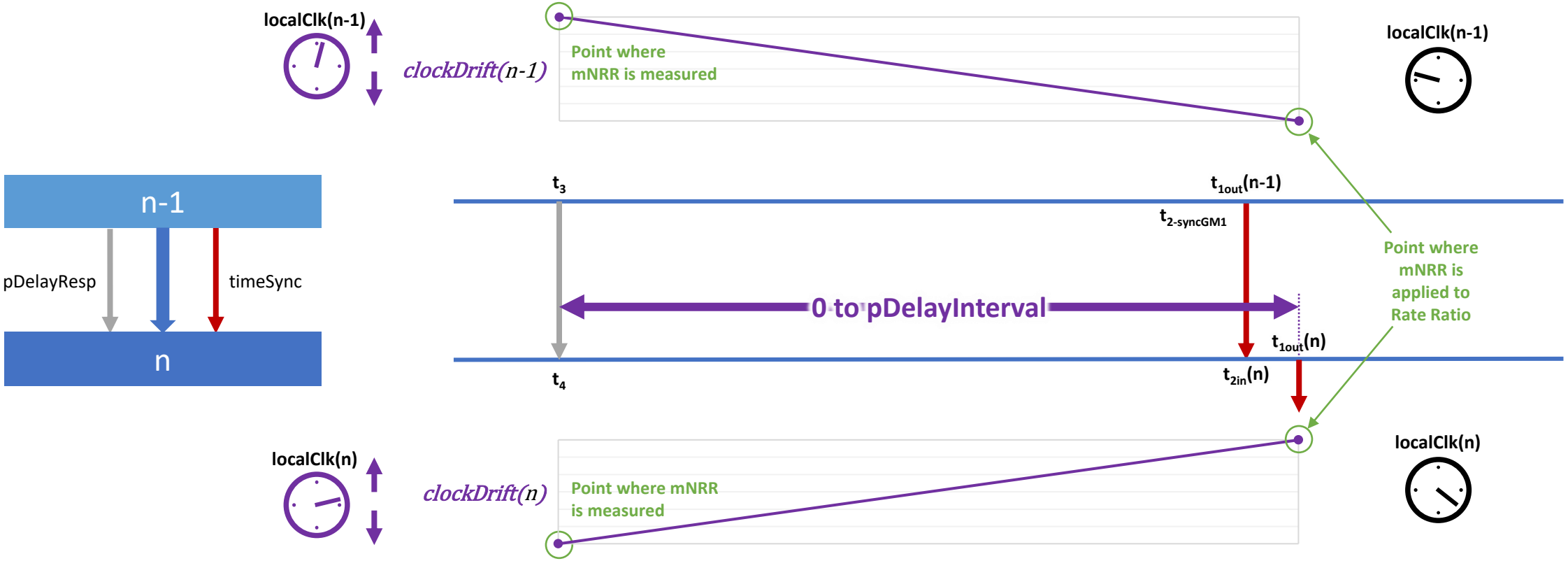
Rate Ratio Error (RR_{error})



$$RR(n) = RR(n-1) + mNRR(n)$$

$$RR_{error}(n) = RR_{error}(n-1) + mNRRer_{ror}(n) + RRer_{rorCD}(n)$$

$RR_{errorCD}$



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

ppm

Formulae – RR_{error}

$$RR_{error}(n) = RR_{error}(n-1) + mNRRerror(n) + RR_{errorCD}(n) \quad \textbf{ppm}$$

$$RR_{errorCD}(n) = (1 - \boldsymbol{driftRate_{errorCorrection}}) \times \frac{delay_{mNRR\_Sync}}{10^3}\left(\boldsymbol{clockDrift}(n-1) - \boldsymbol{clockDrift}(n)\right) \quad \textbf{ppm}$$

$$delay_{mNRR\_Sync} \sim U(0, (1 - \boldsymbol{pDelayRespSync_{correction}})\boldsymbol{pDelayInterval}) \quad \textbf{ms}$$

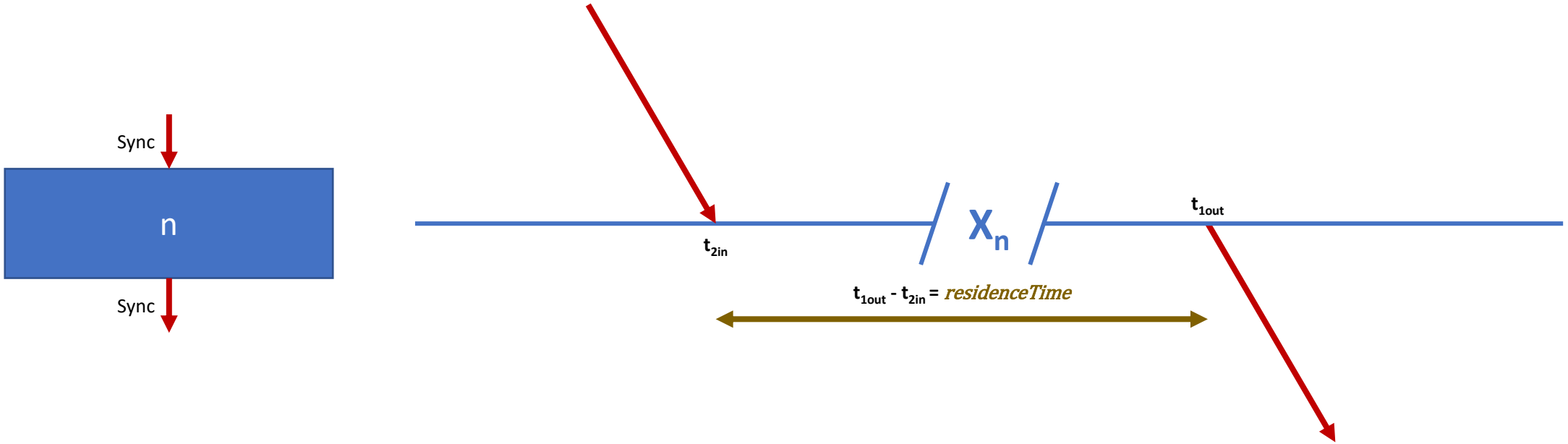
Does not include any effect of changing clock drift (ppm/s²) during Residence Time. See speaker notes in RR_{error} section for details.

Observations of RR_{error} Behaviour

- $mNRR_{error}$ feeds directly into RR_{error} where the errors accumulate
  - Since $mNRR_{error}$ due to clock drift ($mNRR_{errorCD}$) at one node tends to reverse at the next node, the component of RR_{error} due to this ($RR_{errormNRR\_CD}$)will tend not to increase along a chain of devices.
    - This does not apply for error due to GM clock drift as it only appears in $mNRR_{error}$ at the first hop.
  - $mNRR_{error}$ due to timestamp errors are much less likely to cancel out so RR_{error} from this source is more likely to increase along a chain of devices.
- RR_{error} due to clock drift during the delay between measurement of NRR and when it is applied to Rate Ratio ($RR_{errorCD}$) could cancel out from one node to the next…but only if the delay at one node is the same as at the next…which is unlikely.
  - $RR_{errorCD}$ is therefore much more likely to increase along a chain of devices than the clock drift component of $mNRR_{error}$
- RR_{error} due to clock drift can be reduced by decreasing *pDelayInterval* which reduces the maximum possible $delay_{mNRRSync}$
  - Similar effect if pDelay messaging can be aligned to occur just before Sync message; modelled using $pDelayRespSync_{correction}$
  - Note: see earlier in this presentation for the effect reducing *pDelayInterval* has on $mNRR_{error}$

Residence Time Error

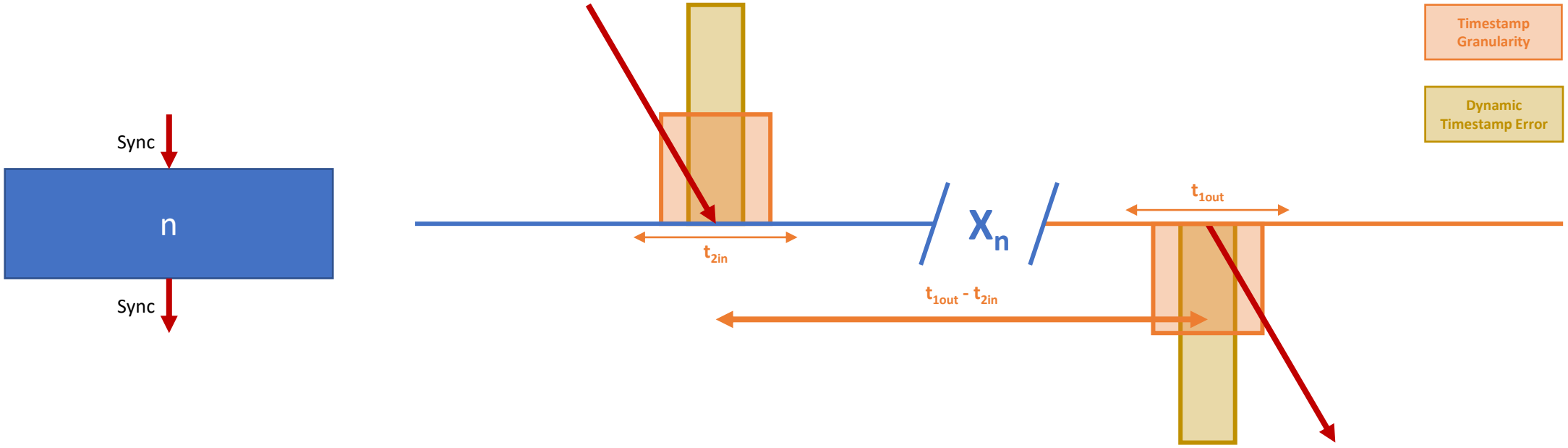
Residence Time Error ($residenceTime_{error}$)



$$residenceTime = RR(t_{1out} - t_{2in})$$ **ns**

$$residenceTime_{error} = residenceTimeerrorT_S + residenceTimeerrorR_R$$ **ns**

$RT_{errorTS}$

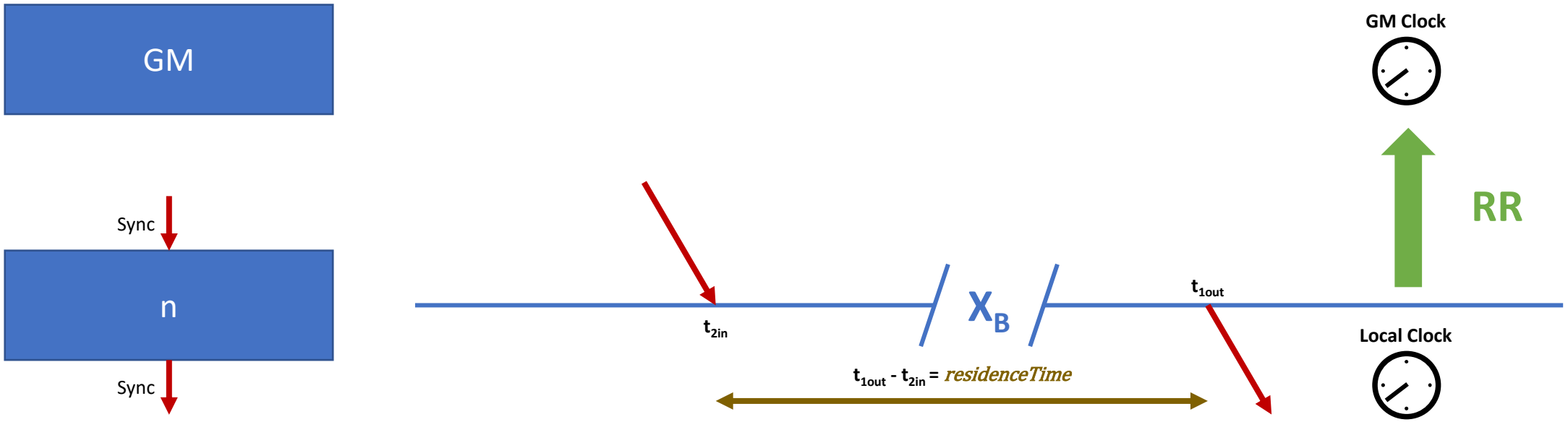


$$residenceTime_{errorTS} = t_{1Serror} - t_{2Serror}$$

ns

residenceTime$_{errorRR}$

Due to Errors in RR Field (accumulation of NRR)

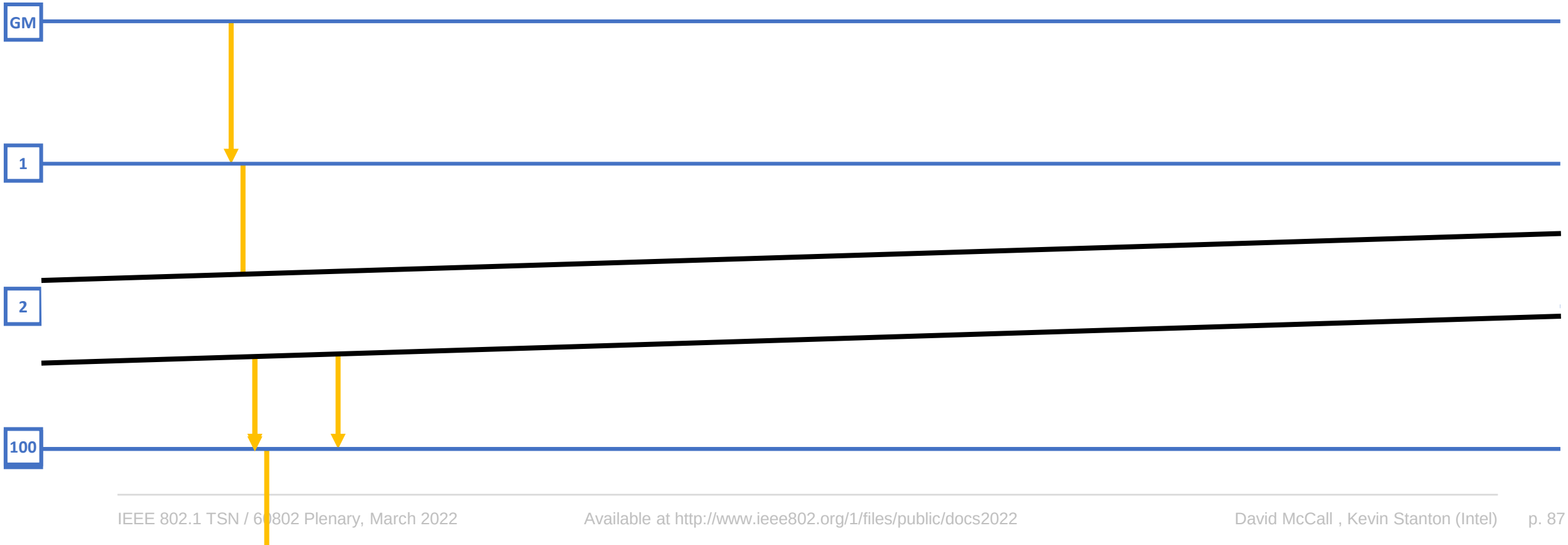


$$residenceTime_{errorRR} = \frac{RR_{error}}{10^6}(\boldsymbol{residenceTime} \times 10^6)$$ **ns**

$$= RRerror \times \boldsymbol{residenceTime}$$ **ns**

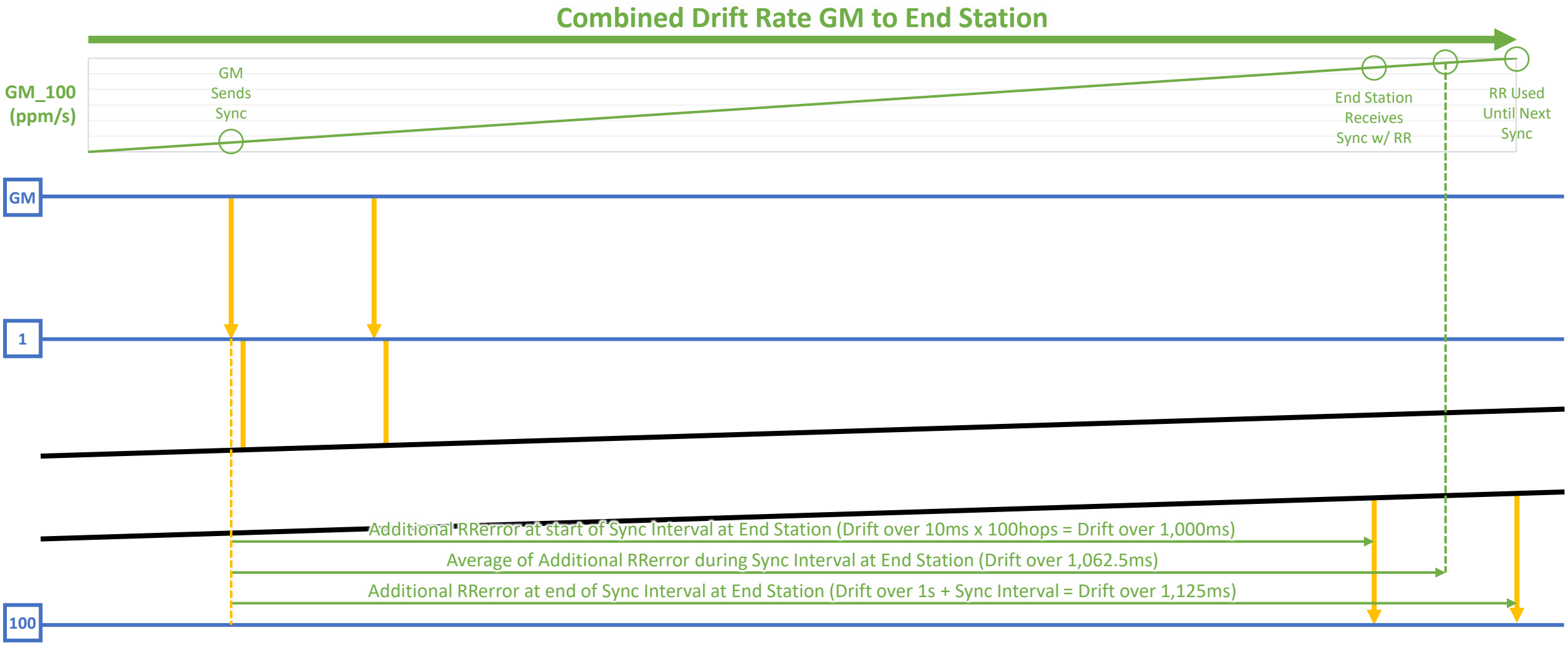
endStation$_{errorCD_duringSync}$

Error due to GM/LocalClock Drift during Sync Messaging



endStation$_{errorCD_duringSync}$

Error due to GM/LocalClock Drift during Sync Messaging



$endStation_{errorCD\_duringSync}$

- Same principle as $endStation_{errorCD\_duringSync}$
- Error due to delay between GM sending Sync and current node processing Sync
- Delay approximates to sum of Residence Times up to and including the current node
  - Also Mean Link Delay times, but those are typically 4 to 5 orders of magnitude smaller and can safely be left out of the Monte Carlo model
  - In the current model the Residence Time at each node is always the nominal value
- Delay generates additional error in the Rate Ratio, but this error does not accumulate in the way that errors in the Rate Ratio field accumulate down the chain
  - This is an additional error between the calculated RR (which already has other errors) and the actual RR
- Amount of error that feeds into the Correction Field is mostly via the multiplication of the Residence Time by the Rate Ratio
  - Again: also multiplication of the Mean Link Delay by Rate Ratio, but those errors are typically 4 to 5 orders of magnitude smaller and can safely be left out of the Monte Carlo model

Formulae – $residenceTime_{error}$

$$residenceTime_{error} = residenceTimee_{rrorTS} + residenceTimee_{rrorRR} + residenceTimeer_{rorCD\_duringSync} \quad \textbf{ns}$$

$$residenceTime_{errorTS} = t_{1Serror} - t_{2Serror} \quad \textbf{ns}$$

$$t_{2Serror} = \boldsymbol{TSGE_{RX}} + \boldsymbol{DTSE_{RX}} \qquad t_{1Serror} = \boldsymbol{TSGE_{TX}} + \boldsymbol{DTSE_{TX}} \quad \textbf{ns}$$

$$residenceTime_{errorRR} = \frac{RR_{error}}{10^6}(\boldsymbol{residenceTime} \times 10^6 + residenceTime_{errorTS}) \quad \textbf{ns}$$

$$= RRerror \times \left(\boldsymbol{residenceTime} + \frac{residenceTime_{errorTS}}{10^6}\right) \quad \textbf{ns}$$

$$residenceTime_{errorCD\_duringSync} = RR_{errorCD\_duringSync}\left(\boldsymbol{residenceTime} + \frac{residenceTime_{errorTS}}{10^6}\right) \quad \textbf{ns}$$

$$RR_{errorCD\_duringSync} = \left(\frac{hop \times residenceTime}{1000}\right) \times (clockDriftGM - clockDrift) \times (1 - \boldsymbol{driftRateRR_{errorCorrection}}) \quad \textbf{ppm}$$

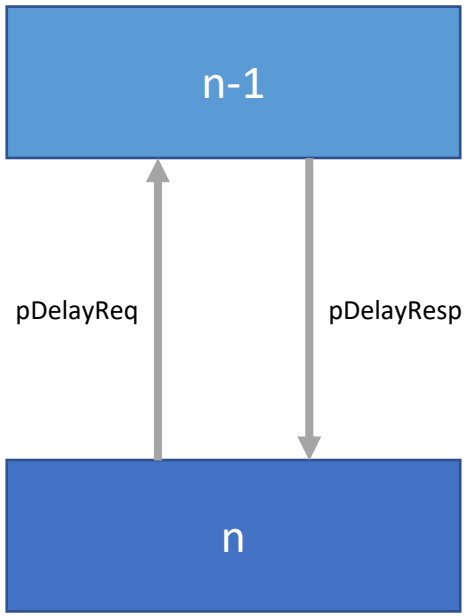
Observations of $residenceTime_{error}$ Behaviour

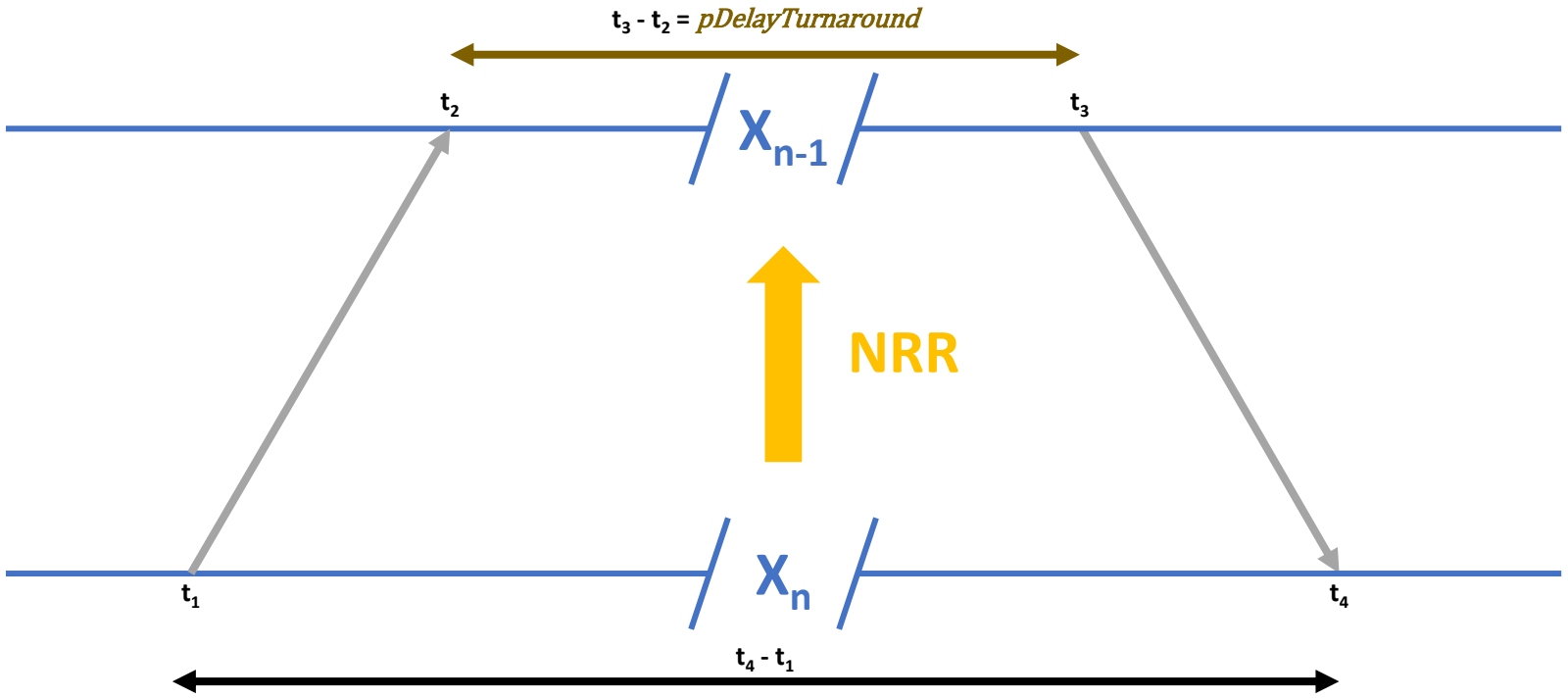
- $residenceTime_{error}$ due to timestamp error is independent of Residence Time.
  - Reducing Residence Time will have no effect on this source of error.
- $residenceTime_{error}$ due to RR_{error} is proportional to Residence Time
  - Reducing Residence Time can reduce this source of error.
- Since larger RR_{error} is more likely further along a chain of devices, the amount of Residence Time Error at each node is also likely to increase.
  - For example: the component of DTE due to Residence Time Error after 100 hops is more likely to be larger from errors in nodes 51-100 than from nodes 1-50.

End Station Error

Mean Link Delay Error

meanLinkDelay

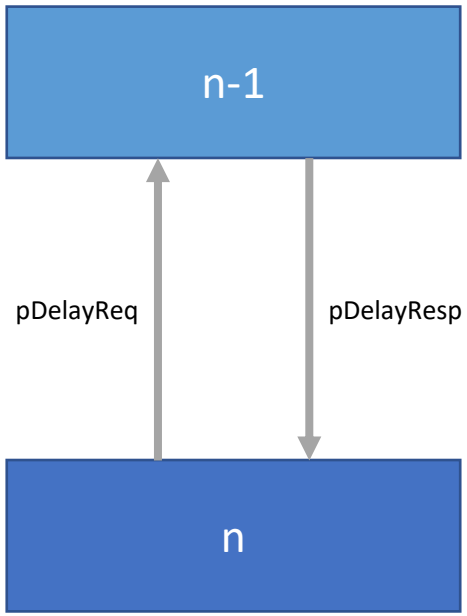


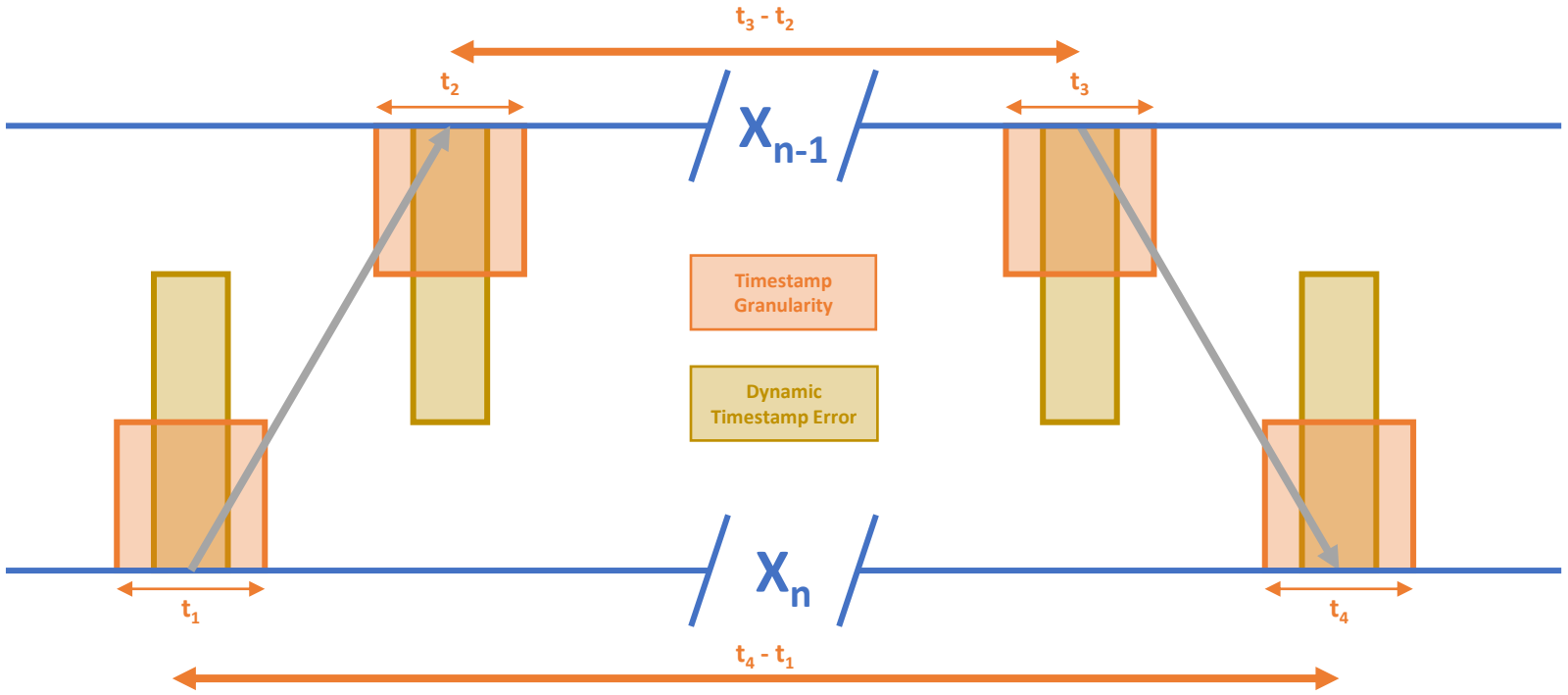


$$meanLinkDelay = RR\left(\frac{(t_4 - t_1) - NRR(t_3 - t_2)}{2}\right)$$ **ns**

$$meanLinkDelay_{error} = (1 - \boldsymbol{meanLinkDelay_{errorCorrection}})(pDelay_{errorTS} + pDelay_{errorNRR} + pDelay_{errorRR})$$ **ns**

$pDelay_{errorTS}$

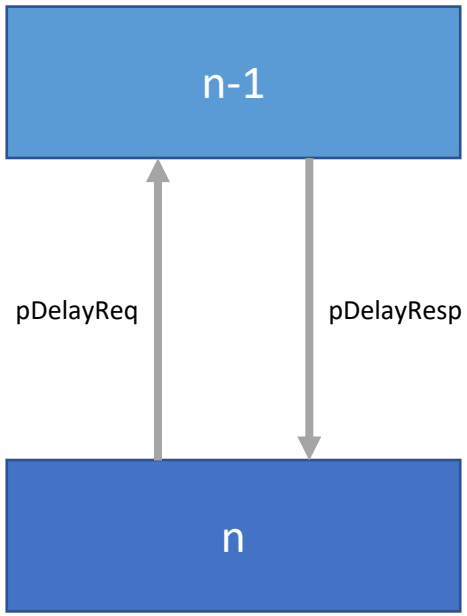


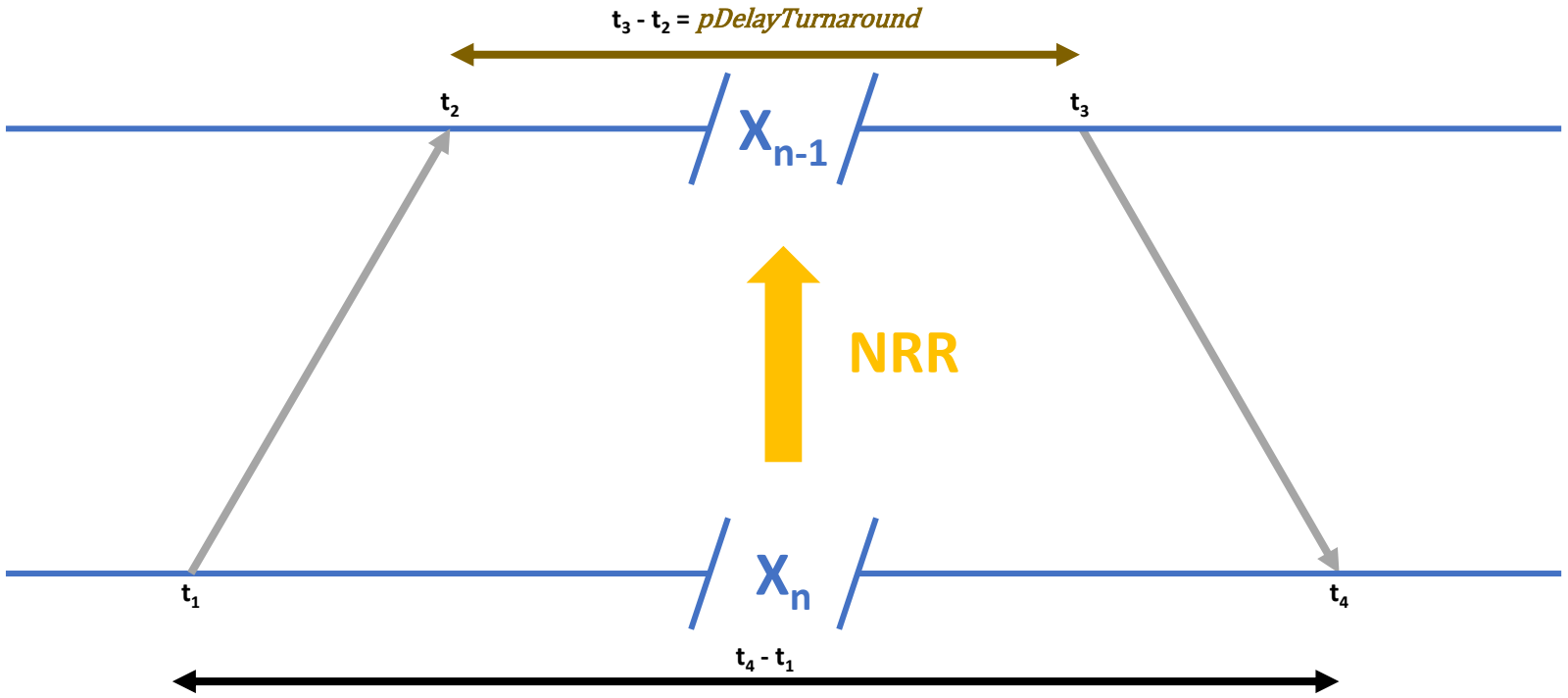


$$pDelay_{errorTS} = \frac{(t_{4PDerror} - t_{1PDerror}) - (t_{3PDerror} - t_{2PDerror})}{2}$$

ns

$pDelay_{errorNRR}$





$$pDelay_{errorNRR} = mNRR_{error}\left(\frac{\boldsymbol{pDelayTurnaround}}{2}\right)$$

ns

Formulae – $meanLinkDelay_{error}$

$$meanLinkDelay_{error} = (1 - \boldsymbol{meanLinkDelay_{errorCorrection}})(pDelay_{errorTS} + pDelay_{errorNRR} + pDelay_{errorRR}) \quad \textbf{ns}$$

$$pDelay_{errorTS} = \frac{(t_{4PDerror} - t_{1PDerror}) - (t_{3PDerror} - t_{2PDerror})}{2} \quad \textbf{ns}$$

$$t_{1PDerror} = \boldsymbol{TSGE_{TX}} + \boldsymbol{DTSE_{TX}} \qquad t_{2PDerror} = \boldsymbol{TSGE_{RX}} + \boldsymbol{DTSE_{RX}} \quad \textbf{ns}$$

$$t_{3PDerror} = \boldsymbol{TSGE_{TX}} + \boldsymbol{DTSE_{TX}} \qquad t_{4PDerror} = \boldsymbol{TSGE_{RX}} + \boldsymbol{DTSE_{RX}} \quad \textbf{ns}$$

$$pDelay_{errorNRR} = \frac{mNRR_{error}}{10^6}\left(\frac{\boldsymbol{pDelayTurnaround} \times 10^6}{2}\right) \quad \textbf{ns}$$

$$= mNRR_{error}\left(\frac{\boldsymbol{pDelayTurnaround}}{2}\right) \quad \textbf{ns}$$

$$pDelay_{errorRR} = \frac{RR_{error}}{10^6}\left(\boldsymbol{pDelay} + (1 - \boldsymbol{pDelay_{errorCorrection}})(pDelay_{errorTS} + pDelayerrorN_{RR})\right) \quad \textbf{ns}$$

Observations of $meanLinkDelay_{error}$ Behaviour

- $meanLinkDelay_{error}$ due to timestamp error is independent of pDelayTurnaround.
  - Reducing pDelayTurnaround will have no effect on this source of error.
- $meanLinkDelay_{error}$ due to $mNRR_{error}$ is proportional to pDelayTurnaroud
  - Reducing pDelayTurnaround can reduce this source of error.
- The actual Link Delay is not a significant source of error
  - Only needs to be included as part of $pDelay_{errorRR}$…which is small enough to ignore for the purposes of this analysis

Formulae – Top Level

$$DTE(x) = \sum_{n=1}^{x} (meanLinkDelay_{error}(n) + residenceTimeerror(n)) \quad \textbf{ns}$$

$$meanLinkDelay_{error} = (1 - \boldsymbol{meanLinkDelay_{errorCorrection}})(pDelay_{errorTS} + pDelay_{errorNRR} + pDelay_{errorRR}) \quad \textbf{ns}$$

$$residenceTime_{error} = residenceTimeerrorTS + residenceTimeerrorRR \quad \textbf{ns}$$

The current analysis does not include factors shown in grey.
x is number of hops.

Special Cases

$$\boldsymbol{clockDrift}(0) = \boldsymbol{clockDrift_{GM}}$$

For the first hop ($n = 1$), $n - 1 = 0$, i.e. the first device in chain, which is the GM.

$$\boldsymbol{residenceTime_{error}}(x) = 0$$

For the last hop ($n = x$), the Sync message is not passed on so there is no Residence Time Error.

Analysis Carried Out Using **r** & **RStudio**

- **r** is available here: https://www.r-project.org/
  - Open source license: various, but mostly GNU GPL v2 and GPL v3
- **RStudio** is available here: https://www.rstudio.com/products/rstudio/download/
  - Open source license: GNU Affero GPL v3
- Model uses write.csv and write.table functions, which are part of R.Utils package
  - Install in RStudio by typing…

    ```
    install.packages("R.utils")
    ```

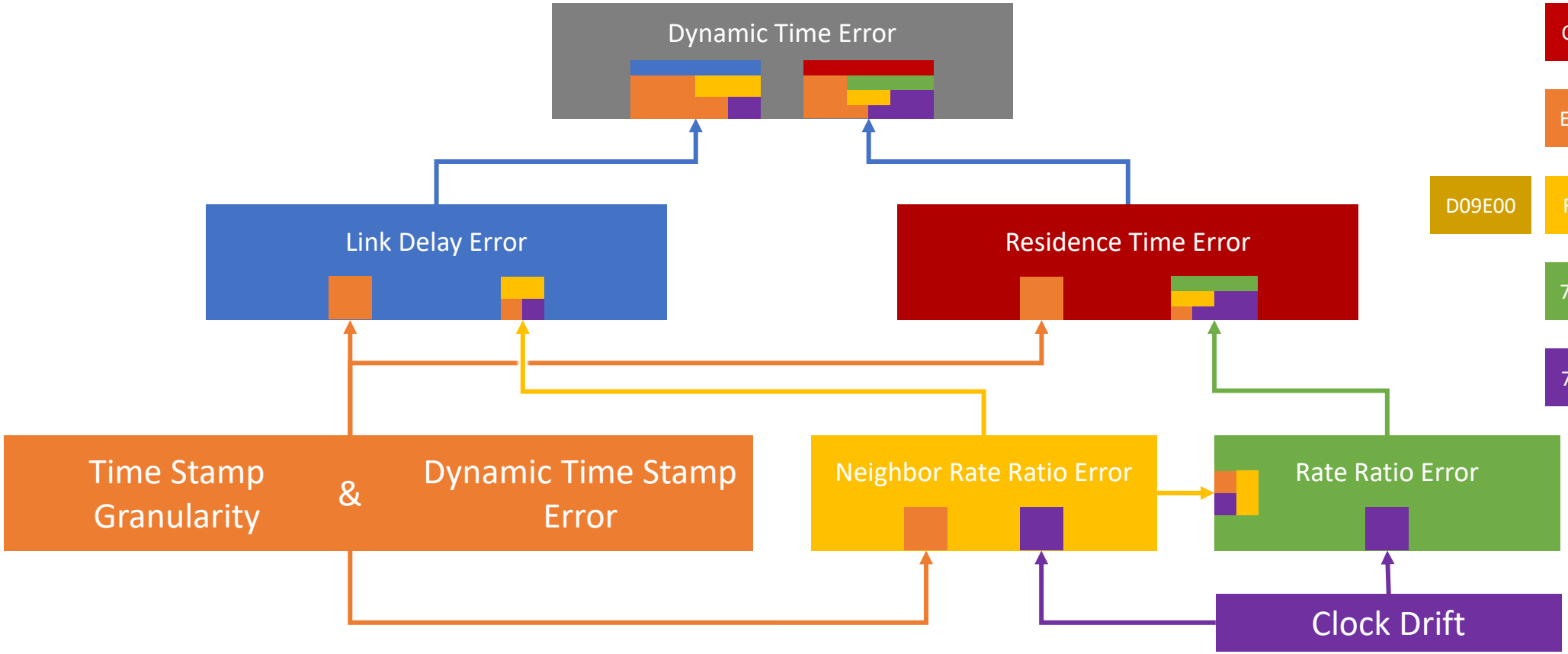
    …in the console window.

Reasons for Choosing r & RStudio

- Powerful tools for building the model then carrying out statistical analysis of the results.
  - Example: Q-Q plot to determine if dataset has a normal distribution with a single command.
- Simple scripting language.
- Easy to generate many charts to visualise the results
- Can also output results to a file for further analysis and graphing.
  - Used to generate visualisation of how errors accumulate.

Backup Material

Graphics Colour Palette



Lines	Areas	Backgrounds
7F7F7F	BFBFBF	F2F2F2
4472C4	B4C7E7	DAE3F3
C00000	FF8181	FFBDBD
ED7D31	F8CBAD	FBE5D6
D09E00	FFC000	FFE699	FFF2CC
70AD47	C5E0B4	E2F0D9
7030A0	C198E0	DFC9EF