60802 Time Synchronisation – NRR Tracking & Error Compensation: 1-hop Model; Piece-wise Linear Clock Drift

David McCall (Intel)

Version 1
Added more detail on proposed algorithm.
Fixed typos and other editorial errors.

References

1. David McCall, “60802 Time Sync Ad Hoc mNRRsmoothing Optimisation Results”, IEC/IEEE 60802 contribution, November 2022 https://www.ieee802.org/1/files/public/docs2022/60802-McCall-Time-Sync-mNRRsmoothingN-Optimisation-Results-1122-v2.pdf

Background

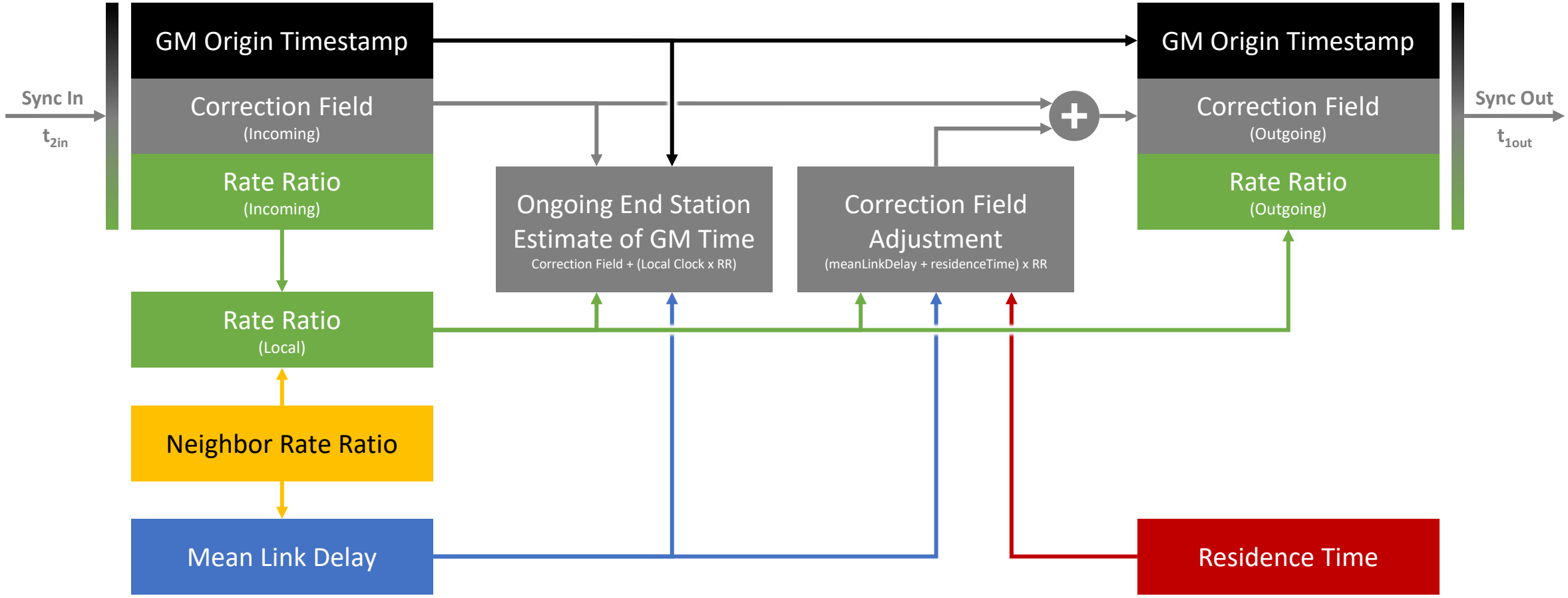
- IEC/IEEE 60802 has a stated requirement of 1us time accuracy over 64 hops (i.e. 65 devices) with a goal of 100 hops (i.e. 101 devices).
- One of the proposed techniques for achieving this goal is to track Clock Drift induced changes to Neighbor Rate Ratio and make adjustments to the NRR calculation to compensate for associated error.
- This presentation describes a potential algorithm for this technique, a Monte Carlo simulation that models it, the results of the simulation and suggested parameters for the algorithm. It also discusses some of the potential implications for the IEC/IEEE 60802 profile.
- As with the multi-hop simulation, preparing a contribution covering **all** the details (every equation, etc…) takes a lot of effort. The plan is to follow up with that at a later date.

Content

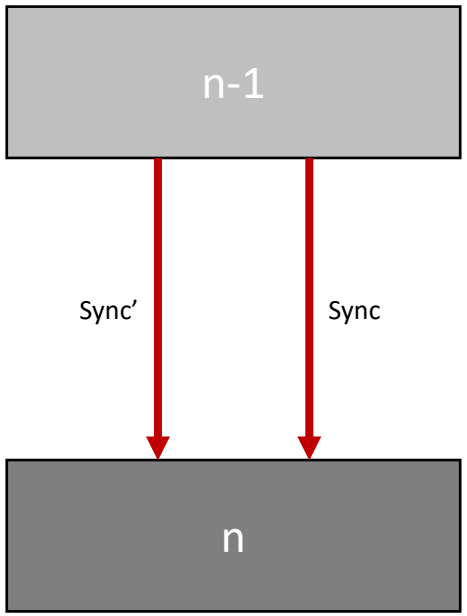
- Background
- Proposed Algorithm
- 1-hop Monte Carlo Simulation
- Headline Results from Simulation
- Additional Results
- Implications for the IEC/IEEE 60802 Profile

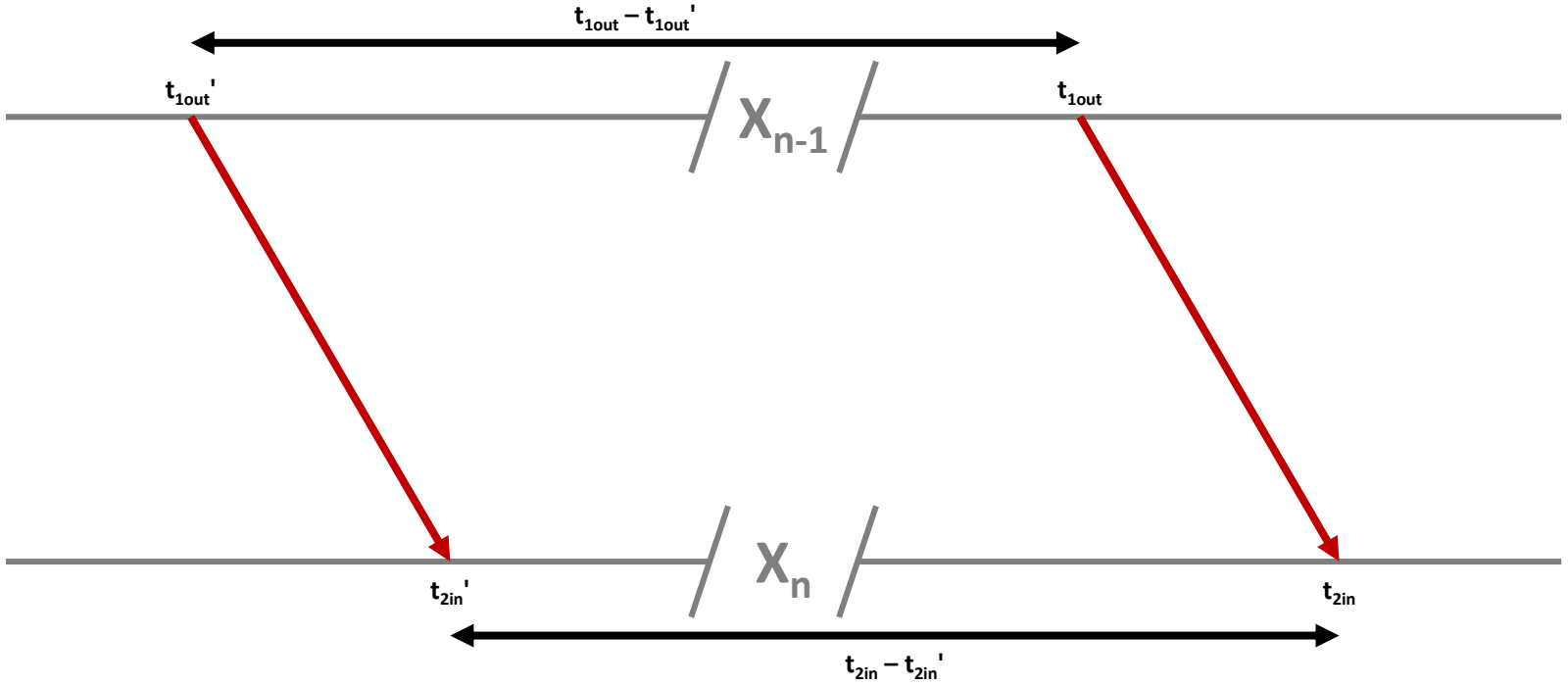
Background

Only Concerned with NRR



Using Sync to measure NRR…

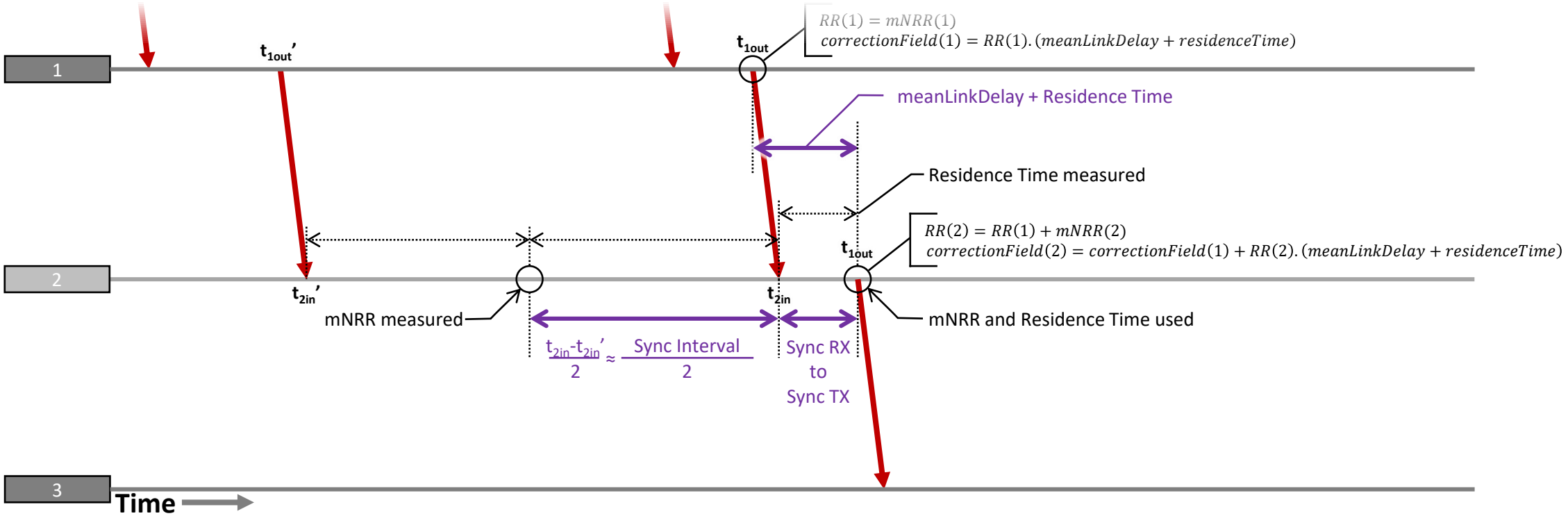




$$mNRR = \left(\frac{t_{1out} - t'_{1out}}{t_{2in} - t'_{2in}}\right)$$

ppm

…which improves timing & consistency.



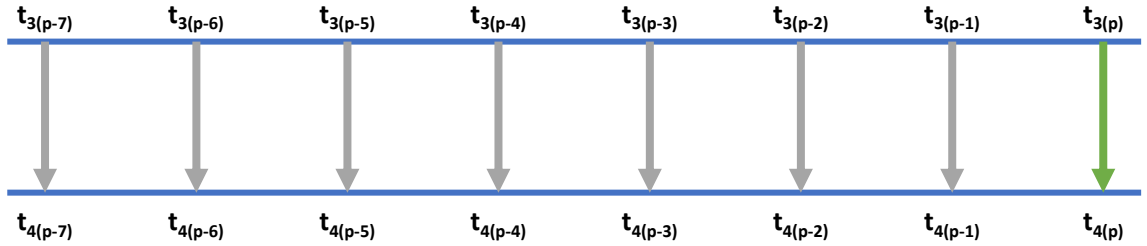
Sync

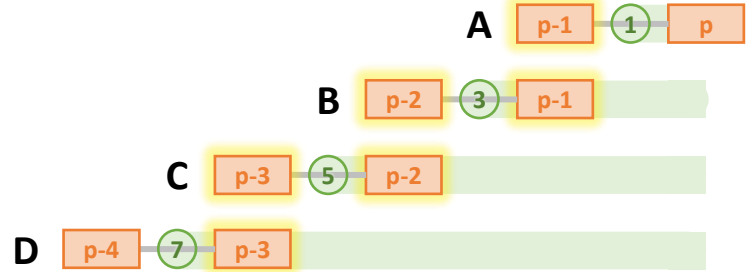
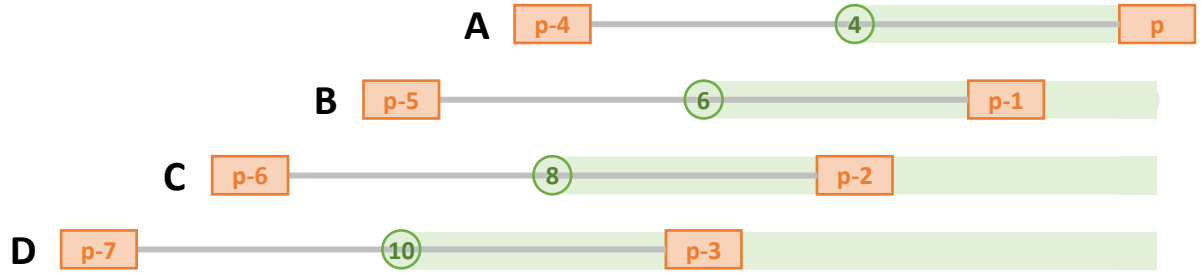
Clock Drift Errors

- Error due to drift during NRR measurement. (**Node 2** to **Node 1** - **NRR**)
- Error due to drift between measuring and using NRR. (**Node 2** to **Node 1** - **NRR**)
- Error due to drift during Residence Time measurement. (**Node 2** to **GM** - **RR**)
- Additional error from drift between RR(1) calculation, at Node 1, and use in calculating RR(2). (**Node 2** to **GM** - **RR at Previous Node**)
  - Note: In the model the contribution from meanLinkDelay is ignored; only Residence Time is used.

Method 3 is best for using older timestamps

See [1]



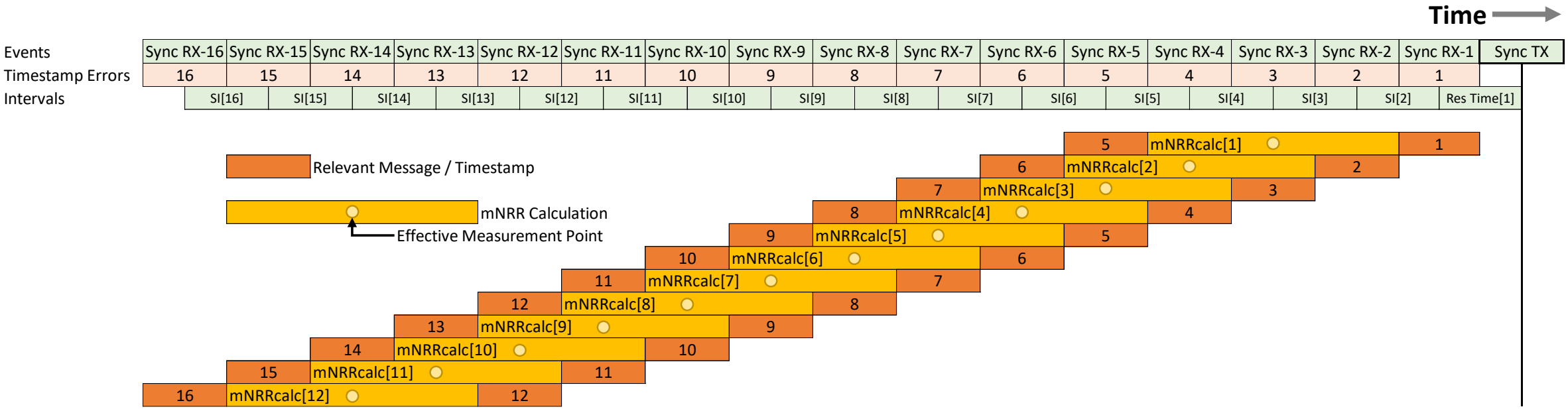
Method 1	mNRRsmoothingN = 1 mNRRsmoothingA = 1	
	mNRRsmoothingN = 4 mNRRsmoothingA = 1	
	mNRRsmoothingN = 7 mNRRsmoothingA = 1	
Method 2	mNRRsmoothingN = 1 mNRRsmoothingA = 4 Average of A, B, C & D	
Method 3	mNRRsmoothingN = 4 mNRRsmoothingA = 4 Average of A, B, C & D	

Proposed Algorithm

Approach Overview

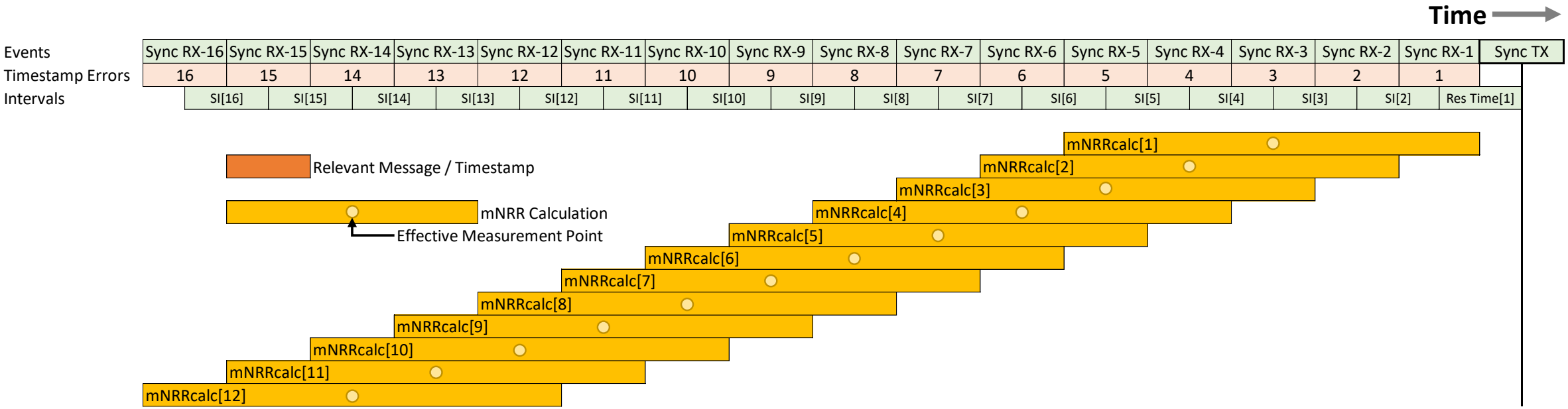
- Assume linear Clock Drift
  - Good enough for short periods of time, unless…
  - …there is a sudden change in drift (discontinuity) when the assumption of linearity & consequent “compensation” could make things worse, but…
  - …even that’s OK at a system level, provided this only happens at a low number of nodes at any one time (or a vast majority of nodes at exactly the same time).
- Take two measurements of NRR, separated in time
  - **Before** taking the measurement of NRR for use in Sync.
  - Take an average of **A** past calculations, each of which looks back **N** timestamps; second measurement starts **P** timestamps back in time.
  - May have different **N** and **A** values than final mNRR measurement as there may be different priorities for Clock Drift / Timestamp Error balance.
  - Keep track of effective measurement time: half way between timestamps for each calculation; average of mNRR calculations that are averaged
- Estimate NRR Drift Rate: difference between the two NRR measurements, divided by the interval between effective measurement points.
- Measure NRR for use in Sync
  - Same **N** & **A** process, but before taking average, adjust each initial NRR calculation to compensate for estimated drift (Drift Rate x Interval between effective measurement point & Sync TX)

NRR Algorithm – 1



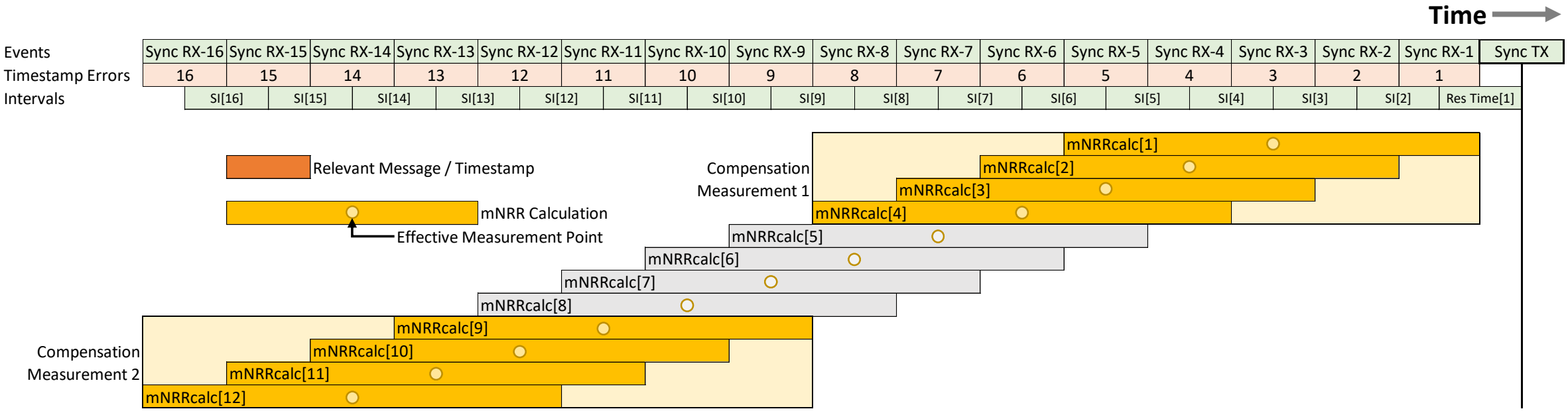
- After arrival of each Sync message, do the initial calculation of mNRR looking **N** timestamps back. In this example **N = 4**
  - Also calculate effective measurement point; half way between relevant timestamps.
- Keep larger of (Compensation: **N**+**A**+**P**) or (Final Measure: **N**+**A**) past calculations

NRR Algorithm – 2



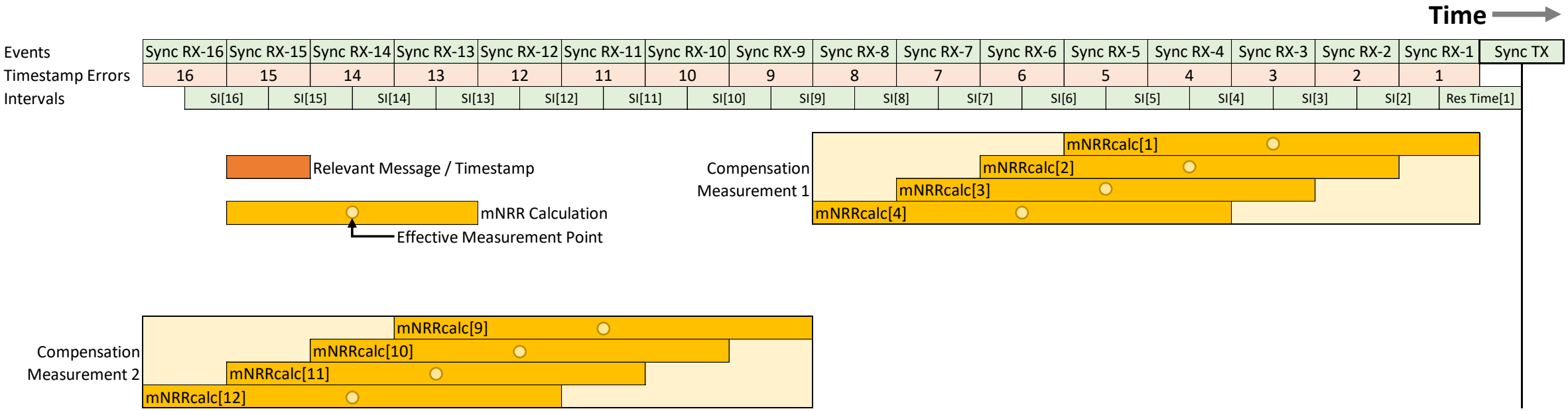
- The timestamps themselves are no longer needed after the initial calculation.
- Note that if final mNRR measurement (for use in Sync) has different **N** value, separate calculations will be required.

NRR Algorithm – 3



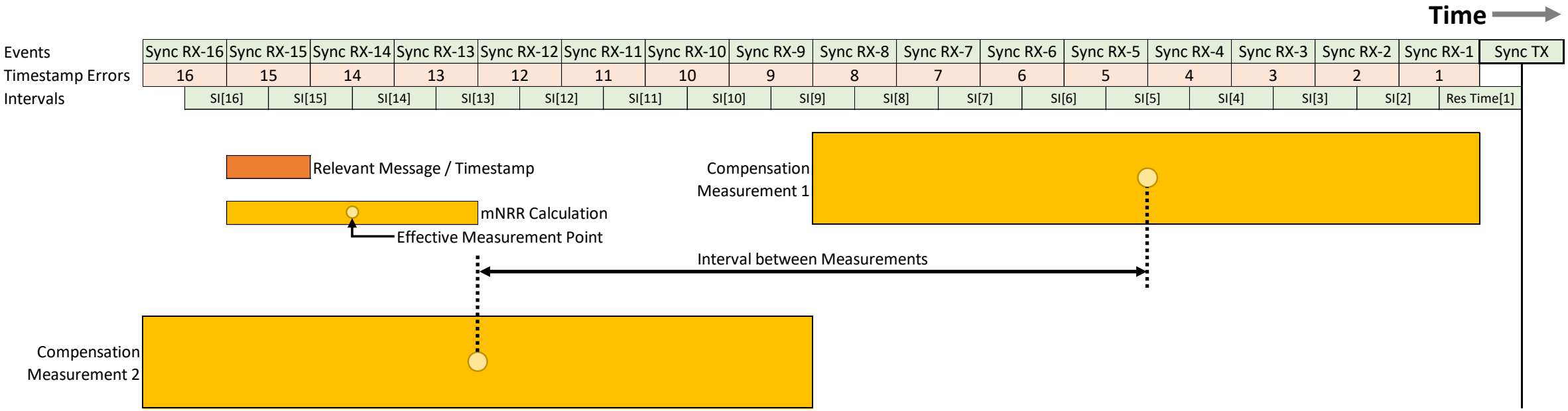
- Each measurement uses **A** initial calculations. In this example **A = 4**.
- The second measurement start P calculations in the past. In this example **P = 8**.
- Typically, to optimise timestamp error rejection, **N = A** and Measurement 1 & 2 will not overlap, so **P = N + A**, which means that some initial NRR calculations will not be used.
  - Although they will be used in future measurements, when the “calculation window” progresses

NRR Algorithm – 4



- For each measurement, NRR value is the average of the A initial calculations.
- The effective measurement point of the average is the average of the initial calculations' measurement points.

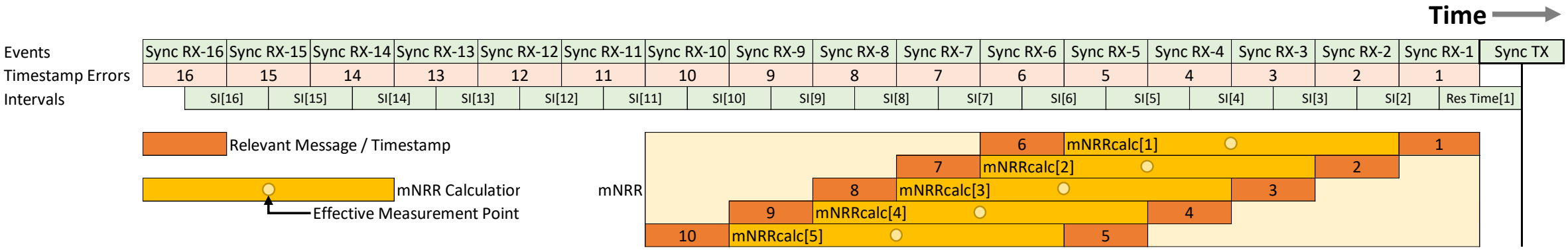
NRR Algorithm – 5



- Estimate the NRR Drift Rate:

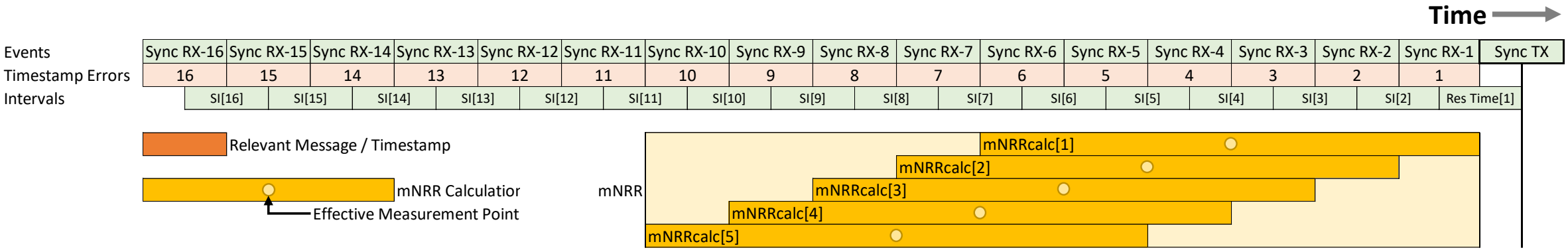
$$NRR\ Drift\ Rate = \frac{Measurement\ 1\ - Measurement\ 2}{Interval\ between\ Measurements}$$ **ppm/s**

NRR Algorithm – 6



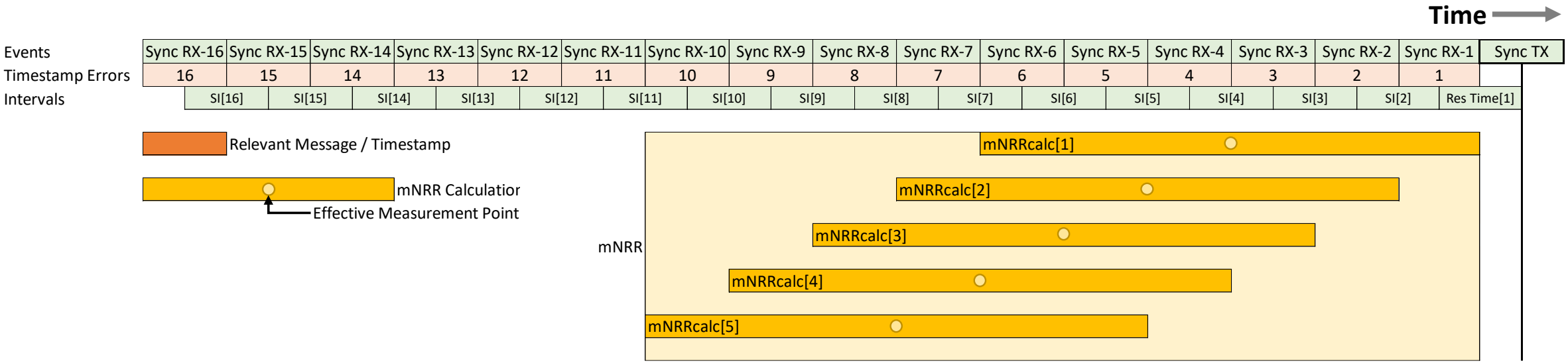
- Start the final mNRR measurement with A initial calculations, each looking N timestamps back.
- Note that N & A may be different from the N & A values for the NRR measurements used for NRR Drift Rate estimation.
- In this example **N = 5** and **A = 5**

NRR Algorithm – 7



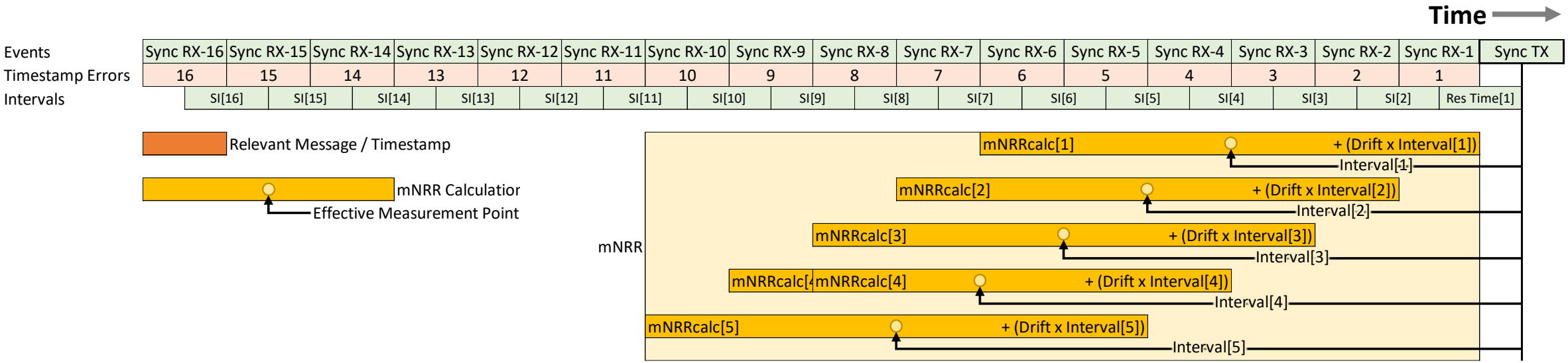
- As earlier, the actual timestamps are no longer needed.

NRR Algorithm – 8



- Apply a correction to the initial calculations, based on the NRR drift rate estimate.

NRR Algorithm – 8

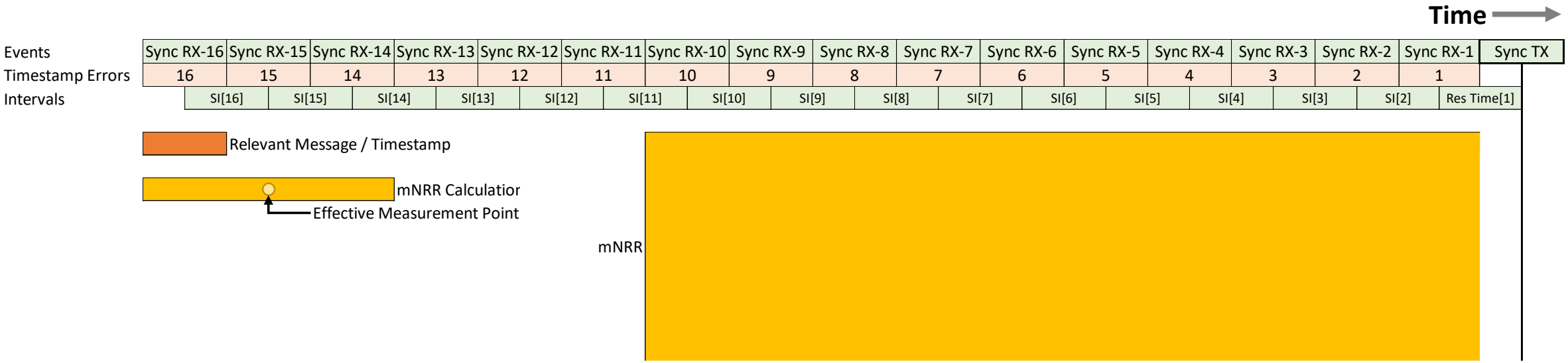


- Apply a correction to the initial calculations, based on the NRR drift rate estimate.
- For each calculation (NRR in ppm), add the correction factor:

$$Correction\ Factor = NRR\ Clock\ Drift\ Estimate \times Interval\ between\ Effective\ Measurment\ Point\ and\ Sync\ TX \quad \textbf{ppm}$$

- Note: same NRR Clock Drift Estimate but different Interval for each calculation

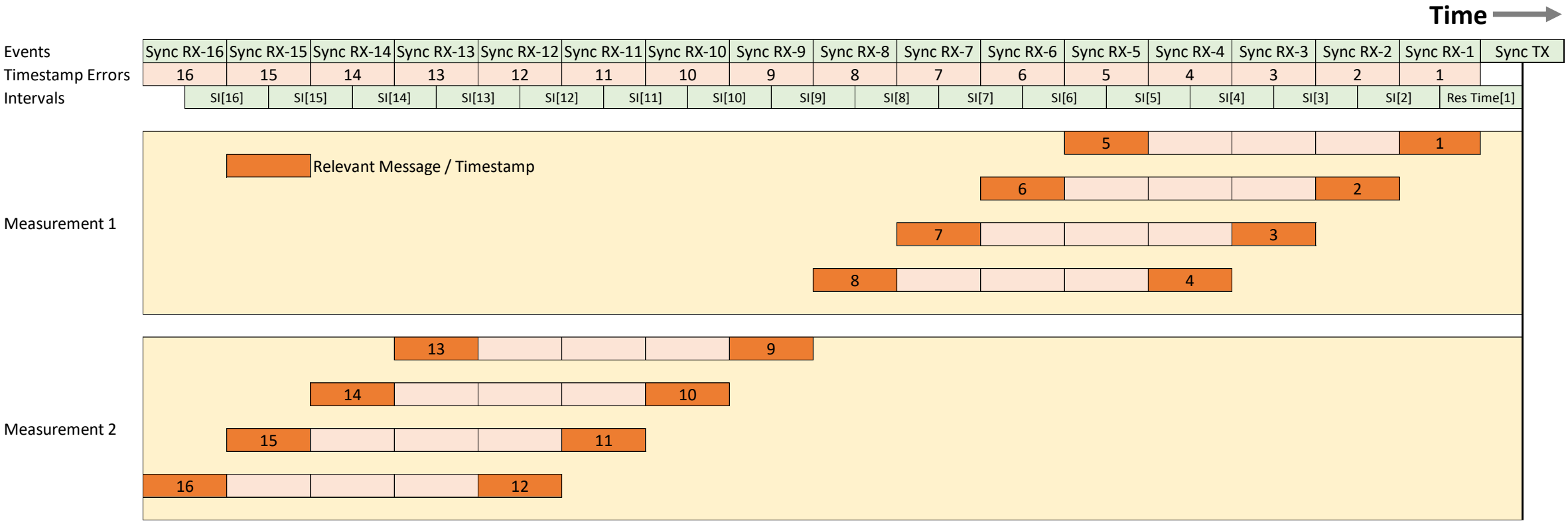
NRR Algorithm – 9



- Take an average of the **A** initial calculations to get the mNRR value to be used during Sync processing (added to incoming RR to generate local RR).

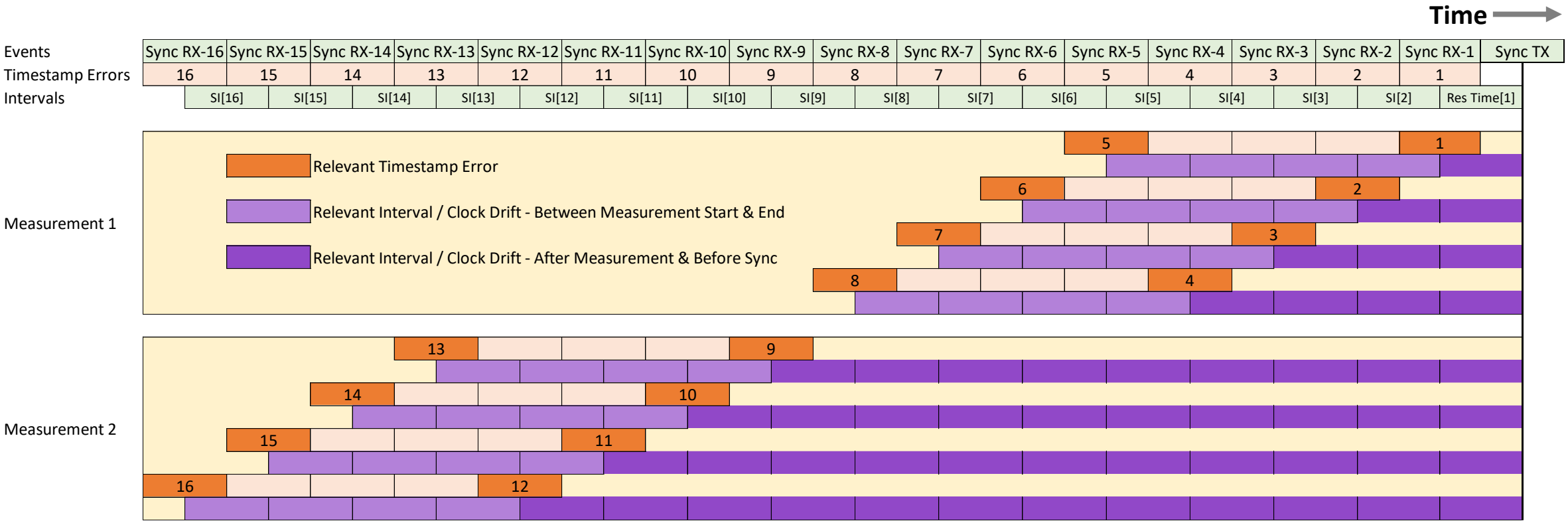
1-hop Monte Carlo Simulation

mNRR Comp Measurements – Example



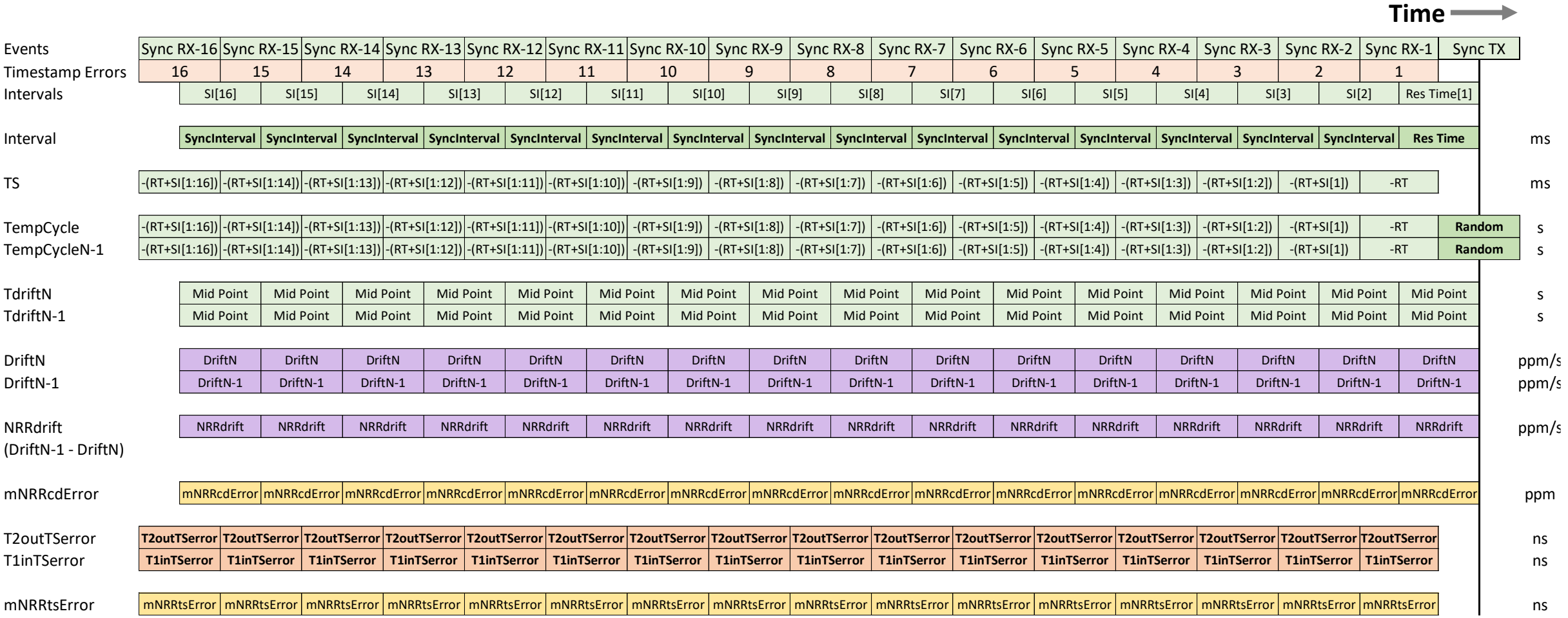
- N = 4, each initial calculation goes back 4 timestamps
- A = 4, each measurement is an average of previous 4 initial calculations
- P = 8, second measurement starts 8 timestamps further in the past than the first measurements

mNRR Comp Measurements – Errors



- N = 4, each initial calculation goes back 4 timestamps
- A = 4, each measurement is an average of previous 4 initial calculations
- P = 8, second measurement starts 8 timestamps further in the past than the first measurements

1-hop Simulation - Overview



mNRR Error Algorithms

mNRR Primary Errors

$$mNRR_{errorTS_X} = \frac{(t_{1outerror} - t'_{1inerror}) - (t_{2inerror} - t'_{2inerror})}{T_{interval}} = \frac{(t_{1out} - t_{2in}) - (t'_{1outerror} - t'_{2inerror})}{T_{interval}}$$ **ppm**

$$mNRR_{errorCD_X} = \frac{T_{interval}(clockDrift_n - clockDrift_{n-1})}{2 \times 10^3} = \frac{T_{interval}(-NRR_{drift})}{2 \times 10^3}$$ **ppm**

$$mNRR_{error_X} = mNRR_{errorTS_X} + mNRR_{errorCD_X}$$ **ppm**

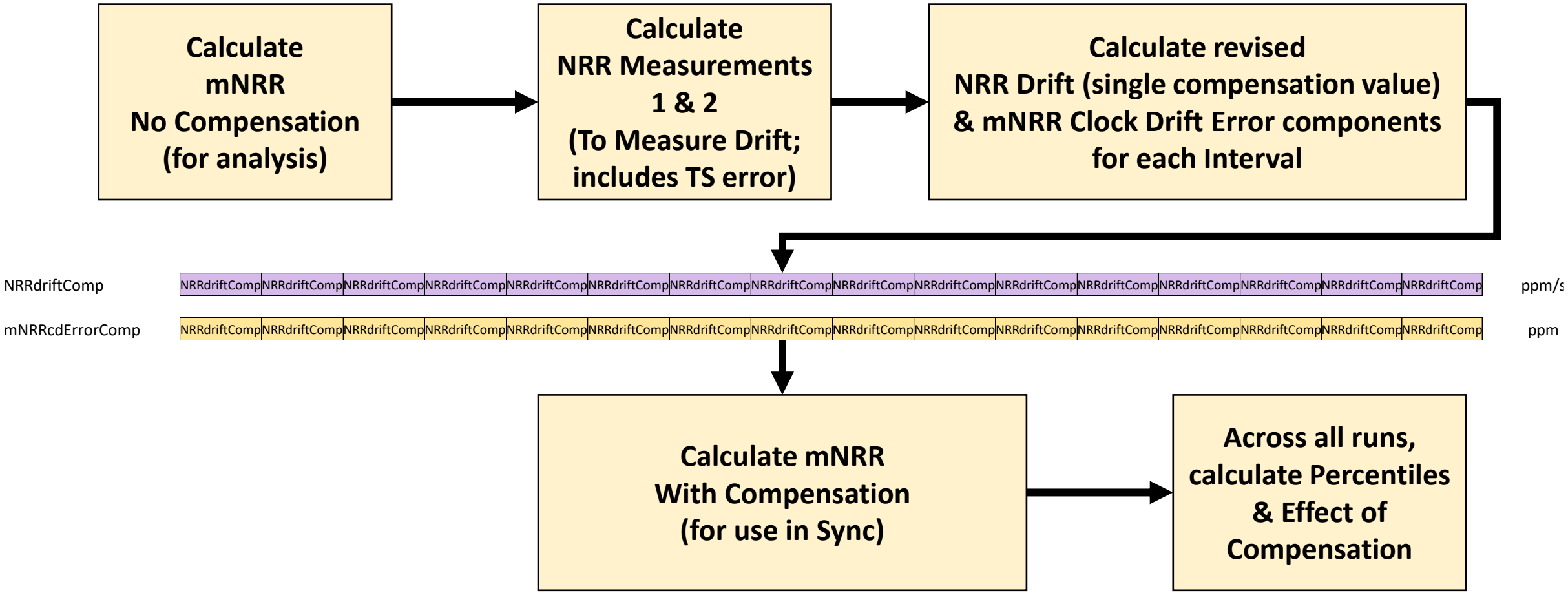
RR Primary Errors

$$RR_{errrorCD_NRR2Sync_X} = \frac{T_{interval}(clockDrift_n - clockDrift_{n-1})}{10^3} = \frac{T_{interval}(-NRR_{drift})}{10^3}$$ **ppm**

$$RR_{errrorCD_RR2Sync_X} = \frac{residenceTime(clockDrift_{n-1} - clockDrift_{GM})}{10^3}$$ **ppm**

$$RR_{errror_SUM}(n) = RR_{error_SUM}(n-1) + mNRR_{error_X} + RR_{errorCD_NRRtoSync_X} + RR_{errorCD_RRtoSync_X}$$ **ppm**

1-hop Simulation Overview

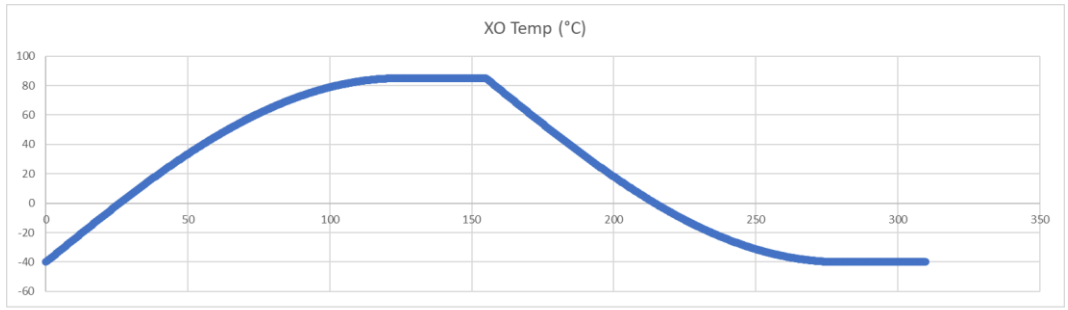


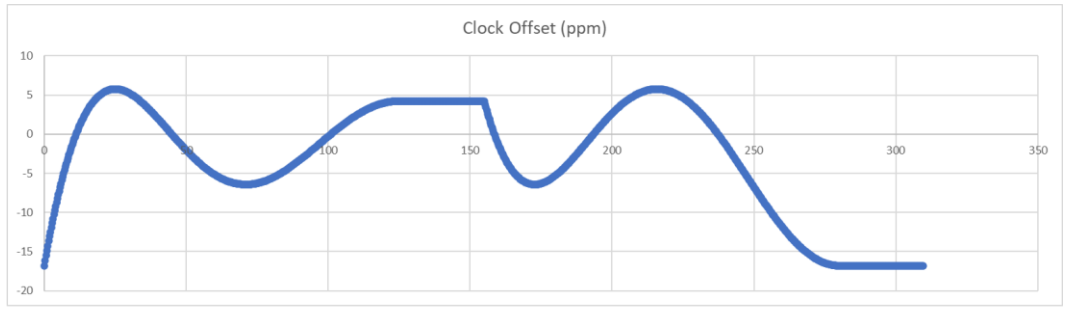
Headline Results

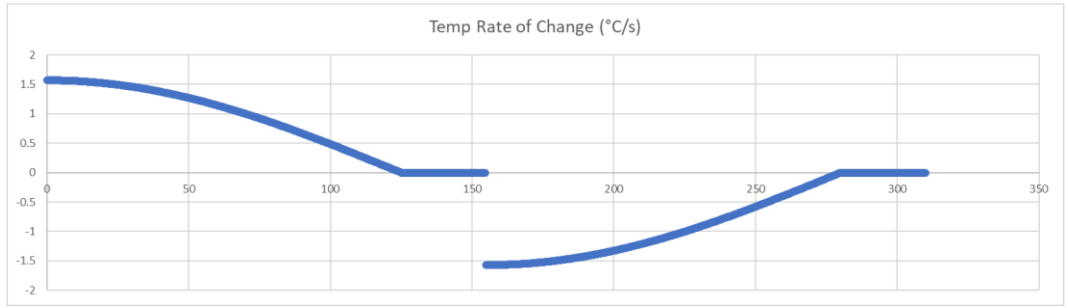
Configuration

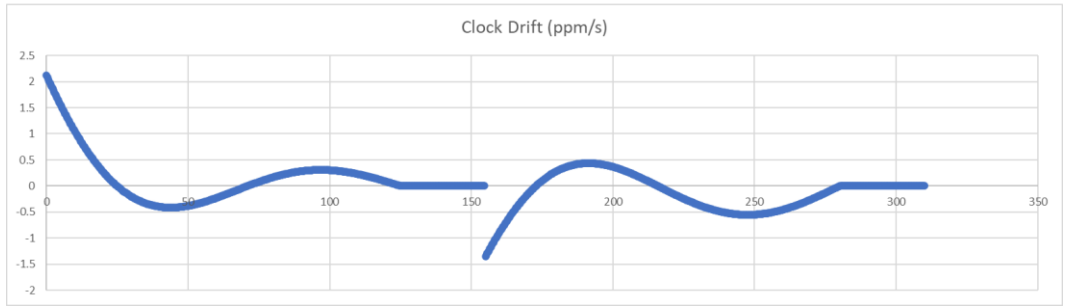
- XO Temp Dift: ¼-Sinusoidal, 125s ramp; 30s hold
- 8ns Timestamp Granularity; ±4ns Dynamic Timestamp Error
- Sync Interval; parameter (nominal value; not part of simulation) 125ms; uniform distribution 120-130ms
- Residence Time: parameter (nominal max; not part of simulation) 10ms; mean 5ms; standard deviation 1.8ms; max (used in simulation) 15ms
- mNRRsmoothingN: 4
- mNRRsmoothingA: 4

Clock Drift – ¼-Sinusoidal Temperature Ramp: 125s ↕







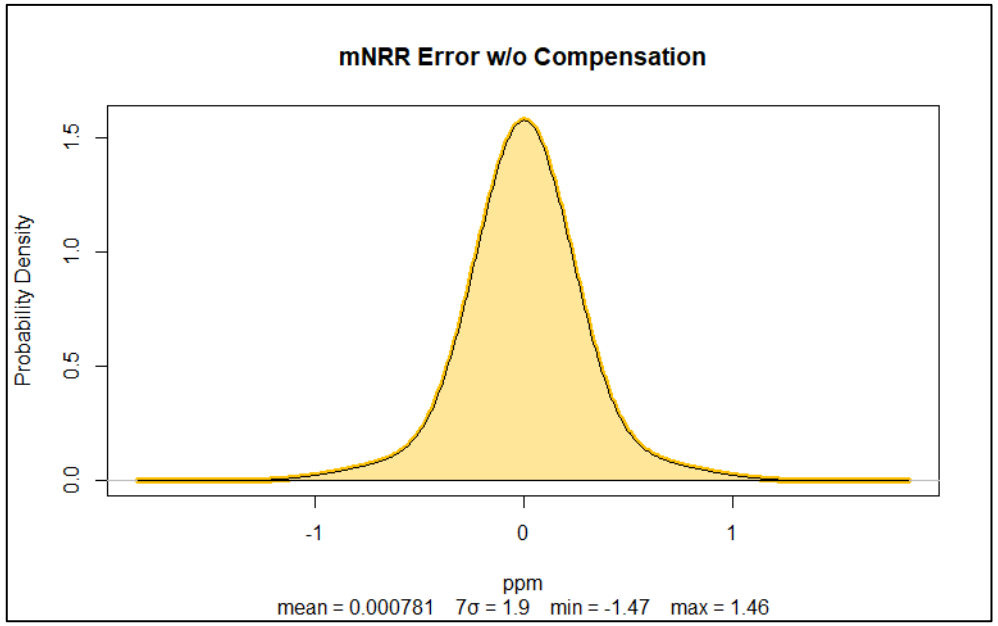


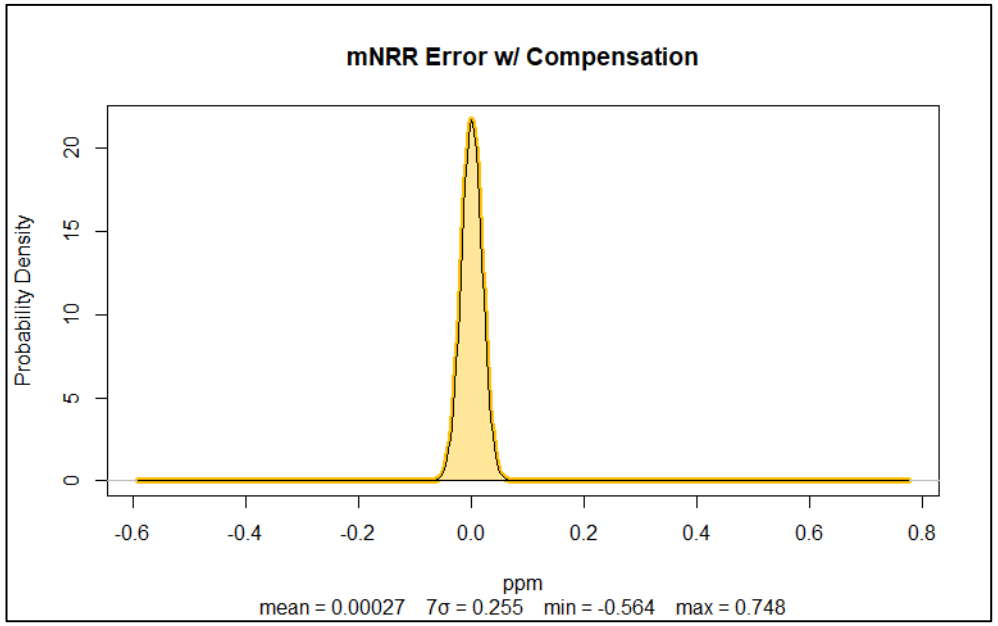
Inputs		
Temp Max	85	°C
Temp Min	-40	°C
Temp Ramp Period	125	s
Temp Hold	30	s

Temp Rate of Change		
MAX	1.57	°C/s
MIN	-1.57	°C/s

Clock Drift		
MAX	2.12	ppm/s
MIN	-1.35	ppm/s

Best Result (so far…)





mNRRcompN = 3
mNRRcompA = 3
mNRRcompP = 6

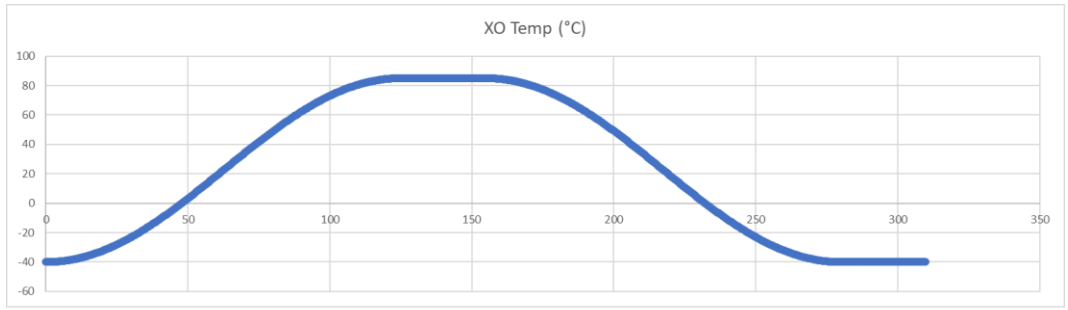
mNRR (for Sync)		mNRR for Compensation			Spreads					
N	A	N	A	P	7σ	Min-Max	99%	98%	95%	90%
4	4				3.81	2.94	1.8	1.8	1.21	0.836
4	4	3	3	6	0.51	1.31	0.311	0.0876	0.0666	0.0541
				Percentage Change	-87%	-55%	-83%	-95%	-94%	-94%

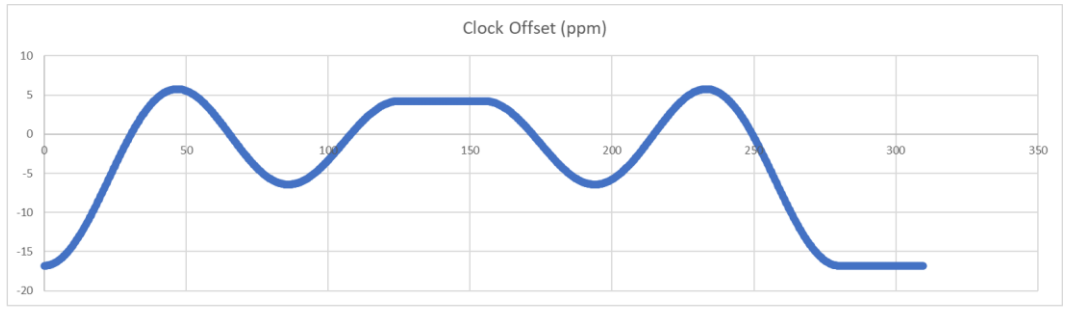
Additional Results

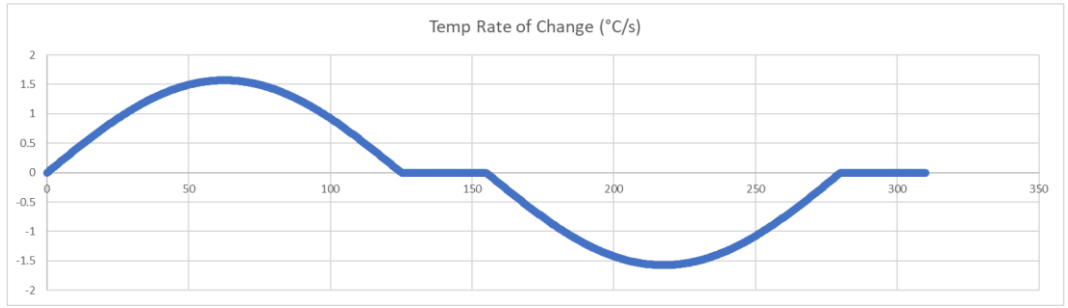
Varying N/A/P for Compensation Calculation

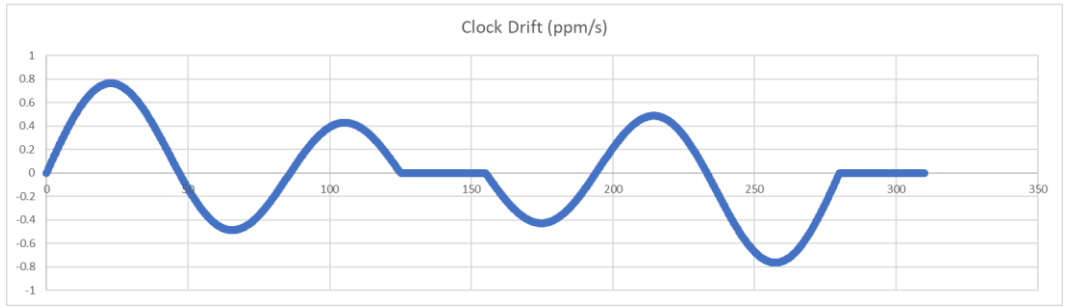
mNRR (for Sync)		mNRR for Compensation			Spreads						Percentage Improvement					
N	A	N	A	P	7σ	Min-Max	99%	98%	95%	90%	7σ	Min-Max	99%	98%	95%	90%
4	4				3.81	2.94	1.8	1.8	1.21	0.836						
4	4	1	1	2	1.85	1.48	0.677	0.612	0.516	0.433	-51%	-50%	-62%	-66%	-57%	-48%
4	4	2	2	4	0.393	0.763	0.14	0.12	0.0979	0.0814	-90%	-74%	-92%	-93%	-92%	-90%
4	4	3	3	6	0.51	1.31	0.311	0.0876	0.0666	0.0541	-87%	-55%	-83%	-95%	-94%	-94%
4	4	4	4	8	0.705	1.91	0.554	0.111	0.075	0.0588	-81%	-35%	-69%	-94%	-94%	-93%
4	4	5	5	10	0.876	2.2	0.762	0.227	0.096	0.075	-77%	-25%	-58%	-87%	-92%	-91%
4	4	6	6	12	1.03	2.36	0.866	0.423	0.121	0.0945	-73%	-20%	-52%	-77%	-90%	-89%
4	4	7	7	14	1.16	2.57	0.95	0.554	0.147	0.115	-70%	-13%	-47%	-69%	-88%	-86%
4	4	8	8	16	1.28	2.62	1.02	0.7	0.175	0.135	-66%	-11%	-43%	-61%	-86%	-84%
4	4	9	9	18	1.39	2.69	1.08	0.784	0.205	0.156	-64%	-9%	-40%	-56%	-83%	-81%
4	4	10	10	20	1.48	2.75	1.15	0.848	0.236	0.177	-61%	-6%	-36%	-53%	-80%	-79%

Clock Drift – ½-Sinusoidal Temperature Ramp: 125s ↕









Inputs		
Temp Max	85	°C
Temp Min	-40	°C
Temp Ramp Period	125	s
Temp Hold	30	s

Temp Rate of Change		
MAX	1.57	°C/s
MIN	-1.57	°C/s

Clock Drift		
MAX	0.76	ppm/s
MIN	-0.76	ppm/s

¼-Sinusoidal vs ½-Sinusoidal

mNRR (for Sync)		mNRR for Compensation				Spreads					
N	A	N	A	P	Ramp	7σ	Min-Max	99%	98%	95%	90%
4	4				½ SIN	3.27	1.39	1.12	1.12	0.919	0.764
4	4	3	3	6	½ SIN	0.208	0.135	0.076	0.0687	0.0581	0.0489
				Percentage Change		-94%	-90%	-93%	-94%	-94%	-94%

mNRR (for Sync)		mNRR for Compensation				Spreads					
N	A	N	A	P	Ramp	7σ	Min-Max	99%	98%	95%	90%
4	4	3	3	6	¼ SIN	0.51	1.31	0.311	0.0876	0.0666	0.0541
4	4	3	3	6	½ SIN	0.208	0.135	0.076	0.0687	0.0581	0.0489
				Percentage Change		-59%	-90%	-76%	-22%	-13%	-10%

- ½-Sinusoidal **slightly** reduces error before compensation, but…
- Is **much** easier to track than ¼-Sinusoidal (no discontinuities to throw off the algorithm)

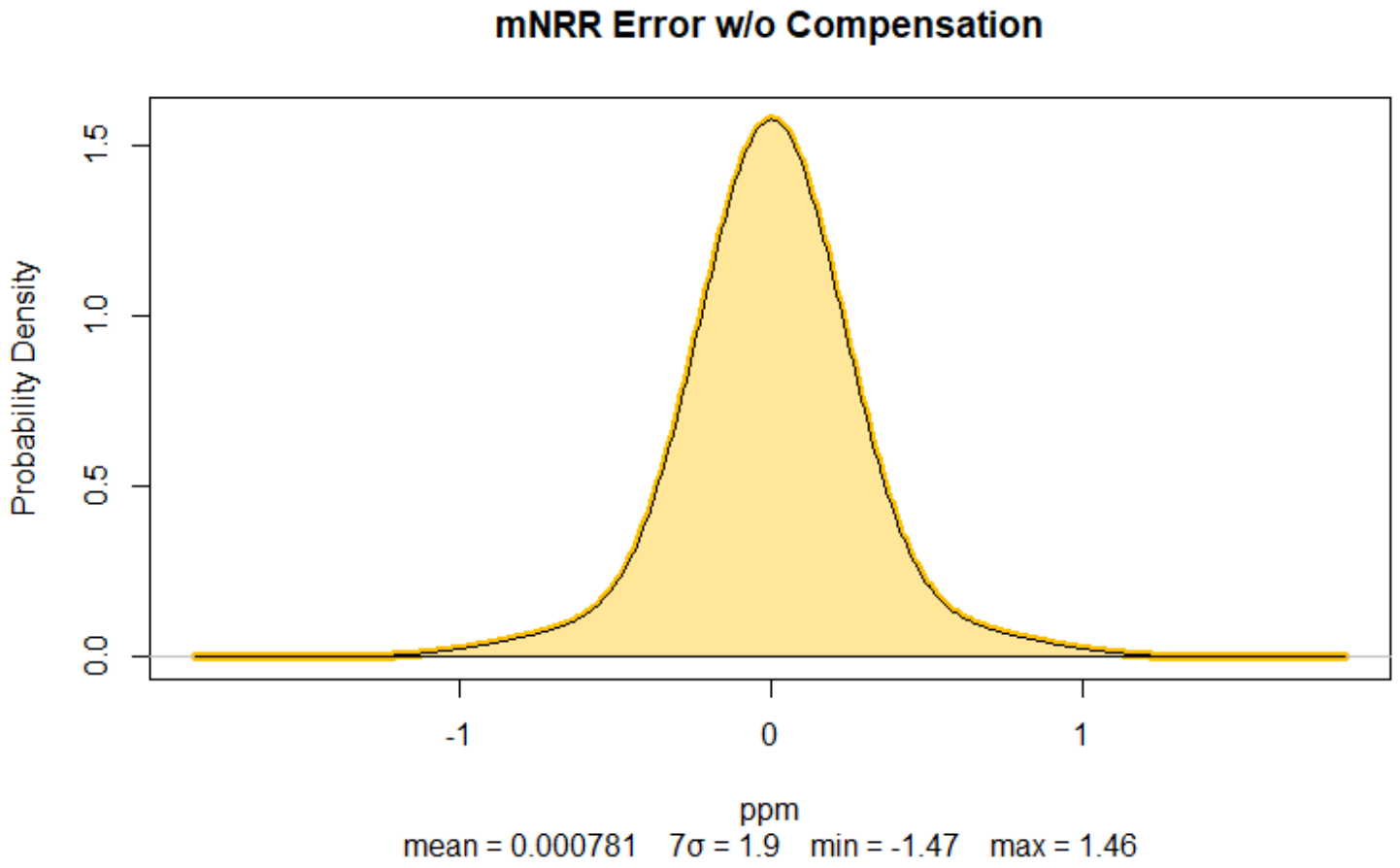
Implications

Implications for 60802

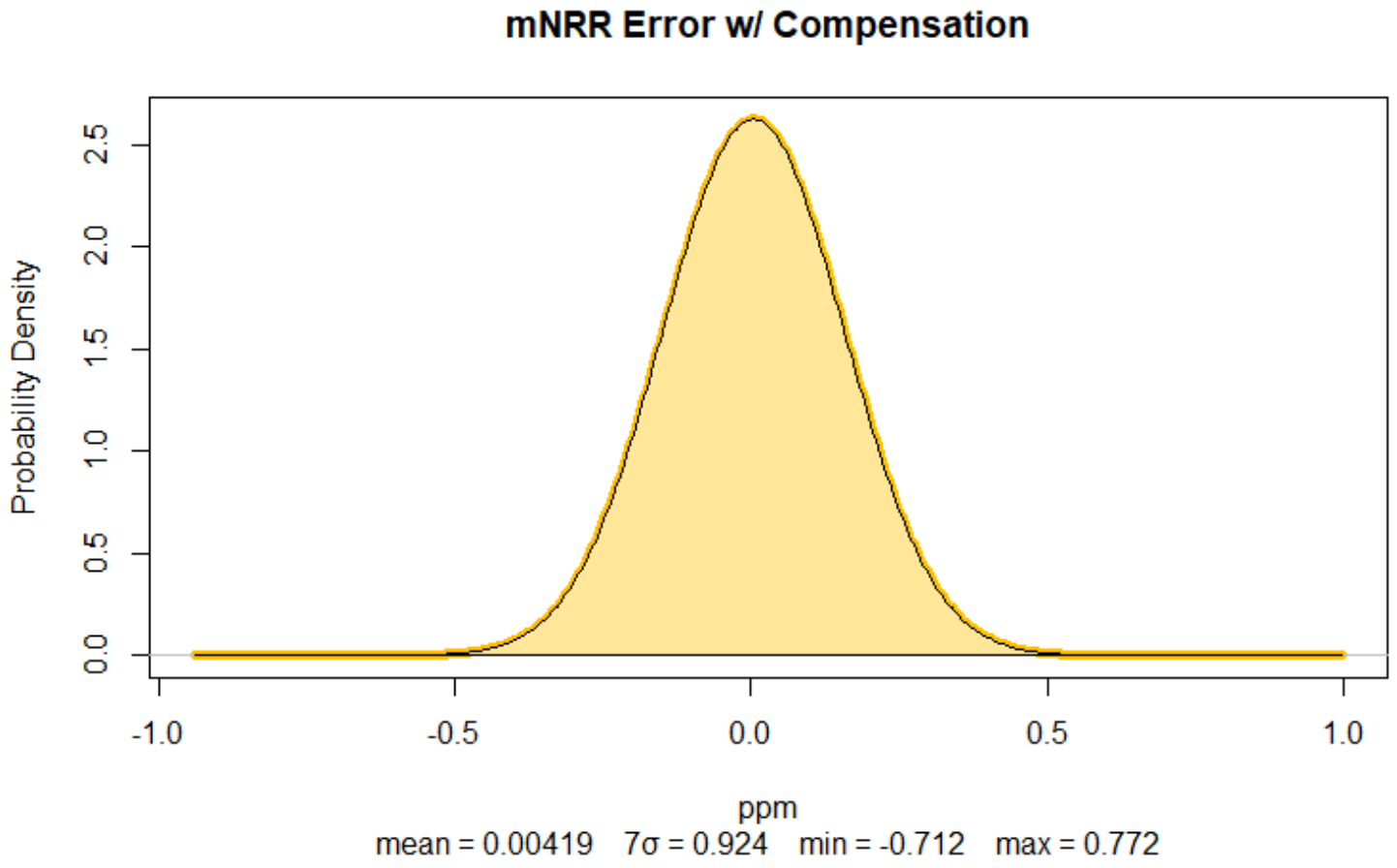
- Algorithm (mostly) works well (according to this simulation)
- As expected, it doesn't work as well when there are sudden changes in NRR Drift
  - Effect can be limited via careful choice of N/A/P parameters
  - More complex algorithms (e.g. Kalman Filter) would help, but more complex to implement (and compensation needs to be finalised between Sync TX & Follow-Up TX…although tracking can be done after Sync RX)
- May need to specify not just steady-state performance but response time to change? ("Best" parameters imply 1.5s response time.)
- Not entirely clear how this will play out at a system level
  - Simple assumption of % effective may not be accurate…but is it too kind, or too harsh? 😗
  - Could potentially analyse NRR drift vs % effective to keep 100-hop simulation executing quickly…or just suck it up and let simulator run for a few hours (or longer?)
  - Or…use % effective, unless one of the nodes' position on the temperature cycle is just after a discontinuity. Carry out a more detailed analysis of the 1-hop results around the discontiunities to figure out what to do in that case?

Backup

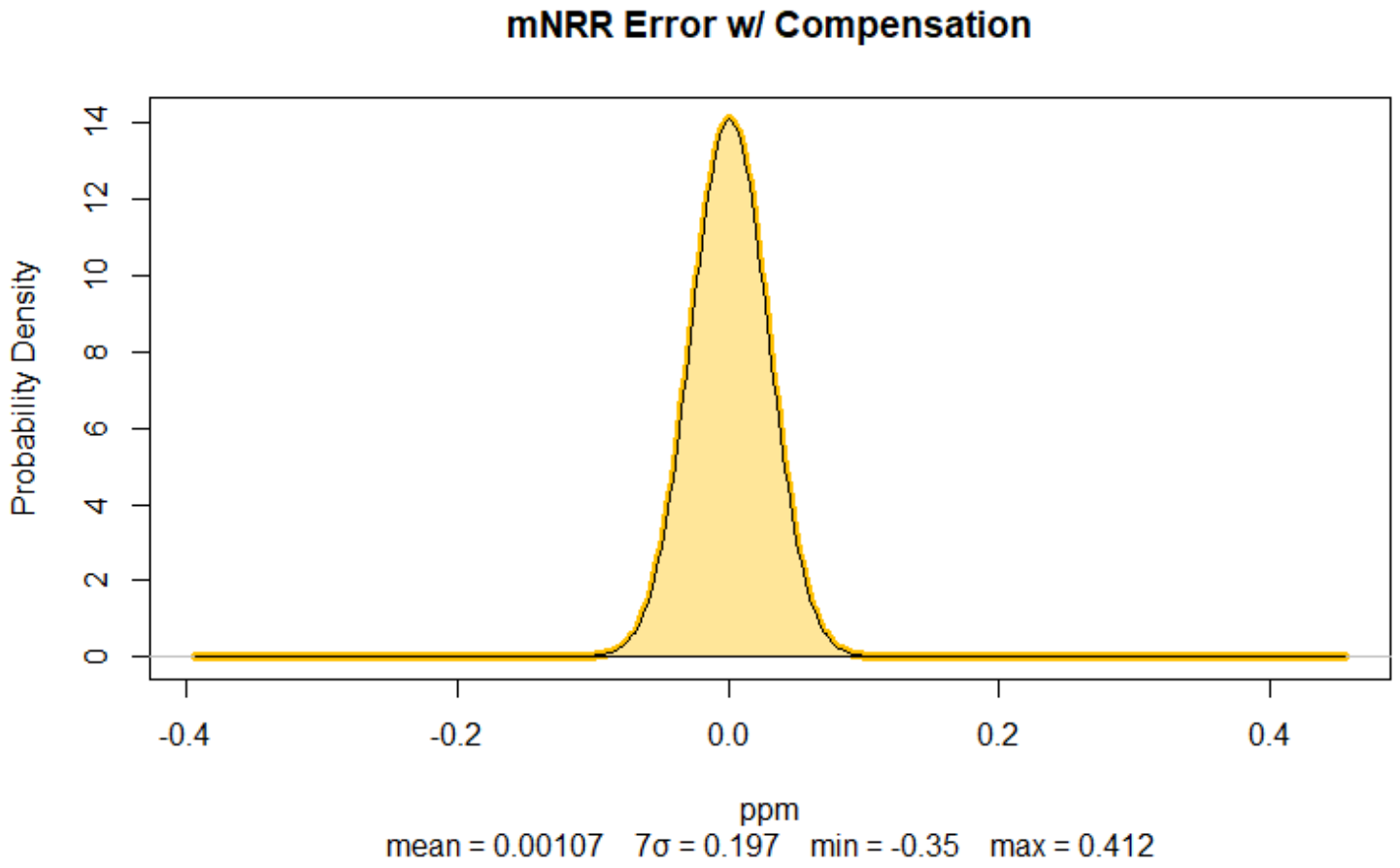
Base mNRR w/o Comp



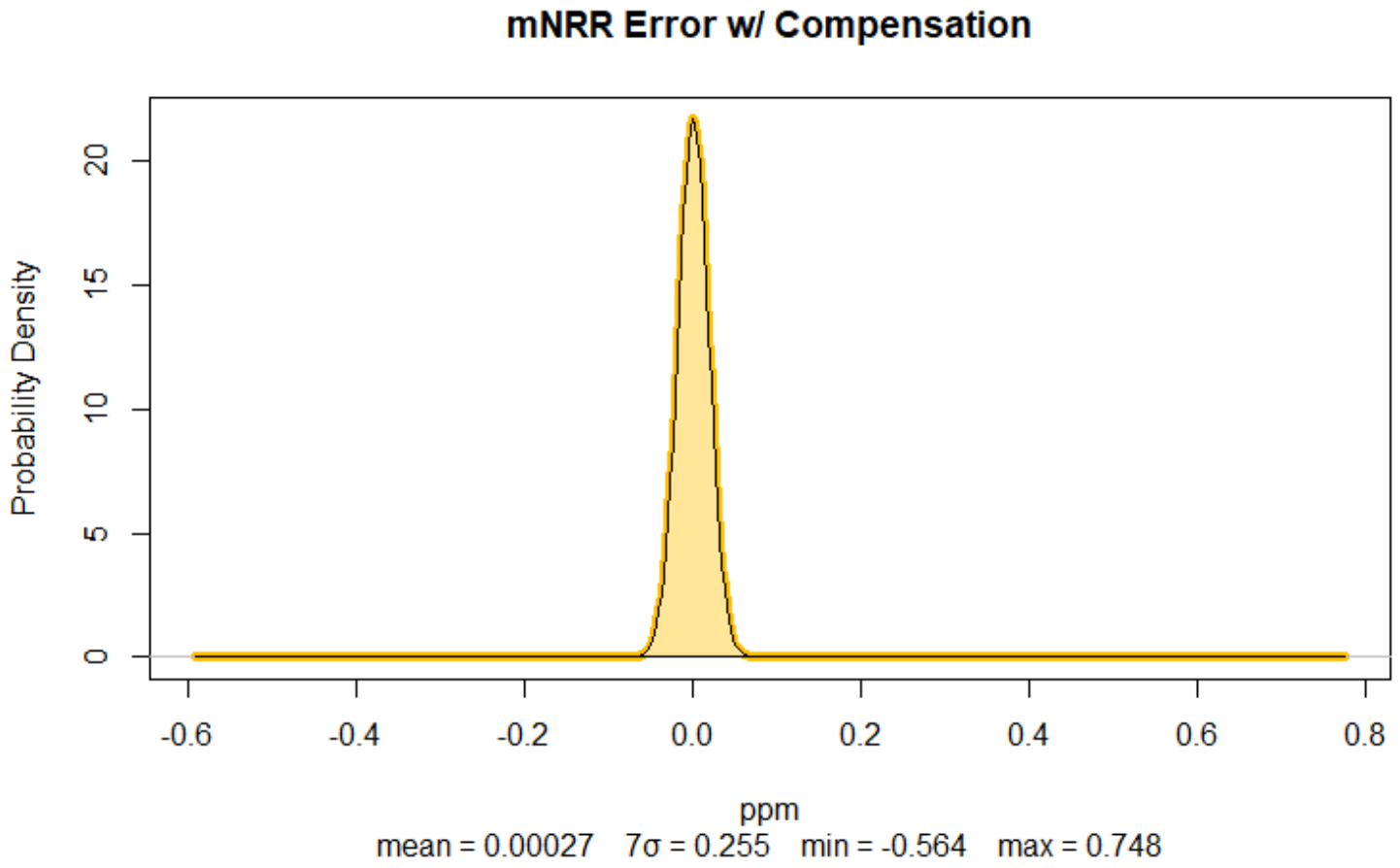
mNRR w/ Comp – N:1 A:1 P:2



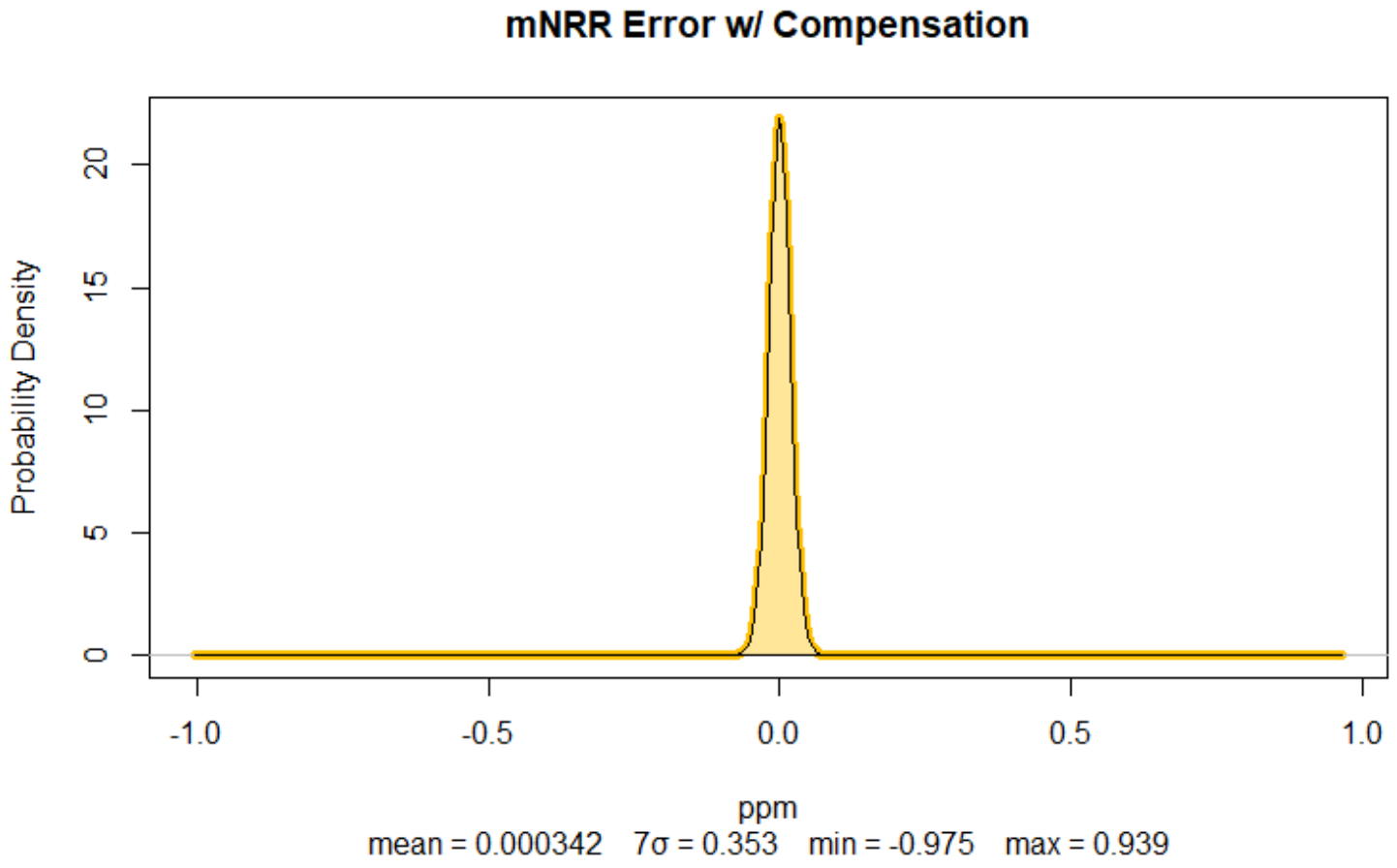
mNRR w/ Comp – N:2 A:2 P:4



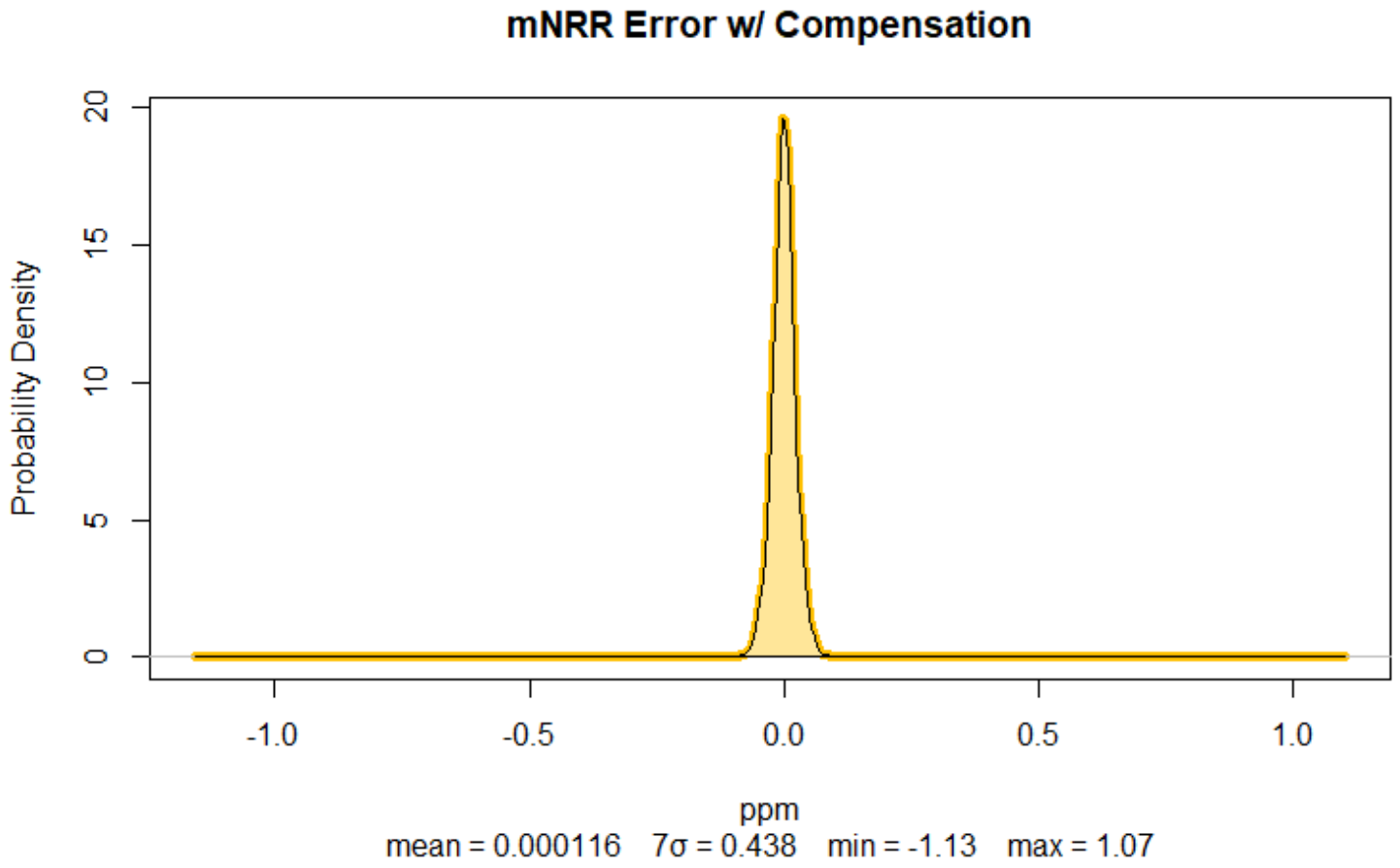
mNRR w/ Comp – N:3 A:3 P:6



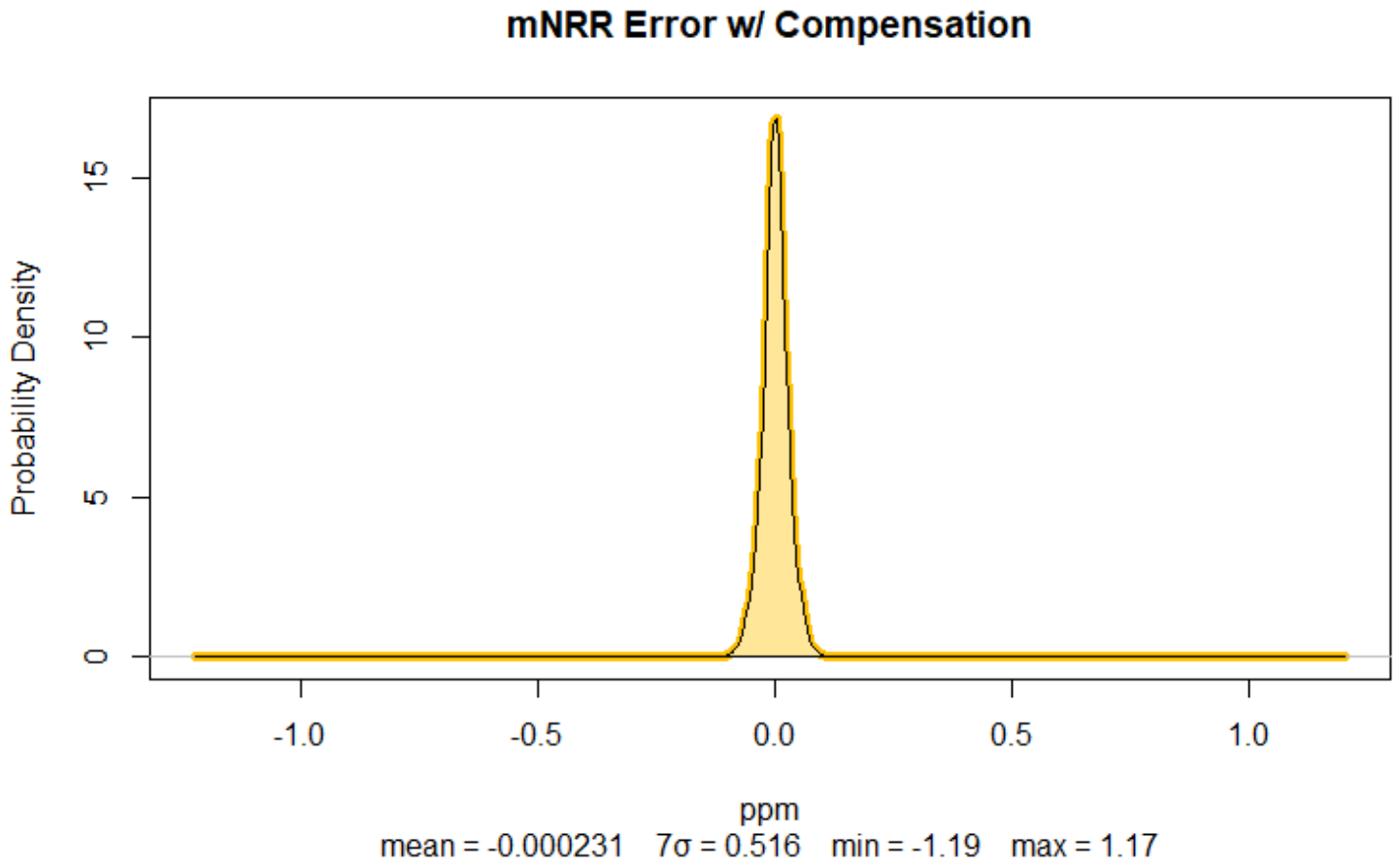
mNRR w/ Comp – N:4 A:4 P:8



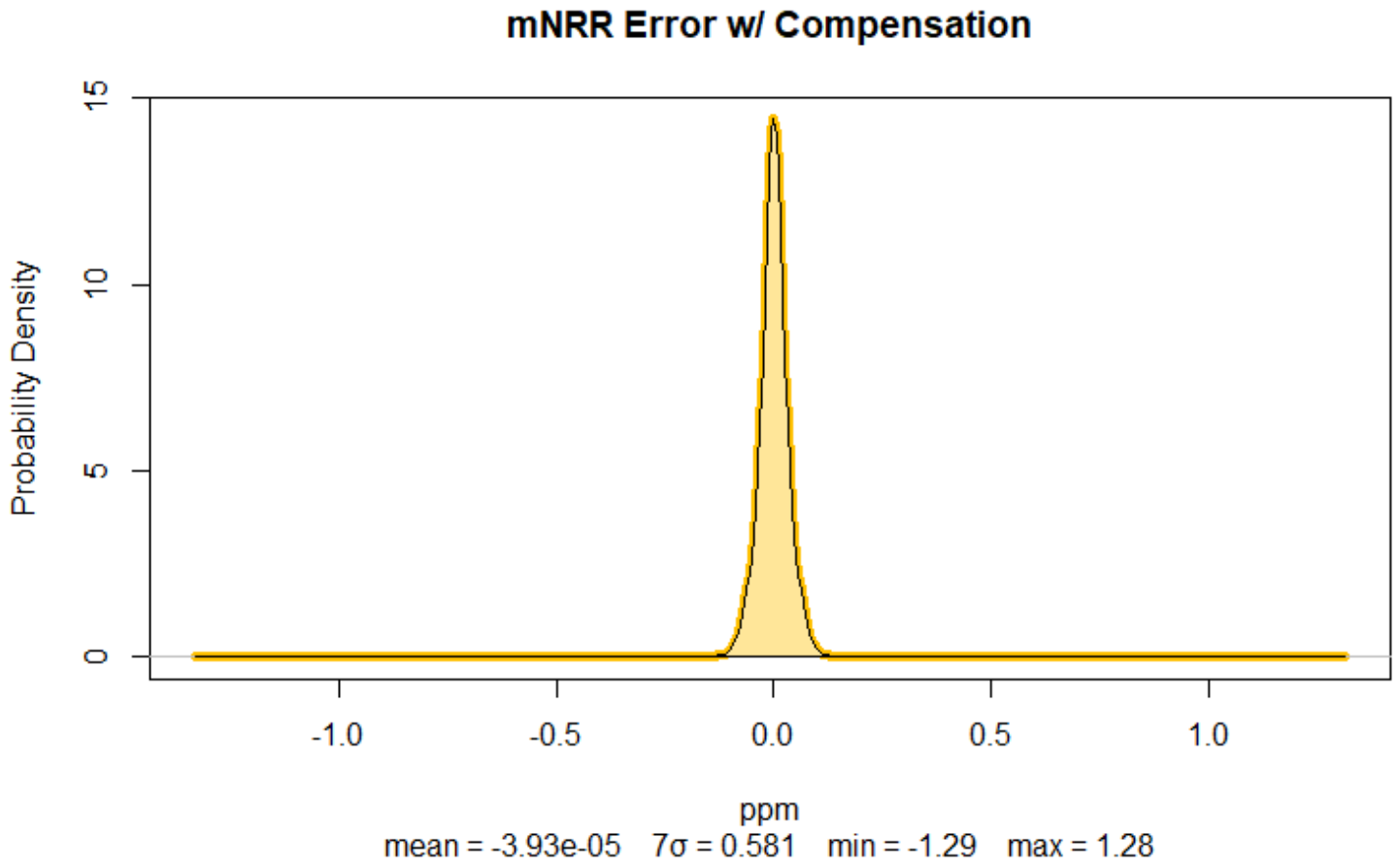
mNRR w/ Comp – N:5 A:5 P:10



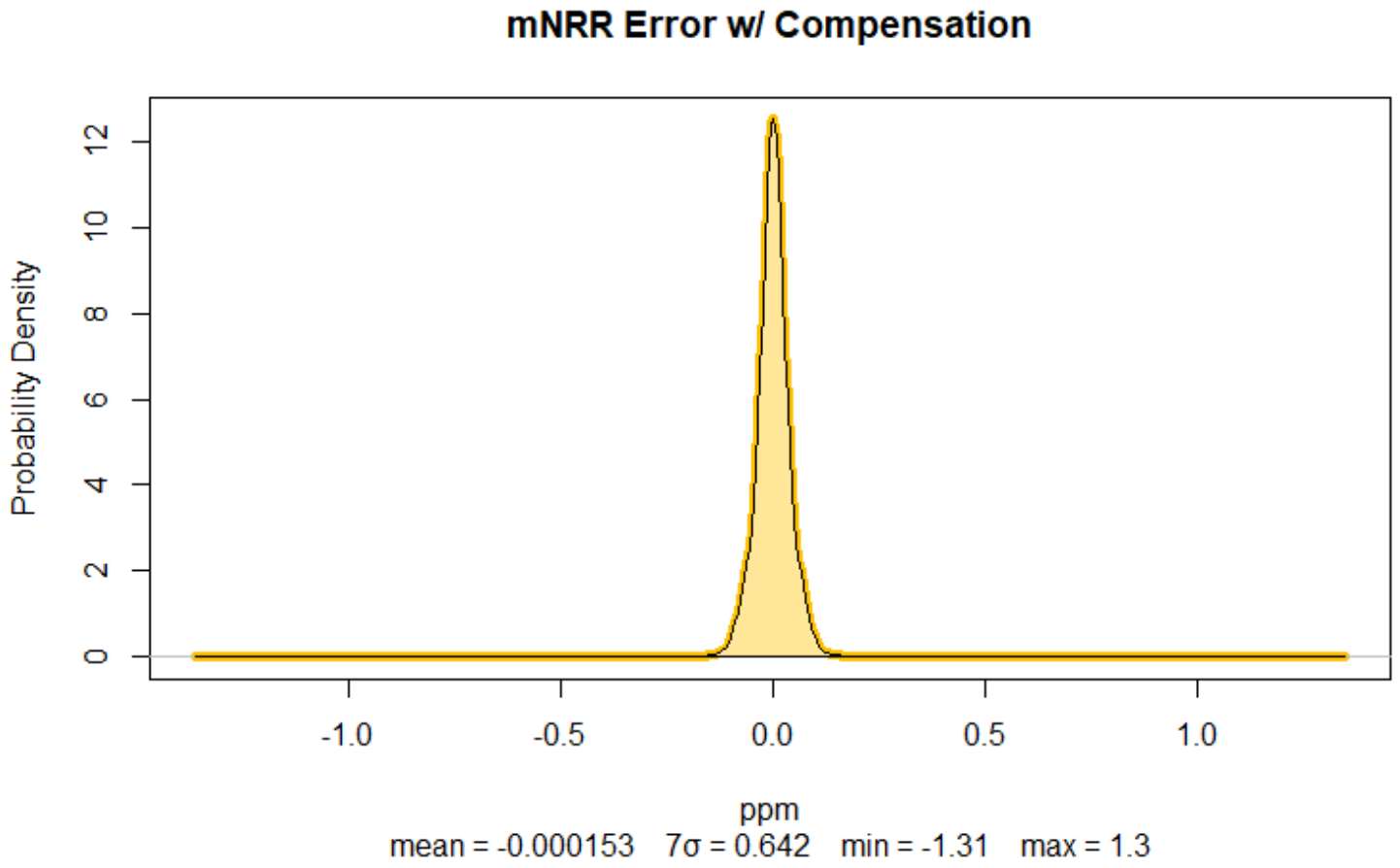
mNRR w/ Comp – N:6 A:6 P:12



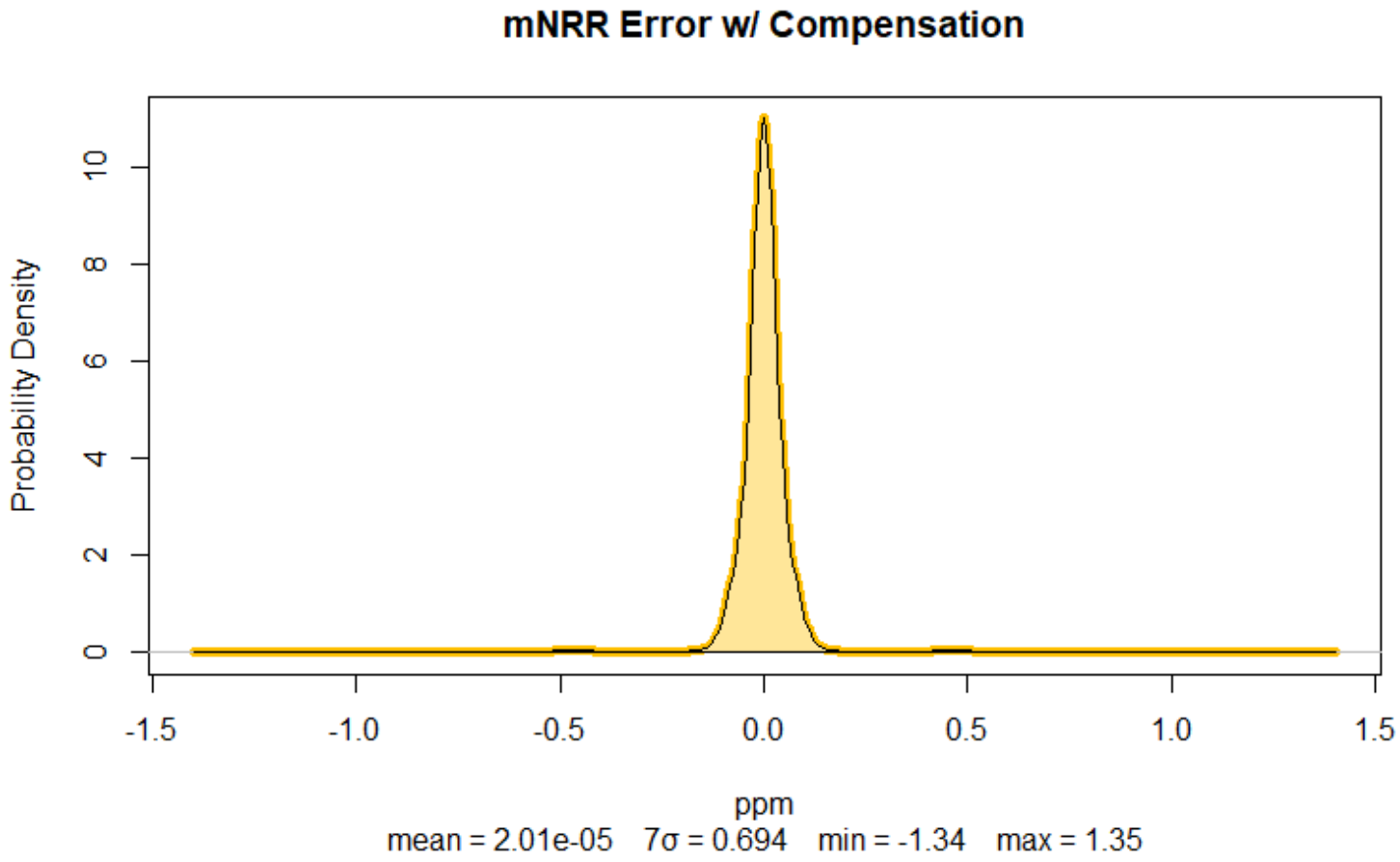
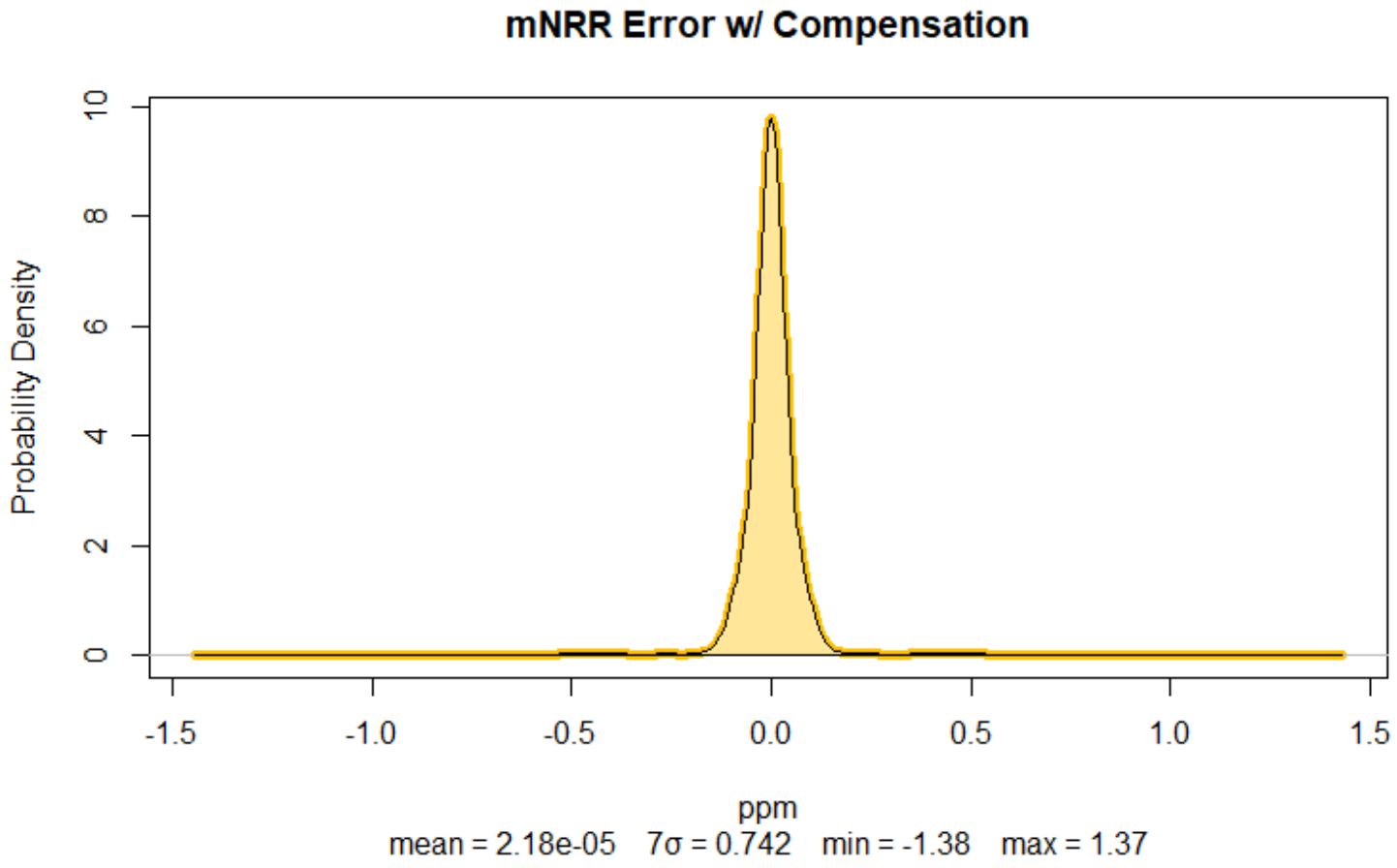
mNRR w/ Comp – N:7 A:7 P:14



mNRR w/ Comp – N:8 A:8 P:16



mNRR w/ Comp – N:9 A:9 P:18



mNRR w/ Comp – N:10 A:10 P:20