60802 Time Sync – Rate Ratio Drift Tracking & Error Compensation

David McCall – Intel Corporation

Version 2

References

[1] David McCall “60802 Time Sync – Normative Requirements & Testing”, v05, IEEE 802.1 TSN Contribution, April 2023

[2] David McCall “60802 Time Sync – Benefits of All PTP Relays using Same NRR Tracking Algorithm”, v01, IEEE 802.1 TSN Contribution, May 2023

[3] David McCall “60802 Time Synchronisation – Monte Carlo Analysis: 100-hop Model, “Linear” Clock Drift1, NRR Accumulation2 Overview & Details, Including Equations”, v2, IEEE 802.1 TSN Contribution, September 2022

[4] Dragan Obradovic “GrandMaster Frequency-Drift and its influence on the RR and MasterTime estimation at Slave Elements”, v1, IEEE 802.1 TSN Contribution, May 2023

Contents

- Recap of Overall Time Sync Approach
- Recap of dTE Related to RR Drift
- Problems with using NRR Drift Tracking & Error Compensation Approach for RR
- Alternative Approach: Additional Field in New TLV & Additional Calculations
- Implications for Grandmasters, End Stations & Normative Requirements
- Next Steps
  - 802.1ASdm PAR & CSD
- Summary

Recap of Overall Time Sync Approach

Recap…

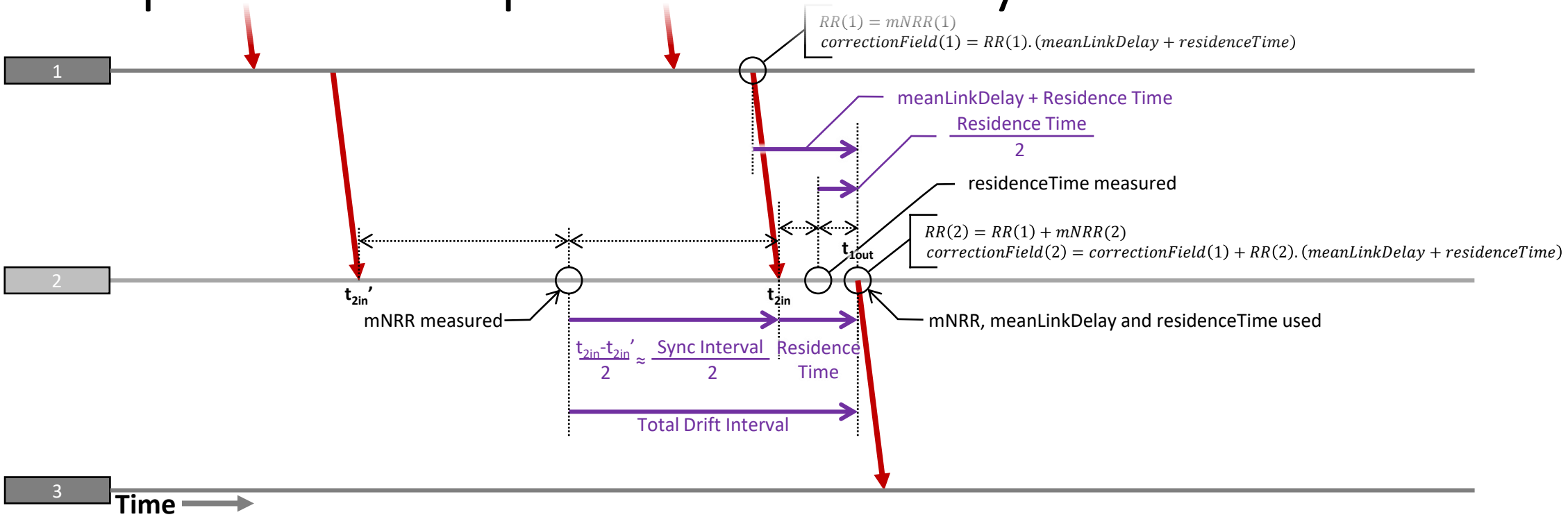
From [1]

- 60802's approach to achieving 1 µs time sync accuracy over 100 hops is (short version)…
  - Set parameters and limits (Sync Interval, Residence Time) and use NRR measurement techniques to limit errors, and also bias error generation away from Timestamp errors and towards errors due to Clock Drift
    - Use new TLV so that NRR can be calculated from Sync messages
  - Track NRR drift and RR drift and compensate for resulting errors due to Clock Drift
  - Assume worst case scenarios…
    - Due to reasonable temperature changes at individual nodes (modelled via temperature ramps over time) and use of regular Xos
    - At any given time, some nodes will experience varying amounts of clock drift while others experience zero clock drift.
  - Take a statistical approach to achieving performance guarantees.
    - Worst case modelling indicates errors much greater than 1 µs are possible
    - Statistical modelling is targeting mean time between failure >5,000 years

Recap of dTE Related to RR Drift

Clock Drift Error – Relevant Intervals
4 Hops – 2nd Hop – NRR from Sync

From [2] Updated



Sync

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

Equations - $mNRR_{error}$

From [3] Updated

Primary Errors

$$mNRR_{errorTS\_X} = \frac{\left(t_{3pderror} - t_{3pderror}{}'\right) - \left(t_{4pderror} - t_{4pderror}{}'\right)}{T_{pdelay2pdelay}} \quad \text{ppm}$$

$$mNRR_{errorCD\_X} = \frac{T_{sync2sync}(clockDrift_n - clockDrift_{n-1})}{2 \times 10^3} \quad \text{ppm}$$

① **NRR**

$$mNRR_{error\_X} = mNRR_{errorTS\_X} + mNRR_{errorCD\_X} \quad \text{ppm}$$

Error Components

$mNRR_{error}$ breaks down into a Timestamp Error and an error due to Clock Drift, so there are no additional error components or calculations.

Equations - RR_{error}

From [3] Updated

Primary Errors

$$RR_{errrorCD\_NRR2Sync\_X} = \frac{residenceTime(clockDrift_n - clockDrift_{n-1})}{10^3}$$ ② **NRR** **ppm**

$$RR_{errrorCD\_RR2Sync\_X} = \frac{residenceTime(clockDrift_{n-1} - clockDrift_{GM})}{10^3}$$ ③ **RR_{n-1}** **ppm**

$$RR_{errror\_SUM}(n) = RR_{error\_SUM}(n-1) + mNRR_{error\_X} + RR_{errorCD\_NRRtoSync\_X} + RR_{errorCD\_RRtoSync\_X}$$ **ppm**

Error Components

$$RR_{errrorNRR\_CD\_SUM}(n) = RR_{errrorNRR\_CD\_SUM}(n-1) + mNRR_{errrorCD\_X}(n)$$ **ppm**

$$RR_{errrorCD\_NRR2sync\_SUM}(n) = RR_{errrorCD\_NRR2sync\_SUM}(n-1) + RR_{errrorCD\_NRR2Sync\_X}(n)$$ **ppm**

$$RR_{errrorCD\_RR2sync\_SUM}(n) = RR_{errrorCD\_RR2sync\_SUM}(n-1) + RR_{errrorCD\_RR2Sync\_X}(n)$$ **ppm**

$$RR_{errrorNRR\_SUM}(n) = RR_{errrorNRR\_SUM}(n-1) + mNRR_{error\_X}(n)$$ **ppm**

$$RR_{errrorTS\_SUM}(n) = RR_{errrorTS\_SUM}(n-1) + mNRR_{errrorTS\_X}(n)$$ **ppm**

$$RR_{errrorCD\_SUM}(n) = RR_{errrorNRR\_CD\_SUM}(n) + RR_{errrorCD\_NRR2sync\_SUM}(n) + RR_{errrorCD\_RR2sync\_SUM}(n)$$ **ppm**

At the last hop, there is no Residence Time, so $RR_{errorCD\_RR2Sync\_SUM} = 0$.

Equations – RT_{error} – Per Hop (Except Last)

From [3] Updated

Primary Errors

$$RT_{errorTSdirect\_X} = t_{1souterror} - t_{2sinerror} \quad \text{ns}$$

$$RT_{errrorCDdirect\_X} = \frac{residenceTime^2(clockDrift_n - clockDrift_{GM})}{2 \times 10^3} \quad \text{ns}$$

④ $\mathbf{RR_n}$

$$RT_{errrorRR\_X} = residenceTime \times RR_{error\_SUM} \quad \text{ns}$$

$$RT_{errror\_X} = RT_{errorTSdirect\_X} + RT_{errrorRR\_X} + RT_{errorCDdirect\_X} \quad \text{ns}$$

Error Components

$$RT_{errrorRR\_NRR\_CD\_X} = residenceTime \times RR_{errorNRR\_CD\_SUM} \quad \text{ns}$$

$$RT_{errrorRR\_CD\_NRR2sync\_X} = residenceTime \times RR_{errorCD\_NRR2sync\_SUM} \quad \text{ns}$$

$$RT_{errrorRR\_CD\_RR2sync\_X} = residenceTime \times RR_{errorCD\_RR2sync\_SUM} \quad \text{ns}$$

$$RT_{errrorRR\_TS\_X} = residenceTime \times RR_{errorTS\_SUM} \quad \text{ns}$$

$$RT_{errrorRR\_NRR\_X} = RT_{errrorRR\_NRR\_CD\_X} + RT_{errrorRR\_TS\_X} \quad \text{ns}$$

$$RT_{errrorRR\_CD\_X} = RT_{errrorRR\_NRR\_CD\_X} + RT_{errrorRR\_CD\_NRR2sync\_X} + RT_{errrorRR\_CD\_RR2sync\_X} \quad \text{ns}$$

$$RT_{errrorCD\_X} = RT_{errrorCDdirect\_X} + RT_{errrorRR\_CD\_X} \quad \text{ns}$$

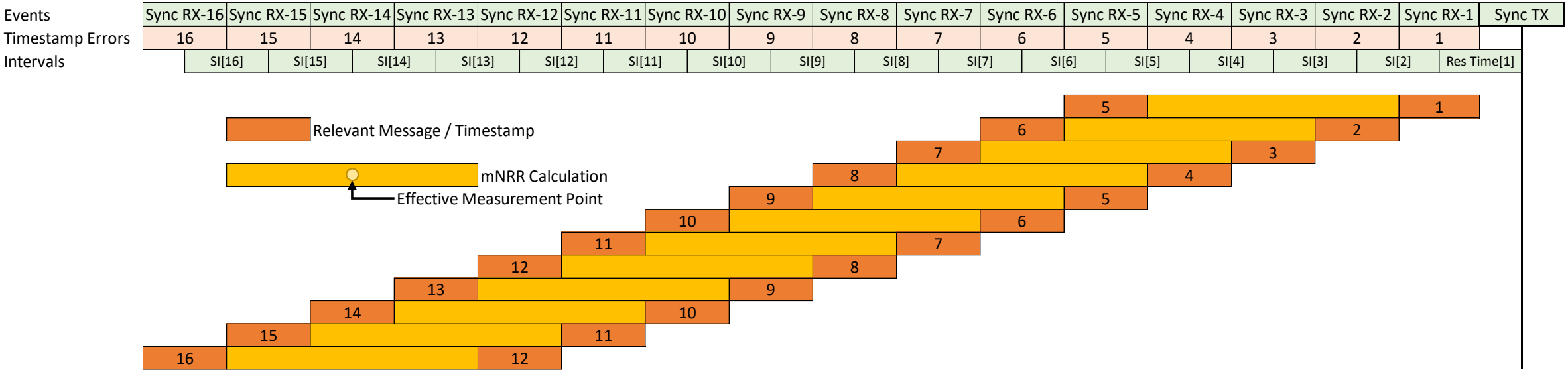
$$RT_{errrorTS\_X} = RT_{errorTSdirect\_X} + RT_{errrorRR\_TS\_X} \quad \text{ns}$$

Problems with using NRR Drift Tracking & Error Compensation Approach for RR

Content of this section…

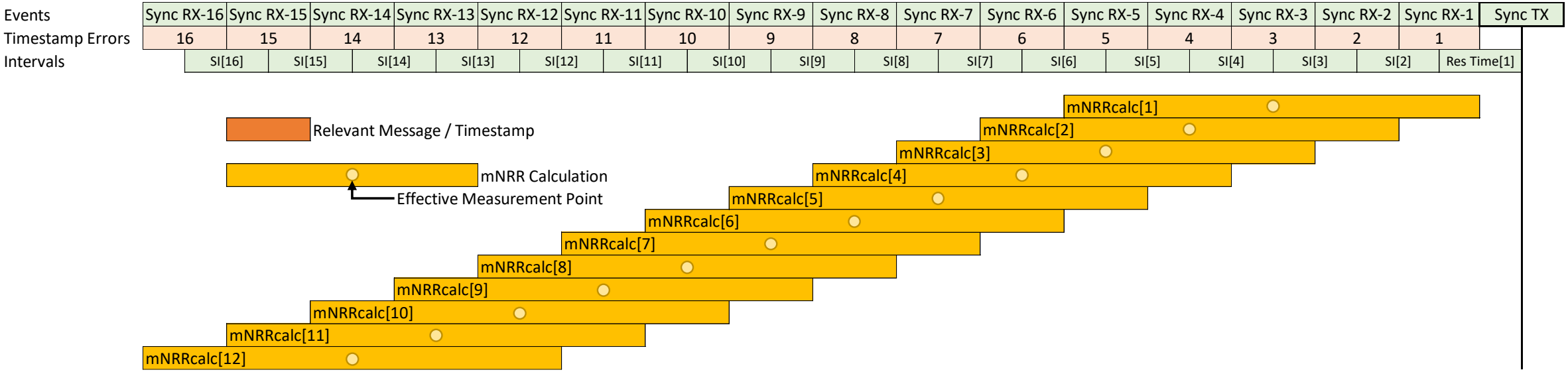
- Recap NRR drift tracking & compensation algorithm.
  - Update to take account of [4]
- Applying same approach to RR drift tracking and compensation
- Alternative no-averaging approach

mNRR Drift Tracking



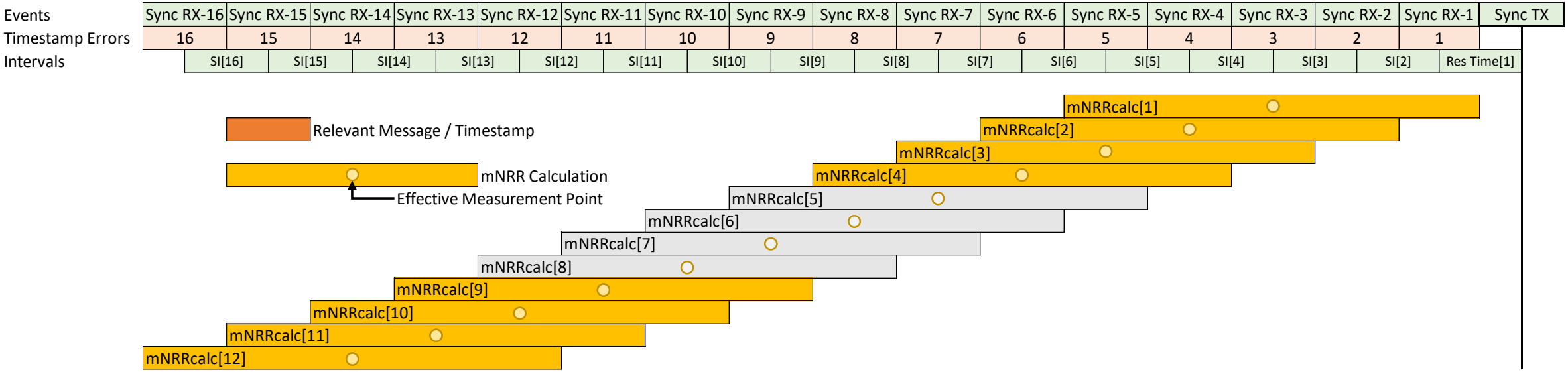
- Instead of keeping track of the past 4 calculation, each node keeps track of the past 12.

mNRR Drift Tracking



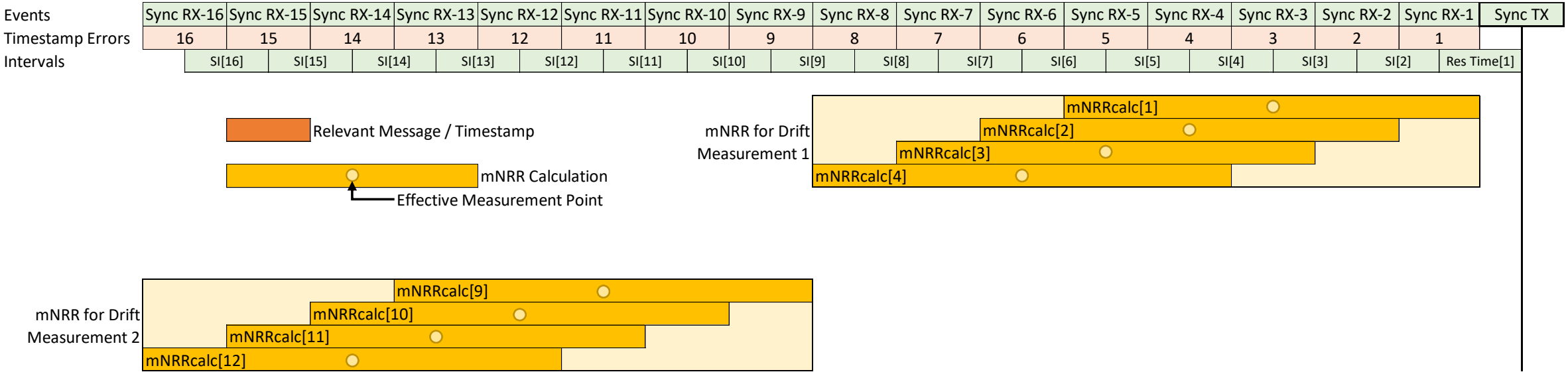
- Each calculation has an associated effective measurement time.

mNRR Drift Tracking



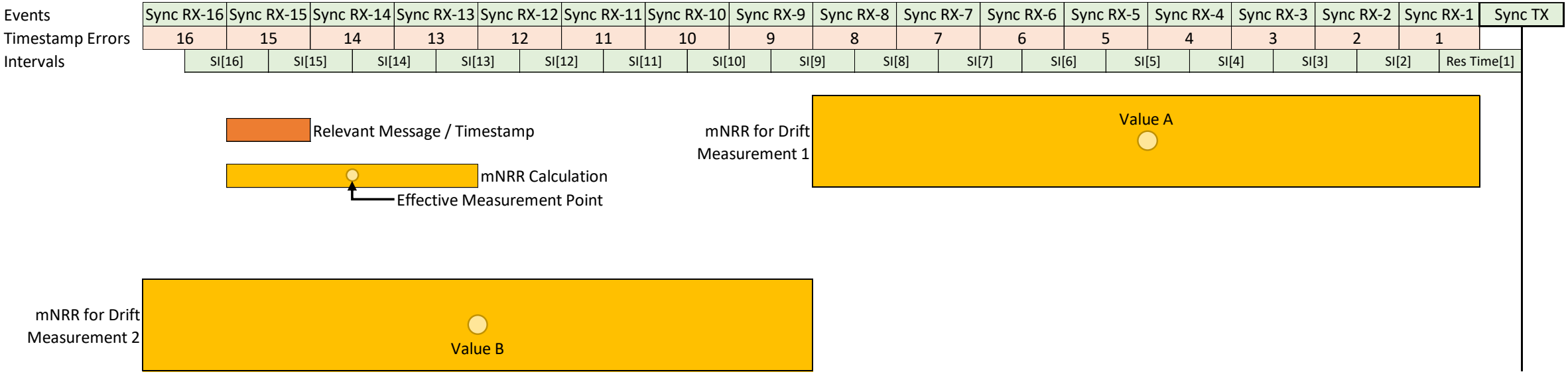
- The middle 4 calculations aren't used (but are remembered for future drift tracking)

mNRR Drift Tracking



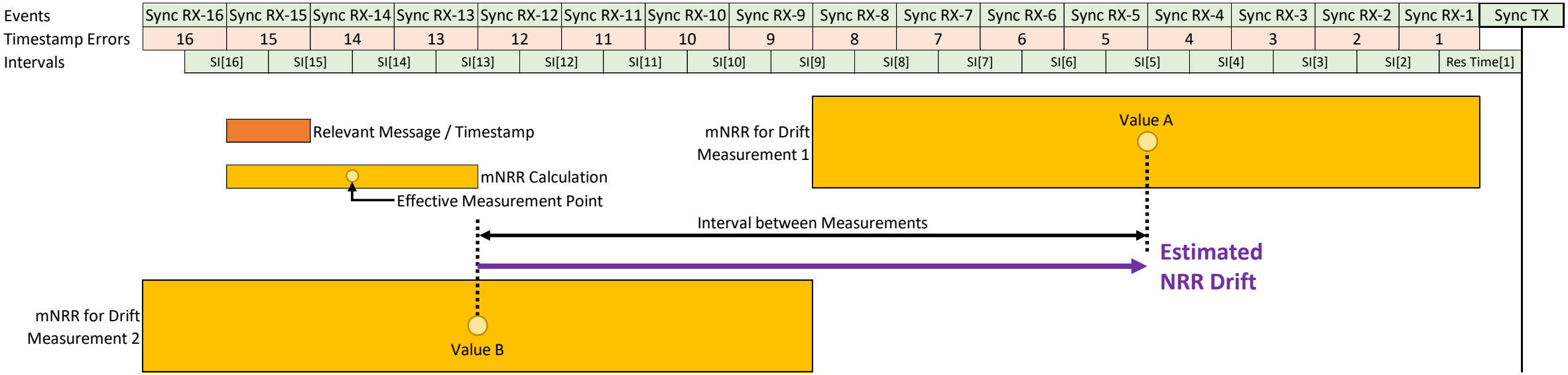
- The other calculations are used to make two measurements of NRR.

mNRR Drift Tracking



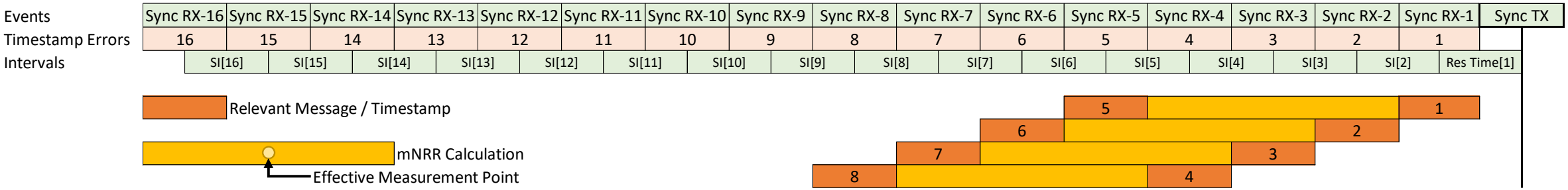
- Average of the values of each group of calculations.
- Average of the effective measurement points.

mNRR Drift Tracking



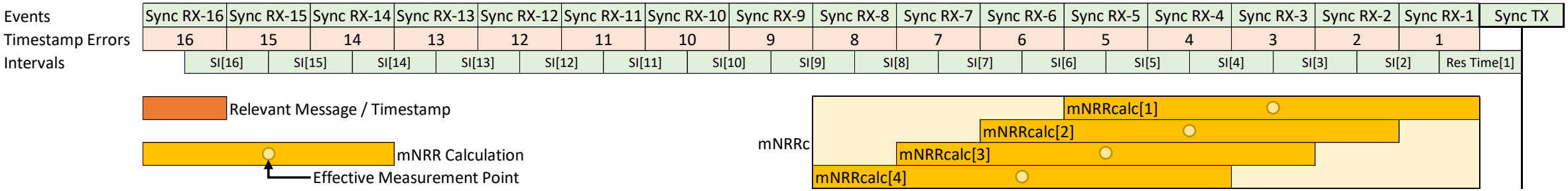
- Estimated NRR drift is the rate of change between the two values.

mNRR Drift – Error Compensation



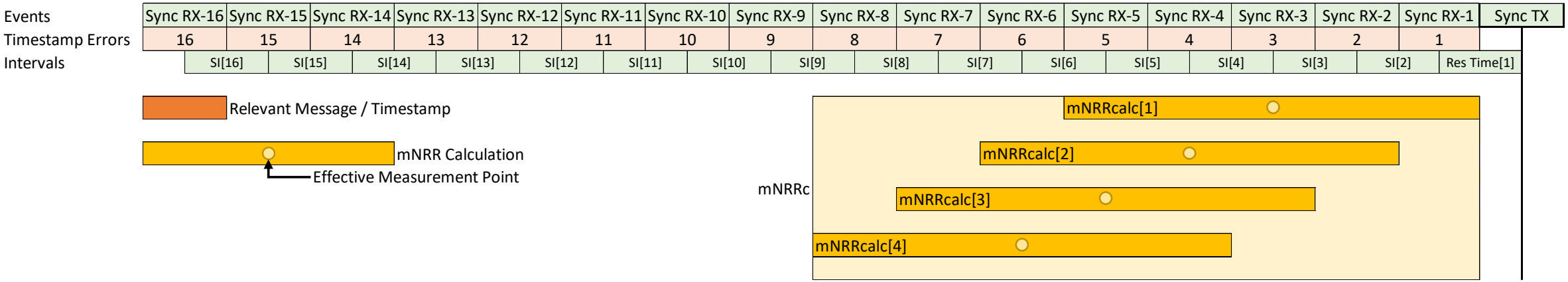
- Compensation for error due to clock drift in the mNRR measurement starts with the same timestamp information as before.
- Four calculations: (s, s-4), (s-1 & s-5), (s-2 & s-6), (s-3 & s-7)

mNRR Drift – Error Compensation



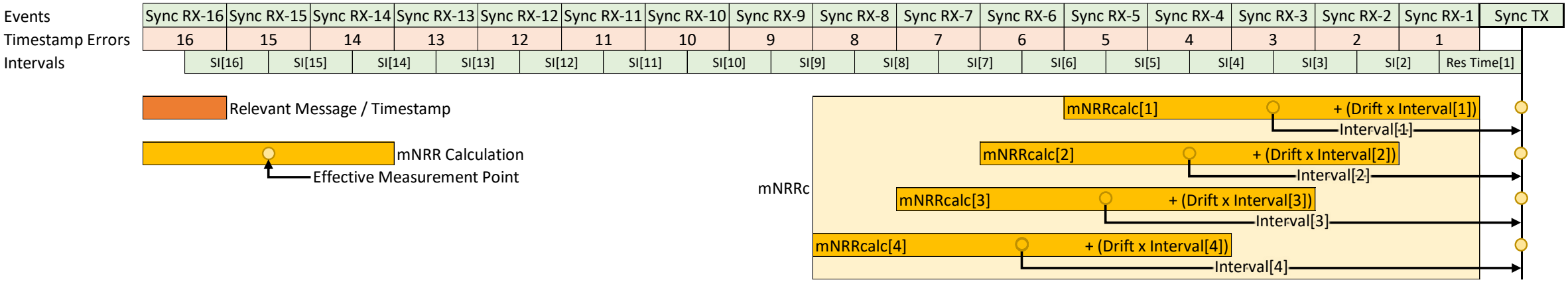
- Again, each value has an associated effective measurement point.

mNRR Drift – Error Compensation



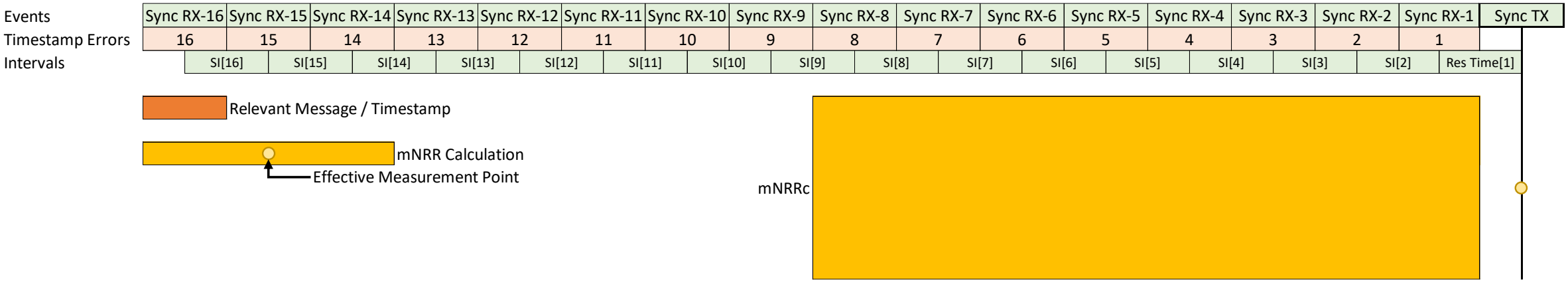
- But this time, prior to averaging, each value is adjusted using the effective measurement point and estimated drift rate to an estimate of the NRR at the point of Sync TX.

mNRR Drift – Error Compensation



- "Older" calculations require more adjustment to compensate for NRR drift.
- This process effectively moves the effective measurement point to Sync TX

mNRR Drift – Error Compensation

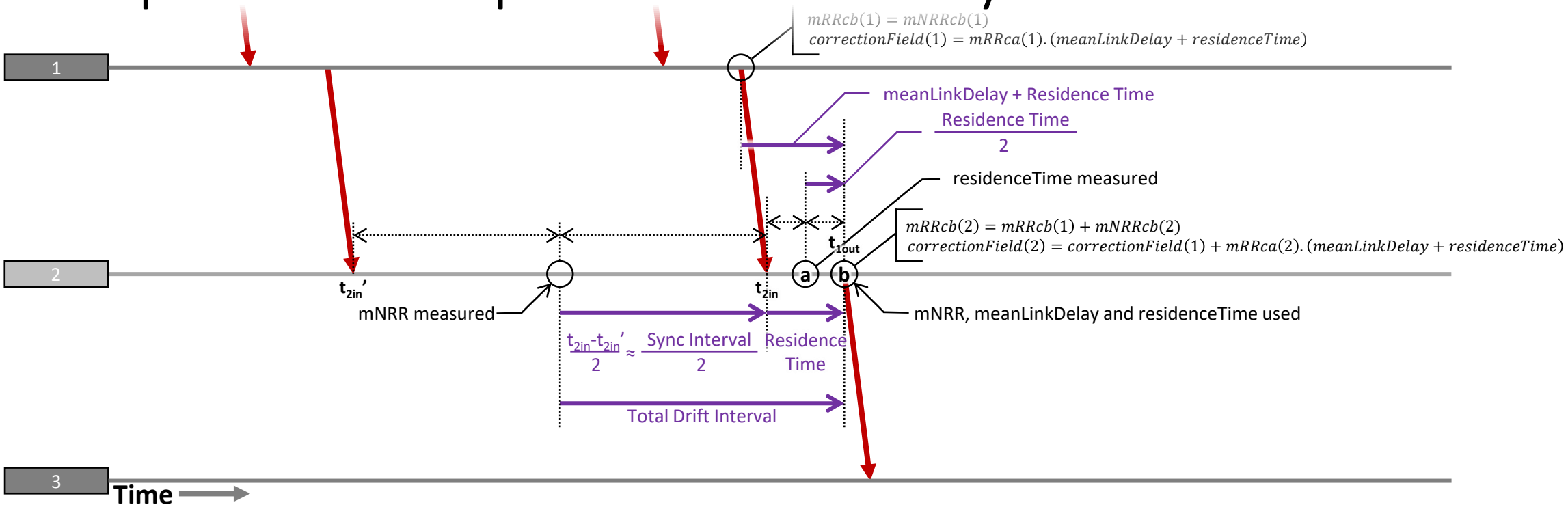


- Take an average of the four adjusted values to determine the measured, with error compensation, NRR: mNRRc.

Two mNRRc Calculations

- From [4], ideally there would be two RR calculations per PTP Relay Instance…

Clock Drift Error – Relevant Intervals
4 Hops – 2nd Hop – NRR from Sync



Sync

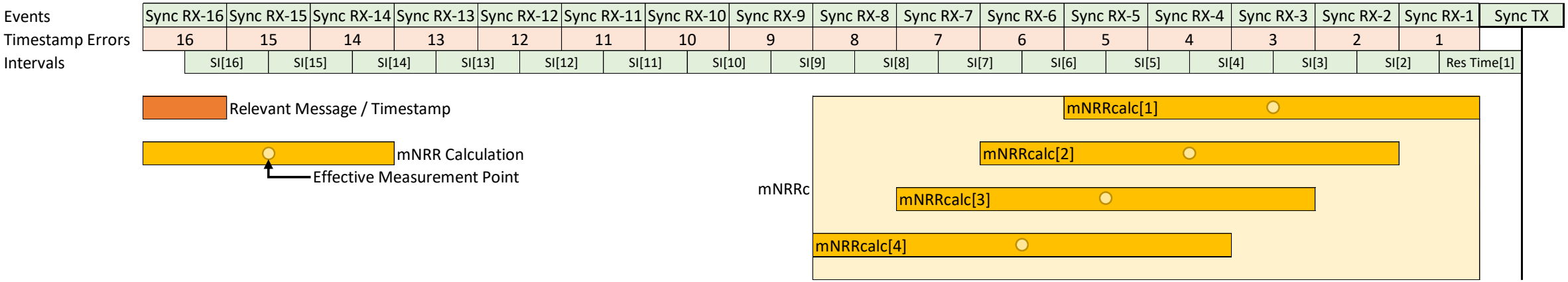
- mRRca for use when calculating correctionField – half way between Sync egress at Node n-1 and Sync egress at Node n

$$correctionField(2) = correctionField(1) + mRRca(2).(meanLinkDelay + residenceTime) \qquad mRRca(2) = mRRca(1) + mNRRca(2)$$

- mRRcb for use when calculating rateRatio value for output Sync message – Sync egress at Node n

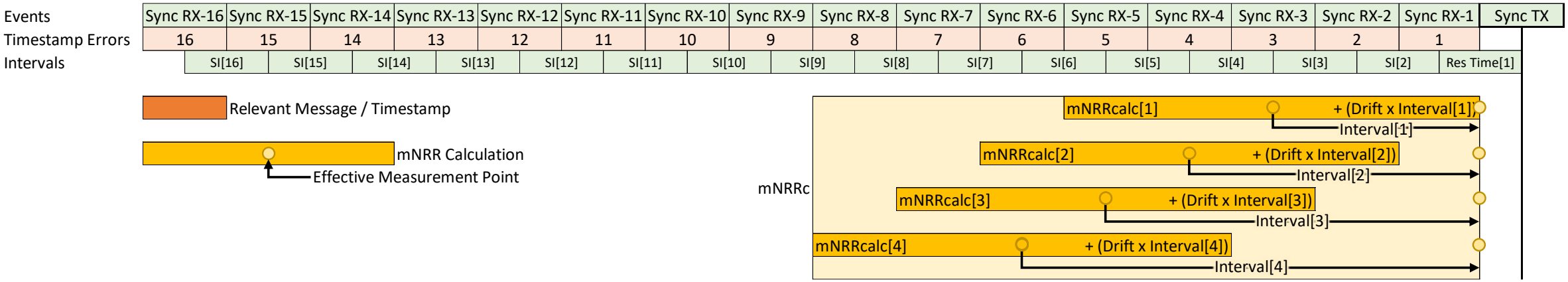
$$mRRcb(2) = mRRcb(1) + mNRRcb(2)$$

NRR Drift – Error Compensation



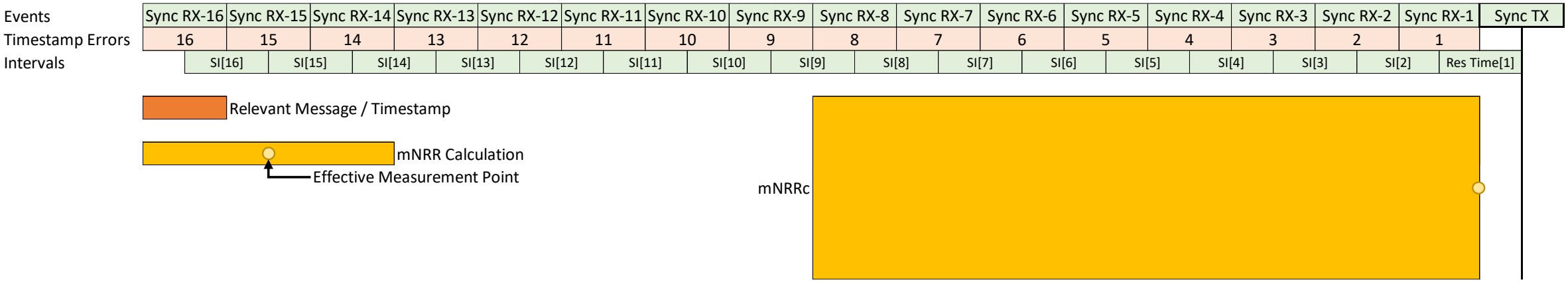
- Same starting point for compensation.

NRR Drift – Error Compensation – mNRRca



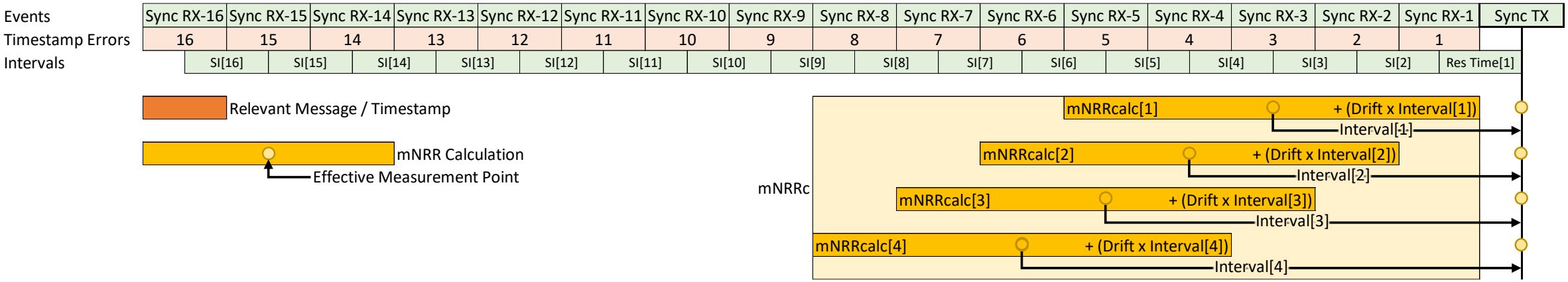
- "Older" calculations require more adjustment to compensate for NRR drift.
- This process effectively moves the effective measurement point to half-way between Sync egress at Node n-1 and Sync egress at Node n

NRR Drift – Error Compensation – mNRRca



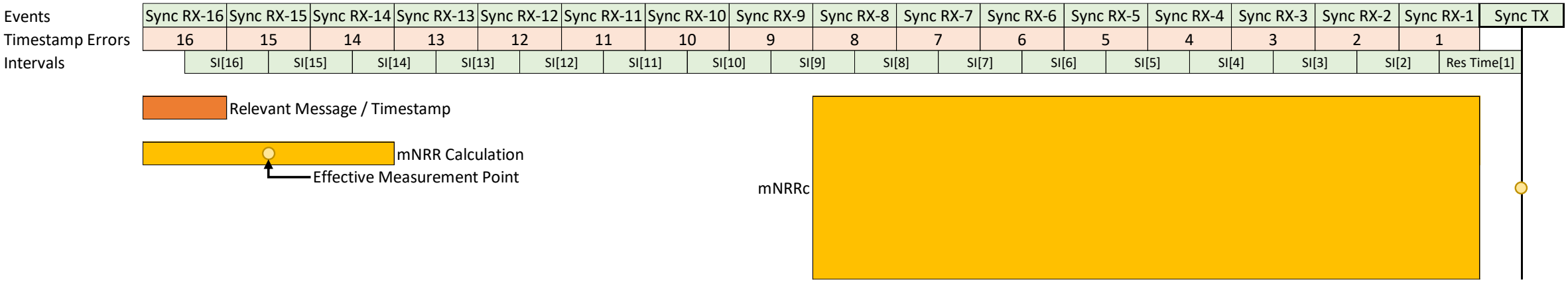
- Take an average of the four adjusted values to determine the measured, with error compensation for correctionField: mNRRca.

NRR Drift – Error Compensation – mNRRcb



- "Older" calculations require more adjustment to compensate for NRR drift.
- This process effectively moves the effective measurement point to Sync TX

NRR Drift – Error Compensation – mNRRcb

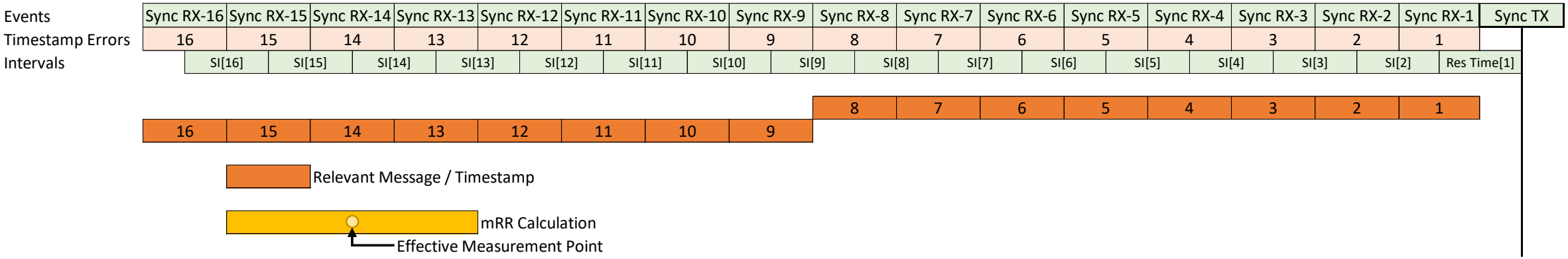


- Take an average of the four adjusted values to determine the measured, with error compensation for rateRatio: mNRRcb.

RR Drift – Tracking & Error Compensation

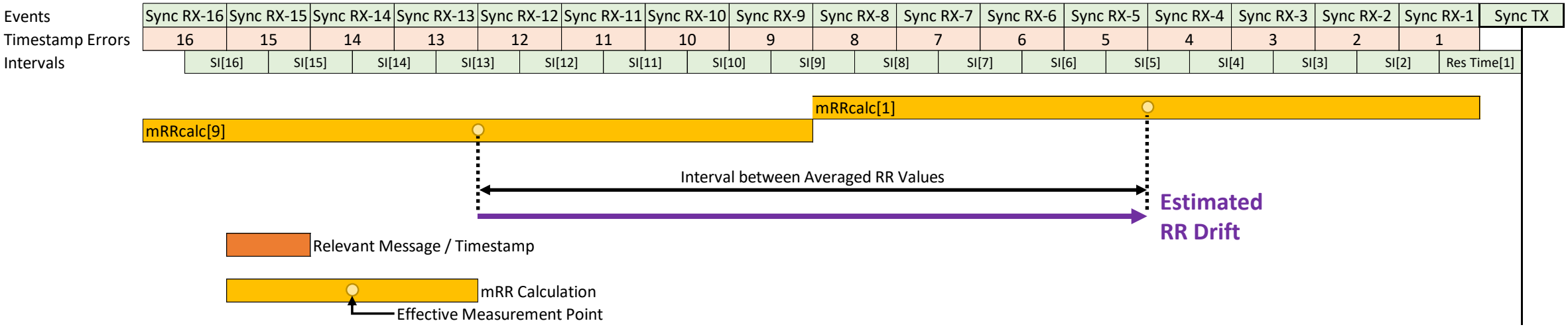
- RR values aren't measured by the current node, they are provided in Sync messages from the previous node.
- Amount of error in RR values increases with each hop.
  - Due to Timestamp errors and errors due to Clock Drift
- Similarly to NRR
  - Errors due to Timestamp errors can be reduced via averaging
  - Errors due to Clock Drift can be compensated for
    - Effectiveness of compensation will be limited by Timestamp errors and non-linear drift (if algorithm assumes linearity)
    - Algorithms that draw from data in the past can be vulnerable to failures due to discontinuities at any point in their input data window…which can be OK if the window is small.

RR Drift – Tracking



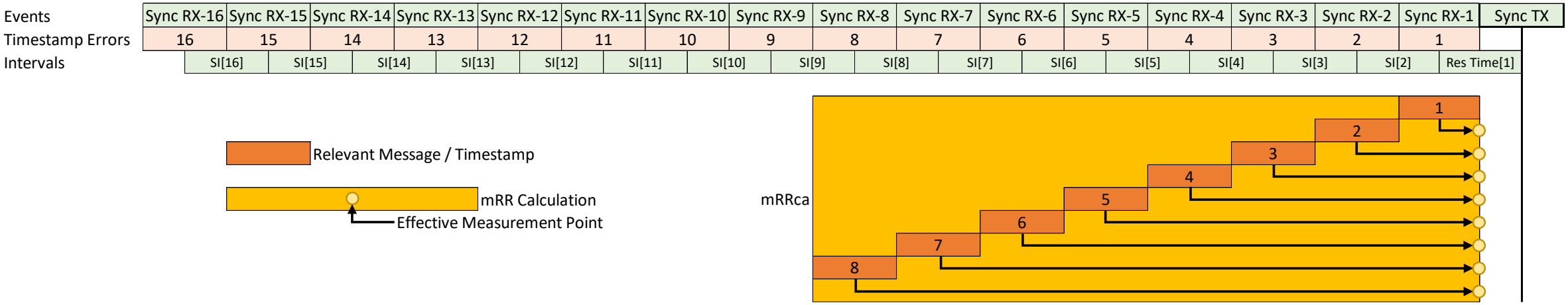
- A similar approach to the NRR algorithm would involve taking 2 seconds of data to calculate RR Drift.

RR Drift – Tracking



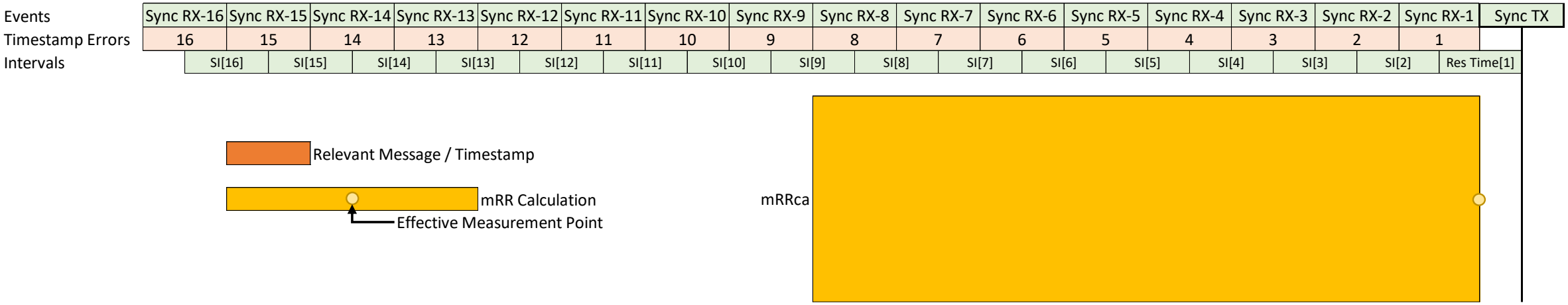
- Take two averages: 8 most recent RR values; next 8 most recent values.
  - Effective measurement point for an average is the average of the related Sync egress from the previous node, measured in local node time, i.e. Sync ingress timestamp minus relevant meanLinkDelay.
- Estimated RR Drift Rate is rate of change between the two averages.

RR Drift – Error Compensation - mRRca



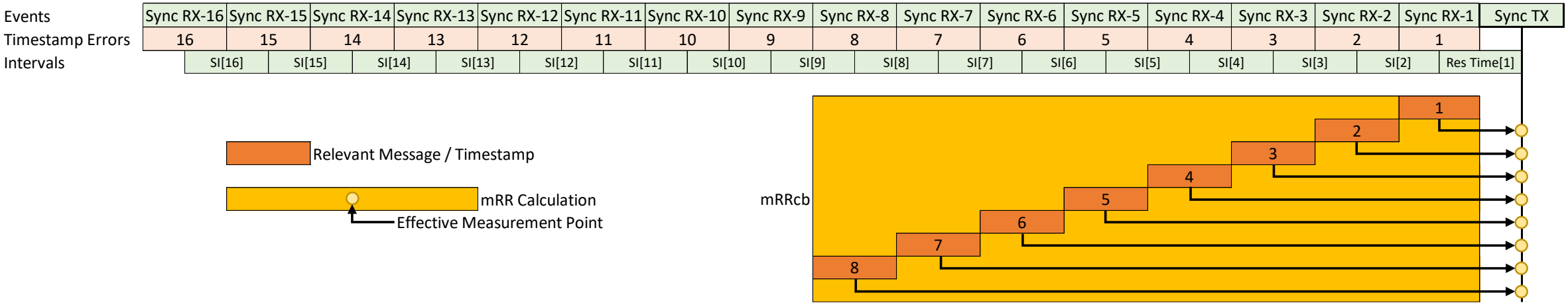
- Similar to NRR algorithm, prior to averaging, each RR value is adjusted using the effective measurement point and estimated drift rate to an estimate of the RR half-way between Sync egress from Node n-1 and Sync egress from Node n.

RR Drift – Error Compensation - mRRca



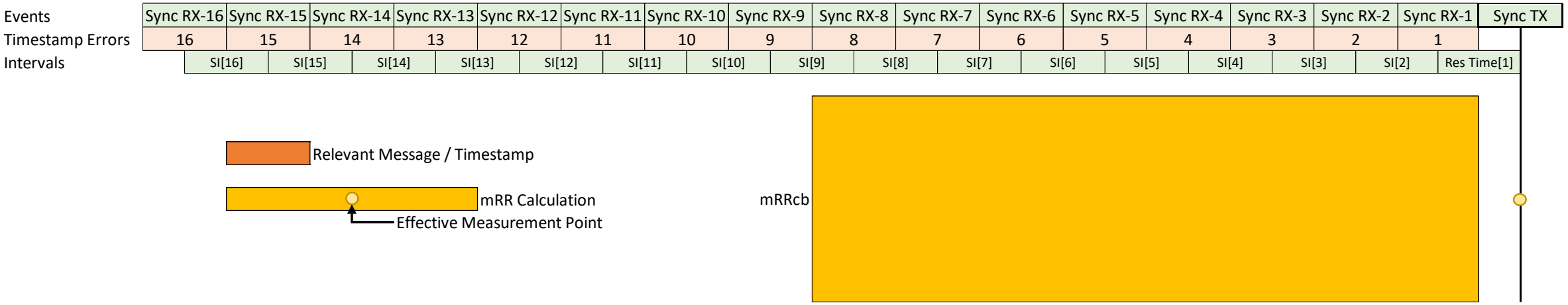
- RR measurement can be used for correctionField calculation: mRRca

RR Drift – Error Compensation - mRRcb



- Same calculation, but RR value is adjusted using the effective measurement point and estimated drift rate to an estimate of the RR at Sync egress from Node n.

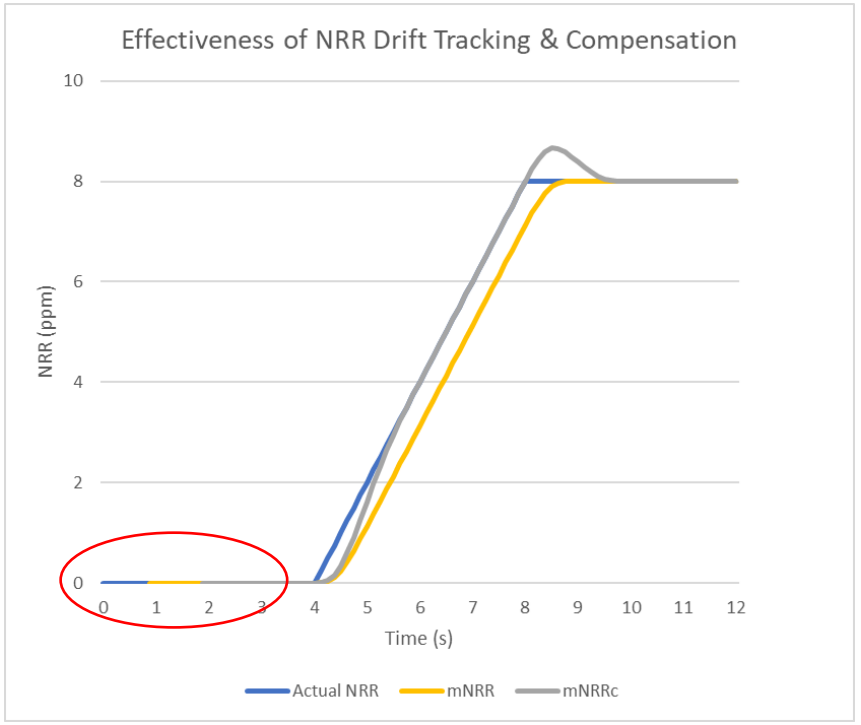
RR Drift – Error Compensation - mRRcb



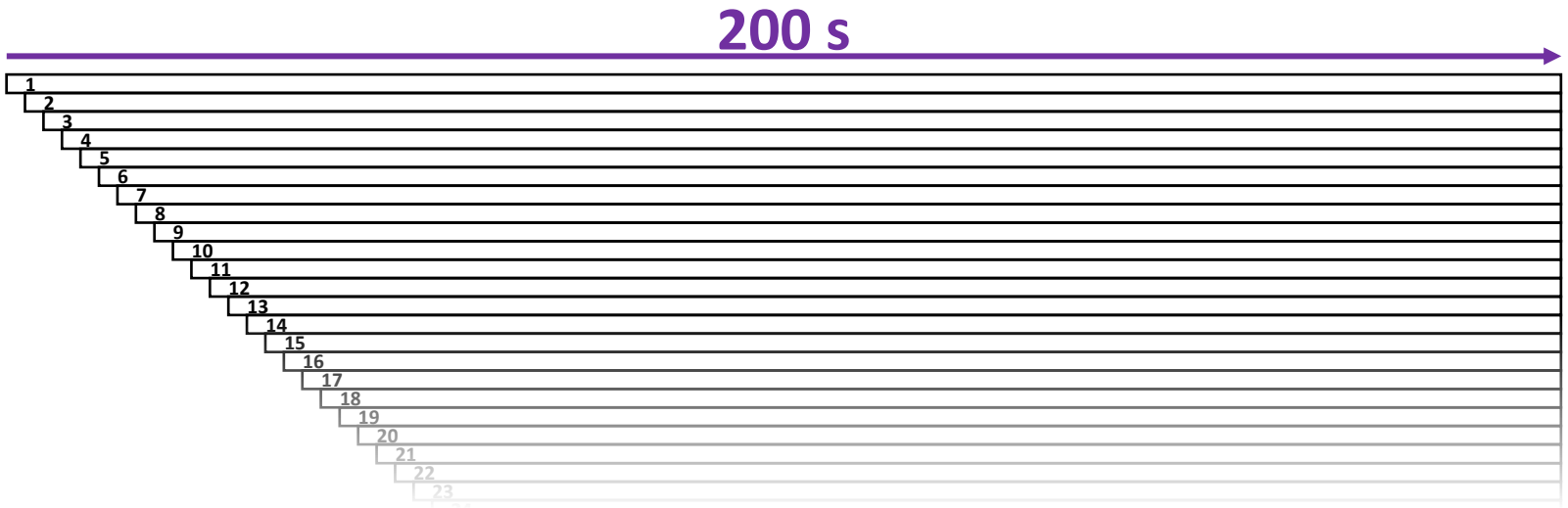
- RR measurement can be used for rateRatio field calculation: mRRcb

Startup / Historical Data Problem

- This approach works in some ways, but is problematic.
- The the root cause of the problem can be seen in this graph from [1].
- It takes 1 second for a “regular” mNRR measurement to start at Node 1, 2 seconds for mNRRcb (corrected) measurement to start.
  - At Node 1, mNRRcb is the same as mRRcb.
- RR algorithm at Node 2 needs 2 seconds of valid mRRcb data from Node 1 before it can start generating mRRcb data to pass on to Node 3.
- RR Algorithm at Node 3 needs 2 seconds of valid mRRcb data from Node 1 before it can start generating mRRcb data to pass on to Node 4.

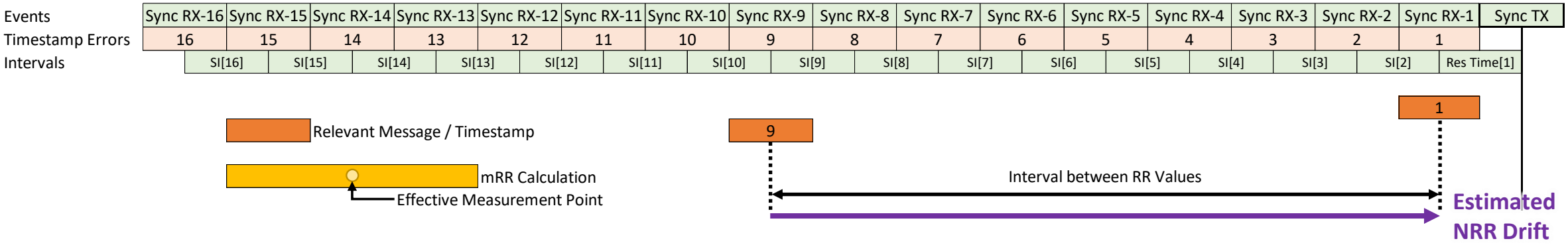


Propagation of RR Drift Tracking



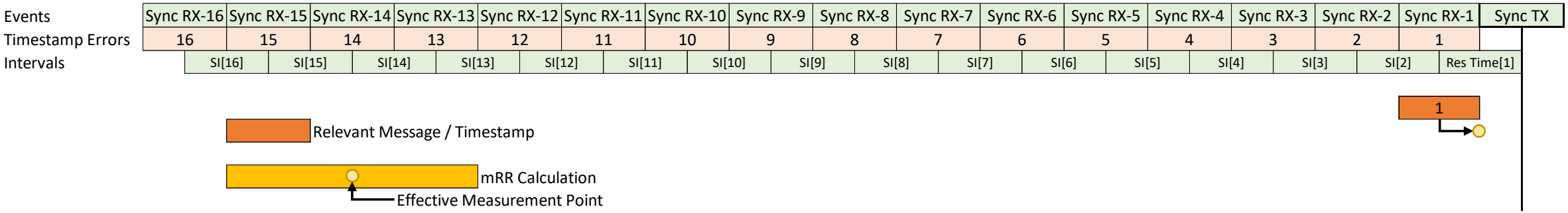
- Node 100 needs 2 seconds of valid mRRcb data from Node 99…
- …which means mRRcd data at Node 1 must have been valid for 200 seconds (3 mins 20 sec)!
- This isn't just a startup problem. Data from 200 seconds ago at the GM is still affecting RR at End Station.

RR Drift – Tracking 2



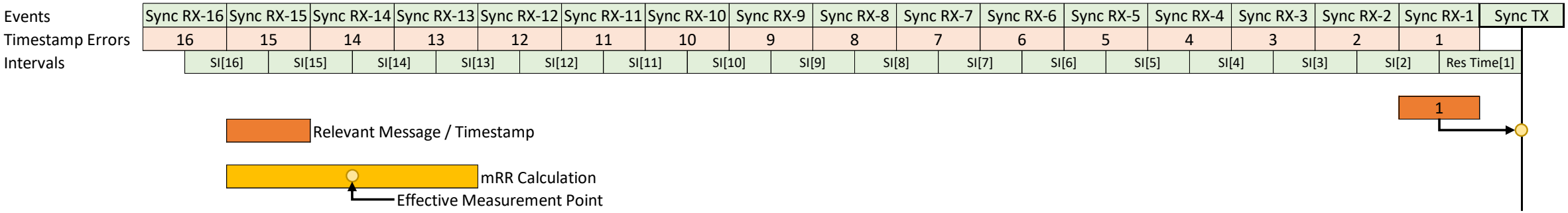
- Maybe we don't need to take an average of RR values? After all, it's an accumulation of NRR values, which have already been averaged.

RR Drift – Error Compensation - mRRca



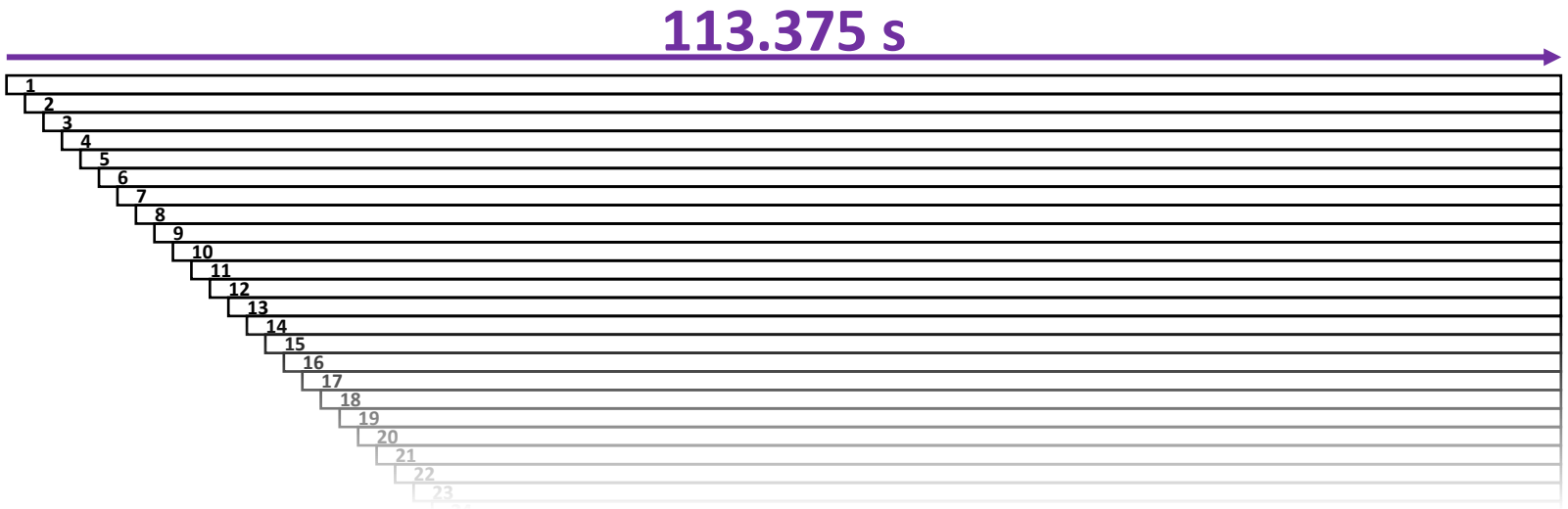
- Most recent RR value is adjusted using the effective measurement point and estimated drift rate to an estimate of the RR half-way between Sync egress from Node n-1 and Sync egress from Node n.
- RR measurement can be used for correctionField calculation: mRRca

RR Drift – Error Compensation - mRRcb



- Most recent RR value is adjusted using the effective measurement point and estimated drift rate to an estimate of the RR at Sync egress from Node n.
- RR measurement can be used for rateRatio field calculation: mRRcb

Propagation of RR Drift Tracking



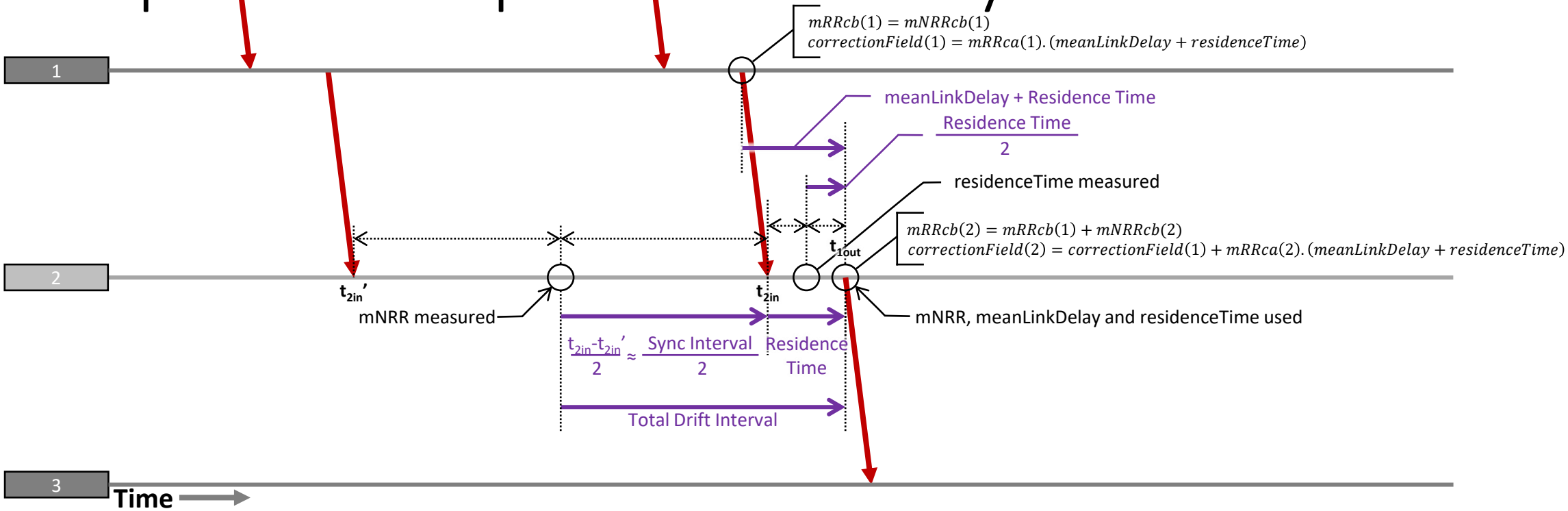
- Node 100 needs 1.125 seconds of valid mRRcb data from Node 99…
- …which means mRRcd data at Node 1 must have been valid for 113.375 seconds (almost 2 mins)!
  - Bonus points if you can tell why it's not 112.5 seconds.

Limits of This Approach

- Always need at least 2 values to calculate drift, i.e. 125 ms of data. So minimum startup / latency time is 14.375 seconds.
- Initial calculations suggest there is too much noise, especially at higher numbered nodes, for accurate RR drift estimate if only 125 ms of data is used.
- But, there may be a better way…

Alternative Approach: Additional Field in New TLV & Additional Calculations

Clock Drift Error – Relevant Intervals
4 Hops – 2nd Hop – NRR from Sync



- Node 2 wants to know drift rate of RR(1)
- Node 1 already knows the drift rate of RR(1)
  - It's the drift rate of mNRR(1)

Sync

New Field in Drift_Tracking TLV

- Node 1 could pass rateRatioDrift (RRb(1) drift rate) to Node 2 in Drift_Tracking TLV
- Removes need for Node 2 to estimate RR Drift Rate at Node 1, it can just use the value from the previous node.
- Node 3 wants to know RR(2) drift rate…which Node 2 can calculate via a combination of NRR(2) Drift Rate and RR(1) drift rate
  - Similar to the way that each node calculates RR(n) from RR(n-1) and NRR(n).

RR(n) Drift Rate from RR(n-1) Drift Rate

$$RR(n) = RR(n-1) \times NRR(n)$$ **ratio**

$$\frac{dRR(n)}{dt_n} = \left(\frac{dRR(n-1)}{dt_n} \times NRR(n)\right) + \left(\frac{dNRR(n)}{dt_n} \times RR(n-1)\right)$$ **ratio/s**

Local Clocks at Node n and Node n-1 may have different offsets (ppm), so…

$$\frac{dRR(n-1)}{dt_n} \neq \frac{dRR(n-1)}{dt_{n-1}}$$

Thanks to Geoff Garner for assistance with equations on this & following pages.

RR(n) Drift Rate from RR(n-1) Drift Rate

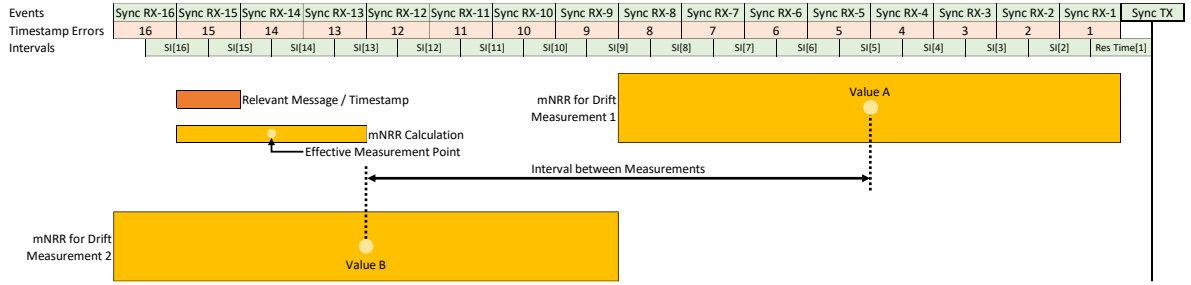
Easiest to think of an approach to remedy this in terms of 2nd hop where Node n-1 is Node 1 and at, Node 1, RR = NRR

$$\frac{dRR(2)}{dt_2} = \left(\frac{dRR(1)}{dt_2} \times NRR(2)\right) + \left(\frac{dNRR(2)}{dt_2} \times RR(1)\right)$$

$$= \left(\frac{dNRR(1)}{dt_2} \times NRR(2)\right) + \left(\frac{dNRR(2)}{dt_2} \times NRR(1)\right)$$

RR(n) Drift Rate from RR(n-1) Drift Rate

NRR drift is calculated as a rate of change between two values (A &B) over a time interval (see previous slide). The time interval is in terms of the Local Clock current node (Node n).



Effective measurement time for each NRRcalc is the half way between two Sync ingress timestamps, which are from the Local Clock. Thus, effective measurement point of the two values used for calculating drift rate are also based on Local Clock.

$$NRRdrift_{tn}\,(n) = \frac{mNRRdriftValueA(n) - mNRRdriftValueB(n)}{mNRRdriftInterval_{tn}}$$

RR(n) Drift Rate from RR(n-1) Drift Rate

Node 1 passes $NRRdrift_{t1}(1)$ to Node 2 in rateRatioDrift field (as NRR is RR for Node 1). Node 2 wants drift in terms of it's Local Clock, as this is what is used for error correction (drift rate x correction interval = correction value). In other words, Node 2 receives…

$$NRRdrift_{t1}\,(1) = \frac{mNRRdriftValueA(1) - mNRRdriftValueB(1)}{mNRRdriftInterval_{t1}}$$

…but Node 2 wants…

$$NRRdrift_{t2}\,(1) = \frac{mNRRdriftValueA(1) - mNRRdriftValueB(1)}{mNRRdriftInterval_{t2}}$$

Fortunately…

$$NRR(n) = \frac{Interval_{tn-1}}{Interval_{tn}}$$

RR(n) Drift Rate from RR(n-1) Drift Rate

So…

$$NRR(2) = \frac{Interval_{t1}}{Interval_{t2}}$$ …and therefore… $$mNRRdriftInterval_{t2} = \frac{mNRRdriftInterval_{t1}}{NRR(2)}$$

Which means…

$$NRRdrift_{t2}\,(1) = \frac{mNRRdriftValueA(1) - mNRRdriftValueB(1)}{mNRRdriftInterval_{t1}} \times NRR(2)$$

$$NRRdrift_{t2}\,(1) = NRRdrift_{t1}(1) \times NRR(2)$$

And, more generally…

$$RRdrift_{t2}\,(1) = RRdrift_{t1}(1) \times NRR(2)$$

RR(n) Drift Rate from RR(n-1) Drift Rate

$$RR(n) = RR(n-1) \times NRR(n)$$ **ratio**

$$\frac{dRR(n)}{dt_n} = \left(\frac{dRR(n-1)}{dt_n} \times NRR(n)\right) + \left(\frac{dNRR(n)}{dt_n} \times RR(n-1)\right)$$ **ratio/s**

$$\frac{dRR(n)}{dt_n} = \left(\frac{dRR(n-1)}{dt_{n-1}} \times NRR(n)^2\right) + \left(\frac{dNRR(n)}{dt_n} \times RR(n-1)\right)$$ **ratio/s**

RR(n) Drift Rate from RR(n-1) Drift Rate

$$RR(n) = RR(n-1) \times NRR(n)$$ **ratio**

For small frequency offsets $y_{a \to b}$

$$1 + y_{n \to GM} = (1 + y_{n-1 \to GM}) \times (1 + y_{n \to n-1})$$ **offset**

$$1 + y_{n \to GM} = 1 + y_{n-1 \to GM} + y_{n \to n-1} + (y_{n-1 \to GM} \times y_{n \to n-1})$$ **offset**

$$1 + y_{n \to GM} \approx 1 + y_{n-1 \to GM} + y_{n \to n-1}$$ **offset**

* Small Number x Small Number = Very Small Number

RR(n) Drift Rate from RR(n-1) Drift Rate

$$1 + y_{n \to GM} = 1 + y_{n-1 \to GM} + y_{n \to n-1} + (y_{n-1 \to GM} \times y_{n \to n-1})$$ **offset**

$$\frac{dy_{n \to GM}}{dt_n} \approx \frac{dy_{n-1 \to GM}}{dt_{n-1}} . NRR(n) + \frac{dy_{n \to n-1}}{dt_n} + y_{n \to n-1} \frac{dy_{n-1 \to GM}}{dt_{n-1}} . NRR(n) + y_{n-1 \to GM} \frac{dy_{n \to n-1}}{dt_n}$$ **offset/s**

$$\frac{dy_{n \to GM}}{dt_n} \approx \frac{dy_{n-1 \to GM}}{dt_{n-1}} . NRR(n) \times (1 + y_{n \to n-1}) + \frac{dy_{n \to n-1}}{dt_n} \times (1 + y_{n-1 \to GM})$$ **offset/s**

$$\frac{dy_{n \to GM}}{dt_n} \approx \frac{dy_{n-1 \to GM}}{dt_{n-1}} . NRR(n) + \frac{dy_{n \to n-1}}{dt_n}$$ **offset/s**

* 1 + Small Number ≈ 1

Summary of Calculations

- If offsets are small…

$$\frac{dy_{n\to GM}}{dt_n} \approx \frac{dy_{n-1\to GM}}{dt_{n-1}}.NRR(n) + \frac{dy_{n\to n-1}}{dt_n}$$ **offset/s**

- If offsets are not small…

$$\frac{dRR(n)}{dt_n} = \left(\frac{dRR(n-1)}{dt_{n-1}} \times NRR(n)^2\right) + \left(\frac{dNRR(n)}{dt_n} \times RR(n-1)\right)$$ **ratio/s**

Summary of Proposal

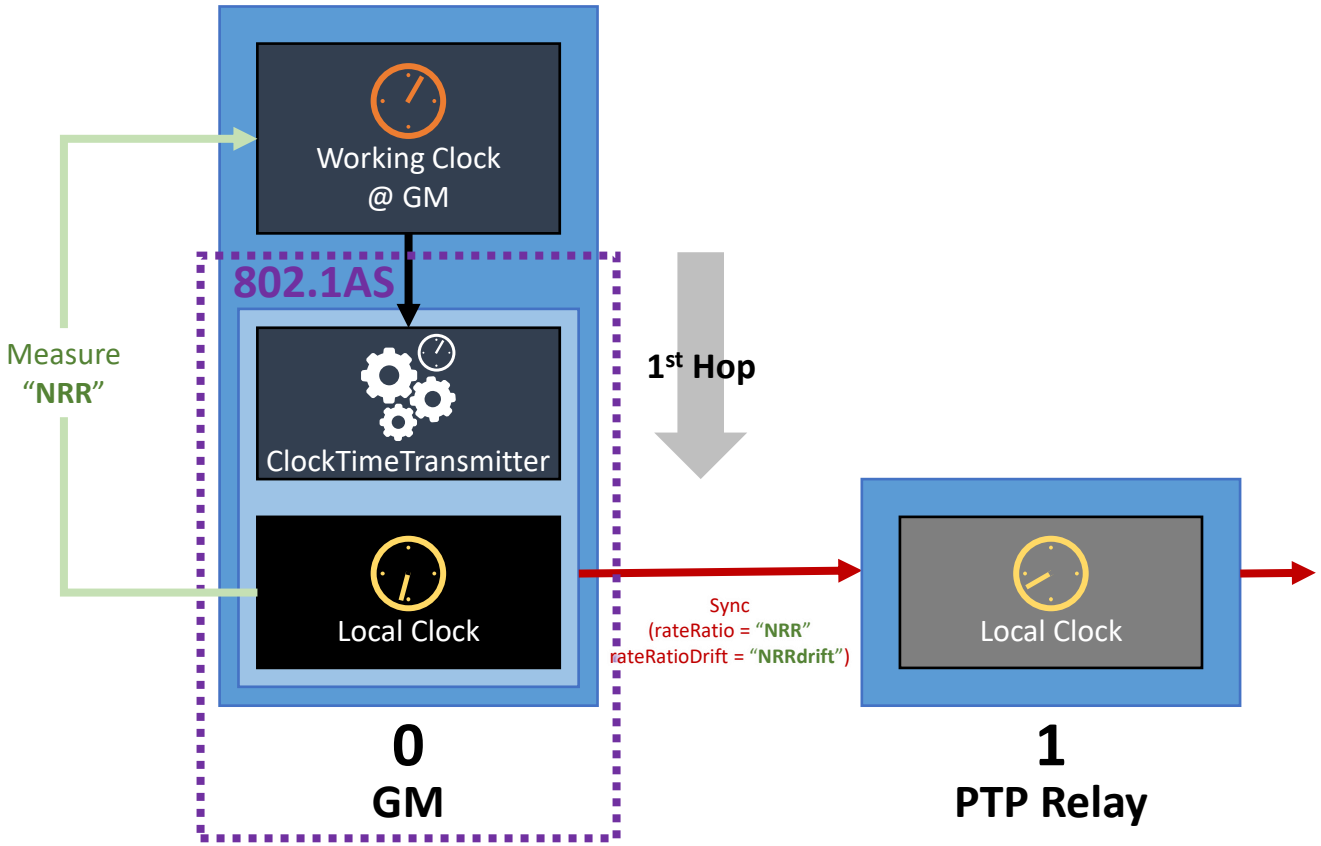
- PTP Relay Instances pass on their estimate of rateRatioDrift in a field in the new Drift_Tracking TLV.
  - Details of field TBD
- PTP Relay Instances and End Stations calculate their local rateRatioDrift using…

$$\frac{dy_{n \rightrightarrows GM}}{dt} \approx \frac{dy_{n-1 \rightrightarrows GM}}{dt} + \frac{dy_{n \rightrightarrows n-1}}{dt}$$

  - Simulations to determine if offsets are small enough for this to be effective; if not, use the more accurate (but expensive) equation.
- PTP Relays use NRR and RR drift estimates for two error-corrected RR values, one for correctionField, one for rateRatio field.
  - correctionField: corrected to half-way between Sync egress at Nodes n-1 and n
  - rateRatio: corrected to Sync egress at Node n

Implications for Grandmasters, End Stations & Normative Requirements

Implications for Grandmasters

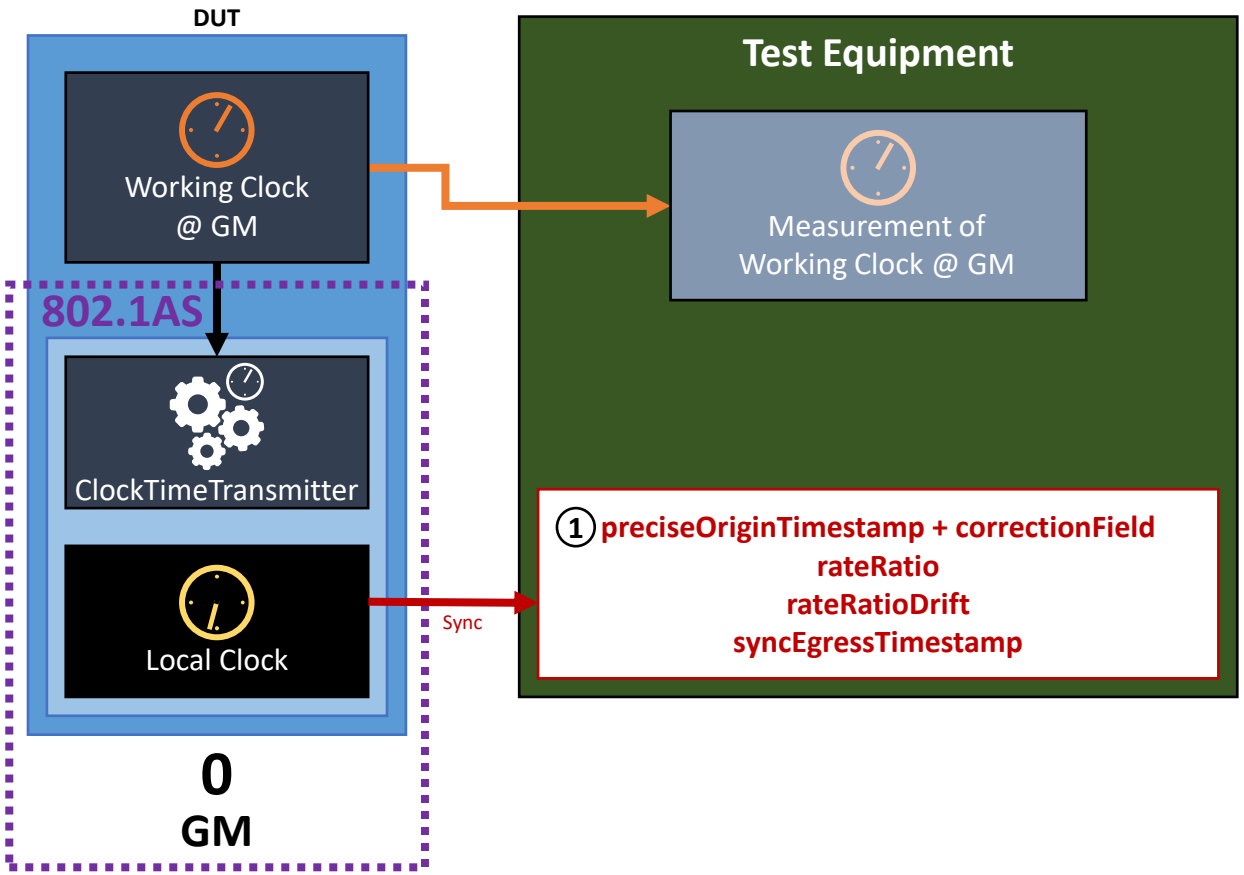


- If Working Clock @ GM (or other domain ClockSource) is not the same as Local Clock @ GM, then GM must track drift between the two and send in Drift_Tracking TLV to Node 1
  - GM effectively contains 1st hop

Implications for End Stations

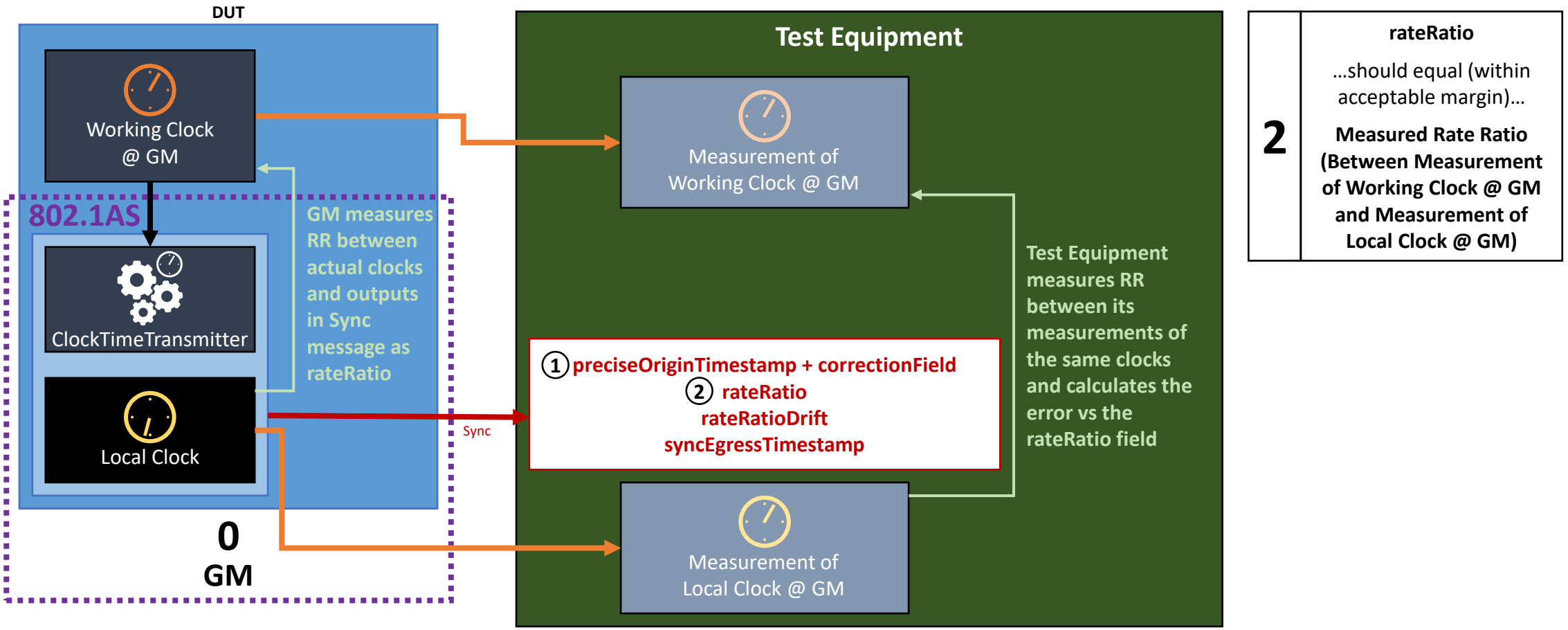
- ClockTimeReceiver at End Stations should calculate RR taking account of RR Drift and carrying out appropriate error correction until arrival of next Sync message.
  - Question: implications for filtering output when generating Working Clock @ End Station (or other time domain).

Device-Level Error Generation Measurement at Grandmaster



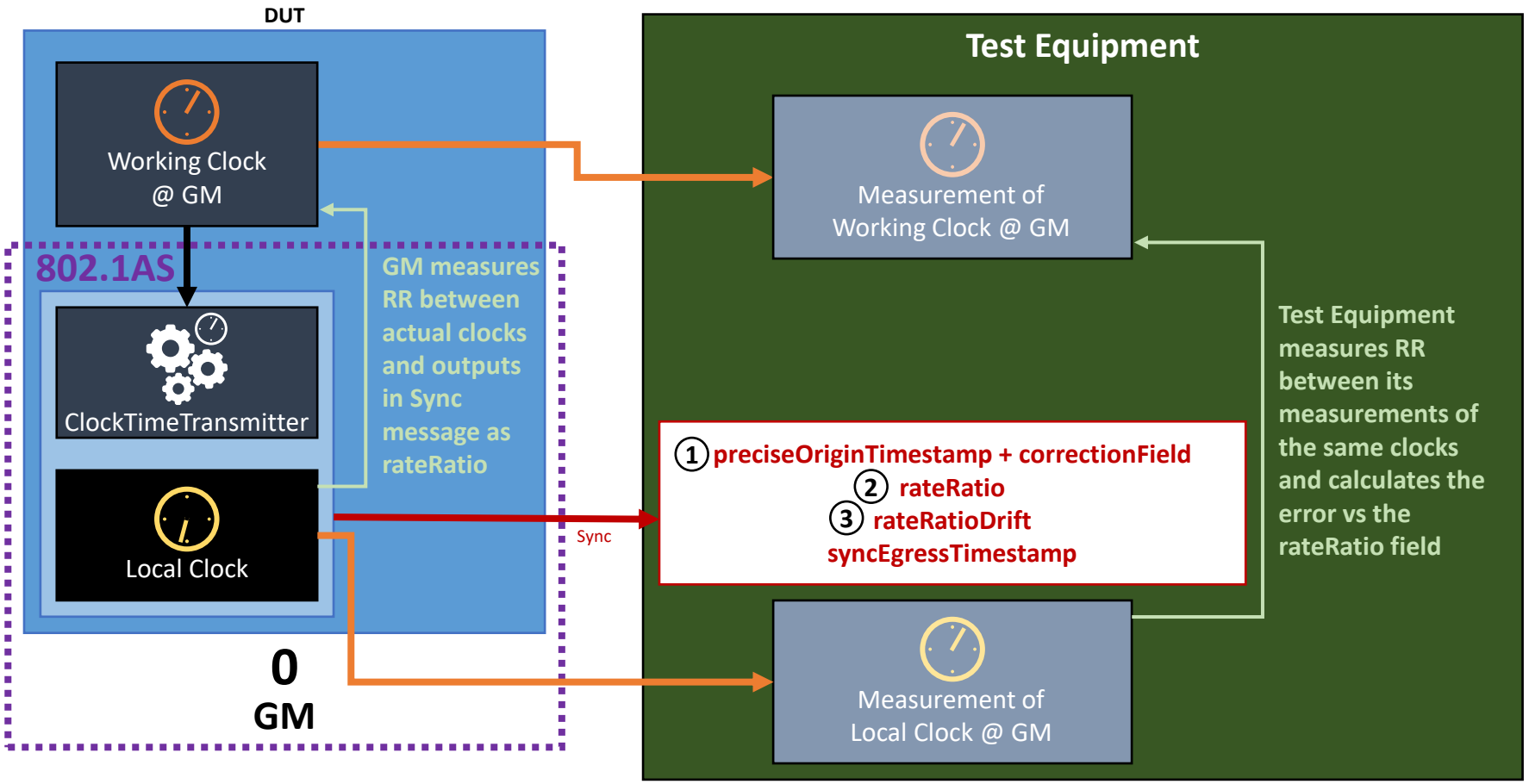
1	**Precise Origin Timestamp + Correction Field** …should equal (within acceptable margin)… **Measured Egress Time of Sync Message (measured against Measurement of Working Clock @ GM)**

Device-Level Error Generation Measurement at Grandmaster



2	**rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio (Between Measurement of Working Clock @ GM and Measurement of Local Clock @ GM)**

Device-Level Error Generation Measurement at Grandmaster

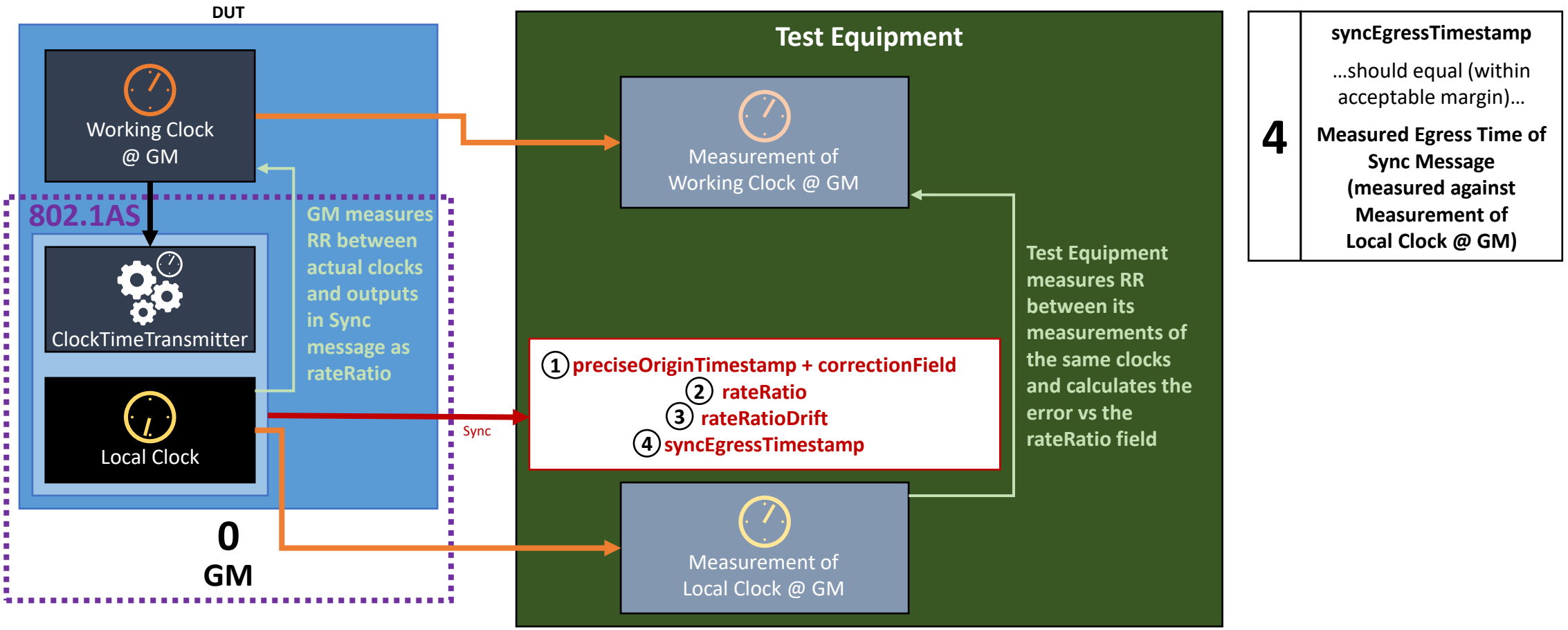


3	**rateRatioDrift** NEW …should equal (within acceptable margin)… **Measured Rate Ratio Drift (Between Measurement of Working Clock @ GM and Measurement of Local Clock @ GM)**

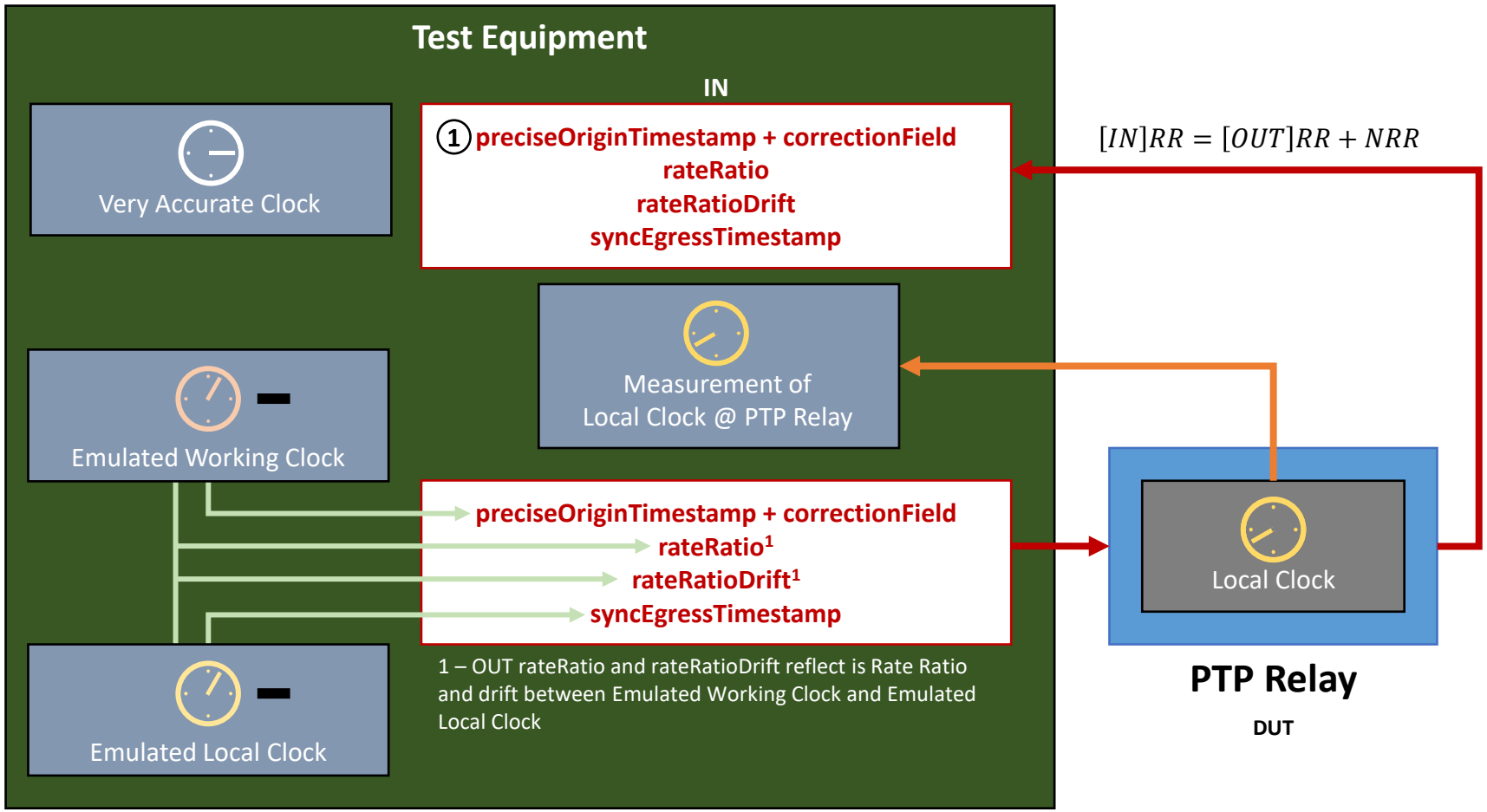
If Working Clock and Local Clock are the same, rateRatioDrift will always be zero.

Probably not possible for test equipment to cause controllable drift of either Working Clock @ GM or Local Clock @GM unless the former is under external control…in which case, should it be tested? (Recommend discussing as part of External Control topic.)

Device-Level Error Generation Measurement at Grandmaster

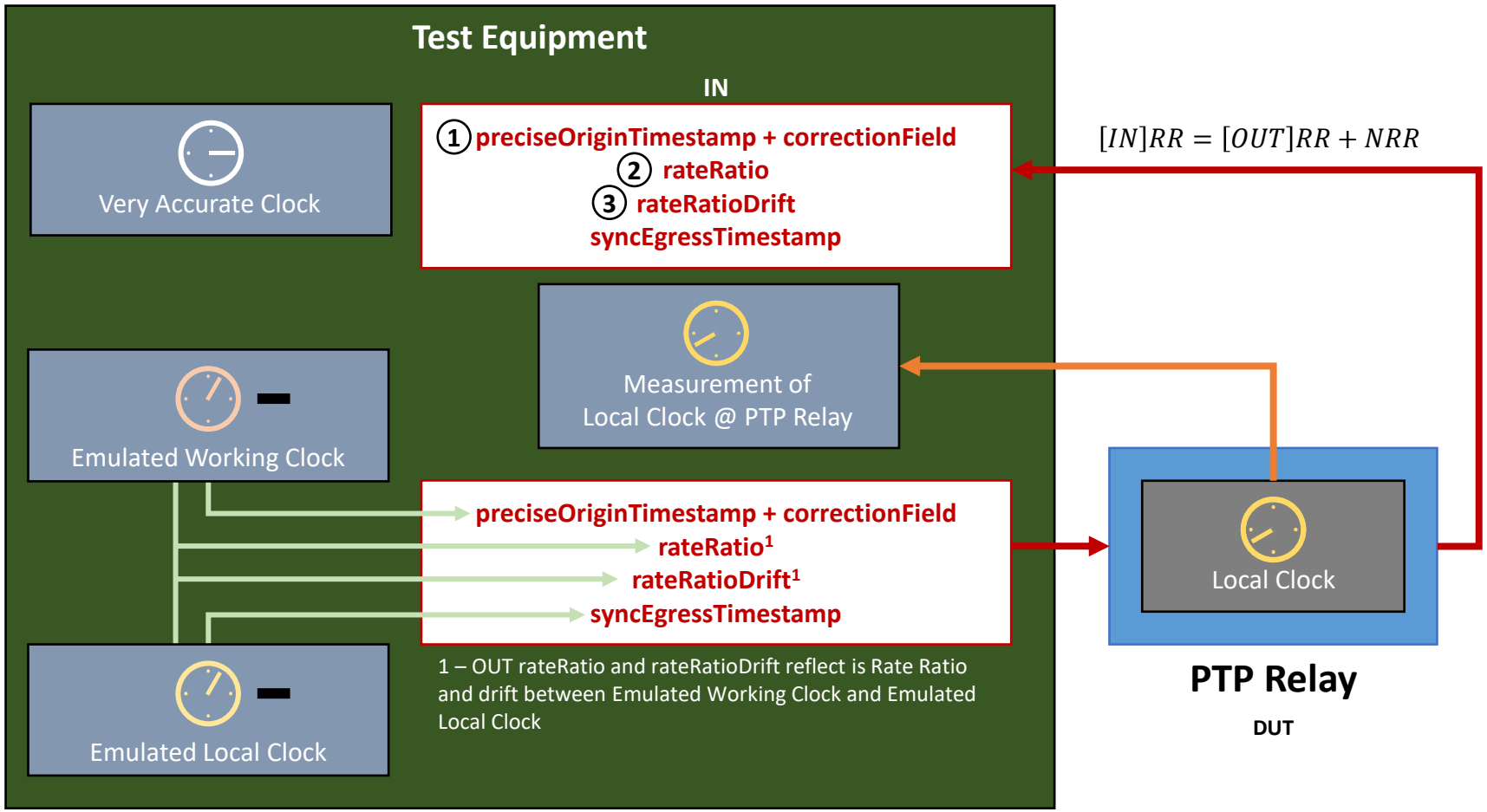


Device-Level Error Generation Measurement at PTP Relay



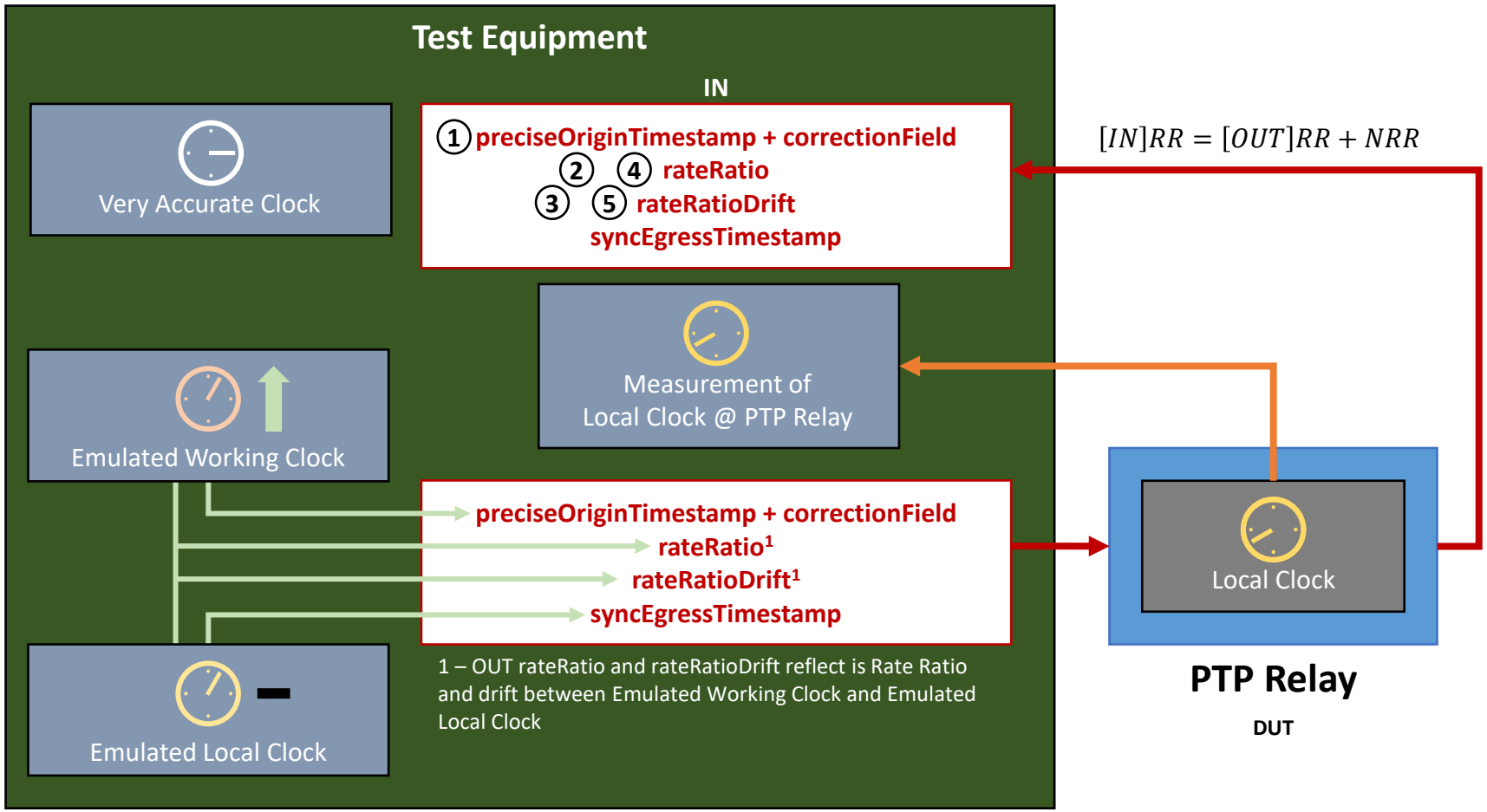
	Emulated Working Clock: Stable	Emulated Local Clock: Stable
1	**[IN] preciseOriginTimestamp + correctionField** …should equal (within acceptable margin)… **Measured Egress Time of [IN] Sync Message (measured against Emulated Working Clock)**	

Device-Level Error Generation Measurement at PTP Relay



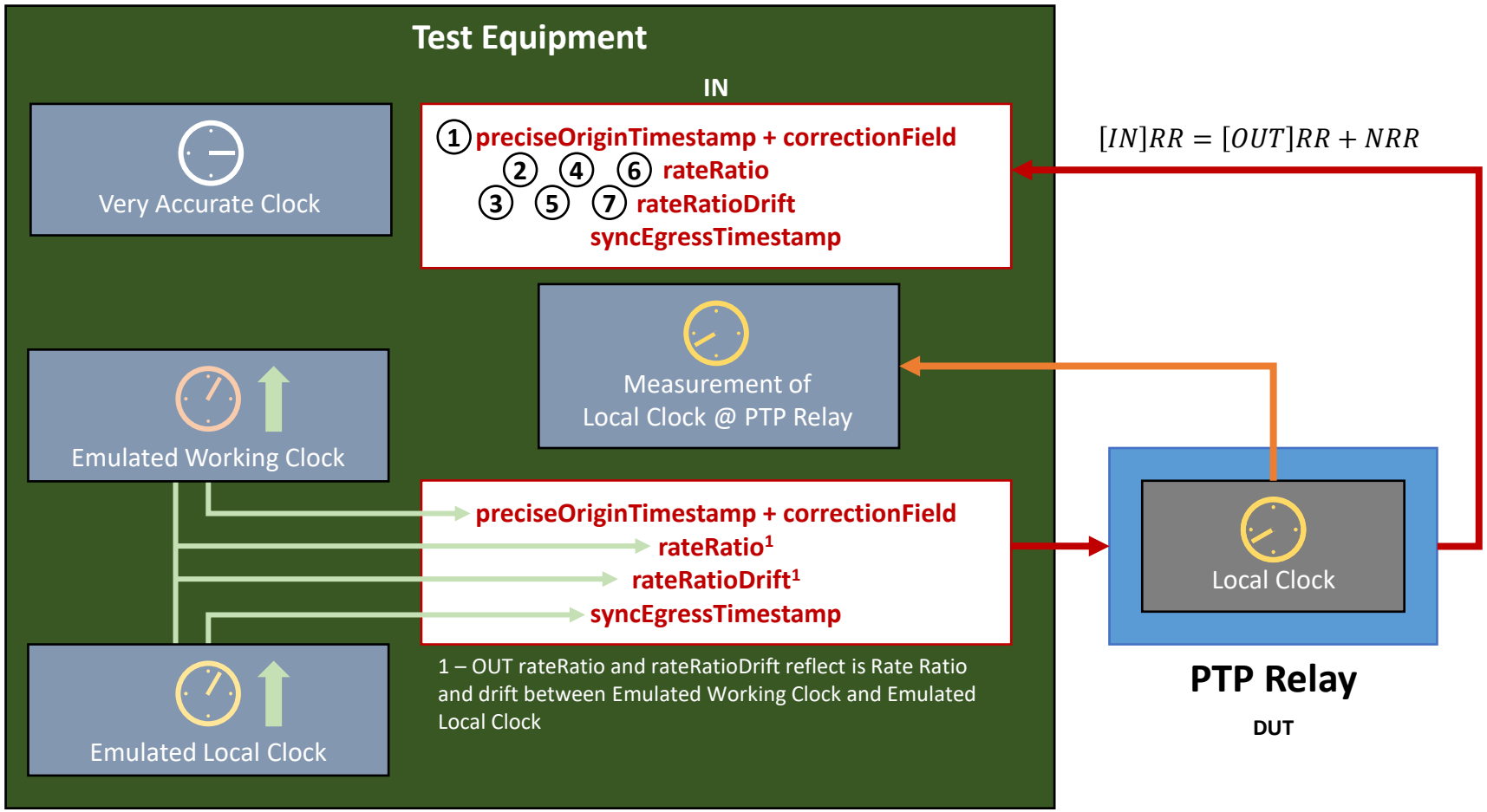
	Emulated Working Clock: Stable	Emulated Local Clock: Stable
2	**[IN] rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio ([OUT] rateRatio plus Rate Ratio between Emulated Local Clock and Measurement of Local Clock @ PTP Relay)**	
3	**[IN] rateRatioDrift** …should equal (within acceptable margin)… **Measured Rate Ratio drift ([OUT] rateRatioDrift plus Rate Ratio between Emulated Local Clock and Measurement of Local Clock @ PTP Relay)**	NEW

Device-Level Error Generation Measurement at PTP Relay



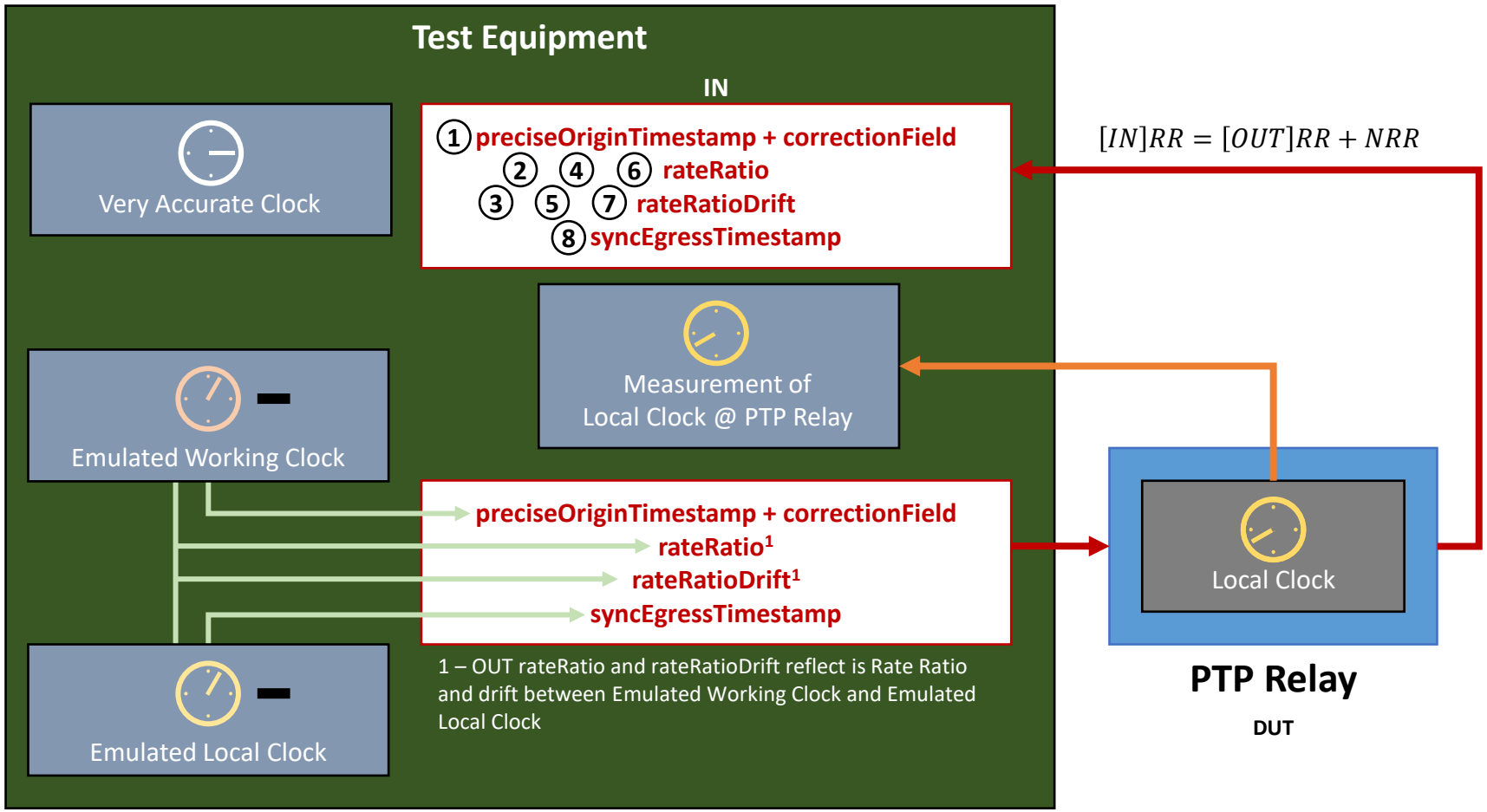
	Emulated Working Clock: Offset Increasing	Emulated Local Clock: Stable
4	**[IN] rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio ([OUT] rateRatio plus Rate Ratio between Emulated Local Clock and Measurement of Local Clock @ PTP Relay)**	
5	**[IN] rateRatioDrift** NEW …should equal (within acceptable margin)… **Measured Rate Ratio drift ([OUT] rateRatioDrift plus Rate Ratio between Emulated Local Clock and Measurement of Local Clock @ PTP Relay)**	

Device-Level Error Generation Measurement at PTP Relay



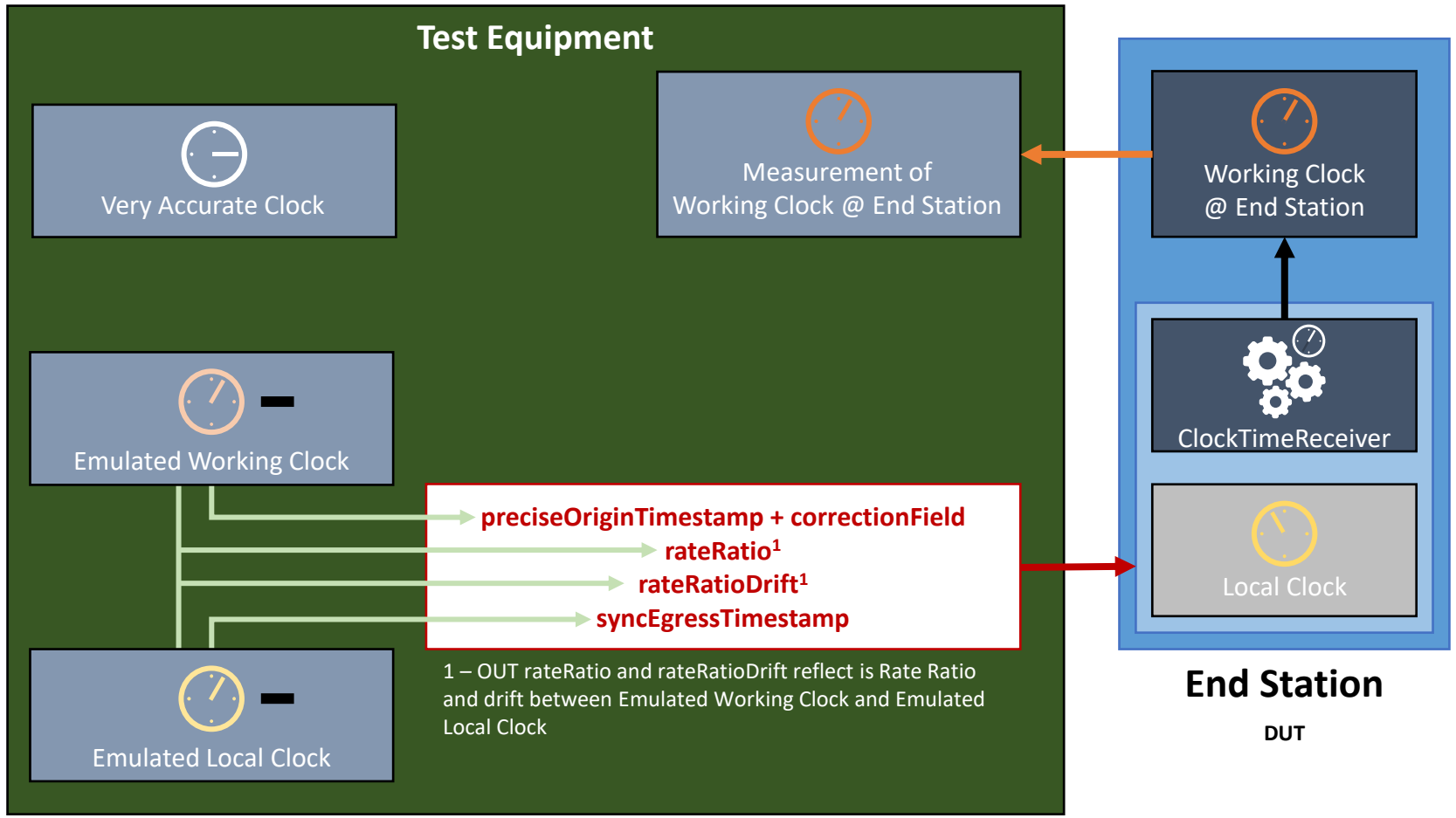
	Emulated Working Clock: Offset Increasing	Emulated Local Clock: Offset Increasing
6	**[IN] rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio ([OUT] rateRatio plus Rate Ratio between Emulated Local Clock and Measurement of Local Clock @ PTP Relay)**	
7	**[IN] rateRatioDrift** …should equal (within acceptable margin)… **Measured Rate Ratio drift ([OUT] rateRatioDrift plus Rate Ratio between Emulated Local Clock and Measurement of Local Clock @ PTP Relay)**	NEW

Device-Level Error Generation Measurement at PTP Relay



Emulated Working Clock: Stable	Emulated Local Clock: Stable
8	**[IN]syncEgressTimestamp** …should equal (within acceptable margin)… **Measured Egress Time of Sync Message (measured against Measurement of Local Clock @ GM)**

Device-Level Error Generation Measurement at End Station



Emulated Working Clock: Stable	Emulated Local Clock: Stable
1	**Measurement of Working Clock @ End Station** …should equal (within acceptable margin)… **Emulated Working Clock**

Emulated Working Clock: Increasing	Emulated Local Clock: Stable
2	**Measurement of Working Clock @ End Station** …should equal (within acceptable margin)… **Emulated Working Clock**

Emulated Working Clock: Increasing	Emulated Local Clock: Increasing
3	**Measurement of Working Clock @ End Station** …should equal (within acceptable margin)… **Emulated Working Clock**

Next Steps

Next Steps

- Revise 802.1ASdm contribution regarding TLV to include rateRatioDrift field
- Monte Carlo & Time Series simulations of complete 60802 approach including NRR & RR drift tracking and error compensation
- Error budgeting for normative requirements, including rateRatioDrift
  - Also, potentially relaxed parameters for Global Time
- Contribution for normative requirements of next version of specification
- Contribution for informative annex for next version of specification
- Real-world testing to confirm simulation results

802.1ASdm PAR

2.1 Project Title: Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications
Amendment: Hot Standby and Clock Drift Error Tracking
Change to Title: Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications Amendment: Hot Standby and Clock Drift Error Reduction

Change to scope of the project: This amendment specifies protocols, procedures, and managed objects for hot standby without use of the Best Master Clock Algorithm (BMCA), for time-aware systems, including:
- A function that transforms the synchronized times of two generalized Precision Time Protocol (gPTP) domains into one synchronized time for use by applications;
- A function that directs the synchronized time of one gPTP domain into a different gPTP domain; and
- Mechanisms that determine whether a gPTP domain has sufficient quality to be used for hot standby.

This amendment specifies a Type-Length-Value (TLV) that allows more accurate Neighbor Rate Ratio calculation and more accurate tracking of clock frequency drift.
This amendment also addresses errors and omissions in the description of existing functionality.

802.1ASdm CSD

Title

P802.1ASdm Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications Amendment: Hot Standby and Clock Drift Error Reduction

1.2.1 Broad market potential

b) The need for this project is driven by requirements of industrial automation applications, as part of ongoing work on IEC/IEEE 60802 Time-Sensitive Networking Profile for Industrial Automation. These requirements include 1 µs time synchronization accuracy over 64 network hops, with a goal of 100 network hops if possible, while using existing silicon and low-cost crystal oscillators, i.e., not, for example, temperature compensated crystal oscillators. A new TLV will enable calculation of Neighbor Rate Ratio using the Sync mechanism and provide additional information that is used when tracking Rate Ratio to compensate for time synchronization errors due to clock frequency drift, both of which are required to achieve the time synchronization goal. The IEC/IEEE 60802 project applies to multiple industrial automation applications, and multiple industrial automation vendors and users are participating in its development.

1.2.3 Distinct Identity

IEEE Std 802.1AS specifies the transport of synchronized time; however, it does not provide for hot-standby in the transport. There is no other IEEE standard or project that defines hot-standby for IEEE Std 802.1AS. There is no other IEEE standard or project that is defining a solution for IEEE Std 802.1AS to enable calculation of Neighbor Rate Ratio using the Sync mechanism or provide the additional information used when tracking Rate Ratio.

1.2.4 Technical Feasibility

a) Hot-standby techniques have been feasibly used in existing standards for fieldbus applications (e.g., IEC 61784-2). The improvements enabled by the new TLV will not alter the already demonstrated feasibility of 802.1AS systems.

b) The proposed standard will use hot-standby techniques for which the technology has been proven. In other contexts, hot-standby techniques can be referred to as a method to achieve seamless redundancy, high availability, resiliency, and protection for time synchronization. See item a) for references. The effectiveness of the techniques enabled by a new TLV at reducing dynamic time error (dTE) have been demonstrated by extensive simulations.

Summary

Summary

- Rate Ratio Drift should be calculated as an accumulation of Neighbor Rate Ratio Drift, similar to how Rate Ratio is calculated.
  - This is more accurate and more responsive than the alternatives.
- 802.1ASdm contribution should be revised to support rateRatioDrift field in new Drift_Tracking TLV
  - No changes to PAR or CSD are necessary.
- Some new normative requirements will be necessary. Simulations will provide values.
- Real-world testing will confirm approach and normative requirements (🤞).

Thank you!

Backup