60802 Time Sync - Monte Carlo and Time Series Simulation Configuration Including NRR and RR Drift Tracking & Error Compensation

David McCall – Intel Corporation

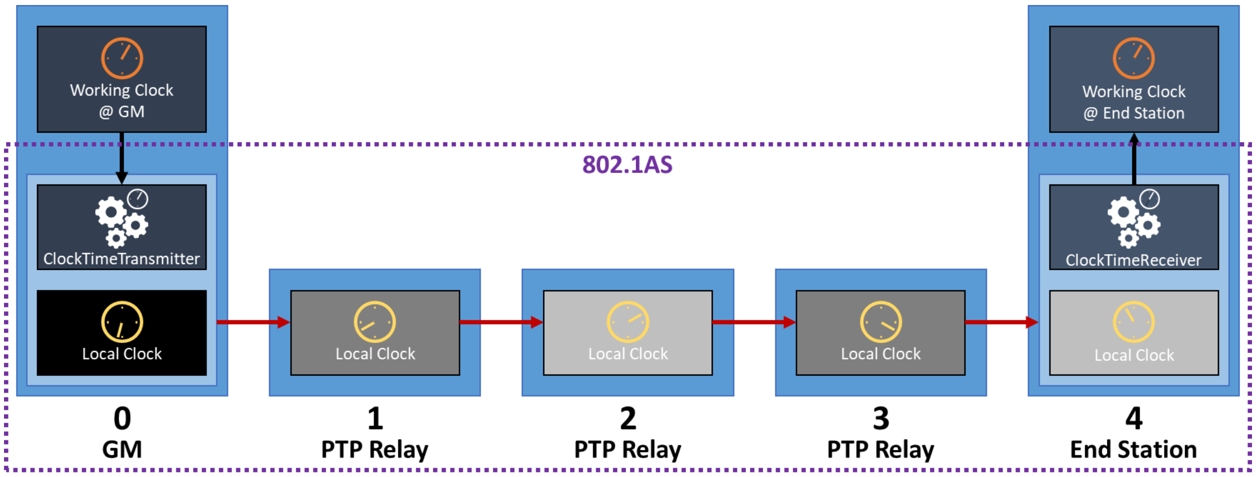
Version 3
June 2023

1 Introduction

As a result of discussions in the 60802 Time Sync ad hoc group, consensus was achieved on a preferred approach to achieving the group's goal of 1 µs time synchronisation accuracy over 100 network hops. It is planned to conduct both Time Series and Monte Carlo simulations to validate that this approach can achieve the goal and provide data on appropriate normative requirements. This document describes the input parameters, probability distributions (for random elements) and other configuration parameters for these simulations, in part to ensure clarity and consistency between the different simulations. In particular it describes the algorithms used for NRR and RR drift tracking and error compensation.

2 Scope of Simulations

Using this 4-hop example:



The Monte Carlo simulation covers only the portion of the diagram in the "802.11AS" box. At the GM it effectively assumes that the Working Clock @ GM is the same as the Local Clock @ GM. At the End Station it only models the (unfiltered) output of the ClockTimeReceiver.

The Time Series simulation also assumes that the Working Clock @ GM is the same as Local Clock @ GM.

The Time Series simulation also models filtering at the ClockTimeReceiver. It produces both filtered and unfiltered results. [Note: a future version of this document, or a new document will include details of the filtering for Time Series simulations.]

Future simulations may add the ability to have separate Working Clock @ GM and Local Clock @ GM.

3 Protocol Parameters & Probability Distributions

Note that use of the new Drift_Tracking TLV and the calculation of NRR from Sync message timestamps that it enables, as opposed to Pdelay_Resp message time stamps, is assumed. Thus Pdelay and Pdelay_Resp messages are not used for calculation of NRR, but are still used for calculation of meanLinkDelay.

All of the intervals that are randomly generated are independent of each other.

Parameter	Value / Probability Distribution
Sync Interval at GM – Nominal	125 ms
Interval between Sync messages at GM - Simulated	Uniform Distribution Minimum: 119 ms Maximum: 131 ms
Interval between Sync message and related Follow_Up message – Simulated	Not simulated. Assumed to arrive in time to not affect Residence Time.
Residence Time – Simulated	Normal/Gaussian Distribution Mean: 5 ms Standard Deviation: 1.8 ms Truncated at… Minimum: 1 ms Maximum 15 ms …with values below minimum changed to 1 ms; values above maximum changed to 15 ms
Pdelay Interval – Nominal	125 ms
Interval between Pdelay messages – Simulated	Uniform Distribution Minimum: 112.5 ms (Nominal x 0.9) Maximum: 162. ms (Nominal x 1.3)
Interval between Pdelay message and related Pdelay_Resp message - Nominal	10 ms
Interval between Pdelay message and related Pdelay_Resp message - Simulated	Uniform Distribution Minimum: 9 ms (Nominal x 0.9) Maximum: 13 ms ((Nominal x 0.9)
Interval between Pdelay_Resp message and related Pdelay_Resp_Follow_Up message - Simulated	Not simulated. Assumed to arrive in time to not affect calculation of meanLinkDelay prior to next Sync message transmission.

4 Timestamp Error Modelling

Every timestamp is modelled to have two sources of error.

1. Timestamp Granularity Error (TSGE): Related to the nominal frequency of the node's local clock. Timestamps are modelled as being generated on the clock "tick" following an event. Thus the error may be from a minimum of 0 ns, if the clock tick occurs at exactly the time of the event, up to a maximum of $\frac{10^9}{f}$ ns, where f is the clock frequency, if the clock tick occurs just before the event.
2. Dynamic Time Stamp Error (DTSE): A representation of all other errors due to variations and inaccuracies of measurement. By design this should not only be minimised, but also average as close as possible to zero, i.e. positive and negative errors on individual timestamps should average, over many measurements, to as close to zero as possible.

All randomly generated timestamp errors are independent of each other.

Error	Model
Timestamp Granularity Error (Assumes 125 MHz clock)	Uniform Distribution Minimum: 0 ns Maximum: 8 ns
Dynamic Timestamp Error	Uniform Distribution Minimum: -6 ns Maximum: +6 ns

5 Clock Drift Modelling

Clock drift is modelled via a combination of a repeating temperature cycle and model of the relationship between temperature and fractional frequency offset relative to the nominal frequency.

The temperature ramp has four phases.

- Section A – Ramp from minimum to maximum temperature with a “quarter-sinusoidal” curve, i.e. starts steep (SIN at 0°), ends flat (SIN at 90°).
- Section B – Level at maximum temperature.
- Section C – Ramp from maximum to minimum temperature with a “quarter-sinusoidal” curve, i.e. starts steep (SIN at 180°), ends flat (SIN at 270°).
- Section D – Level at minimum temperature.

The relationship between temperature and fractional frequency offset relative to the nominal frequency is modelled as a cubic equation, designed to match real-world measured data.

Here are the relevant equations and parameters.

Parameter	Value	Unit
tempMax	85	°C
tempMin	-20	°C
tempRampPeriod	125	s
tempHold	30	s

$$tempCyclePeriod = 2 \times (\boldsymbol{tempRampPeriod} + \boldsymbol{tempHold}) = 310 \text{ s}$$

$$tempRange = \boldsymbol{tempMax} - \boldsymbol{tempMin} = 105 \text{ °C}$$

$$\tau = \frac{\pi}{\boldsymbol{tempRampPeriod} \times 2} = 0.0125663$$

$$sectionAend = \boldsymbol{tempRampPeriod} = 125 \text{ s}$$

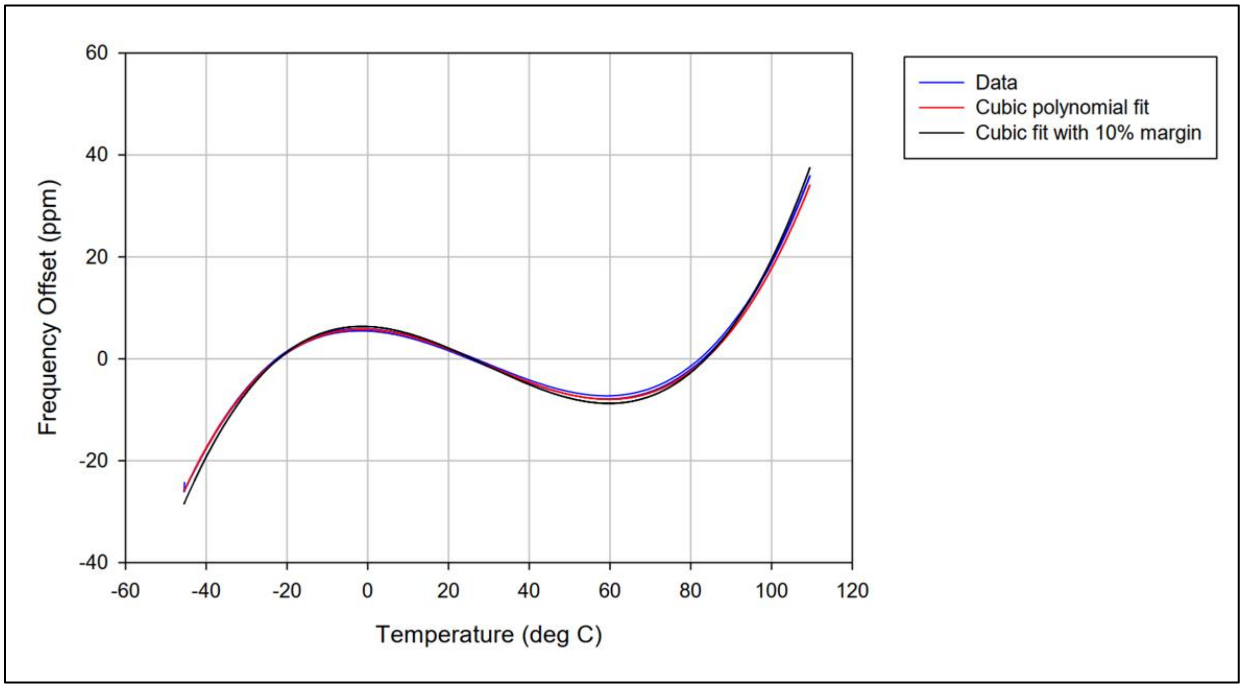
$$sectionBend = sectionAend + \boldsymbol{tempHold} = 155 \text{ s}$$

$$sectionCend = sectionBend + sectionAend = 280 \text{ s}$$

Equation for Frequency Offset (ppm) vs. Temperature:

$$freqOffset = \boldsymbol{a}.tempXO^3 + \boldsymbol{b}.tempXO^2 + \boldsymbol{c}.tempXO + \boldsymbol{d}$$

Cubic Constant	Value
a	0.00012
b	-0.01005
c	-0.0305
d	5.73845



From Geoff Garner, "Phase and Frequency Offset, and Frequency Drift Rate Time History Plots Based on New Frequency Stability Data", contribution to IEC/IEEE 60802, March 2021

For the Time Series simulation, at the start of each replication, each node is assigned a random position on the temperature ramp, independently of any other node, and then progresses along the cyclic ramp as the simulation executes.

For the Monte Carlo simulation, each node is assigned a random position on the temperature ramp for each run. This is the position when the node transmits its Sync message to the next node (or, for the End Station, receives it from the previous node). For each run, other calculations may be made relative to that position, but the position is assigned independently of any other run or any other node.

The probability distribution for assigned position for each node is the same for both simulations.

Value	Distribution
Assigned Position on Along Temperature Ramp (***tOffset*** for Time Series or ***t*** for Monte Carlo)	Uniform Distribution Minimum: 0 s Maximum: 310 ns

The temperature ($tempXO$), temperature rate of change ($tempRoC$), and clock frequency drift rate ($clockDrift$) at a particular node can be calculated from…

- $\boldsymbol{t}$ for Monte Carlo
- ($\boldsymbol{t} + \boldsymbol{tOffset}$) % $tempCyclePeriod$ for Time Series where t is the simulated time and % is the modulo operation.

Using $\boldsymbol{t}$ for the input value, i.e. the Monte Carlo value, the equations are…

$$if(0 \leq t < sectionAend)$$

$$tempXO = \boldsymbol{tempMin} + \boldsymbol{tempRange}.\ \sin(\tau.t)$$

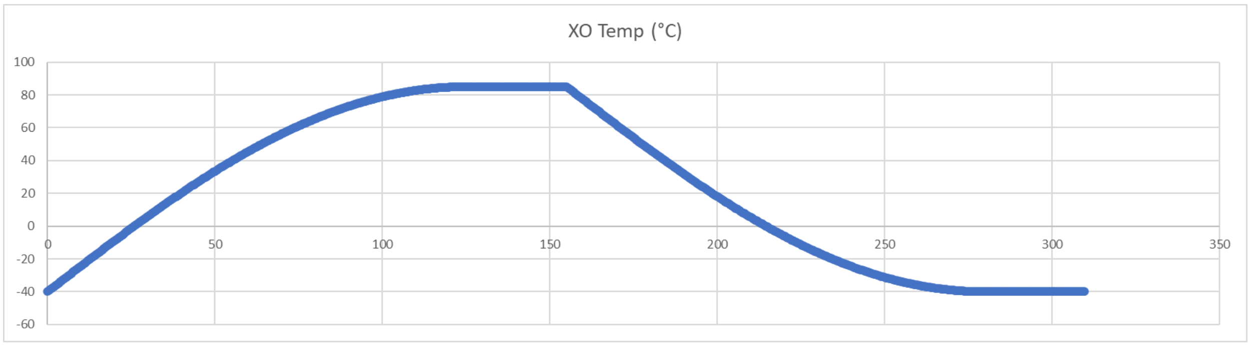
$$tempRoC = \tau.\boldsymbol{tempRange}.cos(\tau.t)$$

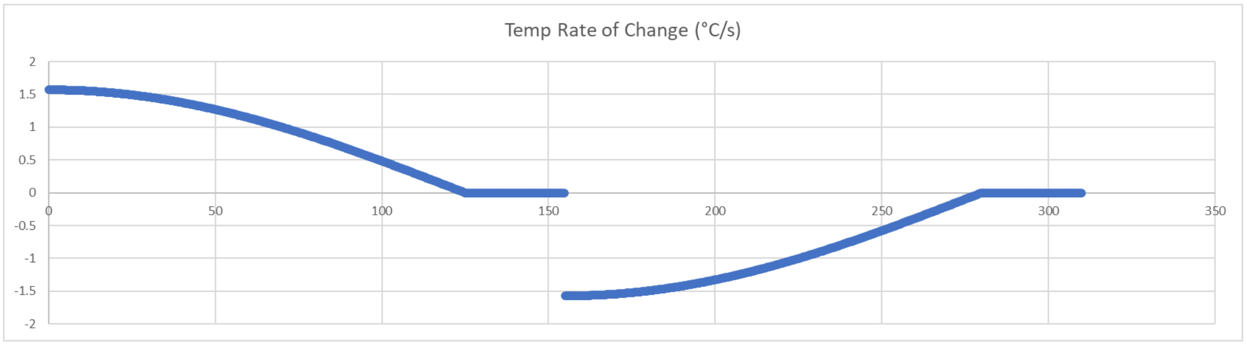
$$clockDrift = (3.\boldsymbol{a}.tempXO^2 + 2.\boldsymbol{b}.tempXO + \boldsymbol{c}) \times \left(\tau.\boldsymbol{tempRange}.cos(\tau.t)\right)$$

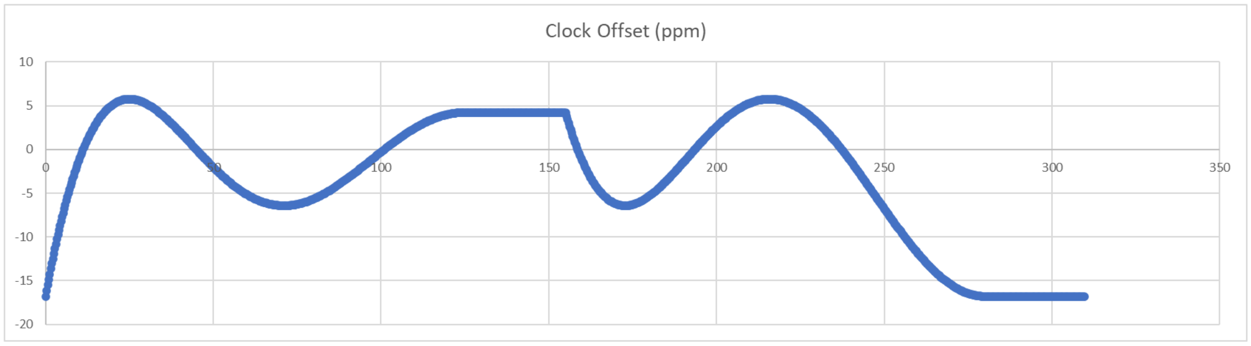
$$if(sectionAend \leq t < sectionBend)$$

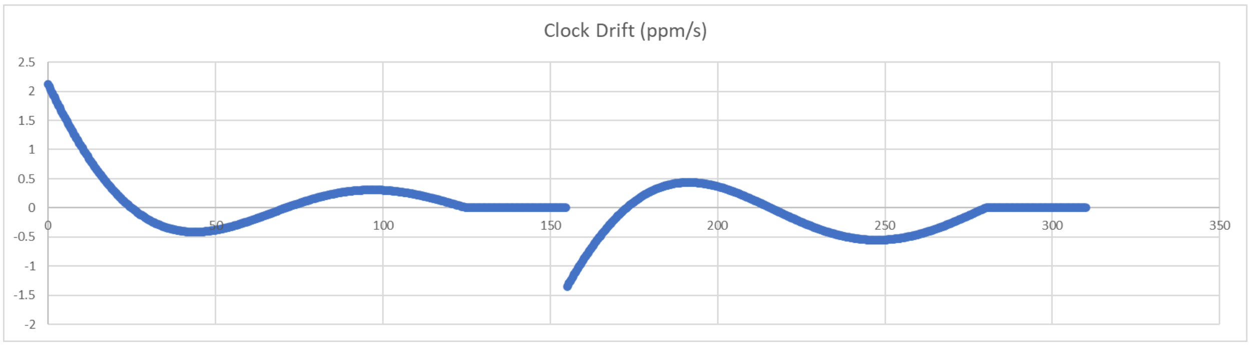
```
    $tempXO = \boldsymbol{tempMax}$
    $tempRoC = 0$
    $clockDrift = 0$
$if\,(sectionBend \le t < sectionCend)$
    $tempXO = \boldsymbol{tempMax} - \boldsymbol{tempRange}.\ \sin\left(\tau.(t - sectionBend)\right)$
    $tempRoC = -\tau.\boldsymbol{tempRange}.cos\left(\tau.(t - sectionBend)\right)$
    $clockDrift = -(3.\boldsymbol{a}.tempXO^2 + 2.\boldsymbol{b}.tempXO + \boldsymbol{c}) \times \left(\tau.\boldsymbol{tempRange}.cos\left(\tau.(t - sectionBend)\right)\right)$
$if\,(sectionCend \le t)$
    $tempXO = \boldsymbol{tempMin}$
    $tempRoC = 0$
    $clockDrift = 0$
```

These equations produce the following behaviors…









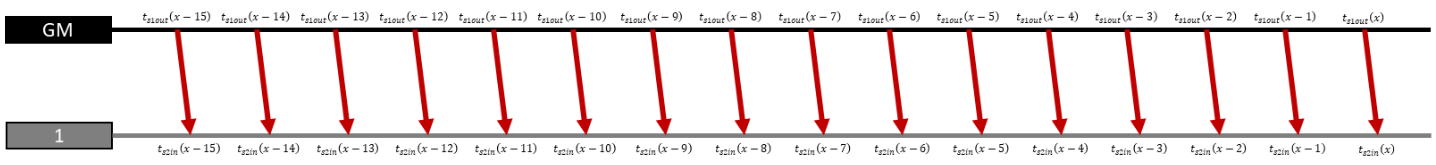
6 NRR Drift Tracking

NRR Drift Tracking and Error Correction is carried out for each network hop, i.e. at every node other than the Grandmaster. It is based on pairs of timestamps with each pair associated with a Sync message transmitted from the previous node (n-1) to the current node (n)

- t_{s1outP} – Timestamp of the Sync message egress from the **previous** node (n-1), timestamped by that node's Local Clock. Unit: **ns**.
- t_{s2in} – Timestamp of the Sync message ingress to the current node (n), timestamped by that node's Local Clock. Unit: **ns**.

All timestamps are affected by the relevant Timestamp Errors.

The algorithm uses information from the 16 most recent sync messages



However, a node keeps track of (i.e. remember) the 5 most recent pairs of timestamps from the most recent (x) to the 5th most recent (x-4) Sync message.

On arrival of a new timestamp pair (x), a node executes a NRR calculation…

$$mNRRcalc(x) = \left(\frac{t_{1outP}(x)-t_{1outP}(x-4)}{t_{2in}(x)-t_{2in}(x-4)} - 1\right) \times 10^6 \qquad \textbf{ppm}$$

…with an associated effective measurement point…

$$mNRRcalcT(x) = \frac{t_{2in}(x)+t_{2in}(x-4)}{2} \qquad \textbf{ns}$$

A node keeps track of the 12 most recent NRR calculations and effective measurement points from the most recent (x) to the 12th most recent (x-11).

After of a new most-recent NNR calculation, a node calculates an NRR drift rate…

$$NRRaverageA = \sum_{i=x-3}^{x} \frac{mNRRcalc(i)}{4} \qquad \textbf{ppm}$$

$$NRRaverageB = \sum_{i=x-11}^{x-8} \frac{mNRRcalc(i)}{4} \qquad \textbf{ppm}$$

$$NRRdriftInterval = \sum_{i=x-3}^{x} \frac{mNRRcalcT(i)}{4} - \sum_{i=x-11}^{x-8} \frac{mNRRcalcT(i)}{4} \qquad \textbf{ns}$$

$$NRRdriftRate(n) = \left(\frac{NRRaverageA-NRRaverageB}{NRRdriftInterval}\right) \times 10^9 \qquad \textbf{ppm/s}$$

…where *NRRdriftRate(n)* is the NRR drift rate for the current Node n.

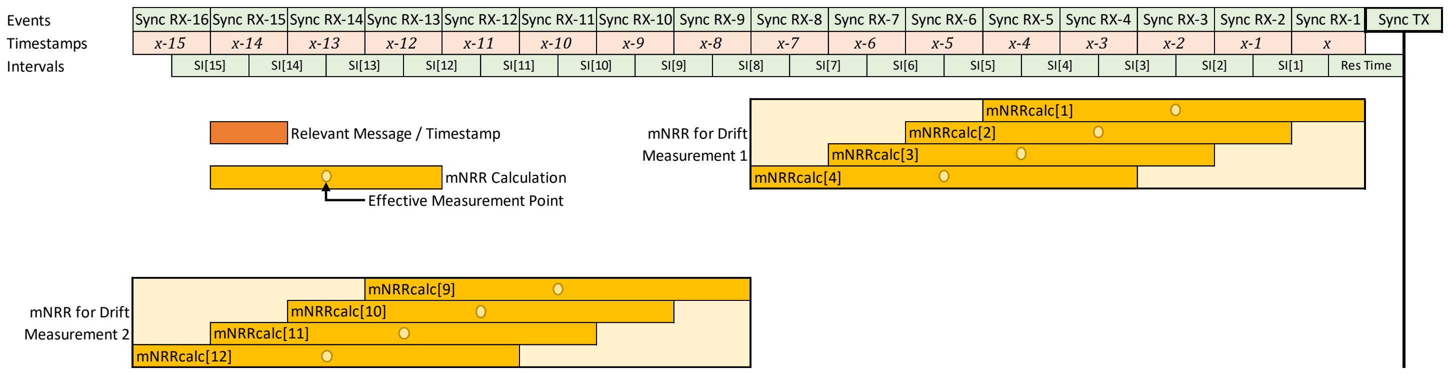
The node then calculates an error-corrected NRR value.

$$For\ i = x\ to\ (x-3)$$

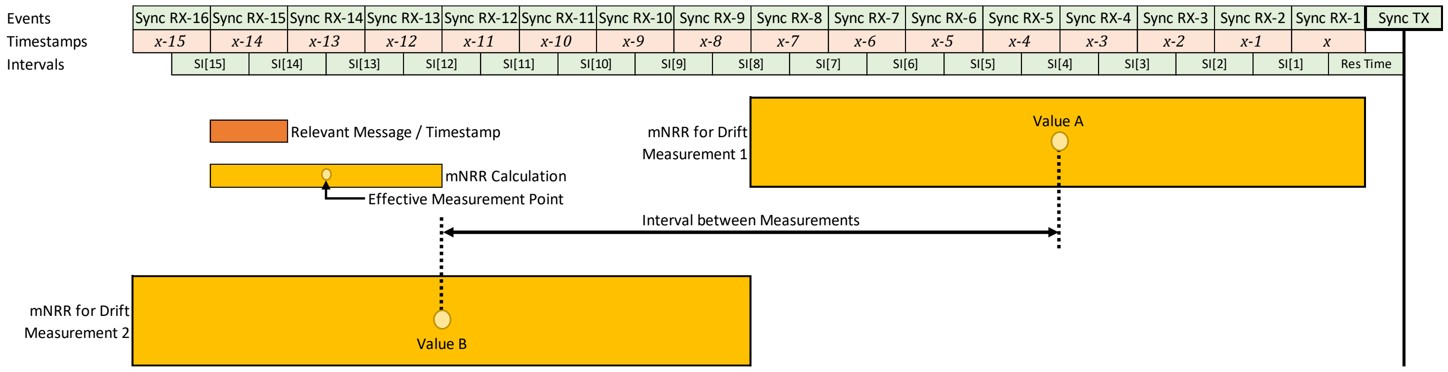
$$mNRRcorrected(i) = mNRRcalc(i) + \left(mNRRdriftRate(n) \times \frac{(t_{s2in}(x)-mNRRcalcT(i))}{10^9}\right) \qquad \textbf{ppm}$$

$$mNRRc = \sum_{i=x-3}^{x} \frac{mNRRcorrected(i)}{4} \qquad \textbf{ppm}$$

The result is a drift estimate from two sets of averaged measurements.

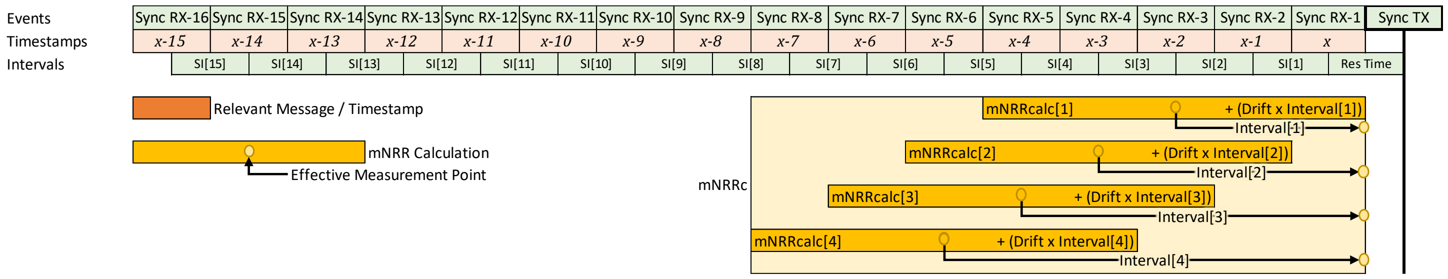


Events
Timestamps
Intervals
Sync RX-16 Sync RX-15 Sync RX-14 Sync RX-13 Sync RX-12 Sync RX-11 Sync RX-10 Sync RX-9 Sync RX-8 Sync RX-7 Sync RX-6 Sync RX-5 Sync RX-4 Sync RX-3 Sync RX-2 Sync RX-1 Sync TX
x-15 x-14 x-13 x-12 x-11 x-10 x-9 x-8 x-7 x-6 x-5 x-4 x-3 x-2 x-1 x
SI[15] SI[14] SI[13] SI[12] SI[11] SI[10] SI[9] SI[8] SI[7] SI[6] SI[5] SI[4] SI[3] SI[2] SI[1] Res Time
Relevant Message / Timestamp
mNRR Calculation
Effective Measurement Point
mNRR for Drift Measurement 1
mNRRcalc[1]
mNRRcalc[2]
mNRRcalc[3]
mNRRcalc[4]
mNRR for Drift Measurement 2
mNRRcalc[9]
mNRRcalc[10]
mNRRcalc[11]
mNRRcalc[12]

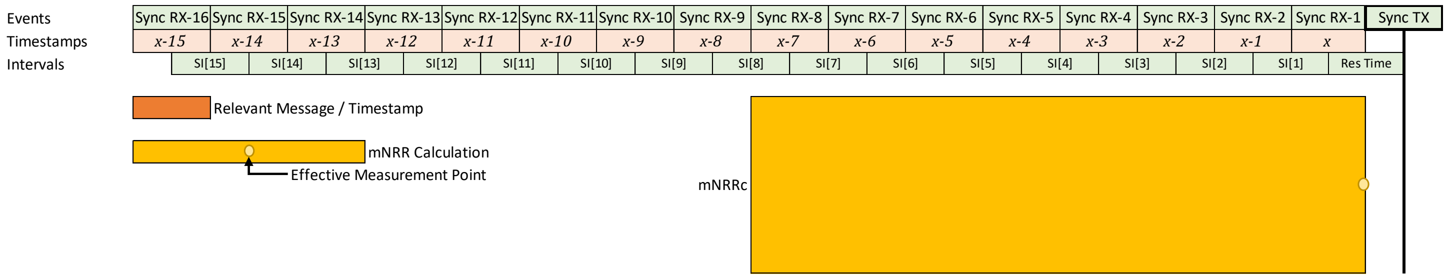


Events
Timestamps
Intervals
Sync RX-16 Sync RX-15 Sync RX-14 Sync RX-13 Sync RX-12 Sync RX-11 Sync RX-10 Sync RX-9 Sync RX-8 Sync RX-7 Sync RX-6 Sync RX-5 Sync RX-4 Sync RX-3 Sync RX-2 Sync RX-1 Sync TX
x-15 x-14 x-13 x-12 x-11 x-10 x-9 x-8 x-7 x-6 x-5 x-4 x-3 x-2 x-1 x
SI[15] SI[14] SI[13] SI[12] SI[11] SI[10] SI[9] SI[8] SI[7] SI[6] SI[5] SI[4] SI[3] SI[2] SI[1] Res Time
Relevant Message / Timestamp
mNRR Calculation
Effective Measurement Point
mNRR for Drift Measurement 1
Value A
Interval between Measurements
mNRR for Drift Measurement 2
Value B

The drift estimate is then used to create an error corrected NRR measurement, which is an average of four error corrected calculations.



Events
Timestamps
Intervals
Sync RX-16 Sync RX-15 Sync RX-14 Sync RX-13 Sync RX-12 Sync RX-11 Sync RX-10 Sync RX-9 Sync RX-8 Sync RX-7 Sync RX-6 Sync RX-5 Sync RX-4 Sync RX-3 Sync RX-2 Sync RX-1 Sync TX
x-15 x-14 x-13 x-12 x-11 x-10 x-9 x-8 x-7 x-6 x-5 x-4 x-3 x-2 x-1 x
SI[15] SI[14] SI[13] SI[12] SI[11] SI[10] SI[9] SI[8] SI[7] SI[6] SI[5] SI[4] SI[3] SI[2] SI[1] Res Time
Relevant Message / Timestamp
mNRR Calculation
Effective Measurement Point
mNRRc
mNRRcalc[1] + (Drift x Interval[1])
Interval[1]
mNRRcalc[2] + (Drift x Interval[2])
Interval[2]
mNRRcalc[3] + (Drift x Interval[3])
Interval[3]
mNRRcalc[4] + (Drift x Interval[4])
Interval[4]



Events
Timestamps
Intervals
Sync RX-16 Sync RX-15 Sync RX-14 Sync RX-13 Sync RX-12 Sync RX-11 Sync RX-10 Sync RX-9 Sync RX-8 Sync RX-7 Sync RX-6 Sync RX-5 Sync RX-4 Sync RX-3 Sync RX-2 Sync RX-1 Sync TX
x-15 x-14 x-13 x-12 x-11 x-10 x-9 x-8 x-7 x-6 x-5 x-4 x-3 x-2 x-1 x
SI[15] SI[14] SI[13] SI[12] SI[11] SI[10] SI[9] SI[8] SI[7] SI[6] SI[5] SI[4] SI[3] SI[2] SI[1] Res Time
Relevant Message / Timestamp
mNRR Calculation
Effective Measurement Point
mNRRc

6.1 NRR Drift Tracking – Start-up Behaviour

This section only applies to the Time Series simulation. Each run of the Monte Carlo simulates a single Sync message transmitted from the GM and consequent Sync messages from PTP Relays until arrival at an End Station with associated errors, including from previous messages, but assuming "steady state" operation, i.e. after at least 2 seconds of operation.

NRR is used when calculating meanLinkDelay and output Sync message fields. The first mNRRca and mNRRcb values will only be available after receipt of 16 Sync messages, i.e. at least 2 seconds of operation given the 125 ms Sync Interval. NRR is used when calculating meanLinkDelay and three fields in a Sync message. During this time meanLinkDelay and output Sync messages fields must still be calculated, so an alternative must be used.

At least two Pdelay_Req messages or two Sync messages must be received before calculating a NRR value. Note that it is technically possible to calculate a NRR using a combination of NRR and Sync messages but if using only two consecutive messages (i.e. no averaging) this can be very risky due to the potential for very short intervals and resulting high error due to timestamp errors, so it not recommended.

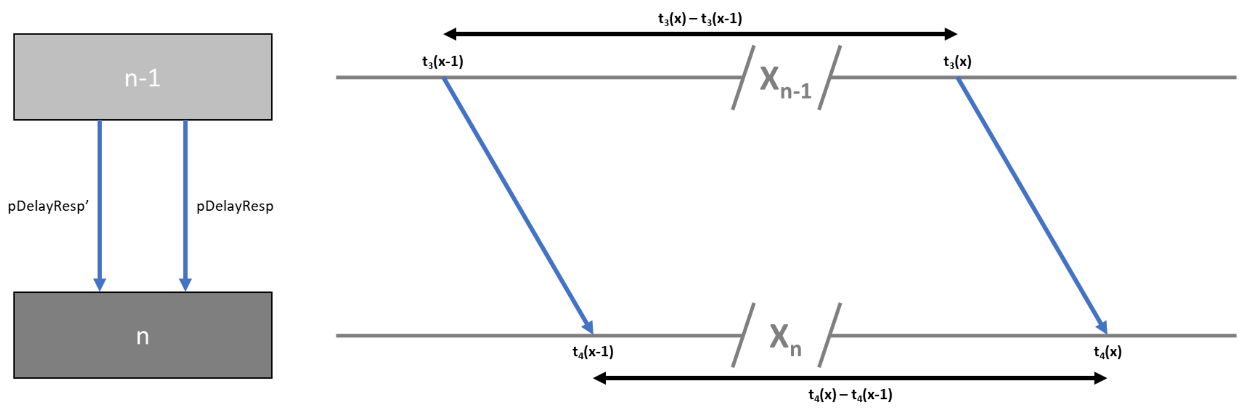
Prior to two messages being received, NRR = 1 (i.e. 0 ppm) should be used.

If two Pdelay_Req – Pdelay_Resp message exchanges occur prior to two Sync messages being received, NRR should be calculated using the formula…

$$mNRR = \left(\left(\frac{t_3(x)-t_3(x-1)}{t_4(x)-t_4(x-1)}\right)-1\right)\times 10^6 \qquad \textbf{ppm}$$

Where…

- t_3 – Timestamp of the Pdelay_Resp message egress from the previous node (n-1), timestamped by that node's Local Clock. Unit: **ns**.
- t_4 – Timestamp of the Pdelay_Resp message ingress to the current node (n), timestamped by that node's Local Clock. Unit: **ns**.



Once two Sync messages have been received, NRR should be calculated using the following formula…

2nd Sync Message: $$mNRR = \left(\left(\frac{t_{1outP}(x)-t_{1outP}(x-1)}{t_{2in}(x)-t_{2in}(x-1)}\right)-1\right)\times 10^6 \qquad \textbf{ppm}$$

Note that, since Pdelay Interval and Sync Interval are both 125 ms, two Sync messages should be received before three Pdelay_Resp messages have been received. When three or four Sync messages have been received, NRR should be calculated using the following formula…

3rd Sync message: $$mNRR = \left(\left(\frac{t_{1outP}(x)-t_{1outP}(x-2)}{t_{2in}(x)-t_{2in}(x-2)}\right)-1\right)\times 10^6 \qquad \textbf{ppm}$$

4th Sync message: $mNRR = \left(\left(\frac{t_{1outP}(x)-t_{1outP}(x-3)}{t_{2in}(x)-t_{2in}(x-3)}\right) - 1\right) \times 10^6$ **ppm**

On arrival of the 5th Sync message the first mNRRcalc and mNRRcalcT calculations can take place and should be used for NRR…

5th Sync message: $mNRRcalc(x) = \left(\frac{t_{1outP}(x)-t_{1outP}(x-4)}{t_{2in}(x)-t_{2in}(x-4)} - 1\right) \times 10^6$ **ppm**

As the 6th, 7th and 8th messages arrive an average can be taken and used for NRR, so…

6th Sync message: $mNRR = \sum_{i=x-1}^{x} \frac{mNRRcalc(i)}{2}$ **ppm**

7th Sync message: $mNRR = \sum_{i=x-2}^{x} \frac{mNRRcalc(i)}{3}$ **ppm**

8th Sync message: $mNRR = \sum_{i=x-3}^{x} \frac{mNRRcalc(i)}{4}$ **ppm**

The equation for the 8th Sync message is the same as for mNRRaverageA, but is not used for calculating NRR drift rate until arrival of the 16th Sync message.

For the 9th to the 15th Sync message, the same equation as for the 8th message should be used.

Once the 16th Sync message arrives, the regular equations will NRR drift tracking and error correction should be used.

7 meanLinkDelay Averaging

Following the 1st Pdelay message exchange, the meanLinkDelay is calculated as follows…

[To be completed – see 60802-McCall-Time-Sync-Recommended-Parameters-Correction-Factors-0322-v04.pdf (ieee802.org)]

8 Rate Ratio Drift Tracking & Error Compensation

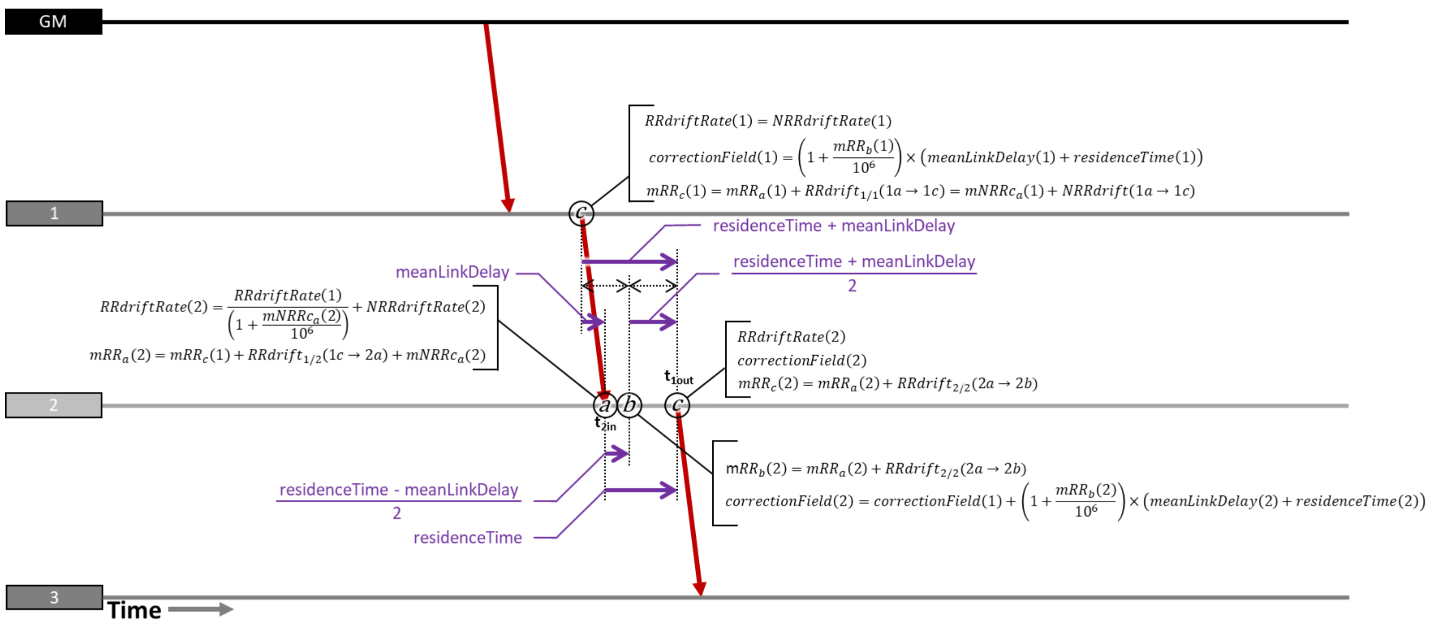
8.1 PTP Relay Instances

Relevant to Rate Ratio drift tracking & error compensation, each node (n), other than the GM, receives from the previous node (n), three fields in Sync and Sync_Follow_Up messages…

- $rateRatio(n-1)$ **ppm**
- $rateRatioDrift(n-1)$ **ppm/s**
- $correctionField(n-1)$ **ns**

…and generates local equivalents (n) that are forwarded to the next node. So, for example, Node 2 receives…

- $rateRatio(1)$ **ppm**
- $rateRatioDrift(1)$ **ppm/s**
- $correctionField(1)$ **ns**


GM
1
2
3
Time
$RRdriftRate(1) = NRRdriftRate(1)$
$correctionField(1) = \left(1 + \frac{mRR_b(1)}{10^6}\right) \times (meanLinkDelay(1) + residenceTime(1))$
$mRR_c(1) = mRR_a(1) + RRdrift_{1/1}(1a \to 1c) = mNRRc_a(1) + NRRdrift(1a \to 1c)$
residenceTime + meanLinkDelay
$\frac{\text{residenceTime + meanLinkDelay}}{2}$
meanLinkDelay
$RRdriftRate(2) = \frac{RRdriftRate(1)}{\left(1 + \frac{mNRRc_a(2)}{10^6}\right)} + NRRdriftRate(2)$
$mRR_a(2) = mRR_c(1) + RRdrift_{1/2}(1c \to 2a) + mNRRc_a(2)$
$RRdriftRate(2)$
$correctionField(2)$
$mRR_c(2) = mRR_a(2) + RRdrift_{2/2}(2a \to 2b)$
t_{1out}
t_{2in}
$mRR_b(2) = mRR_a(2) + RRdrift_{2/2}(2a \to 2b)$
$correctionField(2) = correctionField(1) + \left(1 + \frac{mRR_b(2)}{10^6}\right) \times (meanLinkDelay(2) + residenceTime(2))$
$\frac{\text{residenceTime - meanLinkDelay}}{2}$
residenceTime

There are four points in time that are of interest, illustrated, for Node 2, in the diagram above…

1. When the previous node transmits the Sync message, (c) for Node 1 above or [1c].
2. When the current node receives the Sync message, (a) for Node 2 above or [2a]. This is also the effective measurement point of the error corrected mNRRc value for the local node.
3. The effective measurement point of the change to the Correction Field, (b) for Node 2 above or [2b], which is half way between [1c] and…
4. When the current node transmits the Sync message, (c) for Node 2 above or [2c].

Note that, because [2b] is defined as halfway between [1c] and [2c], it's position can't be known until [2c] is know, i.e. Node 2 transmits the Sync message. Thus, any calculations associated with it can only happen once the Sync transmission time is known. Also, [2b] is the "effective measurement point" in terms of Rate Ratio if the Rate Ratio is drifting linearly, which the tracking and compensation algorithms assume.

Looking at the calculations for Node 2, and subsequent PTP Relay instances, i.e. not Node 1, the GM, or End Station…

First, calculate the local Rate Ratio Drift Rate by adding the ppm/s values. The Rate Ratio Drift Rate from the previous node is in ppm/s relative to the timebase of its Local Clock (i.e. the "s" in "ppm/s"). For highest precision, this should be converted to the timebase of the current node's Local Clock…

$$RRdriftRate(2) = \frac{RRdriftRate(1)}{\left(1+\frac{mNRRc_a(2)}{10^6}\right)} + NRRdriftRate(2) \qquad \textbf{ppm/s}$$

…however, given that adding ppm/s is already lacking the precision of multiplying actual ratios, this simplification is acceptable…

$$RRdriftRate(2) = RRdriftRate(1) + NRRdriftRate(2) \qquad \textbf{ppm/s}$$

Then calculate the $mRR_a(2)$ at [2a]. This is the mRR(1) plus mNRR(2), assuming adding ppm values is acceptable, both at [2a]. (For maximum precision the ratios would be multiplied, but 802.1AS uses addition of ppm values to decrease calculation time.). Also for maximum precision, mRR(1) would be modified to account for the passage of time since [1c]…

$$mRR_a(2) = mRR_c(1) + RRdrift_{1/2}(1c \rightarrow 2a) + mNRRc_a(2) \qquad \textbf{ppm}$$

$$= mRR_c(1) + \left(\frac{RRdriftRate(1)}{\left(1+\frac{mNRRc_a(2)}{10^6}\right)} \times meanLinkDelay(2) \right) + mNRRc_a(2) \qquad \textbf{ppm}$$

…but again, this simplification is acceptable…

$$mRR_a(2) = mRR_c(1) + \left(RRdriftRate(1) \times meanLinkDelay(2)\right) + mNRRc_a(2) \qquad \textbf{ppm}$$

Once [2c] is known, calculate the correctionField by first calculating the mRR at [2b]…

$$mRR_b(2) = mRR_a(2) + RRdrift_{2/2}(2a \rightarrow 2b) \qquad \textbf{ppm}$$

$$= mRR_a(2) + \left(RRdriftRate(2) \times \frac{residenceTime(2) - meanLinkDelay(2)}{2}\right) \qquad \textbf{ppm}$$

…then the correctionField itself…

$$correctionField(2) = correctionField(1) + \left(1 + \frac{mRR_b(2)}{10^6}\right) \times \left(meanLinkDelay(2) + residenceTime(2)\right) \qquad \textbf{ns}$$

…and finally the mRR at [2c]…

$$mRR_c(2) = mRR_a(2) + RRdrift_{2/2}(2a \rightarrow 2c) \qquad \textbf{ppm}$$

$$= mRR_a(2) + \left(RRdriftRate(2) \times residenceTime(2)\right) \qquad \textbf{ppm}$$

Three values are then passed on to the next node

- $rateRatio(2) = mRR_c(2)$ **ppm**
- $rateRatioDrift(2) = RRdriftRate(2)$ **ppm/s**
- $correctionField(2)$ **ns**

8.2 Grandmaster

At the Grandmaster there are no incoming fields and, because there is no meanLinkDelay or residenceTime and – in the current simulations – Working Clock @ GM is the same as Local Clock at GM…

- $rateRatio(0) = 0$ **ppm**
- $rateRatioDrift(0) = 0$ **ppm/s**
- $correctionField(0) = 0$ **ns**

8.3 Node 1

At Node 1, calculations are similar to Node 2, but the incoming fields from the GM are…

- $rateRatio(0) = 0$ **ppm**
- $rateRatioDrift(0) = 0$ **ppm/s**
- $correctionField(0) = 0$ **ns**

Therefore…

$$RRdriftRate(1) = NRRdriftRate(1) \qquad \textbf{ppm/s}$$

$$mRR_a(1) = mNRRc_a(2) \qquad \textbf{ppm}$$

$$correctionField(1) = \left(1 + \frac{mRR_b(2)}{10^6}\right) \times \left(meanLinkDelay(2) + residenceTime(2)\right) \qquad \textbf{ns}$$

…while all other calculations remain the same.

8.4 End Station

There is no residenceTime at an End Station and no transmitted Sync message, so it only needs to calculate RRdriftRate and mRR_a. These are then used by the ClockTimeReceiver, along with the received preciseOriginTimestamp + correctionField, to keep the End Station's Working Clock @ End Station on track.