60802 Time Sync – Normative Requirements & Testing

David McCall – Intel Corporation

Version 5

Contents

Note…

- This presentation **is** about the principles of what can and should be tested and therefore what 60802 should specify as normative requirements.
- It is **not** about the exact values of the normative requirements.

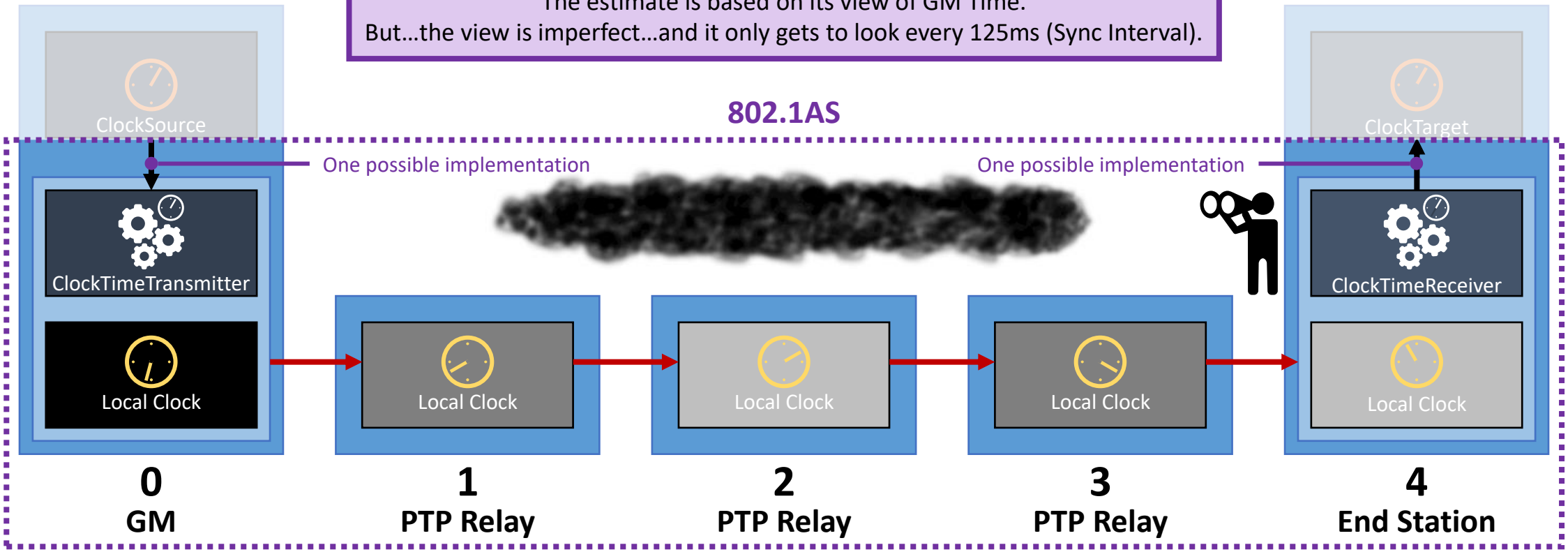
Recap…

- 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

Scope of 802.1AS

From David McCall "60802 Time Sync – Clock Relationships & Normative Requirements v3" contribution 17 Feb 2023

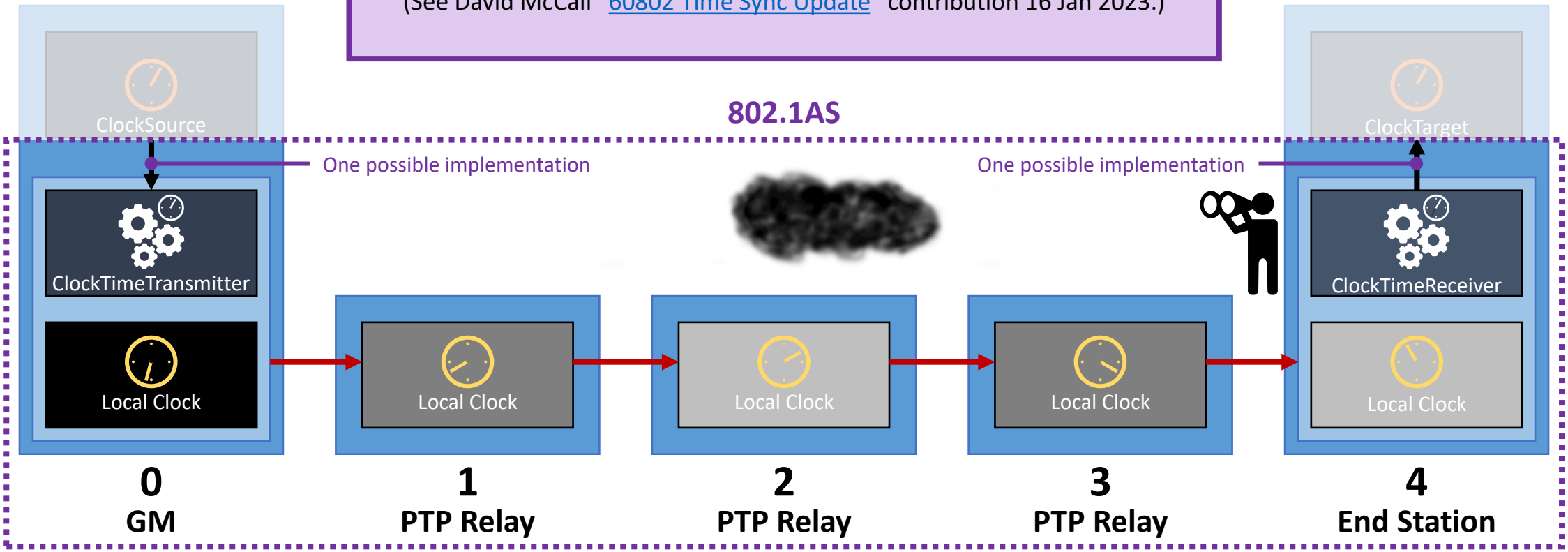
802.1AS provides a mechanism to keep the ClockTimeReceiver's estimate of GM Time (at the ClockTimeTransmitter) as accurate as possible.
The estimate is based on its view of GM Time.
But…the view is imperfect…and it only gets to look every 125ms (Sync Interval).



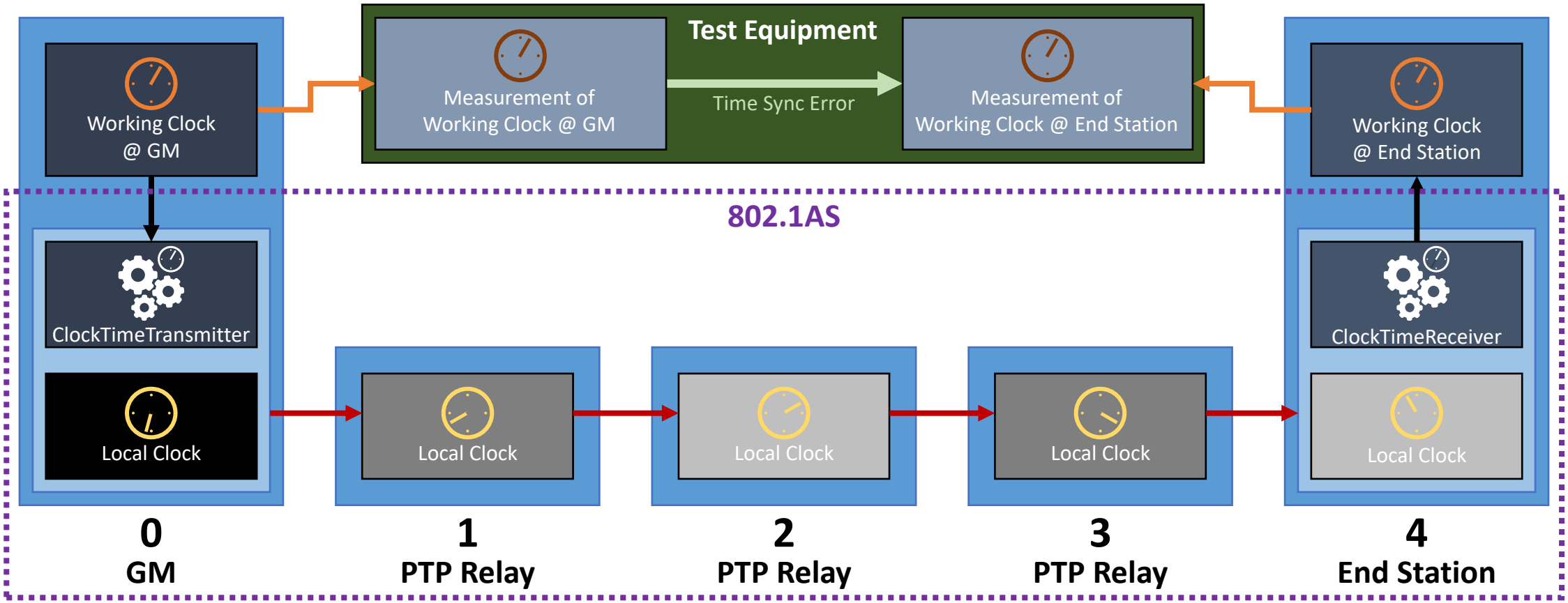
Scope of 802.1AS

From David McCall "60802 Time Sync – Clock Relationships & Normative Requirements v3" contribution 17 Feb 2023

IEC/IEEE 60802 improves the view.
(See David McCall "60802 Time Sync Update" contribution 16 Jan 2023.)



System-Level Time Error Measurement

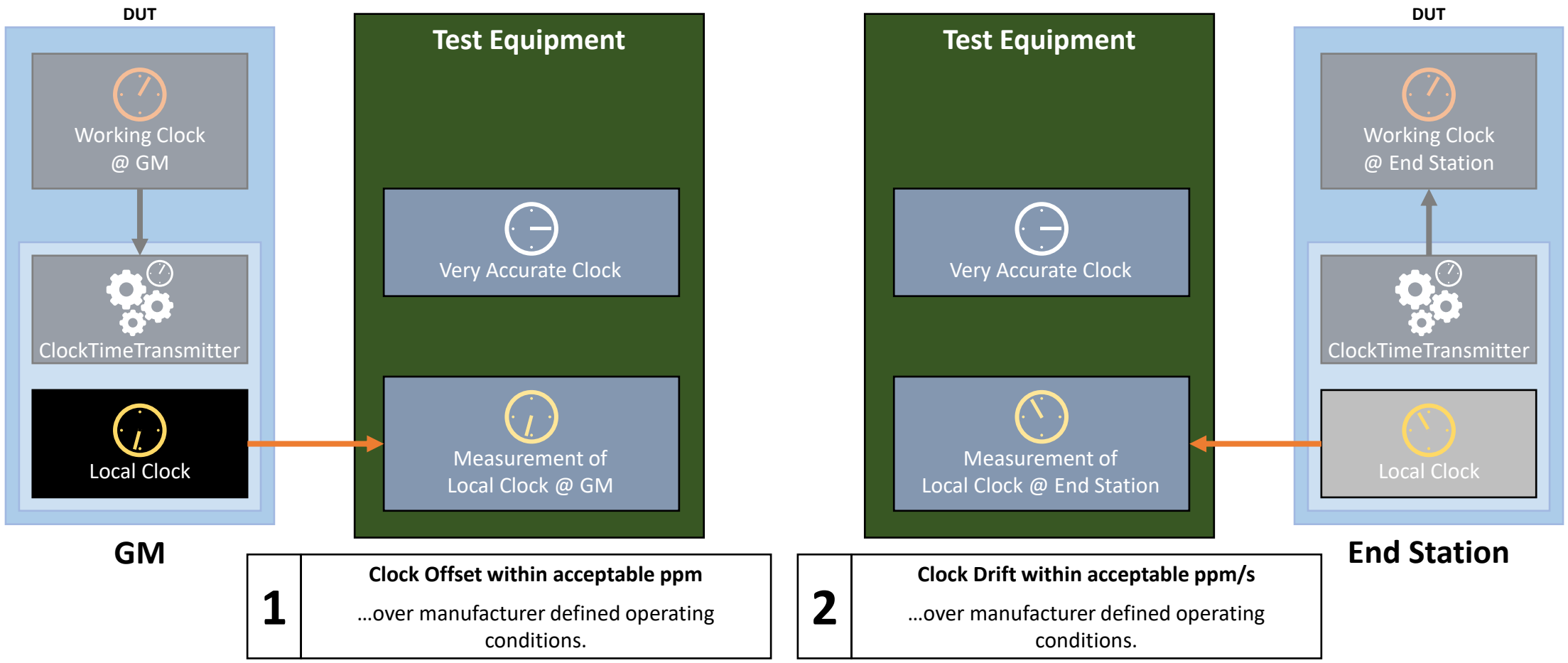


Test Equipment Assumptions

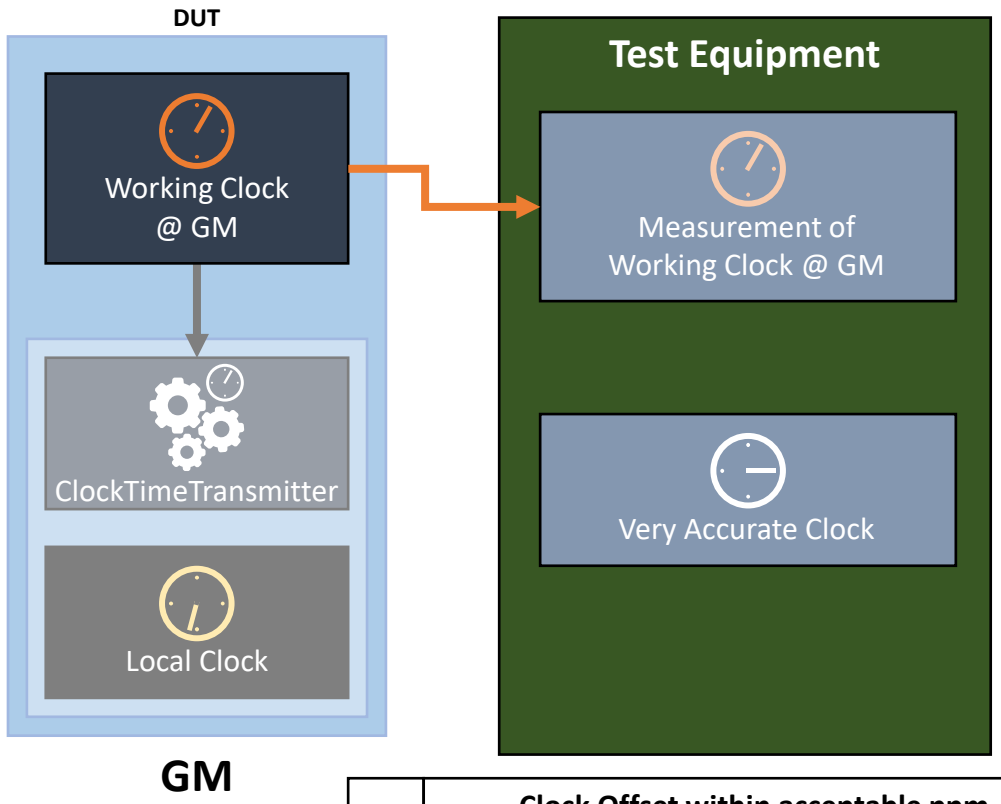
- Test equipment has access to accurate measurements of Working Clocks at GM & End Station (or, at GM, External Clock).
  - For example: PPS (Pulse Per Second)
- If information updates from actual clocks is infrequent, test equipment maintains its measurement of relevant clocks between updates.
  - Information from actual clocks arrive frequently enough that any errors between actual clocks and measured clocks are small relative to the Time Sync Error being measured.
- Test equipment also has a very accurate clock available, although not every test requires it.
  - Essential for measuring clock offset and drift…

Measuring Clock Offset & Drift

Measuring Local Clocks @ GM or End Station against Accurate Clock



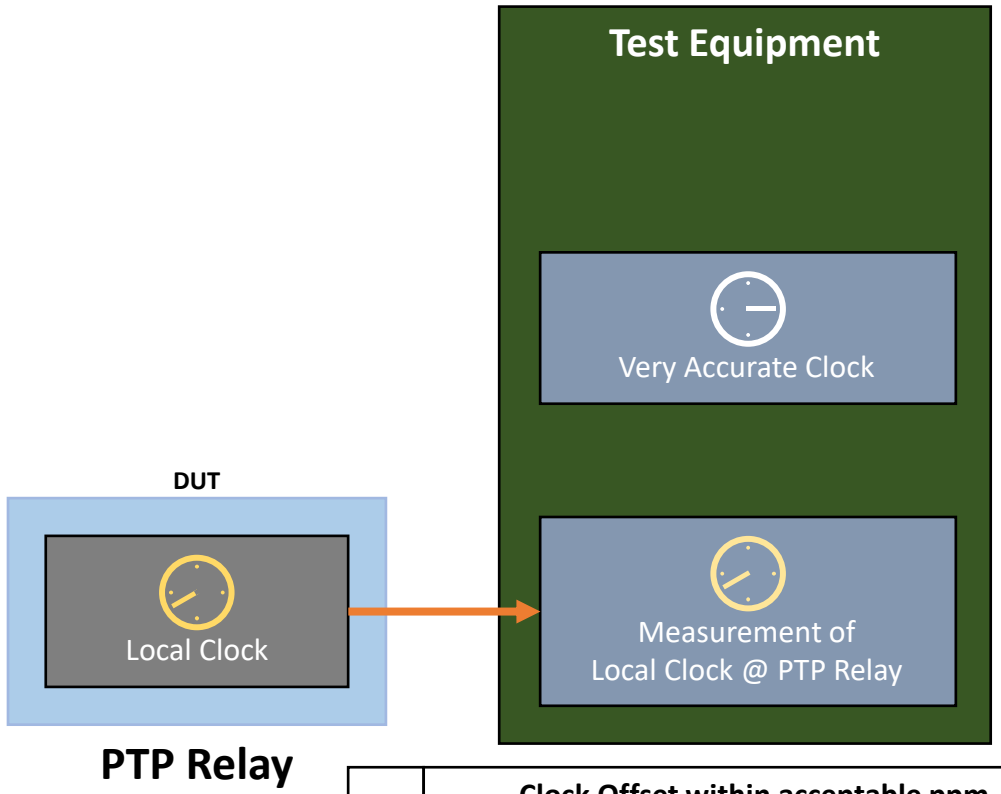
Measuring Working Clock @ GM against Accurate Clock



1	**Clock Offset within acceptable ppm** …over manufacturer defined operating conditions.

2	**Clock Drift within acceptable ppm/s** …over manufacturer defined operating conditions.

Measuring Local Clock @ PTP Relay against Accurate Clock

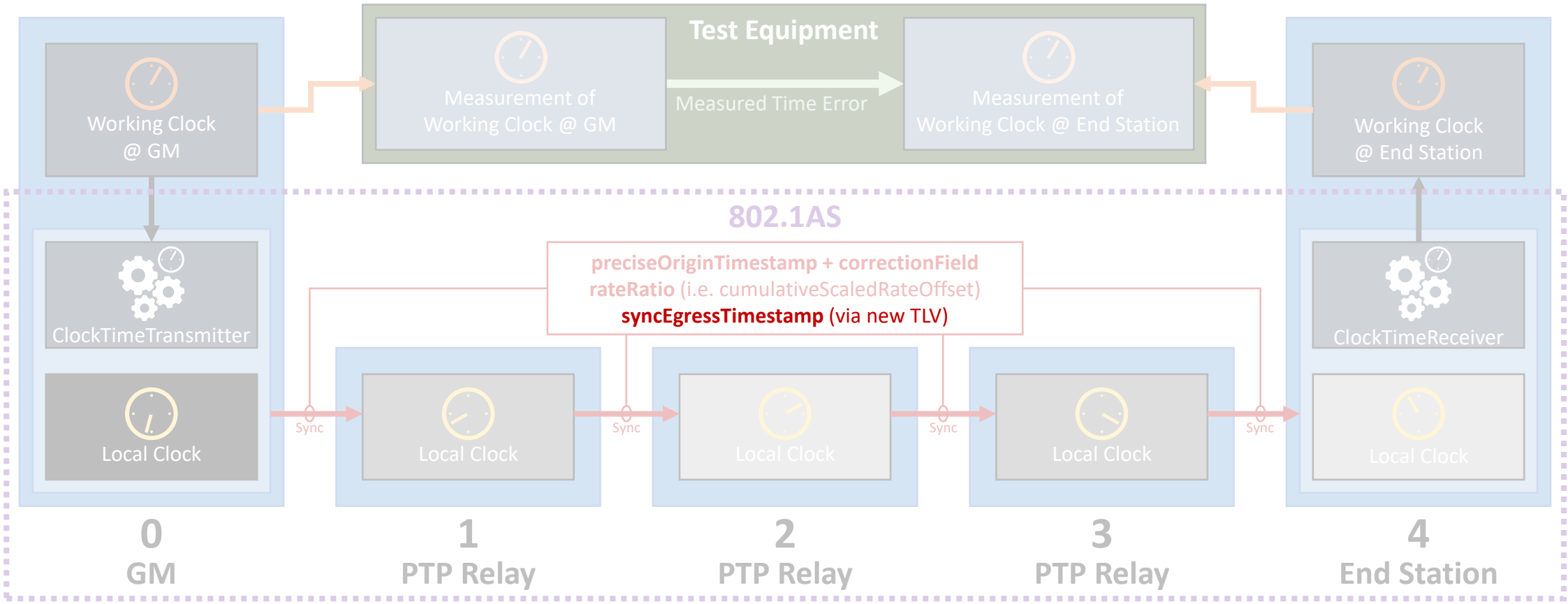


1	**Clock Offset within acceptable ppm** …over manufacturer defined operating conditions.

2	**Clock Drift within acceptable ppm/s** …over manufacturer defined operating conditions.

Measuring Error Generation at Grandmaster

System-Level Time Error Measurement



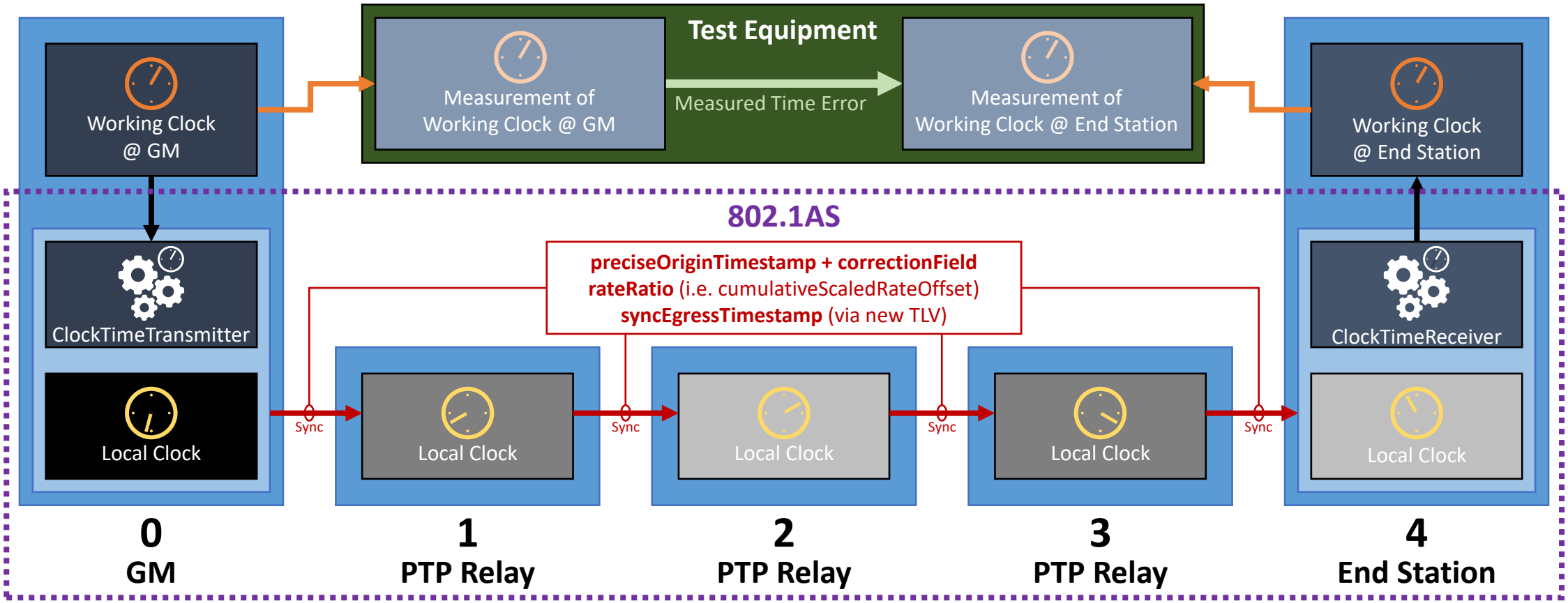
syncEgressTimestamp

- Field in new TLV, specified in upcoming 802.1ASdm specification. Added to Sync (1-step) or Follow_Up (2-step)
- From "802.1ASdm Contribution - New Drift Tracking TLV" contribution to IEEE 802.1 TSN, David McCall, March 2023…

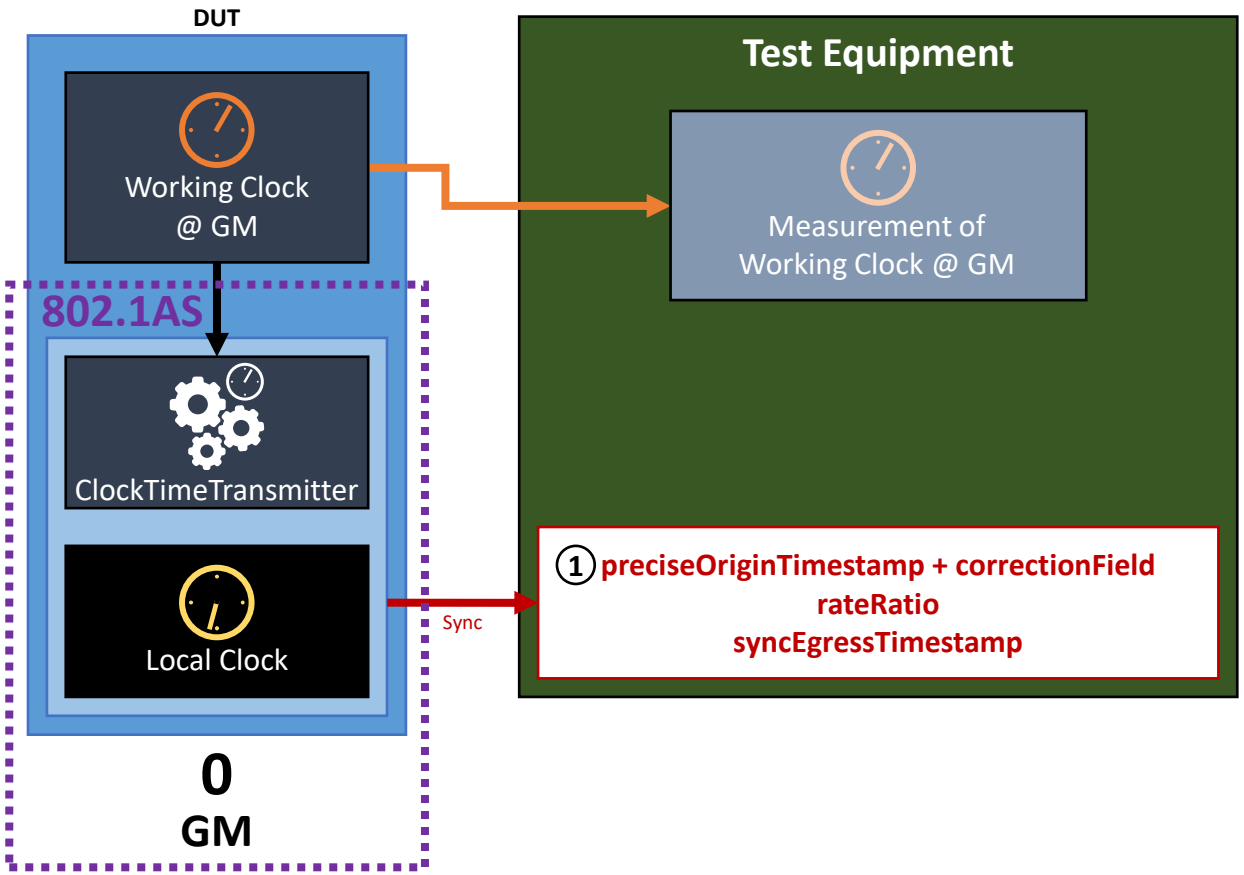
> **11.4.4.4.6 syncOriginTimestamp (Timestamp)**
> The value is the seconds and nanoseconds portion of the syncEventEgressTimestamp of the associated Sync message (see 11.4.3.2).

- After discussion, the field will be renamed **syncEgressTimestamp** to avoid confusion with **preciseOriginTimestamp**
  - **preciseOriginTimestamp** is in terms of the **Working Clock @ GM** (or other domain) and is not altered by PTP Relay instances.
  - **syncEgressTimestamp** is in terms of each instance's **Local Clock** and reflects the Sync message's egress from that instance.
- It enables the calculation of NRR using Sync messages (as opposed to pDelay_Resp messages), and 60802 will require support.
  - It will be optional in 802.1ASdm

System-Level Time Error Measurement

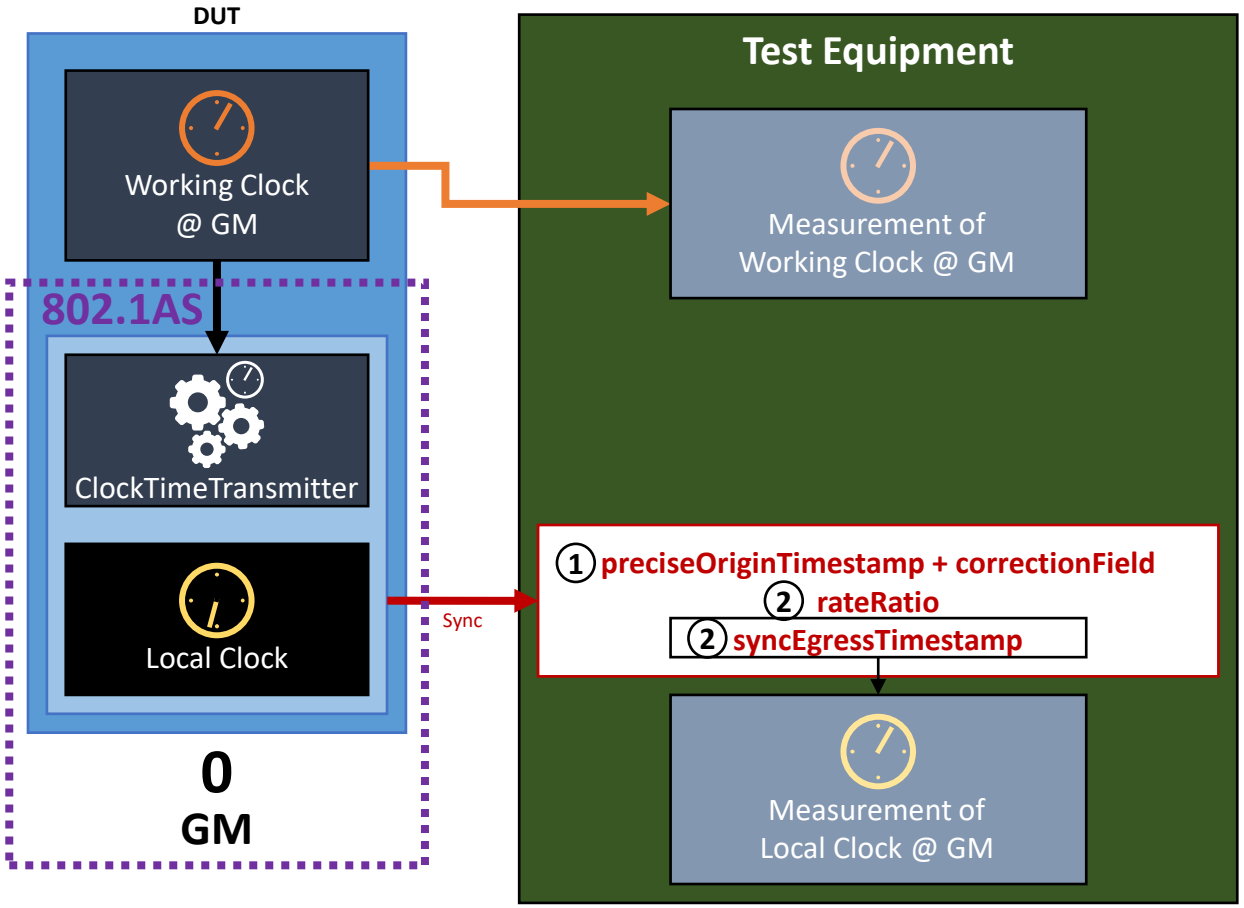


Device-Level Error Generation Measurement at GM



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

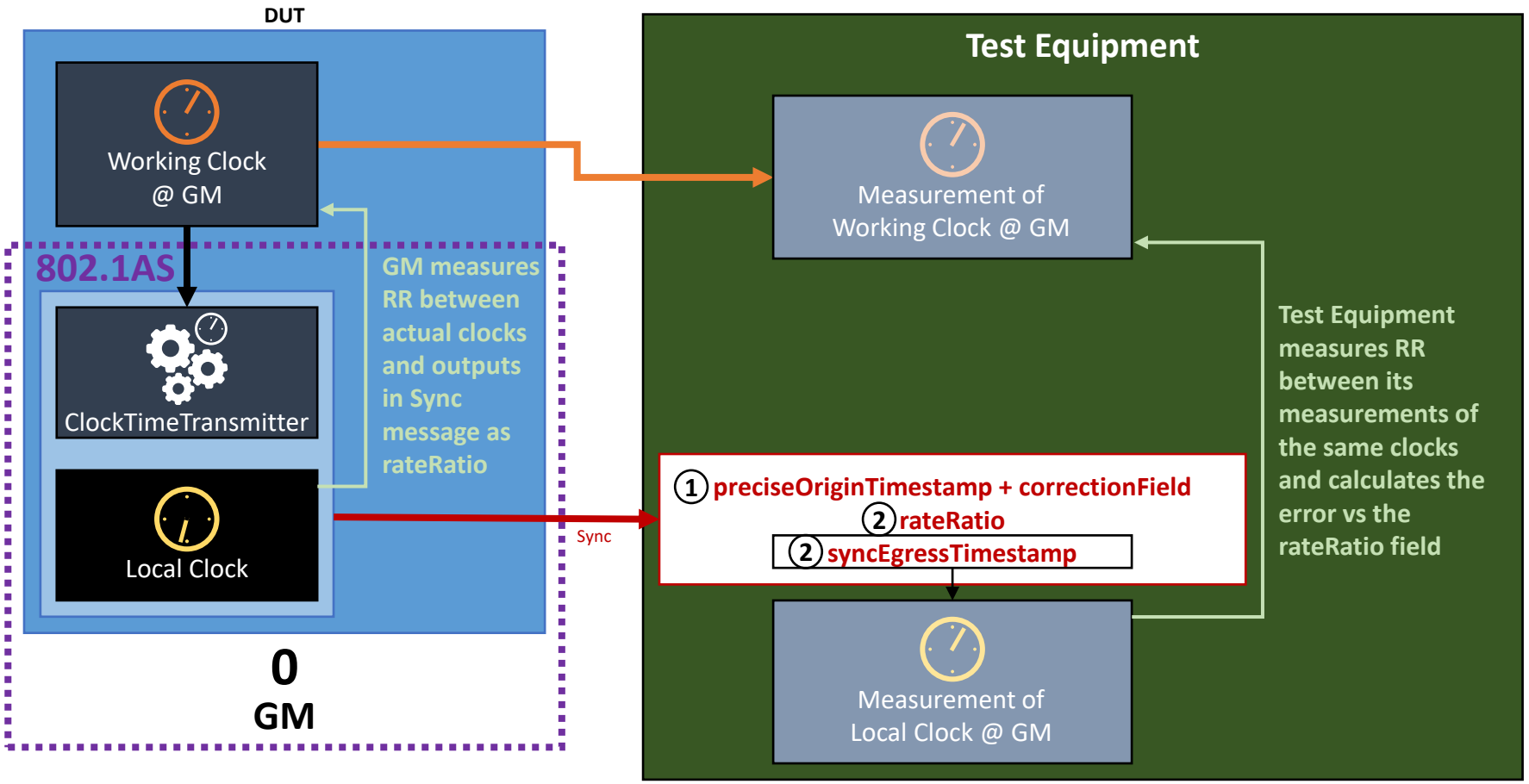
Device-Level Error Generation Measurement at GM



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

For 2, it is possible to make "Measurement of Local Clock @ GM" via syncEgressTimestamp, and not a more direct measurement (similar to "Measurement of Working Clock @ GM"), but this means the measurement includes timestamp errors. If "Measurement of Local Clock @ GM" is via a more direct measurement, an additional Normative Requirement is necessary.

Difference between Measurements of Local Clock @ GM



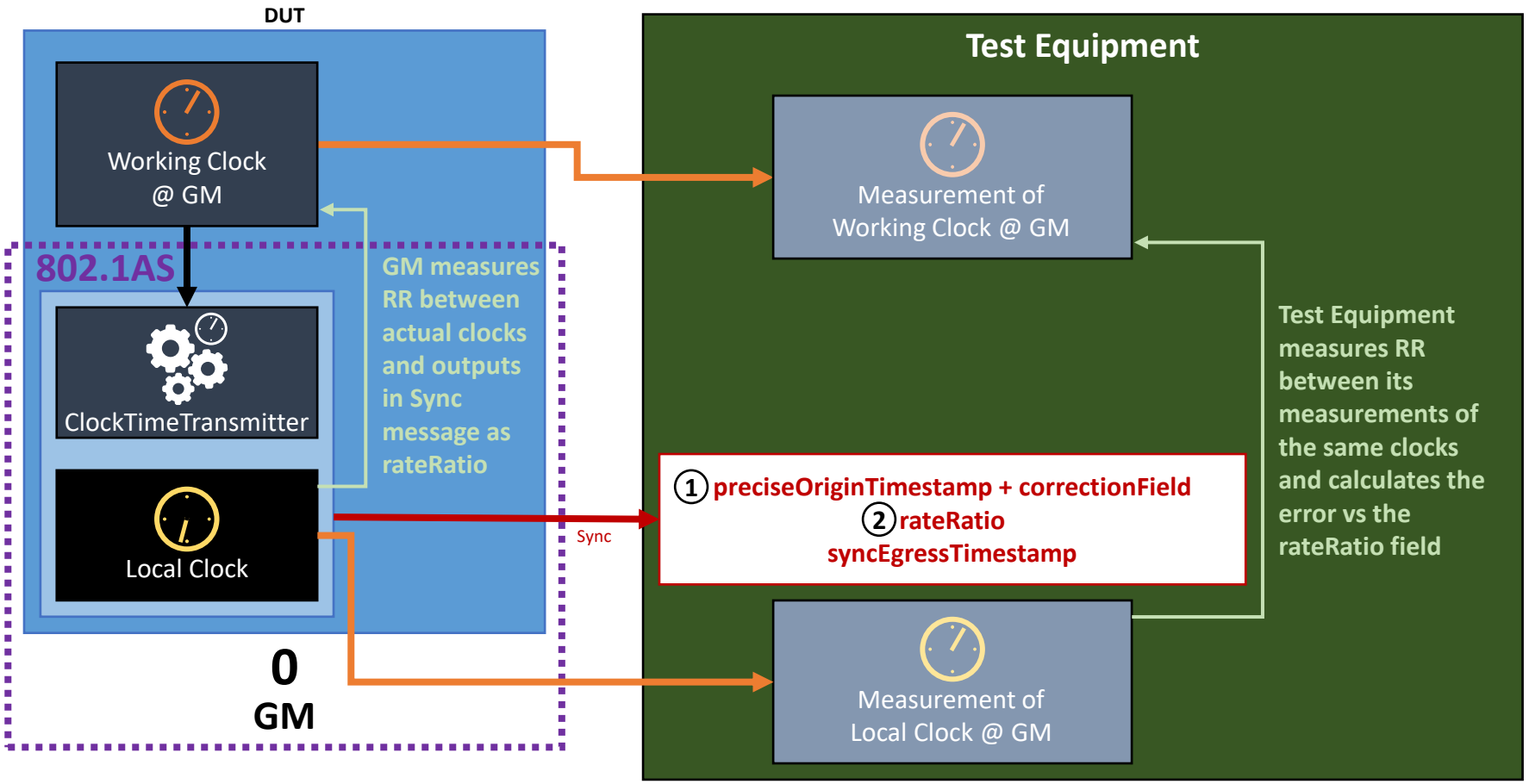
2	**rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio (Between Measurement of Working Clock @ GM and Measurement of Local Clock @ GM)**

rateRatio is checked. Sufficiently large errors in the GM's measurement of RR would cause the test to fail.

Sufficiently large errors in syncEgressTimestamp (which would affect the ability of a downstream node to accurately measure NRR) would also cause the test to fail.

(If the errors somehow cancel each other out…then there isn't a error. *Is this a good thing?*😮)

Difference between Measurements of Local Clock @ GM



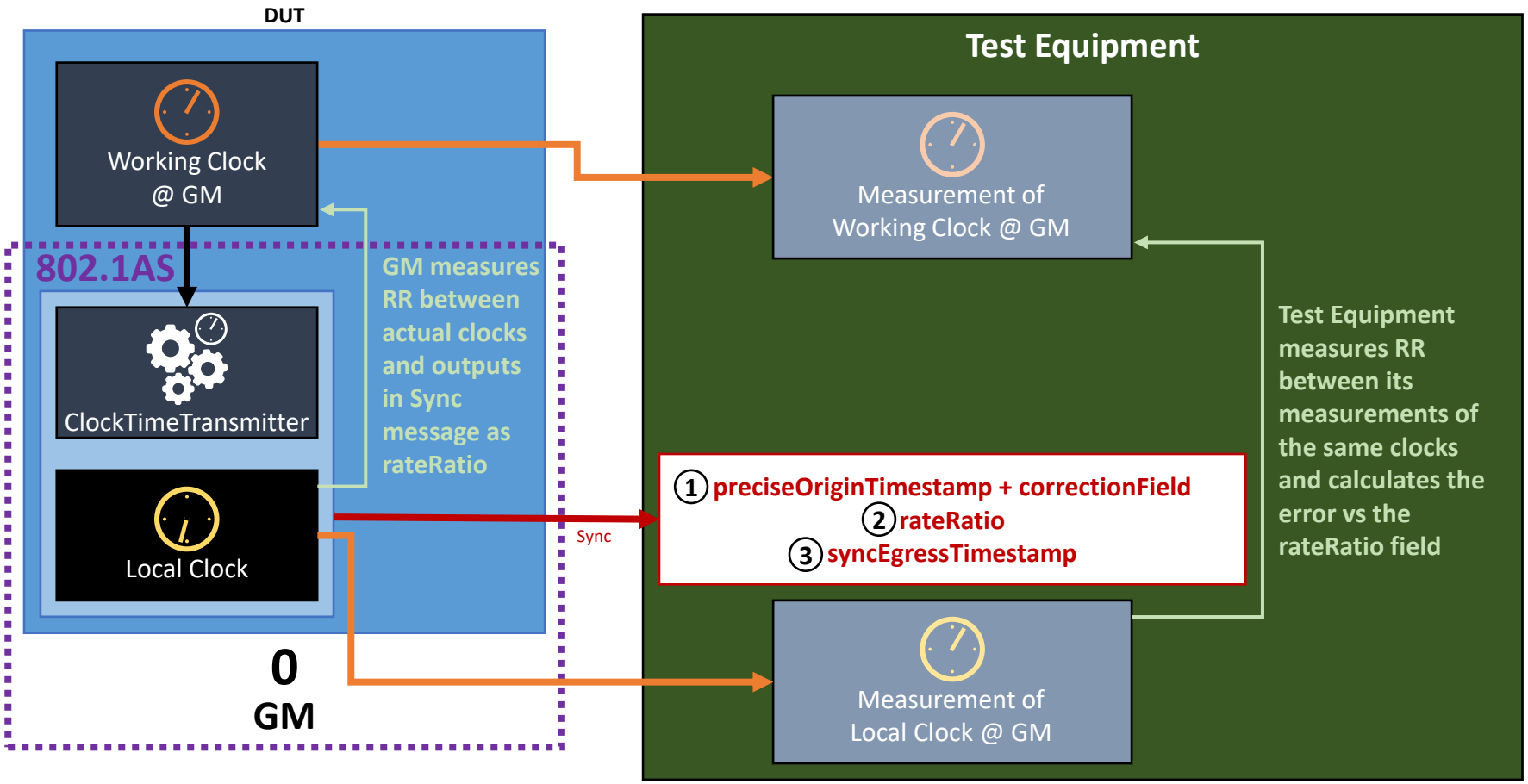
2	**rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio (Between Measurement of Working Clock @ GM and Measurement of Local Clock @ GM)**

In this case, only rateRatio is checked.

syncEgressTimestamp could have large errors (affecting the ability of the downstream node to accurately measure NRR) without the test failing.

Another normative requirement could fill the gap…

Difference between Measurements of Local Clock @ GM



2	**rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio (Between Measurement of Working Clock @ GM and Measurement of Local Clock @ GM)**
3	**syncEgressTimestamp** …should equal (within acceptable margin)… **Measured Egress Time of Sync Message (measured against Measurement of Local Clock @ GM)**

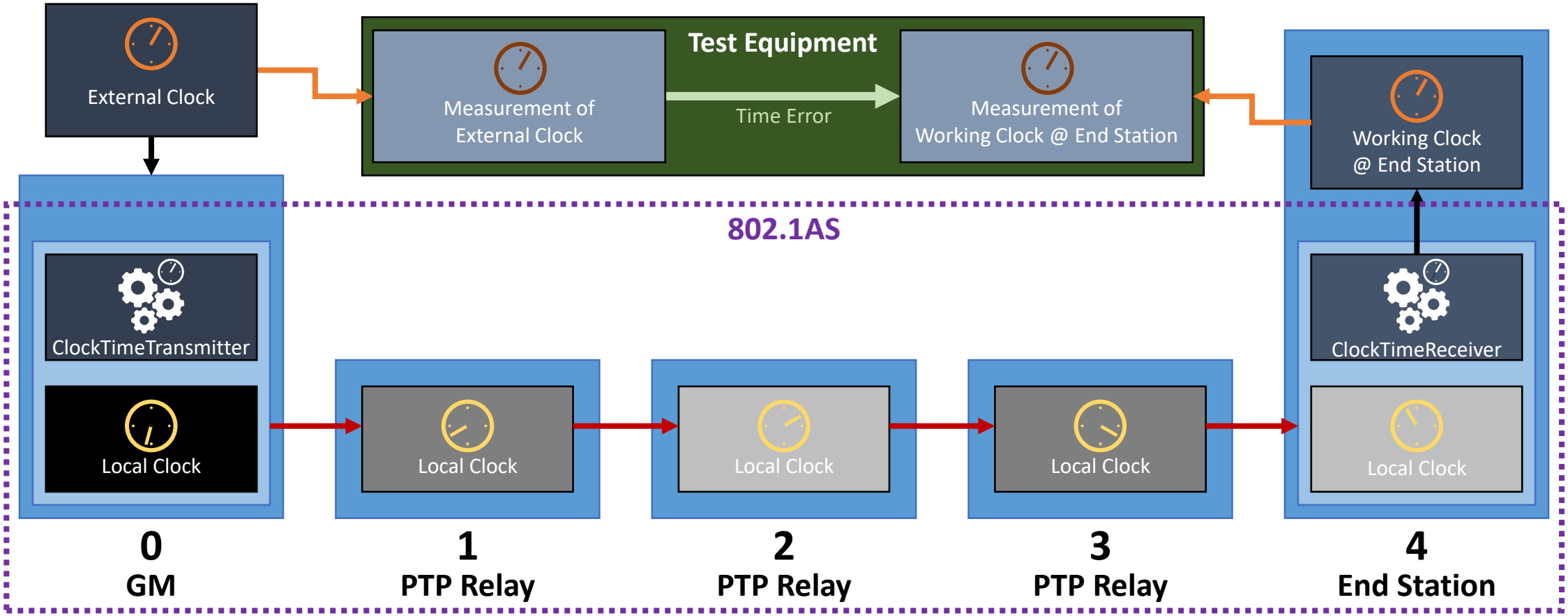
Recommendation…

- Specify three normative requirements…
  - Accuracy of preciseOriginTimestamp + correctionField vs “direct” measurement of Working Clock @ GM…
  - Accuracy of rateRatio vs measured Rate Ratio
    - Measured Rate Ratio via “direct” measurements of both Working Clock @ GM & Local Clock @ GM
  - Accuracy of syncEgressTimestamp vs “direct” measurement of Local Clock @ GM…
- …rather than two normative requirements.
  - Accuracy of preciseOriginTimestamp + correctionField vs “direct” measurement of Working Clock @ GM…
  - Accuracy of rateRatio vs. measured Rate Ratio
    - Measured Rate Ratio via “direct” measurement of Working Clock @ GM and measurement of Local Clock @ GM via syncEgressTimestamp
- All errors are not equal. Clock drift can be tracked over time and errors compensated for. This is not the case for timestamp errors. If we have a single normative requirement, combining the two based on our assumptions, an implementation may take a very different approach.

NRR and RR drift measurement & error compensation at GM

- If Working Clock @ GM and Local Clock @ GM are the same, or at least linked so their ppm offsets are synchronised, (which is implementation dependant) the GM doesn't experience errors due to clock drift. There's nothing to measure and no errors to compensate for.
  - This means there is no need for normative requirements under different clock drift scenarios, in there way there is at PTP Relay and End Station instances.
- If Working Clock @ GM and Local Clock @ GM are different, which they can be, for example…

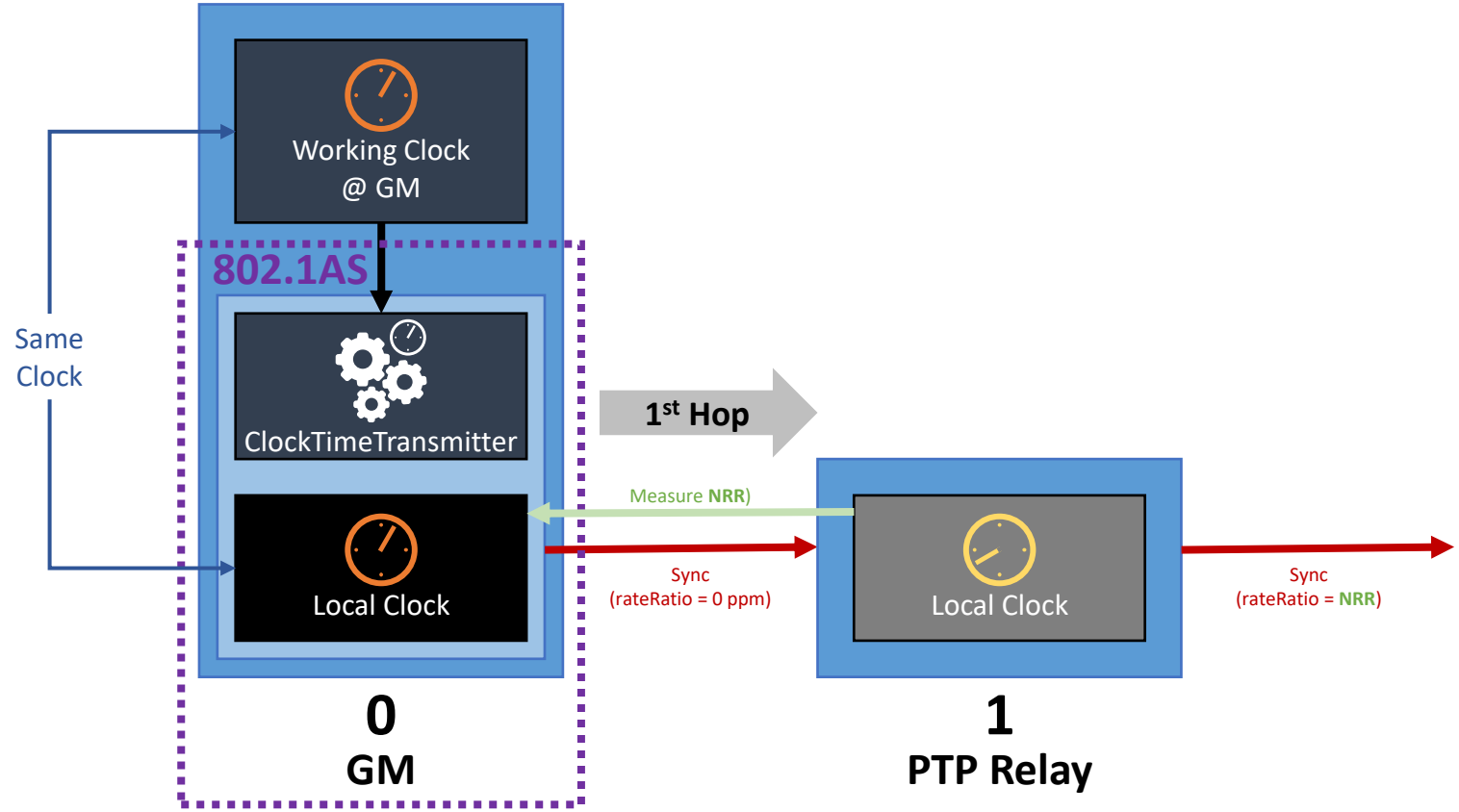
System-Level Time Error Measurement – External Clock



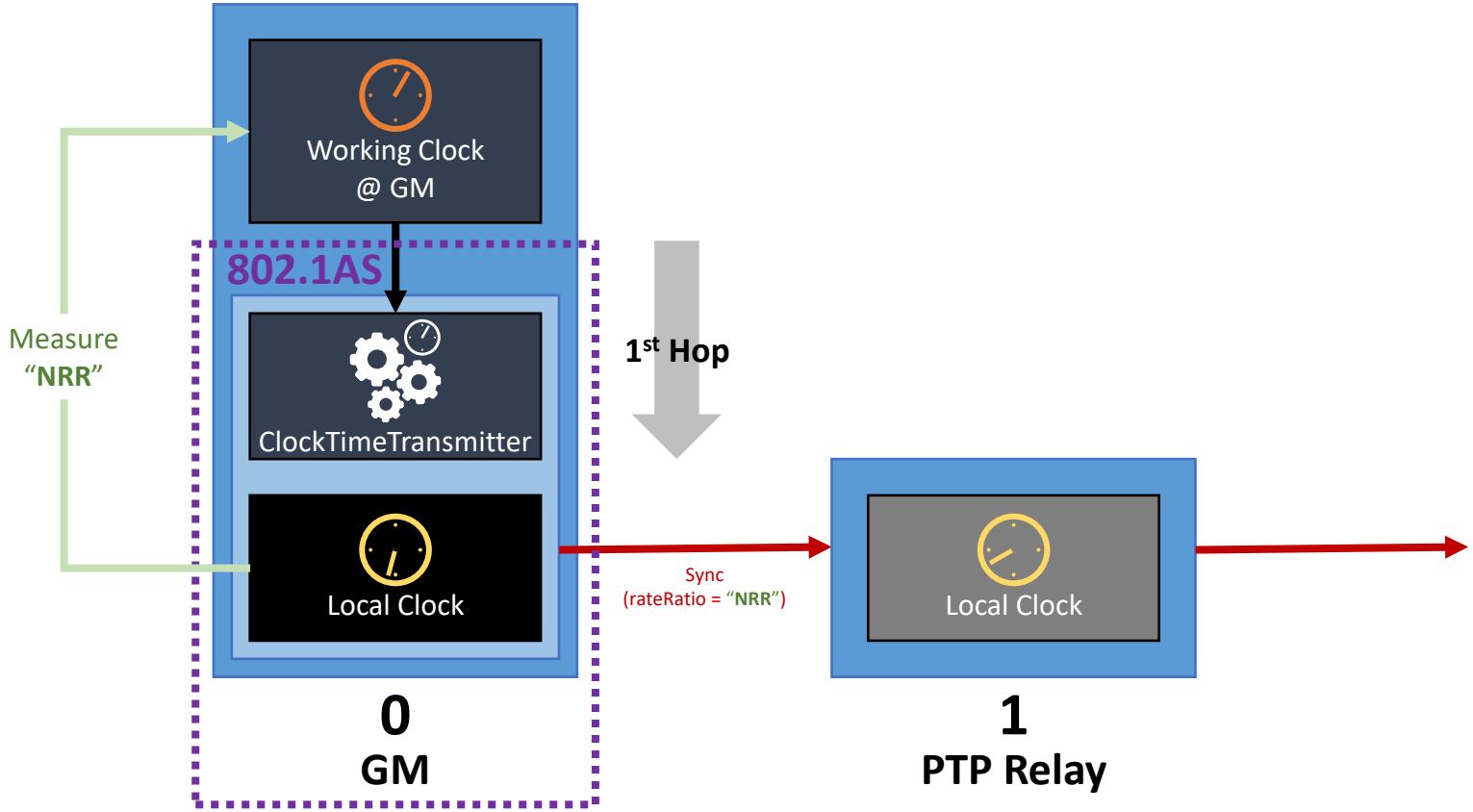
NRR and RR drift measurement & error compensation at GM

- If Working Clock @ GM and Local Clock @ GM are different, which is another implementation possibility, then it is similar to Local Clock @ GM being the first node in the chain after the GM.

GM: Working & Local Clocks the Same…



GM: Working & Local Clocks Different…



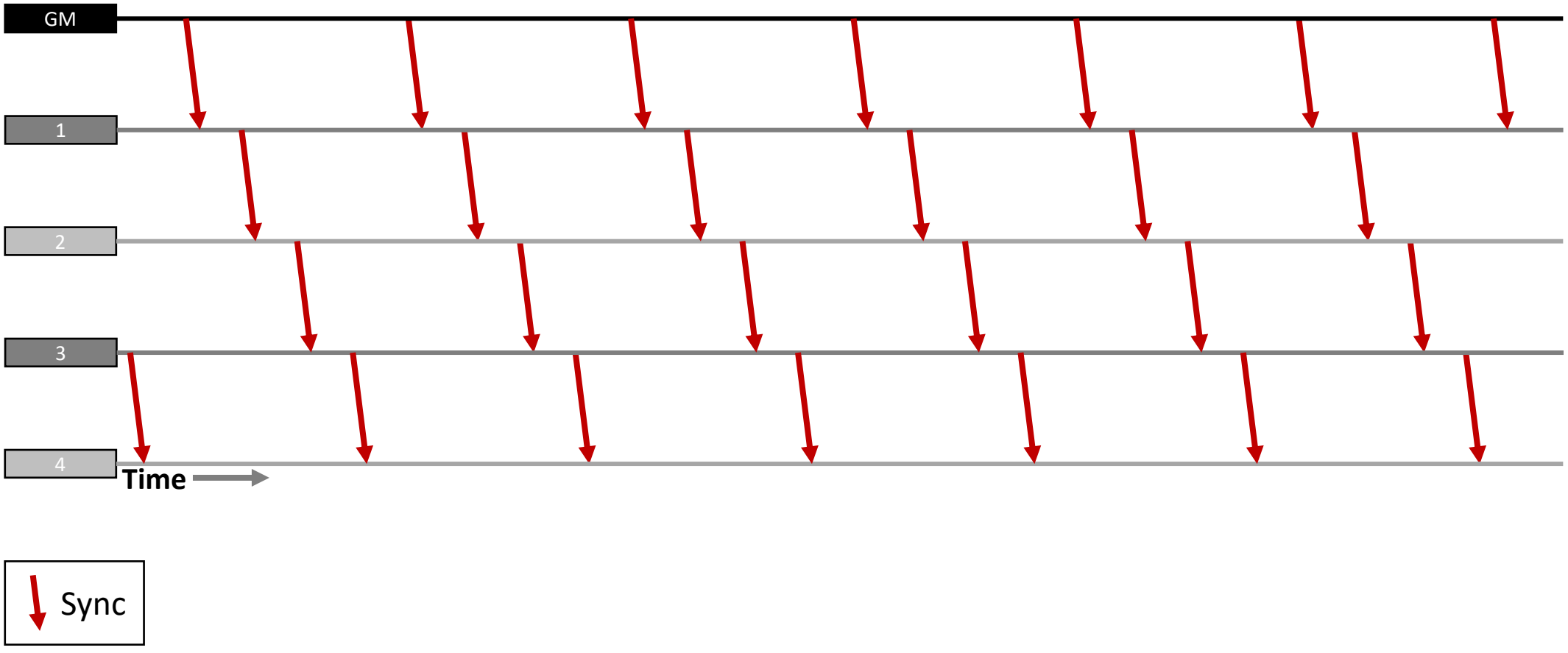
Open Questions…

- Should the GM track the “NRR” between Local Clock @ GM and Working Clock @ GM and attempt to compensate for any errors?
- If so…what’s the best approach? Should there be a section on this in the informative annex?
- Should there be normative requirements around this behaviour, beyond what’s already specified?
  - Are they required to meet performance goals? / Could failure to meet a requirement in this area cause failure to meet performance goals?
- **Could** there be normative requirements? (Is it feasible to measure the behaviour of interest?
- This is particularly important as errors in RR due to Working Clock @ GM clock drift at the 1st hop survive (i.e. there is no node-to-node cancelation effect) down the entire chain of devices, generating errors in the same direction (i.e. additively) at every hop.

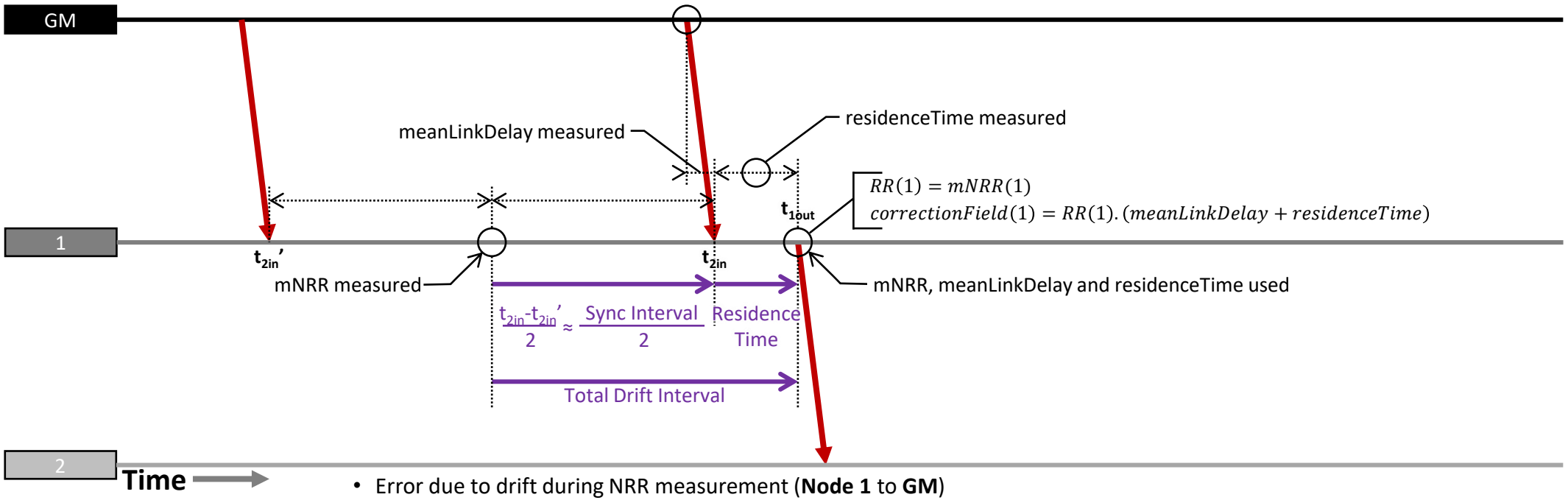
We will return to these questions after looking at measuring NRR & RR drift tracking and error compensation in the next sections.

Measuring NRR and RR Drift Tracking & Error Compensation Performance

Clock Drift Error – Relevant Intervals
4 Hops



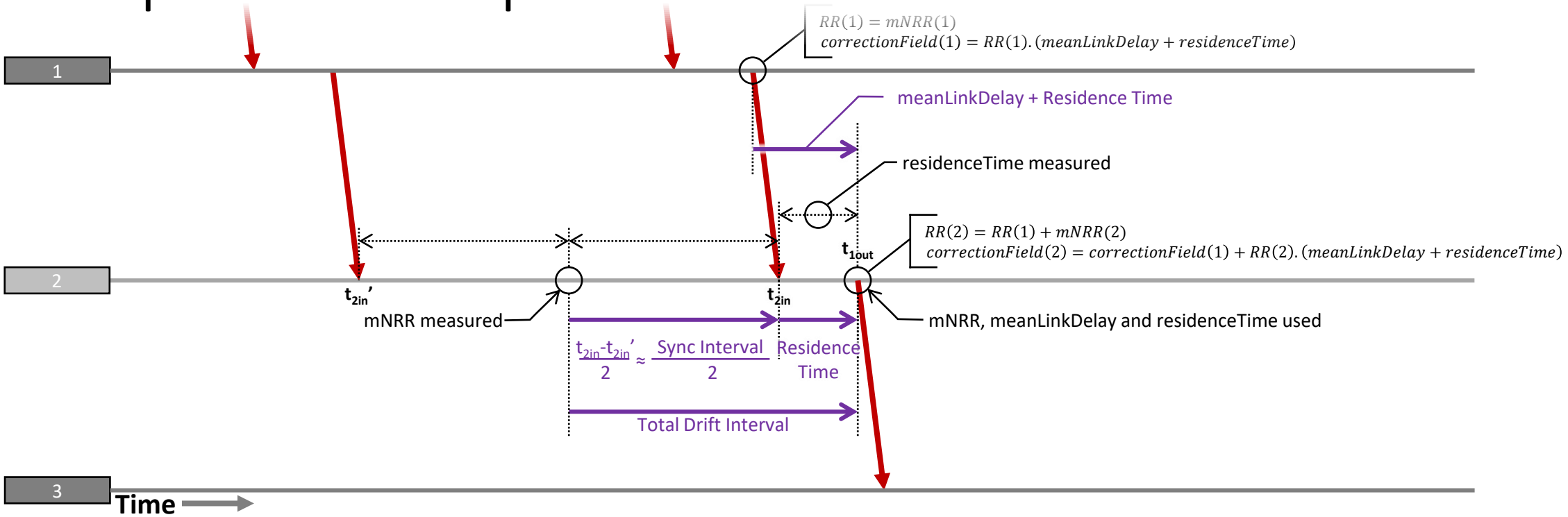
Clock Drift Error – Relevant Intervals
4 Hops – 1st Hop



- Error due to drift during NRR measurement (**Node 1** to **GM**)
  - Interval is half t_4-t_4' which is nominally half the pDelay Interval, but actual pDelay Interval varies.
- Error due to drift between measuring and using NRR (**Node 1** to **GM**)
  - Interval is between zero and the maximum pDelay Interval, which is larger than the nominal pDelay Interval.
  - Assumes that pDelayResp arriving between t2in and t1out will trigger new mNRR calculation; not unreasonable as information is included in Follow-up, not Sync, if 2-step Sync is used.
- Error due to drift during residenceTime measurement (**Node 1** to **GM**)
  - meanLinkDelay is measured separately, is much smaller, and can be averaged to remove errors, so is ignored for Drift Error.

Sync

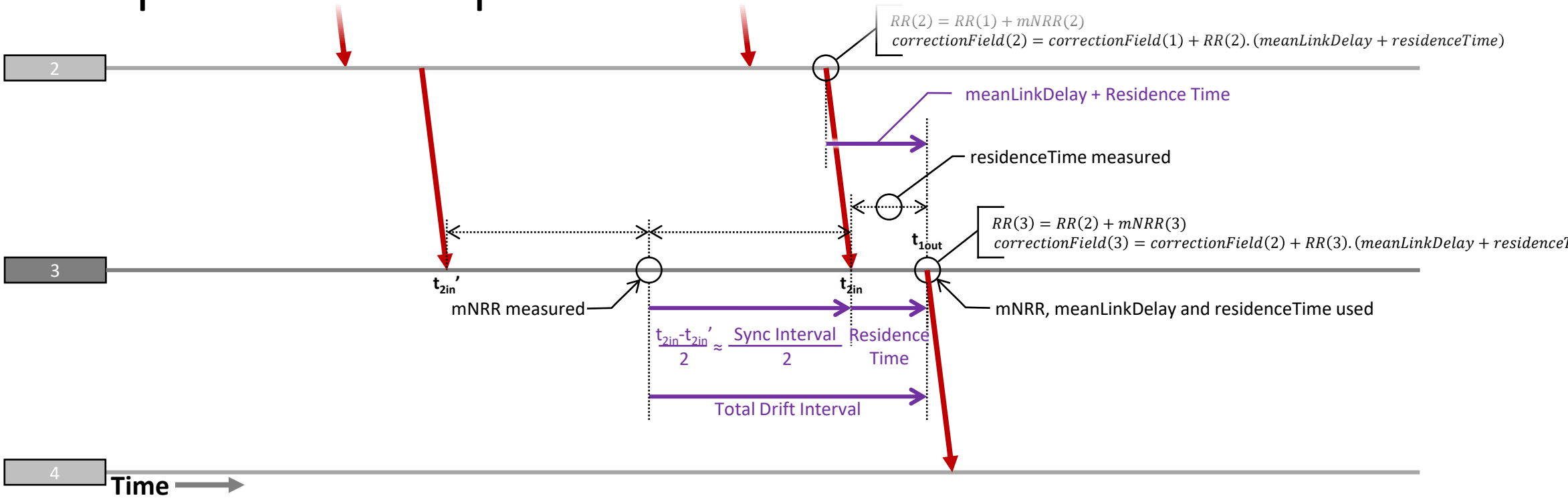
Clock Drift Error – Relevant Intervals
4 Hops – 2nd Hop



- Same errors in mNRR as 1st Hop.
  - Error due to drift during NRR measurement. (**Node 2** to **Node 1**)
  - Error due to drift between measuring and using NRR. (**Node 2** to **Node 1**)
  - Error due to drift during Residence Time measurement. (**Node 2** to **GM**)
- Additional error from 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.

Sync

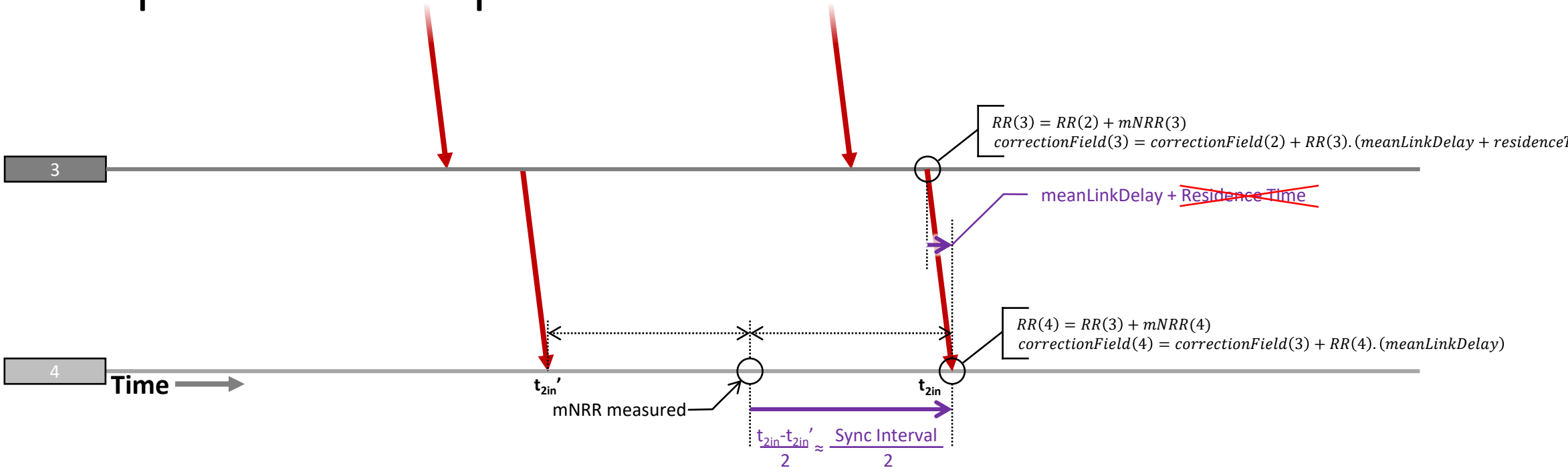
Clock Drift Error – Relevant Intervals
4 Hops – 3rd Hop



Sync

- Same errors in NRR and RR as 2nd Hop.
  - Error due to drift during NRR measurement. (**Node 3** to **Node 2**)
  - Error due to drift between measuring and using NRR. (**Node 3** to **Node 2**)
  - Error due to drift during Residence Time measurement. (**Node 3** to **GM**)
  - Error due to drift between RR(2) calculation, at Node 2, and use in calculating RR(3). (**Node 2** to **GM**)

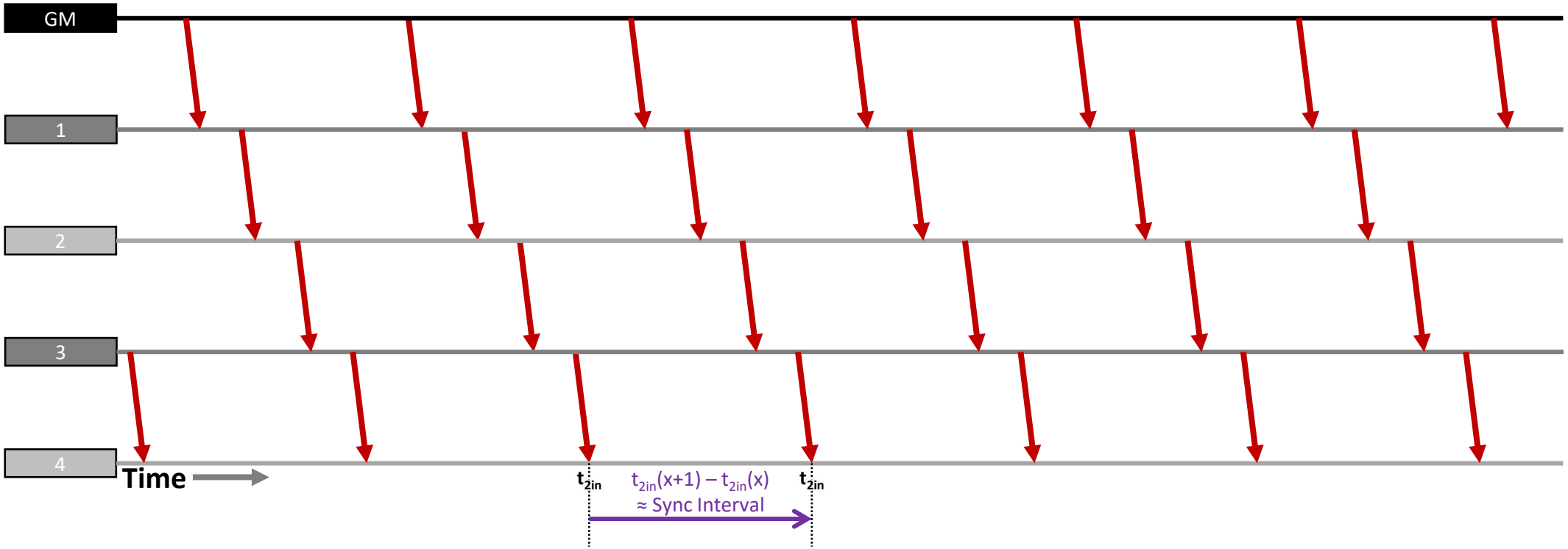
Clock Drift Error – Relevant Intervals
4 Hops – 4th Hop



- Similar errors in NRR and RR as 2nd & 3rd Hop.
  - Error due to drift during NRR measurement. (**Node 4** to **Node 3**)
  - Error due to drift between RR(3) calculation, at Node 3, and use starting at **receipt of Sync**. (**Node 3** to **GM**)
    - Modeled as zero due to the absence of Residence Time at the final node.
- No error due to drift between measuring and using NRR as RR(4) is calculated when Sync message is received.
- No Residence Time, so no error due to drift during measurement.
- There is additional error during the period until the next Sync message...

Sync

Clock Drift Error – Relevant Intervals
4 Hops – 4th Hop



Sync

- Dynamic time error (dTE) due to error in RR at start of interval, i.e. if RR has positive error, ClockTimeReceiver will run too fast. (**Node 4 to GM**)
- Dynamic time error (dTE) due to RR drift between receipt of one Sync message and the next. (**Node 4** to **GM**)
  - Even if RR is accurate at start of interval it can drift between then and the end of the interval, inducing dynamic time errors (dTE)
- Interval is nominally the Sync Interval, but there is some variation.

Effects of NRR & RR Drift Tracking & Compensation

- In 60802, PTP Relay Instances and End Stations include separate algorithms to track and correct for two sources of clock drift related error:
  - Error in incoming RR field (from previous node)
  - Error in measured NRR (between current and previous node)

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

- Both algorithms affect the amount of error in $RR(n)$ which, for PTP Relays, in turn affects the amount of error in the $correctionField(n)$ via…

$$correctionField(n) = correctionField(n-1) + RR(n).(meanLinkDelay + residenceTime)$$

- End Stations do not output a Sync message, so do not generate a $correctionField(n)$. Instead the effect of compensation algorithms will show up in how effectively Working Clock @ End Station tracks Working Clock @ GM between arrival of Sync messages.
  - End Stations can also track drift in their Local RR (i.e. RR(n) as well as RR(n-1)) and compensate for errors arising due to drift between arrival of Sync messages.

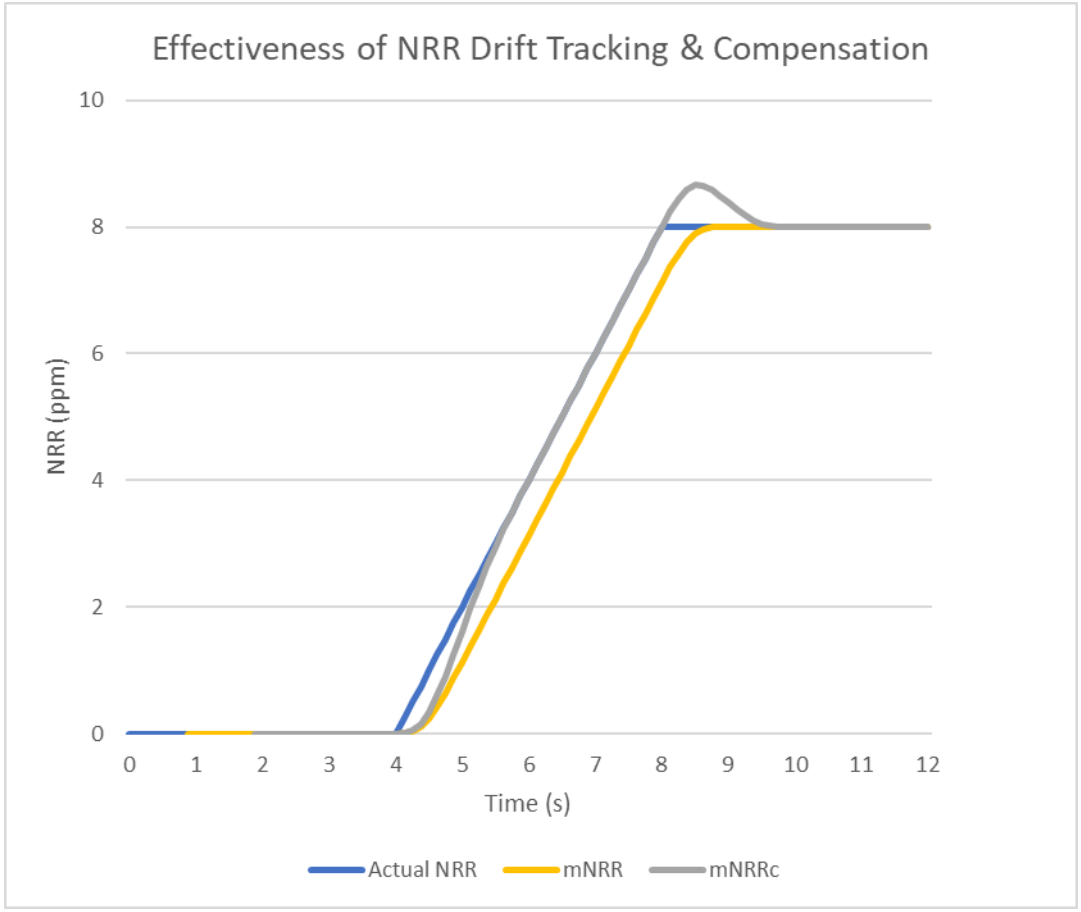
NRR & RR Drift Tracking & Compensation – What are we Measuring?

- **For PTP Relays**: Both *RR(n)* and *correctionField(n)* are in the output Sync message and could be used to measure the effectiveness of the algorithms, but *correctionField(n)* includes more sources of error and the relative magnitude of the algorithms' effect is lower (lower signal to noise ratio for the characteristic we are interested in measuring). **So, use *RR(n)* output in Sync message to measure algorithm effectiveness.**
- **For End Stations**: **Use accuracy of Working Clock @ End Station vs. Working Clock @ GM to measure algorithm effectiveness.** This is directly affected by the End Station's calculation of *RR(n)* but we may not have direct access to *RR(n)* and ultimately it's the accuracy of the Working Clock @ End Station that matters.
- Need a way to separate out performance of NRR drift tracking & compensation vs. RR drift tracking & compensation.
  - Both are required to reach performance goal.

NRR & RR Drift Tracking & Compensation – What are we Measuring?

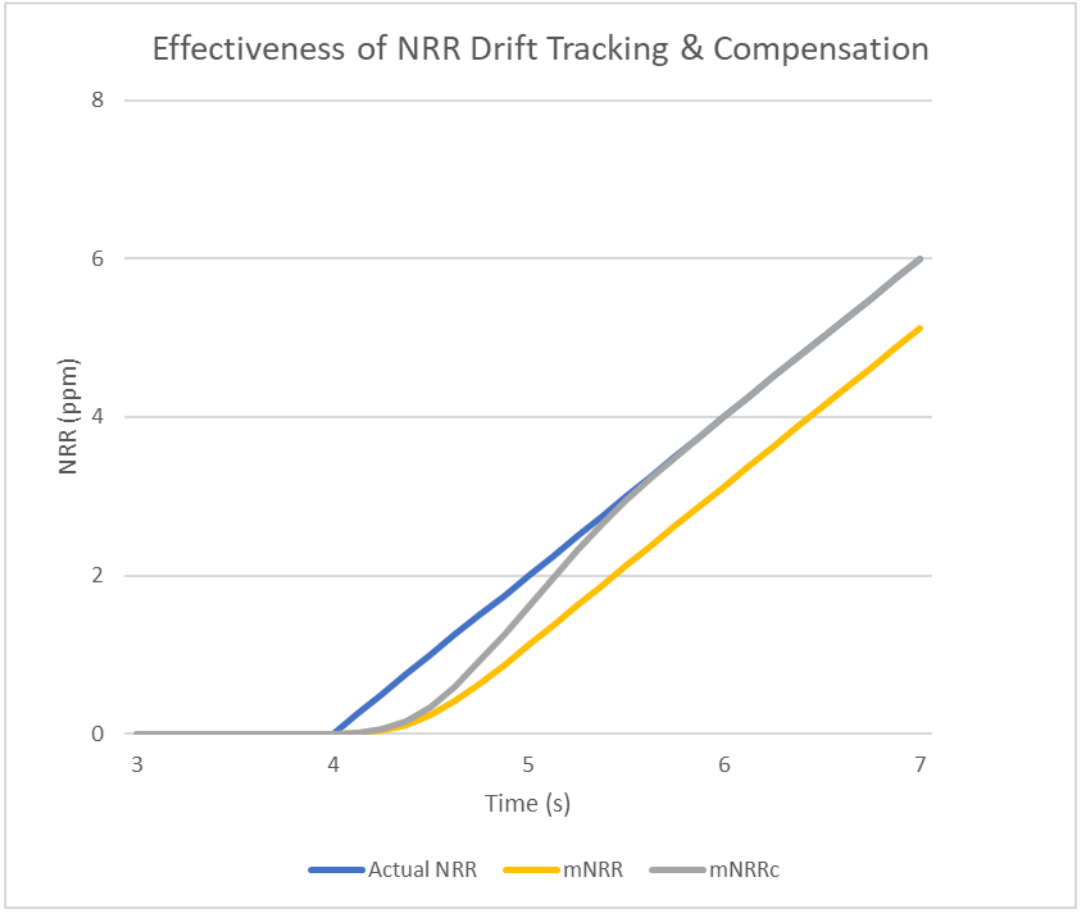
- Simple time series simulation in Excel
  - NRR; Two Devices; One Hop; 12 seconds
  - 125ms Sync Interval; exact, no variation.
  - No Timestamp Errors
  - No Residence Time; looking at NRR calculation on receipt of Sync message, i.e. value will include errors due to drift during measurement if no compensation algorithm is applied.
- Node n clock is nominal and stable throughout
- Node n-1 clock starts (0 s) at nominal and stable; after 4 seconds (4 s) starts to increase above nominal at +2 ppm/s; after another 4 seconds (8 s) remains stable at at +8 ppm above nominal.
- Therefore, NRR starts (0 s) at 0 ppm; after 4 seconds (4 s) starts to increase at 2 ppm/s; after another 4 seconds (8 s) remains stable at +8 ppm.
- Compare NRR calculations with no tracking and compensation (mNRR) vs. tracking and compensation (mNRRc) described in contribution "60802 Time Synchronisation – NRR Tracking & Error Compensation: 1-hop Model; Piece-wise Linear Clock Drift" January 2023, David McCall

NRR & RR Drift Tracking & Compensation – What are we Measuring?



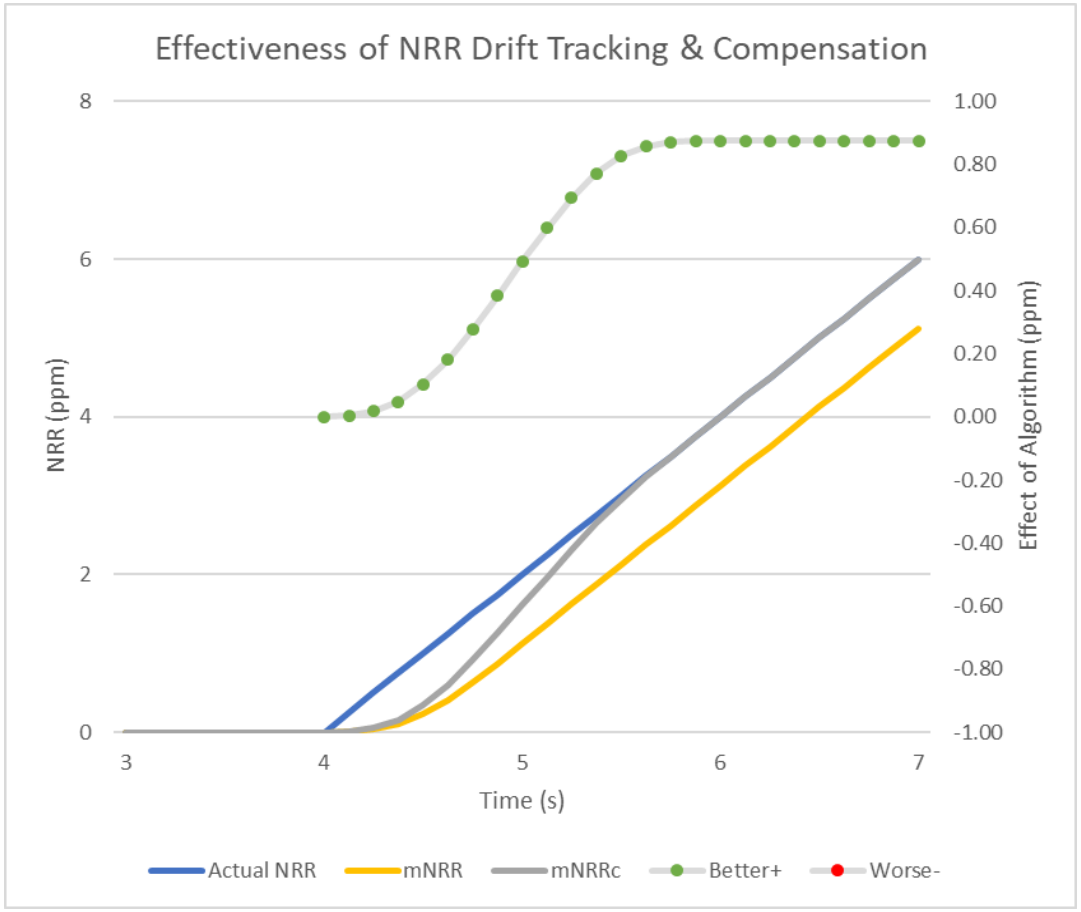
- mNRR – measurement with no compensation starts at receipt of 8th Sync message (at 0.875 s)
- mNRRc – measurement with tracking and compensation algorithm starts at receipt of 16th Sync message (at 1.875 s)
- Zooming in a bit…

NRR & RR Drift Tracking & Compensation – What are we Measuring?



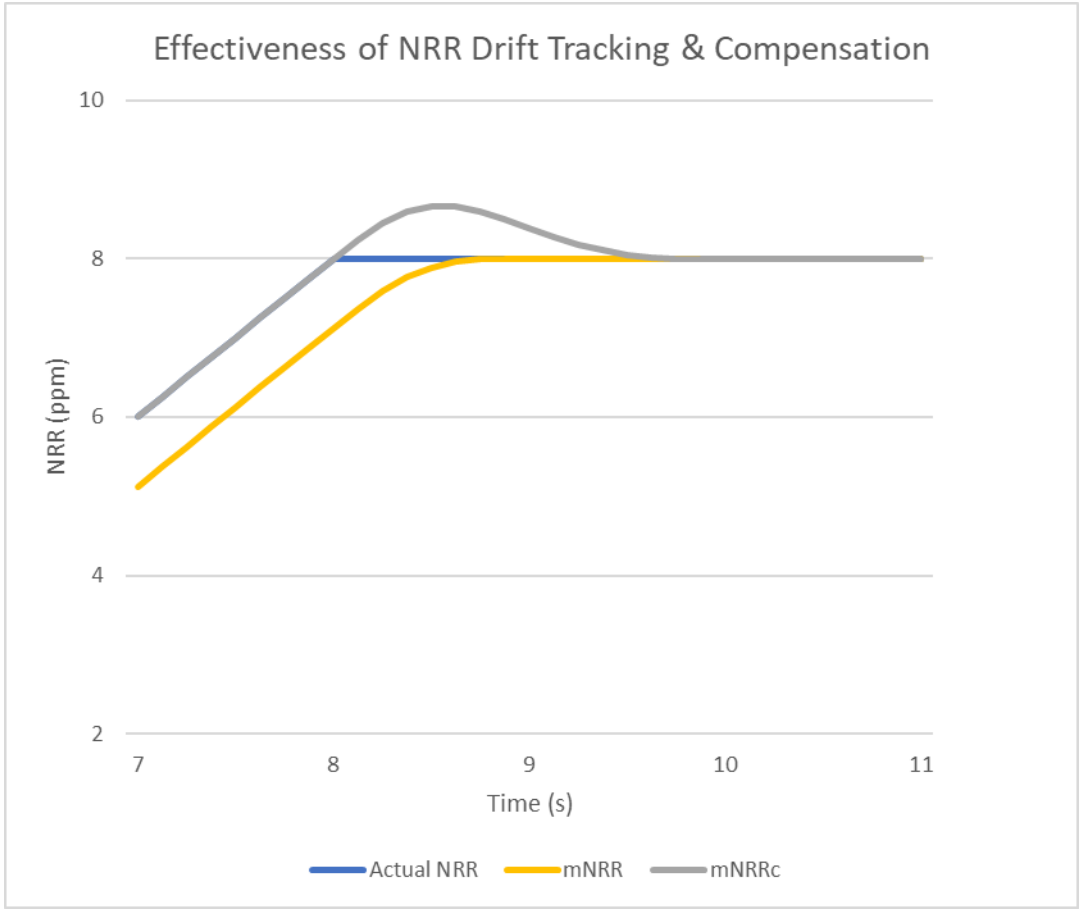
- mNRR – stabalises 1 s after NRR starts rising; always lags due to clock drift during measurement.
- mNRRc – stabalises 2 s after NRR starts rising; algorithm assumes linear clock drift and drift is linear so, when stable, it perfectly compensates.
  - This simple model ignores impact of Timestamp Errors…but NRR calculations are deliberately constructed to trade-off Timestamp Errors against errors due to Clock Drift

NRR & RR Drift Tracking & Compensation – What are we Measuring?



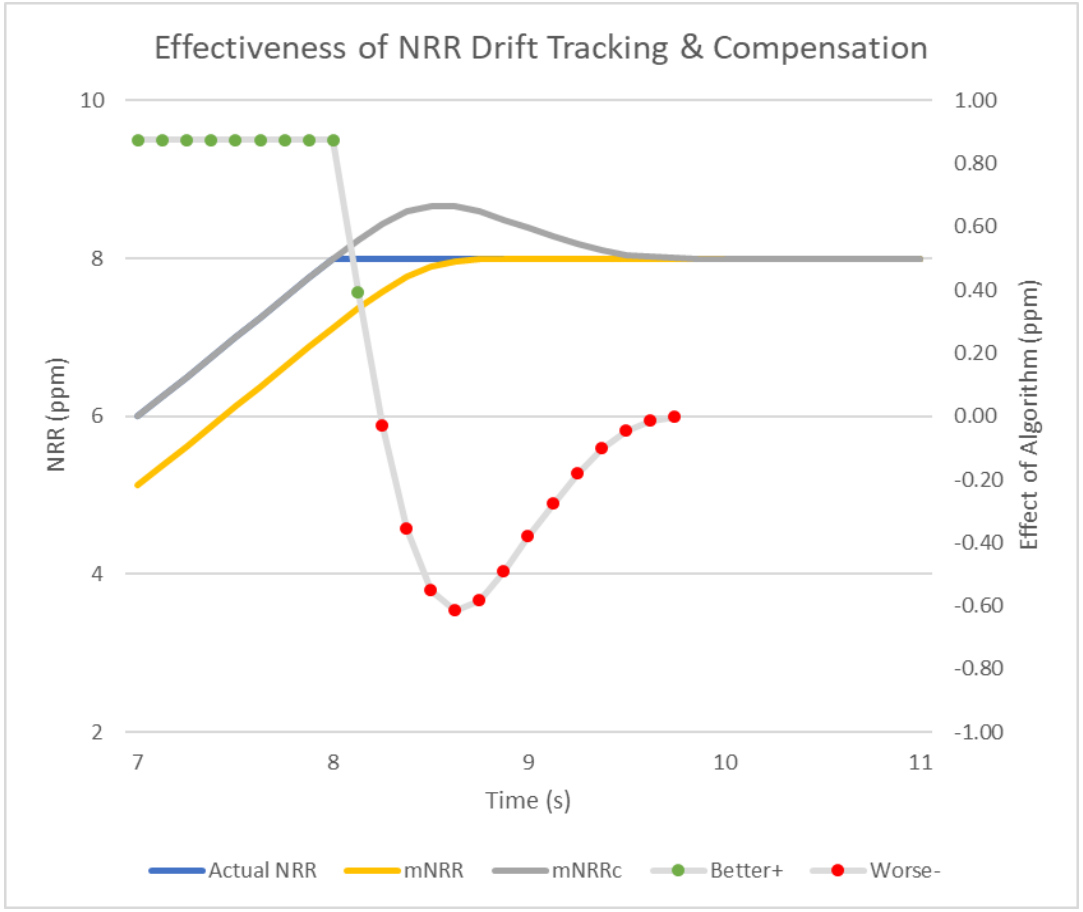
- mNRRc always performs better than mNRR when going from no clock drift to some clock drift or, more generally, when amount of clock drift increases (away from zero).
- Stabalises at its maximum of 0.875 ppm better than mNRR

NRR & RR Drift Tracking & Compensation – What are we Measuring?



- mNRR – stabalises 1 s after NRR stops rising; lags gradually reduces during that interval.
- mNRRc – stabalises 2 s after NRR stops rising; algorithm assumes linear clock drift based on historic measurements, initially applies too much “correction”, and then continues to apply some correction when none is required until old data indicating drift expires at end of interval.

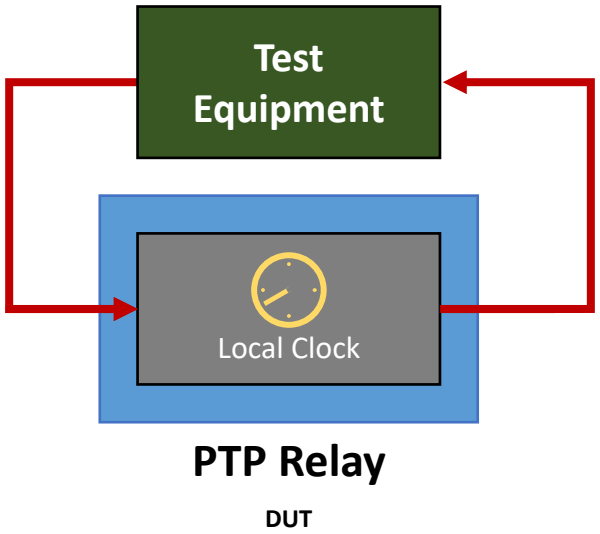
NRR & RR Drift Tracking & Compensation – What are we Measuring?

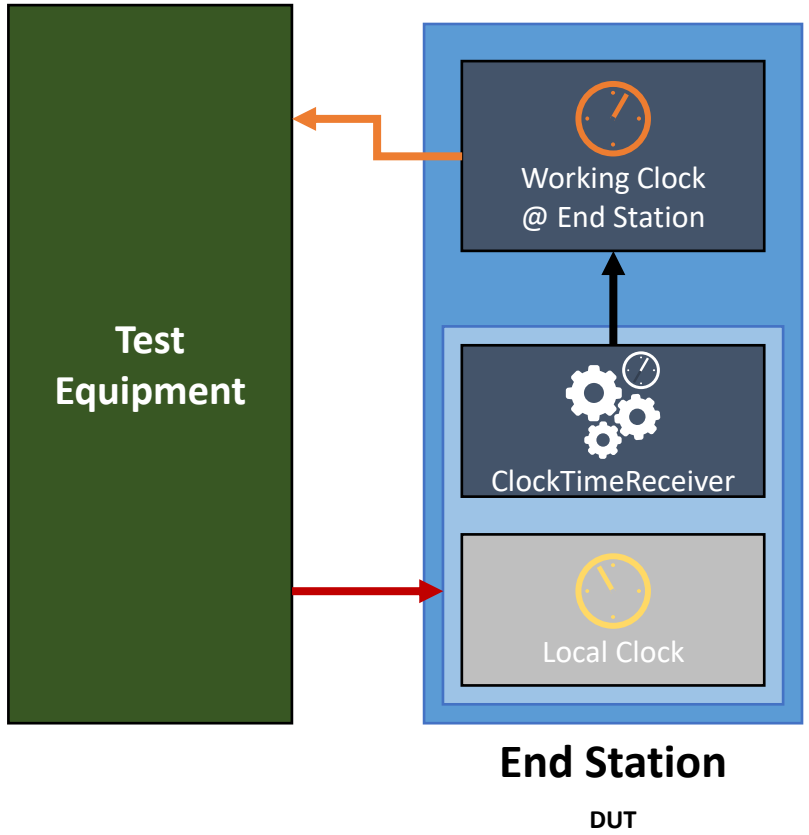


- mNRRc is initially better than mNRR (for first 250 ms after NRR stops rising), but is then worse, until old measurements expire.
- Worst case occurs 625 ms after NRR stops rising: mNRRc is 0.616 ppm less accurate than mNRR.
- Note that clock drift is more likely to suddenly increase in magnitude (due to a sudden change in ambient temperature causing a ramp in crystal temperature) than to suddenly stabalise (due to ambient temperature suddenly matching a ramping crystal temperature).
  - Algorithm performance during the former is more important than during the latter.

What do we need from Test Equipment?

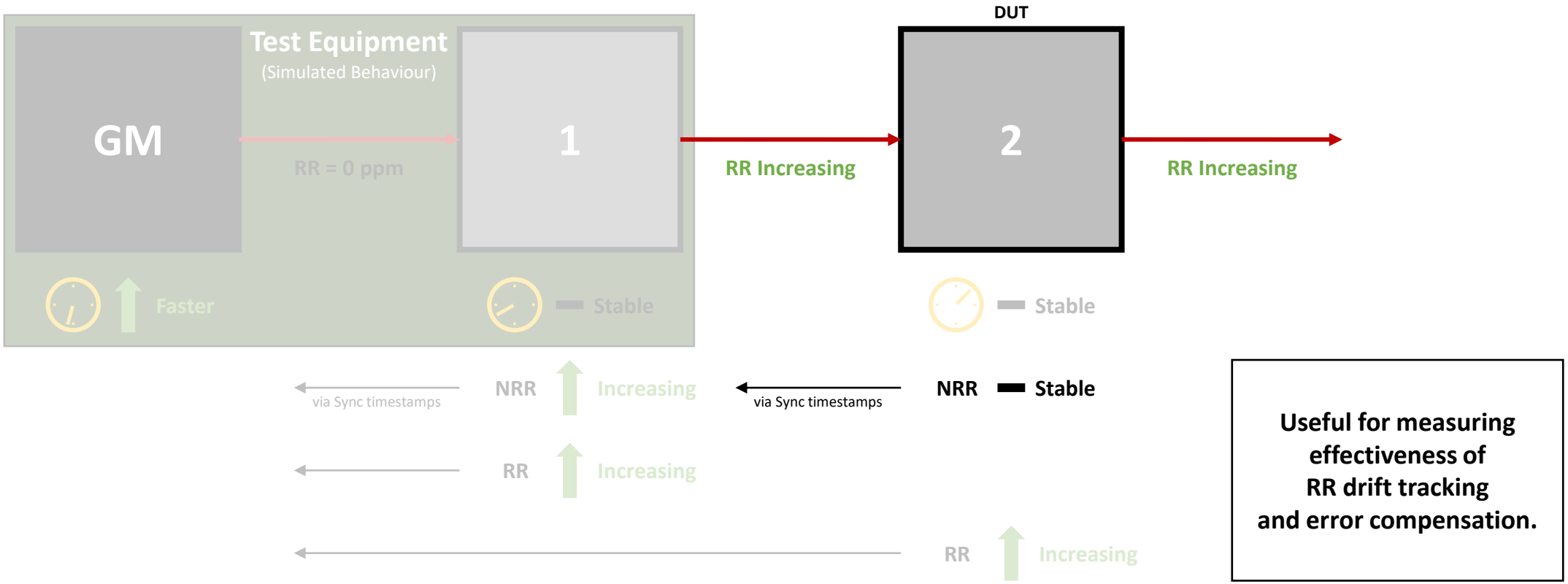
- Is there a way for the test equipment to generate controlled amounts of NRR and RR drift and measure the resulting errors?





Effects of Clock Drift on Rate Ratio - GM

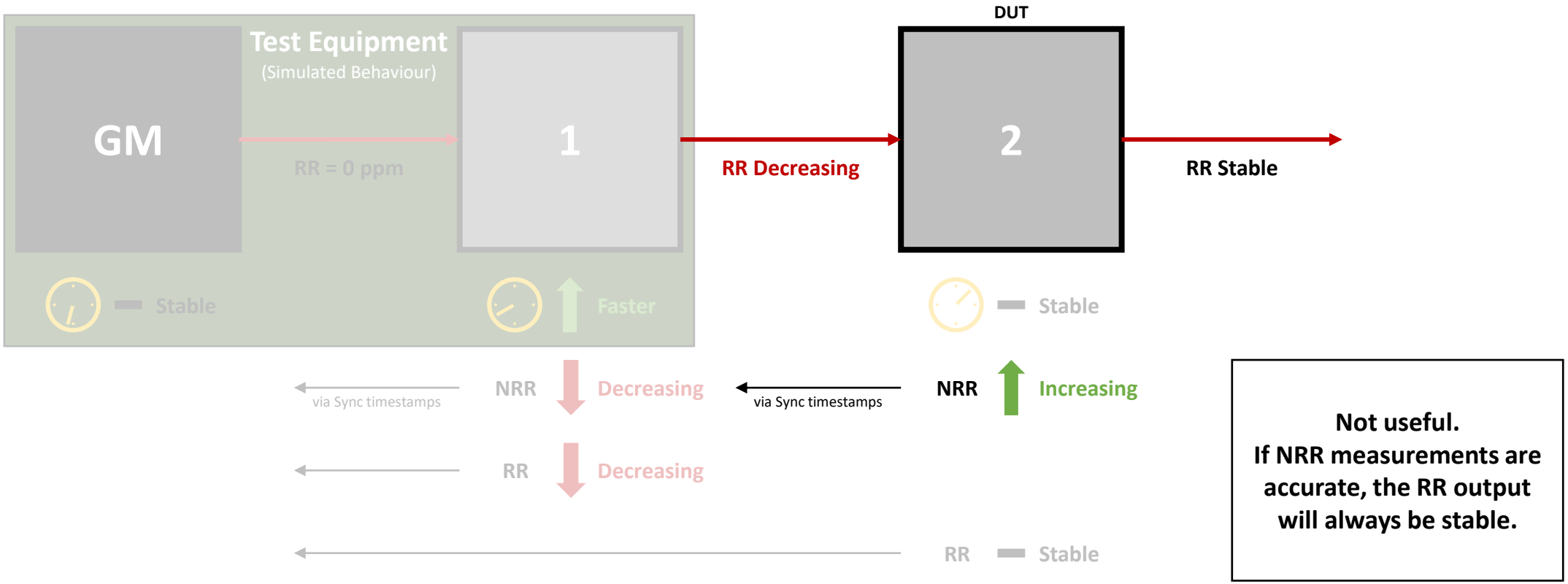
(Assuming zero Timestamp Error and perfect NRR measurements, but no error compensation algorithms.)



Assumes the Working Clock @ GM and Local Clock @ GM are the same clock in this implementation. If not, Local Clock @ GM effectively becomes Node 1.

Effects of Clock Drift on Rate Ratio – Node 1

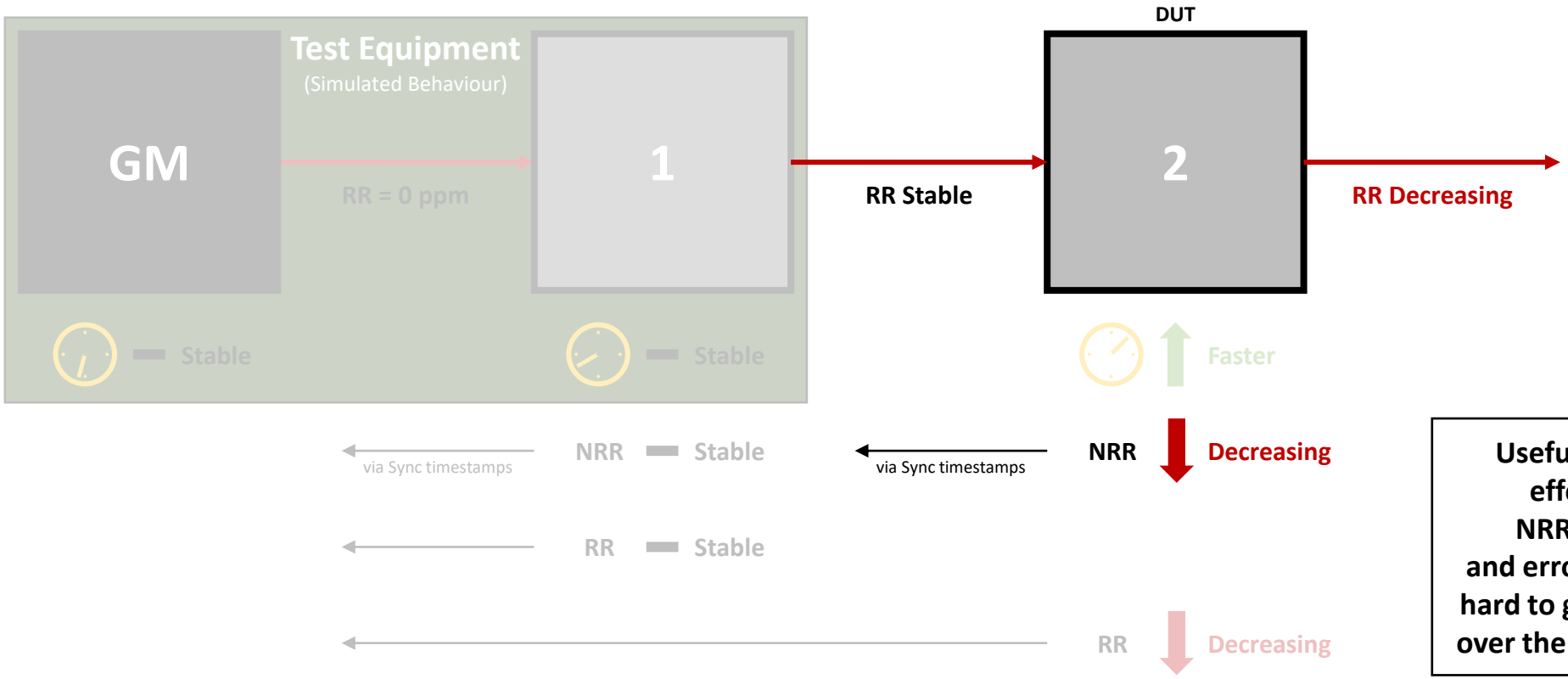
(Assuming zero Timestamp Error and perfect NRR measurements, but no error compensation algorithms.)



Assumes the Working Clock @ GM and Local Clock @ GM are the same clock in this implementation. If not, Local Clock @ GM effectively becomes Node 1.

Effects of Clock Drift on Rate Ratio – Node 2

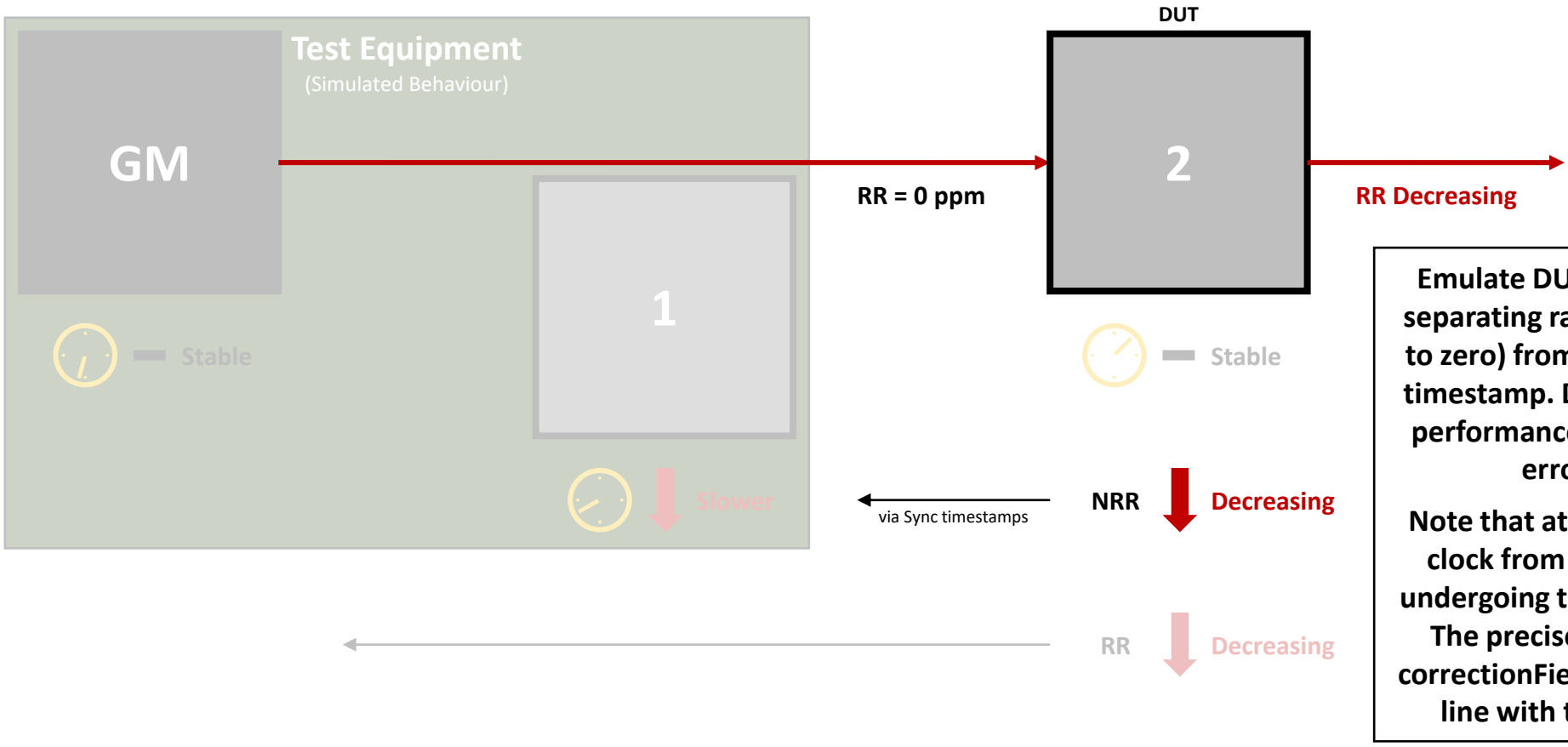
(Assuming zero Timestamp Error and perfect NRR measurements, but no error compensation algorithms.)



Assumes the Working Clock @ GM and Local Clock @ GM are the same clock in this implementation. If not, Local Clock @ GM effectively becomes Node 1.

Effects of Clock Drift on Rate Ratio – Test Setup

(Assuming zero Timestamp Error and perfect NRR measurements, but no error compensation algorithms.)



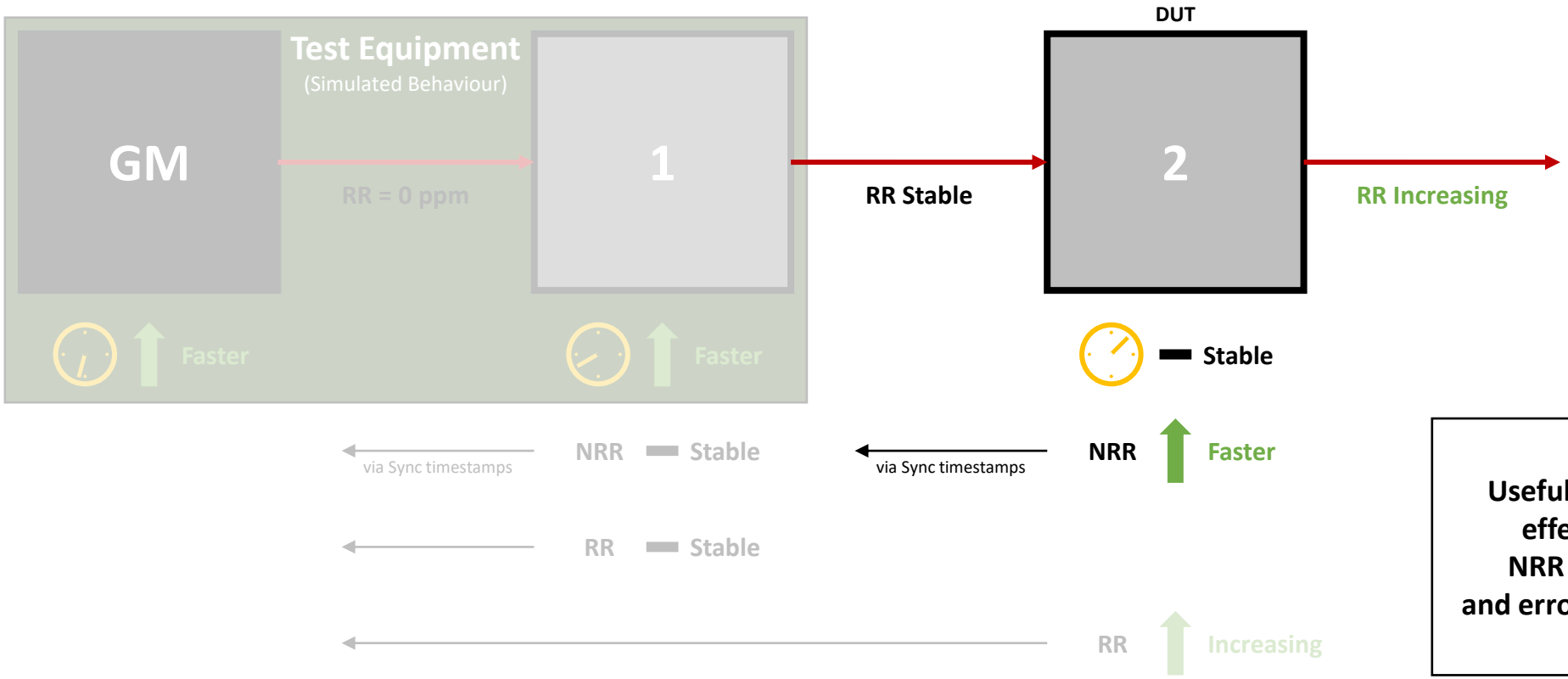
Emulate DUT Local Clock drift by separating rateRatio field drift (set to zero) from Sync message egress timestamp. Drift latter to measure performance of NRR tracking and error correction.

Note that attempting to recover a clock from the output of a DUT undergoing this test may not work. The preciseOriginTimestamp + correctionField may not progress in line with the DUT output RR.

Assumes the Working Clock @ GM and Local Clock @ GM are the same clock in this implementation. If not, Local Clock @ GM effectively becomes Node 1.

Effects of Clock Drift on Rate Ratio - GM

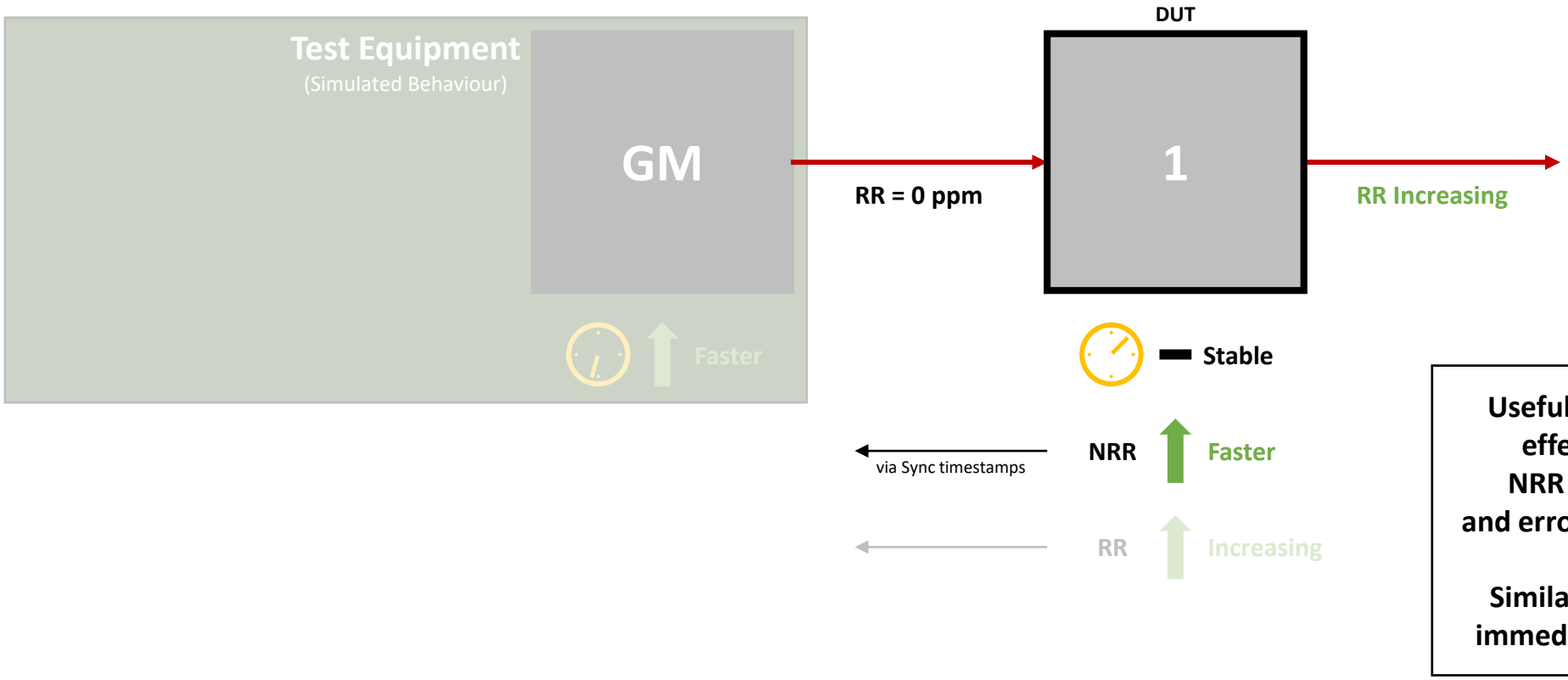
(Assuming zero Timestamp Error and perfect NRR measurements, but no error compensation algorithms.)



Assumes the Working Clock @ GM and Local Clock @ GM are the same clock in this implementation. If not, Local Clock @ GM effectively becomes Node 1.

Effects of Clock Drift on Rate Ratio - GM

(Assuming zero Timestamp Error and perfect NRR measurements, but no error compensation algorithms.)



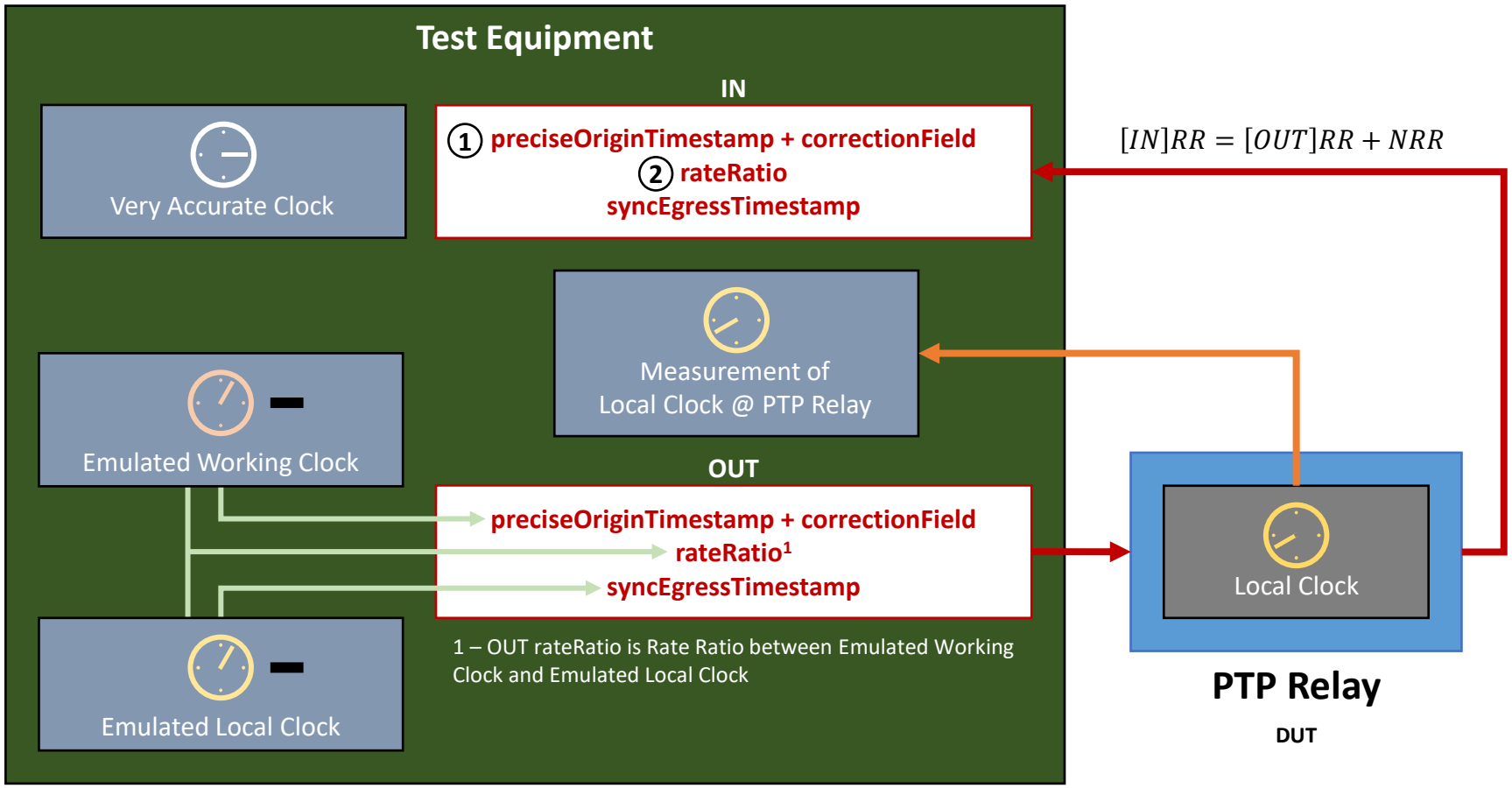
Useful for measuring effectiveness of NRR drift tracking and error compensation.

Similar to DUT being immediately after GM.

Assumes the Working Clock @ GM and Local Clock @ GM are the same clock in this implementation. If not, Local Clock @ GM effectively becomes Node 1.

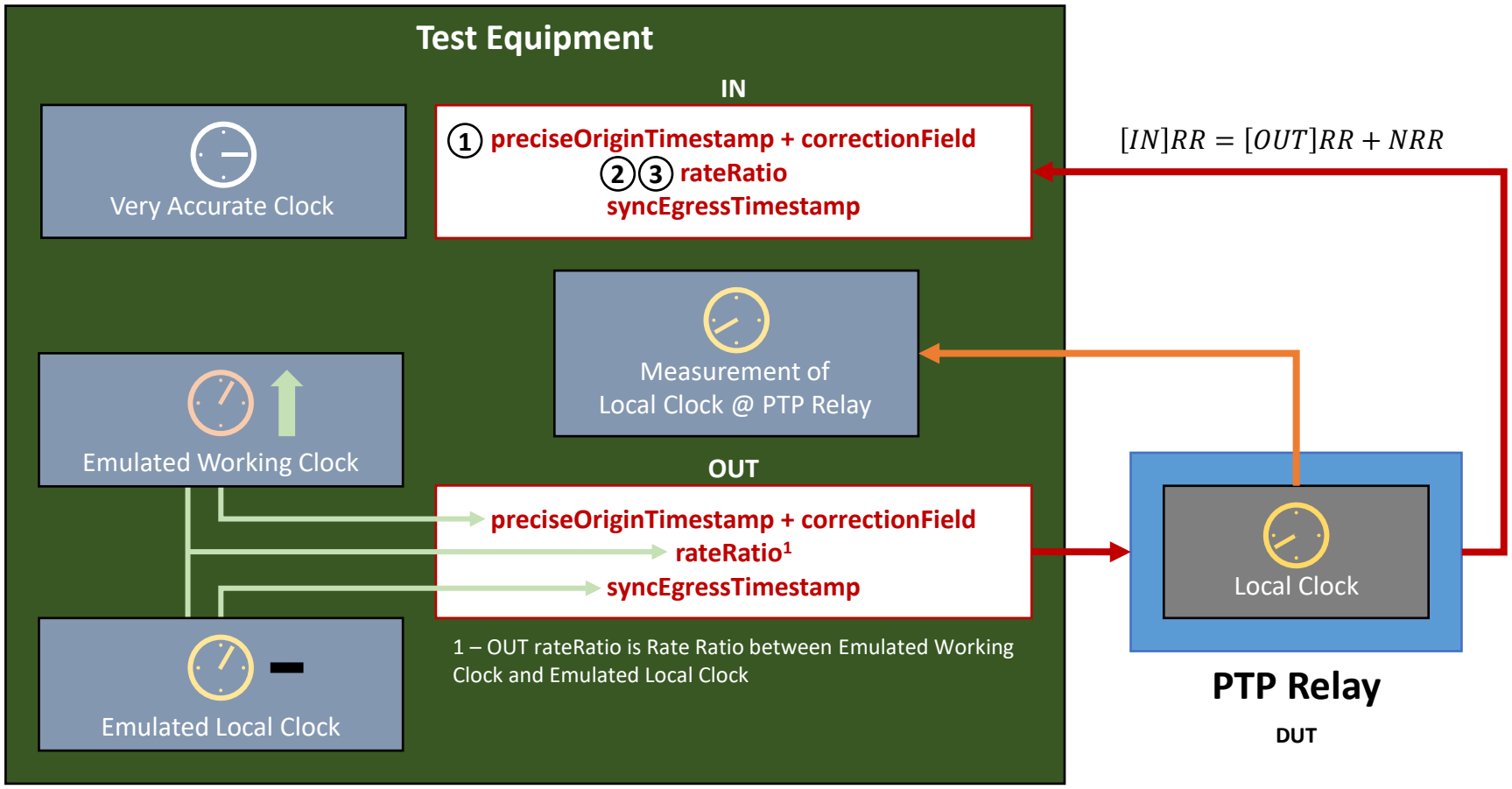
Measuring Error Generation at PTP Relay

Measuring Error Generation at PTP Relay



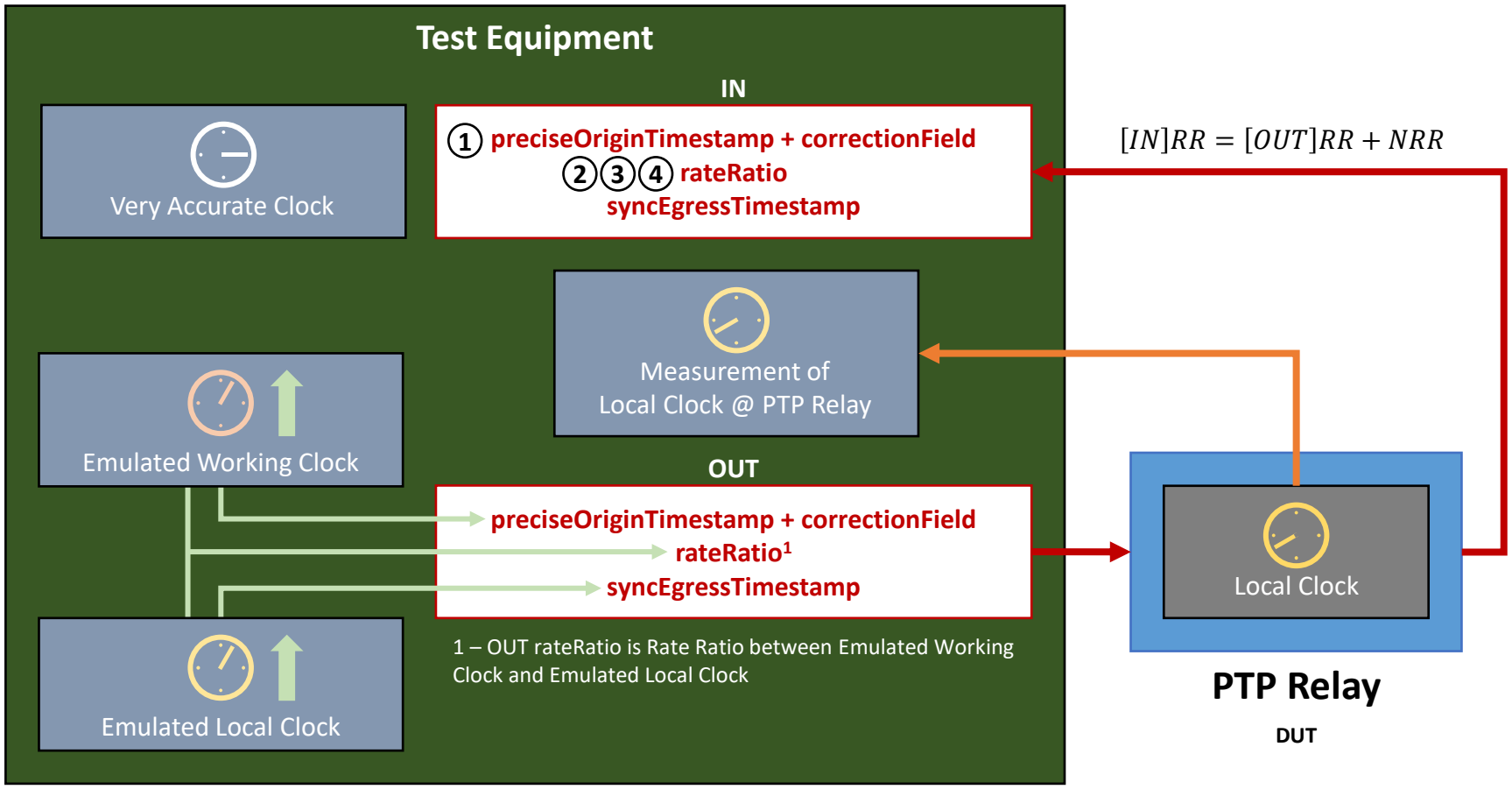
	Emulated Working Clock: Stable	Emulated Local Clock: Stable
1	**[IN] preciseOriginTimestamp + correctionField** …should equal (within acceptable margin)… **Measured Egress Time of [IN] Sync Message (measured against Emulated Working Clock)**	
2	**[IN] rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio ([OUT] rateRatio plus Rate Ratio between Emulated Local Clock and Measurement of Local Clock @ PTP Relay)**	

Measuring Error Generation at PTP Relay



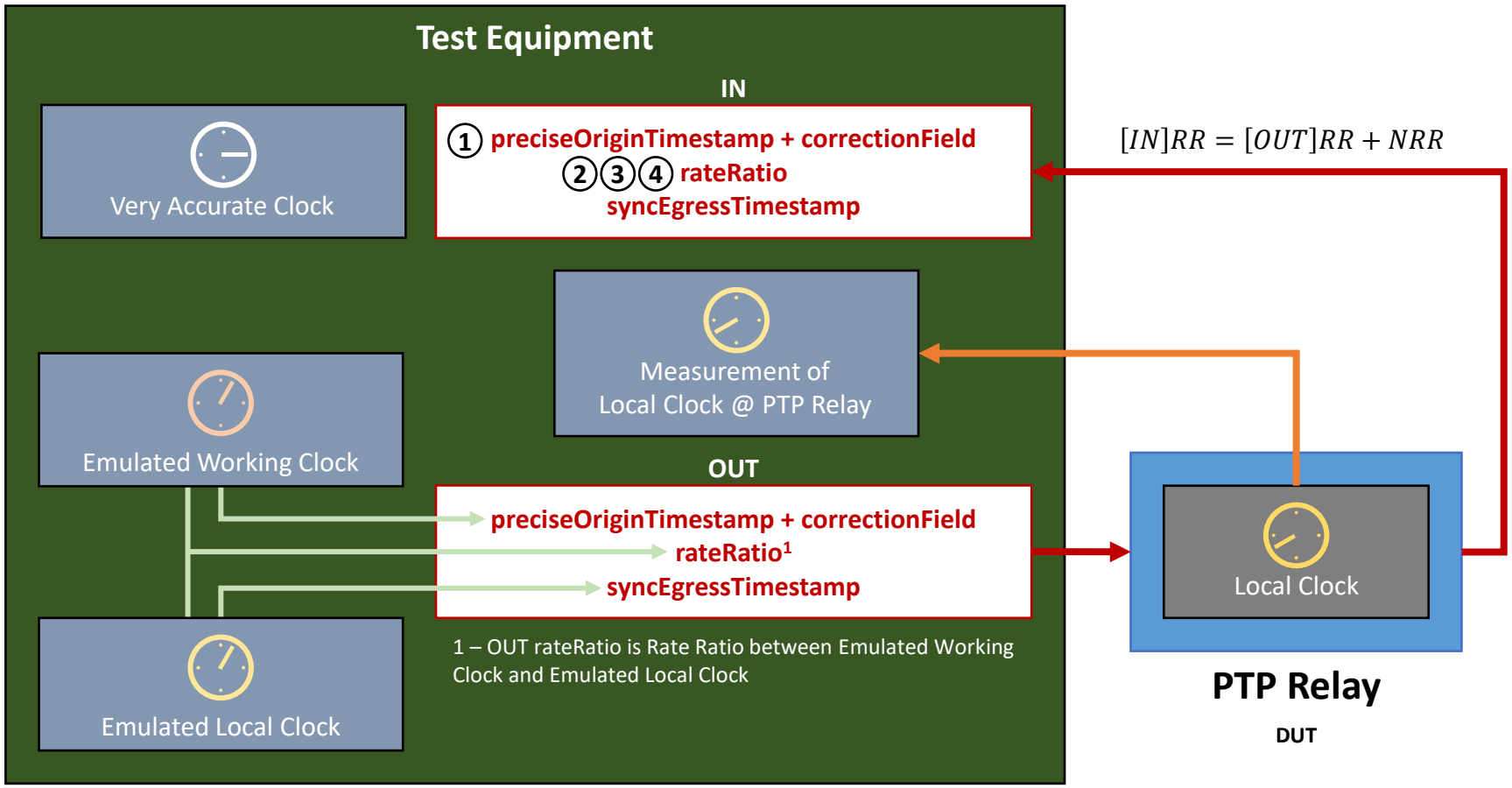
Emulated Working Clock: Offset Increasing	Emulated Local Clock: Stable
3	**[IN] rateRatio** …should equal (within acceptable margin)… **Measured Rate Ratio (OUT rateRatio plus Rate Ratio between Emulated Local Clock and Measurement of Local Clock @ PTP Relay)**

Measuring Error Generation at PTP Relay



	Emulated Working Clock: Offset Increasing	Emulated Local Clock: Offset Increasing
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)**	

Measuring Error Generation at PTP Relay



$$[IN]RR = [OUT]RR \times mNRR$$

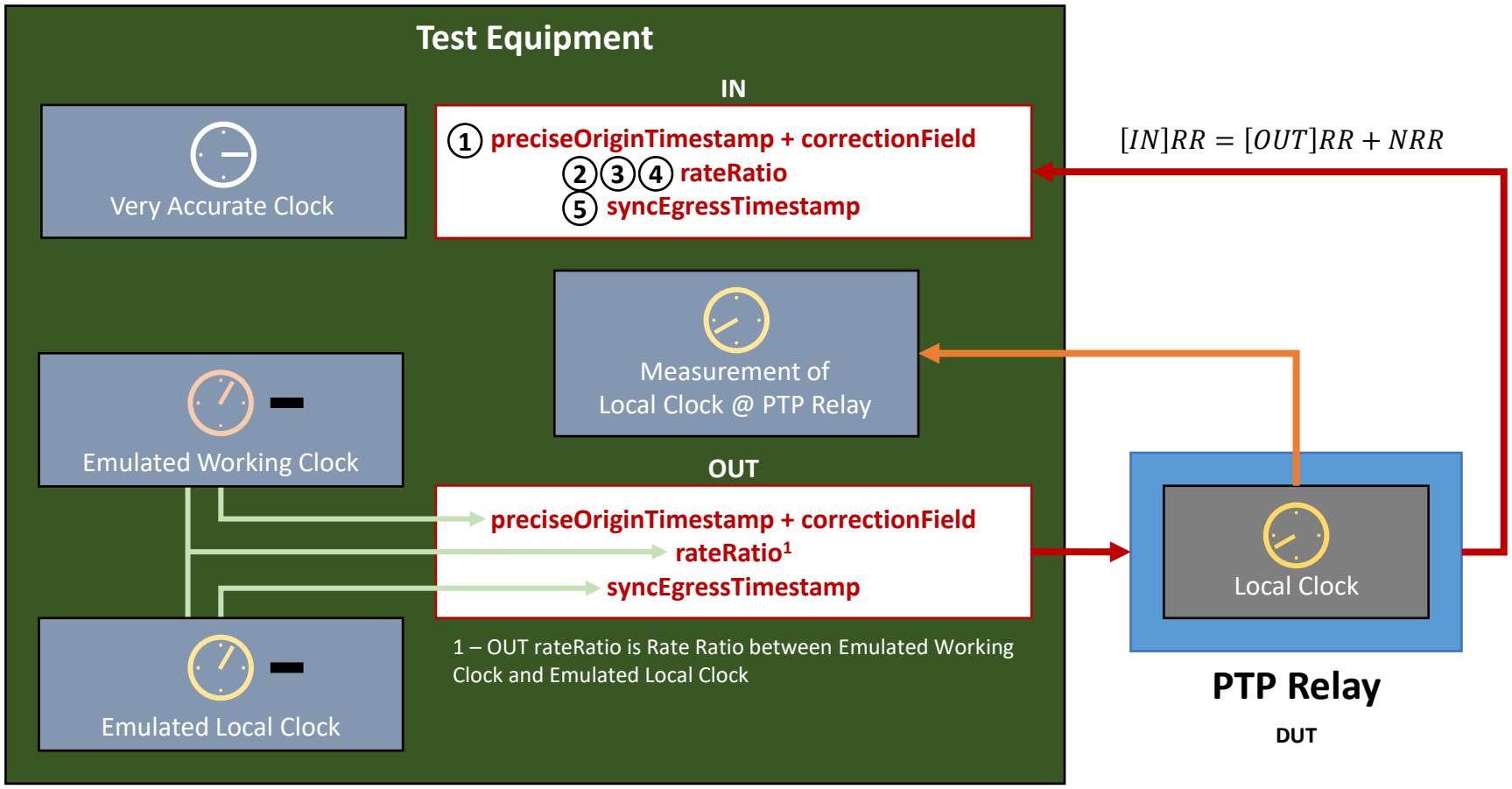
$$[OUT]RR = \frac{\boldsymbol{EmulatedWorking}}{\boldsymbol{Emulated\ Local}}$$

$$mNRR = \frac{\boldsymbol{EmulatedLocal}}{\boldsymbol{Local}}$$

$$[IN]RR = \frac{\boldsymbol{EmulatedWorking}}{\boldsymbol{Emulated\ Local}} \times \frac{\boldsymbol{EmulatedLocal}}{\boldsymbol{Local}}$$

Error measurement compares DUT's best estimate of [IN]RR (via IN Sync message) with Test Equipment's more accurate view of the same value
replacing "Local Clock" with "Measurement of Local Clock @ PTP Relay"

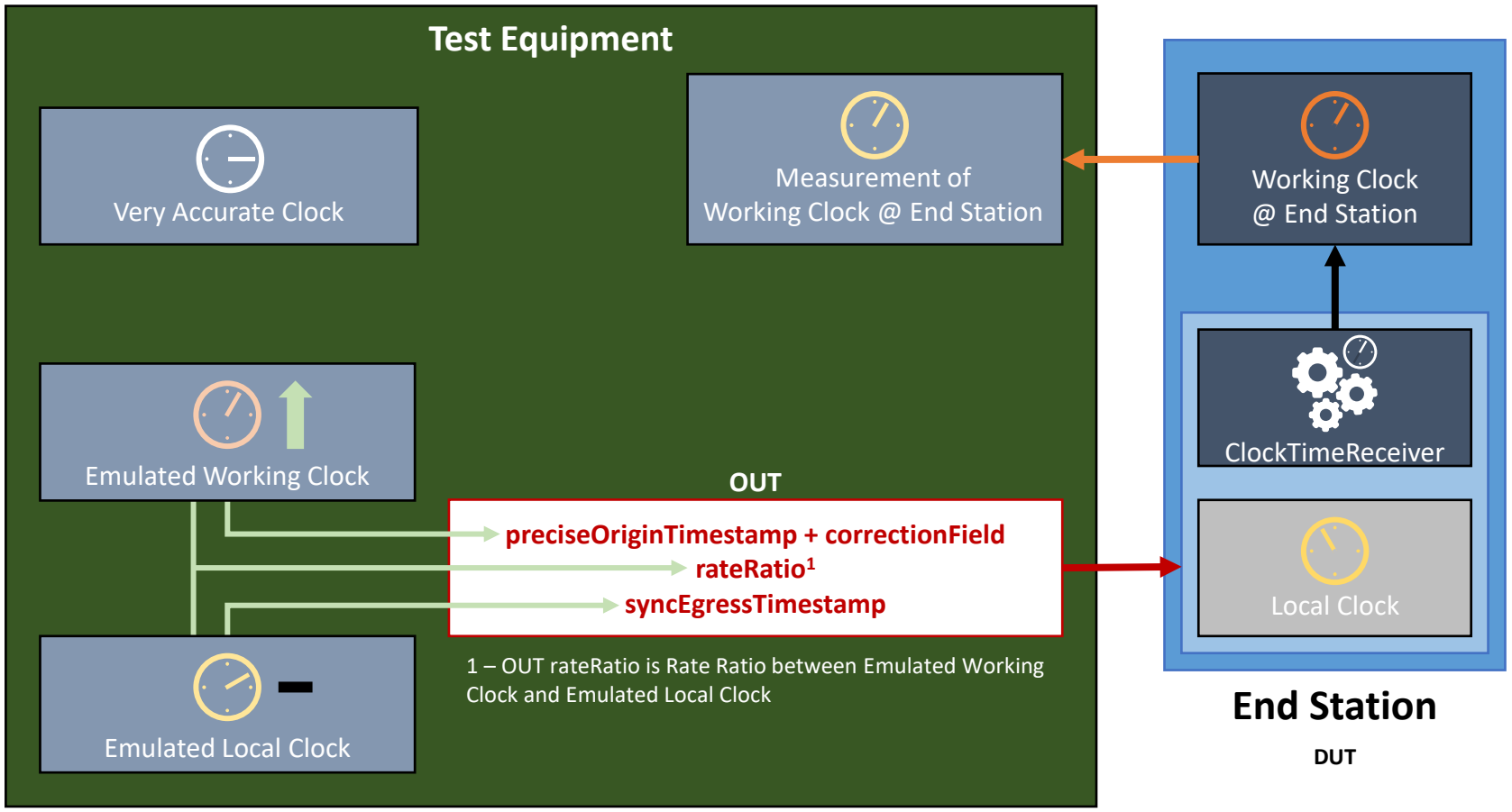
Measuring Error Generation at PTP Relay



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

Measuring Error Generation at End Station

Measuring Error Generation at End Station



End Station Drives Working Clock @ End Station with the goal that…

$$RR_{EndStation} = \frac{WorkingClock_{EndStation}}{LocalClock_{EndStation}} = \frac{WorkingClock_{GM}}{LocalClock_{EndStation}}$$

$RR_{EndStation}$ is calculated by End Station…

$$RR_{EndStation} = [OUT]RR \times mNRR$$

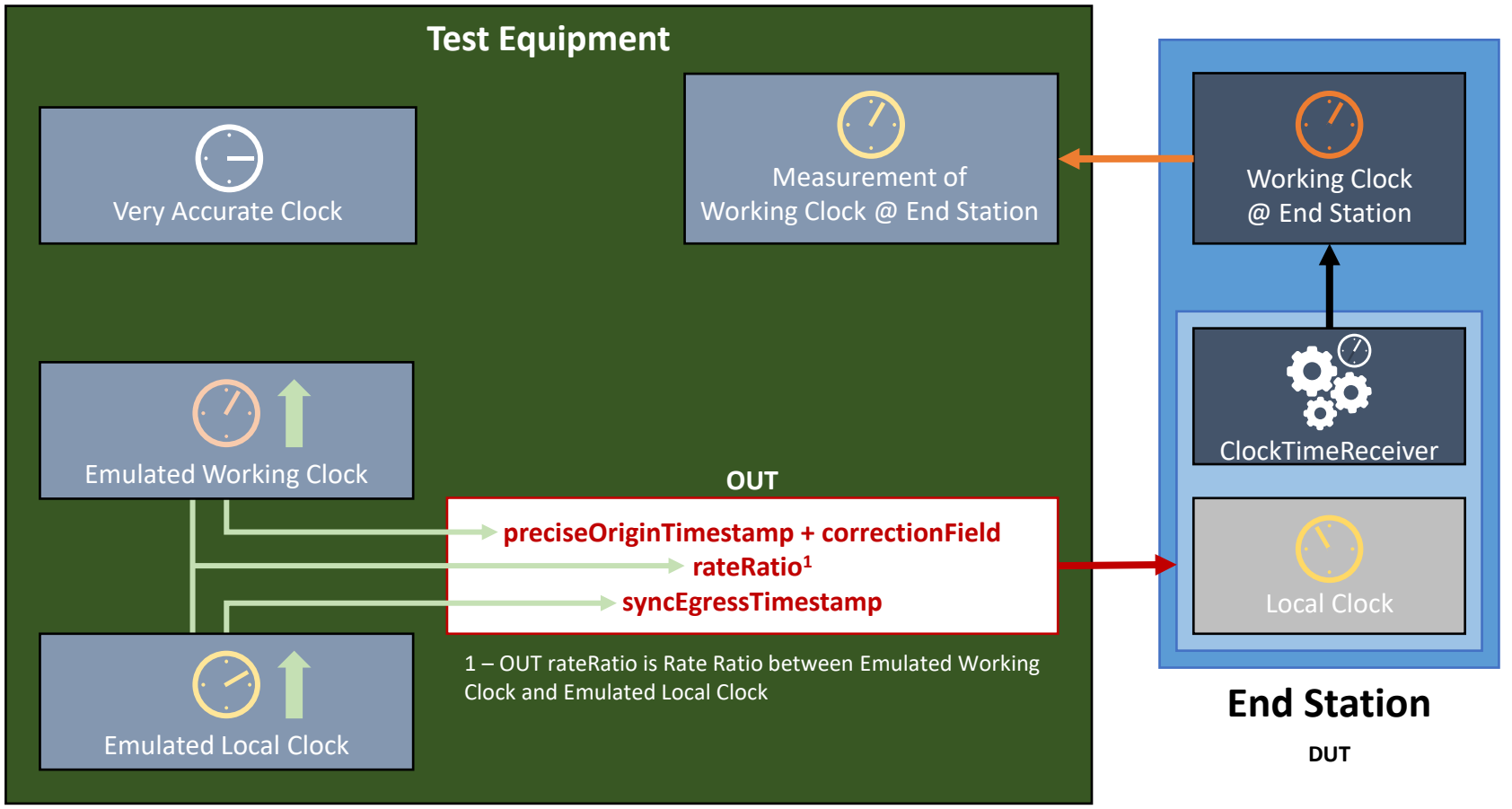
$$RR_{EndStation} = \frac{\boldsymbol{EmulatedWorking}}{\boldsymbol{EmulatedLocal}} \times \frac{\boldsymbol{EmulatedLocal}}{\boldsymbol{Local}}$$

If Emulated Local Clock is stable and Emulated Working Clock is drifting…

$$RR_{EndStation} = \underbrace{\frac{\boldsymbol{EmulatedWorking}}{\boldsymbol{EmulatedLocal}}}_{\text{Drifting}} \times \underbrace{\frac{\boldsymbol{EmulatedLocal}}{\boldsymbol{Local}}}_{\textbf{Stable}}$$

…so accuracy of Working Clock @ End Station (vs. Emulated Working Clock) will depend on RR drift tracking & error correction

Measuring Error Generation at End Station



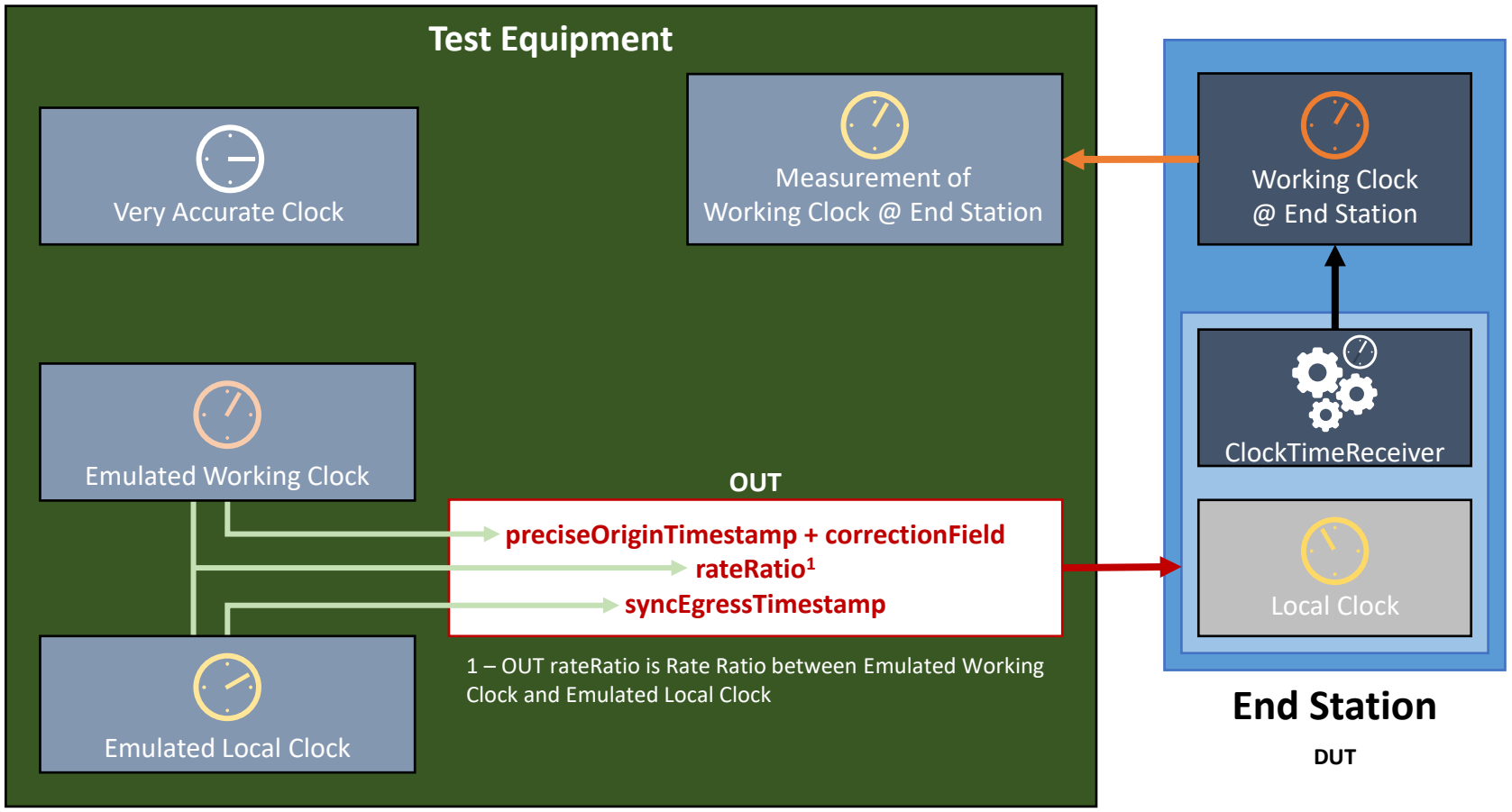
If Emulated Local Clock is and Emulated Working Clock are both drifting in sync…

$$RR_{EndStation} = \frac{EmulatedWorking}{EmulatedLocal} \times \frac{EmulatedLocal}{Local}$$

Stable | Drifting

…so accuracy of Working Clock @ End Station (vs. Emulated Working Clock) will depend on NRR drift tracking & error correction

Measuring Error Generation 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**

Why not just measure Time Error?

Reasons to for these measurements and normative requirements…

- 60802 approach takes advantage of the way errors due to clock drift…
  - Can cancel out, node-to-node, at a system level.
  - Can be compensated for by tracking at a system level.
- The system is set up in a way to bias errors away from timestamp errors and towards errors due to clock drift, so that the latter can cancel out and be compensated for at a system level.
- A Time Error measurement of a Grandmaster or PTP relay merges errors due to timestamp error and due to clock drift.
  - A device could pass due to very low clock drift (via a TCXO for example) but with high timestamp error…which can not be addressed at a system level. **A network of such devices would not achieve the performance goals.**

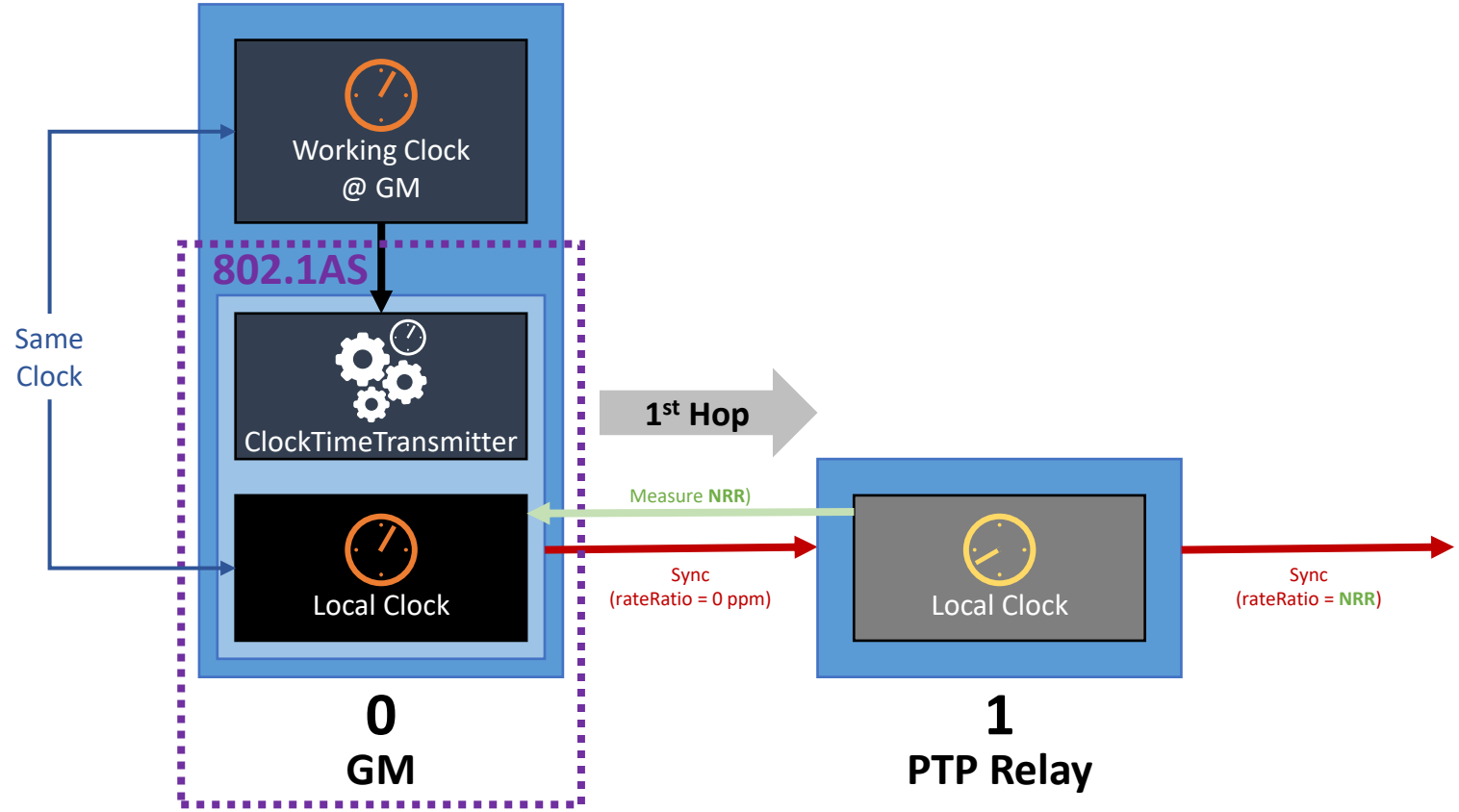
Summary

Summary

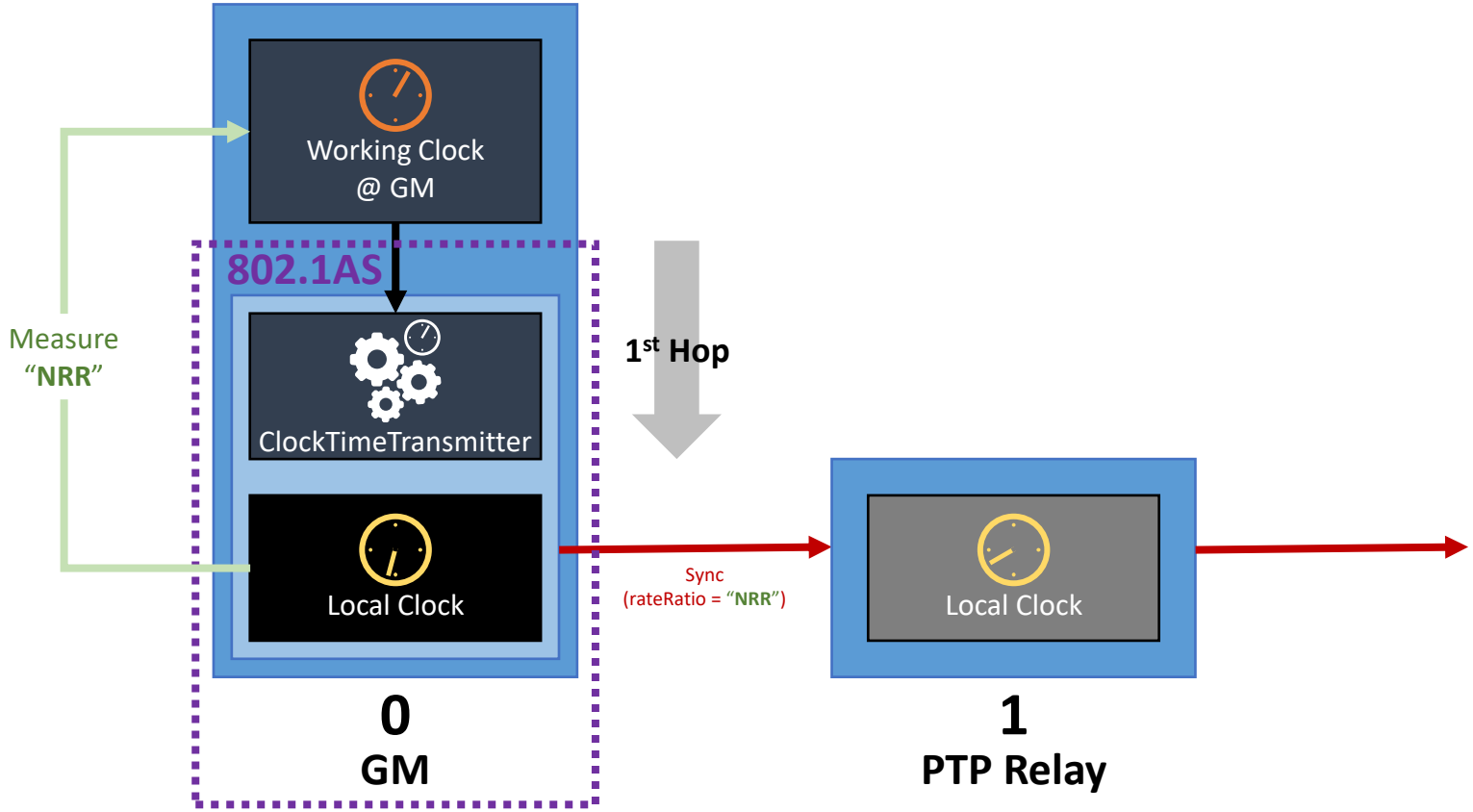
- 60802's approach to achieving performance goals requires a new approach to device-level performance testing in order to ensure system-level performance.
- The required device-level behaviours can be tested and current testers already have the required capabilities and gather a lot of the necessary information.
  - Needs to be validated, especially with regard to emulating Working Clock and Local Clock independently; measuring against a Target Working Clock.
- Normative requirements can be written against these testable behaviours.

Clock Drift Tracking at the Grandmaster

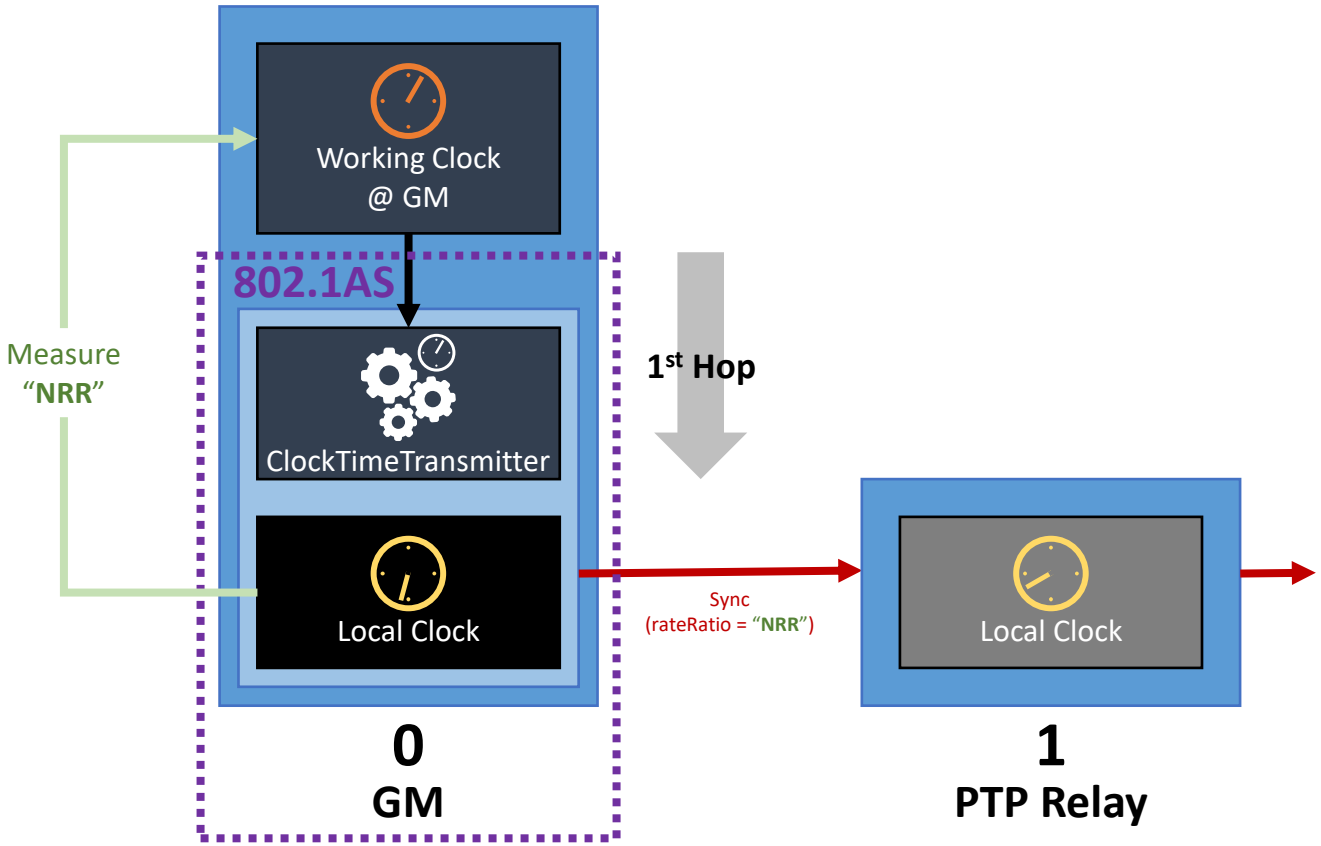
GM: Working & Local Clocks the Same…



GM: Working & Local Clocks Different…



GM: Working & Local Clocks Different…



- Errors calculating rateRatio during an internal 1st hop are just as impactful as if the 1st hop is external
- Depending on implementation, there may be many more options for more accurate calculation of rateRatio, and keeping it within acceptable limits.
  - Device manufacturers should be aware of this and create implementations that perform well.
- Very hard for test equipment to emulate drift of Working Clock at GM
- No way to manipulate an equivalent of the [OUT]rateRatio field.
- **Recommendation**: carry out simulation work to determine if drift tracking is necessary to meet performance goals. If so, include the required performance in existing normative requirements for GM (i.e. how closely rateRatio field must track actual Rate Ratio between Working and Local Clocks @ GM).

Thank you!