60802 Time Sync – Clocks Relationships & Normative Requirements

David McCall – Intel Corporation

Version 1

ClockSource/Target & timeTransmitter/Receiver APIs

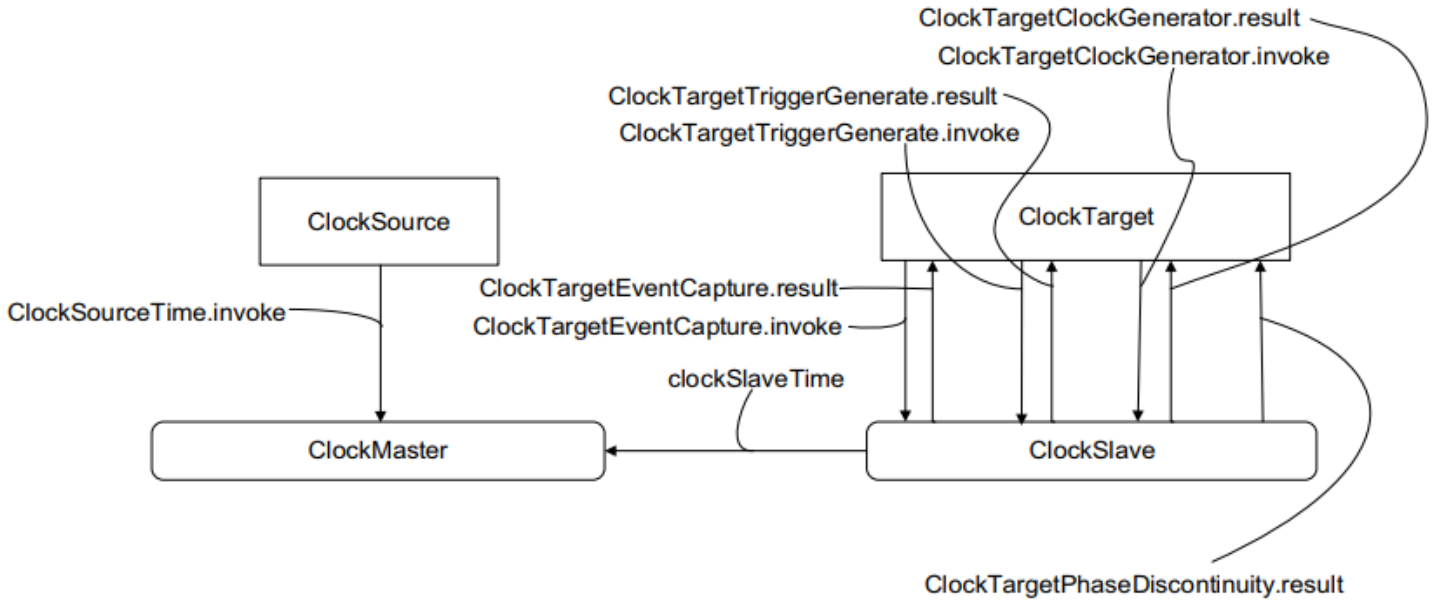


Figure 9-1—Application interfaces

timeTransmitter/Receiver & Local Clock

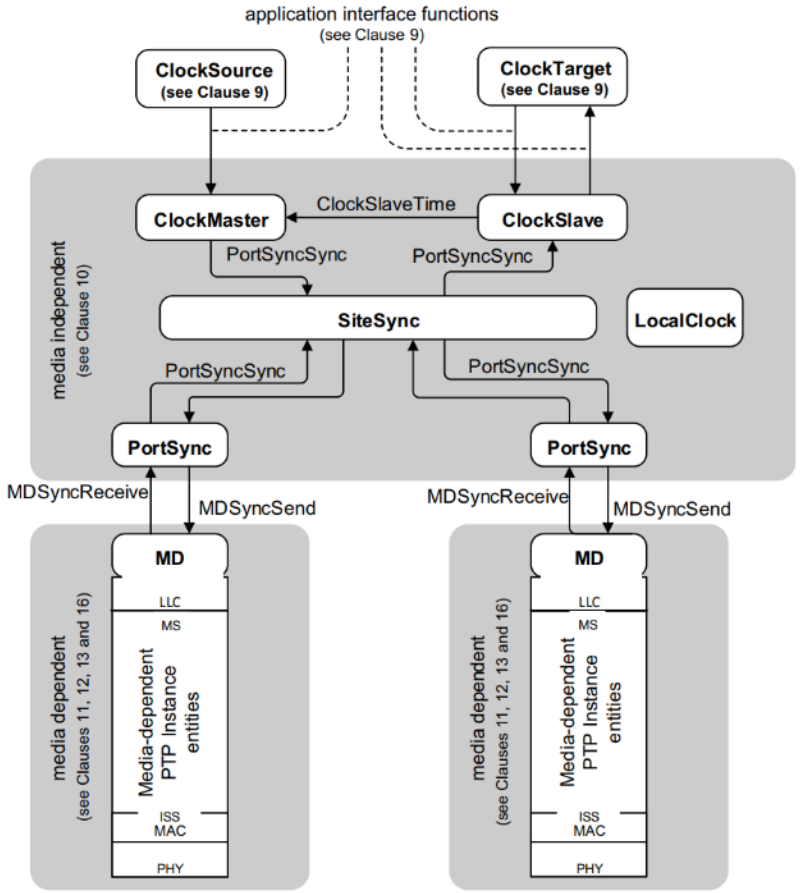
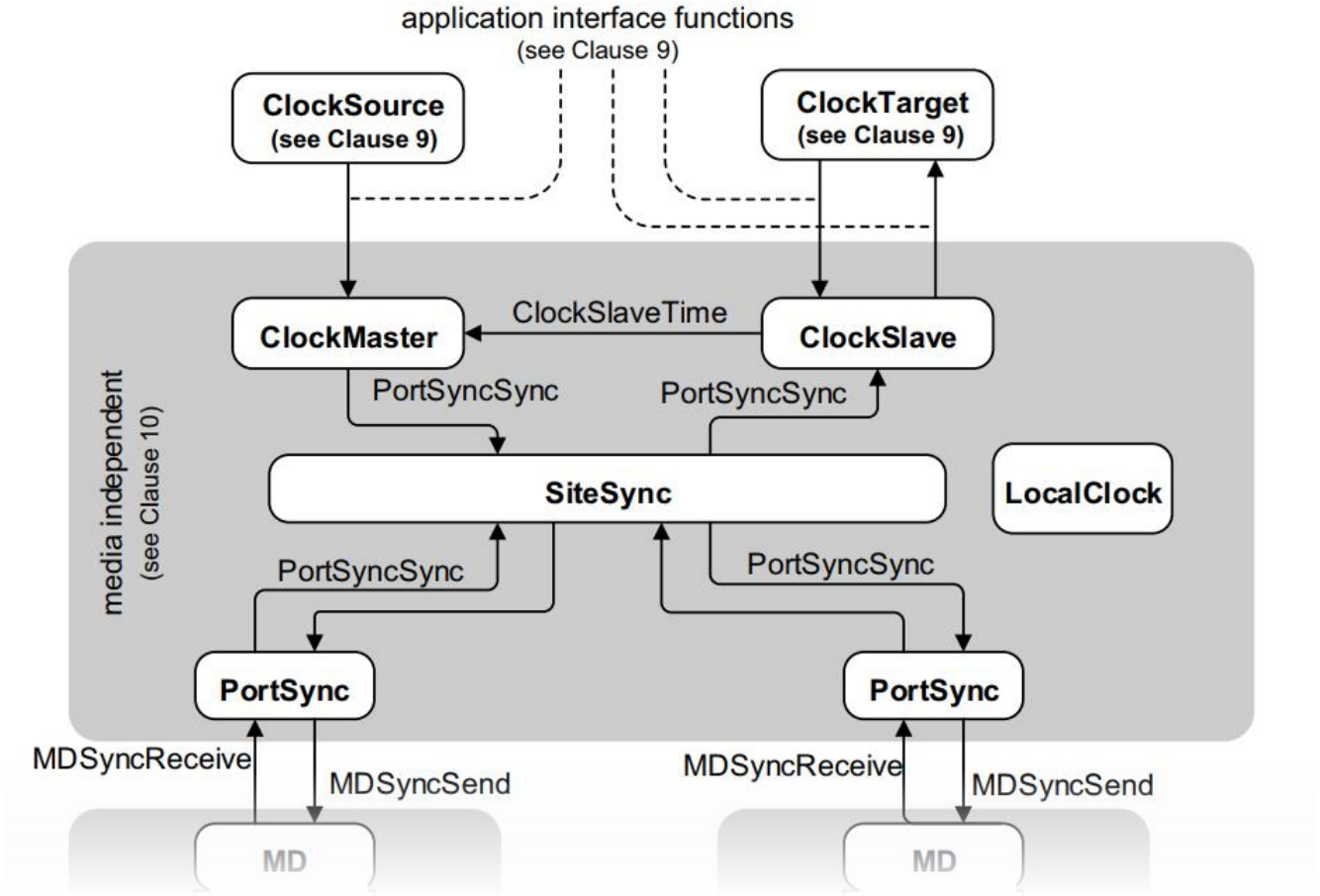


Figure 10-1—Model for media-independent layer of PTP Instance

timeTransmitter/Receiver & Local Clock



802.1AS-2020 10.2.13 Clock~~Slave~~ReceiverSync state machine

10.2.13.2 State machine functions

10.2.13.2.1 updateSlaveTime(): Updates the global variable clockSlaveTime (see 10.2.4.3), based on information received from the SiteSync and LocalClock entities. It is the responsibility of the application to filter slave times appropriately (see B.3 and B.4 for examples). As one example, clockSlaveTime can be:

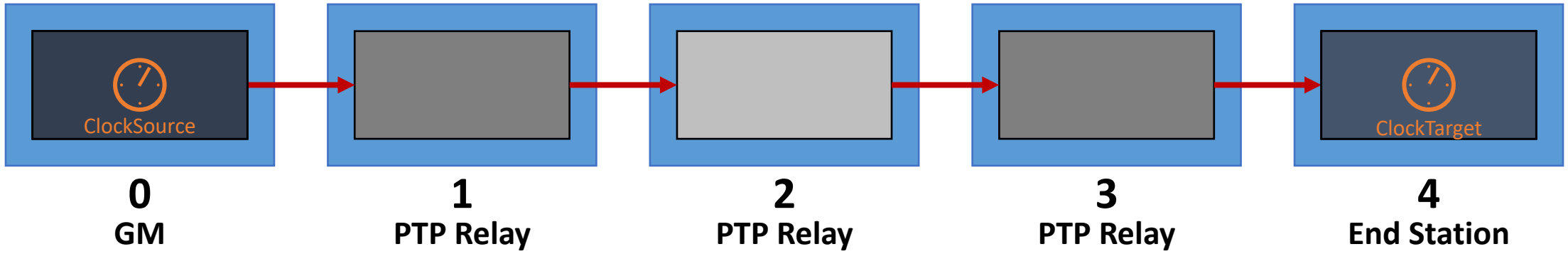
a) Set to syncReceiptTime at every LocalClock update immediately after a PortSyncSync structure is received, and

b) Incremented by localClockTickInterval (see 10.2.4.18) multiplied by the rateRatio member of the previously received PortSyncSync structure during all other LocalClock updates.

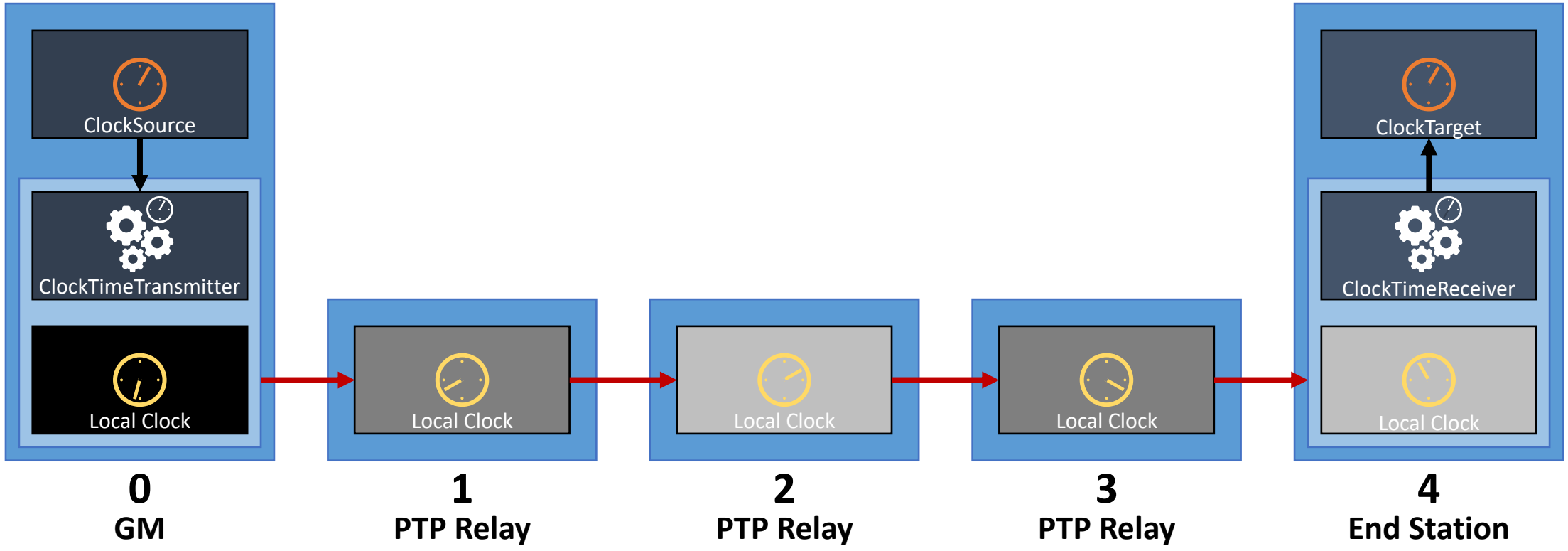
If no PTP Instance is grandmaster-capable, i.e., gmPresent is FALSE, then clockSlaveTime is set to the time provided by the LocalClock. This function is invoked when rcvdLocalClockTickCSS is TRUE.

10.2.13.2.2 invokeApplicationInterfaceFunction (functionName): Invokes the application interface function whose name is functionName. For the ClockSlaveSync state machine, functionName is clockTargetPhaseDiscontinuity.result (see 9.6.2).

Clocks & State Machines

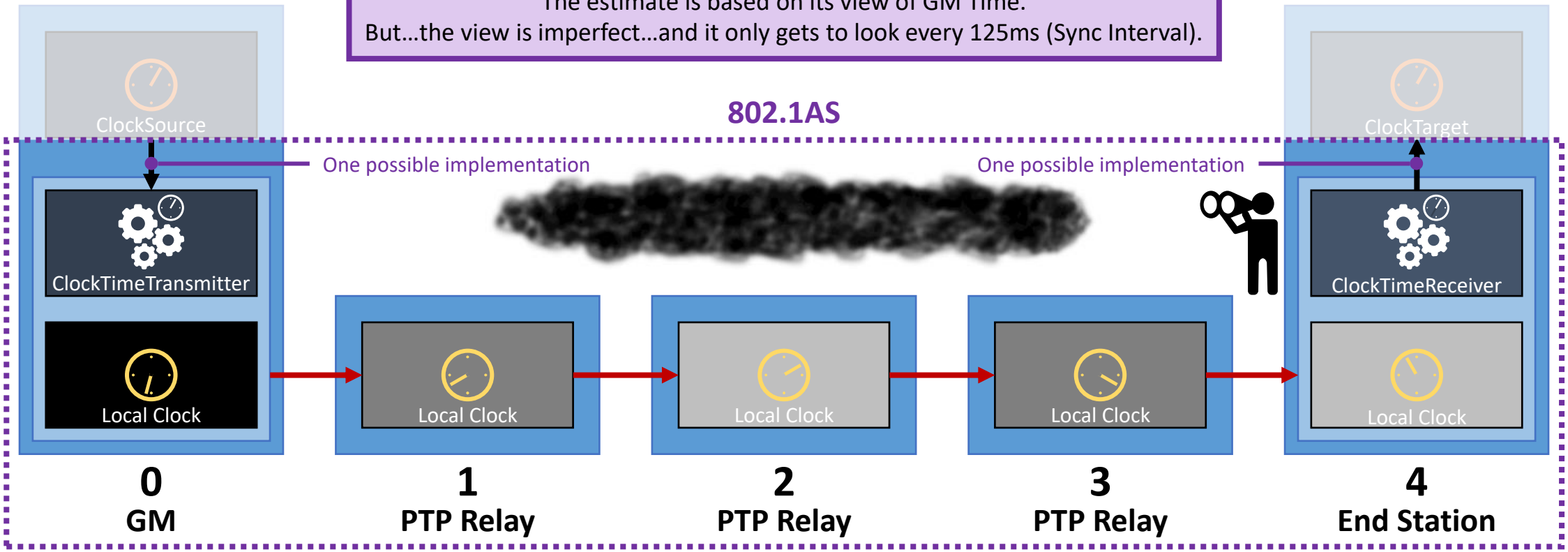


Clocks & State Machines



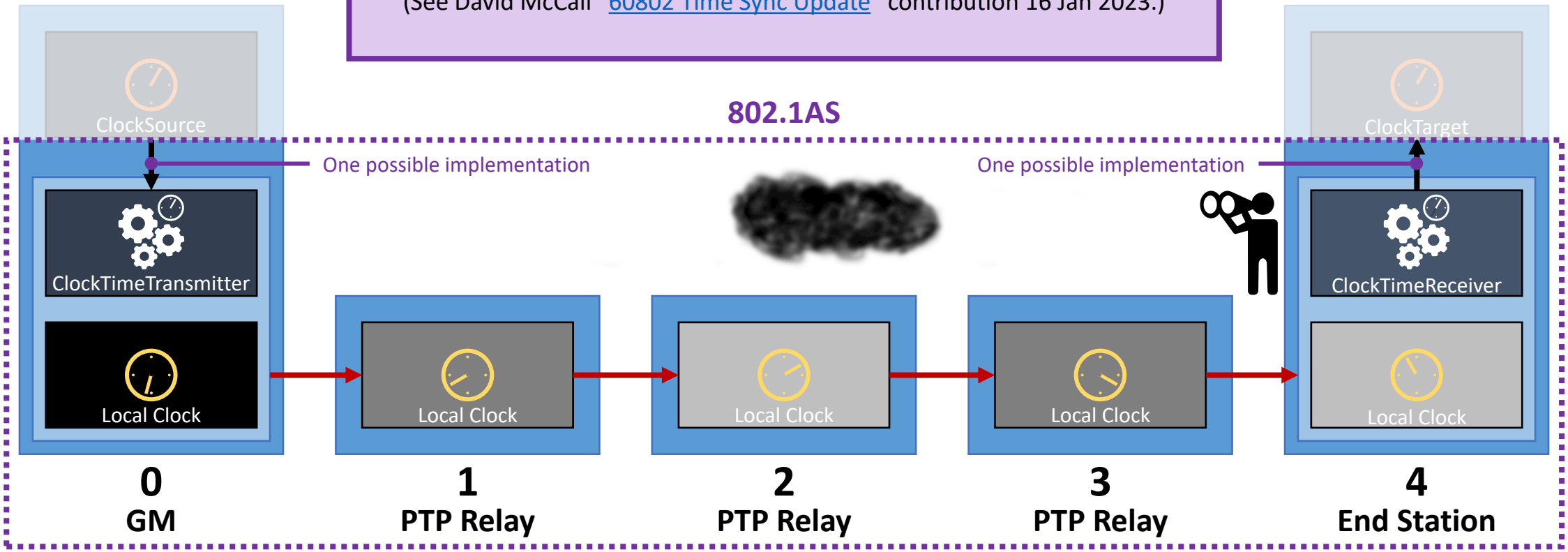
Scope of 802.1AS

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

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



For discussion…

- 802.1AS defines timeTransmitter & timeReceiver as state machines, not as oscillators.
  - There is an entity that reflects a concept of “GM Time” but that time could change dramatically…although when it does change in a ClockTimeTransmitter any changes are measured against local ClockTimeReceiver (based on LocalClock) and communicated in the Follow_Up information TLV.
- 802.1AS defines ClockSource & ClockTarget as entities that interface with ClockTimeTransmitter & ClockTimeReceiver respectively.
- But…

802.1AS-2020
9.1 Overview of the Interfaces

NOTE 1—The manner in which the ClockSource entity obtains time from a clock is outside the scope of this standard. The manner in which the ClockTarget uses the information provided by application interfaces is outside the scope of this standard.

The five interfaces are illustrated in Figure 9-1:

a) ClockSourceTime interface, which provides external timing to a PTP Instance,

b) ClockTargetEventCapture interface, which returns the synchronized time of an event signaled by a ClockTarget entity,

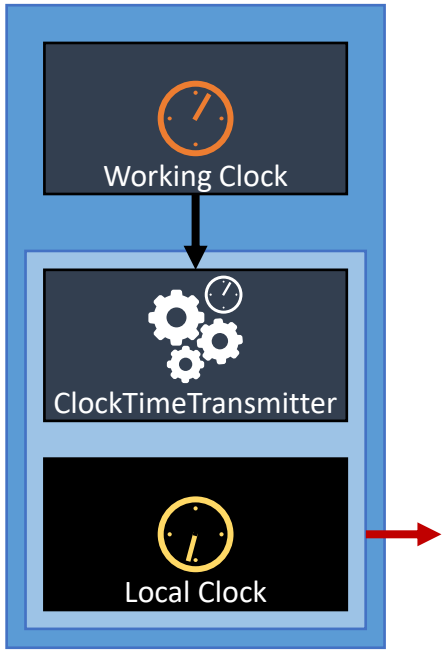
c) ClockTargetTriggerGenerate interface, which causes an event to be signaled at a synchronized time specified by a ClockTarget entity,

d) ClockTargetClockGenerator interface, which causes a periodic sequence of results to be generated, with a phase and rate specified by a ClockTarget entity, and

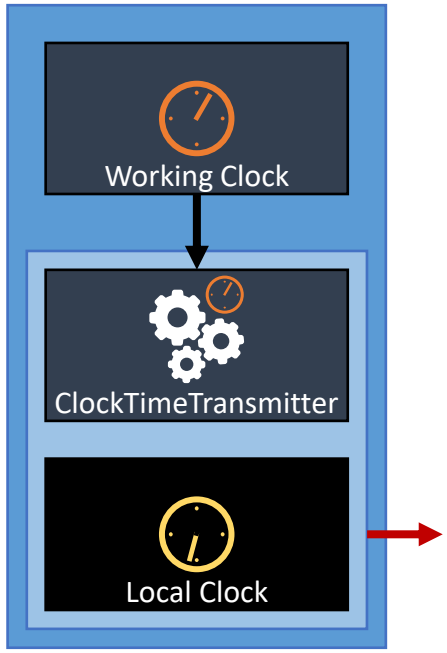
e) ClockTargetPhaseDiscontinuity interface, which supplies information that an application can use to determine if a discontinuity in Grandmaster Clock phase or frequency has occurred.

NOTE 2—The application interfaces described in this clause are models for behavior and not application program interfaces. Other application interfaces besides items a) through e) in this subclause are possible, but are not described here. In addition, there can be multiple instances of a particular interface.

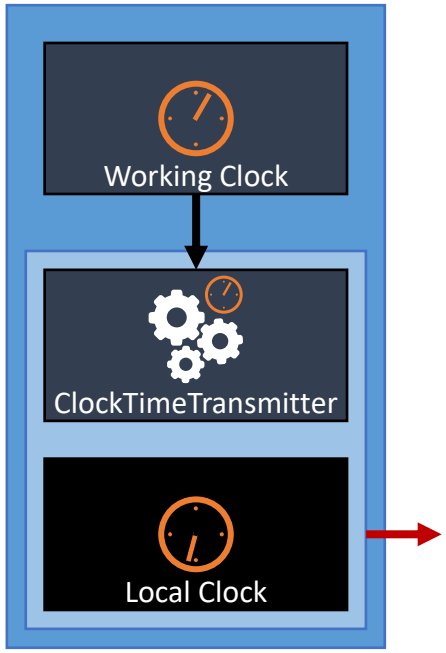
Possible Implementations at GM



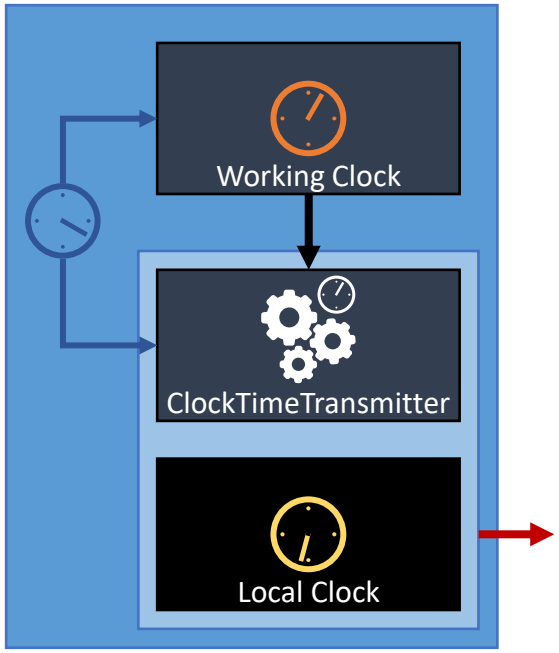
ClockTimeTransmitter receives retular updates from Working Clock via an API (Working Clock is ClockSource)



Working Clock and ClockTimeTransmitter are the same.
(There is no API; no ClockSource)



Same clock used for all clocks!



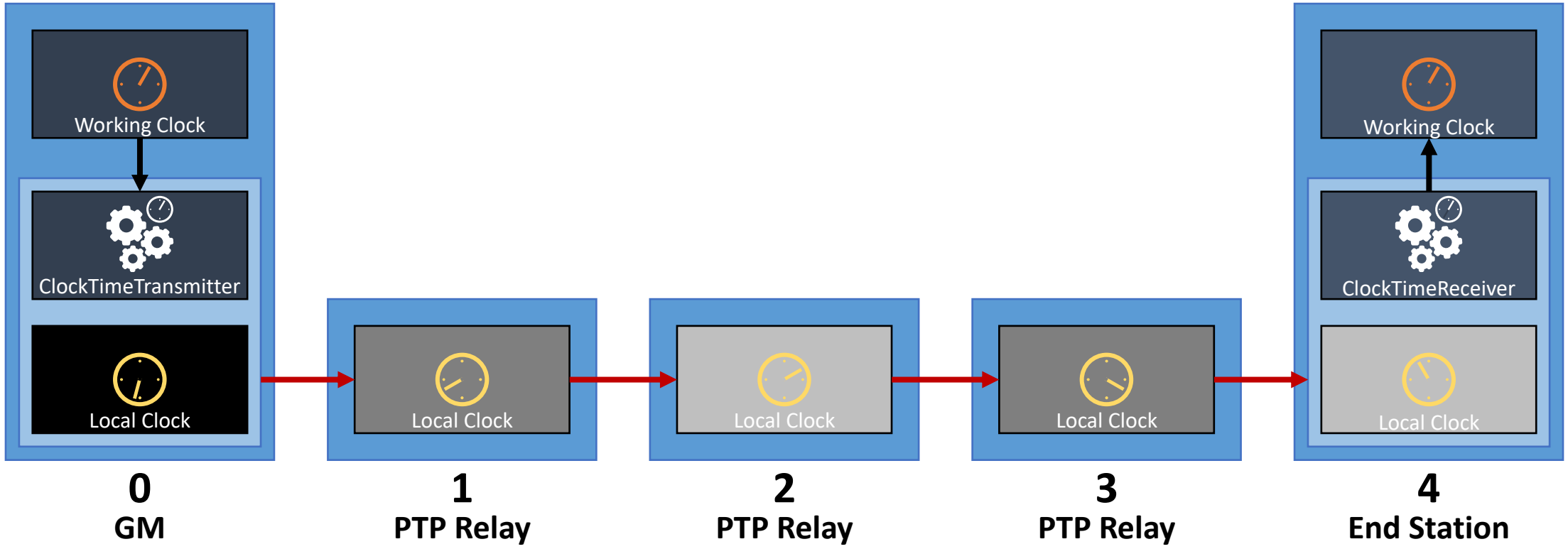
Wokring Clock and ClockTimeTransmitter are derived from the same SystemClock
(There is a ClockSource…but it only calls API once to initialise.)

What matters is the representation of GM Time and how well that matches with time at the Application. (Application Time?)

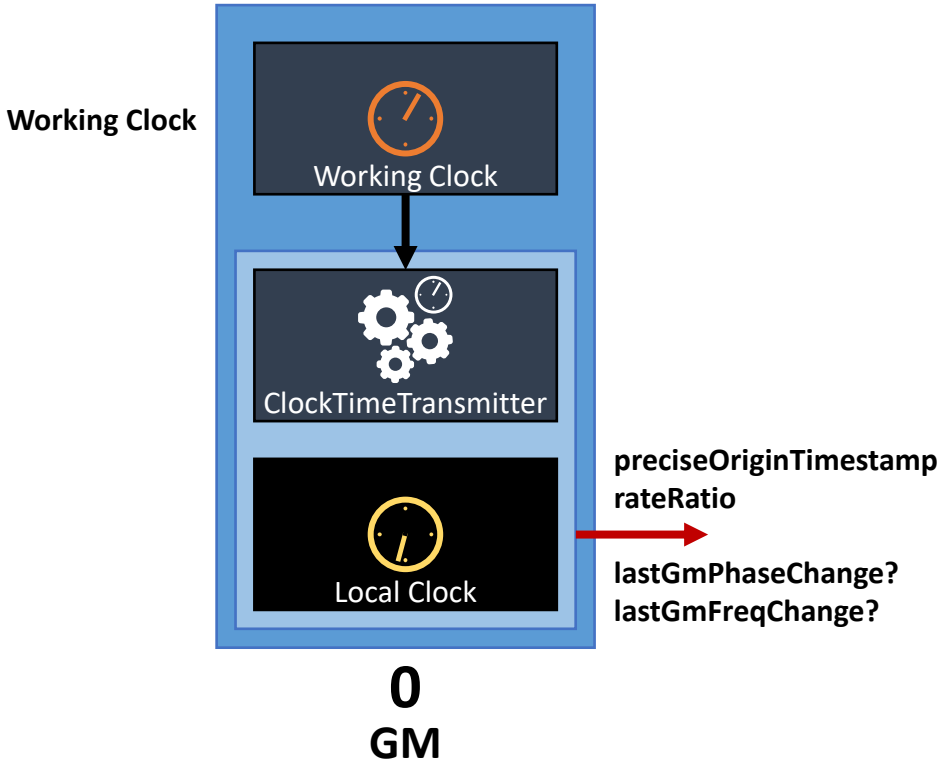
ClockSource or ClockTarget may not exist…

- 802.1AS gives ClockSource and ClockTarget as an illustration / example implementation.
  - Alternative implementations may or may not have equivalent entities.
- ClockTimeTransmitter and ClockTimeReceiver are more defined: state machines that are part of the distributed mechanism which delivers a view of time at one (GM Time at the ClockTimeTransmitter, which determines the preciseOriginTimestamp) to the other (ClockTimeReceiver's sense of time based on the received messages).
- Ultimately 60802 cares about the accuracy of the Working Clock and Global Time
  - At the Grandmaster PTP Instance vs. at the PTP End Instance
- So, we should place normative requirements on those things.

Clocks & State Machines

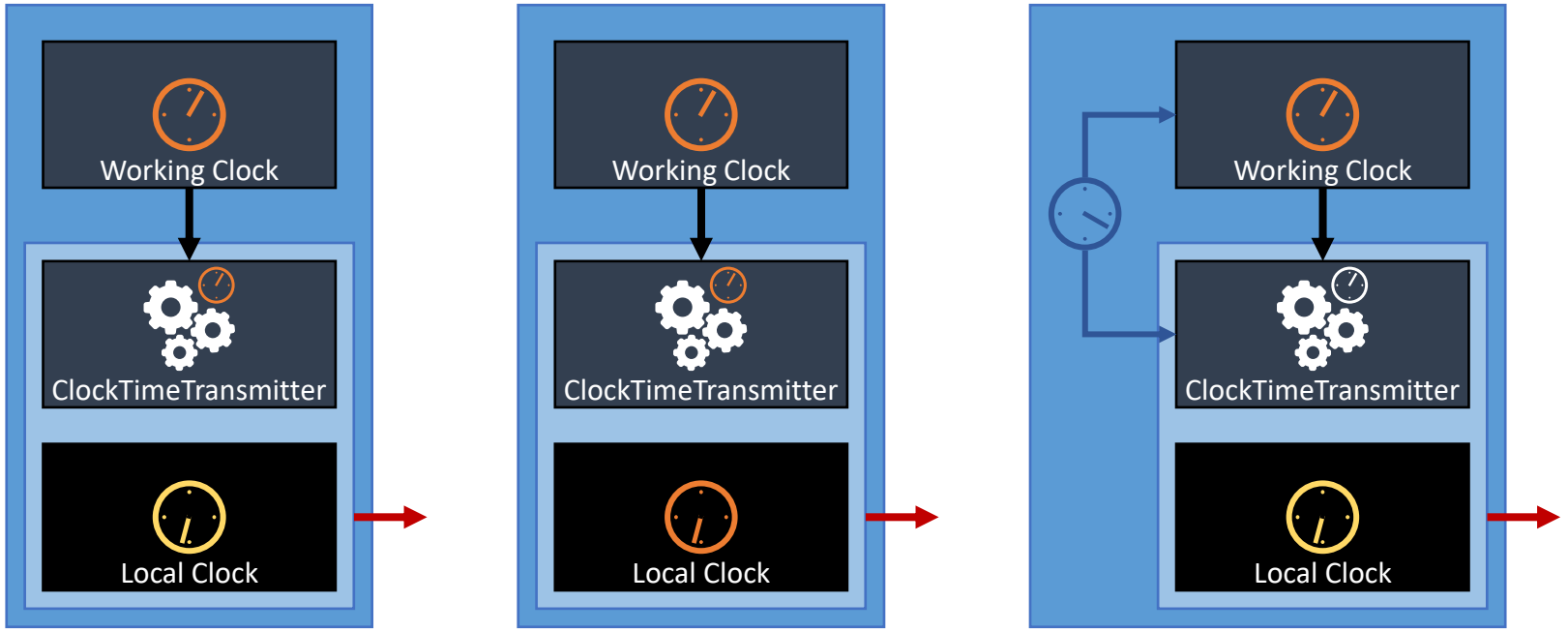


Error Generation limits at GM



- preciseOriginTimestamp shouldn't deviate too much from Working Clock at time of Sync message transmission
- rateRatio should reflect actual ratio between LocalClock (used for pDelayResp and Sync message egress timestamps and hence NRR calculation) and Working Clock
- How much error is too much?...

Error Generation Limits at GM



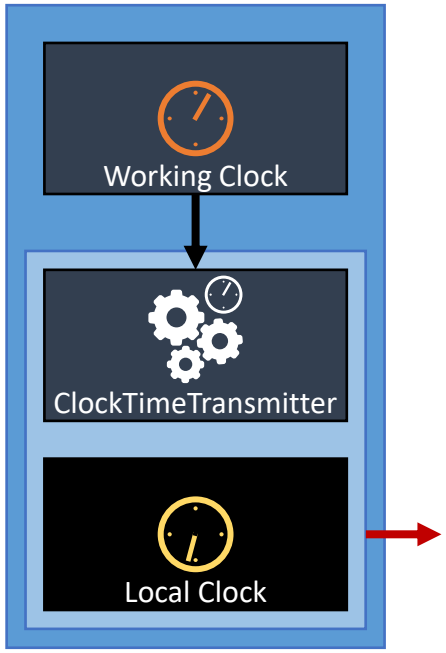
Working Clock and ClockTimeTransmitter are the same.
(There is no API; no ClockSource)

Same clock used for all clocks!

Wokring Clock and ClockTimeTransmitter are derived from the same SystemClock
(There is a ClockSource…but it only calls API once to initialise.)

- In these scenarios, there shouldn't be much of a problem as the Working Clock and the ClockTimeTransmitter's sense of time are closely coupled.

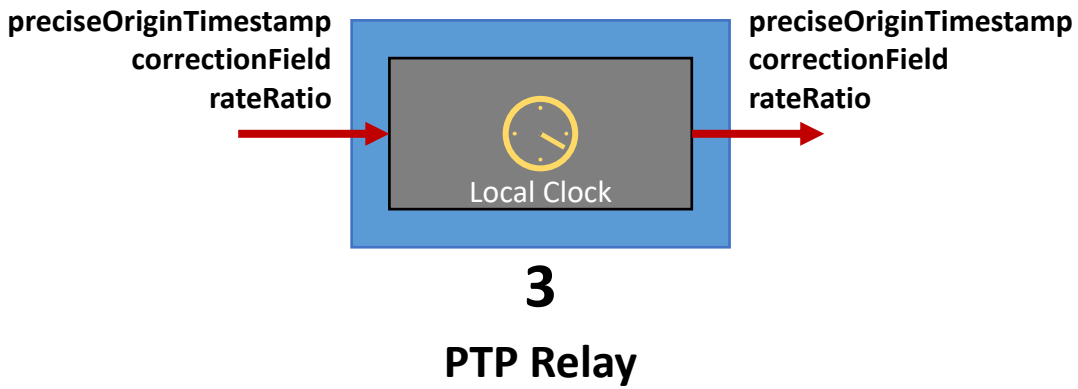
Error Generation Limits at GM



ClockTimeTransmitter receives retular updates from Working Clock via an API (Working Clock is ClockSource)

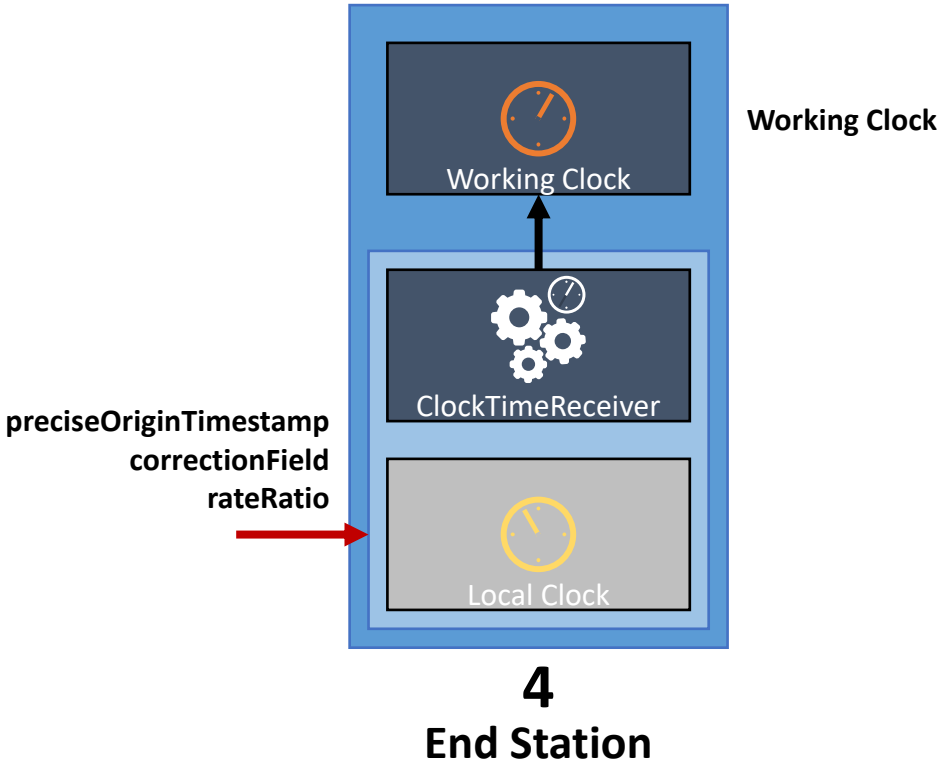
- In this scenario, a lot depends on how frequently the ClockTimeTransmitter is updated
- Note that this is similar to the “External Control” scenario
- If updates are every 125ms (same as Sync Interval) drift could be as large as at End Station, which could have a huge impact on the overall clock accuracy.

Error Generation limits at PTP Relay



- preciseOriginTimestamp + correctionField should increase by residenceTime
  - Measure when there is no clock drift
- rateRatio should be sufficiently accurate in three scenarios
  - No clock drift
  - RR clock drift (drifting RR input field)
  - NRR clock drift (simulate drifting of Local Clock by drifting incoming Sync message egress timestamps without drifting rateRatio)

Error Generation limits at GM



- Working Clock at End Station should track implied Working Clock at GM in three scenarios
  - No clock drift
  - RR clock drift (drifting RR input field)
  - NRR clock drift (simulate drifting of Local Clock by drifting incoming Sync message egress timestamps without drifting rateRatio)

For discussion…

- Local Clock…
  - -50 ppm to +50 ppm
  - -1,35 ppm/s to +2,12 ppm/s
- Working Clock at GM…
  - -50 ppm to +50 ppm
  - -1,35 ppm/s to +2,12 ppm/s
- Working Clock at End Station
  - ±250 ppm over any observation interval of 1 ms
  - Plus filtering?
- Global Time at GM
  - -200 ppm to +200 ppm
  - -10 ppm/s to +10 ppm/s
- Global Time at End Station
  - ±1000 ppm over any observation interval of 1 ms
  - Plus filtering?

For discussion…

- Local Clock doesn't require any further normative requirements.
- Working Clock at GM should have the same requirements as Local Clock to ensure target accuracy of Working Clock at End Station can be maintained.
  - Depending on implementation (whether Working Clock is directly coupled to ClockTimeTransmitter) there may or may not be drift between Working Clock and Local Clock. If there is, it should be limited to what could be expected when an extra node to the chain (101 hops), i.e. same clock drift requirements, 125ms update interval.
- Global Time is similar to Working Clock but with more lax requirements.

For discussion (and written to provoke it!)…

- When ClockSource is driven by an External Clock, drift between External Clock and GM Local Clock **may** or **may not** cause additional errors at the ClockTarget
  - Depends on how frequently the timeTransmitter is updated via the ClockSourceTime.invoke API is called.
  - Frequent updates relative to Sync Interval → minimal additional errors
  - Infrequent updates relative to Sync Interval → additional errors based on difference between External Clock offset + GM Local Clock offset
    - If GM Local Clock is running faster than External Clock (e.g. + 10 ppm), even if stable (i.e. 0 ppm/s) an error will build up until the next time timeTransmitter is updated
    - Note that **ppm/s isn't the critical parameter** regarding the **additional** error, as the External Clock isn't part of the PTP process, e.g. there is no mechanism to measure the rate ratio between the GM Local Clock and the External Clock.
    - However, ppm/s is still a source of error. If the External Clock is drifting, it will have the same impact as the GM Local Clock drifting.
- To ensure the same time sync accuracy (±1 µs at node 100) as an independent system, an External Clock should have the same ppm/s limits as the GM Local Clock, but…
- The ClockTarget at node 100 of a 60802 system will have slightly worse ppm/s characteristics than the GM Local Clock, so…should we calculate (via simulation) how much worse…and how much buffer there is…and therefore how many systems can be chained together while still maintaining accuracy relative to the first system's ClockSource / timeTransmitter?

Thank you!

Time Sync – Elements & Relationships

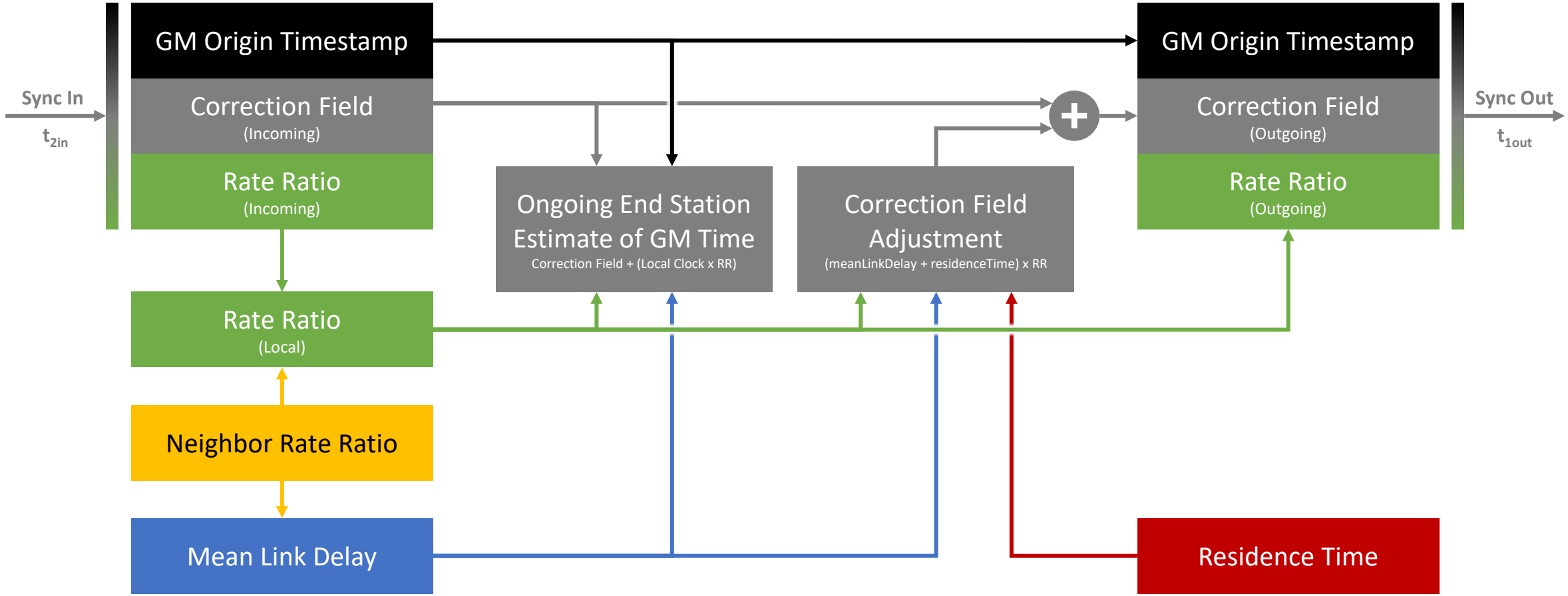


Figure 10-2—Time-synchronization state machines—overview and interrelationships