60802 Time Sync – Clocks Relationships & Normative Requirements

David McCall – Intel Corporation

Version 1

Shared last time…

Table 9 – Required values

Topic	Value
Maximum fractional frequency offset relative to the TAI frequency for LocalClock (used for timeReceiver, Grandmaster, or PTP Relay instance) or Clock Target	-50 ppm to +50 ppm
Maximum fractional frequency offset relative to the nominal frequency for ClockTarget	-50 ppm to +50 ppm
Maximum absolute value of rate of change of fractional frequency offset for ClockTarget	-1,35 ppm/s to +2,12 ppm/s
Maximum value of frequency adjustment for ClockTarget used for Global Time	+/-1000 ppm over any observation interval of 1 ms
Maximum value of frequency adjustment for ClockTarget used for Working Clock	+/-250 ppm over any observation interval of 1 ms

NOTE The Maximum value of frequency adjustment represents an upper bound that limits how much a PTP instance can change the frequency of its ClockTarget during a given period. However, these adjustments would be incremental rather than instantaneous over the defined interval.

ClockSource/Target & timeTransmitter/Receiver APIs

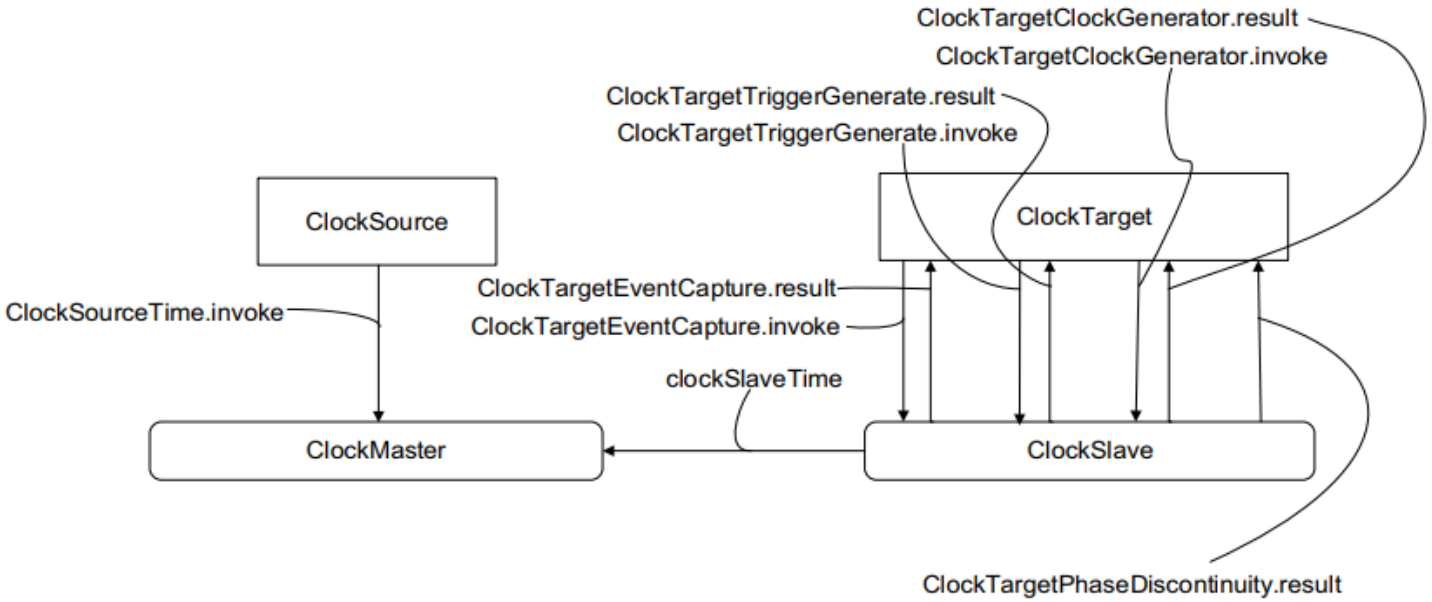


Figure 9-1—Application interfaces

timeTransmitter/Receiver & Local Clock

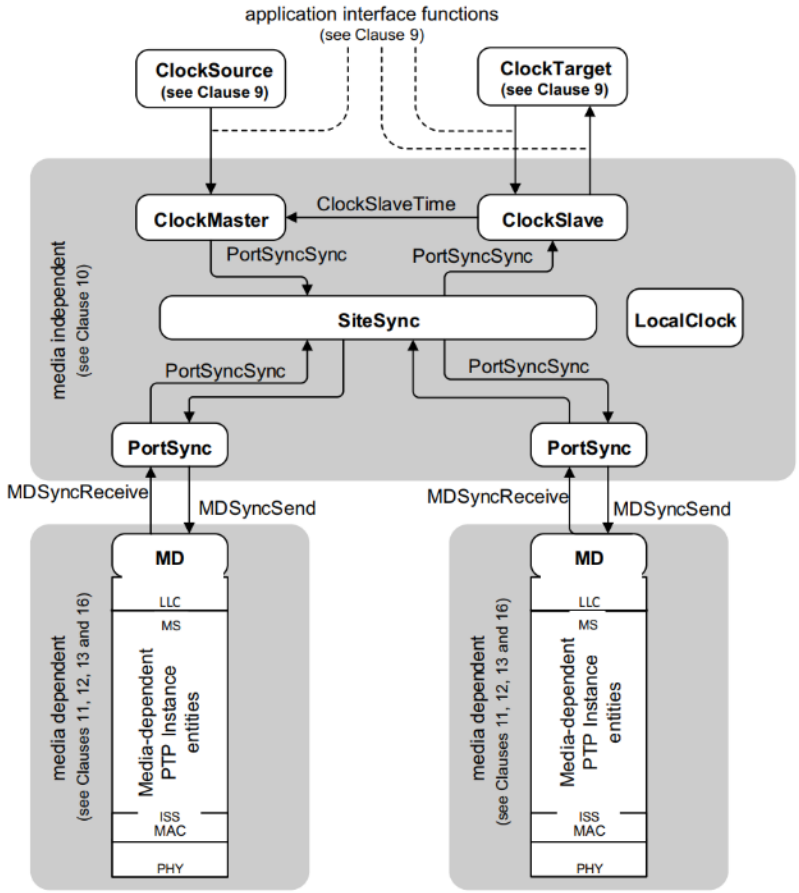
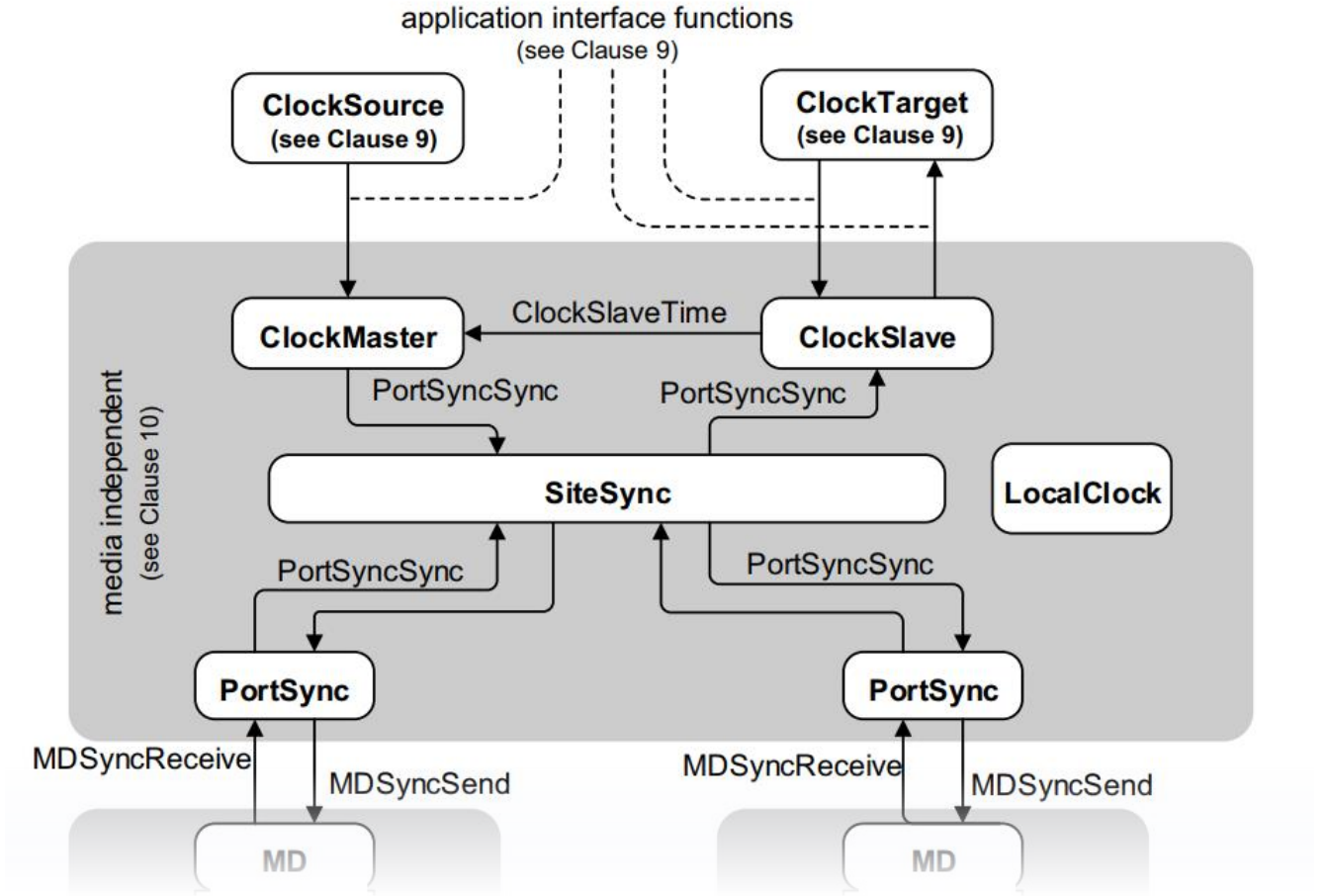


Figure 10-1—Model for media-independent layer of PTP Instance

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:

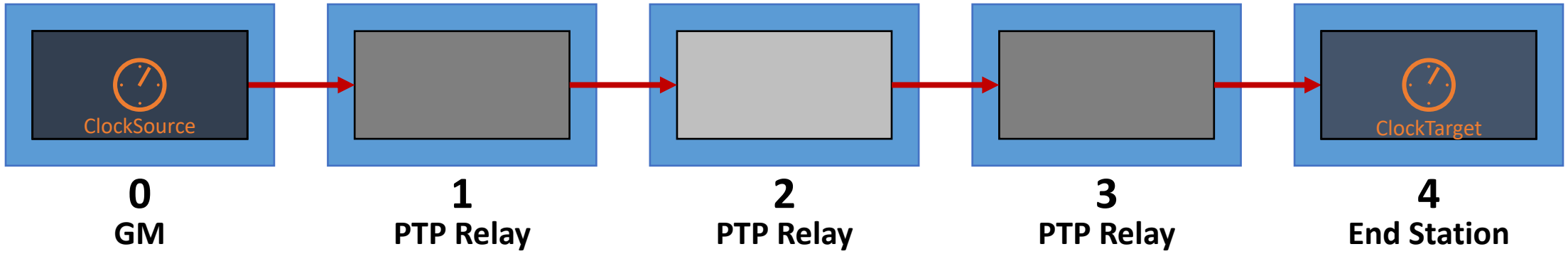
a) Set to syncReceiptTime at every LocalClock update immediately after a PortSyncSync structure is received, and

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.

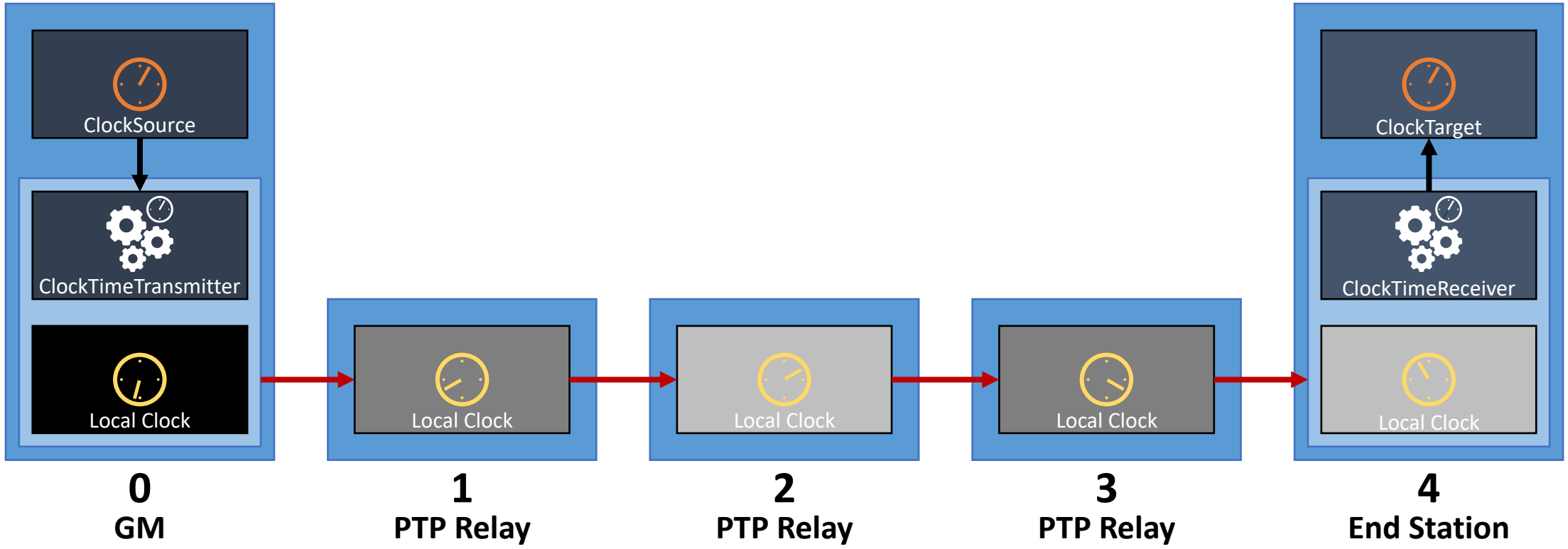
If no PTP Instance is grandmaster-capable, i.e., gmPresent is FALSE, then clockSlaveTime is set to the time provided by the LocalClock. This function is invoked when rcvdLocalClockTickCSS is TRUE.

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

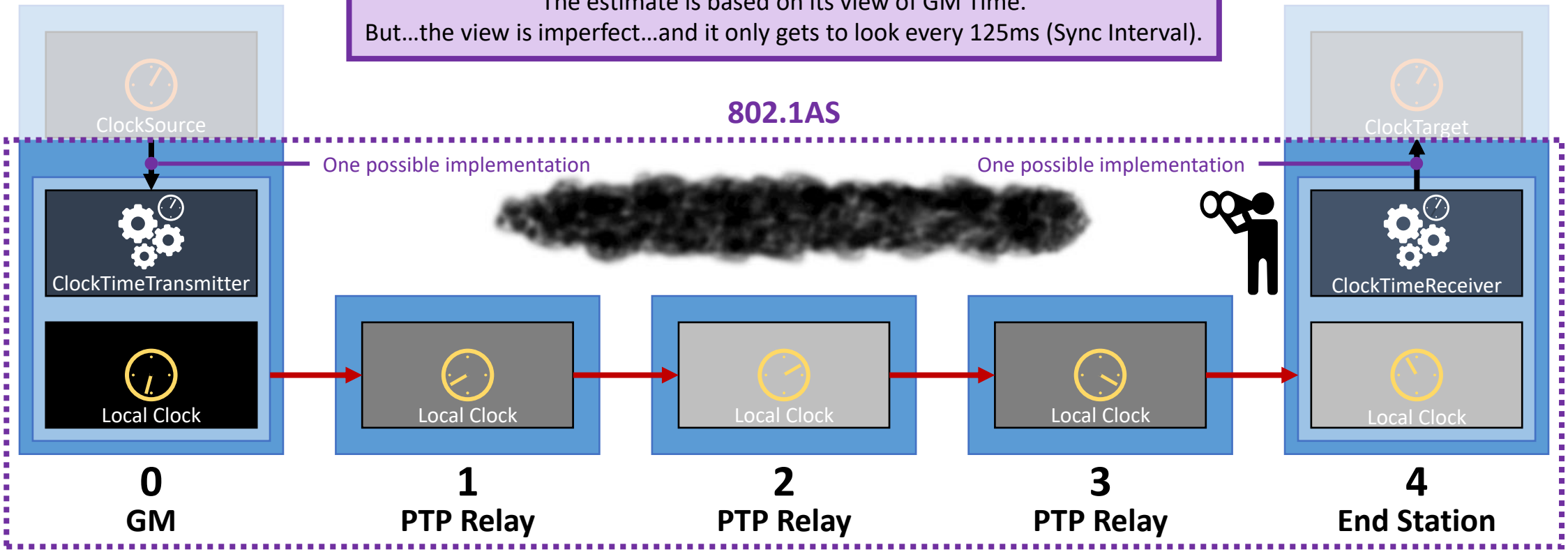


Clocks & State Machines



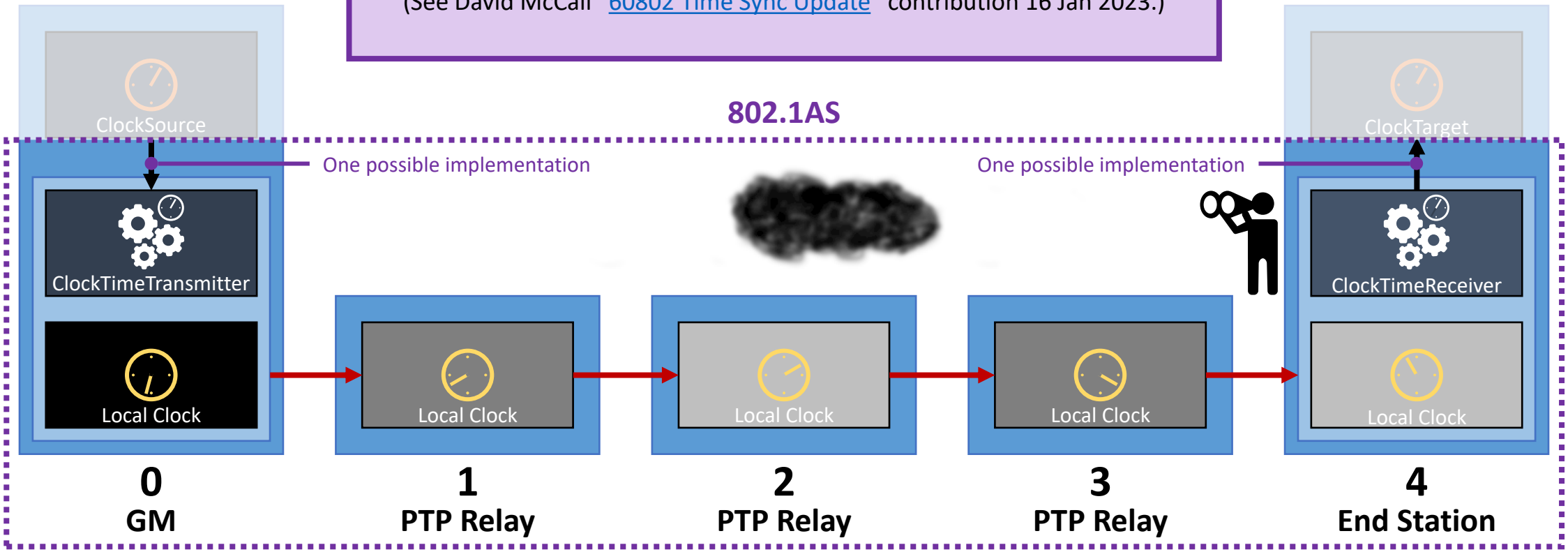
Scope of 802.1AS

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 (and written to provoke it!)…

- 802.1AS defines timeTransmitter & timeReceiver as state machines, not as oscillators.
  - There is an entity that reflects its concept of “GM Time” but that time could change dramatically…although when it does any changes are measured against local ClockReceiverTime (based on LocalClock) and communicated.
- 802.1AS defines ClockSource & ClockTarget as entities that interface with timeTransmitter & via defined APIs, not as oscillators.
- But…

802.1AS-2020
9.1 Overview of the Interfaces

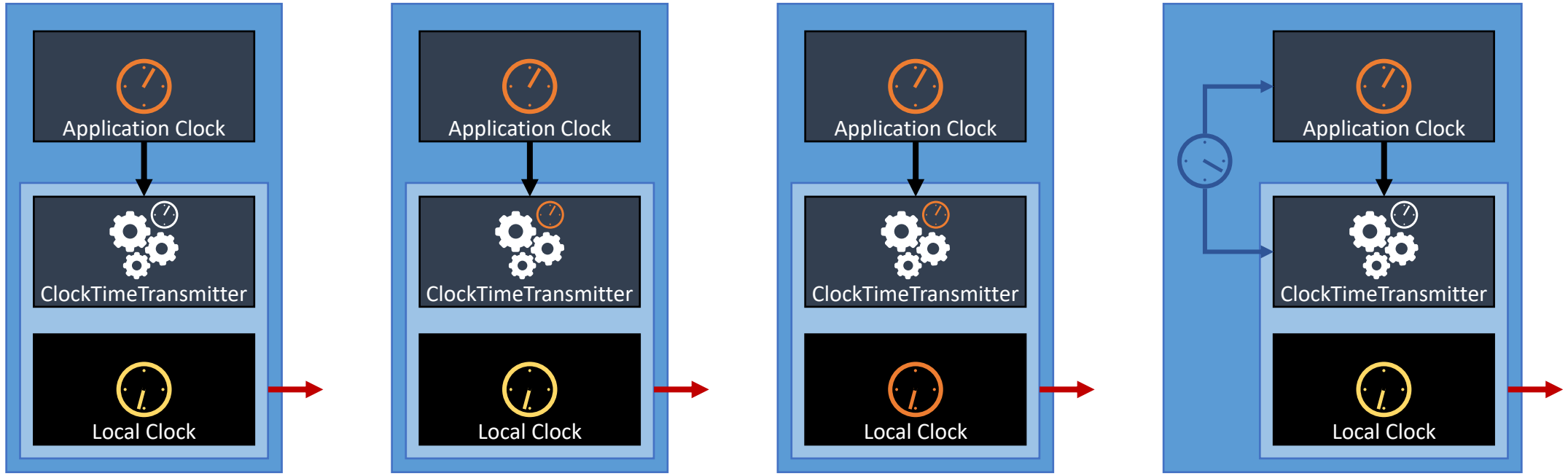
NOTE 1—The manner in which the ClockSource entity obtains time from a clock is outside the scope of this standard. The manner in which the ClockTarget uses the information provided by application interfaces is outside the scope of this standard.

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 Application Clock via an API (Application Clock is ClockSource)

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

Same clock used for all clocks!

Application Clock and ClockTimeTransmitter are derived from the same SystemClock (ClockSource…but only calls API to initialise.)

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

For discussion (and written to provoke it!)…

- Local Clock needs to be have ppm and ppm/s limits.
- It doesn’t make sense to define ppm and ppm/s limits for ClockTimeTransmitter, ClockTimeReceiver, ClockSource or ClockTarget.
  - ClockSource & ClockTarget may not even exist in a particular implementation.
- It does make sense to define how closely the Application Clock at the GM matches the GM Time (in ClockTimeTransmitter).
- The filtering / control loop requirements are normative and applied to the output of the timeTransmitter received in order to obtain the Application Time at the EndStation.
  - The error generation limits at an End Station are on the filtered values / output of the control loop. (Which is not the case for a PTP Relay instance, which operates based on the Local Clock only. For that, the normative requirements on error generation apply to the Correction Field & Rate Ratio output.)

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

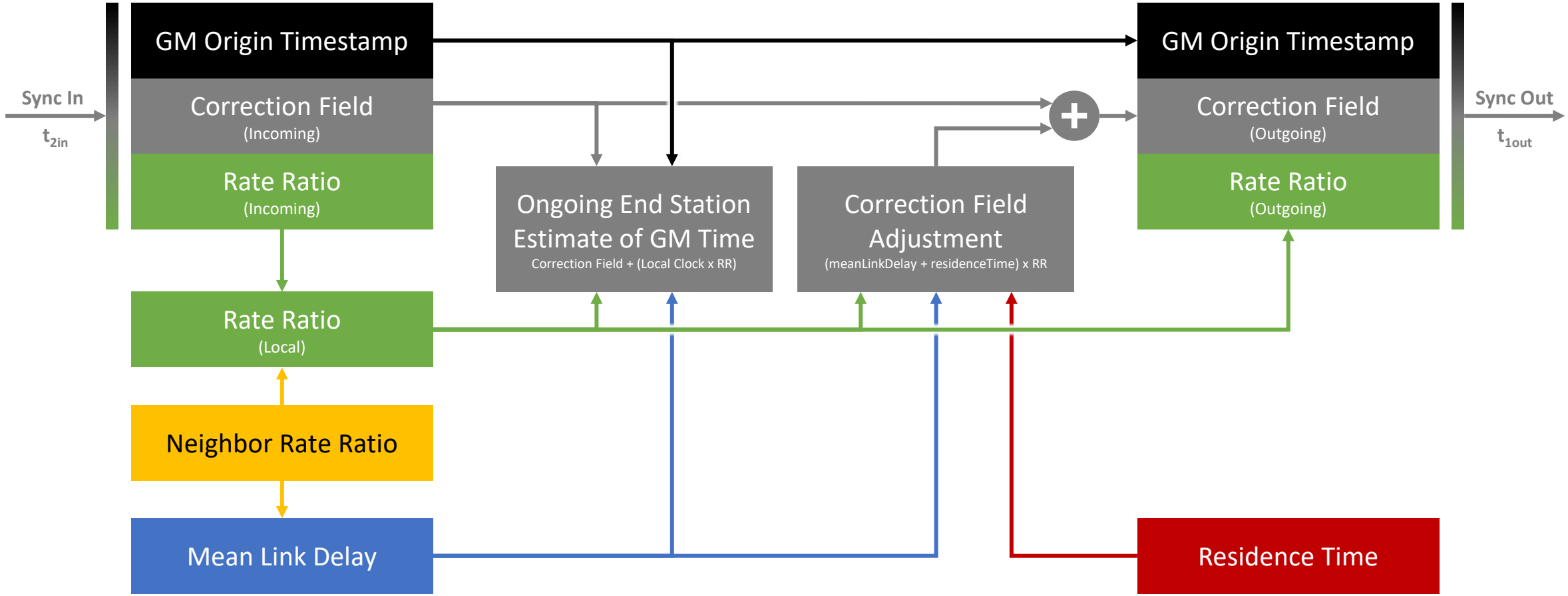


Figure 10-2—Time-synchronization state machines—overview and interrelationships