IEEE P802.1ASdm/D1.0 Rogue Comments related to Figure 17-3

Johannes Specht
(Self)

Introduction

- **Background**
  - **I discovered a couple of problems in P802.1ASdm/D1.0 while creating my own version of Figure 17-3 (https://www.ieee802.org/1/files/public/docs2023/dm-specht-d10-figure173-0523-v00.pdf).**
  - **Some of these issues are addressed already in my own version of that figure, some are not …**
- **Purpose**
  - **This slide set summarizes the problems not covered in my version of Figure 17-3**
    - **Explain the issues**
    - **Making suggestions**

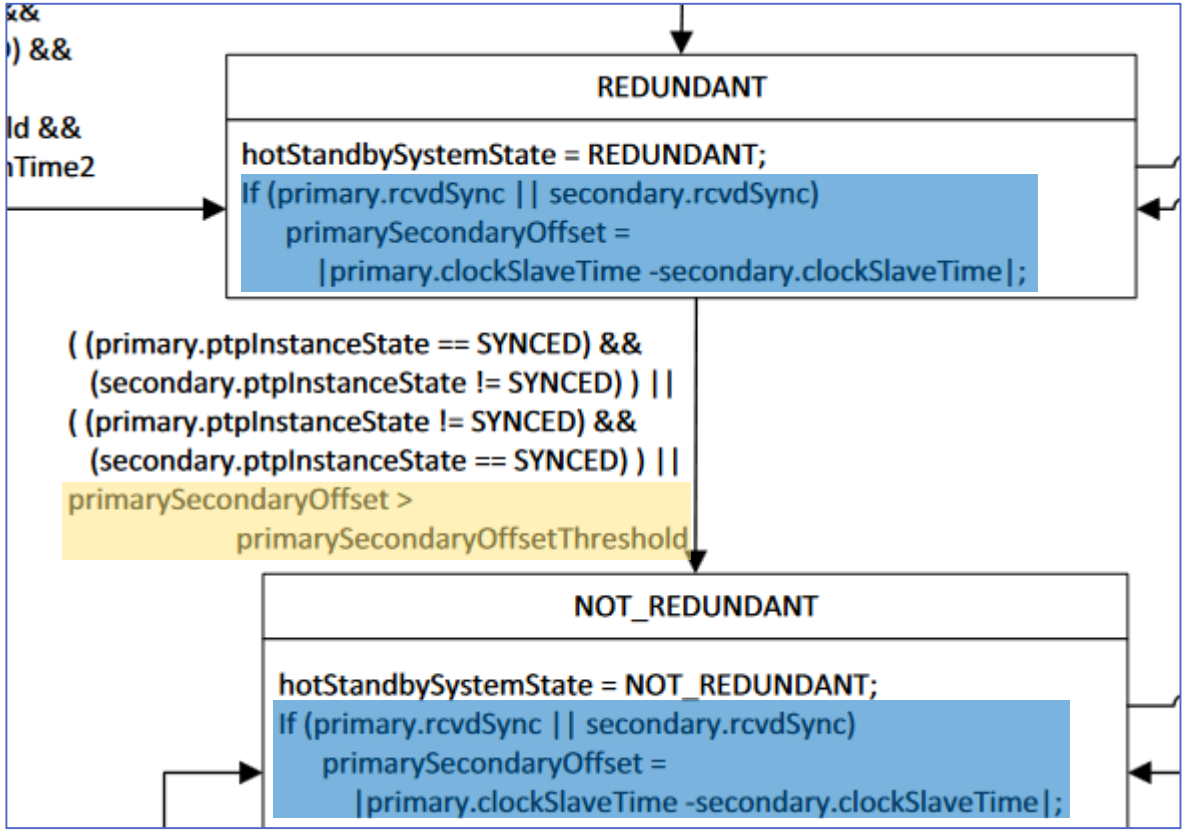
Rogue comment #1: Computing/using the offset between primary and secondary

#1: Explanation

17.6.2.6 primarySecondaryOffset: The value of the managed object hotStandbySystemDS.primarySecondaryOffset (see 14.19.10), which is the absolute value of the difference between the clockSlaveTimes (see 10.2.4.3) of the primary and secondary PTP Instances.

Source: P802.1ASdm/D1.0

- **primarySecondaryOffset** constraints when REDUNDANT can be entered/shall be left:
  - If above some threshold, leave
  - If below that threshold, enter is possible
- **Yellow**: One example from the current Draft, in *[Offset-Ok Condition]* the figure I contributed earlier.
- **Blue**: Computation of primarySecondaryOffset.



Source: Figure 17-3 of P802.1ASdm/D1.0

#1, Sub-issue A: It rarely becomes effective

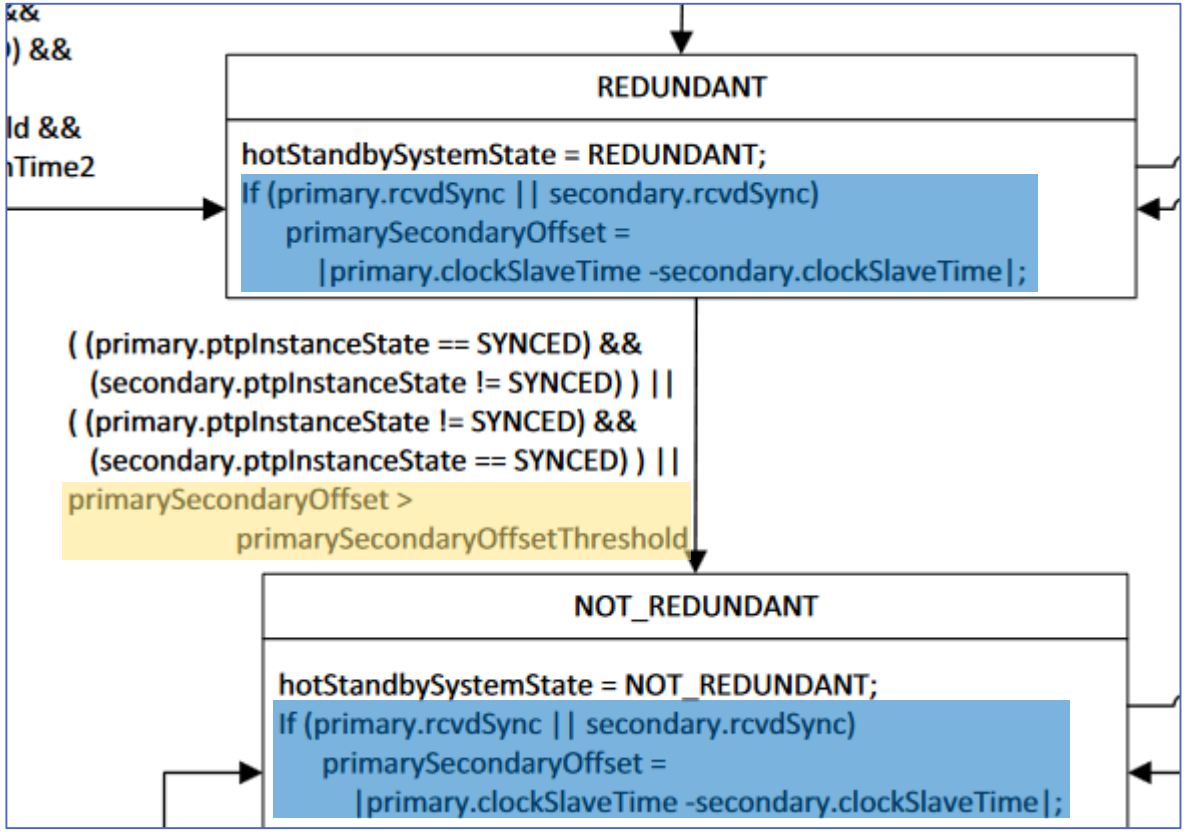
17.6.2.6 primarySecondaryOffset: The value of the managed object hotStandbySystemDS.primarySecondaryOffset (see 14.19.10), which is the absolute value of the difference between the clockSlaveTimes (see 10.2.4.3) of the primary and secondary PTP Instances.

Source: P802.1ASdm/D1.0

Problem description

- In Figure 17-3, **primarySecondaryOffset** is **computed only once**, each time a state is entered (blue).
- While the state machine resides in a state, it is **never updated**.

→ The associated conditions (yellow "bail-out" in the example) **rarely become effective!**



Source: Figure 17-3 of P802.1ASdm/D1.0

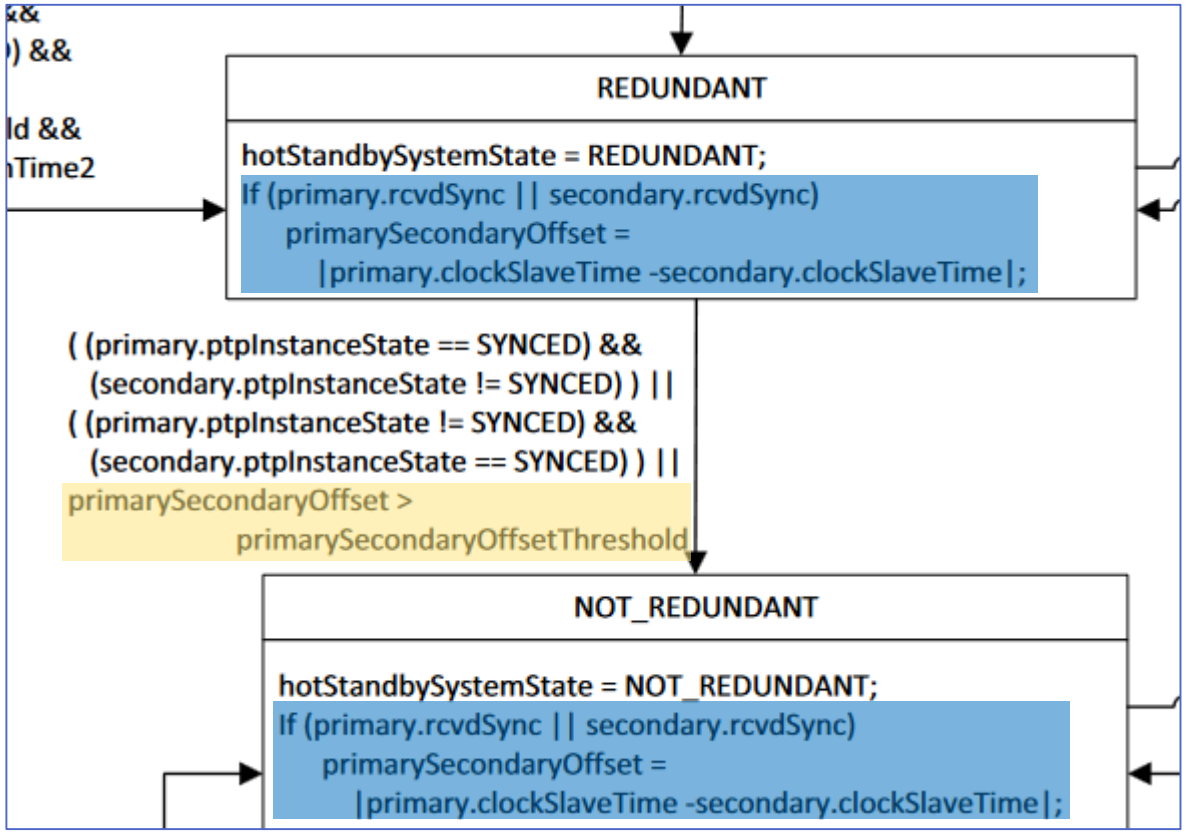
#1, Sub-issue B: Race condition

17.6.2.6 primarySecondaryOffset: The value of the managed object hotStandbySystemDS.primarySecondaryOffset (see 14.19.10), which is the absolute value of the difference between the clockSlaveTimes (see 10.2.4.3) of the primary and secondary PTP Instances.

Source: P802.1ASdm/D1.0

Problem description

- … even if the conditional update (blue) would be executed on more frequently (which it doesn't, as shown earlier), the condition (if-expression in blue) would not work reliable.
- Primary/secondary.**rcvdSync** is rarely TRUE and most of the time FALSE, because it is a synchronization variable consumed and reset by another state machine, resulting in a race condition.
- Next slide …



Source: Figure 17-3 of P802.1ASdm/D1.0

#1, Sub-issue B: From cross-ref. Material

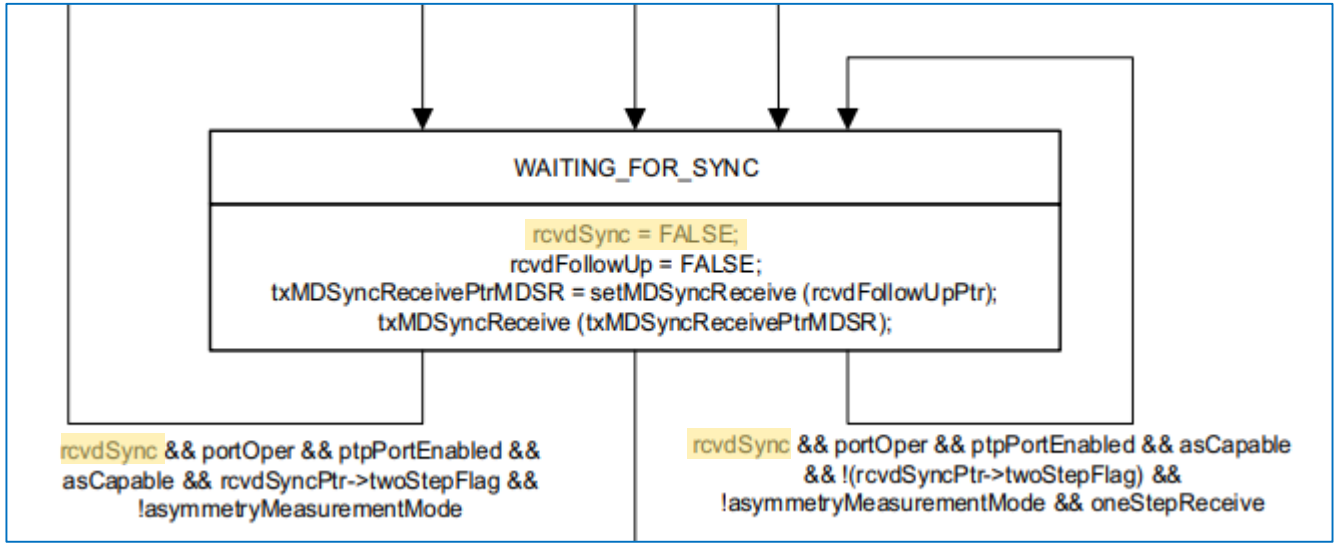
11.2.14.1.2 rcvdSync: A Boolean variable that notifies the current state machine when a Sync message is received. This variable is reset by the current state machine.

Source: Std 802.1AS-2020

- Once rcvdSync==TRUE and causes a transition in MDSyncReceiveSM, it is **set to FALSE immediately by MDSyncReceiveSM** in most cases (not only the example shown).

→ The Figure 17-3 FSM may miss rcvdSync==TRUE.

- Remark
  - Setting rcvdSync to TRUE is unspecified in Std 802.1AS-2020.
  - The statement in 11.2.14.1.2 may be enhanced (i.e., “sub-sub-issue”).



Source: Figure 11-6 of P802.1ASdm/D1.0, MDSyncReceiveSM

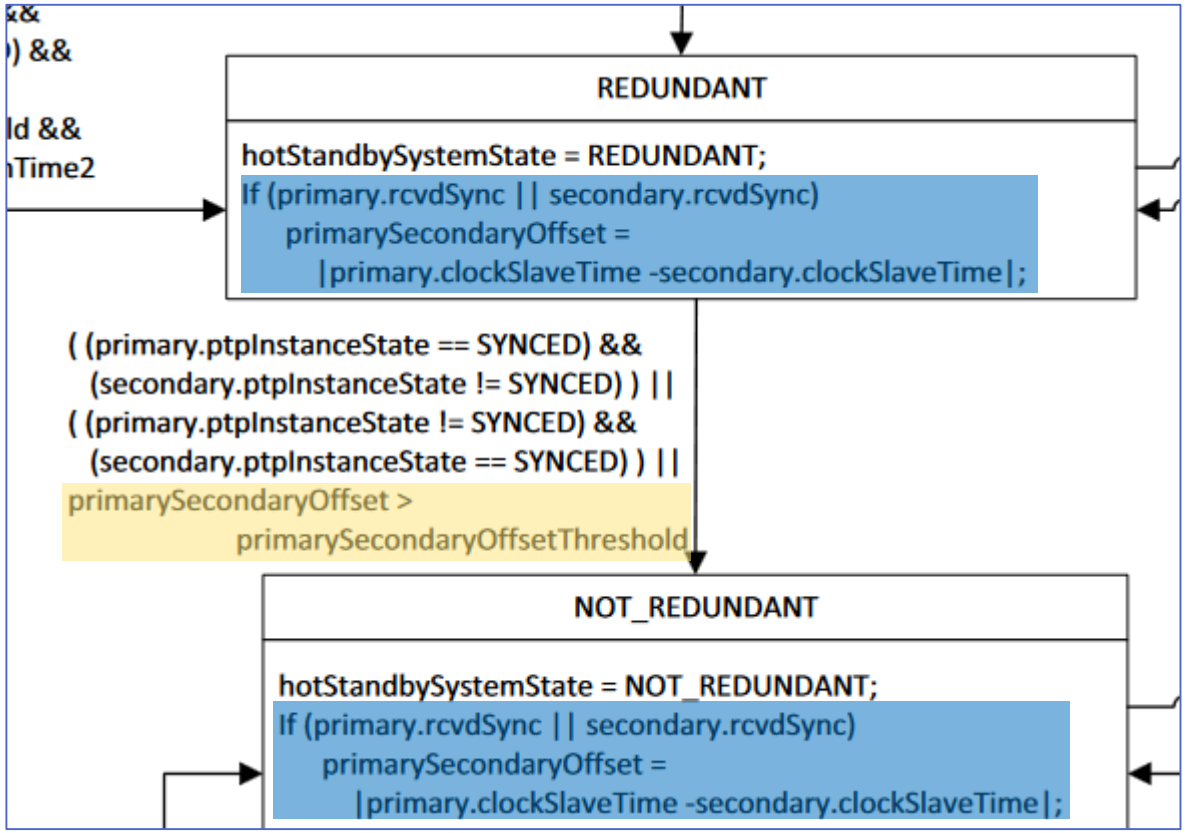
#1, Sub-issue C: Update frequency too low

17.6.2.6 primarySecondaryOffset: The value of the managed object hotStandbySystemDS.primarySecondaryOffset (see 14.19.10), which is the absolute value of the difference between the clockSlaveTimes (see 10.2.4.3) of the primary and secondary PTP Instances.

Source: P802.1ASdm/D1.0

Problem description

- … even if the update (blue) would be executed on every sync message (which it doesn't, as shown earlier), the given update frequency on rcvdSync (i.e., on sync message reception) would be too low.
- The offset between the underlying variables, clockSlaveTime in 10.2.4.3, can become larger between such messages.
- Next slide …



Source: Figure 17-3 of P802.1ASdm/D1.0

#1, Sub-issue C: From cross-ref. Material

10.2.4.3 clockSlaveTime: The synchronized time maintained, at the slave, at the granularity of the LocalClock entity [i.e., a new value is computed every localClockTickInterval (see 10.2.4.18) by the ClockSlave entity]. The data type for clockSlaveTime is ExtendedTimestamp.

10.2.4.18 localClockTickInterval: The time interval between two successive significant instants (i.e., "ticks") of the LocalClock entity. The data type for localClockTickInterval is TimeInterval.

Source of all that textboxes:
Std 802.1AS-2020

→ Update **every LocalClock tick** required (**not** just every sync message)…

Plus, there is no linearity between messages in clockSlaveTime that can be argued with, because of implementation-specifics.

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:

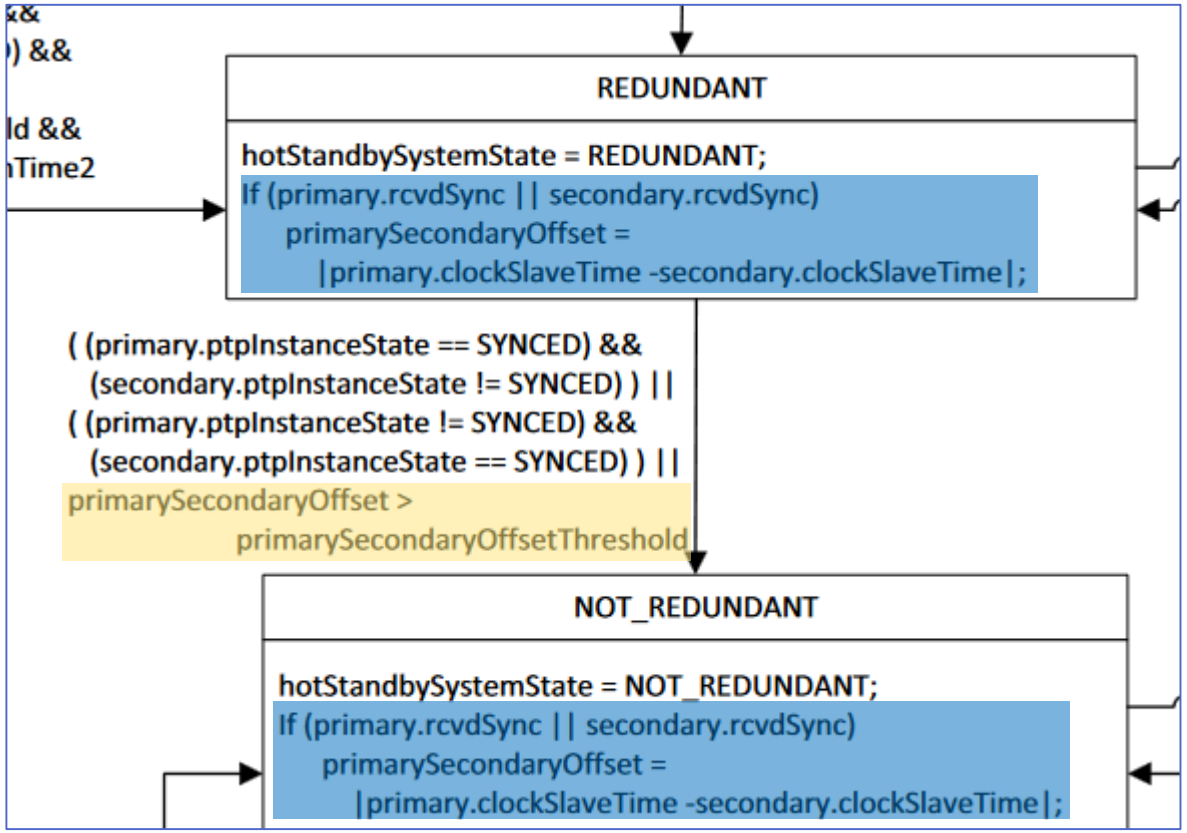
a) Set to syncReceiptTime at every LocalClock update immediately after a PortSyncSync structure is received, and

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.

#1: Suggested Remedy

1. Move primarySecondaryOffset calculations from Figure 17-3 (blue) to a single state in a new state machine, PrimarySecondaryOffsetCalculation.
2. Cause a self-transition of that new state whenever updateSlaveTime() is invoked, potentially additional events.
3. Keep the the offset-OK condition (yellow) where they are.



Source: Figure 17-3 of P802.1ASdm/D1.0

Rogue comment #2:
More than one rcvdSync?

#2: Explanation and Problems

Explanation

- **rcvdSync** is a variable of MDSyncReceiveSM.
- MDSyncReceiveSM is a media-dependent state machine for full-duplex (e.g., 802.3).
- P802.1ASdm/D1.0 exposes rcvdSync via management…

11.2.14.1.2 rcvdSync: A Boolean variable that notifies the current state machine when a Sync message is received. This variable is reset by the current state machine.

Source: Std 802.1AS-2020

14.8.7 rcvdSync

The value of the variable rcvdMDSyncPSSR (see 10.2.8.1.1), which notifies the PortSyncSyncReceive state machine of the PTP Instance that time synchronization information has been received (i.e., that an MDSyncReceive structure has been received from an MD entity of the same PTP Port (see 10.2.2.1). The value is TRUE when time synchronization information is received, and the variable is reset by the PortSyncSyncReceive state machine.

Source: 802.1ASdm/D1.0

Problem A

- …,well, it does not expose the rcvdSync variable, but instead exposes rcvdMDSyncPSSR with name rcvdSync, where rcvdMDSyncPSSR is consumed by media-independent state machine PortSyncSyncReceive:
  - **Alias** ☹
  - **Ambiguity** ☹

Problem B

- Like the rcvdSync variable, the rvcdMDSyncPSSR variable is immediately reset to FALSE by PortSyncSyncReceive in most cases (next slide).
- The value of exposing rvcdMDSyncPSSR seems low:
  - Reading/polling it (e.g., via NETCONF) returns “FALSE”, “TRUE” seems like a glitch.
  - But if it is there, it needs to be tested ☹

#2: Explanation and Problems

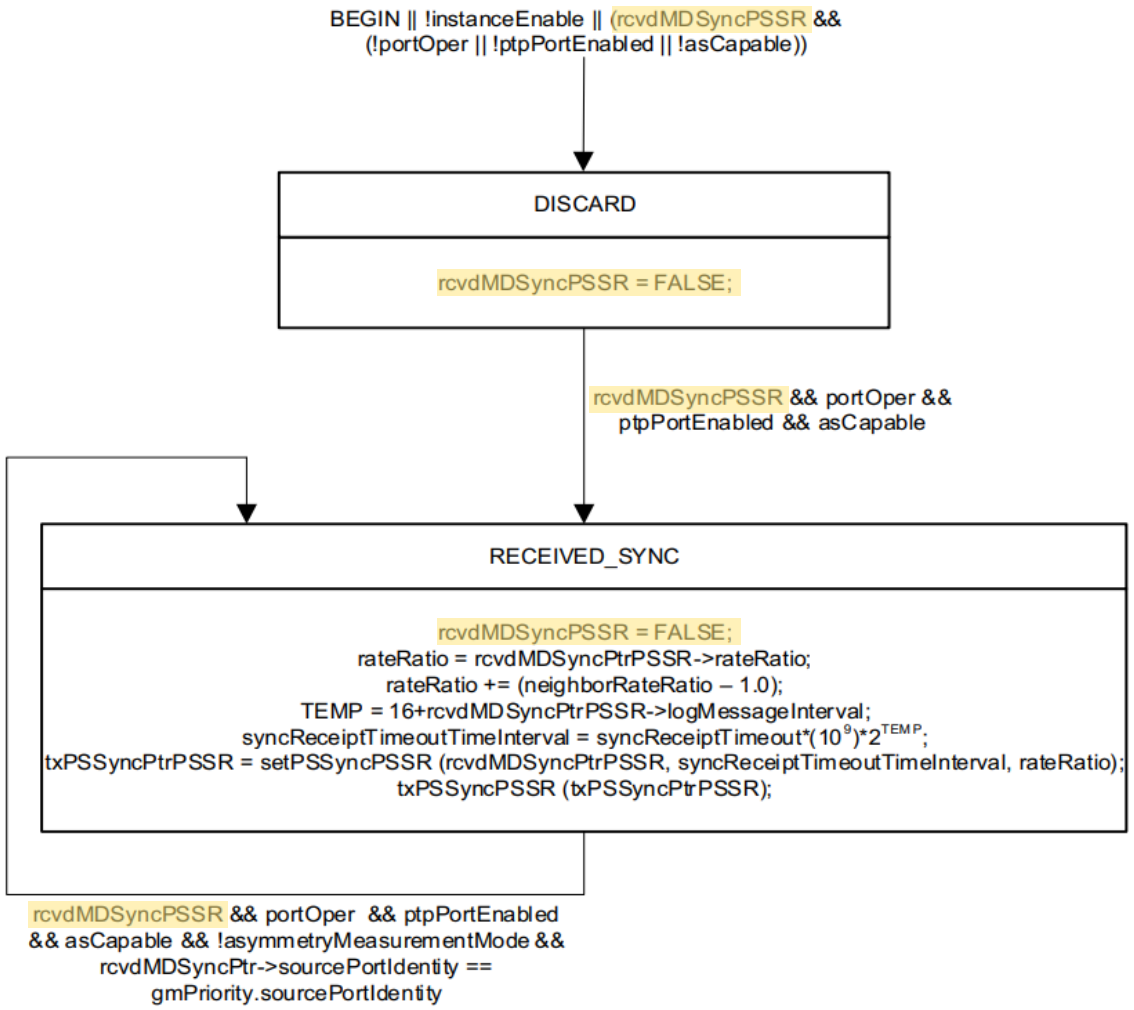


Figure 10-4—PortSyncSyncReceive state machine

#2: Suggested Remedy

1. Remove 14.8.7 (i.e.,rcvdSync Management Variable) from P802.1ASdm.
2. Potentially use rcvdMDSyncPSSR (media independent) instead of rcvdSync (media dependent) in the new suggested state machine from issue #1 in this slide set.

Thank You for Your Attention!

Questions,
Comments,
Opinions,
Ideas?