Several New Minor Issues Pertaining to 802.1ASbt

Geoffrey M. Garner
Consultant

IEEE 802.1 AVB TG
2014.01.17

gmgarner@alum.mit.edu

Introduction

- This presentation describes several minor issues related to 802.1AS that were brought to the attention of the editor since the last 802.1 meeting (November, 2013)
  - Several proposals/recommendations are made on addressing these items
- This presentation also provides the result of the action item given to the editor during the November, 2013, to look into whether making Annex E of 802.1AS (CSN) a numbered clause would be contrary to IEEE or IEEE 802 policies or would require a large effort to justify doing this
  - The editor was also given the action item to update [1]
    - This has been done; revision 1 [1] has been prepared for the current meeting

Action Item

- It was suggested in the November, 2013 802.1 TSN TG meeting that, since CSN (which includes MoCA) is a transport for 802.1AS, just as full-duplex Ethernet, 802.11, and EPON are transports, that Annex E (where the media-dependent layer above CSN transport is specified) should be made a numbered clause
- The CSN numbered clause would immediately follow the current media-dependent numbered clauses, for full-duplex Ethernet, 802.11, and EPON
- The editor as asked to investigate with the IEEE editors whether this would be allowed, given that CSN is not an 802 technology
  - The editor was also asked to find out whether, if this is allowed and is done, a large effort would be needed to justify it
- The editor checked with the IEEE 802.1 chair on to find out who the appropriate person(s) are in IEEE to ask about this
  - However, on being told the question the 802.1 chair indicated that there is not reason why this cannot be done, assuming the TSN TG wishes to do it
- Therefore, it appears it is perfectly acceptable to make Annex E a numbered clause, if the TSN TG wishes to do this

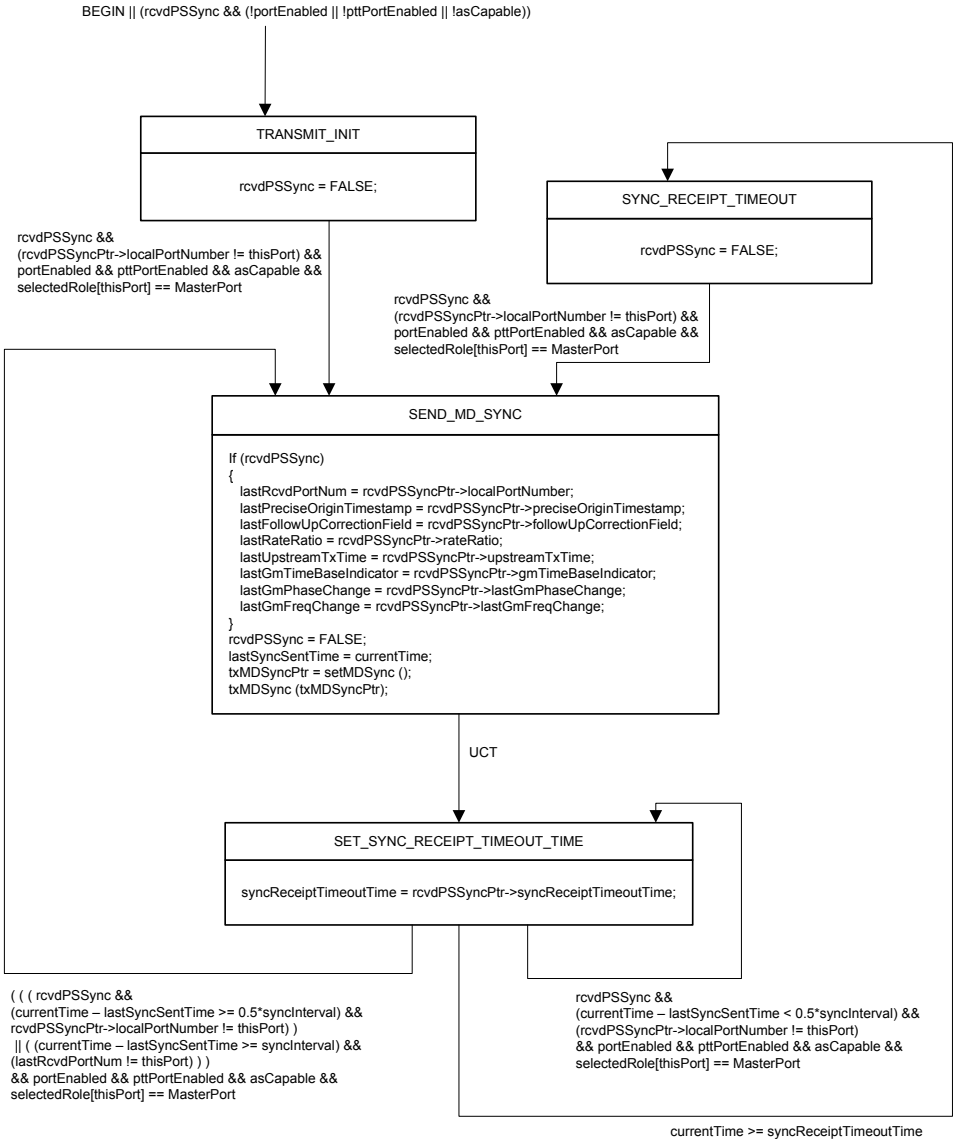
Action Item (cont.)

- In view of the above, it is proposed that the TSN TG decide if it still would like to make Annex E (media-dependent layer specification for CSN transport) a numbered clause
  - It is assumed that, if so, Annex E would be moved to follow the current clause 13
  - All necessary renumbering of clauses and subclauses and fixing of cross-references would need to be done
    - At least at the outset, this would be indicated via one or more editing instructions

Issue 1 - 1

- This issue relates to potential continuous cycling of the PortSyncSyncSend state machine (SM) if a Sync message is received less than ½ Sync interval since the last Sync message was received
  - This issue was pointed out to the editor by [2]
- Consider the PortSyncSyncSend state machine, shown on the next slide (taken from Figure 10-8 of 802.1AS – 2011)

Issue 1 - 2

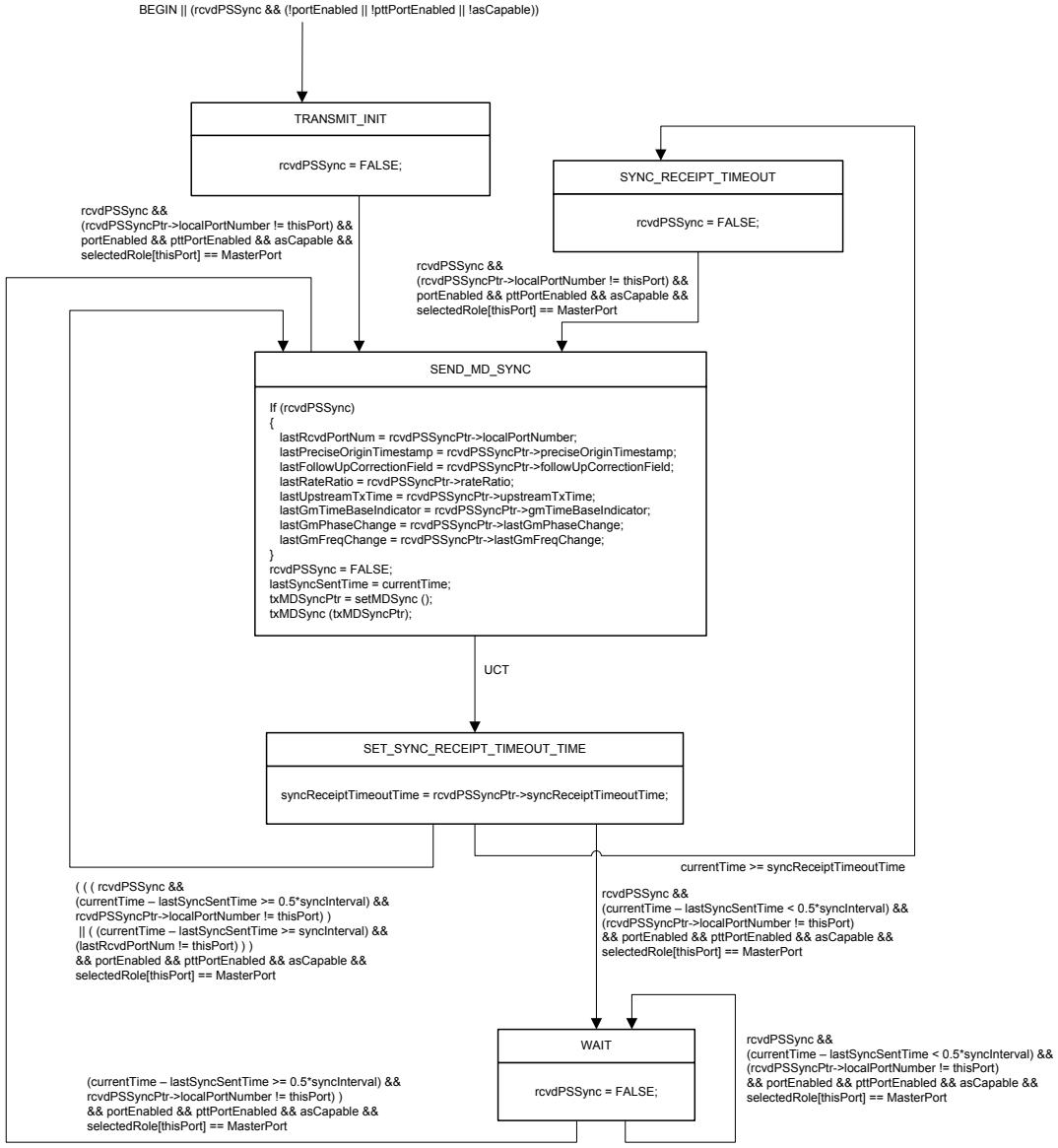


- Suppose the state machine is in the SET_SYNC_RECEIPT_TIMEOUT_TIME state waiting for a Sync message, and a Sync message is received before ½ Sync interval has elapsed since receipt of the last Sync message
- In this case, rcvdPSSync is TRUE, and currentTime – lastSyncSentTime < 0.5*syncInterval

Issue 1 - 3

- In that case the rightmost branch, which cycles back to the SET_SYNC_RECEIPT_TIMEOUT_TIME state, is taken
- However, rcvdPSSync is not set to FALSE
- As a result, this transition continues to occur (as long as portEnabled, pttPortEnabled, asCapable, and other conditions in the branch remain TRUE
- This cycling continues until currentTime – lastSyncSentTime >= 0.5*syncInterval
- This behavior is not desired (among other things, if this were followed exactly it would needlessly use CPU cycles and power)
- A proposed modification to the SM to eliminate this problem is shown on the next slide
- A new state, WAIT, is added, in which rcvdPSSync is set to FALSE
- The SM waits in this state until currentTime – lastSyncSentTime >= 0.5*syncInterval
  - At that point, the SM transitions to the SEND_MD_SYNC state

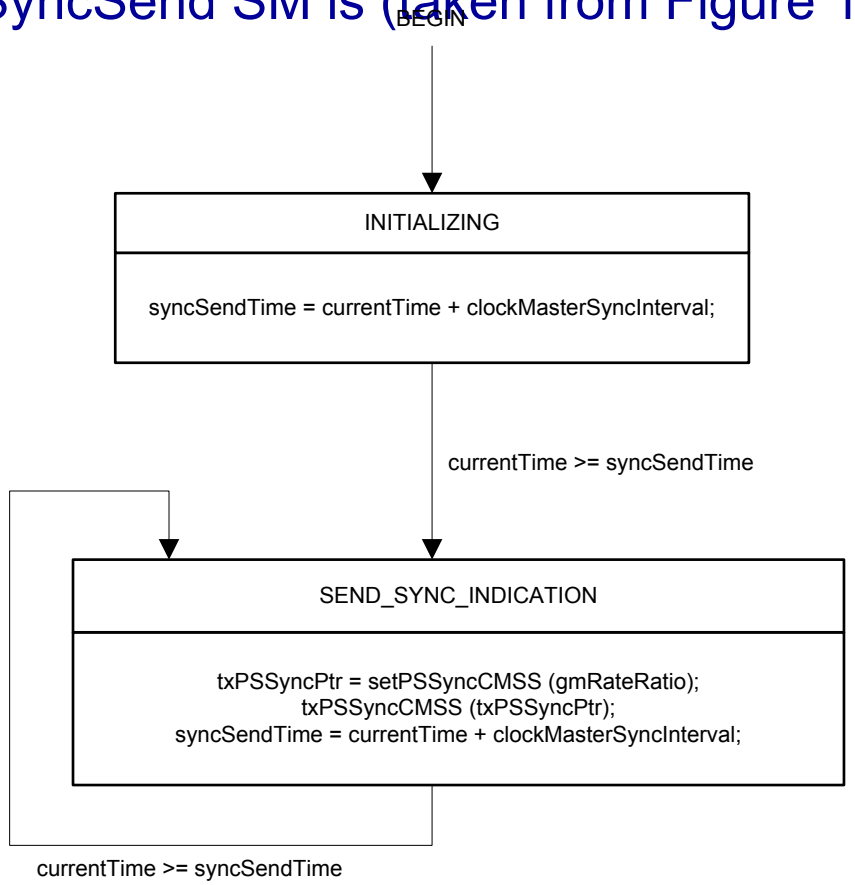
Issue 1 - 4



- If a new Sync message is received while waiting, the SM transitions once back to this state and sets rcvdPSSync to FALSE
- It is proposed to replace the existing PortSyncSyncSend SM (Figure 10-8) with the SM on this slide

Issue 1 - 5

- An additional issue, related to this state machine and the ClockMasterSyncSend SM for the case where the clock is GM, also was identified [2]
- The ClockMasterSyncSend SM is (taken from Figure 10-5 of 802.1AS– 2011)



Issue 1 - 6

- The SM sends a PortSyncSync structure to the SiteSync SM when currentTime exceeds syncSendTime
- syncSendTime is computed as currentTime plus clockMasterSyncInterval
- If the current clock is GM, the SiteSync SM sends this structure to the PortSyncSync Send SM
- Therefore, the time between the sending of successive PortSyncSync structures by the ClockMaster entity to the PortSyncSyncSend SM is clockMasterSyncInterval
- Note that there is no requirement in 802.1AS that clockMasterSyncInterval shall be the same as syncInterval for each port (though it would seem that they should be the same, as both are intended to be the mean interval between successive Sync messages

Issue 1 - 7

- Careful examination of the ClockMasterSyncSend SM and PortSyncSyncSend SM indicates that if clockMasterSyncInterval is slighlty longer than SyncInterval, the time between sending of successive Sync messages will periodically be 0.5*syncInterval
  - In this case, if a Sync message is sent when a PortSyncSync structure is received from the ClockMasterSyncSend SM, then another Sync message will be send by the PortSyncSync SM after syncInterval has elapsed
  - But, a PortSyncSync structure will be received from the ClockMasterSyncSend SM shortly after this, which will result in another Sync message being sent 0.5*syncInterval later
  - The next PortSyncSync structure will be received slightly more than 0.5*syncInterval later, and another Sync message will be sent
  - The next Sync message is sent one syncInterval later, but the next PortSyncSync structure has not yet been received from the ClockMasterSyncSend SM because clockMasterSyncInterval is slightly longer
  - The PortSyncSync structure is received slightly after this, and a Sync message is sent 0.5*syncInterval later
  - The process continues

Issue 1 - 8

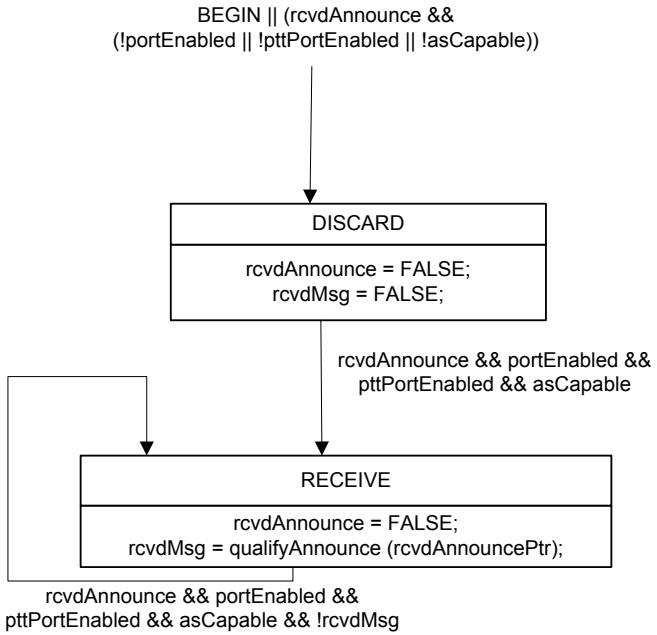
- This behavior was not intended, i.e., if a clock is GM, it was intended that Sync messages be send at the mean Sync rate
  - But the TSN TG should confirm this and, if this is not correct, it should confirm what was/is intended
- Assuming the above bullet item is correct, i.e., it was intended that the GM send Sync messages at the mean Sync rate, then the state machines need to be fixed
- One possible fix would be to make clockMasterSyncInterval and syncInterval equal, i.e., use syncInterval in the ClockMasterSyncSend SM
- In addition, logic could be added to the PortSyncSyncSend SM to check whether the clock is GM and, if so, send Sync as soon as a PortSyncSync structure is received
  - Note that with this logic a clock might send Sync sooner than 0.5*syncInterval if it becomes GM right after sending a Sync Message
  - Comments and suggestions on fixes are welcome

Issue 1 - 9

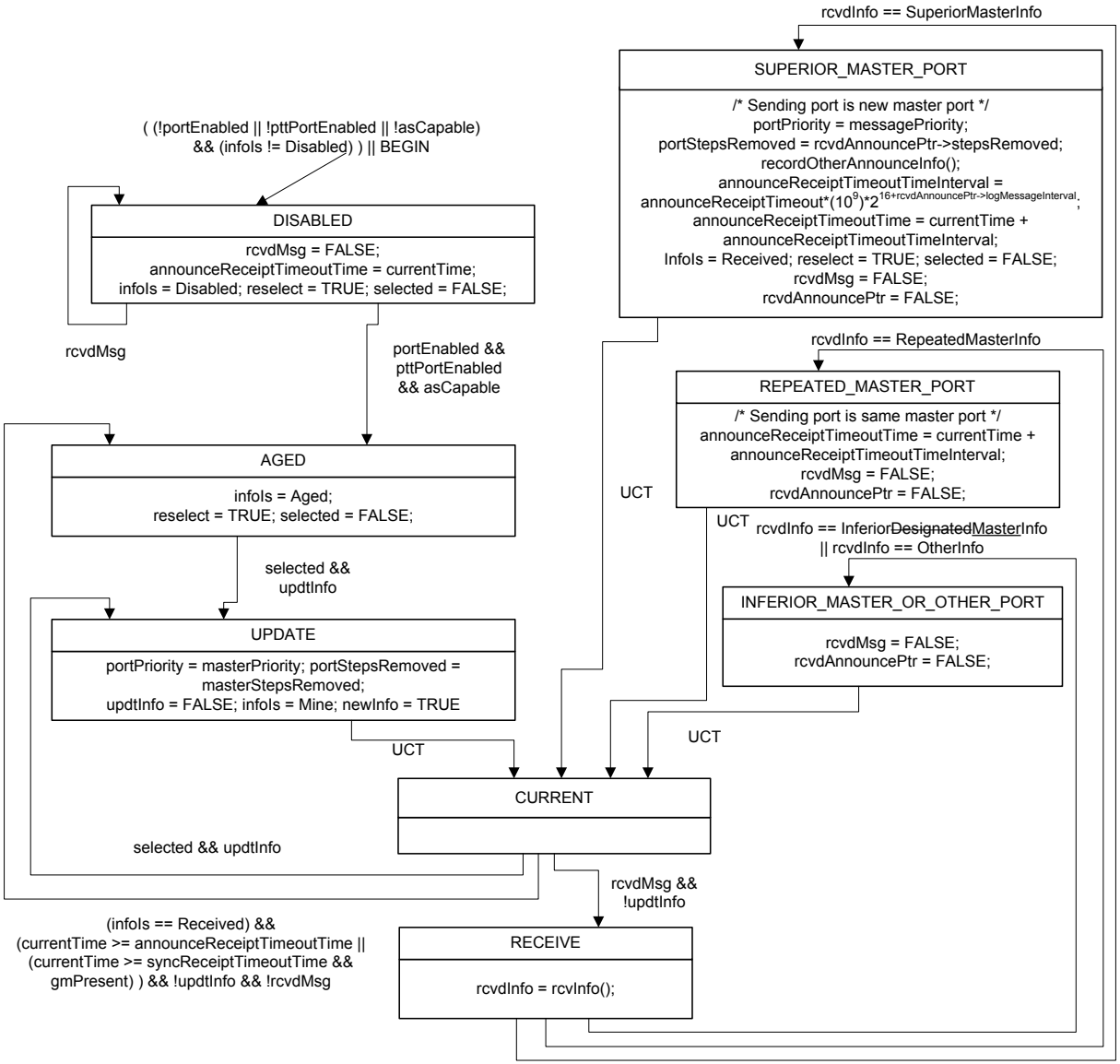
- Note also that IEEE 1588 – 2008 allows some variation in the actual sync interval for a BC, as it is not possible for the actual sync interval to be exactly equal to the specified mean sync interval (i.e., to syncInterval)
  - The time between successive Sync, Announce, and Pdelay_Req messages shall be within ±30% of the mean interval, with 90% confidence
    - This means that 90% of the interval samples shall have values within ±30% of syncInterval
    - The requirement for sending Delay_Req is different (and more complicated and possibly problematic); however, this is not relevant here because 802.1AS does not use Delay_Req
- Some tolerance should be allowed in the sending of Sync, Announce, and Pdelay_Req in 802.1AS
  - The tolerance could be tighter than the 1588 requirement if desired, but should not be looser if compliance with 1588 is desired
    - An absolute upper bound could also be added (1588 does not require an upper bound, nor does it define sync receipt timeout)

Issue 2 - 1

- This issue relates to whether sync receipt timeout should occur if there is a GM change after receipt of an Announce message but no Sync messages are received
  - This issue was pointed out to the editor by [3]
- Consider the PortAnnounceReceive state machine below (taken from Figure 10-12 of 802.1AS-2011) PortAnnounceInformation state machine, shown on the next slide (taken from Figure 10-13 of 802.1AS-Cor1 – 2013)



Issue 2 - 2



- PortAnnounceInformation SM
- Note that the strikeout and insertion is a correction from 802.1AS-Cor1-2013

Issue 2 - 3

- When an Announce message is received an qualified, the PortAnnounceReceive SM sets rcvdMsg to TRUE
  - Note that the pointer rcvdAnnouncePtr is a pointer to a structure that contains the fields of the Announce message
- The PortAnnounceInformation SM transitions from the CURRENT state to the RECEIVE state, and the function rcvInfo() is invoked
  - rcvInfo() compares the received Announce information (messagePriorityVector) with the information on the current GM (portPriorityVector; note that when a new GM is chosen, the PortAnnounceInformation SM and PortRoleSelection SM will together cause the portPriorityVector of each port to contain the information on the current GM)
  - If the new Announce information is better than or the same as the current GM (i.e., messagePriorityVector is superior to or the same as the portPriorityVector), the Announce information is used, and the new announceReceiptTimeoutTime is set
  - **However, a new syncReceiptTimeoutTime is not set**
    - **SyncReceiptTimeoutTime is set by the PortSyncSyncSend SM when a Sync message is sent (see slides 6 and 8)**

Issue 2 - 4

- This is correct (i.e., not setting a new syncReceiptTimeoutTime) if the GM has not changed and Sync (and Announce) messages have been received all along
- However, consider the case where an Announce message is received from a clock (time-aware system) that is better than the current GM, and that clock becomes the new GM, but for some reason it never sends a Sync message
  - In this case, syncReceiptTimeoutTime is not set
  - If the new GM also stops sending Announce after the initial Announce, eventually announce receipt timeout will occur; however, this will take on the order of 3 s if default parameters are used because the default announce interval is 1 s and default announceReceiptTimeout is 3
  - If the new GM does not stop sending Announce, then neither sync receipt timeout nor announce receipt timeout will occur; the new GM will remain GM but never send timing information
    - In this case, the clocks synchronized to this GM will act as if they are free-running, and this will continue indefinitely, or until announce receipt timeout occurs or an Announce message is received from a better clock

Issue 2 - 5

- Both cases in the above main bullet (first-level sub-bullets 2 and 3) violate the goal of GM changeover occurring with in 1 s
  - Neither case is desirable, though the second case is worse
- It would appear that this problem could be fixed by simply setting the syncReceiptTimeoutTime in the SUPERIOR_MASTER_PORT state of the PortAnnounceInformation SM
- However, recall the definition of superior (see 10.3.5 of 802.1AS; note that this definition follows the 13.10 of 802.1Q-2012)
  - The messagePriorityVector is superior to the portPriorityVector if and only if:
    a) The messagePriorityVector is better than the portPriorityVector, or
    b) The Announce message has been transmitted from the same master time-aware system and MasterPort as the portPriorityVector
  - Note that, given 2 priority vectors A and B, A is better, the same as or worse than B if and only if A > B, A = B, or A < B, respectively

Issue 2 - 5

- The reason for (b) in the definition of *superior* is so that the BMCA will be triggered if the current GM downgrades its quality (i.e., a worse Announce message is received from the same clock and port, which results in a worse messagePriorityVector), just as it is triggered a better Announce message is received from a different clock
- However, it appears that (b) will also cause an Announce message received from the current master port, reflecting the current GM, i.e., if the messagePriorityVector is the same as the portPriorityVector, it will still be considered superior, and the PortAnnounceInformation SM will branch to the state SUPERIOR_MASTER_PORT rather than REPEATED_MASTER_PORT
- This behavior seems incorrect and, in any case, with the proposed fix of setting syncReceiptTimeoutTime in the SUPERIOR_MASTER_PORT state would cause it to be set whenever an Announce message is received (whereas it would be desired to set it in this state only when the GM first becomes GM

Issue 2 - 6

- This point will be investigated before any changes are made in the 802.1ASbt draft
- Comments on this are welcome

Issue 3 - 1

- This issue relates to the computation of mean propagation delay using the peer delay mechanism
  - IEEE 802.1AS – 2011 and IEEE 1588 – 2008 do equivalent arithmetic computations when computing mean delay, but the computations are organized differently
    - This appears to not be a problem, because the two sets of computations are equivalent and the same result is obtained given the same input values
  - This issue was pointed out to the editor by [3]
- Recall the notation:
  - T_1 = pdelayReqEventEgressTimestamp
  - T_2 = pdelayReqEventIngressTimestamp
  - T_3 = pdelayRespEventEgressTimestamp
  - T_4 = pdelayRespEventIngressTimestamp

Issue 3 - 2

- In IEEE 802.1AS, the mean path delay is computed as

$$D = \frac{r \cdot (T_4 - T_1) - (T_3 - T_2)}{2}$$

- Where r is the neighborRateRatio, which converts T1 and T4 to the timebase of the peer delay responder
- In IEEE 1588:
  - T_3 above is replaced by the responseOriginTimestamp (of the Pdelay_Resp_Follow_Up message), which contains the pdelayRespEventEgressTimestamp, except for any fractional ns portion
  - T_2 above is replaced by the requestReceiptTimestamp (of the Pdelay_Resp message), which contains the pdelayReqEventIngressTimestamp, except for any fractional ns portion
  - In addition, the correctionFields of Pdelay_Resp and Pdelay_Resp_Follow_Up are subtracted in the numerator
  - The result is (we retain the neighborRateRatio factor):

Issue 3 - 3

$$D = [r \cdot (T_4 - T_1) - (responseOriginTimestamp - requestReceiptTimestamp) - correctionFieldofPdelay\_Resp - correctionFieldofPdelay\_Resp\_Follow\_Up] / 2$$

- In 1588, the fractional ns portion of T_2 is subtracted from the correctionField of Pdelay_Resp, rather than added as in 802.1AS
  - However, the correctionField of Pdelay_Resp is then subtracted in the equation above, and the two minus signs cancel
- The computations in 802.1AS and 1588 are mathematically equivalent, and no change is needed in 802.1ASbt

References - 1

[1] Geoffrey M. Garner, *Interoperability with a One-Step Clock on Receive in 802.1ASbt*, presentation for IEEE 802.1 TSN TG, July 16, 2013, Revision 1, January 15, 2014.

[2] Rune Haugom, emails of November 25, 2013,January 14, 2014, and January 17, 2014.

[3] Chris Hall, email of November 19, 2013.

[4] Christian Boiger, email of November 4, 2013