# Discussion of Maintenance Items 331 and 332

# Revision 1

**Geoffrey M. Garner**
**Huawei (Consultant)**

**gmgarner@alum.mit.edu**

*IEEE 802.1 TSN TG*
2021.08.09

# Introduction

- This presentation provides background information and discussion for 802.1 Maintenance Items 331 and 332
- Maintenance item 331 corrects an error in the computation of meanLinkDelay in the case where common mean link delay service (CMLDS) is used
  - The error occurs only in the case of CMLDS; the computation is correct in the case of instance-specific peer delay
- Maintenance item 332 corrects an error, for the case of external port configuration, that results in an Announce message not being sent immediately when the port state changes to MasterPort
  - The error occurs only in the case of external port configuration; the operation is correct for the case of BMCA

# Maintenance Item #331 - 1

- The computation of meanLinkDelay is organized differently for instance-specific peer delay and CMLDS
  - The difference is only in the organization of the computation; the resulting value for meanLinkDelay is intended to be the same for both (given the same inputs)
- The difference arose because IEEE Std 802.1AS-2011 and IEEE Std 1588-2008 were developed independently and separately
  - This was previously not a problem because gPTP, using transportSpecific 0x1 in 802.1AS-2011 and sdoId 0x100 in 802.1AS-2020, is isolated from other PTP profiles (which use transport specific 0x0 (IEEE 1588-2008) or values of sdoId other than 0x100 (IEEE 1588-2019)
- However, CMLDS is common across all PTP domains, and therefore meanLinkDelay computations by CMLDS must be consistent and comply with IEEE Std 1588-2019
  - Since the CMLDS specifications in IEEE 1588-2019 organize the meanLinkDelay computations in the same way as for instance-specific peer delay in IEEE 1588-2019, 802.1AS-2020 also must organize the CMLDS computations in this manner

# Maintenance Item #331 - 2

- The above was pointed out in comment #46 against 802.1AS-Rev/D5.0
  - The resolution of the comment intended to fix the problem; unfortunately, the implementation of the fix was not correct
  - Some confusion arose from the fact that the 1588 organization of the meanLinkDelay computation involves delayAsymmetry
    - This led to the mistaken impression that 1588 actually adjusted the meanLinkDelay value to account for delayAsymmetry
    - However, more recent examination of 1588 (both 2008 and 2019) indicates that this is not correct; while delayAsymmetry is used in the computation, it is added and subtracted such that its effect cancels out
    - The resulting meanLinkDelay computed in 1588 is the arithmetic mean of the delays in the two directions, just as for instance-specific peer delay in 802.1AS
    - However, the fact that delayAsymmetry is added and subtracted in the 1588 computation, in addition to other differences in organization of the computation, means that the meanLinkDelay computation in 802.1AS must be specified differently for instance-specific peer delay and CMLDS (even though the resulting value is intended to be the same in both cases)
  - In any case, the meanLinkDelay computation in 802.1AS-2020 for the case of CMLDS must be corrected; this is the intent of maintenance item #331

# Maintenance Item #331 - 3

- Relevant aspects of the meanLinkDelay computation, as specified in IEEE 1588-2019, subclause 11.4.2 (note that only the relevant aspects are summarized below, i.e., missing list items are not relevant; see 1588-2019 for the full specification):

Except when using option 16.10, in which case the modifications of this subclause specified in 16.10 shall hold, the actual value of the <meanLinkDelay> is computed as follows for each instance of a peer-to-peer delay measurement between PTP Ports:

a) If required to send a Pdelay_Req message based on the requirements of 9.5.13, the requester PTP Port on PTP Instance-A prepares a Pdelay_Req message as follows:

1) The correctionField (see 13.3.2.9) shall be set to 0.

3) Prior to transmission on an egress PTP Port, the correctionField of the transmitted Pdelay_Req message shall be modified by subtracting the value of the egress path <delayAsymmetry> from the correctionField of the transmitted Pdelay_Req message.

4) The originTimestamp shall be set to 0 or an estimate no worse than ±1 s of the egress timestamp, $t_1$, of the Pdelay_Req message.

5) PTP Instance-A shall send the Pdelay_Req message and generate and save timestamp t1.

# Maintenance Item #331 - 4

❑Relevant aspects of the meanLinkDelay computation (cont.)

c) If the delay responder is a two-step PTP Port, it shall:

1) Generate timestamp $t_2$ upon receipt of the Pdelay_Req message.

2) Prepare a Pdelay_Resp and a Pdelay_Resp_Follow_Up message with the common header of the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively, as specified in 13.3.2 , except for the sequenceId, correctionField, and domainNumber fields, which are as specified in the subsequent points.

3) Copy the correctionField from the Pdelay_Req message to the correctionField of the Pdelay_Resp_Follow_Up message, and set correctionField of the Pdelay_Resp message to 0.

7) Execute either Option A or Option B:

Option B:

i) In the Pdelay_Resp message, set the requestReceiptTimestamp field to the seconds and nanoseconds portion of the time $t_2$, and subtract any fractional nanosecond portion of t2 from the correctionField.

ii) Issue the Pdelay_Resp message and generate timestamp $t_3$ upon sending based on the requirements of 9.5.14.

# Maintenance Item #331 - 5

❑Relevant aspects of the meanLinkDelay computation (cont.)

iii) In the Pdelay_Resp_Follow_Up message, set the responseOriginTimestamp field to the seconds and nanoseconds portion of the time $t_3$, and add any fractional nanosecond portion of $t_3$ to the correctionField.

iv) Issue the Pdelay_Resp_Follow_Up message based on the requirements of 9.5.15.

d) The requester PTP Port on PTP Instance-A, upon receiving a Pdelay_Resp message shall:

1) Generate timestamp $t_4$ upon receipt of the Pdelay_Resp message.

2) To correct for asymmetry of the PTP Link connected to the ingress PTP Port, compute the <correctedPdelayRespCorrectionField> by adding the value of the ingress <delayAsymmetry> to the correctionField of the received Pdelay_Resp message.

4) If the twoStepFlag of the received Pdelay_Resp message is TRUE indicating that a Pdelay_Resp_Follow_Up will be received, compute the <meanLinkDelay> as follows:

<meanLinkDelay> = [($t_4 - t_1$) − (responseOriginTimestamp − requestReceiptTimestamp) − <correctedPdelayRespCorrectionField> − correctionField of Pdelay_Resp_Follow_Up]/2

# Maintenance Item #331 - 6

- In a)3) above, delayAsymmetry is subtracted from the Pdelay_Req correctionField
- In c)3) above, the Pdelay_Req correctionField is copied to the Pdelay_Resp_Follow_Up correctionField, which means that the negative of delayAsymmetry ends up in the Pdelay_Resp_Follow_Up correctionField
- In d)2) above, delayAsymmetry is added to the value of the Pdelay_Resp correctionField
- In the meanLinkDelay computation above, the correctionFields of both Pdelay_Resp and Pdelay_Resp_Follow_Up are subtracted
  - This means that the delayAsymmetry value, which is included in the Pdelay_Resp correctionField with a positive sign and in the Pdelay_Resp_Follow_Up correctionField with a negative sign, cancels

# Maintenance Item #331 - 7

- Also in the meanLinkDelay computation
  - Any fractional ns part of $t_2$ (the timestamp of the arrival of Pdelay_Req at the responder) is subtracted from the correctionField of Pdelay_Resp, which is subtracted when computing meanLinkDelay
  - This means that the fractional ns part of $t_2$ is added (just as the requestReceiptTimestamp is added)
  - Any fractional ns part of $t_3$ (the timestamp of the departure of Pdelay_Resp from the responder) is added to the correctionField of Pdelay_Resp_Follow_Up, which is subtracted when computing meanLinkDelay
  - This means that the fractional ns part of $t_3$ is subtracted (just as the responseOriginTimestamp is subtracted)

# Maintenance Item #331 - 8

- Relevant aspects of the meanLinkDelay computation for the CMLDS case, as specified (and as published) in IEEE 802.1AS-2020, subclauses 11.2.19 and 11.2.20 (note that only the relevant aspects are summarized below, i.e., missing list items are not relevant; see 802.1AS-2020 for the full specification):

11.2.19.2.13 s: A variable whose value is +1 if this state machine is invoked by the instance-specific peer-to-peer delay mechanism and –1 if this state machine is invoked by the CMLDS. The data type for s is Integer8.

11.2.19.3.1 setPdelayReq(): Creates a structure containing the parameters (see 11.4) of a Pdelay_Req message to be transmitted, and returns a pointer, txPdelayReqPtr (see 11.2.19.2.6), to this structure. The parameters are set as follows:

c) correctionField is set to

1) 0 if this state machine is invoked by the instance-specific peer-to-peer delay mechanism, and

2) –delayAsymmetry (i.e., the negative of delayAsymmetry) if this state machine is invoked by the CMLDS.

# Maintenance Item #331 - 9

- Relevant aspects of the meanLinkDelay computation (cont.):

11.2.20.3.1 setPdelayResp(): Creates a structure containing the parameters (see 11.4) of a Pdelay_Resp message to be transmitted, and returns a pointer, txPdelayRespPtr (see 11.2.20.2.3), to this structure. The parameters are set as follows:

c) requestReceiptTimestamp is set equal to the pdelayReqEventIngressTimestamp (see 11.3.2) of the corresponding Pdelay_Req message, with any fractional nanoseconds portion truncated.

d) correctionField is set equal to the following:

1) The fractional nanoseconds portion of the pdelayReqEventIngressTimestamp of the corresponding Pdelay_Req message if this state machine is invoked by the instance-specific peer-to-peer delay mechanism and

2) Minus the fractional nanoseconds portion of the pdelayReqEventIngressTimestamp of the corresponding Pdelay_Req message if this state machine is invoked by CMLDS.

# Maintenance Item #331 - 10

❑Relevant aspects of the meanLinkDelay computation (cont.):

11.2.20.3.1 setPdelayRespFollowUp(): Creates a structure containing the parameters (see 11.4) of a Pdelay_Resp_Follow_Up message to be transmitted, and returns a pointer, txPdelayRespFollowUpPtr (see 11.2.20.2.4), to this structure. The parameters are set as follows:

c) responseOriginTimestamp is set equal to the pdelayRespEventEgressTimestamp (see 11.3.2) of the corresponding Pdelay_Resp message, with any fractional nanoseconds truncated.

d) correctionField is set equal to the following:

1) The fractional nanoseconds portion of the pdelayRespEventEgressTimestamp of the corresponding Pdelay_Resp message if this state machine is invoked by the instance-specific peer-to-peer delay mechanism and

2) The sum of the correctionField of the corresponding Pdelay_Req message and the fractional nanoseconds portion of the pdelayRespEventEgressTimestamp of the corresponding Pdelay_Resp message if this state machine is invoked by CMLDS,

# Maintenance Item #331 - 11

## ❑Relevant aspects of the meanLinkDelay computation (cont.):

11.2.19.3.4 computePropTime(): Computes the mean propagation delay on the PTP Link attached to this MD entity, D, and returns this value. D is given by Equation (11-5).

$$D = \frac{r \cdot (t_4 - t_1) - (t_3 - t_2)}{2} \qquad (11\text{-}5)$$

where

$t_4$ is pdelayRespEventIngressTimestamp (see 11.3.2.1) for the Pdelay_Resp message received in response to the Pdelay_Req message sent by the MD entity, in nanoseconds; the pdelayRespEventIngressTimestamp is equal to the timestamp value measured relative to the timestamp measurement plane, minus any ingressLatency (see 8.4.3)

$t_1$ is pdelayReqEventEgressTimestamp (see 11.3.2.1) for the Pdelay_Req message sent by the P2PPort entity, in nanoseconds

$t_2$ is the sum of (1) the ns field of the requestReceiptTimestamp, (2) the seconds field of the requestReceiptTimestamp multiplied by $10^9$, and (3) the correctionField multiplied by s (see 11.2.19.2.13) and then divided by $2^{16}$ (i.e., the correctionField is expressed in nanoseconds plus fractional nanoseconds), of the Pdelay_Resp message received in response to the Pdelay_Req message sent by the MD entity

# Maintenance Item #331 - 12

- Relevant aspects of the meanLinkDelay computation (cont.):

$t_3$ is the sum of (1) the ns field of the responseOriginTimestamp, (2) the seconds field of the responseOriginTimestamp multiplied by $10^9$, and (3) the correctionField divided by $2^{16}$ (i.e., the correctionField is expressed in nanoseconds plus fractional nanoseconds), of the Pdelay_Resp_Follow_Up message received in response to the Pdelay_Req message sent by the MD entity

$r$ is the current value of neighborRateRatio for this MD entity (see 10.2.5.7)

# Maintenance Item #331 - 13

- For the case of CMLDS:
  - In 11.2.19.3.1, c)2), above, delayAsymmetry is subtracted from the Pdelay_Req correctionField
  - In 11.2.20.3.3, d)2), above, the Pdelay_Resp_Follow_Up correctionField is set equal to the sum of the Pdelay_Req correctionField and the fractional ns portion of the pdelayRespEventEgressTimestamp, which means that the negative of delayAsymmetry ends up in the Pdelay_Resp_Follow_Up correctionField
  - This means that the negative of delayAsymmetry is included in $t_3$
  - However, unlike in IEEE 1588-2019, 11.4.2, d)2) (slide 7), delayAsymmetry is not added to the value of the Pdelay_Resp correctionField, which means that delayAsymmetry is not included in $t_2$
  - In the meanLinkDelay computation above, the correctionFields of both Pdelay_Resp and Pdelay_Resp_Follow_Up are subtracted ($t_3$ includes the correctionField of Pdelay_Resp_Follow_Up, and is multiplied by $s = -1$, which means it has a negative sign)
  - But, since delayAsymmetry is not added to the Pdelay_Resp correctionField, the –delayAsymmetry of the Pdelay_Resp_Follow_Up correctionField does not cancel

# Maintenance Item #331 - 14

- Also in the meanLinkDelay computation
  - $t_2$ includes the correctionField of Pdelay_Resp multiplied by s = -1, and the correctionField of Pdelay_Resp includes minus the fractional ns portion of the pdelayReqEventIngress timestamp as indicated in 11.2.20.3.1 d)2) (see slide 11)
  - Since $t_2$ is subtracted in Eq. (11-5) (see slide 13), the fractional ns part of $t_2$ is added (just as the requestReceiptTimestamp is added)
  - $t_3$ includes the correctionField of Pdelay_Resp_Follow_Up, which is subtracted when computing meanLinkDelay
  - This means that the fractional ns part of $t_3$ is subtracted (just as the requestReceiptTimestamp is subtracted)

# Maintenance Item #331 - 15

- The above indicates that the quantity delayAsymmetry must be added to one of the terms when computing meanLinkDelay, to cancel the –delayAsymmetry contribution that is already included
- A simple fix is to add an additional term delayAsymmetry to $t_3$ when computing meanLinkDelay using Eq. (11-5)
- With this fix, meanLinkDelay for the case of CMLDS is equal to the mean of the link delays in the two directions, expressed in the responder timebase (due to the presence of the neighborRateRatio *r* that multiplies the term $t_4 - t_1$ in Eq. (11-5)
  - Note that this fix is simpler than the fix given in Maintenance Item #331
  - The writeup in the maintenance item is not correct; the quantity (4) of t2 in the maintenance item writeup must be subtracted, not added
  - However, the fix described here is simpler; it can be obtained from the fix in the maintenance writeup, but corrected by subtracting rather than adding the quantity (4) in t2, by adding neighborRateRatio*delayAsymmetry to t2 and to t3 (doing this cancels the –neighborRateRatio*delayAsymmetry of term (4) of t2 and leaves delayAsymmetry in term (4) of t3 of the maintenance writeup

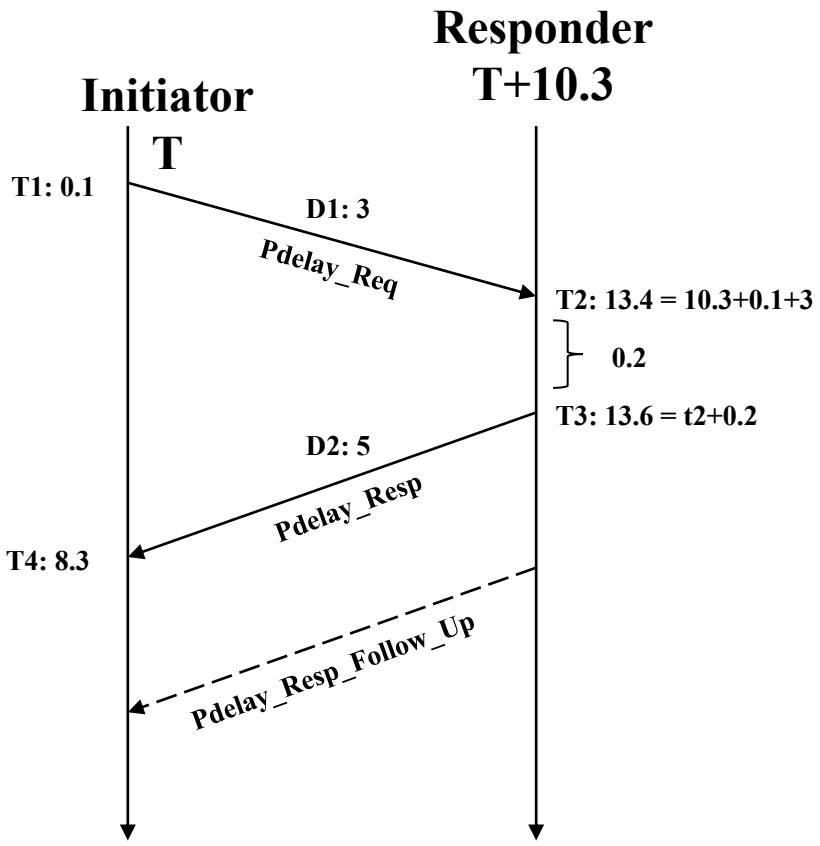
# Maintenance Item #331 - 16

- Note: As indicated in 10.2.5.9 of 802.1AS-2020, delayAsymmetry is expressed in the grandmaster timebase; however, this is not relevant in the computation of meanLinkDelay because, with the fix, the quantity is added and subtracted and, therefore, does not affect the value of meanLinkDelay
- Note also: The description of differences between the instance-specific peer-to-peer delay mechanism and CMLDS in computations of meanLinkDelay, in 11.2.17.2, must be revised to reflect the above.
  - In addition, the term (correctionField of Pdelay_Resp) in Eq. (11-6) must be multiplied by $s$ (see 11.2.19.2.13)
  - In addition, in NOTE 1 of 11.2.19.3.3, “subtract” must be changed to “add” for consistency with b)4) of 11.2.19.3.3
- Finally, the computation of upstreamTxTime in 11.2.14.2.1 f) in the case of CMLDS must be the same as in the case of instance-specific peer delay, i.e.,

$$\text{upstreamTxTime} = \text{syncEventIngressTimestamp} - (\text{meanLinkDelay}/\text{neighborRateRatio}) - (\text{delayAsymmetry}/\text{rateRatio})$$

# Maintenance Item #331 - 17 (Example 1)

**neighborRatito is 1.**




- delayAsy = (D2 – D1) / 2 = 1 (According to Clause 8.3)
- P2P mechanism (published 802.1AS-2020)
  - ① Preq_TS = 0, Preq_CF = 0 (item b of 11.2.17.2)
  - ② Presp_TS = 13, Presp_CF = 0.4 (item c of 11.2.17.2)
  - ③ Presp_Follow_TS = 13, Presp_Follow_CF = 0.6 (item d of 11.2.17.2)
  - ④ According to 11.2.19.3.4,
    - I. t1 = T1 = 0.1
    - II. t2 = Presp_TS + Presp_CF = 13.4 = T2
    - III. t3 = Presp_Follow_TS + Presp_Follow_CF = 13.6 = T3
    - IV. t4 = T4 = 8.3
  - ⑤ According to 11.2.19.3.4,
    MeanLinkDelay = [(t4-t1) – (t3-t2)]/2 = 4

    Correct
- Common_P2P mechanism (published 802.1AS-2020)
  - ① Preq_TS = 0, Preq_CF = -delayAsy = -1 (item *b* of 11.2.17.2)
  - ② Presp_TS = 13, Presp_CF = -0.4 (item *c* of 11.2.17.2)
  - ③ Presp_Follow_TS = 13, Presp_Follow_CF = Preq_CF + 0.6 = -0.4 (item *d* of 11.2.17.2)
  - ④ According to 11.2.19.3.4,
    - I. t1 = T1 = 0.1
    - II. t2 = Presp_TS – Presp_CF = 13.4 (item *e* of 11.2.17.2)
    - III. t3 = Presp_Follow_TS + Presp_Follow_CF = 12.6
    - IV. t4 = T4 = 8.3
  - ⑤ According to 11.2.19.3.4,
    MeanLinkDelay = [(t4-t1) – (t3-t2)]/2 = 4.5

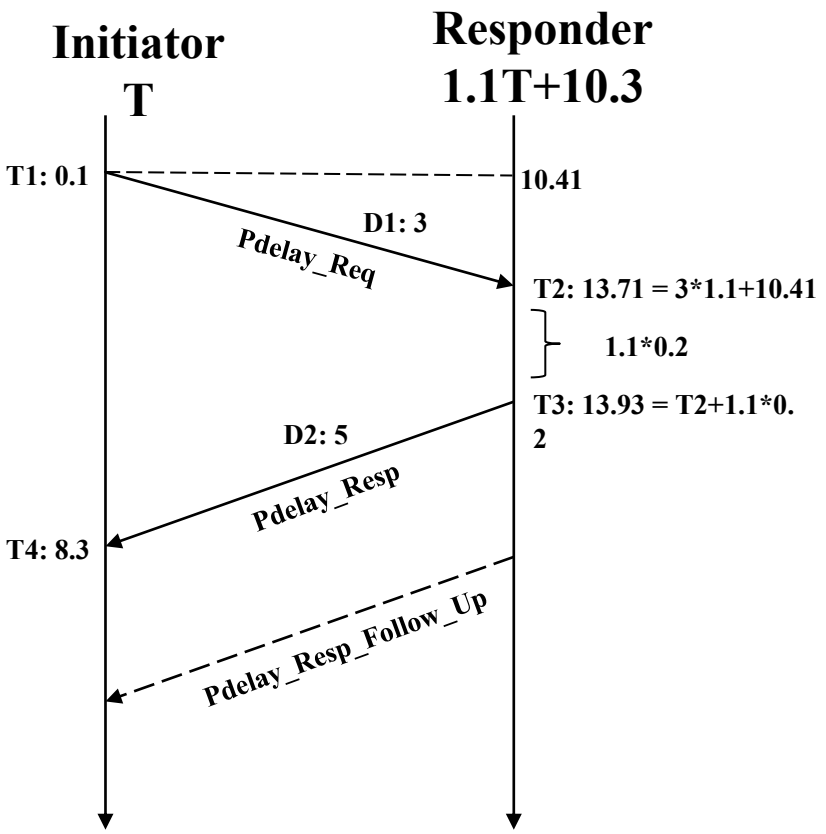
    Incorrect

Common P2P mechanism with correction of this presentation
III. t3 = Presp_Follow_TS + Presp_Follow_CF + delayAsy
= 13 – 0.4 + 1 = 13.6
MeanLinkDelay = [(8.3 – 0.1) – (13.6 – 13.4)]/2 = 4

Correct

# Maintenance Item #331 - 18 (Example 2)

**neighborRateRatio isn't 1.**




- dealyAsy = (D2 – D1) / 2 = 1 (According to Clause 8.3)
- P2P mechanism (published 802.1AS-2020)
  - ① Preq_TS = 0, Preq_CF = 0
  - ② Presp_TS = 13, Presp_CF = 0.71
  - ③ Presp_Follow_TS = 13, Presp_Follow_CF = 0.93
  - ④ According to 11.2.19.3.4,
    - I. t1 = T1 = 0.1
    - II. t2 = Presp_TS + Presp_CF = 13.71 = T2
    - III. t3 = Presp_Follow_TS + Presp_Follow_CF = 13.93 = T3
    - IV. t4 = T4 = 8.3
  - ⑤ According to 11.2.19.3.4

    meanLinkDelay = [1.1*(t4-t1) – (t3-t2)]/2 =
    [1.1*(8.3-0.1)-(13.93-13.71)/2 =
    (1.1*8.2 – 0.22)/2 = (9.02-0.22)/2= 4.4

    Correct
- Common_P2P mechanism (published 802.1AS-2020)
  - ① Preq_TS = 0, Preq_CF = -delayAsy = -1
  - ② Presp_TS = 13, Presp_CF = – 0.71
  - ③ Presp_Follow_TS = 13, Presp_Follow_CF = Preq_CF + 0.93 = -0.07
  - ④ According to 11.2.19.3.4,
    - I. t1 = T1 = 0.1
    - II. t2 = Presp_TS – Presp_CF = 13.71
    - III. t3 = Presp_Follow_TS + Presp_Follow_CF = 12.93
    - IV. t4 = T4 = 8.3
  - ⑤ According to 11.2.19.3.4

    meanLinkDelay = [1.1*(t4-t1) – (t3-t2)]/2 =
    [1.1*(8.3-0.1)-(12.93-13.71)/2 =
    (1.1*8.2 +0.78)/2 = (9.02+0.78)/2= 4.9

    Incorrect

  Common P2P mechanism with correction of this presentation

  III. t3 = Presp_Follow_TS + Presp_Follow_CF + delayAsy
  = 13 – 0.07 + 1 = 13.93

  MeanLinkDelay = [1.1(8.3 – 0.1) – (13.93 – 13.71)]/2 = 4.4

  Correct

# Maintenance Item #331 Proposal - 1

- Change the description of $t_3$ in 11.2.19.3.4 to read:

$t_3$ is the sum of (1) the ns field of the responseOriginTimestamp, (2) the seconds field of the responseOriginTimestamp multiplied by $10^9$, (3) the correctionField divided by $2^{16}$ (i.e., the correctionField is expressed in nanoseconds plus fractional nanoseconds), of the Pdelay_Resp_Follow_Up message received in response to the Pdelay_Req message sent by the MD entity, and (4) delayAsymmetry if this state machine is invoked by CMLDS

# Maintenance Item #331 Proposal - 1

- Change Note 3 of 11.2.19.3.4, including Eq. (11-6), to read (in Eq. (11-6), the term (correctionField of Pdelay_Resp) must be multiplied by s):

NOTE 3—In IEEE Std 1588-2019, the computation of Equation (11-5) is organized differently from the organization used for instance-specific peer delay in the present standard. Using the definitions of t2 and t3 above, Equation (11-5) can be rewritten as shown in Equation (11-6).

$$D = [r\cdot(t_4 - t_1) - (\text{responseOriginTimestamp} - \text{requestReceiptTimestamp}) + s\cdot(\text{correctionField of Pdelay\_Resp}) - (\text{correctionField of Pdelay\_Resp\_Follow\_Up})] / 2$$

where each term is expressed in units of nanoseconds as described in the definitions of t1, t2, t3, and t4 above. In IEEE Std 1588-2019, the fractional nanoseconds portion of t2 is subtracted from the correctionField of Pdelay_Resp, rather than added as in ~~this~~ the present standard for instance-specific peer delay [see 11.2.20.3.1 d)1]; however, the correctionField of Pdelay_Resp is then subtracted in Equation (11-6) rather than added [in Eq. (11-6) for instance-specific peer delay, where $s = 1$], and the two minus signs cancel each other. The computations of *D* in this standard and IEEE Std 1588-2019 are mathematically equivalent. The organization of the computation used in IEEE Std 1588-2019 must be used with CMLDS in the present standard for interoperability with IEEE Std 1588-2019 (see 11.2.17.2).

# Maintenance Item #331 Proposal – 2

- Change NOTE 1 of 11.2.19.3.3 to read:

NOTE 1—If delayAsymmetry does not change during the time interval over which neighborRateRatio is computed, it is not necessary to ~~subtract~~ add it if this state machine is invoked by CMLDS because in that case it will be canceled when computing the difference between earlier and later correctedResponderEventTimestamps.

  - i.e., “subtract” is changed to “add”

- Edit 11.2.14.2.1 f) as indicated on the following two slides
  - Note that in the revision, delayAsymmetry must be divided by the sum of rateRatio and neighborRateRatio, because the current value of rateRatio is the value at the upstream PTP Instance, while the value needed is the value at the receiving PTP Instance. rateRatio is updated by the PortSyncSyncReceive state machine (see 10.2.8), which has not been invoked yet.
- Edit 11.2.17.2 as indicated in the attached pdf file (it can be accessed by making the panel on the left side of the Acrobat window visible and then clicking on the paper clip; the attached pdf file will then appear)

# Maintenance Item #331 Proposal – 3

f) upstreamTxTime is set equal to the syncEventIngressTimestamp for the most recently received Sync message (see 11.4.3), minus the mean propagation time on the PTP Link attached to this PTP Port (meanLinkDelay; see 10.2.5.8) divided by neighborRateRatio (see 10.2.5.7), ~~and, if and only if the state machine is invoked by the instance-specific peer-to-peer delay mechanism,~~ minus delayAsymmetry (see 10.2.5.9) for this PTP Port divided by the sum of rateRatio and neighborRateRatio [see item e) in this subclause]. The syncEventIngressTimestamp is equal to the timestamp value measured relative to the timestamp measurement plane, minus any ingressLatency (see 8.4.3). The upstreamTxTime can be written as follows:

~~State machine invoked by instance-specific peer-to-peer delay mechanism:~~

upstreamTxTime = syncEventIngressTimestamp –(meanLinkDelay/neighborRateRatio) – (delayAsymmetry/(rateRatio + neighborRateRatio))

~~State machine invoked by CMLDS:~~

~~upstreamTxTime = syncEventIngressTimestamp – (meanLinkDelay/neighborRateRatio)~~

# Maintenance Item #331 Proposal – 4

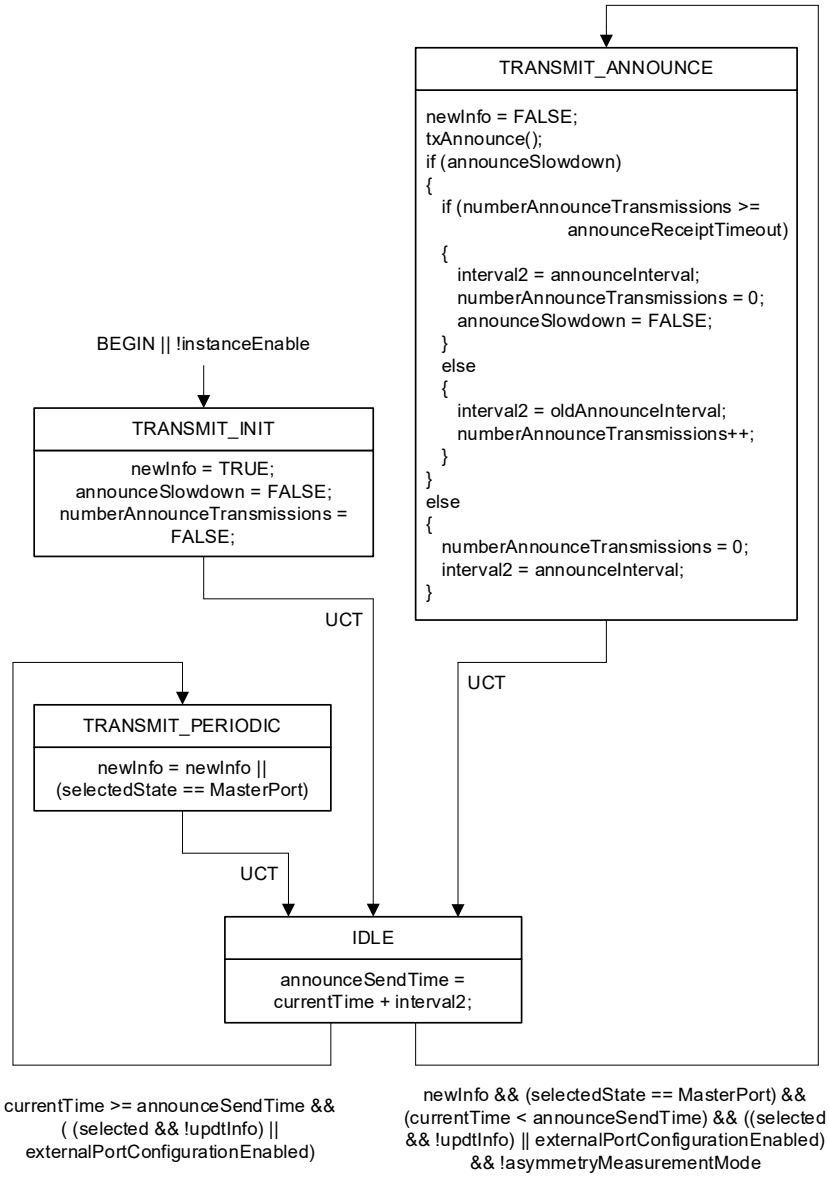
NOTE 1—The mean propagation time is divided by neighborRateRatio to convert it from the time base of the PTP Instance at the other end of the attached PTP Link to the time base of the current PTP Instance. ~~If the instance-specific peer-to-peer delay mechanism is used (i.e., portDS.delayMechanism is P2P),~~ The quantity delayAsymmetry is divided by rateRatio to convert it from the time base of the Grandmaster Clock to the time base of the current PTP Instance. ~~The first~~ These two quotients ~~is~~ are then subtracted from syncEventIngressTimestamp~~, and the second quotient is subtracted from syncEventIngressTimestamp if the instance-specific peer-to-peer delay mechanism is used~~. The syncEventIngressTimestamp is measured relative to the time base of the current PTP Instance. See 11.2.17.2 for more detail.

# Maintenance Item #332 - 1

- When using the BMCA in both IEEE Std 802.1AS-2011 and IEEE Std 802.1AS-2020, an Announce message is sent on a port whenever the state of the port changes to MasterPort
- This behavior is neither required nor prohibited by IEEE Std 1588-2008 and IEEE Std 1588-2019
  - 1588 requires only that Announce messages be sent at an average rate based on the announce interval (and with requirements on the allowed variability of the actual rate); however, an Announce message need not be sent at the instant the port state changes to MASTER
  - However, during the development of 802.1AS-2011, it was desired that an Announce message be sent when the state changes to MasterPort
    - One reason for this was to ensure that information on a new grandmaster (GM) be propagated as soon as possible so that downstream PTP Instances know who the current GM is (this was needed because the Sync messages do not indicate the current GM)
- While this behavior is specified correctly for BMCA, the specification is not correct for the case of external port configuration
  - For external port configuration, the specifications of 1588-2019 do not result in an Announce message being sent immediately when the port state change to MasterPort
- The intent of maintenance item #332 is to fix this

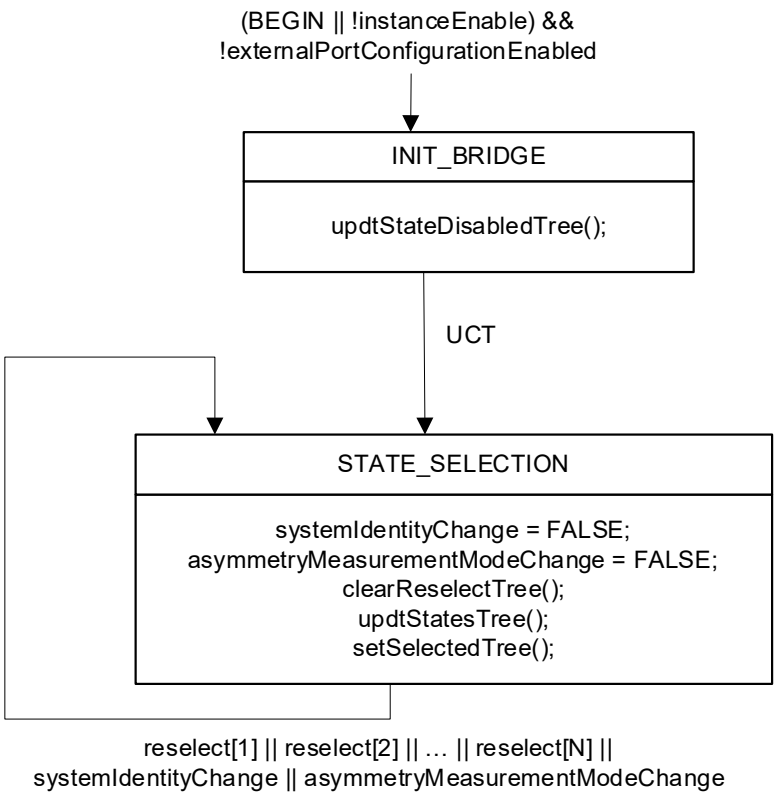
# Maintenance Item #332 - 2

- An Announce message is transmitted by the PortAnnounceTransmit state machine when the global variable newInfo is set to TRUE
- From the IDLE state, if newInfo is TRUE and selectedState is MasterPort (and currentTime is not yet announceSendTime), the right branch is taken and an Announce message is sent



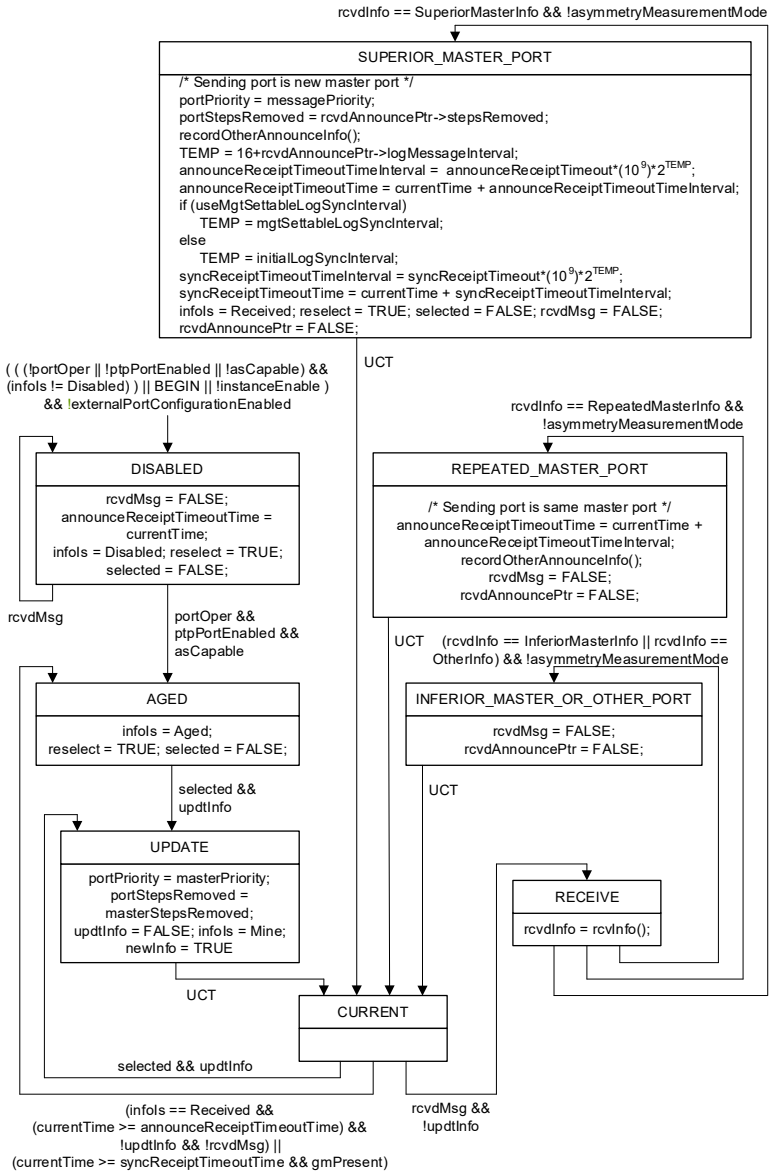
# Maintenance Item #332 - 3

- In the case of BMCA, newInfo is set to TRUE as a result of actions by the PortStateSelection state machine (via the function updtStatesTree(), see 10.3.13.2.4) and the PortAnnounceInformation state machine
- updtStatesTree() is invoked when reselect[j] is becomes TRUE for any port j (i.e., the triggering of the BMCA)
  - This could occur, e.g., due to receipt of an Announce message (see the PortAnnounceInformation state machine in 10.3.12 and on the next slide)



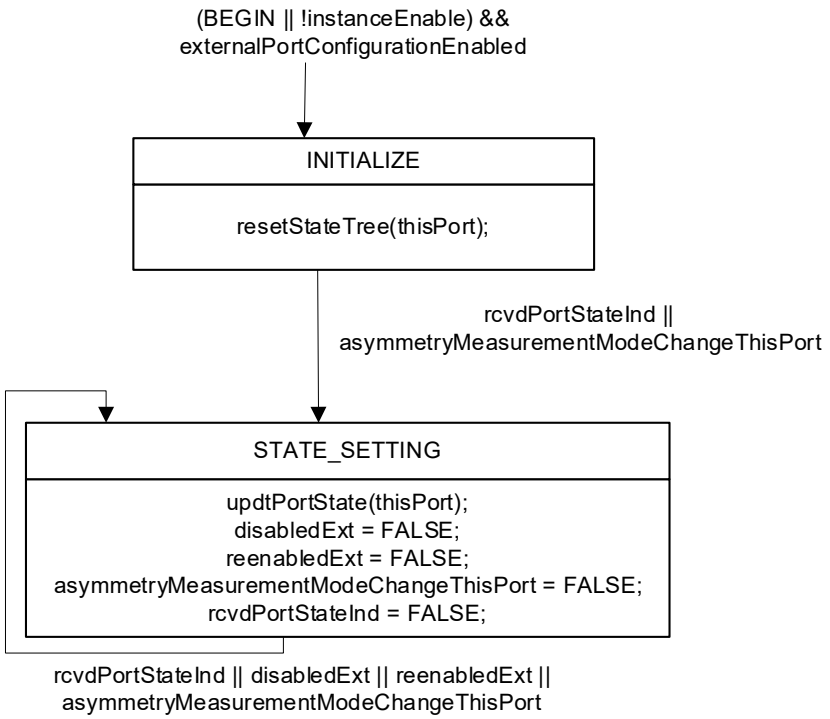
# Maintenance Item #332 - 4

- updtStatesTree() determines the port state in item f) of 10.3.13.2.4; in f)3), f)4), and f)8) of 10.3.13.2.4, the port state is set to MasterPort and the variable updtInfo is set to TRUE
- This in turn causes the PortAnnounceInformation state machine to transition from the CURRENT state to the UPDATE state, which in turn sets updtInfo to FALSE and newInfo to TRUE
  - As indicated two slides previously, this causes the PortAnnounceTransmit state machine to transmit an Announce message



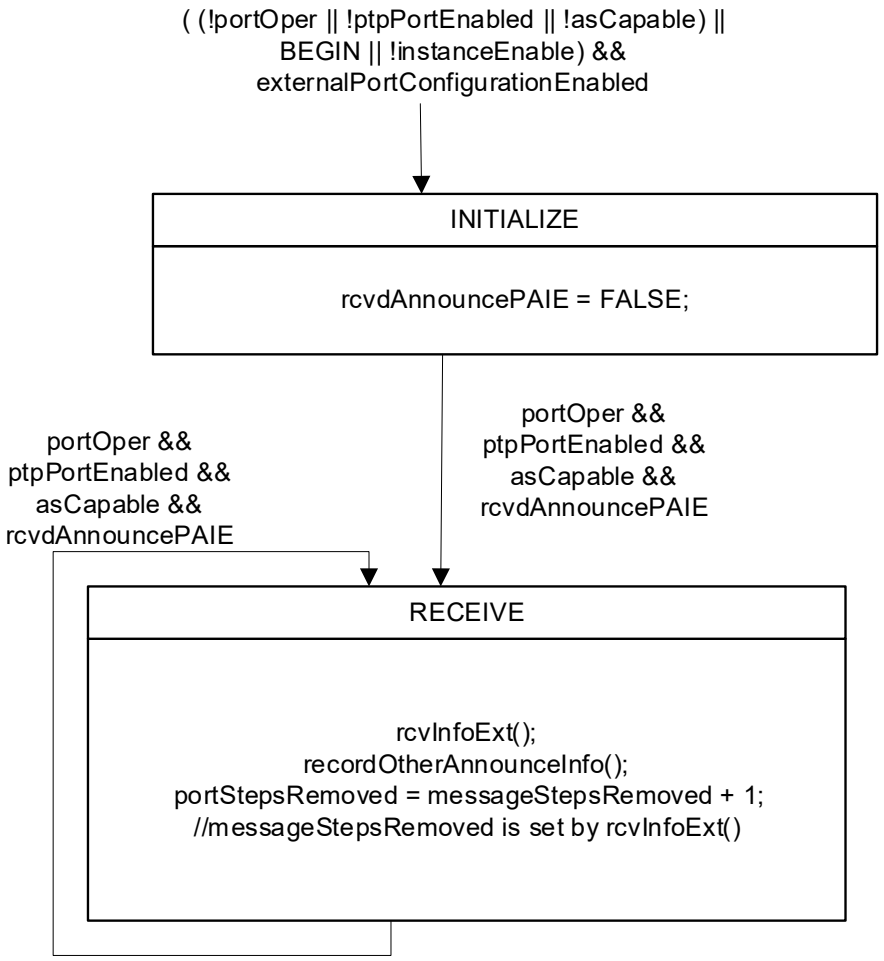
# Maintenance Item #332 - 5

- In the published 802.1AS-2020, when external port configuration is used newInfo is not set to TRUE, and an Announce message is not immediately sent
- In external port configuration, the port state is set by the function updtPortState(thisPort) in the PortStateSettingExt state machine



# Maintenance Item #332 - 6

- In external port configuration, the PortAnnounceInformation state machine is not invoked because the BMCA is not triggered by receipt of an Announce message
  - Instead, the PortAnnounceInformationExt state machine is invoked, which simply records information contained in the Announce message (e.g., time properties information)
- The problem can be fixed by setting newInfo to TRUE in item c)3) of updtPortState(j), where the port state is set



# Maintenance Item #332 - 7

- In addition, minor changes are needed for the overview diagram for External Port Configuation (Figure 10-12)
  - The variable newInfo should be removed from the arrow going from the PortAnnounceInformationExt block to the PortAnnounceTransmit block
  - The variable newInfo should be added to the arrow going from the PortStateSetting block to the PortAnnounceTransmit block.

# Maintenance Item #332 - 8

- Add the following sentence to item c)3)of the updtPortState(j) function (see 10.3.15.2.2): If portStateInd is equal to MasterPort, set newInfo to TRUE. Item c)3) then reads:
  - 3) selectedState[j] is set to portStateInd. If portStateInd is equal to MasterPort, set newInfo to TRUE.
- In Figure 10-12, remove the variable newInfo from the arrow going from the PortAnnounceInformationExt block to the PortAnnounceTransmit block (note that newInfo is not set by the PortAnnounceInformationExt block)
- In Figure 10-12, add the variable newInfo to the arrow going from the PortStateSetting block to the PortAnnounceTransmit block

# Thank you