11. Media-dependent layer specification for full-duplex point-to-point links

11.1 Overview

11.1.1 General

A port attached to a full-duplex point-to-point PTP Link uses the PTP peer delay protocol to measure propagation delay on the PTP Link. An overview of the propagation delay measurement is given in 11.1.2. Synchronization information is transported using the PTP messages Sync and Follow_Up. An overview of the transport of synchronization information is given in 11.1.3. An overview of the MD entity model for a full-duplex point-to-point medium is given in 11.1.4.

11.1.2 Propagation delay measurement

The measurement of propagation delay on a full-duplex point-to-point PTP Link using the peer-to-peer delay mechanism is illustrated in Figure 11-1. The mechanism is the same as the peer-to-peer delay mechanism described in IEEE Std 1588-2019, specialized to a two-step PTP Port[14] and sending the requestReceiptTimestamp and the responseOriginTimestamp separately [see item c) 8) of 11.4.2 in IEEE Std 1588-2019]. The measurement is made by each port at the end of every full-duplex point-to-point PTP Link. Thus, both ports sharing a PTP Link will independently make the measurement, and both ports will know the propagation delay as a result. This allows the time-synchronization information described in 11.1.3 to be transported irrespective of the direction taken by a Sync message. The propagation delay measurement is made on ports otherwise blocked by non-PTP algorithms (e.g., Rapid Spanning Tree Protocol) used to eliminate cyclic topologies. This enables either no loss of synchronization or faster resynchronization, after a reconfiguration, because propagation delays are already known and do not have to be initially measured when the reconfiguration occurs.

Since the propagation delay measurement is made using timestamps relative to the LocalClock entities at each port at the ends of the PTP Link and the resulting mean delay is expressed in the responder timebase (see 11.2.19.3.4), there is no need to measure the mean delay for the PTP Link in each domain because the mean delay is the same in each domain. In addition, the quantity neighborRateRatio (see 10.2.5.7) is the ratio of the responder to requester LocalClock frequency and is also the same in all domains. Therefore, the propagation delay and neighborRateRatio measurements are domain-independent. Single instances of the respective state machines that cause these measurements to be made are invoked, rather than one instance per domain, and the results are available to all domains. The PTP messages used for the measurements (i.e., Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up; see 11.4.5 through 11.4.7) carry 0 in the domainNumber field, but this value is not used.

In Figure 11-1, the propagation delay measurement is initiated by a time-aware system at one end of a PTP Link; this time-aware system is known as the *peer delay initiator*. For purposes of the measurement, the other time-aware system is the *peer delay responder.* A similar measurement occurs in the opposite direction, with the initiator and responder interchanged and the directions of the messages in Figure 11-1 reversed.

[14]See 3.1.86 of IEEE Std 1588-2019 for the definition of a *two-step PTP Port*.

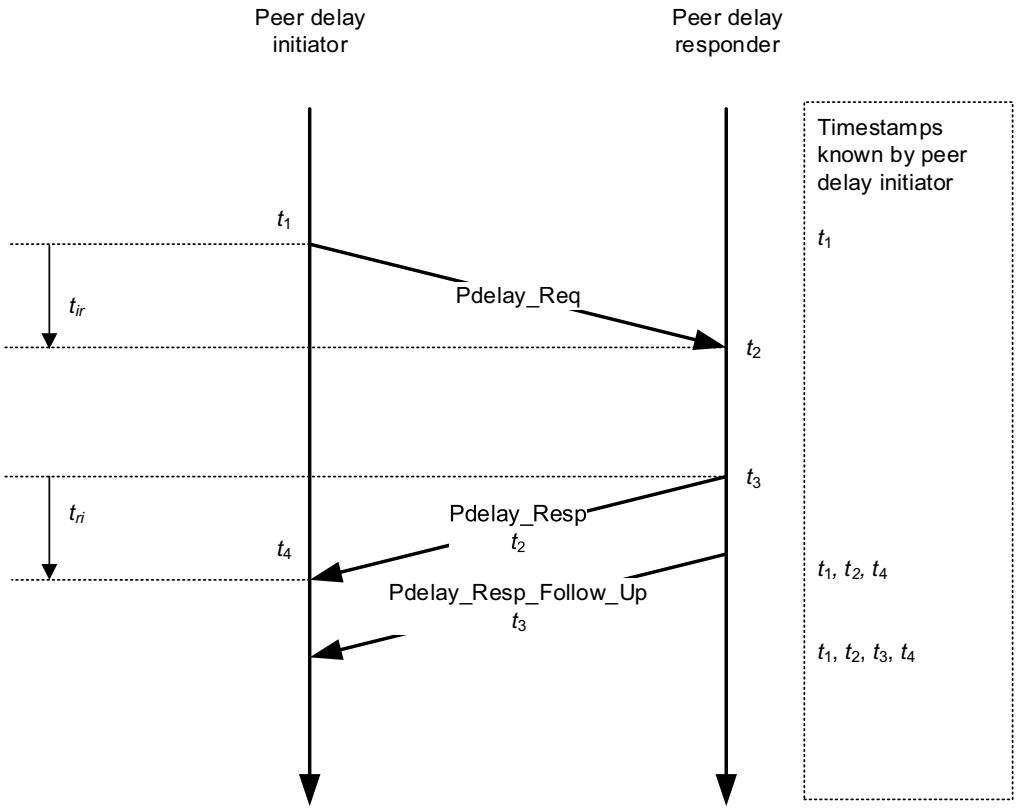

Figure 11-1—Propagation delay measurement using peer-to-peer delay mechanism

The propagation delay measurement starts with the initiator issuing a Pdelay_Req message and generating a timestamp, t_1. The responder receives this message and timestamps it with time t_2. The responder returns a Pdelay_Resp message and timestamps it with time t_3. The responder returns the time t_2 in the Pdelay_Resp message and the time t_3 in a Pdelay_Resp_Follow_Up message. The initiator generates a timestamp, t_4, upon receiving the Pdelay_Resp message. The initiator then uses these four timestamps to compute the mean propagation delay (i.e., meanLinkDelay; see 8.3) as shown in Equation (11-1).

$$\begin{aligned} t_{ir} &= t_2 - t_1 \\ t_{ri} &= t_4 - t_3 \\ D &= \frac{t_{ir} + t_{ri}}{2} = \frac{(t_4 - t_1) - (t_3 - t_2)}{2} \end{aligned} \tag{11-1}$$

where D is the measured mean propagation delay and the other quantities are defined in Figure 11-1.

Note that it is the mean propagation delay that is measured here. Any PTP Link asymmetry is modeled as described in 8.3. Any asymmetry that is not corrected for introduces an error in the transported synchronized time value.

The accuracy of the mean propagation delay measurement depends on how accurately the times t_1, t_2, t_3, and t_4 are measured. In addition, Equation (11-1) assumes that the initiator and responder timestamps are taken relative to clocks that have the same frequency. In practice, t_1 and t_4 are measured relative to the LocalClock entity of the initiator time-aware system, and t_2 and t_3 are measured relative to the LocalClock entity of the responder time-aware system. If the propagation delay measurement is desired relative to the responder time

base, the term $(t_4 - t_1)$ in Equation (11-1) must be multiplied by the rate ratio of the responder relative to the initiator, otherwise there will be an error equal to $0.5y(t_4 - t_1)$, where y is the frequency offset of the responder relative to the initiator. Likewise, if the propagation delay measurement is desired relative to the initiator time base, the term $(t_3 - t_2)$ in Equation (11-1) must be multiplied by the rate ratio of the initiator relative to the responder, otherwise there will be an error equal to $0.5y(t_3 - t_2)$, where y is the frequency offset of the initiator relative to the responder. Finally, if the propagation delay measurement is desired relative to the Grandmaster Clock time base, each term must be multiplied by the rate ratio of the Grandmaster Clock relative to the time base in which the term is expressed.

There can also be an error in measured propagation delay due to time measurement granularity (see B.1.2). For example, if the time measurement granularity is 40 ns (as specified in B.1.2), the timestamps t_1, t_2, t_3, and/or t_4 can undergo 40 ns step changes. When this occurs, the measured propagation delay, D, will change by 20 ns (or by a multiple of 20 ns if more than one of the timestamps has undergone a 40 ns step change). The actual propagation delay has not changed by 20 ns; the effect is due to time measurement granularity. The effect can be reduced, and the accuracy improved, by averaging successive measured propagation delay values. For example, an exponential averaging filter can be used, i.e., as shown in Equation (11-2).

$$D_{avg,k} = aD_{avg,k-1} + (1 - a)D_{k-1} \tag{11-2}$$

where

D_k is the k^{th} propagation delay measurement
$D_{avg,k}$ is the k^{th} computed average propagation delay
k is an index for the propagation delay measurements (i.e., peer delay message exchange)

The quantity a is the exponential weighting factor; it can be set so that the weight of a past propagation delay measurement is $1/e$ after M measurements, i.e., as shown in Equation (11-3).

$$a = e^{-\frac{1}{M}} \tag{11-3}$$

The above averager must be initialized. One method is to use a simple average (i.e., the sum of the sample values divided by the number of samples) of the measurements made up to the current measurement until a window of M measurements has been accumulated. In this case, Equation (11-2) is used only for $k > M$. For $k \leq M$, the averaged propagation delay is given by Equation (11-4).

$$D_{avg,k} = \frac{(k-1)D_{avg,k-1} + D_{k-1}}{k} \tag{11-4}$$

The rate ratio of the responder relative to the initiator is the quantity neighborRateRatio (see 10.2.5.7). It is computed by the function computePdelayRateRatio() (see 11.2.19.3.3) of the MDPdelayReq state machine (see 11.2.19) using successive values of t_3 and t_4. As indicated in the description of computePdelayRateRatio(), any scheme that uses this information is acceptable as long as the performance requirements of B.2.4 are met. One example scheme is given in NOTE 1 of 11.2.19.3.3.

11.1.3 Transport of time-synchronization information

The transport of time-synchronization information by a PTP Instance, using Sync and Follow_Up (or just Sync) messages, is illustrated in Figure 11-2. The mechanism is mathematically equivalent to the mechanism described in IEEE Std 1588-2019 for a peer-to-peer Transparent Clock that is syntonized (see 10.1, 10.3, 11.1, and 11.4 of IEEE Std 1588-2019). However, the processes of transporting synchronization by a peer-to-peer Transparent Clock that is syntonized and by a Boundary Clock are mathematically and functionally equivalent. The main functional difference between the two types of clocks

is that the Boundary Clock participates in best master selection and invokes the BMCA, while the peer-to-peer Transparent Clock does not participate in best master selection and does not invoke the BMCA (and implementations of the two types of clocks can be different).

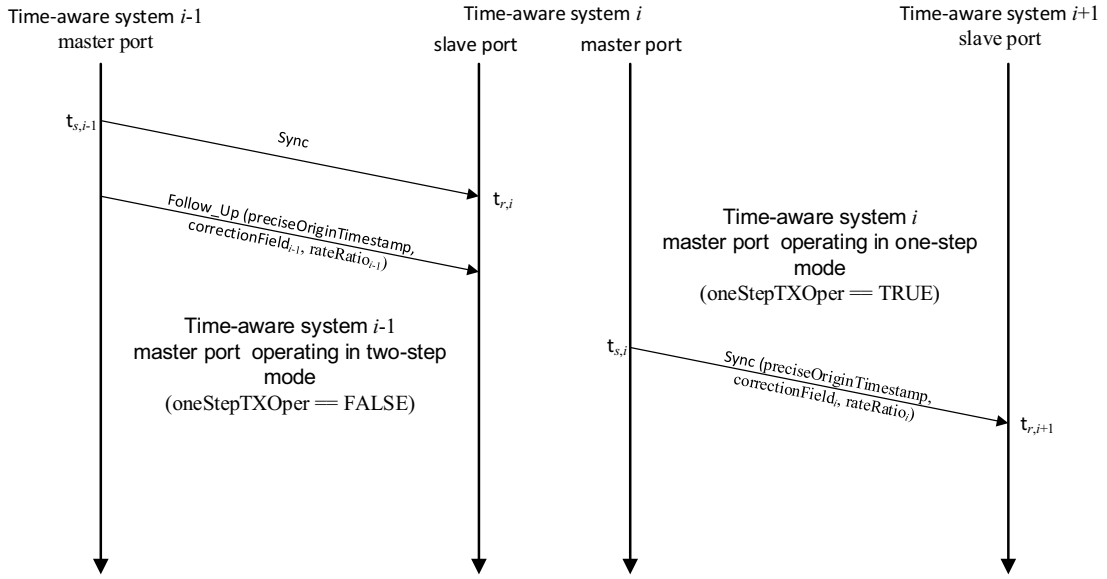

Figure 11-2—Transport of time-synchronization information

Figure 11-2 shows three adjacent PTP Instances, indexed i–1, i, and i+1. Synchronization is transported from PTP Instance i–1 to PTP Instance i using two-step time transport, and then to PTP Instance i+1 using one-step time transport. PTP Instance i–1 sends a Sync message to PTP Instance i at time $t_{s,i-1}$, relative to the LocalClock entity of PTP Instance i–1. At a later time (since it is using two-step time transport), PTP Instance i–1 sends an associated Follow_Up message to PTP Instance i, which contains a preciseOriginTimestamp, correctionField$_{i-1}$, and rateRatio$_{i-1}$. The preciseOriginTimestamp contains the time of the Grandmaster Clock when it originally sent this synchronization information. It is not indexed here because it normally does not change as the Sync and Follow_Up messages traverse the network. The quantity correctionField$_{i-1}$ contains the difference between the synchronized time when the Sync message is sent (i.e., the synchronized time that corresponds to the local time $t_{s,i-1}$) and the preciseOriginTimestamp. The sum of preciseOriginTimestamp and correctionField$_{i-1}$ gives the synchronized time that corresponds to $t_{s,i-1}$. The quantity rateRatio$_{i-1}$ is the ratio of the Grandmaster Clock frequency to the frequency of the LocalClock entity of PTP Instance i–1.

PTP Instance i receives the Sync message from PTP Instance i–1 at time $t_{r,i}$, relative to its LocalClock entity. It timestamps the receipt of the Sync message, and the timestamp value is $t_{r,i}$. It receives the associated Follow_Up message some time later.

PTP Instance i will eventually send a new Sync message at time $t_{s,i}$, relative to its LocalClock entity. The time $t_{s,i}$ occurs after the Follow_Up message is received from PTP Instance i–1. It will have to compute correctionField$_i$, i.e., the difference between the synchronized time that corresponds to $t_{s,i}$ and the preciseOriginTimestamp. To do this calculation, it must compute the value of the time interval between $t_{s,i-1}$ and $t_{s,i}$, expressed in the Grandmaster Clock time base. This interval is equal to the sum of the following quantities:

a) The propagation delay on the PTP Link between PTP Instances i–1 and i, expressed in the Grandmaster Clock time base, and

b) The difference between $t_{s,i}$ and $t_{r,i}$ (i.e., the residence time), expressed in the Grandmaster Clock time base.

The mean propagation delay on the PTP Link between PTP Instances i–1 and i, relative to the LocalClock entity of PTP Instance i–1, is equal to meanLinkDelay (see 10.2.5.8). This must be multiplied by rateRatio_{i-1} to express it in the Grandmaster Clock time base. The total propagation delay is equal to the mean propagation delay plus the quantity delayAsymmetry (see 8.3 and 10.2.5.9); delayAsymmetry is already expressed in the Grandmaster Clock time base. The residence time, $t_{s,i}–t_{r,i}$, must be multiplied by rateRatio_i to express it in the Grandmaster Clock time base.

The preceding computation is organized slightly differently in the state machines of 11.2.14 and 11.2.15. Rather than explicitly expressing the PTP Link propagation delay in the Grandmaster Clock time base, the local time at PTP Instance i that corresponds to $t_{s,i-1}$ is computed; this is the upstreamTxTime member of the MDSyncReceive structure (see 10.2.2.2.7; recall that $t_{s,i-1}$ is relative to the LocalClock entity of PTP Instance i–1). upstreamTxTime is equal to the quantity $t_{r,i}$ minus the PTP Link propagation delay expressed relative to the LocalClock entity of PTP Instance i. The PTP Link propagation delay expressed relative to the LocalClock entity of PTP Instance i is equal to the sum of the following:

c) The quantity meanLinkDelay (see 10.2.5.8) divided by neighborRateRatio (see 10.2.5.7), and

d) The quantity delayAsymmetry (see 10.2.5.9) divided by rateRatio_i.

The division of delayAsymmetry by rateRatio_i is performed after rateRatio_i has been updated. The computation of upstreamTxTime is done by the MDSyncReceiveSM state machine in the function setMDSyncReceiveMDSR() (see 11.2.14.2.1). When PTP Instance i sends a Sync message to PTP Instance i+1, it computes the sum of the PTP Link propagation delay and residence time, expressed in the Grandmaster Clock time base, as shown in item e).

e) The quantity $(t_{s,i} – \text{upstreamTxTime})(\text{rateRatio}_i)$.

As in item d) in this subclause, this computation is performed after rateRatio_i has been updated. The quantity of item e) is added to $\text{correctionField}_{i-1}$ to obtain correctionField_i. The computation of item e) and correctionField_i is done by the MDSyncSendSM state machine in the function setFollowUp() (see 11.2.15.2.3). The quantity correctionField_i is inserted in the Sync message sent by PTP Instance i.

The difference between mean propagation delay relative to the Grandmaster Clock time base and relative to the time bases of the PTP Instance at the other end of the attached PTP Link or of the current PTP Instance is usually negligible. The former can be obtained from the latter by multiplying the latter by the ratio of the Grandmaster Clock frequency to the frequency of the LocalClock entity of the PTP Instance at the other end of the PTP Link attached to this PTP Port. This ratio differs from 1 by 200 ppm or less. For example, for a worst-case frequency offset of the LocalClock entity of the PTP Instance at the other end of the PTP Link, relative to the Grandmaster Clock, of 200 ppm, and a measured propagation time of 100 ns, the difference in D relative to the two time bases is 20 ps. The corresponding difference for PTP Link delay asymmetry in this example is also negligible because the magnitude of the PTP Link delay asymmetry is of the same order of magnitude as the mean propagation time, or less. However, the difference is usually not negligible for residence time because residence time can be much larger (see B.2.2).

It was indicated in the first paragraph of this subclause that the processes of transporting synchronization by a peer-to-peer Transparent Clock that is syntonized and by a Boundary Clock are mathematically and functionally equivalent. The reason for this equivalency is that the computations described so far in this subclause compute the synchronized time when the Sync message is sent by the PTP Instance. The same computations are done if PTP Instance *i* sends a Sync message without having received a new Sync message, i.e., if Sync receipt timeout occurs (see 10.7.3.1). In this case, PTP Instance *i* uses the most recently received time-synchronization information from PTP Instance *i*–1, which would be prior to when PTP Instance *i* sent its most recent Sync message. The synchronized time corresponding to the sending of a Sync message is equal to the sum of the preciseOriginTimestamp and correctionField. Normally a Boundary Clock places this entire value, except for any sub-nanosecond portion, in the preciseOriginTimestamp, while a Transparent Clock retains the preciseOriginTimestamp and updates the correctionField. However, the sum of the two fields is equal to the synchronized time when the Sync message is sent in both cases.

The ratio of the Grandmaster Clock frequency to the frequency of the LocalClock entity at PTP Instance *i*, rateRatio_i, is equal to the same quantity at PTP Instance *i*–1, rateRatio_{i-1}, multiplied by the ratio of the frequency of the LocalClock entity at PTP Instance *i*–1 to the frequency of the LocalClock entity at PTP Instance *i*, neighborRateRatio (see 10.2.5.7). If neighborRateRatio is sufficiently small, this is approximately equal to the sum of rateRatio_{i-1} and the quantity neighborRateRatio–1, which is the frequency offset of PTP Instance *i*–1 relative to PTP Instance *i*. This computation is done by the PortSyncSyncReceive state machine (see 10.2.8).

NOTE—The sending of time-synchronization information by the master ports of a PTP Instance might or might not be tightly synchronized with the receipt of time-synchronization information by the slave port. If a master port has the same logMessageInterval as the slave port, it will transmit timing event messages as soon as possible after the slave port has received the corresponding timing event messages and the master port is operating in “syncLocked” mode (see 10.2.5.15). If a master port and slave port have different logMessageInterval values, then the master port can send timing event messages without any synchronization with the slave port.

11.1.4 Model of operation

A PTP Instance contains one MD entity per PTP Instance, per PTP Port. This entity contains functions generic to all media, which are described in Clause 10, and functions specific to the respective medium for the PTP Link. Functions specific to full-duplex point-to-point links are described in the current clause.

NOTE—Full-duplex point-to-point IEEE 802.3 links are in the category of links specified in this clause.

The model for a PTP Instance of a time-aware system with full-duplex point-to-point links is shown in Figure 11-3. It assumes the presence of one full-duplex point-to-point MD entity per PTP Port. The media-independent entities shown in Figure 11-3 are described in 10.1.2.

A general, media-independent description of the generation of timestamps is given in 8.4.3. A more specific description for PTP event messages is given in 11.3.2.1. A PTP event message is timestamped relative to the LocalClock entity when the message timestamp point (see 3.17) crosses the timestamp measurement plane (see 3.33). The timestamp is corrected for any ingressLatency or egressLatency (see 8.4.3) to produce a timestamp relative to the reference plane (see 3.26). The corrected timestamp value is provided to the MD entity.

The MD entity behavior and detailed state machines specific to full-duplex point-to-point links are described in 11.2. The behavior of the MD entity that is generic to all media is described in Clause 10.

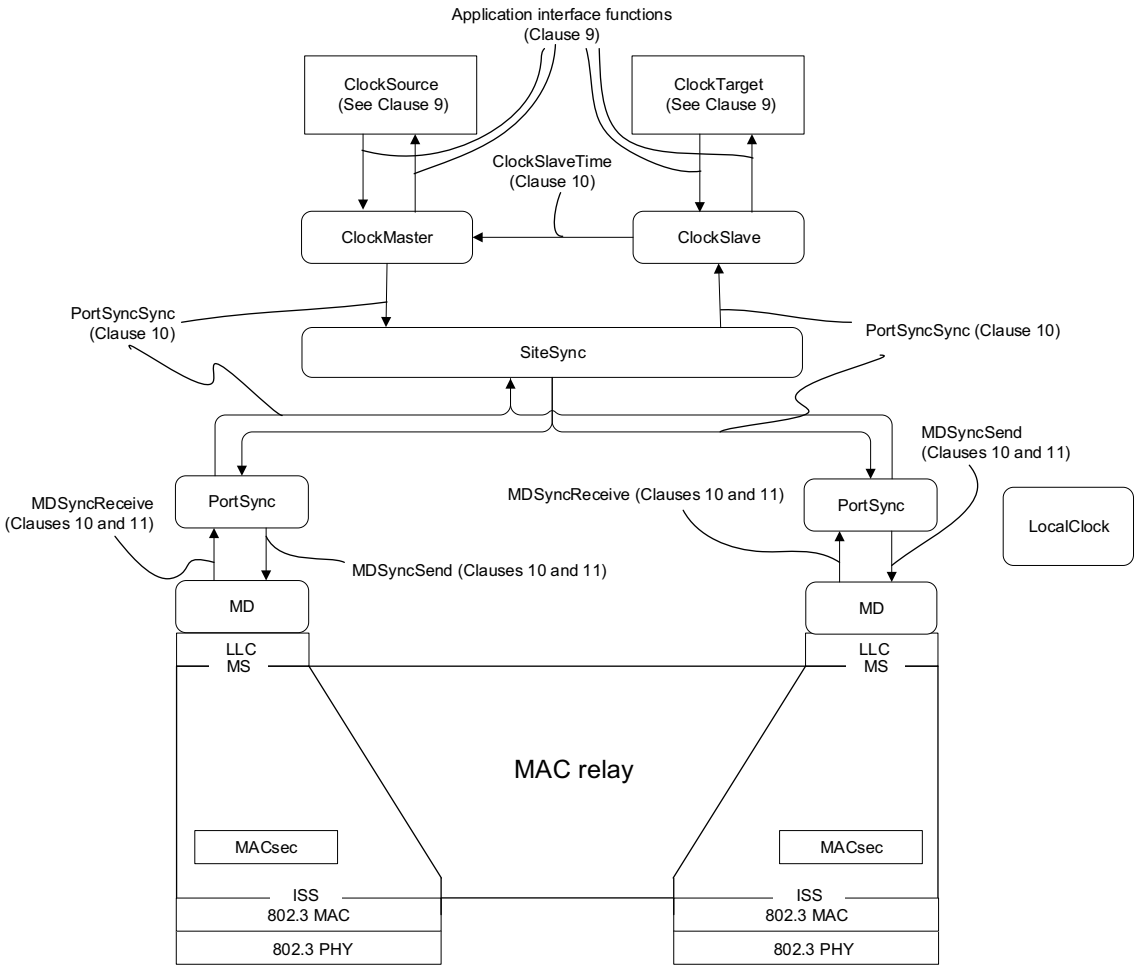

Figure 11-3—Model for a PTP Instance of a time-aware system with full-duplex point-to-point links

11.2 State machines for MD entity specific to full-duplex point-to-point links

11.2.1 General

This subclause describes the media-dependent state machines for an MD entity, for full-duplex point-to-point links. The state machines are all per port because an instance of each is associated with an MD entity. The state machines are as follows:

a) MDSyncReceiveSM (shown in Figure 10-2, and in more detail in Figure 11-4): receives Sync and, if the received information is two-step, Follow_Up messages, and sends the time-synchronization information carried in these messages to the PortSync entity of the same PTP Port. There is one instance of this state machine per PTP Port, per domain.

b) MDSyncSendSM (shown in Figure 10-2, and in more detail in Figure 11-4): receives an MDSyncSend structure from the PortSync entity of the same PTP Port; transmits a Sync message; uses the syncEventEgressTimestamp, corrected for egressLatency, and information contained in the MDSyncSend structure to compute information needed for the Sync message if the PTP Port is currently operating as a one-step PTP Port and for the corresponding Follow_Up message if the PTP Port is currently operating as a two-step PTP Port; and transmits the Follow_Up message if the PTP Port is two-step. There is one instance of this state machine per PTP Port, per domain.

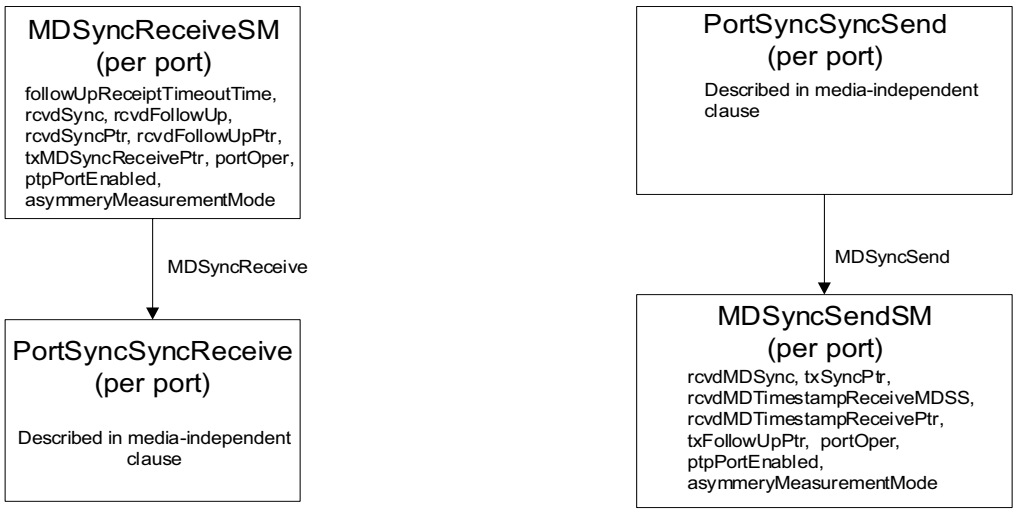

Figure 11-4—Detail of MD entity time-synchronization state machines for full-duplex point-to-point links

c) MDPdelayReq (shown in Figure 11-5): transmits a Pdelay_Req message, receives Pdelay_Resp and Pdelay_Resp_Follow_Up messages corresponding to the transmitted Pdelay_Req message, uses the information contained in successive Pdelay_Resp and Pdelay_Resp_Follow_Up messages to compute the ratio of the frequency of the LocalClock entity in the PTP Instance at the other end of the attached PTP Link to the frequency of the LocalClock entity in this PTP Instance, and uses the information obtained from the message exchange and the computed frequency ratio to compute propagation delay on the attached PTP Link. There is one instance of this state machine for all the domains, per port.

d) MDPdelayResp (shown in Figure 11-5): receives a Pdelay_Req message from the MD entity at the other end of the attached PTP Link, and responds with Pdelay_Resp and Pdelay_Resp_Follow_Up messages. There is one instance of this state machine for all the domains, per port.

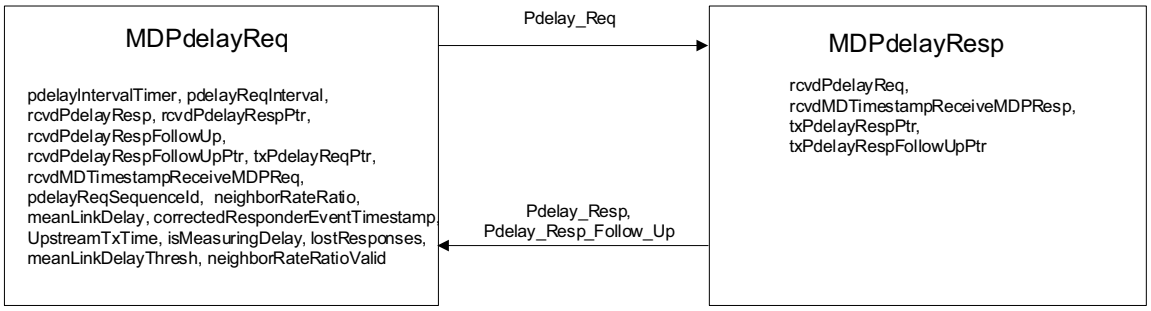

Figure 11-5—Peer-to-peer delay mechanism state machines—overview and interrelationships

e) SyncIntervalSetting state machine (not shown): receives a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3), and sets the global variables that give the duration of the mean intervals between successive Sync messages. There is one instance of this state machine per PTP Port, per domain.

f) LinkDelayIntervalSetting state machine (not shown): receives a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3), and sets the global variables that give the duration of the mean intervals between successive Pdelay_Req messages. There is one instance of this state machine for all the domains, per port.

Figure 10-2, Figure 11-4, and Figure 11-5 are not themselves state machines, but illustrate the machines, their interrelationships, the principle variables and messages used to communicate between them, their local variables, and performance parameters. The figures do not show the service interface primitives between the media-dependent layer and the logical link control (LLC). Figure 11-5 is analogous to Figure 10-2; while Figure 10-2 applies to the general time-synchronization protocol, Figure 11-5 is limited to the peer-to-peer delay mechanism for measurement of propagation delay in full-duplex point-to-point links. Figure 11-4 shows greater detail of the MDSyncReceiveSM and MDSyncSendSM state machines for full-duplex point-to-point links than Figure 10-2.

The state machines described in 11.2 use some of the global per PTP Instance system variables specified in 10.2.4, the global per-port variables specified in 10.2.5, and the functions specified in 10.2.6.

11.2.2 Determination of asCapable and asCapableAcrossDomains

There is one instance of the global variable asCapable (see 10.2.5.1) per PTP Port, per domain. There is one instance of the global variable asCapableAcrossDomains (see 11.2.13.12), per port, that is common across, and accessible by, all the domains.

The per-PTP Port global variable asCapable (see 10.2.5.1) indicates whether the IEEE 802.1AS protocol is operating, in this domain, on the PTP Link attached to this PTP Port, and can provide the time-synchronization performance described in B.3. asCapable is used by the PortSync entity, which is media-independent; however, the determination of asCapable is media-dependent.

The per-port global variable asCapableAcrossDomains is set by the MDPdelayReq state machine (see 11.2.19 and Figure 11-9). For a port attached to a full-duplex point-to-point PTP Link, asCapableAcrossDomains shall be set to TRUE if and only if it is determined, via the peer-to-peer delay mechanism, that the following conditions hold for the port:

a) The port is exchanging peer delay messages with its neighbor,

b) The measured delay does not exceed meanLinkDelayThresh,

c) The port does not receive multiple Pdelay_Resp or Pdelay_Resp_Follow_Up messages in response to a single Pdelay_Req message, and

d) The port does not receive a response from itself or another PTP Port of the same PTP Instance.

NOTE 1—If a PTP Instance implements only domain 0 and the MDPdelayReq and MDPdelayResp state machines are invoked on domain 0 (see 11.2.19), asCapableAcrossDomains is still set by the MDPdelayReq state machine.

The default value of meanLinkDelayThresh shall be set as specified in Table 11-1.

Table 11-1—Value of meanLinkDelayThresh for various links

Link	Value of meanLinkDelayThresh (ns) (see NOTE)
100BASE-TX, 1000BASE-T	800_{10}
100BASE-FX, 1000BASE-X	FFFF FFFF FFFF FFFF FFFF FFFF$_{16}$
NOTE—The actual propagation delay for 100BASE-TX and 1000BASE-T links is expected to be smaller than the above respective threshold. If the measured mean propagation delay (i.e., meanLinkDelay; see 10.2.5.8) exceeds this threshold, it is assumed that this is due to the presence of equipment that does not implement gPTP. For 100BASE-FX and 1000BASE-X links, the actual propagation delay can be on the order of, or larger than, the delay produced by equipment that does not implement gPTP; therefore, such equipment cannot be detected by comparing measured propagation delay with a threshold. In this case, meanLinkDelayThresh is set to the largest possible value (i.e., all 1s).	

The per-PTP Port, per-domain global variable asCapable shall be set to TRUE if and only if the following conditions hold:

e) The value of asCapableAcrossDomains is TRUE, and
f) One of the following conditions holds:
    1) The value of neighborGptpCapable for this PTP Port is TRUE, or
    2) The value of domainNumber is zero, and the value of sdoId for peer delay messages received on this PTP Port is 0x100.

NOTE 2—Condition f) 2) ensures backward compatibility with the 2011 edition of this standard. A PTP Instance compliant with the current edition of this standard that is attached, via a full-duplex point-to-point PTP Link, to a PTP Instance compliant with the 2011 edition of this standard will not receive Signaling messages that contain the gPTP capable TLV and will not set neighborGptpCapable to TRUE. However, condition f) 2) ensures that asCapable for this PTP Port and domain (i.e., domain 0) will still be set to TRUE if condition e) holds because the peer delay messages received from the time-aware system compliant with the 2011 edition of this standard will have sdoId set to 0x100.

11.2.3 Use of MAC Control PAUSE operation

A PTP Instance shall not use the MAC Control PAUSE operation.

11.2.4 Use of priority-based flow control

A PTP Instance that implements priority-based flow control shall neither transmit nor honor upon receipt priority-based flow control messages that act on the IEEE 802.1AS message priority code point (see 8.4.4).

11.2.5 Use of link aggregation

gPTP PDUs, when used with physical ports that are attached to an IEEE 802.1AX™ Link Aggregation Group, shall be transmitted and received over the physical ports (Aggregation Ports), not the Aggregator Port. Using the Parser/Multiplexer functions described in 6.1.3 and 6.2.7 of IEEE Std 802.1AX-2014, received gPTP PDUs are recognized as control frames, separated from the incoming data stream, and directed to the gPTP layers. gPTP PDUs output from the gPTP layers are integrated into the port's data stream by the Parser/Multiplexer.

Assuming that ptpPortEnabled is true for all the physical links (Aggregation Links) in a Link Aggregation Group and that all the physical links are connected to the same two systems, gPTP will measure the delay on each physical link, and the IEEE 802.1AS protocol will choose one of the physical links for transmitting time from the master clock.

11.2.6 Service interface primitives and data structures communicated between state machines

The following subclauses describe the service primitives and data structures communicated between the time-synchronization state machines of the MD entity. First the service primitives are described, followed by the data structures.

11.2.7 DL-UNITDATA.request

This service primitive is used by an MD entity to request to the associated LLC the transmission of a Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, or Pdelay_Resp_Follow_Up message. The primitive is described in 2.2.1.1.1 of ISO/IEC 8802-2:1998 [B16].

11.2.8 DL-UNITDATA.indication

This service primitive is used by the LLC to indicate to the associated MD entity the receipt of a Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, or Pdelay_Resp_Follow_Up message. The primitive is described in 2.2.1.1.1 of ISO/IEC 8802-2:1998 [B16].

11.2.9 MDTimestampReceive

11.2.9.1 General

This structure provides the timestamp, relative to the timestamp measurement plane, of the event message that was just sent or just received. The structure is received by the MD entity of the PTP Port. The MD entity corrects this timestamp for any ingressLatency or egressLatency (see 8.4.3) before using it and/or passing it to higher layer entities.

```
MDTimestampReceive{
        timestamp
}
```

The member of the structure is defined in the following subclause.

11.2.9.2 timestamp (UScaledNs)

The timestamp is the value of the timestamp of the event message (i.e., Sync, Pdelay_Req, Pdelay_Resp) that was just transmitted or received. This timestamp is taken relative to the timestamp measurement plane.

11.2.10 MDSyncReceive

This structure is specified in 10.2.2.2.

11.2.11 MDSyncSend

This structure is specified in 10.2.2.1.

11.2.12 Overview of MD entity global variables

Subclause 11.2.13 defines global variables used by the MD entity state machines whose scopes are as follows:

— Per PTP Instance (i.e., per domain)
— Per PTP Instance, per PTP Port

— Instances used by CMLDS (see 11.2.17) (i.e., variable is common across all LinkPorts)
— Instances used by CMLDS, per LinkPort

Table 11-2 summarizes the scope of each global variable of 11.2.13.

Table 11-2 — Summary of scope of global variables used by time synchronization state machines (see 10.2.4 and 10.2.5)

Variable name	Subclause of definition	Per PTP Instance (i.e., per domain)	Per PTP Instance, per PTP Port	Instance used by CMLDS (i.e., variable is common across all LinkPorts)	Instance used by CMLDS, per LinkPort
currentLogPdelayReqInterval	11.2.13.1	No	Yes[a]	No	Yes
initialLogPdelayReqInterval	11.2.13.2	No	Yes[a]	No	Yes
pdelayReqInterval	11.2.13.3	No	Yes[a]	No	Yes
allowedLostResponses	11.2.13.4	No	Yes[a]	No	Yes
allowedFaults	11.2.13.5	No	Yes[a]	No	Yes
isMeasuringDelay	11.2.13.6	No	Yes[a]	No	Yes
meanLinkDelayThresh	11.2.13.7	No	Yes[a]	No	Yes
syncSequenceId	11.2.13.8	No	Yes	No	No
oneStepReceive	11.2.13.9	No	Yes	No	No
oneStepTransmit	11.2.13.10	No	Yes	No	No
oneStepTxOper	11.2.13.11	No	Yes	No	No
asCapableAcrossDomains	11.2.13.12	No	No	No	Yes

[a] The instance of this variable that is per PTP Instance, per PTP Port exists only for domain 0.

11.2.13 MD entity global variables

11.2.13.1 currentLogPdelayReqInterval: The current value of the logarithm to base 2 of the mean time interval, in seconds, between the sending of successive Pdelay_Req messages (see 11.5.2.2). This value is set in the LinkDelayIntervalSetting state machine (see 11.2.21). The data type for currentLogPdelayReqInterval is Integer8. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.13.2 initialLogPdelayReqInterval: The initial value of the logarithm to base 2 of the mean time interval, in seconds, between the sending of successive Pdelay_Req messages (see 11.5.2.2). The data type for initialLogPdelayReqInterval is Integer8. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.13.3 pdelayReqInterval: A variable containing the mean Pdelay_Req message transmission interval for the port corresponding to this MD entity. The value is set in the LinkDelayIntervalSetting state machine (see 11.2.21). The data type for pdelayReqInterval is UScaledNs. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.13.4 allowedLostResponses: The number of Pdelay_Req messages without valid responses above which a port is considered to be not exchanging peer delay messages with its neighbor. The data type for allowedLostResponses is UInteger8. The default value and range of allowedLostResponses is given in 11.5.3. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.13.5 allowedFaults: The number of faults [i.e., instances where meanLinkDelay (see 10.2.5.8) exceeds meanLinkDelayThresh (see 11.2.13.7) and/or the computation of neighborRateRatio is invalid (see 11.2.19.2.10)] above which asCapableAcrossDomains is set to FALSE, i.e., the port is considered not capable of interoperating with its neighbor via the IEEE 802.1AS protocol (see 10.2.5.1). The data type for allowedFaults is UInteger8. The default value and range of allowedFaults is given in 11.5.4. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.13.6 isMeasuringDelay: A Boolean that is TRUE if the port is measuring PTP Link propagation delay. For a full-duplex point-to-point PTP Link, the port is measuring PTP Link propagation delay if it is receiving Pdelay_Resp and Pdelay_Resp_Follow_Up messages from the port at the other end of the PTP Link (i.e., it performs the measurement using the peer-to-peer delay mechanism). There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.13.7 meanLinkDelayThresh: The propagation time threshold above which a port is considered not capable of participating in the IEEE 802.1AS protocol. If meanLinkDelay (see 10.2.5.8) exceeds meanLinkDelayThresh, then asCapableAcrossDomains (see 11.2.13.12) is set to FALSE. The data type for meanLinkDelayThresh is UScaledNs. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

NOTE—The variable meanLinkDelayThresh was named neighborPropDelayThresh in the 2011 edition of this standard.

11.2.13.8 syncSequenceId: The sequenceId (see 11.4.2.8) for the next Sync message to be sent by this MD entity. The data type for syncSequenceId is UInteger16.

11.2.13.9 oneStepReceive: A Boolean variable that is TRUE if the PTP Port is capable of receiving one-step Sync messages.

11.2.13.10 oneStepTransmit: A Boolean variable that is TRUE if the PTP Port is capable of transmitting one-step Sync messages.

11.2.13.11 oneStepTxOper: A Boolean variable that is TRUE if the PTP Port will be transmitting one-step Sync messages (see 11.1.3).

NOTE—Since a one-step port modifies Sync message fields (e.g., the correctionField) “on the fly” as the Sync message is being transmitted, the modifying of the correctionField of the Sync message is often implemented in hardware. Each distinct PTP Instance of a time-aware system runs in a distinct domain and tracks a distinct time. If a time-aware system supports more than one PTP Instance, support for one-step often requires hardware for each PTP Instance (e.g., four domains requires four hardware components), but support for two-step can share hardware among all PTP Instances.

11.2.13.12 asCapableAcrossDomains: A Boolean that isTRUE if and only if conditions a) through d) of 11.2.2 are satisfied. This Boolean is set by the MDPdelayReq state machine and is used in determining asCapable for a port (see 11.2.2). There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains. When only one domain is active, asCapableAcrossDomains is equivalent to the variable asCapable (see 10.2.5.1).

11.2.14 MDSyncReceiveSM state machine

11.2.14.1 State machine variables

The following variables are used in the state diagram in Figure 11-6 (in 11.2.14.3):

11.2.14.1.1 followUpReceiptTimeoutTime: A variable used to save the time at which the information conveyed by a received Sync message will be discarded if the associated Follow_Up message is not received by then. The data type for followUpReceiptTimeoutTime is UScaledNs.

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.

11.2.14.1.3 rcvdFollowUp: A Boolean variable that notifies the current state machine when a Follow_Up message is received. This variable is reset by the current state machine.

11.2.14.1.4 rcvdSyncPtr: A pointer to a structure whose members contain the values of the fields of the Sync message whose receipt is indicated by rcvdSync (see 11.2.14.1.2).

11.2.14.1.5 rcvdFollowUpPtr: A pointer to a structure whose members contain the values of the fields of the Follow_Up message whose receipt is indicated by rcvdFollowUp (see 11.2.14.1.3).

11.2.14.1.6 txMDSyncReceivePtrMDSR: A pointer to a structure whose members contain the values of the parameters of an MDSyncReceive structure to be transmitted.

11.2.14.1.7 upstreamSyncInterval: The sync interval (see 10.7.2.1) for the upstream PTP Port that sent the received Sync message. The data type for upstreamSyncInterval is UScaledNs.

11.2.14.2 State machine functions

11.2.14.2.1 setMDSyncReceiveMDSR(): Creates an MDSyncReceive structure, and returns a pointer to this structure. The members of this structure are set as follows:

a) If twoStepFlag of the most recently received Sync message is TRUE, followUpCorrectionField is set equal to the sum of the correctionField (see 11.4.2.6) of the most recently received Sync message, plus the correctionField of the most recently received Follow_Up message. Otherwise, followUpCorrectionField is set equal to the correctionField of the most recently received Sync message.

b) sourcePortIdentity is set equal to the sourcePortIdentity (see 11.4.2.7) of the most recently received Sync message (see 11.4.3).

c) logMessageInterval is set equal to the logMessageInterval (see 11.4.2.9) of the most recently received Sync message (see 11.4.3).

d) If twoStepFlag of the most recently received Sync message is TRUE, preciseOriginTimestamp is set equal to the preciseOriginTimestamp (see 11.4.4.2.1) of the most recently received Follow_Up message. Otherwise, preciseOriginTimestamp is set equal to the originTimestamp (see 11.4.3.2.1) of the most recently received Sync message.

e) rateRatio is set equal to the quantity (cumulativeScaledRateOffset $\times 2^{-41}$) + 1.0, where the cumulativeScaledRateOffset field is for the most recently received Follow_Up information TLV (see 11.4.4.3.6).

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 rateRatio [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)

State machine invoked by CMLDS:

upstreamTxTime = syncEventIngressTimestamp − (meanLinkDelay/neighborRateRatio)

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), 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 quotient is 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.

NOTE 2—The difference between the mean propagation time in the Grandmaster Clock time base, the time base of the PTP Instance at the other end of the PTP Link, and the time base of the current PTP Instance is usually negligible. The same is true of any delayAsymmetry (see NOTE 2 of 11.2.19.3.4).

g) gmTimeBaseIndicator is set equal to the gmTimeBaseIndicator of the most recently received Follow_Up information TLV (see 11.4.4.3.7).
h) lastGmPhaseChange is set equal to the lastGmPhaseChange of the most recently received Follow_Up information TLV (see 11.4.4.3.8).
i) lastGmFreqChange is set equal to the scaledLastGmFreqChange of the most recently received Follow_Up information TLV (see 11.4.4.3.9), multiplied by 2^{-41}.
j) domainNumber is set equal to the domainNumber of the most recently received Sync message (see 11.4.3).

11.2.14.2.2 txMDSyncReceive (txMDSyncReceivePtrMDSR): Transmits an MDSyncReceive structure to the PortSyncSyncReceive state machine of the PortSync entity of this PTP Port.

11.2.14.3 State diagram

The MDSyncReceiveSM state machine shall implement the function specified by the state diagram in Figure 11-6, the local variables specified in 11.2.14.1, the functions specified in 11.2.14.2, the structure specified in 10.2.2.2, the messages specified in 11.4, the MD entity global variables specified in 11.2.13, and the relevant global variables and functions specified in 10.2.4 through 10.2.6. The state machine receives Sync and, if the received information is two-step, Follow_Up messages, places the time-synchronization information in an MDSyncReceive structure, and sends the structure to the PortSyncSyncReceive state machine of the PortSync entity of this PTP Port.

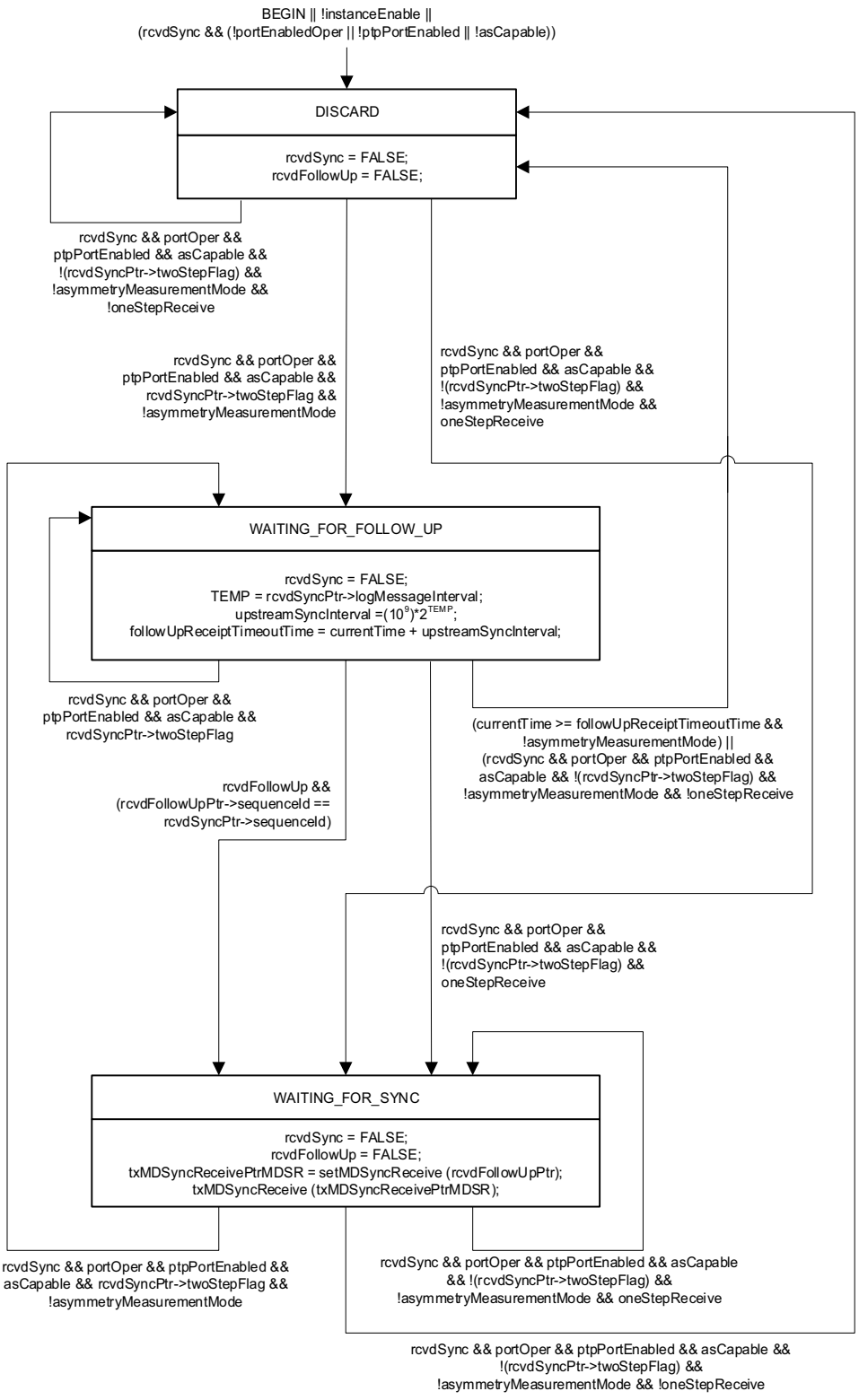

Figure 11-6—MDSyncReceiveSM state machine

11.2.15 MDSyncSendSM state machine

11.2.15.1 State machine variables

The following variables are used in the state diagram in Figure 11-7 (in 11.2.15.3):

11.2.15.1.1 rcvdMDSyncMDSS: A Boolean variable that notifies the current state machine when an MDSyncSend structure is received. This variable is reset by the current state machine.

11.2.15.1.2 txSyncPtr: A pointer to a structure whose members contain the values of the fields of a Sync message to be transmitted.

11.2.15.1.3 rcvdMDTimestampReceiveMDSS: A Boolean variable that notifies the current state machine when the syncEventEgressTimestamp (see 11.3.2.1) for a transmitted Sync message is received. This variable is reset by the current state machine.

11.2.15.1.4 rcvdMDTimestampReceivePtr: A pointer to the received MDTimestampReceive structure (see 11.2.9).

11.2.15.1.5 txFollowUpPtr: A pointer to a structure whose members contain the values of the fields of a Follow_Up message to be transmitted.

11.2.15.2 State machine functions

11.2.15.2.1 setSyncTwoStep(): Creates a structure whose parameters contain the fields (see 11.4) of a Sync message to be transmitted, and returns a pointer to this structure. The parameters are set as follows:

a) correctionField is set equal to 0.
b) sourcePortIdentity is set equal to the sourcePortIdentity member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).
c) sequenceId is set equal to syncSequenceId (see 11.2.13.8).
d) logMessageInterval is set equal to the logMessageInterval member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).
e) The remaining fields are set as specified in 11.4.2 and 11.4.3 when twoStepFlag is TRUE.

11.2.15.2.2 txSync (txSyncPtr): Transmits a Sync message from this MD entity, whose fields contain the parameters in the structure pointed to by txSyncPtr (see 11.2.15.1.2).

11.2.15.2.3 setFollowUp(): Creates a structure whose parameters contain the fields (see 11.4) of a Follow_Up message to be transmitted, and returns a pointer to this structure. The parameters are set as follows:

a) followUpCorrectionField is set equal to the sum of the following:
   1) The followUpCorrectionField member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11)
   2) The quantity

$$\text{rateRatio} \times (\text{syncEventEgressTimestamp} - \text{upstreamTxTime})$$

   where

rateRatio	is the rateRatio member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11)
upstreamTxTime	is the upstreamTxTime member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11
syncEventEgressTimestamp	is the timestamp pointed to by rcvdMDTimestampReceivePtr, corrected for egressLatency (see 8.4.3)

NOTE—If the PTP Instance that contains this PortSync entity is a PTP Relay Instance, the quantity

syncEventEgressTimestamp – upstreamTxTime

is the sum of the residence time and PTP Link propagation delay on the upstream PTP Link, relative to the LocalClock entity, and the quantity

rateRatio × (syncEventEgressTimestamp – upstreamTxTime)

is the sum of the residence time and PTP Link propagation delay on the upstream PTP Link, relative to the Grandmaster Clock (see 11.2.14.2.1 for details on the setting of upstreamTxTime).

b) sourcePortIdentity is set equal to the sourcePortIdentity member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

c) sequenceId is set equal to syncSequenceId (see 11.2.13.8).

d) logMessageInterval is set equal to the logMessageInterval member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

e) preciseOriginTimestamp is set equal to the preciseOriginTimestamp member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

f) rateRatio is set equal to the rateRatio member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

g) gmTimeBaseIndicator is set equal to the gmTimeBaseIndicator member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

h) lastGmPhaseChange is set equal to the lastGmPhaseChange member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

i) lastGmFreqChange is set equal to the scaledLastGmFreqChange member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11), multiplied by 2^{41}.

j) domainNumber is set equal to the domainNumber of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

k) The remaining fields are set as specified in 11.4.2 and 11.4.4.

11.2.15.2.4 txFollowUp (txFollowUpPtr): Transmits a Follow_Up message from this MD entity, whose fields contain the parameters in the structure pointed to by txFollowUpPtr (see 11.2.15.1.5).

11.2.15.2.5 setSyncOneStep(): Creates a structure whose parameters contain the fields (see 11.4) of a Sync message to be transmitted, and returns a pointer to this structure. The parameters are set as follows:

a) correctionField is set equal to the followUpCorrectionField member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

b) sourcePortIdentity is set equal to the sourcePortIdentity member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

c) sequenceId is set equal to syncSequenceId (see 11.2.13.8).

d) logMessageInterval is set equal to the logMessageInterval member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

e) originTimestamp is set equal to the preciseOriginTimestamp member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

f) rateRatio is set equal to the rateRatio member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

g) gmTimeBaseIndicator is set equal to the gmTimeBaseIndicator member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

h) lastGmPhaseChange is set equal to the lastGmPhaseChange member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

i) lastGmFreqChange is set equal to the scaledLastGmFreqChange member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11), multiplied by 2^{41}.

j) domainNumber is set equal to the domainNumber of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11).

k) The remaining fields are set as specified in 11.4.2 and 11.4.3 when twoStepFlag is FALSE.

11.2.15.2.6 modifySync(): Adds to the correctionField of the transmitted Sync message, after the syncEventEgressTimestamp is taken and known, the quantity

rateRatio × (syncEventEgressTimestamp − upstreamTxTime)

where

rateRatio	is the rateRatio member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11)
upstreamTxTime	is the upstreamTxTime member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11)
syncEventEgressTimestamp	is the timestamp pointed to by rcvdMDTimestampReceivePtr, corrected for egressLatency (see 8.4.3)

NOTE—If the PTP Instance that contains this PortSync entity is a PTP Relay Instance, the quantity

syncEventEgressTimestamp − upstreamTxTime

is the sum of the residence time and PTP Link propagation delay on the upstream PTP Link, relative to the LocalClock entity, and the quantity

rateRatio × (syncEventEgressTimestamp − upstreamTxTime)

is the sum of the residence time and PTP Link propagation delay on the upstream PTP Link, relative to the Grandmaster Clock (see 11.2.14.2.1 for details on the setting of upstreamTxTime).

11.2.15.3 State diagram

The MDSyncSendSM state machine shall implement the function specified by the state diagram in Figure 11-7; the local variables specified in 11.2.15.1; the functions specified in 11.2.15.2; the structures specified in 11.2.9 and 10.2.2.1; the messages specified in 11.4; the MD entity global variables specified in 11.2.13; and the relevant global variables and functions specified in 10.2.4 through 10.2.6. The state machine receives an MDSyncSend structure from the PortSyncSyncSend state machine of the PortSync entity of this PTP Port and transmits a Sync and corresponding Follow_Up message.

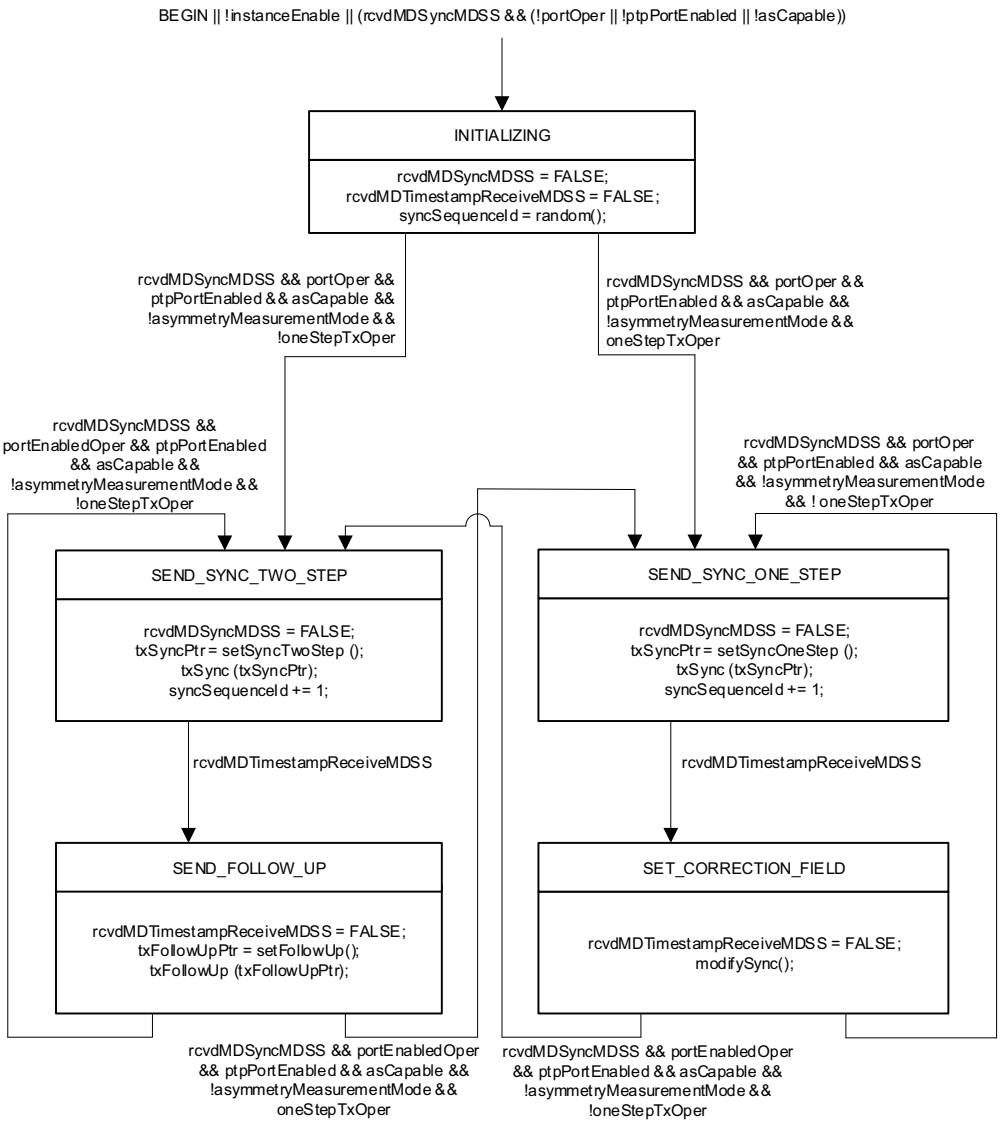

Figure 11-7—MDSyncSendSM state machine

11.2.16 OneStepTxOperSetting state machine

11.2.16.1 State machine variables

The following variables are used in the state diagram in Figure 11-8 (in 11.2.16.2):

11.2.16.1.1 rcvdSignalingMsg4: A Boolean variable that notifies the current state machine when a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3) is received. This variable is reset by the current state machine.

11.2.16.1.2 rcvdSignalingPtrOSTOS: A pointer to a structure whose members contain the values of the fields of the received Signaling message that contains a Message Interval Request TLV (see 10.6.4.3).

11.2.16.2 State diagram

The OneStepTxOperSetting state machine shall implement the function specified by the state diagram in Figure 11-8, the local variables specified in 11.2.16.1, the messages specified in 10.6 and 11.4, the relevant global variables specified in 10.2.5 and 11.2.13, and the relevant managed objects specified in 14.8. This state machine is responsible for setting the relevant global variables and managed objects pertaining to one-step/two-step operation.

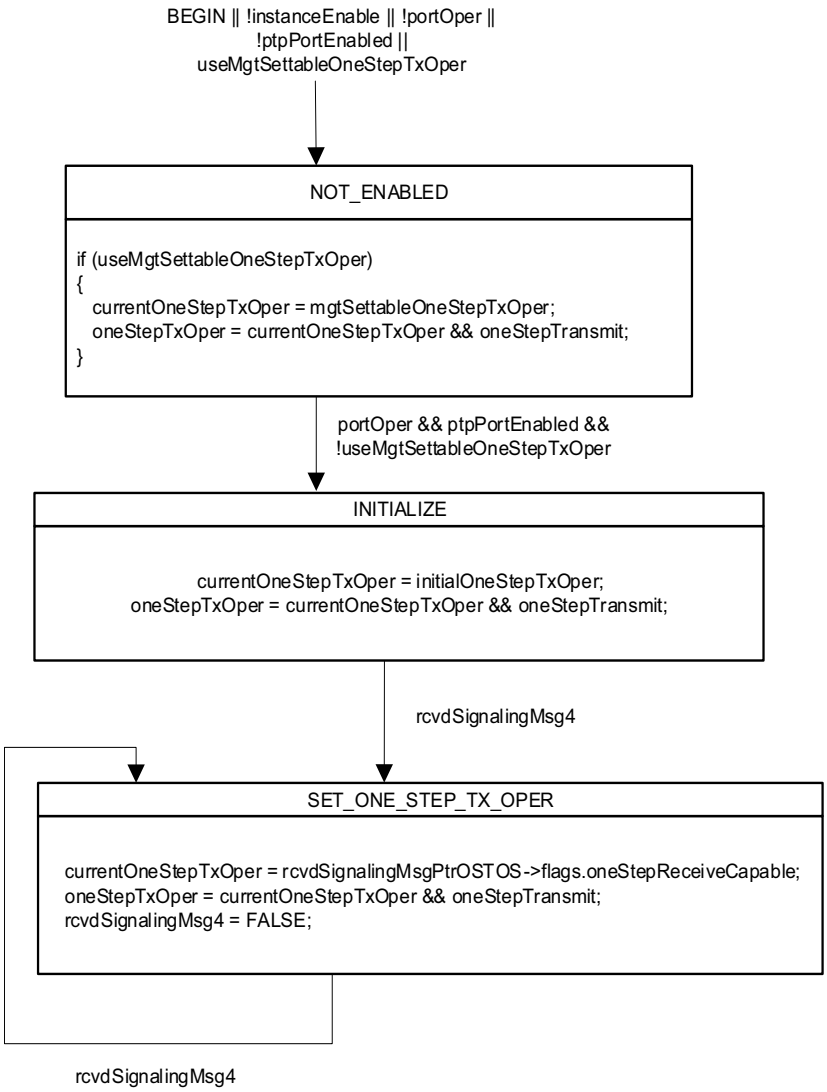

Figure 11-8—OneStepTxOperSetting state machine

11.2.17 Common Mean Link Delay Service (CMLDS)

11.2.17.1 General

Each port of a time-aware system invokes a single instance of the MDPdelayReq state machine (see 11.2.19) and the MDPdelayResp state machine (see 11.2.20). If the time-aware system implements more than one domain, these two state machines shall provide a Common Mean Link Delay Service (CMLDS), as described in this subclause, that measures mean propagation delay on the PTP Link attached to the port and the neighbor rate ratio for the port (i.e., the ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the PTP Link attached to this port, to the frequency of the LocalClock entity of this time-aware system). The CMLDS makes the mean propagation delay and neighbor rate ratio available to all active domains. If the time-aware system implements one domain (the domainNumber of this domain is 0; see 8.1), these two state machines may provide the CMLDS; however, if they do not provide the CMLDS (i.e., if only the PTP Instance-specific peer delay mechanism is provided), they shall be invoked on domain 0. In other words, if the domain number is not 0, portDS.delayMechanism (see Table 14-8 in 14.8.5) must not be P2P.

NOTE 1—In the above sentence, the condition that the time-aware system implements only one domain implicitly assumes that IEEE 802.1AS is the only PTP profile present on the respective port of the time-aware system, i.e., no other PTP profiles are implemented on that port. If other PTP profiles that use the CMLDS are present on the port, the CMLDS must be provided.

In accordance with IEEE Std 1588-2019, the term *Link Port* refers to a port of the CMLDS. A PTP Port for which portDS.delayMechanism is COMMON_P2P uses the CMLDS provided by the Link Port whose cmldsLinkPortDS.portIdentity.portNumber (see 14.16.2) is equal to the commonServicesPortDS.cmldsLinkPortPortNumber (see 14.14.2) for this PTP Port.

The value of majorSdoId for the CMLDS shall be 0x2. The value of minorSdoId for the Common Mean Link Delay Service shall be 0x00. As a result, the value of sdoId for the Common Mean Link Delay Service is 0x200.

NOTE 2—The above requirements for majorSdoId and minorSdoId are for the CMLDS. The requirements for gPTP domains, including instance-specific peer delay messages, are given in 8.1.

If a PTP Port that invokes the CMLDS receives a Pdelay_Req message with majorSdoId value of 0x1, minorSdoId value of 0x00, and domainNumber value of 0, the PTP Port shall respond with PTP Instance-specific peer delay messages (i.e., the Pdelay_Resp and Pdelay_Resp_Follow_Up corresponding to this Pdelay_Req) using the instance-specific peer-to-peer delay mechanism. These instance-specific messages have majorSdoId value of 0x1, minorSdoId value of 0x00, and domainNumber value of 0.

NOTE 3—The above requirement ensures:

a) Backward compatibility with time-aware systems that comply with the 2011 version of this standard and
b) Compatibility with time-aware systems that implement only one domain and invoke the MDPdelayReq and MDPdelayResp state machines on domain 0.

NOTE 4—In general, a port can receive:

a) Peer delay messages of the CMLDS,
b) PTP Instance-specific peer delay messages of domain 0 (with sdoId of 0x100), and
c) If there are other PTP profiles on the neighbor port that use instance-specific peer delay, peer delay messages of those profiles.

The port responds to the messages of type a) if it invokes CMLDS, the messages of type b) if it invokes gPTP domain 0, and the messages of type c) if it invokes the respective other PTP profiles.

The CMLDS shall be enabled on a Link Port if the value of portDS.delayMechanism (see 14.8.5) is COMMON_P2P for at least one PTP Port that is enabled (i.e., for which portOper and ptpPortEnabled are

both TRUE) and corresponds to the same physical port as the Link Port (see 14.1). The value of cmldsLinkPortEnabled is TRUE if the CMLDS is enabled on the Link Port and FALSE if the CMLDS is not enabled on the Link Port.

11.2.17.2 Differences between instance-specific peer-to-peer delay mechanism and CMLDS in computations of mean link delay and effect of delayAsymmetry

The MDPdelayReq state machine (see 11.2.19), MDPdelay_Resp state machine (see 11.2.20), and MDSyncReceiveSM state machine (see 11.2.14) perform various computations of mean link delay and of the effect of delayAsymmetry differently, depending on whether the respective computations are done using the instance-specific peer-to-peer delay mechanism or using CMLDS. The differences are described as follows:

a) Instance-specific peer delay computes mean link delay averaged over the two directions and adds delayAsymmetry separately when computing upstreamTxTime by the setMDSyncReceiveMDSR() function of the MDSyncReceive state machine, while CMLDS corrects the computed mean link delay for delayAsymmetry and therefore does not need to add it separately (see 11.2.14.2.1).

b) Instance-specific peer delay sets the correctionField of a transmitted Pdelay_Req message to 0, while CMLDS sets it to –delayAsymmetry (see 11.2.19.3.1).

c) Instance-specific peer delay sets the correctionField of Pdelay_Resp equal to the fractional nanoseconds portion of the pdelayReqEventIngressTimestamp of the corresponding Pdelay_Req, while CMLDS sets the correctionField of Pdelay_Resp equal to minus the fractional nanoseconds portion of the pdelayReqEventIngressTimestamp of the corresponding Pdelay_Req (see 11.2.20.3.1).

d) Instance-specific peer delay sets the correctionField of Pdelay_Resp_Follow_Up equal to the fractional nanoseconds portion of the pdelayRespEventEgressTimestamp, while CMLDS sets the correctionField of Pdelay_Resp_Follow_Up equal to the sum of the correctionField of the corresponding Pdelay_Req and the fractional nanoseconds portion of the pdelayRespEventEgressTimestamp (see 11.2.20.3.3).

e) When computing mean link delay [i.e., the quantity D in Equation (11-5)], the correctionField of the Pdelay_Resp message, divided by 2^{16}, is added when computing the quantity t_2 if instance-specific peer delay is used, while it is subtracted if CMLDS is used (see 11.2.19.3.4).

f) When computing neighborRateRatio, the computation of the correctResponderEventTimestamp must be corrected for delayAsymmetry if, and only if, CMLDS is used. The reason for this correction is that, with CMLDS, delayAsymmetry was subtracted from the Pdelay_Req correctionField, and then the Pdelay_Resp_Follow_Up correctionField was set equal to the sum of the Pdelay_Req correctionField and the fractional nanoseconds portion of the PdelayRespEventEgressTimestamp, while with instance-specific peer delay, the correctionField of Pdelay_Req was set equal to 0 (see 11.2.19.3.1, 11.2.19.3.3, and 11.2.20.3.3).

The computations in this standard for the instance-specific peer-to-peer delay mechanism are the same as in IEEE Std 802.1AS-2011, for backward compatibility. However, the computations in this standard for CMLDS must be consistent with IEEE Std 1588-2019 because CMLDS can be used by other PTP profiles, in addition to the PTP profile included in IEEE Std 802.1AS, that might be present in a gPTP node. Therefore, the computations for the instance-specific peer-to-peer delay mechanism and CMLDS are different.

11.2.18 Common Mean Link Delay Service (CMLDS) global variables

11.2.18.1 cmldsLinkPortEnabled: A per-Link-Port Boolean that is TRUE if both the value of portDS.delayMechanism is Common_P2P and the value of portDS.ptpPortEnabled is TRUE, for at least one PTP Port that uses the CMLDS that is invoked on the Link Port; otherwise, the value is FALSE.

11.2.19 MDPdelayReq state machine

11.2.19.1 General

This state machine is invoked as part of the Common Mean Link Delay Service (CMLDS). There is one instance of this state machine for all the domains (per port). As a result, there also is one instance of each of the state machine variables of 11.2.19.2, state machine functions of 11.2.19.3, and relevant global variables of 10.2.5, 11.2.13, and 11.2.18 for all the domains (per port). None of the variables used or functions invoked in this state machine are specific to a single domain. However, the single instances of all of these objects or entities are accessible to all the domains.

NOTE—This state machine uses the variable asCapableAcrossDomains (see 11.2.13.12). When only one domain is active, asCapableAcrossDomains is equivalent to the variable asCapable.

11.2.19.2 State machine variables

The following variables are used in the state diagram in Figure 11-9 (in 11.2.19.4):

11.2.19.2.1 pdelayIntervalTimer: A variable used to save the time at which the Pdelay_Req interval timer is started. A Pdelay_Req message is sent when this timer expires. The data type for pdelayIntervalTimer is UScaledNs.

11.2.19.2.2 rcvdPdelayResp: A Boolean variable that notifies the current state machine when a Pdelay_Resp message is received. This variable is reset by the current state machine.

11.2.19.2.3 rcvdPdelayRespPtr: A pointer to a structure whose members contain the values of the fields of the Pdelay_Resp message whose receipt is indicated by rcvdPdelayResp (see 11.2.19.2.2).

11.2.19.2.4 rcvdPdelayRespFollowUp: A Boolean variable that notifies the current state machine when a Pdelay_Resp_Follow_Up message is received. This variable is reset by the current state machine.

11.2.19.2.5 rcvdPdelayRespFollowUpPtr: A pointer to a structure whose members contain the values of the fields of the Pdelay_Resp_Follow_Up message whose receipt is indicated by rcvdPdelayRespFollowUp (see 11.2.19.2.4).

11.2.19.2.6 txPdelayReqPtr: A pointer to a structure whose members contain the values of the fields of a Pdelay_Req message to be transmitted.

11.2.19.2.7 rcvdMDTimestampReceiveMDPReq: A Boolean variable that notifies the current state machine when the pdelayReqEventEgressTimestamp (see 11.3.2.1) for a transmitted Pdelay_Req message is received. This variable is reset by the current state machine.

11.2.19.2.8 pdelayReqSequenceId: A variable that holds the sequenceId for the next Pdelay_Req message to be transmitted by this MD entity. The data type for pdelayReqSequenceId is UInteger16.

11.2.19.2.9 lostResponses: A count of the number of consecutive Pdelay_Req messages sent by the port, for which Pdelay_Resp and/or Pdelay_Resp_Follow_Up messages are not received. The data type for lostResponses is UInteger16.

11.2.19.2.10 neighborRateRatioValid: A Boolean variable that indicates whether the function computePdelayRateRatio() (see 11.2.19.3.3) successfully computed neighborRateRatio (see 10.2.5.7).

11.2.19.2.11 detectedFaults: A count of the number of consecutive faults (see 11.5.4 for the definition of a fault).

11.2.19.2.12 portEnabled0: A Boolean variable whose value is equal to ptpPortEnabled (see 10.2.5.13) if this state machine is invoked by the instance-specific peer-to-peer delay mechanism and is equal to cmldsLinkPortEnabled (see 11.2.18.1) if this state machine is invoked by the CMLDS.

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 State machine functions

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:

a) sourcePortIdentity is set equal to the port identity of the port corresponding to this MD entity (see 8.5.2).
b) sequenceId is set equal to pdelayReqSequenceId (see 11.2.19.2.8).
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.
d) The remaining parameters are set as specified in 11.4.2 and 11.4.5.

11.2.19.3.2 txPdelayReq(txPdelayReqPtr): Transmits a Pdelay_Req message from the MD entity, containing the parameters in the structure pointed to by txPdelayReqPtr (see 11.2.19.2.6).

11.2.19.3.3 computePdelayRateRatio(): Computes neighborRateRatio (see 10.2.5.7) using the following information conveyed by successive Pdelay_Resp and Pdelay_Resp_Follow_Up messages:

a) The pdelayRespEventIngressTimestamp (see 11.3.2.1) values for the respective Pdelay_Resp messages
b) The correctedResponderEventTimestamp values, whose data type is UScaledNs, obtained by adding the following fields of the received Pdelay_Resp_Follow_Up message:
    1) The seconds field of the responseOriginTimestamp field, multiplied by 10^9
    2) The nanoseconds field of the responseOriginTimestamp parameter
    3) The correctionField, divided by 2^{16}
    4) delayAsymmetry, if and only if this state machine is invoked by CMLDS

NOTE 1—If delayAsymmetry does not change during the time interval over which neighborRateRatio is computed, it is not necessary to subtract 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.

Any scheme that uses the preceding information, along with any other information conveyed by the successive Pdelay_Resp and Pdelay_Resp_Follow_Up messages, to compute neighborRateRatio is acceptable as long as the performance requirements specified in B.2.4 are met. If neighborRateRatio is successfully computed, the Boolean neighborRateRatioValid (see 11.2.19.2.10) is set to TRUE. If neighborRateRatio is not successfully computed (e.g., if the MD entity has not yet exchanged a sufficient number of peer delay messages with its peer), the Boolean neighborRateRatioValid is set to FALSE.

NOTE 2—As one example, neighborRateRatio can be estimated as the ratio of the elapsed time of the LocalClock entity of the time-aware system at the other end of the PTP Link attached to this port, to the elapsed time of the LocalClock entity of this time-aware system. This ratio can be computed for the time interval between a set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages and a second set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages some number of Pdelay_Req message transmission intervals later, i.e.,

$$\frac{\text{correctedResponderEventTimestamp}_N - \text{correctedResponderEventTimestamp}_0}{\text{pdelayRespEventIngressTimestamp}_N - \text{pdelayRespEventIngressTimestamp}_0}$$

where N is the number of Pdelay_Req message transmission intervals separating the first set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages and the second set, and the successive sets of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages are indexed from 0 to N with the first set indexed 0.

NOTE 3—This function must account for non-receipt of Pdelay_Resp and/or Pdelay_Resp_Follow_Up for a Pdelay_Req message and also for receipt of multiple Pdelay_Resp messages within one Pdelay_Req message transmission interval.

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).

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

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
- 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)

Propagation delay averaging may be performed, as described in 11.1.2 by Equation (11-2), Equation (11-3), and Equation (11-4). In this case, the successive values of propagation delay computed using Equation (11-5) are input to either Equation (11-2) or Equation (11-4), and the computed average propagation delay is returned by this function.

NOTE 1—Equation (11-5) defines D as the mean propagation delay relative to the time base of the time-aware system at the other end of the attached PTP Link. It is divided by neighborRateRatio (see 10.2.5.7) to convert it to the time base of the current time-aware system when subtracting from syncEventIngressTimestamp to compute upstreamTxTime [see item f) in 11.2.14.2.1].

NOTE 2—The difference between mean propagation delay relative to the Grandmaster Clock time base and relative to the time base of the time-aware system at the other end of the attached PTP Link is usually negligible. To see this, note that the former can be obtained from the latter by multiplying the latter by the ratio of the Grandmaster Clock frequency to the frequency of the LocalClock entity of the time-aware system at the other end of the PTP Link attached to this port.

This ratio differs from 1 by 200 ppm or less. For example, for a worst-case frequency offset of the LocalClock entity of the time-aware system at the other end of the PTP Link, relative to the Grandmaster Clock, of 200 ppm, and a measured propagation time of 100 ns, the difference in *D* relative to the two time bases is 20 ps.

NOTE 3—In IEEE Std 1588-2019, the computation of Equation (11-5) is organized differently from the organization used in the present standard. Using the definitions of t_2 and t_3 above, Equation (11-5) can be rewritten as shown in Equation (11-6).

$$D = [r\cdot(t_4 - t_1) - (\text{responseOriginTimestamp} - \text{requestReceiptTimestamp}) + (\text{correctionField of Pdelay\_Resp}) - (\text{correctionField of Pdelay\_Resp\_Follow\_Up})] / 2 \quad (11\text{-}6)$$

where each term is expressed in units of nanoseconds as described in the definitions of t_1, t_2, t_3, and t_4 above. In IEEE Std 1588-2019, the fractional nanoseconds portion of t_2 is subtracted from the correctionField of Pdelay_Resp, rather than added as in this standard; however, the correctionField of Pdelay_Resp is then subtracted in Equation (11-6) rather than added, 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 for interoperability with IEEE Std 1588-2019 (see 11.2.17.2).

11.2.19.4 State diagram

The MDPdelayReq state machine shall implement the function specified by the state diagram in Figure 11-9, the local variables specified in 11.2.19.2, the functions specified in 11.2.19.3, the messages specified in 11.4, and the relevant global variables and functions specified in 11.2.13 and 11.2.18 and 10.2.4 through 10.2.6. This state machine is responsible for the following:

a) Sending Pdelay_Req messages and restarting the pdelayIntervalTimer.
b) Detecting that the peer mechanism is running.
c) Detecting if Pdelay_Resp and/or Pdelay_Resp_Follow_Up messages corresponding to a Pdelay_Req message sent are not received.
d) Detecting whether more than one Pdelay_Resp is received within one Pdelay_Req message transmission interval (see 11.5.2.2).
e) Computing propagation time on the attached PTP Link when Pdelay_Resp and Pdelay_Resp_Follow_Up messages are received.
f) Computing the ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the attached PTP Link to the frequency of the LocalClock entity of the current time-aware system.

NOTE 1—The ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the attached PTP Link to the frequency of the LocalClock entity of the current time-aware system, neighborRateRatio, retains its most recent value when a Pdelay_Resp and/or Pdelay_Resp_Follow_Up message is lost.

NOTE 2—Normally, Pdelay_Resp should be received within a time interval after sending Pdelay_Req that is less than or equal to the Pdelay turnaround time plus twice the mean propagation delay. However, while receiving Pdelay_Resp after a time interval that is longer than this can result in worse time-synchronization performance (see 11.1.2 and B.2.3), the peer delay protocol will still operate. It is expected that a peer delay initiator can receive and process Pdelay_Resp and Pdelay_Resp_Follow_Up messages within a time interval after sending Pdelay_Req that is as large as the current Pdelay Req message transmission interval (see 11.5.2.2).

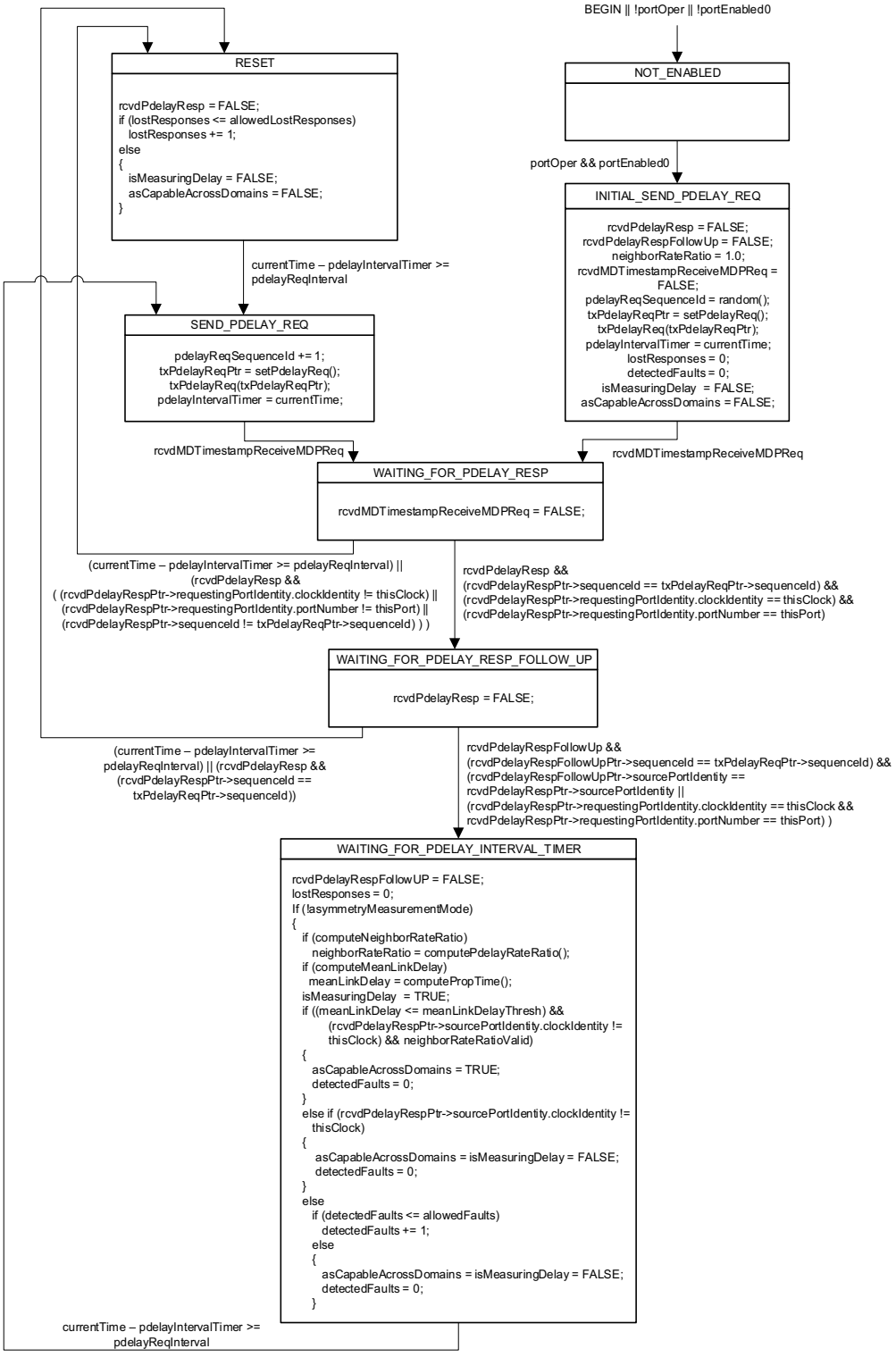

Figure 11-9—MDPdelayReq state machine

11.2.20 MDPdelayResp state machine

11.2.20.1 General

This state machine is invoked as part of the Common Mean Link Delay Service (CMLDS). There is one instance of this state machine for all the domains (per port). As a result, there also is one instance of each of the state machine variables of 11.2.20.2, state machine functions of 11.2.20.3, and relevant global variables of 10.2.5, 11.2.13, and 11.2.18 for all the domains (per port). None of the variables used or functions invoked in this state machine are specific to a single domain. However, the single instances of all of these objects or entities are accessible to all the domains.

11.2.20.2 State machine variables

The following variables are used in the state diagram in Figure 11-10 (in 11.2.20.4):

11.2.20.2.1 rcvdPdelayReq: A Boolean variable that notifies the current state machine when a Pdelay_Req message is received. This variable is reset by the current state machine.

11.2.20.2.2 rcvdMDTimestampReceiveMDPResp: A Boolean variable that notifies the current state machine when the pdelayRespEventEgressTimestamp (see 11.3.2.1) for a transmitted Pdelay_Resp message is received. This variable is reset by the current state machine.

11.2.20.2.3 txPdelayRespPtr: A pointer to a structure whose members contain the values of the fields of a Pdelay_Resp message to be transmitted.

11.2.20.2.4 txPdelayRespFollowUpPtr: A pointer to a structure whose members contain the values of the fields of a Pdelay_Resp_Follow_Up message to be transmitted.

11.2.20.2.5 portEnabled1: A Boolean variable whose value is equal to ptpPortEnabled (see 10.2.5.13) if this state machine is invoked by the instance-specific peer-to-peer delay mechanism and is equal to cmldsLinkPortEnabled (see 11.2.18.1) if this state machine is invoked by the CMLDS.

11.2.20.3 State machine functions

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:

a) sourcePortIdentity is set equal to the port identity of the port corresponding to this MD entity (see 8.5.2).
b) sequenceId is set equal to the sequenceId field of the corresponding Pdelay_Req message.
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.
e) requestingPortIdentity is set equal to the sourcePortIdentity field of the corresponding Pdelay_Req message.
f) The remaining parameters are set as specified in 11.4.2 and 11.4.6.

11.2.20.3.2 txPdelayResp(txPdelayRespPtr): Transmits a Pdelay_Resp message from the MD entity, containing the parameters in the structure pointed to by txPdelayRespPtr (see 11.2.20.2.3).

11.2.20.3.3 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:

a) sourcePortIdentity is set equal to the port identity of the port corresponding to this MD entity (see 8.5.2).
b) sequenceId is set equal to the sequenceId field of the corresponding Pdelay_Req message.
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,
e) requestingPortIdentity is set equal to the sourcePortIdentity field of the corresponding Pdelay_Req message.
f) The remaining parameters are set as specified in 11.4.2 and 11.4.6.

11.2.20.3.4 txPdelayRespFollowUp(txPdelayRespFollowUpPtr): Transmits a Pdelay_Resp_Follow_Up message from the P2PPort entity containing the parameters in the structure pointed to by txPdelayRespFollowUpPtr (see 11.2.20.2.4).

11.2.20.4 State diagram

The MDPdelayResp state machine shall implement the function specified by the state diagram in Figure 11-10, the local variables specified in 11.2.20.2, the functions specified in 11.2.20.3, the messages specified in 11.4, and the relevant global variables and functions specified in 10.2.4 through 10.2.6, 11.2.13, and 11.2.18. This state machine is responsible for responding to Pdelay_Req messages, received from the MD entity at the other end of the attached PTP Link, with Pdelay_Resp and Pdelay_Resp_Follow_Up messages.

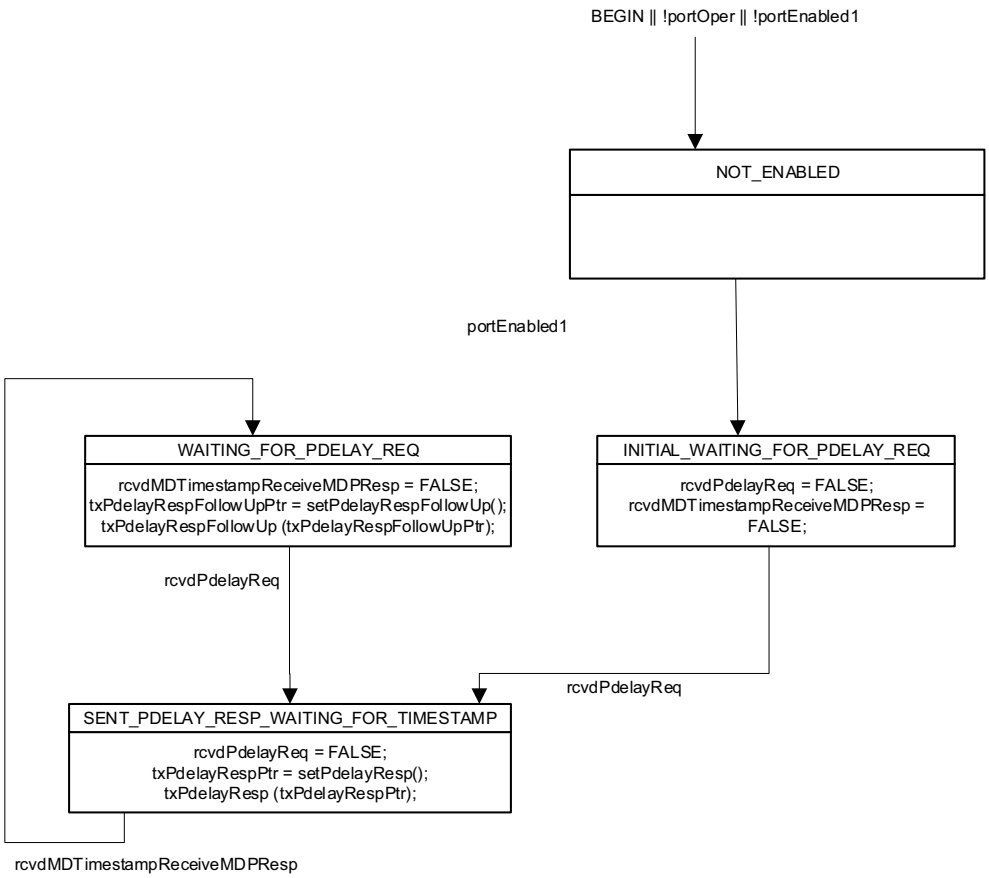

Figure 11-10—MDPdelayResp state machine

11.2.21 LinkDelayIntervalSetting state machine

11.2.21.1 General

This state machine is part of the Common Mean Link Delay Service (CMLDS). There is one instance of this state machine per port, for the Common Service of the time-aware system.

11.2.21.2 State machine variables

The following variables are used in the state diagram in Figure 11-11 (in 11.2.21.4):

11.2.21.2.1 rcvdSignalingMsg1: A Boolean variable that notifies the current state machine when a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3) is received. This variable is reset by the current state machine.

11.2.21.2.2 rcvdSignalingPtrLDIS: A pointer to a structure whose members contain the values of the fields of the received Signaling message that contains a Message Interval Request TLV (see 10.6.4.3).

11.2.21.2.3 portEnabled3: A Boolean variable whose value is equal to ptpPortEnabled (see 10.2.5.13) if this state machine is invoked by the instance-specific peer-to-peer delay mechanism and is equal to cmldsLinkPortEnabled (see 11.2.18.1) if this state machine is invoked by the CMLDS.

11.2.21.2.4 logSupportedPdelayReqIntervalMax: The maximum supported logarithm to base 2 of the Pdelay_Req interval. The data type for logSupportedPdelayReqIntervalMax is Integer8.

11.2.21.2.5 logSupportedClosestLongerPdelayReqInterval: The logarithm to base 2 of the Pdelay_Req interval, such that logSupportedClosestLongerPdelayReqInterval > logRequestedPdelayReqInterval, that is numerically closest to logRequestedPdelayReqInterval, where logRequestedPdelayReqInterval is the argument of the function computeLogPdelayReqInterval() (see 11.2.21.3.2). The data type for logSupportedClosestLongerPdelayReqInterval is Integer8.

11.2.21.2.6 computedLogPdelayReqInterval: A variable used to hold the result of the function computeLogPdelayReqInterval(). The data type for computedLogPdelayReqInterval is Integer8.

11.2.21.3 State machine functions

11.2.21.3.1 isSupportedLogPdelayReqInterval (logPdelayReqInterval): A Boolean function that returns TRUE if the Pdelay_Req interval given by the argument logPdelaytReqInterval is supported by the PTP Port and FALSE if the Pdelay_Req interval is not supported by the PTP Port. The argument logPdelayReqInterval has the same data type and format as the field logLinkDelayInterval of the message interval request TLV (see 10.6.4.3.6).

11.2.21.3.2 computeLogPdelayReqInterval (logRequestedPdelayReqInterval): An Integer8 function that computes and returns the logPdelayReqInterval, based on the logRequestedPdelayReqInterval. This function is defined as indicated below. It is defined here so that the detailed code that it invokes does not need to be placed into the state machine diagram.

```
Integer8 computeLogPdelayReqInterval (logRequestedPdelayReqInterval)
Integer8 logRequestedPdelayReqInterval;
{
        Integer8 logSupportedPdelayReqIntervalMax,
                                logSupportedClosestLongerPdelayReqInterval;
```

```
        if (isSupportedLogPdelayReqInterval (logRequestedPdelayReqInterval))
                // The requested Pdelay_Req Interval is supported and returned
                return (logRequestedPdelayReqInterval)
        else
        {
                if (logRequestedPdelayReqInterval > logSupportedPdelayReqIntervalMax)
                        // Return the fasted supported rate, even if faster than the requested rate
                        return (logSupportedPdelayReqIntervalMax);
                else
                        // Return the fastest supported rate that is still slower than
                        // the requested rate.
                        return (logSupportedClosestLongerPdelayReqInterval);
        }
}
```

11.2.21.4 State diagram

The LinkDelayIntervalSetting state machine shall implement the function specified by the state diagram in Figure 11-11, the local variables specified in 11.2.21.2, the functions specified in 11.2.21.3, the messages specified in 10.6 and 11.4, the relevant global variables specified in 10.2.5, 11.2.13, and 11.2.18, the relevant managed objects specified in 14.8 and 14.14, and the relevant timing attributes specified in 10.7 and 11.5. This state machine is responsible for setting the global variables that give the duration of the mean interval between successive Pdelay_Req messages and also the global variables that control whether meanLinkDelay and neighborRateRatio are computed, both at initialization and in response to the receipt of a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3).

NOTE—A Signaling message received by this state machine that carries the message interval request TLV (see 10.6.4.3) is ignored if multiple profiles are present and the Signaling message is directed to the CMLDS (i.e., if the value of sdoId of the Signaling message is 0x200).

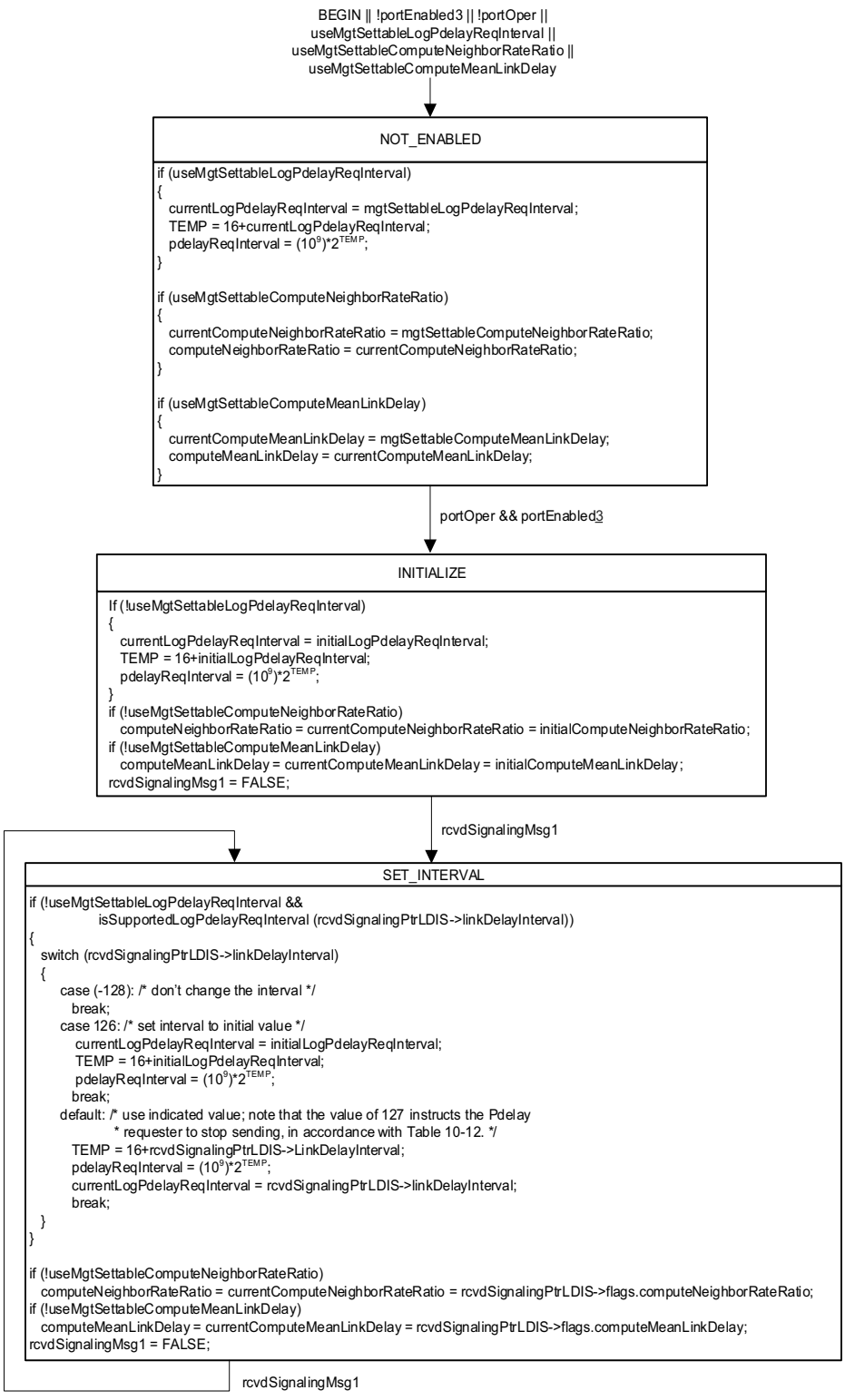

Figure 11-11—LinkDelayIntervalSetting state machine

11.3 Message attributes

11.3.1 General

This subclause describes attributes of the Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages that are not described in 8.4.2.

11.3.2 Message types contained in each message class

11.3.2.1 Event message class

The event message class contains the following message types:

a) Sync: A Sync message contains time-synchronization information that originates at a ClockMaster entity. The appearance of a Sync message at the reference plane of the PTP Port corresponding to an MD entity is an event to which the LocalClock assigns a timestamp, the syncEventIngressTimestamp or syncEventEgressTimestamp, based on the time of the LocalClock. The syncEventIngressTimestamp and syncEventEgressTimestamp are measured relative to the timestamp measurement plane; the MD entity corrects them for ingress and egress latencies, respectively (see 8.4.3). The Sync message is followed by a Follow_Up message containing synchronization information that is based in part on the sum of the syncEventEgressTimestamp and any egressLatency (see 8.4.3).

b) Pdelay_Req: A Pdelay_Req message is transmitted by an MD entity to another MD entity as part of the peer-to-peer delay mechanism (see 11.2.19 and Figure 11-9) to determine the delay on the PTP Link between them. The appearance of a Pdelay_Req message at the reference plane of the port of an MD entity is an event to which the LocalClock assigns a timestamp, the pdelayReqEventIngressTimestamp or pdelayReqEventEgressTimestamp, based on the time of the LocalClock. The pdelayReqEventIngressTimestamp and pdelayReqEventEgressTimestamp are measured relative to the timestamp measurement plane; the MD entity corrects them for ingress and egress latencies, respectively (see 8.4.3).

c) Pdelay_Resp: A Pdelay_Resp message is transmitted by an MD entity to another MD entity in response to the receipt of a Pdelay_Req message. The Pdelay_Resp message contains the pdelayReqEventIngressTimestamp of the Pdelay_Req message to which it is transmitted in response. The appearance of a Pdelay_Resp message at the reference plane of the port of an MD entity is an event to which the LocalClock assigns a timestamp, the pdelayRespEventIngressTimestamp or pdelayRespEventEgressTimestamp, based on the time of the LocalClock. The pdelayRespEventIngressTimestamp and pdelayRespEventEgressTimestamp are measured relative to the timestamp measurement plane; the MD entity corrects them for ingress and egress latencies, respectively (see 8.4.3). The Pdelay_Resp message is followed by a Pdelay_Resp_Follow_Up message containing the sum of the pdelayRespEventEgressTimestamp and any egressLatency (see 8.4.3).

Event messages shall be assigned the timestamps previously defined, in accordance with 8.4.3.

11.3.2.2 General message class

The general message class contains the following message types:

a) Follow_Up: A Follow_Up message communicates the value of the syncEventEgressTimestamp for the associated Sync message.

b) Pdelay_Resp_Follow_Up: A Pdelay_Resp_Follow_Up message communicates the value of the PdelayRespEventEgressTimestamp for the associated Pdelay_Resp message.

General messages are not required to be timestamped.

11.3.3 VLAN tag

A frame that carries an IEEE 802.1AS message shall not have a VLAN tag nor a priority tag (see 3.184 of IEEE Std 802.1Q-2018).

11.3.4 Addresses

The destination address of Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages shall be the reserved multicast address given in Table 11-3.

Table 11-3—Destination address for Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages

Destination address
01-80-C2-00-00-0E
NOTE—This address is taken from Table 8-1, Table 8-2, and Table 8-3 of IEEE Std 802.1Q-2018.

All Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages shall use the MAC address of the respective egress physical port as the source address.

11.3.5 EtherType

The EtherType of Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages shall be the EtherType given in Table 11-4.

Table 11-4—EtherType for Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages

EtherType
88-F7

11.3.6 Subtype

The subtype of the Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages is indicated by the majorSdoId field (see 10.6.2.2.1).

NOTE—The subtype for all PTP messages is indicated by the majorSdoId field.

11.3.7 Source port identity

The Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages each contain a sourcePortIdentity field (see 11.4.2.7) that identifies the egress port (see 8.5) on which the respective message is sent.

11.3.8 Sequence number

Each MD entity shall maintain a separate sequenceId pool for each of the message types Sync and Pdelay_Req.

Each Sync and Pdelay_Req message contains a sequenceId field (see 11.4.2.8), which carries the message sequence number. The sequenceId of a Sync message shall be one greater than the sequenceId of the previous Sync message sent by the transmitting port, subject to the constraints of the rollover of the UInteger16 data type used for the sequenceId field. The sequenceId of a Pdelay_Req message shall be one greater than the sequenceId of the previous Pdelay_Req message sent by the transmitting port, subject to the constraints of the rollover of the UInteger16 data type used for the sequenceId field.

Separate pools of sequenceId are not maintained for the following message types:

a) Pdelay_Resp
b) Follow_Up
c) Pdelay_Resp_Follow_Up

For these exceptions, the sequenceId value is specified in Table 11-7 (in 11.4.2.8).

11.3.9 Event message timestamp point

The message timestamp point for a PTP event message shall be the beginning of the first symbol following the start of frame delimiter.

11.4 Message formats

11.4.1 General

The PTP messages Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up shall each have a header, body, and if present, a suffix that contains one or more TLVs (see 10.6.2, 11.4.3, 11.4.4, 11.4.5, 11.4.6, and 11.4.7 of this standard and Clause 14 of IEEE Std 1588-2019). Reserved fields shall be transmitted with all bits of the field 0 and ignored by the receiver, unless otherwise specified. The data type of the field shall be the type indicated in brackets in the title of each subclause.

Subclause 11.4.4.3 defines the Follow_Up information TLV, which is carried by the Follow_Up message if the corresponding Sync message is two-step (i.e., twoStepFlag of the Sync message is TRUE; see 10.6.2.2.8) and by the Sync message if the message is one-step (i.e., twoStepFlag is FALSE). The Follow_Up information TLV is the first TLV of the Follow_Up message or Sync message. The requirements for the parsing and forwarding of TLVs are given in 10.6.1.

NOTE—The standard Ethernet header and FCS (18 bytes total) must be added to each message.

11.4.2 Header

11.4.2.1 General

The common header for the PTP messages Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up shall be as specified in 10.6.2, except as noted in the following subclauses.

11.4.2.2 messageType (Enumeration4)

The value indicates the type of the message, as defined in Table 11-5.

Table 11-5—Values for messageType field

Message type	Message class	Value
Sync	Event	0x0
Pdelay_Req	Event	0x2
Pdelay_Resp	Event	0x3
Follow_Up	General	0x8
Pdelay_Resp_Follow_Up	General	0xA
NOTE—Other values for the messageType field, except for 0xB that is used for the Announce message and 0xC that is used for the Signaling message (see 10.6.2.2.2), are not used in this standard.		

The most significant bit of the message ID field divides this field in half between event and general messages, i.e., it is 0 for event messages and 1 for general messages.

NOTE—The reserved nibble immediately following messageType is reserved for future expansion of the messageType field.

11.4.2.3 messageLength (UInteger16)

The value is the total number of octets that form the PTP message. The counted octets start with and include the first octet of the header and terminate with and include the last octet of the last TLV or, if there are no TLVs, with the last octet of the message as defined in this subclause.

NOTE—See 10.6.2.2.5 for an example.

11.4.2.4 domainNumber (UInteger8)

The domainNumber for Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages shall be 0. The domainNumber for all other PTP messages is as specified in 10.6.2.2.6.

11.4.2.5 flags (Octet2)

The value of the bits of the array are defined in Table 10-9.

11.4.2.6 correctionField (Integer64)

The correctionField is the value of the correction as specified in Table 11-6, measured in nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x0000000000028000.

A value of one in all bits, except the most significant, of the field, indicates that the correction is too big to be represented.

Table 11-6—Value of correctionField

Message type	Value
Follow_Up Sync (sent by a one-step PTP Port; see 11.1.3 and 11.2.13.9)	Corrections for fractional nanoseconds (see 10.2.9 and Figure 10-5), difference between preciseOriginTimestamp field (if sent by a two-step PTP Port) or originTimestamp field (if sent by a one-step PTP Port) and current synchronized time (see 11.2.15.2.3 and Figure 11-7), and asymmetry corrections (see 8.3, 11.2.14.2.1, and 11.2.15.2.3; the quantity delayAsymmetry is used in the computation of upstreamTxTime in 11.2.14.2.1, and upstreamTxTime is used in computing an addition to the correctionField in 11.2.15.2.3).
Pdelay_Resp, Pdelay_Resp_Follow_Up	Corrections for fractional nanoseconds (see Figure 11-9 and Figure 11-10) if the message is sent by the instance-specific peer-to-peer delay mechanism; or For Pdelay_Resp, minus the corrections for fractional nanoseconds (see 11.2.20.3.1, Figure 11-9, and Figure 11-10), and for Pdelay_Resp_Follow_Up, 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.
Sync (sent by a two-step PTP Port), Pdelay_Req, Announce, Signaling	The value is 0 (see 10.6.2.2.9) if the message is sent by the instance-specific peer-to-peer delay mechanism, or The value is 0 for Sync (sent by a two-step PTP Port), Announce, and Signaling, and –delayAsymmetry for Pdelay_Req (see 11.2.19.3.1), if the message is sent by CMLDS.
NOTE—IEEE Std 1588-2019 describes asymmetry corrections for the Pdelay_Req and Pdelay_Resp messages. However, the peer-to-peer delay mechanism computes the mean propagation delay. Here, where the gPTP communication path is a full-duplex point-to-point PTP Link, these corrections cancel in the mean propagation delay computation and therefore are not needed.	

11.4.2.7 sourcePortIdentity (PortIdentity)

The value is the portIdentity of the egress port (see 8.5.2) on which the respective message is sent.

11.4.2.8 sequenceId (UInteger16)

The value is assigned by the originator of the message in conformance with 11.3.8, except for Follow_Up, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages. The sequenceId field values for these exceptions are defined in the state diagrams given in the figures referenced in Table 11-7.

Table 11-7—References for sequenceId value exceptions

Message type	Reference
Follow_Up	See 11.2.15 and Figure 11-7
Pdelay_Resp	See 11.2.20 and Figure 11-10
Pdelay_Resp_Follow_Up	See 11.2.20 and Figure 11-10

11.4.2.9 logMessageInterval (Integer8)

For Sync and Follow_Up messages, the value is the value of currentLogSyncInterval (see 10.2.5.4 and 10.7.2.3). For Pdelay_Req messages, the value is the value of currentLogPdelayReqInterval. For Pdelay_Resp and Pdelay_Resp_Follow_Up messages, the value is transmitted as 0x7F and ignored on reception.

11.4.3 Sync message

11.4.3.1 General Sync message specifications

If the twoStep flag of the PTP common header (see Table 10-9) of the Sync message is TRUE, the fields of the Sync message shall be as specified in Table 11-8. If the twoStep flag of the PTP common header of the Sync message is FALSE, the fields of the Sync message shall be as specified in Table 11-9 and 11.4.3.2.

Table 11-8—Sync message fields if twoStep flag is TRUE

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 11.4.2)								34	0
reserved								10	34

Table 11-9—Sync message fields if twoStep flag is FALSE

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 11.4.2)								34	0
originTimestamp								10	34
Follow_Up information TLV								32	44

11.4.3.2 Sync message field specifications if twoStep flag is FALSE

11.4.3.2.1 originTimestamp (Timestamp)

The value of the originTimestamp field is the sourceTime of the ClockMaster entity of the Grandmaster PTP Instance, when the Sync message was sent by that Grandmaster PTP Instance, with any fractional nanoseconds truncated (see 10.2.9).

The sum of the correctionField and the originTimestamp field of the Sync message is the value of the synchronized time corresponding to the syncEventEgressTimestamp at the PTP Instance that sent the Sync message, including any fractional nanoseconds.

11.4.3.2.2 Follow_Up information TLV

The Sync message carries the Follow_Up information TLV, defined in 11.4.4.3. This TLV shall be the first TLV after the fixed fields.

11.4.4 Follow_Up message

11.4.4.1 General Follow_Up message specifications

The fields of the Follow_Up message shall be as specified in Table 11-10 and 11.4.4.2.

Table 11-10—Follow_Up message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 11.4.2)								34	0
preciseOriginTimestamp								10	34
Follow_Up information TLV								32	44

11.4.4.2 Follow_Up message field specifications

11.4.4.2.1 preciseOriginTimestamp (Timestamp)

The value of the preciseOriginTimestamp field is the sourceTime of the ClockMaster entity of the Grandmaster PTP Instance, when the associated Sync message was sent by that Grandmaster PTP Instance, with any fractional nanoseconds truncated (see 10.2.9).

The sum of the correctionFields in the Follow_Up and associated Sync messages, added to the preciseOriginTimestamp field of the Follow_Up message, is the value of the synchronized time corresponding to the syncEventEgressTimestamp at the PTP Instance that sent the associated Sync message, including any fractional nanoseconds.

11.4.4.2.2 Follow_Up information TLV

The Follow_Up message carries the Follow_Up information TLV, defined in 11.4.4.3. This TLV shall be the first TLV after the fixed fields.

11.4.4.3 Follow_Up information TLV definition

11.4.4.3.1 General

The fields of the Follow_Up information TLV shall be as specified in Table 11-11 and in 11.4.4.3.2 through 11.4.4.3.9. This TLV is a standard organization extension TLV for the Follow_Up message, as specified in 14.3 of IEEE Std 1588-2019.

NOTE—The Follow_Up information TLV is different from the CUMULATIVE_RATE_RATIO TLV of IEEE Std 1588-2019 (see 16.10 and Table 52 of IEEE Std 1588-2019). While both TLVs carry cumulative rate offset information, the Follow_Up information TLV also carries information on the Grandmaster Clock time base, most recent phase change, and most recent frequency change. The CUMULATIVE_RATE_RATIO TLV is not used by gPTP.

Table 11-11—Follow_Up information TLV

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
organizationId								3	4
organizationSubType								3	7
cumulativeScaledRateOffset								4	10
gmTimeBaseIndicator								2	14
lastGmPhaseChange								12	16
scaledLastGmFreqChange								4	28

11.4.4.3.2 tlvType (Enumeration16)

The value of the tlvType field is 0x3.

NOTE—This value indicates the TLV is a vendor and standard organization extension TLV, as specified in 14.3.2.1 and Table 52 of IEEE Std 1588-2019. The tlvType is specified in that standard as ORGANIZATION_EXTENSION with a value of 0x3.

11.4.4.3.3 lengthField (UInteger16)

The value of the lengthField is 28.

11.4.4.3.4 organizationId (Octet3)

The value of organizationId is 00-80-C2.

11.4.4.3.5 organizationSubType (Enumeration24)

The value of organizationSubType is 1.

11.4.4.3.6 cumulativeScaledRateOffset (Integer32)

The value of cumulativeScaledRateOffset is equal to (rateRatio – 1.0) × (2^{41}), truncated to the next smaller signed integer, where rateRatio is the ratio of the frequency of the Grandmaster Clock to the frequency of the LocalClock entity in the PTP Instance that sends the message.

NOTE—The above scaling allows the representation of fractional frequency offsets in the range [–(2^{-10} – 2^{-41}), 2^{-10} –2^{-41}], with granularity of 2^{-41}. This range is approximately [–9.766×10^{-4}, 9.766×10^{-4}].

11.4.4.3.7 gmTimeBaseIndicator (UInteger16)

The value of gmTimeBaseIndicator is the timeBaseIndicator of the ClockSource entity for the current Grandmaster PTP Instance (see 9.2.2.3).

NOTE—The timeBaseIndicator is supplied by the ClockSource entity to the ClockMaster entity via the ClockSourceTime.invoke function (see 9.2.2.3).

11.4.4.3.8 lastGmPhaseChange (ScaledNs)

The value of lastGmPhaseChange is the time of the current Grandmaster Clock minus the time of the previous Grandmaster Clock, at the time that the current Grandmaster PTP Instance became the Grandmaster PTP Instance. The value is copied from the lastGmPhaseChange member of the MDSyncSend structure whose receipt causes the MD entity to send the Follow_Up message (see 11.2.11).

11.4.4.3.9 scaledLastGmFreqChange (Integer32)

The value of scaledLastGmFreqChange is the fractional frequency offset of the current Grandmaster Clock relative to the previous Grandmaster Clock, at the time that the current Grandmaster PTP Instance became the Grandmaster PTP Instance, or relative to itself prior to the last change in gmTimeBaseIndicator, multiplied by 2^{41} and truncated to the next smaller signed integer. The value is obtained by multiplying the lastGmFreqChange member of MDSyncSend whose receipt causes the MD entity to send the Follow_Up message (see 11.2.11) by 2^{41}, and truncating to the next smaller signed integer.

NOTE—The above scaling allows the representation of fractional frequency offsets in the range $[-(2^{-10} - 2^{-41}), 2^{-10} - 2^{-41}]$, with granularity of 2^{-41}. This range is approximately $[-9.766 \times 10^{-4}, 9.766 \times 10^{-4}]$.

11.4.5 Pdelay_Req message

The fields of the Pdelay_Req message shall be as specified in Table 11-12.

Table 11-12—Pdelay_Req message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 11.4.2)								34	0
reserved								10	34
reserved								10	44

11.4.6 Pdelay_Resp message

11.4.6.1 General Pdelay_Resp message specifications

The fields of the Pdelay_Resp message shall be as specified in Table 11-13 and 11.4.6.2.

Table 11-13—Pdelay_Resp message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 11.4.2)								34	0
requestReceiptTimestamp								10	34
requestingPortIdentity								10	44

11.4.6.2 Pdelay_Resp message field specifications

11.4.6.2.1 requestReceiptTimestamp (Timestamp)

The value is the seconds and nanoseconds portion of the pdelayReqEventIngressTimestamp of the associated Pdelay_Req message (see 11.2.19).

11.4.6.2.2 requestingPortIdentity (PortIdentity)

The value is the value of the sourcePortIdentity field of the associated Pdelay_Req message (see 11.4.5).

11.4.7 Pdelay_Resp_Follow_Up message

11.4.7.1 General Pdelay_Resp_Follow_Up message specifications

The fields of the Pdelay_Resp_Follow_Up message shall be as specified in Table 11-14 and 11.4.7.2.

Table 11-14—Pdelay_Resp_Follow_Up message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 11.4.2)								34	0
responseOriginTimestamp								10	34
requestingPortIdentity								10	44

11.4.7.2 Pdelay_Resp_Follow_Up message field specifications

11.4.7.2.1 responseOriginTimestamp (Timestamp)

The value is the seconds and nanoseconds portion of the pdelayRespEventEgressTimestamp of the associated Pdelay_Resp message (see 11.4.6.2).

11.4.7.2.2 requestingPortIdentity (PortIdentity)

The value is the value of the sourcePortIdentity field of the associated Pdelay_Req message (see 11.4.5).

11.5 Protocol timing characterization

11.5.1 General

This subclause specifies timing attributes for the media-dependent sublayer specified in this clause.

11.5.2 Message transmission intervals

11.5.2.1 General interval specification

The mean time interval between successive Pdelay_Req messages is represented as the logarithm to the base 2 of this time interval measured in seconds. The value of this logarithmic attribute shall be as specified in 11.5.2.2.

The mean time interval between successive Sync messages shall be as specified in 10.7.2.1, 10.7.2.3, and 11.5.2.3.

11.5.2.2 Pdelay_Req message transmission interval

When useMgtSettableLogPdelayReqInterval (see 14.16.12) is FALSE, the initialLogPdelayReqInterval specifies the following:

a) The mean time interval between successive Pdelay_Req messages sent over a PTP Link when the port is initialized, and
b) The value to which the mean time interval between successive Pdelay_Req messages is set when a message interval request TLV is received with the logLinkDelayIntervalField set to 126 (see 11.2.21).

The currentLogPdelayReqInterval specifies the current value of the mean time interval between successive Pdelay_Req messages. The default value of initialLogPdelayReqInterval is 0. Every port supports the value 127; the port does not send Pdelay_Req messages when currentLogPdelayReqInterval has this value (see 11.2.21). A port may support other values, except for the reserved values indicated in Table 10-15. A port shall ignore requests for unsupported values (see 11.2.21). The initialLogPdelayReqInterval and currentLogPdelayReqInterval are per-port attributes.

When useMgtSettableLogPdelayReqInterval is TRUE, currentLogSyncInterval is set equal to mgtSettableLogPdelayReqInterval (see 14.16.13), and initialLogPdelayReqInterval is ignored.

NOTE 1—If useMgtSettableLogPdelayReqInterval is FALSE, the value of initialLogPdelayReqInterval is the value of the mean time interval between successive Pdelay_Req messages when the port is initialized. The value of the mean time interval between successive Pdelay_Req messages can be changed, e.g., if the port receives a Signaling message that carries a message interval request TLV (see 10.6.4.3) and the current value is stored in currentLogPdelayReqInterval. The value of the mean time interval between successive Pdelay_Req messages can be reset to the initial value, e.g., by a message interval request TLV for which the value of the field logLinkDelayInterval is 126 (see 10.6.4.3.6).

NOTE 2—A port that requests (using a Signaling message that contains a message interval request TLV; see 10.6.4 and 11.2.21) that the port at the other end of the attached PTP Link set its currentLogPdelayReqInterval to a specific value can determine if the request was honored by examining the logMessageInterval field of subsequent received Pdelay_Req messages.

NOTE 3—The MDPdelayReq state machine ensures that the times between transmission of successive Pdelay_Req messages, in seconds, are not smaller than $2^{\text{currentLogPdelayReqInterval}}$. This is consistent with IEEE Std 1588-2019, which requires that the logarithm to the base 2 of the mean value of the interval, in seconds, between Pdelay_Req message transmissions is no smaller than the interval computed from the value of the portDS.logMinPdelayReqInterval member of the data set of the transmitting PTP Instance. The sending of Pdelay_Req messages is governed by the LocalClock and not the

synchronized time (i.e., the estimate of the Grandmaster Clock time). Since the LocalClock frequency can be slightly larger than the Grandmaster Clock frequency (e.g., by 100 ppm, which is the specified frequency accuracy of the LocalClock; see B.1.1), it is possible for the time intervals between successive Pdelay_Req messages to be slightly less than $2^{currentLogPdelayReqInterval}$ when measured relative to the synchronized time.

11.5.2.3 Sync message transmission interval default value

The default value of initialLogSyncInterval (see 10.7.2.3) is –3. Every PTP Port supports the value 127; the PTP Port does not send Sync messages when currentLogSyncInterval has this value (see 10.3.18). A PTP Port may support other values, except for the reserved values indicated in Table 10-16. A PTP Port ignores requests for unsupported values (see 10.3.18).

NOTE—A PTP Port that requests (using a Signaling message that contains a message interval request TLV; see 10.6.4 and 10.3.18) that the PTP Port at the other end of the attached PTP Link set its currentLogSyncInterval to a specific value can determine if the request was honored by examining the logMessageInterval field of subsequent received Sync messages.

11.5.3 allowedLostResponses

The variable allowedLostResopnses (see 11.2.13.4) is the number of Pdelay_Req messages without valid responses above which a port is considered to be not exchanging peer delay messages with its neighbor. The default value of allowedLostResponses shall be 9. The range shall be 1 through 255.

11.5.4 allowedFaults

The variable allowedFaults (see 11.2.13.5) is the number of faults above which asCapableAcrossDomains is set to FALSE, i.e., the port is considered not capable of interoperating with its neighbor via the IEEE 802.1AS protocol (see 10.2.5.1). In this context, the term *faults* refers to instances where

a) The computed mean propagation delay, i.e., meanLinkDelay (see 10.2.5.8), exceeds the threshold, meanLinkDelayThresh (see 11.2.13.7) and/or
b) The computation of neighborRateRatio is invalid (see 11.2.19.2.10).

The default value of allowedFaults shall be 9. The range shall be 1 through 255.

NOTE—The above description of allowedFaults uses the variable asCapableAcrossDomains (see 11.2.13.12). When only one domain is active, asCapableAcrossDomains is equivalent to the variable asCapable.

11.6 Control of computation of neighborRateRatio

The variable computeNeighborRateRatio (see 10.2.5.10) indicates whether neighborRateRatio is to be computed by this port when the peer-to-peer delay mechanism is invoked.

When useMgtSettableComputeNeighborRateRatio (see 14.16.16) is FALSE, computeNeighborRateRatio is initialized to the value of initialComputeNeighborRateRatio.

The currentComputeNeighborRateRatio specifies the current value of computeNeighborRateRatio. The default value of initialComputeNeighborRateRatio is TRUE. The initialComputeNeighborRateRatio and currentComputeNeighborRateRatio are per-port attributes.

When useMgtSettableComputeNeighborRateRatio is TRUE, currentComputeNeighborRateRatio is set equal to mgtSettableComputeNeighborRateRatio (see 14.16.17), and initialComputeNeighborRateRatio is ignored.

NOTE—If useMgtSettableComputeNeighborRateRatio is FALSE, the value of initialComputeNeighborRateRatio determines whether neighborRateRatio is computed by the peer delay mechanism when the port is initialized. The value of computeNeighborRateRatio can be changed, e.g., if the port receives a Signaling message that carries a message interval request TLV (see 10.6.4.3) and the current value is stored in currentComputeNeighborRateRatio.

11.7 Control of computation of meanLinkDelay

The variable computeMeanLinkDelay (see 10.2.5.10) indicates whether meanLinkDelay is to be computed by this port when the peer-to-peer delay mechanism is invoked.

When useMgtSettableComputeMeanLinkDelay (see 14.16.20) is FALSE, computeMeanLinkDelay is initialized to the value of initialComputeMeanLinkDelay.

The currentComputeMeanLinkDelay specifies the current value of computeMeanLinkDelay. The default value of initialComputeMeanLinkDelay is TRUE. The initialComputeMeanLinkDelay and currentComputeMeanLinkDelay are per-port attributes.

When useMgtSettableComputeMeanLinkDelay is TRUE, currentComputeMeanLinkDelay is set equal to mgtSettableComputeMeanLinkDelay (see 14.16.21), and initialComputeMeanLinkDelay is ignored.

NOTE—If useMgtSettableComputeMeanLinkDelay is FALSE, the value of initialComputeMeanLinkDelay determines whether meanLinkDelay is computed by the peer delay mechanism when the port is initialized. The value of computeMeanLinkDelay can be changed, e.g., if the port receives a Signaling message that carries a message interval request TLV (see 10.6.4.3) and the current value is stored in currentComputeMeanLinkDelay.