P802.1ASdm/D2.0
February 1, 2024
(Amendment to IEEE Std 802.1AS™-2020, as modified by IEEE 802.1AS-2020/Cor-1-2021 and amended by P802.1ASdr, P802.1ASdr, and P802.1ASdm, in that order)

Draft Standard for Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications

Amendment: Hot Standby and Clock Drift Error Reduction

Prepared by the
Time-Sensitive Networking Task Group of IEEE 802.1

Sponsor
LAN/MAN Standards Committee
of the
IEEE Computer Society

IEEE Standards Activities Department
445 Hoes Lane
Piscataway, NJ 08854, USA

Abstract: This amendment to IEEE Std 802.1AS™-2020 specifies hot standby, and addresses errors and omissions in the description of existing functionality.

Keywords: hot standby, best timeTransmitter, frequency offset, Grandmaster Clock, Grandmaster PTP Instance, PTP End Instance, PTP Relay Instance, IEEE 802.1AS™, phase offset, synchronization, syntonization, time-aware system

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-X-XXX-XXX-X STDXXXXX
Print: ISBN 978-X-XXX-XXX-X STDPDXXXXX

Important Notices and Disclaimers Concerning IEEE Standards Documents

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

Participants

<<The following lists will be updated in the usual way prior to publication>>

At the time this standard was completed, the IEEE 802.1 working group had the following membership:

Glenn Parsons, *Chair*
Jessy Rouyer, *Vice Chair*
János Farkas, *TSN Task Group Chair*
Geoffrey Garner, *Editor*

SeoYoung Baek
Shenghua Bao
Jens Bierschenk
Steinar Bjornstad
Christian Boiger
Paul Bottorff
David Chen
Feng Chen
Weiying Cheng
Rodney Cummings
János Farkas
Norman Finn
Geoffrey Garner
Eric W. Gray
Craig Gunther
Marina Gutierrez
Stephen Haddock
Mark Hantel
Patrick Heffernan
Marc Holness
Lu Huang
Tony Jeffree
Michael Johas Teener
Hal Keen
Stephan Kehrer
Philippe Klein
Jouni Korhonen
Yizhou Li
Christophe Mangin
Tom McBeath
James McIntosh
Tero Mustala
Hiroki Nakano
Bob Noseworthy
Donald R. Pannell
Walter Pienciak
Michael Potts
Karen Randall
Maximilian Riegel
Dan Romascanu
Jessy V. Rouyer
Eero Ryytty
Soheil Samii
Behcet Sarikaya
Frank Schewe
Johannes Specht
Wilfried Steiner
Patricia Thaler
Paul Unbehagen
Hao Wang
Karl Weber
Brian Weis
Jordon Woods
Nader Zein
Helge Zinner
Juan Carlos Zuniga

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

A.N. Other

<<The above lists will be updated in the usual way prior to publication>>

Introduction

This introduction is not part of IEEE Std 802.1ASdm™-20xx, IEEE Standard for Local and metropolitan area networks—Timing and Synchronization for Time-Sensitive Applications—Amendment: Hot Standby

The first edition of IEEE Std 802.1AS was published in 2011. A first corrigendum, IEEE Std 802.1AS™-2011/Cor1-2013, provided technical and editorial corrections. A second corrigendum, IEEE Std 802.1AS™-2011/Cor2-2015 provided additional technical and editorial corrections.

The second edition, IEEE Std 802.1AS-2020, added support for multiple gPTP domains, Common Mean Link Delay Service, external port configuration, and Fine Timing Measurement for 802.11 transport. Backward compatibility with IEEE Std 802.1AS-2011 was maintained. A corrigendum, IEEE Std 802.1AS™-2020/Cor1-2021, provides technical and editorial corrections.

This amendment to IEEE Std 802.1AS™-2020 specifies hot standby, and addresses errors and omissions in the description of existing functionality. Hot standby guards against the failure of a single Grandmaster PTP Instance or the failure of communication from that Grandmaster PTP Instance to a Clock Target.

Table of Contents

3. Definitions 14

4. Acronyms and abbreviations 15

5. Conformance 18

7. Time synchronization model for a packet network 21

8. IEEE 802.1AS concepts and terminology 30

9. Application Interfaces 32

10. Media-independent layer specification 37

11. Media-dependent layer specification for full-duplex point-to-point links 85

13. Media-dependent layer specification for interface to IEEE 802.3 Ethernet passive optical network link
117

14. Timing and synchronization management 118

16. Media-dependent layer specification for CSN 131

17. YANG Data Model 132

18. Hot Standby 148

A.5 Major capabilities 160
A.7 Minimal time-aware system 161
A.8 Signaling 162
A.9 Best timeTransmitter clock 162
A.10 Grandmaster-capable PTP Instance 162
A.11 Media-independent timeTransmitter 163
A.13 Media-dependent, full-duplex point-to-point link 163
A.19 Remote management 164
F.3 PTP attribute values 165
G.3 Measurement procedure 166

List of Figures

Figure 7-3—Time-aware network example for multiple gPTP domains 23
Figure 7-4—Time-aware network example for synchronization path redundancy, with one clock source providing time to two domains 25
Figure 7-5—Time-aware network example for GM redundancy with one primary GM and one secondary GM, which are separated in two gPTP domains... 26
Figure 7-6—Time-aware network example for hot standby with both GM and partial path redundancy 28
Figure 7-7—Model for PTP Instance and its interfaces to higher-layer applicationsmodel 29
Figure 10-2—Time-synchronization state machines—overview and interrelationships.............................. 38
Figure 10-6—ClockTimeTransmitterSyncOffset state machine 46
Figure 10-12—External port configuration state machines—overview and interrelationships 49
Figure 10-16—PortAnnounceInformationExt state machine 61
Figure 10-17—PortStateSettingExt state machine 64
Figure 10-18—PortAnnounceTransmit state machine 65
Figure 10-19—AnnounceIntervalSetting state machine.. 68
Figure 10-20—SyncIntervalSetting state machine 70
Figure 10-21—GptpCapableTransmit state machine 73
Figure 10-22—GptpCapableReceive state machine... 74
Figure 10-23—GptpCapableIntervalSetting state machine 77
Figure 11-4—Detail of MD entity time-synchronization state machines for full-duplex point-to-point links 87
Figure 11-5—Peer-to-peer delay mechanism state machines—overview and interrelationships 88
Figure 11-6—MDSyncReceiveSM state machine... 95
Figure 11-7—MDSyncSendSM state machine... 98
Figure 11-9—MDPdelayReq state machine 106
Figure 11-11—LinkDelayIntervalSetting state machine 110
Figure 17-1—Overview of YANG tree 134
Figure 17-2—PTP Instance detail... 135
Figure 17-3—PTP Port detail 136
Figure 17-4—Common services detail 137
Figure 18-1—Model for hot standby entity in a time-aware system, and its interfaces to higher-layer applications.. 148
Figure 18-2—PtpInstanceSyncStatus State Machine 152
Figure 18-3—HotStandbySystem State Diagram.. 158
Figure 18-4—PrimarySecondaryOffset State Diagram... 159

List of Tables

Table 10-1 Summary of scope of global variables used by
time synchronization state machines (see 10.2.4 and 10.2.5) 40
Table 10-3 Summary of scope of global variables used by
best timeTransmitter clock selection, external port configuration, and announce interval setting state machines (see and 10.3.10) 50
Table 10-9 Values of flag bits 79
Table 10-10 messageTypeSpecific semantics 80
Table 10-14 Interpretation of special values of logGptpCapableMessageInterval 81
Table 10-13 Signaling message fields 81
Table 11-1 Value of meanLinkDelayThresh for various links 89
Table 11-2 Summary of scope of global variables used by
time synchronization state machines (see 10.2.4 and 10.2.5) 90
Table 11-6 Value of correctionField 111
Table 11-8 Sync message fields if twoStep flag is TRUE 112
Table 11-9 Sync message fields if twoStep flag is FALSE 112
Table 11-12 Drift_Tracking TLV 113
Table 11-10 Follow_Up message fields 113
Table 14-3 parentDS table 121
Table 14-8 driftTrackingDS table 123
Table 14-7 ptpInstanceSyncDS table 123
Table 14-9 delayMechanism enumeration 124
Table 14-10 portDS table 125
Table 14-12 portStatisticsDS table 126
Table 14-15 asymmetryMeasurementModeDS table 127
Table 14-18 cmldsLinkPortDS table 127
Table 14-19 cmldsLinkPortStatisticsDS table 128
Table 14-21 hotStandbySystemDescriptionDS table 130
Table 14-20 hotStandbySystemDS table 130
Table 17-1 Summary of the YANG modules 138

Draft IEEE Standard for Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications

Amendment: Hot Standby and Clock Drift Error Reduction

[This amendment is based on IEEE Std 802.1AS™-2020, as modified by IEEE Std 802.AS-2020/Cor-1-2021 and amended by P802.1ASdr, P802.1ASdn, and P802.1ASdm, in that order.]

NOTE—The editing instructions contained in this amendment define how to merge the material contained therein into the existing base standard and its amendments to form the comprehensive standard.

The editing instructions are shown in ***bold italic***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Deletions and insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this NOTE will not be carried over into future editions because the changes will be incorporated into the base standard.[1]

[1]Notes in text, tables, and figures are given for information only, and do not contain requirements needed to implement the standard.

3. Definitions

Insert the following definitions in Clause 3, and renumber the definitions as appropriate:

3.4 default: The value of an implementation parameter or capability selection, in the absence of an overriding profile standard requirement or explicit configuration.

3.7 epoch: The origin of a timescale.

Change 3.14 as follows:

3.14 Grandmaster PTP Instance: A PTP Instance containing the Grandmaster Clock.

NOTE—None of the PTP Ports of a Grandmaster PTP Instance is in the TimeReceiverPort state.

Change 3.23 as follows:

3.23 PTP Link: Within a domain, a network segment between two PTP Ports using the peer-to-peer delay mechanism of IEEE Std 802.1AS. The peer-to-peer delay mechanism is designed to measure the propagation time over such a link.

NOTE—A PTP Link between PTP Ports of PTP Instances is also a gPTP ~~C~~communication ~~P~~path (see 3.11).

Insert the following definitions in Clause 3, and renumber the definitions as appropriate:

3.24 profile standard: A standard specifying the capabilities, options, and parameter values of one or more base standards for use in a specific target environment or application.

3.28 sdoId: An attribute that is the primary mechanism for providing isolation of PTP Instances operating under a PTP Profile specified by one Qualified Standards Development Organization (QSDO) from PTP Instances operating under a PTP Profile specified by a different QSDO (see 7.1.3 and 7.1.4 of IEEE Std 1588-2019). The value of 1 for the majorSdoId is restricted to the PTP Profiles developed by the IEEE 802.1 Working Group (see 16.5 of IEEE Std 1588-2019).

3.33 timescale: A measure of elapsed time since an epoch.

Change Clause 4 as follows:

4. Acronyms and abbreviations

Ack	acknowledgment
ADEV	Allan deviation
ARB	arbitrary
BC	Boundary Clock
BMC	best master clock
BMCA	best master clock algorithm
CID	Company identification (allocated by the IEEE)
CMLDS	Common Mean Link Delay Service
CSN	coordinated shared network
CTC	channel time clock
EPON	IEEE 802.3™ Ethernet Passive Optical Network, as specified in IEEE Std 802.3-2018
FTM	Fine Timing Measurement
G.hn	ITU-T G.9960 and ITU-T G.9961
GM	Grandmaster
GMT	Greenwich mean time
GNSS	global navigation satellite system
GPS	global positioning (satellite) system
gPTP	generalized precision time protocol (IEEE Std 802.1AS)
IERS	International Earth Rotation and Reference Systems Service
IP	Internet Protocol
ISS	Internal Sublayer Service
LAN	local area network
LCI	location configuration information
LLC	logical link control

MAC	media access control
MACsec	media access control security
MIB	Management Information Base
MLME	IEEE 802.11™ MAC layer management entity
MPCP	IEEE 802.3 multipoint control protocol
MPCPDU	IEEE 802.3 MPCP data unit
MD	media-dependent
NMS	Network Management System
NTP	network time protocol[2]
OLT	IEEE 802.3 optical line terminal
ONU	IEEE 802.3 optical network unit
OSSP	organization-specific slow protocol
OUI	Organizationally Unique Identifier
P2P	peer-to-peer
PICS	Protocol Implementation Conformance Statement
POSIX®	portable operating system interface (see ISO/IEC 9945:2003 [B17][3])
PTP	IEEE 1588 precision time protocol
PTPDEV	PTP deviation
QSDO	Qualified Standards Development Organization
RTT	round-trip time
SI	international system of units
SMI	Structure of Management Information
SMIv2	Structure of Management Information version 2
SNMP	Simple Network Management Protocol
STA	station
TAI	International Atomic Time

[2] Information available at https://www.ietf.org/rfc/rfc1305.txt.
[3] The numbers in brackets correspond to the numbers in the bibliography in Annex G.

TC	Transparent Clock
TDEV	time deviation
TDM	time division multiplexing
TDMA	time division multiple access
TLV	type, length, value
TM	Timing Measurement
UCT	unconditional transfer
UTC	Coordinated Universal Time
VLAN	virtual local area network
WLAN	wireless local area network

5. Conformance

Change 5.4.1 as follows:

5.4.1 ~~Summary of~~PTP Instance requirements

An implementation of a PTP Instance shall:

a) Implement the generalized precision time protocol (gPTP) requirements specified in Clause 8.

b) Support the requirements for time-synchronization state machines (10.1.2, 10.2.1, 10.2.2, 10.2.3, 10.2.4, 10.2.5, and 10.2.6).

c) Support at least one PTP Port.

d) On each supported PTP Port, implement the PortSyncSyncReceive state machine (10.2.8).

e) Implement the ClockTimeReceiverSync state machine (10.2.13).

f) ~~Support the following best timeTransmitter clock algorithm (BTCA) requirements:~~

   1) ~~Implement the BTCA (10.3.2, 10.3.3, 10.3.4, 10.3.5, 10.3.6, 10.3.8, and 10.3.10).~~

   2) ~~For domain 0, implement specifications for externalPortConfigurationEnabled value of FALSE (10.3.1).~~

   3) ~~Implement the PortAnnounceReceive state machine (10.3.11).~~

   4) ~~Implement the PortAnnounceInformation state machine (10.3.12).~~

   5) ~~Implement the PortStateSelection state machine (10.3.13).~~

   6) ~~Have the BTCA as the default mode of operation, with externalPortConfiguration FALSE, on domain 0.~~

   7) ~~Implement at least one of the possibilities for externalPortConfigurationEnabled (i.e., FALSE, meaning the BTCA is used, and TRUE, meaning external port configuration is used) on domains other than domain 0.~~

g) Implement the SiteSyncSync state machine (10.2.7).

h) Implement the state machines related to signaling gPTP capability (~~10.4~~10.4.1, 10.4.2, 10.4.3).

NOTE 1—5.4.1 h) does not include the GptpCapableIntervalSetting state machine (10.4.4)

NOTE 2—The GptpCapableTransmit and GptpCapableReceive state machines are required; however, they can be disabled by setting the managed object gptpCapableStateMachinesEnabled (see 14.8.55) to FALSE.

i) For receipt of all messages and for transmission of all messages except Announce (see 10.6.3) ~~and Signaling (see 10.6.4)~~, support the message requirements as specified in 10.5, 10.6, and 10.7.

NOTE 2—The support of Signaling messages is mandatory [5.4.1 i)] because the state machines related to the signaling of gPTP capability are mandatory [5.4.1 h)].

j) Support the performance requirements in B.1 and B.2.4.

5.4.2 PTP Instance options

Change 5.4.2 c) as follows:

c) Support the following for Grandmaster PTP Instance capability:

   1) Support the media-independent timeTransmitter capability specified in item b) of 5.4.2.

   2) Support the requirements for a grandmaster-capable PTP Instance (10.1.3).

   3) Implement the ClockTimeTransmitterSyncSend state machine (10.2.9).

4) Implement the Clock~~TimeTransmitter~~SyncOffset state machine (10.2.10).
5) Implement the ClockTimeTransmitterSyncReceive state machine (10.2.11).

Delete 5.4.2 e), and renumber subsequent list items as appropriate.

Insert the following after 5.4.2 h), and renumber subsequent list items as appropriate:

i) Implement the GptpCapableIntervalSetting state machine.

Change 5.4.2 k) as follows:

k) Support the use of a remote management protocol. A PTP Instance that claims to support remote management shall:
   1) State which remote management protocol standard(s) or specification(s) are supported (see A.19).
   2) State which standard(s) or specification(s) for managed object definitions and encodings are supported for use by the remote management protocol (see A.19).
   3) If the Simple Network Management Protocol (SNMP) is supported as a remote management protocol, support the managed object definitions specified as Structure of Management Information version 2 (SMIv2) Management Information Base (MIB) modules in Clause 15.
   4) If YANG is supported with a remote management protocol, support the YANG data model ieee802-dot1as-ptp in Clause 17.
   5) If YANG is supported with a remote management protocol, and if hot standby is supported, support the YANG data model ieee802-dot1as-hs in Clause 17.

Insert the following after 5.4.2 l), and renumber subsequent list items as appropriate:

m) Implement hot standby as specified in Clause 18, 14.8, 9.3.3.4, 9.4.3.4, 9.5.3.4, and 9.6.2.6.
n) Implement the Drift_Tracking TLV as specified in 10.2.4.25, 10.2.4.26, 10.2.4.27, 11.2.14, 11.2.15, and 11.4.4.

Insert the following new subclause after 5.4.2, and renumber subsequent subclauses as appropriate:

5.4.3 PTP Instance defaults and recommendations

This standard identifies specific capabilities and parameter values as defaults. If these defaults are not further qualified by reference to the use of other capabilities or by profile standards, the defaults are intended to facilitate deployment and to provide interoperability in target environments and applications with time-aware system implementations conformant to this standard (including its 2011 edition). Mandatory requirements that this standard does not identify as defaults, or call out in some other way, are not subsetted or modified by profile standards.

Changes to capability support or parameter defaults arising from conformance to a profile standard should be identified in the PICS for that profile standard. Changes to defaults by pre-configuration of parameter values by the supplier of a protocol implementation should be identified by Additional Information (see A.3.1) in a completed PICS for this standard.

An implementation of a PTP instance shall, by default, support the following best timeTransmitter clock algorithm (BTCA) requirements:

a) Implement the BTCA (10.3.2, 10.3.3, 10.3.4, 10.3.5, 10.3.6, 10.3.8, and 10.3.10).
b) For domain 0, implement specifications for externalPortConfigurationEnabled value of FALSE (10.3.1).

c) Implement the PortAnnounceReceive state machine (10.3.11).
d) Implement the PortAnnounceInformation state machine (10.3.12).
e) Implement the PortStateSelection state machine (10.3.13).
f) Have the BTCA as the default mode of operation, with externalPortConfigurationEnabled FALSE, on domain 0.
g) Implement at least one of the possibilities for externalPortConfigurationEnabled (i.e., FALSE, meaning the BTCA is used, and TRUE, meaning external port configuration is used) on domains other than domain 0.

Change 5.4.4 as follows:

5.4.4 PTP Relay Instance requirements

An implementation of a PTP Relay Instance shall:

a) Support more than one PTP Port.
b) ~~Support the PTP Instance requirements specified in 5.4.~~
c) Support the media-independent timeTransmitter capability specified in item b) of 5.4.2.

5.5 MAC-specific timing and synchronization methods for full-duplex IEEE 802.3 links

Change 5.5 as follows:

An implementation of a time-aware system with IEEE 802.3 media access control (MAC) services to physical ports shall:

a) Support full-duplex operation, as specified in 4.2 and Annex 4A of IEEE Std 802.3-2018.
b) Support the requirements as specified in Clause 11.
c) ~~Implement the SyncIntervalSetting state machine (10.3.18).~~
d) Support two-step capability on receive as specified in 11.2.14.
e) Support two-step capability on transmit as specified in 11.2.15.

An implementation of a PTP Instance with IEEE 802.3 MAC services to physical ports may:

f) Support asymmetry measurement mode as specified in 10.3.12, 10.3.13, 10.3.16, 11.2.14, 11.2.15, 11.2.19, and 14.8.45.
g) Support one-step capability on receive as specified in 11.2.14.
h) Support one-step capability on transmit as specified in 11.2.15.
i) Support the OneStepTxOperSetting state machine specified in 11.2.16.
j) Support propagation delay averaging, as specified in 11.2.19.3.4.
k) Implement the LinkDelayIntervalSetting state machine (11.2.21).

7. Time synchronization model for a packet network

7.2 Architecture of a time-aware network

Change 7.2.1 as follows:

7.2.1 General

A time-aware network consists of a number of interconnected time-aware systems that support the gPTP defined within this standard. These time-aware systems can be any networking device, including, for example, bridges, routers, and end stations. A set of time-aware systems that are interconnected by gPTP-capable network elements is called a *gPTP network*. Each instance of gPTP that the time-aware systems support is in one *gPTP domain*, and the instances of gPTP are said to be part of that gPTP domain. A time-aware system can support, and therefore be part of, more than one gPTP domain. The entity of a single time-aware system that executes gPTP in one gPTP domain is called a *PTP Instance*. A time-aware system can contain multiple PTP Instances, which are each associated with a different gPTP domain. There are two types of PTP Instances:

a) PTP End Instance, which, if not a Grandmaster PTP Instance, is a recipient of time information, and
b) PTP Relay Instance, which, if not a Grandmaster PTP Instance, receives time information from the Grandmaster PTP Instance (perhaps indirectly through other PTP Relay Instances), applies corrections to compensate for delays in the local area network (LAN) and the PTP Relay Instance itself, and retransmits the corrected information.

This standard defines mechanisms for delay measurements using standard-based procedures for the following:

c) IEEE 802.3 Ethernet using full-duplex point-to-point links (11)
d) IEEE 802.3 EPON links (Clause 13)
e) IEEE 802.11 wireless (Clause 12)
f) Generic coordinated shared networks (CSNs, e.g., MoCA and G.hn) (Clause 16)

This standard specifies time distribution mechanisms that are tolerant to some faults if the network paths are redundant. One of these time distribution mechanisms updates the distribution path in reaction to changes, whereas the other one relies on redundant PTP instances on top of the redundant network paths.

7.2.2 Time-aware network consisting of a single gPTP domain

Change the second paragraph of 7.2.2 as follows:

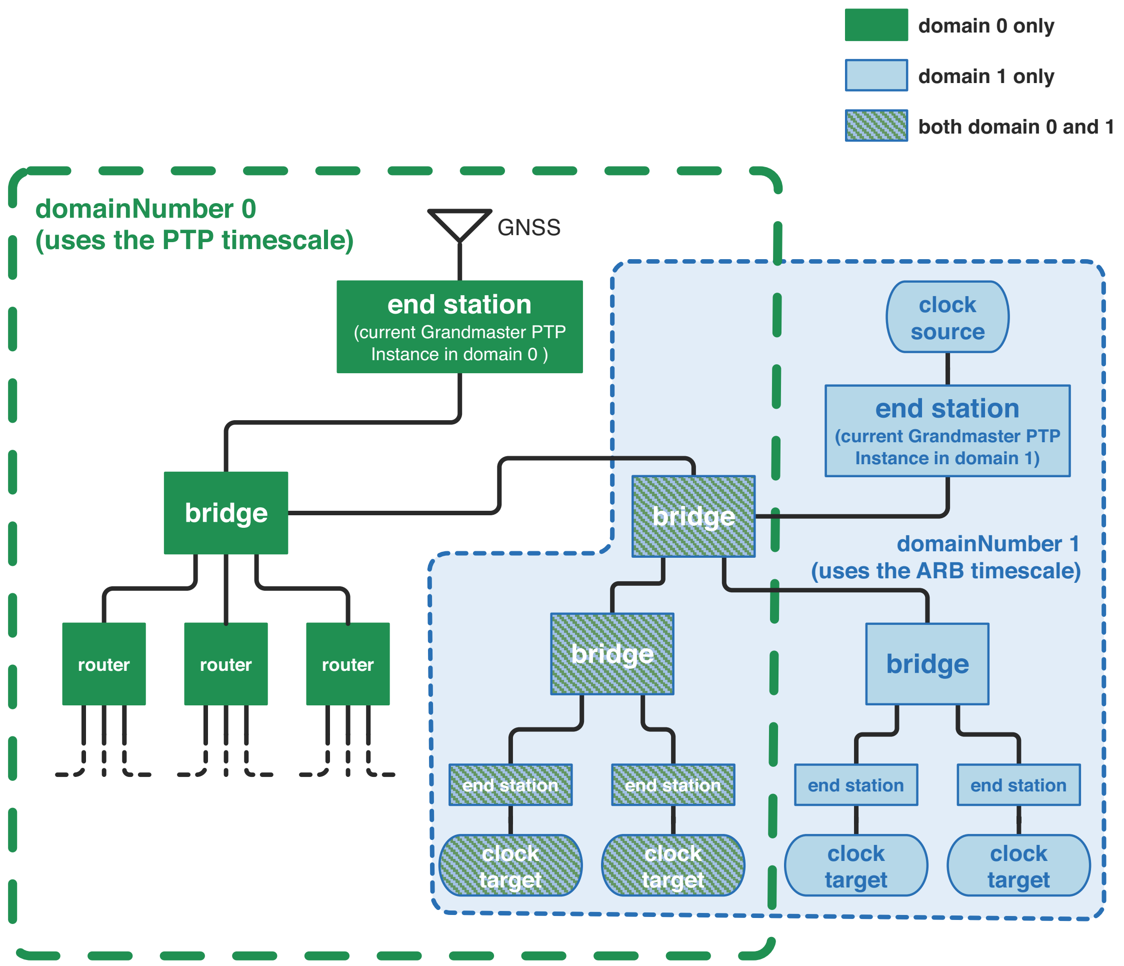
Any PTP Instance with clock sourcing capabilities can be a potential Grandmaster PTP Instance, and a selection method (the *best timeTransmitter clock algorithm*, or BTCA) ensures that all of the PTP Instances in ~~a~~ the gPTP domain use the same Grandmaster PTP Instance.[4] The BTCA is largely identical to that used in
IEEE Std 1588-2019, but somewhat simplified. In Figure 7-1 the BTCA process has resulted in the Grandmaster PTP Instance being on the network backbone. If, however, the access network fails, the systems on a local network automatically switch over to one of the potential Grandmaster PTP Instances on the local network that is as least as “good” as any other. For example, in Figure 7-2, the access network link has failed, and a potential Grandmaster PTP Instance that has a GNSS reference source has become the

[4] There are, however, short periods during network reconfiguration when more than one Grandmaster PTP Instance might be active while the BTCA process is taking place.

active Grandmaster PTP Instance. As a result, now two gPTP domains exist where there used to be one. ~~Finally, note that when a time-aware system supports more than one domain, one of the domains supported must be domain 0 for backward compatibility with the 2011 edition of this standard, though domain 0 is not necessarily active in a time-aware system.~~

7.2.3 Time-aware network consisting of multiple gPTP domains

Replace Figure 7-3 with the following:



NOTE 1—All the “bridges” and “routers" in this figure are examples of time-aware systems that contain at least one PTP Relay Instance, and the end stations are time-aware systems that contain at least one PTP End Instance.
The PTP Links in this figure can use any of the media specified in this standard.

NOTE 2 —This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that “domain number” is replaced by “domainNumber” in two places, ~~and~~ the sentence indicating that the PTP links in the figure can use any of the media specified in this standard is added, and the text at the top of the figure is made a NOTE to this figure.

Figure 7-3—Time-aware network example for multiple gPTP domains

Change 7.2.4 as follows:

7.2.4 ~~Time-aware networks with dundant Grandmaster PTP Instances and/or redundant paths~~Time-aware networks using BTCA

7.2.4.1 ~~General~~

~~Redundancy has many levels of sophistication, performance, and cost. Therefore, the appropriate level and/or amount of redundancy required in a time-aware network can be very different for each application. Nonetheless, all solutions for redundancy consist of a detection component, a correction component, and an action component. The detection component detects that something is not working correctly. The correction component determines the appropriate corrective action. The action component performs the required action(s) to fix the detected problem.~~

7.2.4.2 ~~Redundancy specified in this standard (BTCA)~~

This standard provides a basic level of redundancy as follows:

- A detection component that triggers when no Sync or Announce messages are received for a defined period of time.~~the current Grandmaster PTP Instance stops working (i.e., loss of Sync messages and Announce messages for a period of time) or if the link to the Grandmaster PTP Instance goes down (i.e., immediate loss of Sync messages and Announce messages).~~
- A correction component that triggers the Best TimeTransmitter Clock Algorithm (BTCA) and the sending of Announce messages so that a new Grandmaster PTP Instance can be elected.
- An action component, where the winning Grandmaster PTP Instance starts sending Announce messages and Sync messages ~~and all the PTP Instances listen to this new Grandmaster PTP Instance~~.

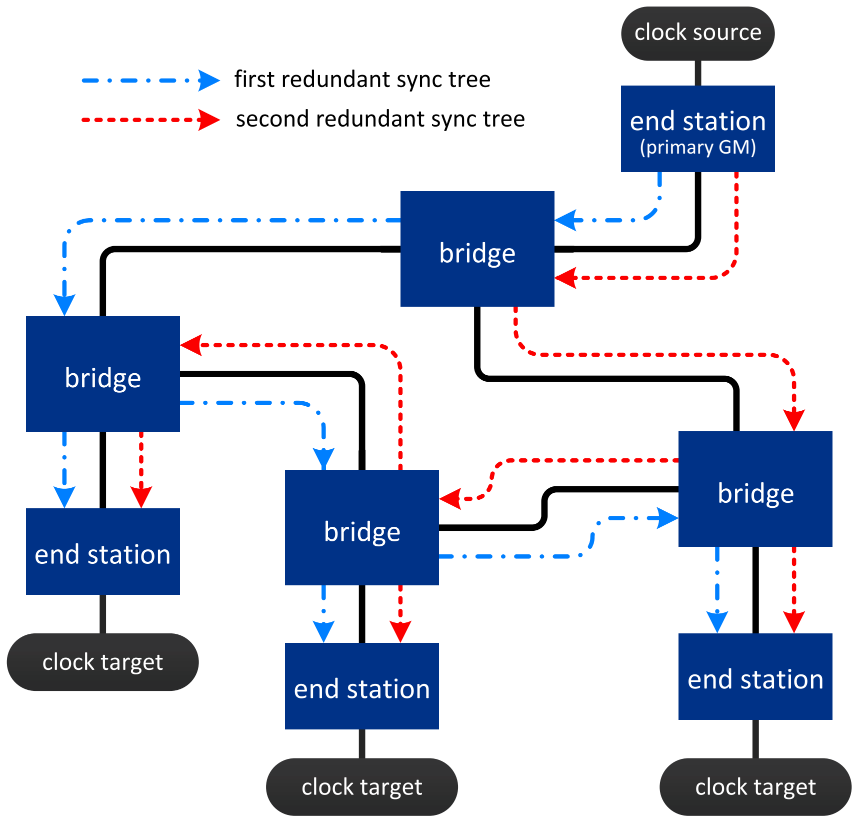
7.2.4.3 ~~Redundacy not fully specified in this standard~~

In addition to providing the basic level of redundancy, this standard provides the ability to support more sophisticated network configurations that provide additional levels of Grandmaster PTP Instance and clock path redundancy. Figure 7-4 ~~through~~and ~~Figure 7-6~~Figure 7-5 are examples of such networks that provide these additional levels of redundancy as a way to deal with these failures. The information necessary to implement and configure these network configurations is contained in this standard.

In order to take advantage of these failure correction configurations, new types of fault detection are required. The category of fault detection where a Grandmaster PTP Instance completely fails and stops sending clock information is supported as mentioned above.

Other types of faults involve instability of the Grandmaster Clock, such as time glitches, excess jitter, or wander, or various other impairments that could occur in the Grandmaster Clock. Techniques for identifying these types of failures, and the appropriate correction necessary, are not specified in this standard. However, if other techniques or standards are used for detection and correction of these types of failures, this standard provides the means to recover from these errors.

Figure 7-4 shows an example network realizing two redundant synchronization trees from a single GM, with each synchronization tree in a different gPTP domain (there are a total of two gPTP domains).



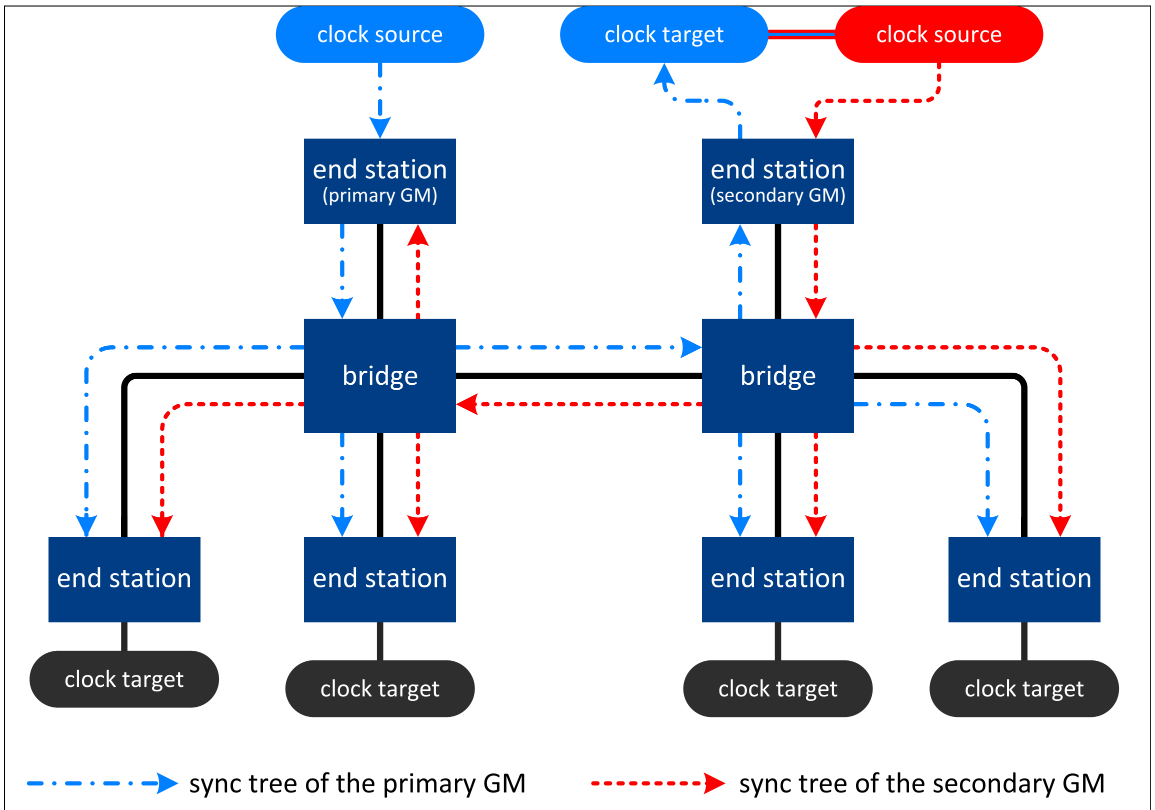
NOTE 1—The methods used for merging the redundant Sync messages received at each end station are not specified in this standard.

NOTE 2—All the "bridges" in this figure are examples of time-aware systems that contain PTP Relay Instances, and the end stations are examples of time-aware systems that contain PTP End Instances.

NOTE 3—GM denotes Grandmaster PTP Instance

Figure 7-4—Time-aware network example for synchronization path redundancy, with one clock source providing time to two domains

Figure 7-5 shows an example network with two redundant GMs, one as primary GM and the other as secondary GM, where each GM has one of the two redundant synchronization trees originating from it. This example supports hot-standby operating mode. In this mode, the secondary GM has to be synchronized to the primary GM, because it is part of the synchronization tree of the primary GM as shown in the figure.



NOTE 1—The methods used for merging the redundant Sync messages received at each end station are not specified in this standard.

NOTE 2—All the “bridges” in this figure are examples of time-aware systems that contain PTP Relay Instances, and the end stations are examples of time-aware systems that contain PTP End Instances.

NOTE 3—GM denotes Grandmaster PTP Instance

Figure 7-5—Time-aware network example for GM redundancy with one primary GM and one secondary GM, which are separated in two gPTP domains

~~Figure 7-6 shows another example network, which involves ring topology, using the redundancy features of both Figure 7-4 and Figure 7-5.~~

~~For the techniques shown in the examples of Figure 7-4, Figure 7-5, and Figure 7-6, the detection component, correction component, and action component are not fully specified in this standard.~~

Delete Figure 7-6.

Add the following new subclause after 7.2.4, and renumber figures as necessary:

7.2.5 Time-aware network with hot standby

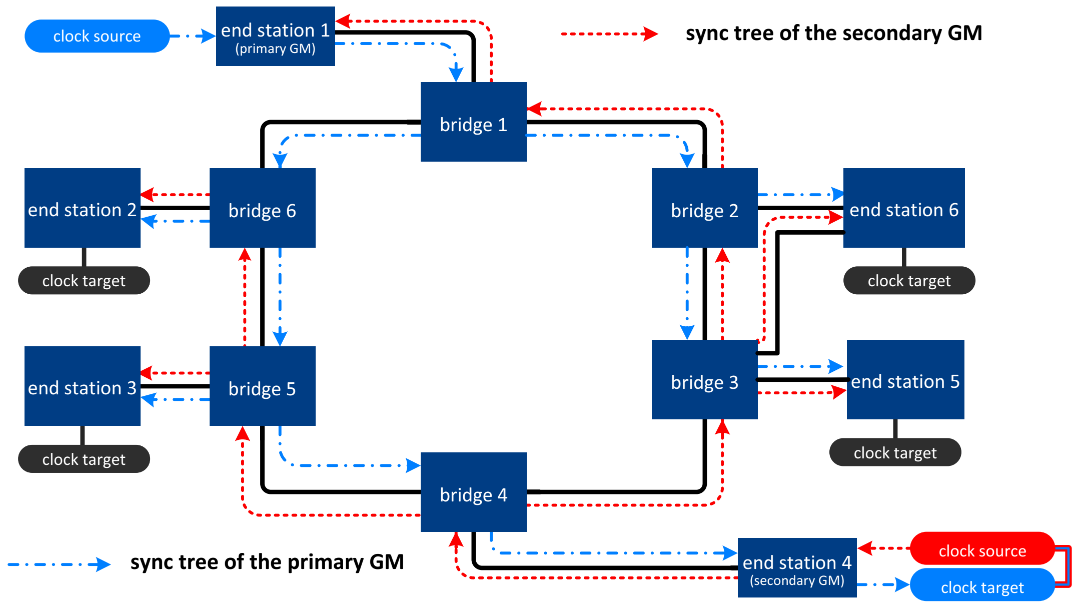
Figure 7-6 shows an example of a time-aware network with hot standby (see Clause 18). The network has GM redundancy and path redundancy to all the bridges and end stations 1, 4, and 6 (path redundancy for end stations 2, 3, and 5 is possible, but is not illustrated to avoid clutter in the figure). In addition, for simplicity the optional split functionality (see 18.5.3.4) is assumed to be disabled.

End station 1 is the primary GM (see Clause 18), and its synchronization spanning tree is illustrated by the blue arrows. End station 4 is the secondary GM, and its synchronization spanning tree is illustrated by the red arrows. Except for the links to end stations 2, 3, and 5, the network can tolerate the failure of any single link or GM and the bridges and end stations remain synchronized by the same GM. For example, if the link between bridge 3 and end station 6 fails, end station 6 is still synchronized by the blue domain via bridge 2. If the link between bridge 5 and bridge 6 fails, bridge 6 is still synchronized by the blue domain and bridge 5 is still synchronized by the red domain. If the secondary GM fails, all the bridges and end stations are still synchronized by the primary GM.

If the primary GM fails, all the bridges and end stations are still synchronized by the secondary GM. In this case, if syncLocked is TRUE the primary PTP Instance of end station 4 stops receiving Sync messages and isSynced changes to FALSE; the hotStandbySystemState of end station 4 is then change to NOT_REDUNDANT (see Clause 18). If syncLocked is FALSE, and assuming the bridges and end stations other than the GMs are not grandmaster capable, the time difference between the primary and secondary domain eventually exceeds the primarySecondaryOffsetThresh (see Clause 18) and the hotStandbySystemState of end station 4 changes to NOT_REDUNDANT. In both syncLocked cases, the secondary GM no longer takes timing from the primary domain.

If the link between bridge 4 and bridge 5 fails, all the PTP Instances except for bridge 4 and end station 4 are then synchronized by the blue domain. Bridge 4 and end station 4 are synchronized by the red domain; in addition, bridges 1, 2, and 3, and end stations 1, 4, 5, and 6 receive timing on the red domain. Since end station 4 does not receive timing on the blue domain, its hotStandbySystemState changes to NOT_REDUNDANT, and the red domain GM does not take timing from the blue domain.

.



NOTE 1—All the "bridges" in this figure are examples of time-aware systems that contain PTP Relay Instances, and the end stations are examples of time-aware systems that contain PTP End Instances.

NOTE 2—GM denotes Grandmaster PTP Instance

Figure 7-6—Time-aware network example for hot standby with both GM and partial path redundancy

7.4 PTP Instance architecture

Change the first paragraph of 7.4 and the caption of Figure 7-8 as follows:

The model of a PTP Instance and its interfaces to higher-layer applications are~~is~~ shown in Figure 7-7. The interfaces are those specified in Clause 9.

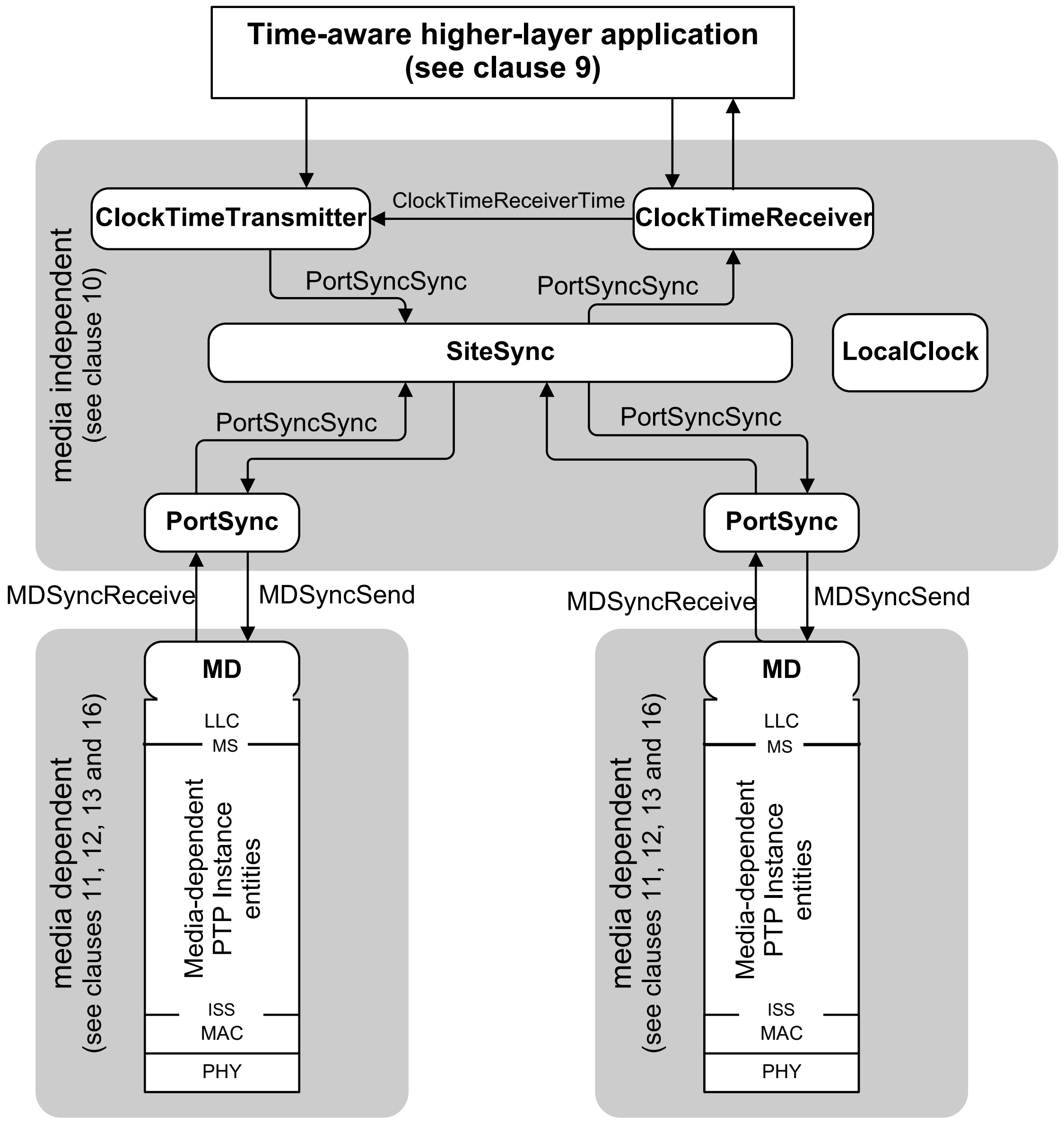

Time-aware higher-layer application (see clause 9)
ClockTimeTransmitter
ClockTimeReceiverTime
ClockTimeReceiver
media independent (see clause 10)
PortSyncSync
PortSyncSync
SiteSync
LocalClock
PortSyncSync
PortSyncSync
PortSync
PortSync
MDSyncReceive
MDSyncSend
MDSyncReceive
MDSyncSend
MD
LLC
MS
Media-dependent PTP Instance entities
ISS
MAC
PHY
media dependent (see clauses 11, 12, 13 and 16)
MD
LLC
MS
Media-dependent PTP Instance entities
ISS
MAC
PHY
media dependent (see clauses 11, 12, 13 and 16)

Figure 7-7—Model for PTP Instance and its interfaces to higher-layer applications~~model~~

8. IEEE 802.1AS concepts and terminology

Change 8.1 as follows:

8.1 gPTP domain

A gPTP domain, hereafter called simply a *domain,* consists of one or more PTP Instances and links that meet the requirements of this standard and communicate with each other as defined by the IEEE 802.1AS protocol. A gPTP domain defines the scope of gPTP message communication, state, operations, data sets, and timescale.

A domain is identified by two attributes: ~~domain number~~domainNumber and sdoId. The sdoId of a domain is a 12-bit unsigned integer. The sdoId is structured as a two-part attribute as follows:

— The most significant 4 bits are named the majorSdoId, and
— The least significant 8 bits are named the minorSdoId.

A time-aware system shall support one or more domains, each with a distinct ~~domain number~~domainNumber in the range 0 through 127. ~~A time-aware system shall support the domain whose domain number is 0, and that domain number shall not be changed to a nonzero value.~~ Unless otherwise specified in this standard, the operation of gPTP and the timescale in any given domain is independent of operation in any other domain.

The value of majorSdoId for a gPTP domain shall be 0x1. The value of minorSdoId for a gPTP domain shall be 0x00.

~~NOTE 1—The above requirements for majorSdoId and minorSdoId are for gPTP domains. The requirements for the Common Mean Link Delay Service (CMLDS) are given in 11.2.17.~~

Both the domainNumber and the sdoId are carried in the common header of all PTP messages (see 10.6.2.2).

NOTE 1—The above requirements for majorSdoId and minorSdoId are for gPTP domains. The requirements for the Common Mean Link Delay Service (CMLDS) are given in 11.2.17.

NOTE 2—In the 2011 edition of this standard, the attribute majorSdoId was named transportSpecific, and its value was specified as 0x1 in 10.5.2.2.1 of Corrigendum 1. The attribute minorSdoId did not exist in the 2011 edition, but its location in the common header was a reserved field, which was specified to be transmitted as 0 and ignored on receipt.

Unless otherwise stated, information in the remainder of this document is per domain.

NOTE 3—In steady state, all PTP Instances in a gPTP domain are traceable to a single Grandmaster PTP Instance.

Since the IEEE 802.1 Working Group has only one single unique sdoId value, the PTP Profile specified in the present standard is isolated from other PTP Profiles (see 16.5.2 of IEEE Std 1588-2019).

8.5 Ports

8.5.2 Port identity

Change 8.5.2.3 as follows:

8.5.2.3 Port number

The portNumber values for the PTP Ports on a time-aware system shall be distinct in the range ~~1, 2, …,~~ 0x1 through 0xFFFE.

The portNumber value 0 is assigned to the interface between the ClockTimeTransmitter and ClockSource entities (see 10.1 and Figure 10-1). The value 0xFFFF is reserved. The assignment of portNumber 0 to this interface helps to simplify the SiteSync and PortStateSelection state machines; this interface is not a PTP Port.

9. Application Interfaces

9.2 ClockSourceTime interface

9.2.2 ClockSourceTime.invoke function parameters

Change 9.2.2.1 as follows:

9.2.2.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the gPTP domain to which this ClockSource entity is providing time.

Change 9.2.2.3 as follows:

9.2.2.3 timeBaseIndicator (UInteger16)

The timeBaseIndicator is a ~~binary~~ value that is set by the ClockSource entity. The ClockSource entity changes the value whenever its time base changes. The ClockSource entity shall change the value of timeBaseIndicator if and only if there is a phase or frequency change.

NOTE—While the clock that supplies time to the ClockSource entity can be lost, ~~i.e., the PTP Instance can enter holdover,~~ the ClockSource entity itself is not lost. The ClockSource entity ensures that timeBaseIndicator changes if the source of time is lost.

9.3 ClockTargetEventCapture interface

9.3.2 ClockTargetEventCapture.invoke parameters

Change 9.3.2.1 as follows:

9.3.2.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the ClockTimeReceiver entity that is requested to provide the synchronized time of the signaled event.

Change 9.3.3 as follows:

9.3.3 ClockTargetEventCapture.result parameters

```
ClockTargetEventCapture.result {
        domainNumber,
        timeReceiverTimeCallback,
        gmPresent,
        isSynced,
        grandmasterIdentity
}
```

9.3.3.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the ClockTimeReceiver entity that is providing the synchronized time of the signaled event.

9.3.3.2 timeReceiverTimeCallback (ExtendedTimestamp)

The value of timeReceiverTimeCallback is the time, relative to the Grandmaster Clock, that the corresponding ClockTargetEventCapture.invoke function is invoked.

NOTE—The invocation of the ClockTargetEventCapture.invoke function and the detection of this invocation by the ClockTimeReceiver entity are simultaneous in this abstract interface.

9.3.3.3 gmPresent (Boolean)

The value of gmPresent is set equal to the value of the global variable gmPresent (see 10.2.4.13). This parameter indicates to the ClockTarget whether a Grandmaster PTP Instance is present.

9.3.3.4 isSynced (Boolean)

The value of the the variable isSynced (see 18.4.1.1). This parameter shall be present if the optional hot standby feature is implemented (see Clause 18).

9.3.3.5 grandmasterIdentity (ClockIdentity)

The value of grandmasterIdentity is the clockIdentity of the Grandmaster PTP Instance.

9.4 ClockTargetTriggerGenerate interface

9.4.2 ClockTargetTriggerGenerate.invoke parameters

Change 9.4.2.1 as follows:

9.4.2.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the ClockTimeReceiver entity that is requested to signal an event at the specified time.

Change 9.4.3 as follows:

9.4.3 ClockTargetTriggerGenerate.result parameters

```
ClockTargetTriggerGenerate.result {
        domainNumber,
        errorCondition,
        gmPresent,
        isSynced,
        grandmasterIdentity
}
```

9.4.3.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the ClockTimeReceiver entity that is triggering an event at the specified time.

9.4.3.2 errorCondition (Boolean)

A value of FALSE indicates that the ClockTargetTriggerGenerate.result function was invoked at the time, relative to the Grandmaster Clock, contained in the corresponding ClockTargetTriggerGenerate.invoke function. A value of TRUE indicates that the ClockTargetTriggerGenerate.result function could not be invoked at the synchronized time contained in the corresponding ClockTargetTriggerGenerate.invoke function.

NOTE—For example, the ClockTargetTriggerGenerate.result function is invoked with errorCondition = TRUE if the requested timeReceiverTimeCallback is a time prior to the synchronized time when the corresponding ClockTargetTriggerGenerate.invoke function is invoked. As another example, the ClockTargetTriggerGenerate.result function is invoked with errorCondition = TRUE if a discontinuity in the synchronized time causes the requested timeReceiverTimeCallback to be skipped over.

9.4.3.3 gmPresent (Boolean)

~~The value of gmPresent is set equal to the value of the global variable gmPresent (see 10.2.4.13). This parameter indicates to the ClockTarget whether a Grandmaster PTP Instance is present.~~As specified in 9.3.3.3.

9.4.3.4 isSynced (Boolean)

As specified in 9.3.3.4.

9.4.3.5 grandmasterIdentity (ClockIdentity)

As specified in 9.3.3.5.

9.5 ClockTargetClockGenerator interface

9.5.2 ClockTargetClockGenerator.invoke parameters

Change 9.5.2.1 as follows:

9.5.2.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the ClockTimeReceiver entity that is requested to deliver a periodic clock signal.

Change 9.5.3 as follows:

9.5.3 ClockTargetClockGenerator.result parameters

```
ClockTargetClockGenerator.result {
        domainNumber,
        gmPresent,
        timeReceiverTimeCallback,
        isSynced,
        grandmasterIdentity
}
```

9.5.3.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the ClockTimeReceiver entity that is delivering a periodic clock signal.

9.5.3.2 gmPresent

As specified in 9.3.3.3.

9.5.3.3 timeReceiverTimeCallback (ExtendedTimestamp)

The value of timeReceiverTimeCallback is the synchronized time of this event.

9.5.3.4 isSynced Boolean)

As specified in 9.3.3.4.

9.5.3.5 grandmasterIdentity (ClockIdentity)

As specified in 9.3.3.5.

9.6 ClockTargetPhaseDiscontinuity interface

Change 9.6.2 as follows:

9.6.2 ClockTargetPhaseDiscontinuity.result parameters

```
ClockTargetPhaseDiscontinuity.result {
            domainNumber,
            grandmasterIdentity,
            ~~gmIdentity,~~
            gmTimeBaseIndicator,
            lastGmPhaseChange,
            lastGmFreqChange,
            isSynced,
            }
```

9.6.2.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the ClockTimeReceiver entity that is providing discontinuity information.

9.6.2.2 grandmasterIdentity (ClockIdentity)

If gmPresent (see 10.2.4.13) is TRUE, the value of grandmasterIdentity is the ClockIdentity of the current Grandmaster PTP Instance. If gmPresent is FALSE, the value of grandmasterIdentity is 0x0.

9.6.2.3 gmTimeBaseIndicator (UInteger16)

The value of gmTimeBaseIndicator is the timeBaseIndicator of the current Grandmaster PTP Instance.

9.6.2.4 lastGmPhaseChange (ScaledNs)

The value of the global lastGmPhaseChange parameter (see 10.2.4.16) received from the Grandmaster PTP Instance.

9.6.2.5 lastGmFreqChange (Float64)

The value of lastGmFreqChange parameter (see 10.2.4.17) received from the Grandmaster PTP Instance.

9.6.2.6 isSynced (Boolean)

As specified in 9.3.3.4.

9.6.2.7 grandmasterIdentity (ClockIdentity)

As specified in 9.3.3.5.

10. Media-independent layer specification

10.2 Time-synchronization state machines

Change 10.2.1 and Figure 10-2 as follows:

10.2.1 Overview

The time-synchronization function in a PTP Instance is specified by a number of cooperating state machines. Figure 10-2 illustrates these state machines, their local variables, their interrelationships, and the global variables and structures used to communicate between them. The figure indicates the interaction between the state machines and the media-dependent layer and LocalClock entity.

The ClockTimeTransmitterSyncReceive, Clock~~TimeTransmitter~~SyncOffset, and ClockTimeTransmitterSyncSend state machines are optional for PTP Instances that are not grandmaster-capable (see 8.6.2.1 and 10.1.3). These state machines may be present in a PTP Instance that is not grandmaster-capable; however, any information supplied by them, via the ClockTimeTransmitterSyncSend state machine, to the SiteSyncSync state machine is not used by the SiteSyncSync state machine if the PTP Instance is not grandmaster-capable.

The media-independent layer state machines in Figure 10-2 are as follows:

a) ClockTimeTransmitterSyncReceive (one instance per PTP Instance): receives ClockSourceTime.invoke functions from the ClockSource entity and notifications of LocalClock entity ticks (see 10.2.4.18), updates timeTransmitterTime, and provides timeTransmitterTime to Clock~~TimeTransmitter~~SyncOffset and ClockTimeTransmitterSyncSend state machines.

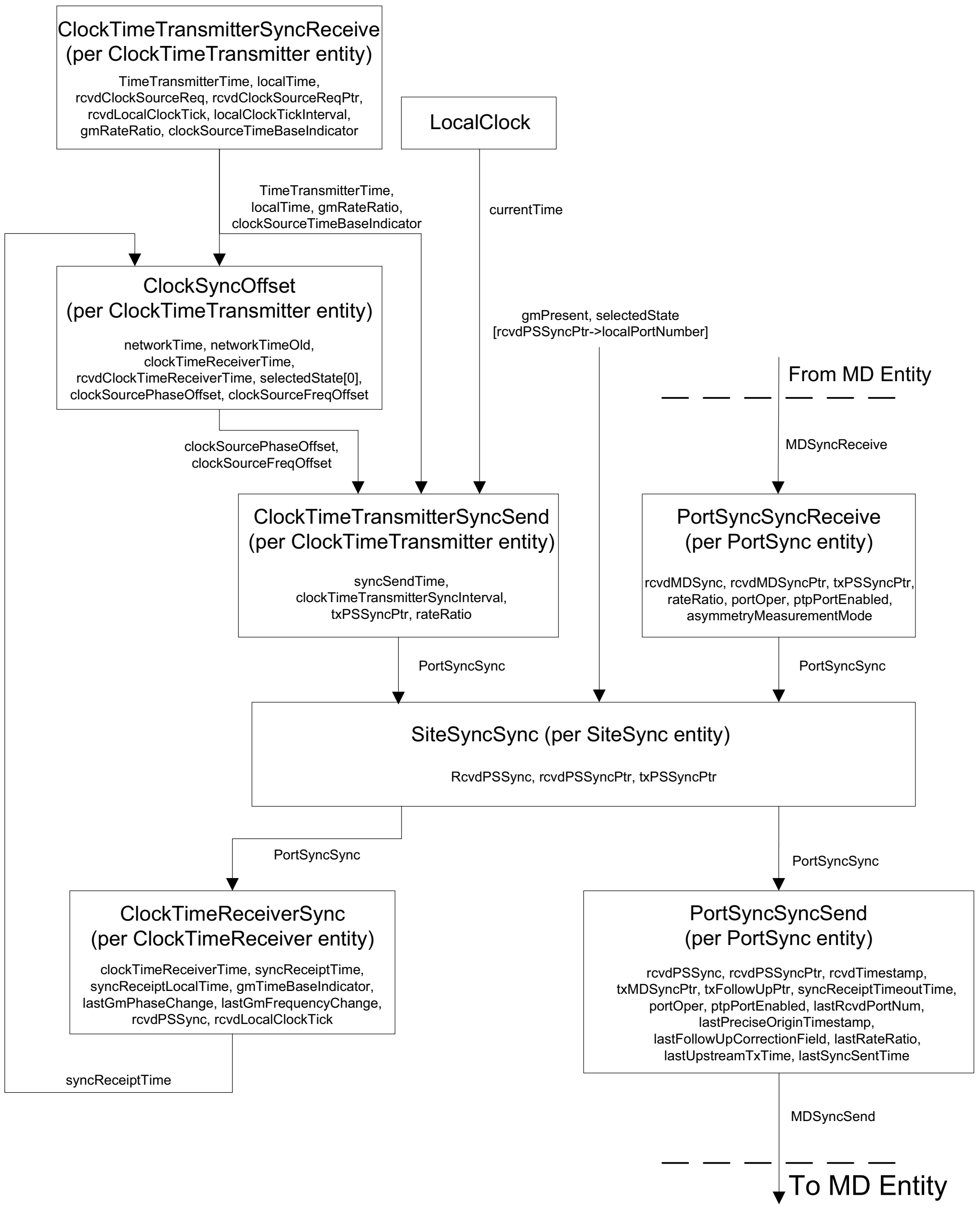
b) Clock~~TimeTransmitter~~SyncOffset (one instance per PTP Instance): receives syncReceiptTime from the ClocktTimeReceiver entity and timeTransmitterTime from the ClockTimeTransmitterSyncReceive state machine, computes phase offset and frequency offset between timeTransmitterTime and syncReceiptTime if the PTP Instance is not the Grandmaster PTP Instance, and provides the frequency and phase offsets to the ClockTimeTransmitterSyncSend state machine.

c) ClockTimeTransmitterSyncSend (one instance per PTP Instance): receives timeTransmitterTime from the ClockTimeTransmitterSyncReceive state machine, receives phase and frequency offset between timeTransmitterTime and syncReceiptTime from the Clock~~TimeTransmitter~~SyncOffset state machine, and provides timeTransmitterTime (i.e., synchronized time) and the phase and frequency offset to the SiteSync entity using a PortSyncSync structure.

d) PortSyncSyncReceive (one instance per PTP Instance, per PTP Port): receives time-synchronization information from the MD entity of the corresponding PTP Port, computes accumulated rateRatio, computes syncReceiptTimeoutTime, and sends the information to the SiteSync entity.

e) SiteSyncSync (one instance per PTP Instance): receives time-synchronization information, accumulated rateRatio, and syncReceiptTimeoutTime from the PortSync entity of the current timeReceiver port or from the ClockTimeTransmitter entity; and sends the information to the PortSync entities of all the ports and to the ClockTimeReceiver entity.

f) PortSyncSyncSend (one instance per PTP Instance, per PTP Port): receives time-synchronization information from the SiteSync entity, requests that the MD entity of the corresponding PTP Port send a time-synchronization event message, receives the syncEventEgressTimestamp for this event message from the MD entity, uses the most recent time-synchronization information received from the SiteSync entity and the timestamp to compute time-synchronization information that will be sent by the MD entity in a general message (e.g., for full-duplex IEEE 802.3 media) or a subsequent event message (e.g., for IEEE 802.11 media), and sends this latter information to the MD entity.


ClockTimeTransmitterSyncReceive
(per ClockTimeTransmitter entity)
TimeTransmitterTime, localTime, rcvdClockSourceReq, rcvdClockSourceReqPtr, rcvdLocalClockTick, localClockTickInterval, gmRateRatio, clockSourceTimeBaseIndicator
LocalClock
TimeTransmitterTime, localTime, gmRateRatio, clockSourceTimeBaseIndicator
currentTime
ClockSyncOffset
(per ClockTimeTransmitter entity)
networkTime, networkTimeOld, clockTimeReceiverTime, rcvdClockTimeReceiverTime, selectedState[0], clockSourcePhaseOffset, clockSourceFreqOffset
gmPresent, selectedState
[rcvdPSSyncPtr->localPortNumber]
From MD Entity
MDSyncReceive
clockSourcePhaseOffset, clockSourceFreqOffset
ClockTimeTransmitterSyncSend
(per ClockTimeTransmitter entity)
syncSendTime, clockTimeTransmitterSyncInterval, txPSSyncPtr, rateRatio
PortSyncSyncReceive
(per PortSync entity)
rcvdMDSync, rcvdMDSyncPtr, txPSSyncPtr, rateRatio, portOper, ptpPortEnabled, asymmetryMeasurementMode
PortSyncSync
PortSyncSync
SiteSyncSync (per SiteSync entity)
RcvdPSSync, rcvdPSSyncPtr, txPSSyncPtr
PortSyncSync
PortSyncSync
ClockTimeReceiverSync
(per ClockTimeReceiver entity)
clockTimeReceiverTime, syncReceiptTime, syncReceiptLocalTime, gmTimeBaseIndicator, lastGmPhaseChange, lastGmFrequencyChange, rcvdPSSync, rcvdLocalClockTick
PortSyncSyncSend
(per PortSync entity)
rcvdPSSync, rcvdPSSyncPtr, rcvdTimestamp, txMDSyncPtr, txFollowUpPtr, syncReceiptTimeoutTime, portOper, ptpPortEnabled, lastRcvdPortNum, lastPreciseOriginTimestamp, lastFollowUpCorrectionField, lastRateRatio, lastUpstreamTxTime, lastSyncSentTime
syncReceiptTime
MDSyncSend
To MD Entity

Notes:
a) selectedState for each port and gmPresent are set by Port State Selection state machine (see 10.3.12)
b) currentTime is a global variable that is always equal to the current time relative to the local oscillator
c) application interfaces to higher layers are not shown
d) the ClockTimeTransmitterSyncReceive, ClockTimeTransmitterSyncSend, and ClockTimeTransmitterSyncOffset state machines are optional for PTP Instances that are not grandmaster-capable.

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

g) ClockTimeReceiverSync (one instance per PTP Instance): receives time-synchronization information from the SiteSync entity; computes clockTimeReceiverTime and syncReceiptTime; sets syncReceiptLocalTime, GmTimeBaseIndicator, lastGmPhaseChange, and lastGmFreqChange; sends clockTimeReceiverTime to the ClockTimeTransmitter entity; and provides information to the ClockTarget entity (via the ClockTargetPhaseDiscontinuity interface; see 9.6) to enable that entity to determine if a phase or frequency discontinuity has occurred.

10.2.2 Data structures communicated between state machines

10.2.2.1 MDSyncSend

Change 10.2.2.1.2 as follows:

10.2.2.1.2 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the gPTP domain in which this structure is sent.

NOTE—The ~~domain number~~domainNumber member is not essential because the state machines that send and receive this structure are per domain, and each state machine implicitly knows the number of the domain in which it operates.

10.2.2.2 MDSyncReceive

Change 10.2.2.2.2 as follows:

10.2.2.2.2 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the gPTP domain in which this structure is sent.

NOTE—The ~~domain number~~domainNumber member is not essential because the state machines that send and receive this structure are per domain, and each state machine implicitly knows the number of the domain in which it operates.

10.2.2.3 PortSyncSync

Change 10.2.2.3.2 as follows:

10.2.2.3.2 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the gPTP domain in which this structure is sent.

NOTE—The ~~domain number~~domainNumber member is not essential because the state machines that send and receive this structure are per domain, and each state machine implicitly knows the number of the domain in which it operates.

10.2.3 Overview of global variables used by time synchronization state machines

Change 10.2.3 as follows:

Subclauses 10.2.4 and 10.2.5 define global variables used by time synchronization state machines whose scopes are as follows:

— Per PTP Instance (i.e., per domain)
— Per PTP Instance, per PTP Port
— Instance used by the Common Mean Link Delay Service (CMLDS) (see 11.2.17) (i.e., variable is common across all ~~LinkPorts~~Link Ports)
— Instance used by CMLDS, per ~~LinkPort~~Link Port

Table 10-1 summarizes the scope of each global variable of 10.2.4 and 10.2.5.

Table 10-1—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~~Link Ports)	Instance used by CMLDS, per ~~LinkPort~~Link Port
BEGIN	10.2.4.1	Yes	No	Yes	No
clockTimeTransmitterSyncInterval	10.2.4.2	Yes	No	No	No
clockTimeReceiverTime	10.2.4.3	Yes	No	No	No
syncReceiptTime	10.2.4.4	Yes	No	No	No
syncReceiptLocalTime	10.2.4.5	Yes	No	No	No
clockSourceFreqOffset	10.2.4.6	Yes	No	No	No
clockSourcePhaseOffset	10.2.4.7	Yes	No	No	No
clockSourceTimeBaseIndicator	10.2.4.8	Yes	No	No	No
clockSourceTimeBaseIndicatorOld	10.2.4.9	Yes	No	No	No
clockSourceLastGmPhaseChange	10.2.4.10	Yes	No	No	No
clockSourceLastGmFreqChange	10.2.4.11	Yes	No	No	No
currentTime	10.2.4.12	Yes	No	No	No
gmPresent	10.2.4.13	Yes	No	No	No
gmRateRatio	10.2.4.14	Yes	No	No	No
gmTimeBaseIndicator	10.2.4.15	Yes	No	No	No
lastGmPhaseChange	10.2.4.16	Yes	No	No	No
lastGmFreqChange	10.2.4.17	Yes	No	No	No
localClockTickInterval	10.2.4.18	Yes	No	No	No
localTime	10.2.4.19	Yes	No	No	No
selectedState	10.2.4.20	Yes	No	No	No
timeTransmitterTime	10.2.4.21	Yes	No	No	No
thisClock	10.2.4.22	Yes	No	Yes	No
parentLogSyncInterval	10.2.4.23	Yes	No	No	No
instanceEnable	10.2.4.24	Yes	No	No	No
syncReceiptTimeoutTime	10.2.4.25	Yes	No	No	No
asCapable	10.2.5.1	No	Yes	No	No
asymmetryMeasurementMode	10.2.5.2	No	Yes[1]	No	Yes
syncReceiptTimeoutTimeInterval	10.2.5.3	No	Yes	No	No
currentLogSyncInterval	10.2.5.4	No	Yes	No	No
initialLogSyncInterval	10.2.5.5	No	Yes	No	No
syncInterval	10.2.5.6	No	Yes	No	No

Table 10-1—Summary of scope of global variables used by time synchronization state machines (see 10.2.4 and 10.2.5) *(continued)*

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~~Link Ports)	Instance used by CMLDS, per ~~LinkPort~~Link Port
neighborRateRatio	10.2.5.7	No	Yes~~[1]~~	No	~~Yes~~No
meanLinkDelay	10.2.5.8	No	Yes[1]	No	Yes
delayAsymmetry	10.2.5.9	No	Yes~~[1]~~	No	Yes
computeNeighborRateRatio	10.2.5.10	No	Yes~~[1]~~	No	Yes
computeMeanLinkDelay	10.2.5.11	No	Yes[1]	No	Yes
portOper[2]	10.2.5.12	No	Yes	No	Yes
ptpPortEnabled	10.2.5.13	No	Yes	No	No
thisPort	10.2.5.14	No	Yes	No	Yes
syncLocked	10.2.5.15	No	Yes	No	No
neighborGptpCapable	10.2.5.16	No	Yes	No	No
syncSlowdown	10.2.5.17	No	Yes	No	No
oldSyncInterval	10.2.5.18	No	Yes	No	No
gPtpCapableMessageSlowdown	10.2.5.19	No	Yes	No	No
gPtpCapableMessageInterval	10.2.5.20	No	Yes	No	No
oldGptpCapableMessageInterval	10.2.5.21	No	Yes	No	No
currentLogGptpCapableMessageInterval	10.2.5.22	No	Yes	No	No
initialLogGptpCapableMessageInterval	10.2.5.23	No	Yes	No	No
syncGrandmasterIdentity	10.2.4.25	Yes	No	No	No
syncStepsRemoved	10.2.4.26	Yes	No	No	No
driftTrackingTlvSupport	10.2.4.27	Yes	No	No	No
rcvdPSSyncCSS	10.2.4.28	Yes	No	No	No
rcvdLocalClockTickCSS	10.2.4.29	Yes	No	No	No
rateRatioDrift	10.2.4.30	Yes	No	No	No

[1] The instance of this variable that is per PTP Instance, per PTP Port exists only for domain 0.
[2] There is one instance of this variable per physical port, which is accessible by all PTP Ports and ~~LinkPorts~~Link Ports associated with the physical port.

10.2.4 Per PTP Instance global variables

Insert 10.2.4.25, 10.2.4.26, 10.2.4.27, 10.2.4.28, 10.2.4.29, and 10.2.4.30, and renumber subsequent subclauses as necessary:

10.2.4.25 syncGrandmasterIdentity: the clockIdentity carried in the syncGrandmassterIdentity field of the Drift_Tracking TLV carried by the most recently received Sync message (twoStep flag FALSE) or Follow_Up message (twoStep flag TRUE). If the received Sync or Follow_Up message does not carry a Drift_Tracking TLV, syncGrandmasterIdentity is set to the null value 0xFFFF FFFF FFFF FFFF (see the section “Unassigned and NULL EUI values” of the IEEE Registration Authority tutorial “Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)” [B30]). The data type for syncGrandmasterIdentity is ClockIdentity.

10.2.4.26 syncStepsRemoved: the value of the syncStepsRemoved field of the Drift_Tracking TLV carried by the most recently received Sync message (twoStep flag FALSE) or Follow_Up message (twoStep flag TRUE). If the received Sync or Follow_Up message does not carry a Drift_Tracking TLV, syncstepsRemoved is set to 0xFFFF. The data type for syncStepsRemoved is UInteger16.

10.2.4.27 driftTrackingTlvSupport: An indicator of whether the PTP Instance supports the Drift_Tracking TLV and the feature is enabled. The value is TRUE if the Drift_Tracking TLV is supported and the managed object driftTrackingTlvSupportEnabled (see 14.9) is TRUE. The value is FALSE if the Drift_Tracking TLV is not support, or if the TLV is supported and the managed object driftTrackingTlvSupportEnabled is FALSE. The data type for driftTrackingTlvSupport is Boolean.

NOTE—The Drift_Tracking TLV is transported only on full-duplex, point-to-point links as specified in Clause 11. Even if driftTrackingTlvSupport is TRUE, the Drift_Tracking TLV is not tranported on links other than full-duplex, point-to-point links, and is not received by the PTP Ports at the other end of these links.

10.2.4.28 rcvdPSSyncCSS: A Boolean variable that is set to TRUE when a PortSyncSync structure is received from the SiteSyncSync state machine of the SiteSync entity by the ClockTimeReceiverSync state machine. This variable is reset by the ClockTimeReceiverSync state machine.

10.2.4.29 rcvdLocalClockTickCSS: A Boolean variable that is set to TRUE when the LocalClock entity updates its time. This variable is reset by the ClockTimeReceiverSync state machine.

10.2.4.30 rateRatioDrift: the value of the rateRatioDrift field of the Drift_Tracking TLV carried by the most recently received Sync message (twoStep flag FALSE) or Follow_Up message (twoStep flag TRUE). If the received Sync or Follow_Up message does not carry a Drift_Tracking TLV, rateRatioDrift is set to 0xFFFFFFFF. The data type for rateRatioDrift is Integer32.

10.2.5 Per port global variables

Change 10.2.5.2 as follows:

10.2.5.2 asymmetryMeasurementMode: A Boolean that contains the value of the managed object asymmetryMeasurementMode (see 14.8.45). For full-duplex IEEE 802.3 media, the value is TRUE if an asymmetry measurement is being performed for the link attached to this port and FALSE otherwise. For all other media, the value is FALSE. There is one instance of this variable for all the domains, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

Change 10.2.5.7 as follows:

10.2.5.7 neighborRateRatio: The measured ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this port, to the frequency of the LocalClock entity of this time-aware system. The data type for neighborRateRatio is Float64. There is one instance of this variable for ~~all the domains~~each domain, i.e., ~~all the~~each PTP Instance~~s~~ (per port).~~ The variable is accessible by all the domains.~~

Change 10.2.5.8 as follows:

10.2.5.8 meanLinkDelay: The measured mean propagation delay (see 8.3) on the link attached to this port, relative to the LocalClock entity of the time-aware system at the other end of the link (i.e., expressed in the time base of the time-aware system at the other end of the link). The data type for meanLinkDelay is UScaledNs. There is one instance of this variable for all the domains, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

NOTE—The variable meanLinkDelay was named neighborPropDelay in the 2011 edition of this standard.

Change 10.2.5.9 as follows:

10.2.5.9 delayAsymmetry: The asymmetry in the propagation delay on the link attached to this port. If propagation delay asymmetry is not modeled, then delayAsymmetry is zero. The data type for delayAsymmetry is ScaledNs. There is one instance of this variable for CMLDS (see 11.2.17), and there is also one instance of this variable for each domain~~ that uses the instance-specific peer-to-peer delay mechanism~~. The instance of this variable for CMLDS is relative to the local clock. The instance of this variable for eachg~~a~~ domain ~~that uses the instance-specific peer-to-peer delay mechanism~~ is relative to the grandmaser time base for that domain. The instance of delayAsymmetry for CMLDS is used where needed in all computations done by the CMLDS peer-to-peer delay mechanism, and also by computations using the syncEgressTimestamp field of the Drift_Tracking TLV (see Clause 11.4.4.2.3) if CMLDS is present. The instance of delayAsymmetry for a domain is used where needed in all computations done for that domain, and also by computations using the Drift_Tracking TLV if CMLDS is not present.

Change 10.2.5.10 as follows:

10.2.5.10 computeNeighborRateRatio: A Boolean, set by the LinkDelayIntervalSetting state machine (see 11.2.21), that indicates whether neighborRateRatio is to be computed by this port. There is one instance of this variable for ~~all the domains, i.e., all the~~each domain, i.e., each PTP Instance~~s~~ (per port), and one instance of this variable for CMLDS. The instance of this variable for a domain indicates whether the neighborRateRatio is to be computed for that domain, using either Sync messages (if the value of nrrCompMethod is Sync and driftTrackingTlvSupport is TRUE (see 11.2.13.15)) or transport-specific peer-to-peer delay messages (if the value of nrrCompMethod is Pdelay or driftTrackingTlvSupport is FALSE. The result of the computation is placed in either nrrSync (see 11.2.13.14), if Sync messages are used, or nrrPdelay (see 11.2.13.13), if transport-specific peer-to-peer delay messages are used. The instance of this variable for CMLDS indicates whether neighborRateRatio is to be computed for that domain using CMLDS peer-to-peer delay messages. The result of the computation is placed in nrrPdelay. The variable is accessible by all the domains.

NOTE—Domain-specific computeNeighborRateRatio controls only whether the computation of nrrSync or nrrPdelay is performed at all, using either Sync messages or transport-specific peer-to-peer delay message depending on nrrCompMethod and driftTrackingTlvSupport. CMLDS computeNeighborRateRatio controls only whether the computation of nrrPdelay is performed at all, using CMLDS peer-to-peer delay messages. If domain-specific computeNeighborRateRatio and CMLDS computeNeighborRateRatio are both TRUE, the computation that is done will depend on the value of nrrCompMethod and whether CMLDS is present. However, if domain-specific computeNeighborRateRatio or CMLDS computeNeighborRateRatio is FALSE, the respective computation of nrrSync or nrrPdelay will not be done.

Change 10.2.5.11 as follows:

10.2.5.11 computeMeanLinkDelay: A Boolean, set by the LinkDelayIntervalSetting state machine (see 11.2.21), that indicates whether meanLinkDelay is to be computed by this port. There is one instance of this variable for all the domains, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

Change 10.2.5.13 as follows:

10.2.5.13 ptpPortEnabled: A Boolean that is administratively set to TRUE if time-synchronization is to be enabled on this PTP Port.

NOTE 1—It is expected that the value of ptpPortEnabled ~~will be~~is set via the management interface (see 14.8.4). A physical port ~~PTP Port~~ can be enabled for data transport but not for synchronization transport.

NOTE 2—The variable ptpPortEnabled was named pttPortEnabled in the 2011 edition of this standard. Only the name of this variable has changed; the definition and function of this variable are the same as in the 2011 edition. The name change is reflected in many state machines.

10.2.8 PortSyncSyncReceive state machine

Add the following definition to 10.2.8.1:

10.2.8.1 State machine variables

10.2.8.1.5 TEMP: A temporary variable used to reduce clutter in the state diagram (see Figure 10-4). The data type for TEMP is Integer16.

10.2.9 ClockTimeTransmitterSyncSend state machine

10.2.9.2 State machine functions

10.2.9.2.1 setPSSyncCMSS (gmRateRatio): Creates a PortSyncSync structure to be transmitted, and returns a pointer to this structure. The members are set as follows:

Add the following two NOTES to 10.2.9.2.1, just after item c)2), and renumber subsequent NOTES as needed:

NOTE 1—Both localTime (10.2.4.19) and timeTransmitterTime (10.2.4.21) are updated by the ClockTimeTransmitterSyncReceive state machine (10.2.11). The updates occur both when sourceTime (9.2.2.2) is received from the ClockSource (9.2) - indicated by the variable rcvdClockSourcReq (10.2.11.1.1) being TRUE - and when the LocalClock entity (10.1.2.1) itself updates by one tick - indicated by the variable rcvdLocalClockTickCMSR (10.2.11.1.3) being TRUE. timeTransmitterTime is updated by updateTimeTransmitterTime() (see 10.2.11.2.2). localTime is updated by setting it equal to currentTime (Figure 10-7). The result is that localTime is equal to the value of currentTime when timeTransmitterTime was most recently updated. localTime, currentTime, and timeTransmitterTime are per PTP Instance global variables. When they are updated, their values are known to all state machines.

NOTE 2—currentTime (10.2.4.19) and localTime (10.2.4.19) may not always be equal, despite the statement localTime = currentTime in the RECEIVE_SOURCE_TIME state of the ClockTimeTransmitterSyncReceive state machine (10.2.11). For example, if the ClockTimeTransmitterSyncReceive state machine is implemented as a different module than the ClockTimeTransmitterSyncSend state machine (10.2.9), with timeTransmitterTime (10.2.4.21) and localTime being sent from the ClockTimeTransmitterSyncReceive state machine to the ClockTimeTransmitterSyncSend state machine, then currentTime and localTime will not be equal. In this case, currentTime in c)2) is the value of the LocalClock entity (10.1.2.1) when the ClockTimeTransmitterSyncSend state machine receives timeTransmitterTime and localTime from the ClockTimeTransmitterSyncReceive state machine.>>

Change item m) of 10.2.9.2.1 as follows:

m) domainNumber is set equal to the ~~domain number~~domainNumber of this gPTP domain.

Change the first paragraph in 10.2.9.3 as follows:

10.2.9.3 State diagram

The ClockTimeTransmitterSyncSend state machine shall implement the function specified by the state diagram in Figure 10-5, the local variables specified in 10.2.9.1, the functions specified in 10.2.9.2, the structure specified in 10.2.2.3, and the relevant global variables and functions specified in 10.2.4 through 10.2.6. The state machine receives timeTransmitterTime and clockSourceTimeBaseIndicator from the ClockTimeTransmitterSyncReceive state machine, and phase and frequency offset between timeTransmitterTime and syncReceiptTime from the Clock~~TimeTransmitter~~SyncOffset state machine. It provides timeTransmitterTime (i.e., synchronized time) and the phase and frequency offset to the SiteSync entity via a PortSyncSync structure.

Change the heading of 10.2.10 as follows:

10.2.10 Clock~~TimeTransmitter~~SyncOffset state machine

Change 10.2.10.3 as follows:

10.2.10.3 State diagram

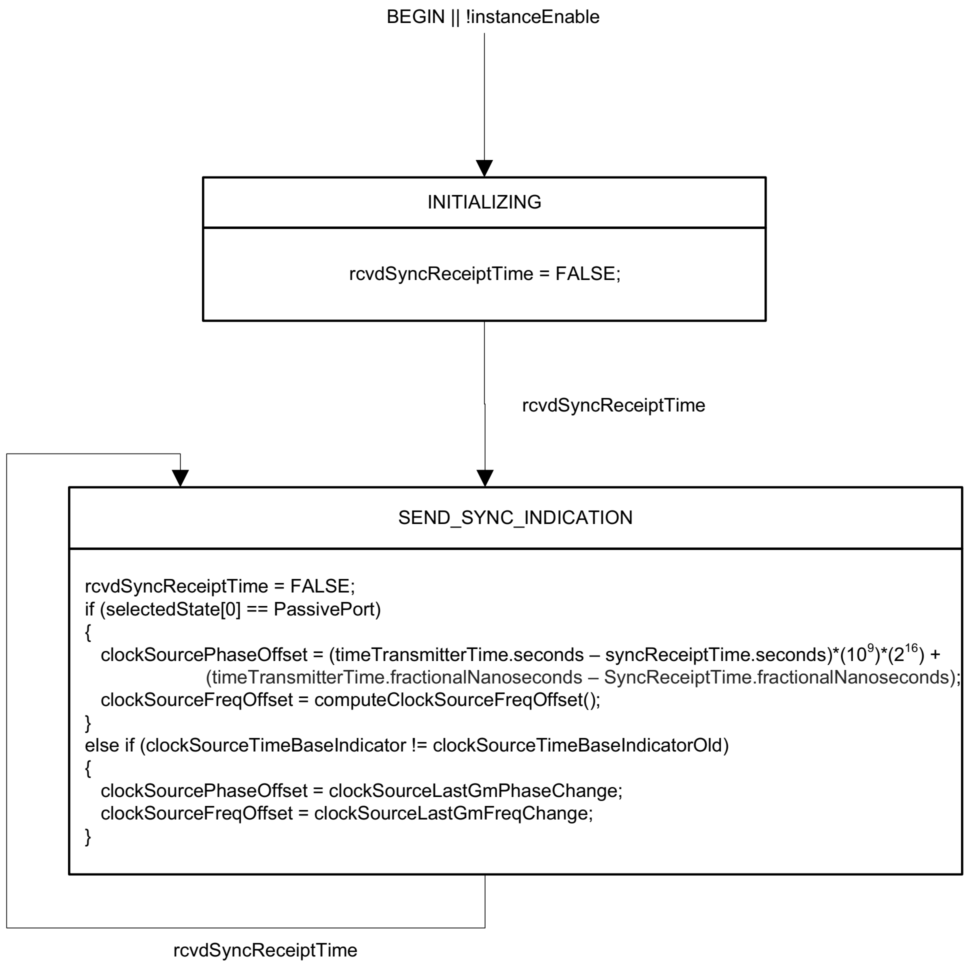
The Clock~~TimeTransmitter~~SyncOffset state machine shall implement the function specified by the state diagram in Figure , the local variable specified in 10.2.10.1, the function specified in 10.2.10.2, and the relevant global variables and functions specified in 10.2.4 through 10.2.6. The state machine receives syncReceiptTime from the ClockTimeReceiverSync state machine and timeTransmitterTime from the ClockTimeTransmitterSyncReceive state machine. It computes clockSourcePhaseOffset and clockSourceFrequency offset if this PTP Instance is not currently the Grandmaster PTP Instance, i.e., if selectedState[0] is equal to PassivePort.

The Clock~~TimeTransmitter~~SyncOffset state machine is optional for PTP Instances that are not grandmaster-capable (see 8.6.2.1, 10.1.3, and 10.2.1). This state machine may be present in a PTP Instance that is not grandmaster-capable; however, any information supplied by it, via the ClockTimeTransmitterSyncSend state machine, to the SiteSyncSync state machine is not used by the SiteSyncSync state machine if the PTP Instance is not grandmaster-capable.

Replace Figure 10-6 with the following:

.

)


BEGIN || !instanceEnable
INITIALIZING
rcvdSyncReceiptTime = FALSE;
rcvdSyncReceiptTime
SEND_SYNC_INDICATION
rcvdSyncReceiptTime = FALSE;
if (selectedState[0] == PassivePort)
{
clockSourcePhaseOffset = (timeTransmitterTime.seconds – syncReceiptTime.seconds)*(10^9)*(2^16) +
(timeTransmitterTime.fractionalNanoseconds – SyncReceiptTime.fractionalNanoseconds);
clockSourceFreqOffset = computeClockSourceFreqOffset();
}
else if (clockSourceTimeBaseIndicator != clockSourceTimeBaseIndicatorOld)
{
clockSourcePhaseOffset = clockSourceLastGmPhaseChange;
clockSourceFreqOffset = clockSourceLastGmFreqChange;
}
rcvdSyncReceiptTime

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 and P802.1ASdr applied, of this standard in that the statement clockSourcePhaseOffset = timeTransmitterTime.seconds – syncReceiptTime;
in the SEND_SYNC_INDICATION state is replaced by
clockSourcePhaseOffset = (timeTransmitterTime.seconds – syncReceiptTime.seconds)*(10^9)*(2^{16}) + (timeTransmitterTime.fractionalNanoseconds – SyncReceiptTime.fractionalNanoseconds);

Figure 10-6—Clock~~TimeTransmitter~~SyncOffset state machine

10.2.12 PortSyncSyncSend state machine

10.2.12.2 State machine functions

Change item j) of 10.2.12.2.1 as follows:

j) domainNumber is set equal to the ~~domain number~~domainNumber of this gPTP domain (see 8.1).

10.2.13 ClockTimeReceiverSync state machine

Change 10.2.13.1 as follows:

10.2.13.1 State machine variable~~s~~

The following variable~~s~~ ~~are~~is used in the state diagram in Figure 10-9 (in 10.2.13.3):

10.2.13.1.1 ~~**rcvdPSSyncCSS:** A Boolean variable that notifies the current state machine when a PortSyncSync structure is received from the SiteSyncSync state machine of the SiteSync entity. This variable is reset by this state machine.~~

10.2.13.1.2 ~~**rcvdLocalClockTickCSS:** A Boolean variable that notifies the current state machine when the LocalClock entity updates its time. This variable is reset by this state machine.~~

10.2.13.1.3 rcvdPSSyncPtrCSS: A pointer to the received PortSyncSync structure.

10.2.13.3 State diagram

Change the first paragraph of 10.2.13.3 as follows:

The ClockTimeReceiverSync state machine shall implement the function specified by the state diagram in Figure 10-9, the local variables specified in 10.2.13.1, the functions specified in 10.2.13.2, and the relevant global variables and functions specified in 10.2.4 through 10.2.6. The state machine receives a PortSyncSync structure from the SiteSyncSync state machine. It computes syncReceiptTime and clockTimeReceiverTime, and sets syncReceiptLocalTime (i.e., the time relative to the LocalClock entity corresponding to syncReceiptTime), GmTimeBaseIndicator, lastGmPhaseChange, and lastGmFreqChange. It provides clockTimeReceiverTime to the Clock~~TimeTransmitter~~SyncOffset state machine, and provides information to the ClockTarget entity (via the ClockTargetPhaseDiscontinuity interface; see 9.6) to enable that entity to determine if a phase or frequency discontinuity has occurred.

10.3 Best timeTransmitter clock selection, external port configuration, and announce interval setting state machines

10.3.1 Best timeTransmitter clock selection and external port configuration overview

Change 10.3.1.3 as follows:

10.3.1.3 External port configuration overview

In external port configuration (i.e., method b) of 10.3.1.1), an external entity determines the synchronization spanning tree and sets the PTP Port states accordingly. The method used by the external entity to determine the synchronization spanning tree is outside the scope of this standard. However, as with the BTCA, Announce messages are used to transport information on the time-synchronization spanning tree and Grandmaster PTP Instance time properties information from one PTP Instance to the next in the tree. The external entity sets the state of a PTP Port by setting the value of externalPortConfigurationPortDS.desiredState to the desired state.

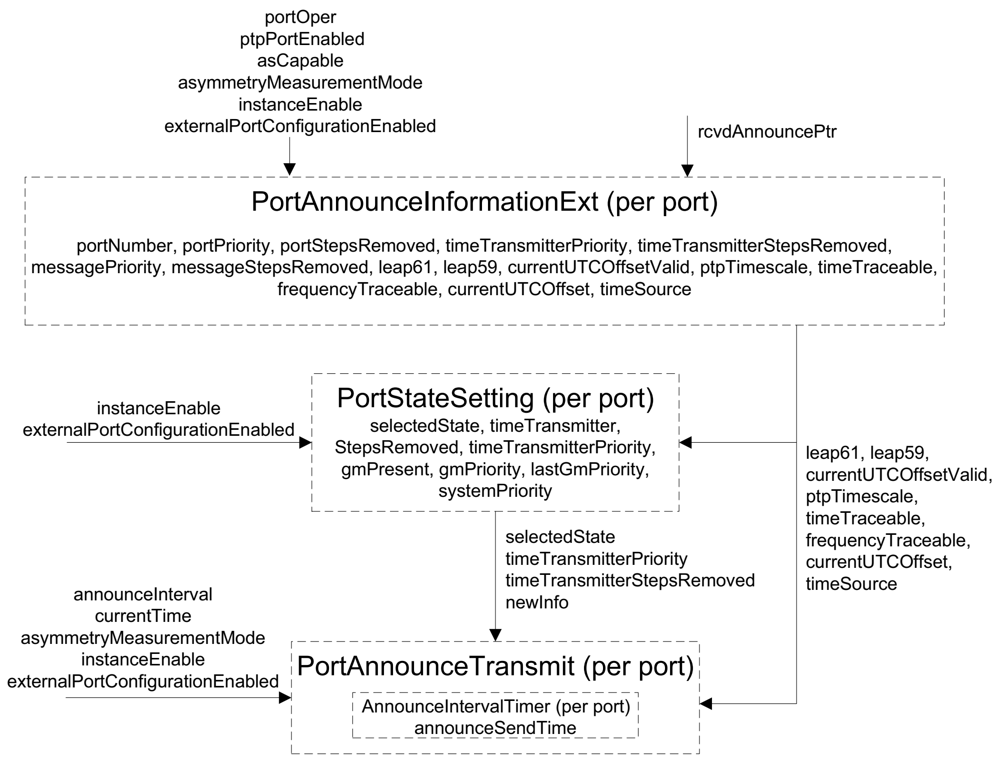
In the case of external port configuration, the time-synchronization spanning tree and the desired PTP Port states are controlled by an external entity, but the Grandmaster PTP Instance might change. If the timeReceiver port of a PTP Instance that is gmCapable (priority1 < 255, see 8.6.2.1) is no longer asCapable, the state of this PTP Port changes from TimeReceiverPort to DisabledPort (see 10.3.6.2) and the PTP Instance becomes Grandmaster for the time synchronization (sub-)tree where it is the root (10.3.15.2.2 d).3)). The original time-synchronization spanning tree can be split into disjunct subtrees with different, mutually unsynchronized, Grandmaster PTP Instances. If the port becomes asCapable again, its PTP Port state is again set to the desired state. If the timeReceiver port of a PTP Instance that is not gmCapable is no longer asCapable, the state of this PTP Port changes from TimeReceiverPort to DisabledPort (see 10.3.6.2). However, in this case gmPresent is set to FALSE (see 10.3.15.2.2 d) 3)), and the PTP Instance does not send Sync messages on any of its ports.

In external port configuration, there is no supervision of announce receipt timeout (see 10.7.3.2).

10.3.7 Overview of best timeTransmitter clock selection, external port configuration, and announce interval setting state machines

10.3.7.2 External port configuration state machines overview

Replace Figure 10-12 with the following:


portOper
ptpPortEnabled
asCapable
asymmetryMeasurementMode
instanceEnable
externalPortConfigurationEnabled
rcvdAnnouncePtr
PortAnnounceInformationExt (per port)
portNumber, portPriority, portStepsRemoved, timeTransmitterPriority, timeTransmitterStepsRemoved, messagePriority, messageStepsRemoved, leap61, leap59, currentUTCOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUTCOffset, timeSource
instanceEnable
externalPortConfigurationEnabled
PortStateSetting (per port)
selectedState, timeTransmitter, StepsRemoved, timeTransmitterPriority, gmPresent, gmPriority, lastGmPriority, systemPriority
leap61, leap59, currentUTCOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUTCOffset, timeSource
selectedState
timeTransmitterPriority
timeTransmitterStepsRemoved
newInfo
announceInterval
currentTime
asymmetryMeasurementMode
instanceEnable
externalPortConfigurationEnabled
PortAnnounceTransmit (per port)
AnnounceIntervalTimer (per port)
announceSendTime

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 and P802.1ASdr applied, of this standard in that the variable newInfo is removed from the link between the PortAnnounceInformationExt and PortAnnounceTransmit blocks and added to the link between the PortStateSetting and PortAnnounceTransmit blocks.

Figure 10-12—External port configuration state machines—overview and interrelationships

10.3.8 Overview of global variables used by best timeTransmitter clock selection, external port configuration, and announce interval setting state machines

Change 10.3.8 as follows:

~~Subclauses~~ 10.3.9 and 10.3.10 define global variables used by best timeTransmitter clock selection, external port configuration, and announce interval setting state machines whose scopes are as follows:

— Per PTP Instance (i.e., per domain)
— Per PTP Instance, per PTP Port
— Instance used by CMLDS (see 11.2.17) (i.e., variable is common across all ~~LinkPorts~~Link Ports)
— Instance used by CMLDS, per ~~LinkPort~~Link Port

Table 10-3 summarizes the scope of each global variable of 10.3.9 and 10.3.10.

Table 10-3—Summary of scope of global variables used by best timeTransmitter clock selection, external port configuration, and announce interval setting state machines (see and 10.3.10)

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~~Link Ports)	Instance used by CMLDS, per ~~LinkPort~~Link Port
reselect	10.3.9.1	Yes	No	No	No
selected	10.3.9.2	Yes	No	No	No
timeTransmitterStepsRemoved	10.3.9.3	Yes	No	No	No
leap61	10.3.9.4	Yes	No	No	No
leap59	10.3.9.5	Yes	No	No	No
currentUtcOffsetValid	10.3.9.6	Yes	No	No	No
ptpTimescale	10.3.9.7	Yes	No	No	No
timeTraceable	10.3.9.8	Yes	No	No	No
frequencyTraceable	10.3.9.9	Yes	No	No	No
currentUtcOffset	10.3.9.10	Yes	No	No	No
timeSource	10.3.9.11	Yes	No	No	No
sysLeap61	10.3.9.12	Yes	No	No	No
sysLeap59	10.3.9.13	Yes	No	No	No
sysCurrentUtcOffsetValid	10.3.9.14	Yes	No	No	No
sysPtpTimescale	10.3.9.15	Yes	No	No	No
sysTimeTraceable	10.3.9.16	Yes	No	No	No
sysFrequencyTraceable	10.3.9.17	Yes	No	No	No
sysCurrentUtcOffset	10.3.9.18	Yes	No	No	No
sysTimeSource	10.3.9.19	Yes	No	No	No

Table 10-3—Summary of scope of global variables used by best timeTransmitter clock selection, external port configuration, and announce interval setting state machines (see and 10.3.10) *(continued)*

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~~Link Ports)	Instance used by CMLDS, per ~~LinkPort~~Link Port
systemPriority	10.3.9.20	Yes	No	No	No
gmPriority	10.3.9.21	Yes	No	No	No
lastGmPriority	10.3.9.22	Yes	No	No	No
pathTrace	10.3.9.23	Yes	No	No	No
externalPortConfigurationEnabled	10.3.9.24	Yes	No	No	No
lastAnnouncePort	10.3.9.25	Yes	No	No	No
announceReceiptTimeoutTimeInterval	10.3.10.1	No	Yes	No	No
announceSlowdown	10.3.10.2	No	Yes	No	No
oldAnnounceInterval	10.3.10.3	No	Yes	No	No
infoIs	10.3.10.4	No	Yes	No	No
timeTransmitterPriority	10.3.10.5	No	Yes	No	No
currentLogAnnounceInterval	10.3.10.6	No	Yes	No	No
initialLogAnnounceInterval	10.3.10.7	No	Yes	No	No
announceInterval	10.3.10.8	No	Yes	No	No
messageStepsRemoved	10.3.10.9	No	Yes	No	No
newInfo	10.3.10.10	No	Yes	No	No
portPriority	10.3.10.11	No	Yes	No	No
portStepsRemoved	10.3.10.12	No	Yes	No	No
rcvdAnnouncePtr	10.3.10.13	No	Yes	No	No
rcvdMsg	10.3.10.14	No	Yes	No	No
updtInfo	10.3.10.15	No	Yes	No	No
annLeap61	10.3.10.16	No	Yes	No	No
annLeap59	10.3.10.17	No	Yes	No	No
annCurrentUtcOffsetValid	10.3.10.18	No	Yes	No	No
annPtpTimescale	10.3.10.19	No	Yes	No	No
annTimeTraceable	10.3.10.20	No	Yes	No	No
annFrequencyTraceable	10.3.10.21	No	Yes	No	No
annCurrentUtcOffset	10.3.10.22	No	Yes	No	No
annTimeSource	10.3.10.23	No	Yes	No	No
receivedPathTrace	10.3.10.24	No	Yes	No	No

Change 10.3.9 as follows:

10.3.9 Per PTP Instance global variables

10.3.9.1 reselect: A Boolean array of length numberPorts+1 (see 8.6.2.8). Setting reselect[j], where 0 ≤ j ≤ numberPorts, to TRUE causes the STATE_SELECTION block of the PortStateSelection state machine (see 10.3.13) to be re-entered, which in turn causes the PTP Port state of each PTP Port of the PTP Instance to be updated (via the function updtStatesTree(); see 10.3.13.2.4). This variable is used only by the BTCA, i.e., not by the ~~explicit port state~~external port configuration option.

10.3.9.2 selected: A Boolean array of length numberPorts+1 (see 8.6.2.8). selected[j], where 0 ≤ j ≤ numberPorts, is set to TRUE immediately after the PTP Port states of all the ports are updated. This value indicates to the PortAnnounceInformation state machine (see 10.3.12) that it can update the portPriorityVector and other variables for each PTP Port. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option; however, its value does not impact the ~~explicit port state~~external port configuration option (see the NOTE in 10.3.16.3).

NOTE—Array elements 0 of the reselect and selected arrays are not used, except that the function clearReselectTree() sets reselect[0] to FALSE when it sets the entire array to zero and the function setSelectedTree() sets selected[0] to TRUE when it sets the entire array to TRUE. This action is taken only for convenience, so that array element j can correspond to PTP Port j. Note also that, in contrast, selectedState[0] is ~~not~~ used (see 10.2.4.20)

10.3.9.3 timeTransmitterStepsRemoved: The value of stepsRemoved for the PTP Instance, after the PTP Port states of all the ports have been updated (see 10.3.13.2.4 for details on the computation of timeTransmitterStepsRemoved). The data type for timeTransmitterStepsRemoved is UInteger16. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.4 leap61: A Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the current Grandmaster Clock, contains 61 s and FALSE if the last minute of the current UTC day does not contain 61 s. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.5 leap59: A Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the current Grandmaster Clock, contains 59 s and FALSE if the last minute of the current UTC day does not contain 59 s. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.6 currentUtcOffsetValid: A Boolean variable whose value is TRUE if currentUtcOffset (see 10.3.9.10), relative to the current Grandmaster Clock, is known to be correct and FALSE if currentUtcOffset is not known to be correct. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.7 ptpTimescale: A Boolean variable whose value is TRUE if the timescale of the current Grandmaster Clock is PTP (see 8.2.1) and FALSE if the timescale is ARB. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.8 timeTraceable: A Boolean variable whose value is TRUE if both clockTimeReceiverTime [i.e., the synchronized time maintained at the timeReceiver (see 10.2.4.3)] and currentUtcOffset (see 10.3.9.10), relative to the current Grandmaster Clock, are traceable to a primary reference and FALSE if one or both are not traceable to a primary reference. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.9 frequencyTraceable: A Boolean variable whose value is TRUE if the frequency that determines clockTimeReceiverTime, i.e., the frequency of the LocalClockEntity multiplied by the most recently computed rateRatio by the PortSyncSyncReceive state machine (see 10.2.8.1.4), is traceable to a primary reference and FALSE if this frequency is not traceable to a primary reference. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.10 currentUtcOffset: The difference between TAI time and UTC time, i.e., TAI time minus UTC time, in seconds, and relative to the current Grandmaster Clock, when known. Otherwise, the value has no meaning (see 10.3.9.6). The data type for currentUtcOffset is Integer16. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

NOTE—For example, 2006-01-01 00:00:00 UTC and 2006-01-01 00:00:33 TAI represent the same instant of time. At this time, currentUtcOffset was equal to 33 s.[1]

10.3.9.11 timeSource: The value of the timeSource attribute of the current Grandmaster PTP Instance. The data type for timeSource is TimeSource (see 8.6.2.7). This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.12 sysLeap61: A Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the ClockTimeTransmitter entity of this PTP Instance, contains 61 s and FALSE if the last minute of the current UTC day does not contain 61 s. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.13 sysLeap59: A Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the ClockTimeTransmitter entity of this PTP Instance, contains 59 s and FALSE if the last minute of the current UTC day does not contain 59 s. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.14 sysCurrentUtcOffsetValid: A Boolean variable whose value is TRUE if currentUtcOffset (see 10.3.9.10), relative to the ClockTimeTransmitter entity of this PTP Instance, is known to be correct and FALSE if currentUtcOffset is not known to be correct. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.15 sysPtpTimescale: A Boolean variable whose value is TRUE if the timescale of the ClockTimeTransmitter entity of this PTP Instance is PTP (see 8.2.1) and FALSE if the timescale of the ClockTimeTransmitter entity of this PTP Instance is ARB. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.16 sysTimeTraceable: A Boolean variable whose value is TRUE if both timeTransmitterTime [i.e., the time maintained by the ClockTimeTransmitter entity of this PTP Instance (see 10.2.4.21)] and currentUtcOffset (see 10.3.9.10), relative to the ClockTimeTransmitter entity of this PTP Instance, are traceable to a primary reference and FALSE if one or both are not traceable to a primary reference. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.17 sysFrequencyTraceable: A Boolean variable whose value is TRUE if the frequency that determines timeTransmitterTime of the ClockTimeTransmitter entity of this PTP Instance, i.e., the frequency of the LocalClockEntity multiplied by the most recently computed gmRateRatio by the ClockTimeTransmitterSyncReceive state machine (see 10.2.4.14 and 10.2.11), is traceable to a primary reference and FALSE if this frequency is not traceable to a primary reference. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

[1]Note also that a leap second was not added at the end of the last UTC minute of 2005-12-31.

10.3.9.18 sysCurrentUtcOffset: The difference between TAI time and UTC time, i.e., TAI time minus UTC time, in seconds, and relative to the ClockTimeTransmitter entity of this PTP Instance, when known. Otherwise, the value has no meaning (see 10.3.9.14). The data type for sysCurrentUtcOffset is Integer16. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

NOTE—See the NOTE in 10.3.9.10 for more detail on the sign convention.

10.3.9.19 sysTimeSource: The value of the timeSource attribute of the ClockTimeTransmitter entity of this PTP Instance (see 8.6.2.7). The data type for sysTimeSource is TimeSource.

10.3.9.20 systemPriority: The systemPriority vector for this PTP Instance. The data type for systemPriority is UInteger224 (see 10.3.5).

10.3.9.21 gmPriority: The current gmPriorityVector for the PTP Instance. The data type for gmPriority is UInteger224 (see 10.3.5).

10.3.9.22 lastGmPriority: The previous gmPriorityVector for the PTP Instance, prior to the most recent invocation of the PortStateSelection state machine. The data type for lastGmPriority is UInteger224 (see 10.3.4). lastGmPriority is used only by the BTCA, i.e., not by the ~~explicit port state~~external port configuration option.

10.3.9.23 pathTrace: An array that contains the clockIdentities of the successive PTP Instances that receive, process, and send Announce messages. The data type for pathTrace is ClockIdentity[N], where N is the number of PTP Instances, including the Grandmaster PTP Instance, that the Announce information has traversed. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

NOTE 1—N is equal to stepsRemoved+1 (see 10.6.3.2.6). The size of the pathTrace array can change after each reception of an Announce message, up to the maximum size for the respective medium. For example, the maximum value of N for a full-duplex IEEE 802.3 medium is 179. This is obtained from the fact that the number of PTP octets in an Announce message is 68 + 8N, where N is the number of entries in the pathTrace array (see 10.6.3.1 and Table 10-11), and the maximum payload size for full-duplex IEEE 802.3 media is 1500 octets. Setting 68 + 8N = 1500, and solving for N gives N = 179.

NOTE 2—The current behavior for the path trace feature is documented in 10.3.11.2.1 and 10.3.16.2.1 and is as follows:

— Item c) of 10.3.11.2.1, the description of the qualifyAnnounce() function of the PortAnnounceReceive state machine, indicates that if a path trace TLV is present and one of the elements of the pathSequence array field is equal to the clockIdentity of the clock where the TLV is being processed, the Announce message is not qualified.
— Item d) of 10.3.11.2.1 (qualifyAnnounce() function) indicates that if the Announce message is qualified and a path trace TLV is present, the pathSequence array of the TLV is copied to the pathTrace array (described in this subclause) and the clockIdentity of the PTP Instance that processes the Announce message is appended to the array. However, if a path trace TLV is not present, the path trace array is empty.
— Item f) of 10.3.16.2.1, the description of the txAnnounce() function of the PortAnnounceTransmit state machine, indicates that a path trace TLV is constructed and appended to an Announce message just before the Announce message is transmitted only if the pathTrace array is not empty and appending the TLV does not cause the media-dependent layer frame to exceed any respective maximum size. If appending the TLV does cause a respective maximum frame size to be exceeded or if the pathTrace array is empty, the TLV is not appended.
— As a result of the behaviors of the qualifyAnnounce() and txAnnounce() functions described in this note, the path trace feature is not used, i.e., a path trace TLV is not appended to an Announce message and the pathTrace array is empty, once appending a clockIdentity to the TLV would cause the frame carrying the Announce message to exceed its maximum size.

NOTE 3—Once the value of stepsRemoved of an Announce message reaches 255, the Announce message is not qualified [see item b) of 10.3.11.2.1].

10.3.9.24 externalPortConfigurationEnabled: A variable whose value indicates whether PTP Port states are externally configured or determined by the BTCA. The data type shall be Boolean. The value TRUE indicates that the PTP Port states are externally configured; the value FALSE indicates that the PTP Port states are determined by the BTCA. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.9.25 lastAnnouncePort: The PTP Port number of the PTP Port on which the most recent Announce message was received. This variable is used by the PortAnnounceInformationExt and PortStateSettingExt state machines for the ~~explicit port state~~external port configuration option. This variable is not used by the BTCA. The data type for this variable is UInteger16.

Change 10.3.10 as follows:

10.3.10 Per-port global variables

10.3.10.1 announceReceiptTimeoutTimeInterval: The time interval after which announce receipt timeout occurs if an Announce message has not been received during the interval. The value of announceReceiptTimeoutTimeInterval is equal to announceReceiptTimeout (see 10.7.3.2) multiplied by the announceInterval (see 10.3.10.8) for the PTP Port at the other end of the link to which this PTP Port is attached. The value of announceInterval for the PTP Port at the other end of the link is computed from logMessageInterval of the received Announce message (see 10.6.2.2.14). The data type for announceReceiptTimeoutTimeInterval is UScaledNs. This variable is used only by the BTCA, i.e., not by the ~~explicit port state~~external port configuration option.

10.3.10.2 announceSlowdown: A Boolean that is set to TRUE if the AnnounceIntervalSetting state machine (see Figure 10-19 in item 10.3.17.3) receives a TLV that requests a larger Announce message transmission interval (see 10.7.2.2) and FALSE otherwise. When announceSlowdown is set to TRUE, the PortAnnounceTransmit state machine (see Figure) continues to send Announce messages at the old (i.e., faster) rate until a number of Announce messages equal to announceReceiptTimeout (see 10.7.3.2) have been sent, but with the logMessageInterval field of the PTP common header set equal to the new announce interval (i.e., corresponding to the slower rate). After announceReceiptTimeout Announce messages have been sent, subsequent Announce messages are sent at the new (i.e., slower) rate and with the logMessageInterval field of the PTP common header set to the new announce interval. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option. When announceSlowdown is set to FALSE, the PortAnnounceTransmit state machine immediately sends Announce messages at the new (i.e., slower) rate.

NOTE—If a receiver of Announce messages requests a slower rate, the receiver ~~will~~ continues to use the upstream announceInterval value, which it obtains from the logMessageInterval field of received Announce messages, until it receives an Announce message where that value has changed. If, immediately after requesting a slower Announce message rate, up to announceReceiptTimeout minus one consecutive Announce messages sent to the receiver are lost, announce receipt timeout could occur if the sender had changed to the slower rate immediately. Delaying the slowing down of the sending rate of Announce messages for announceReceiptTimeout messages prevents announce receipt timeout from occurring until at least announceReceiptTimeout Announce messages have been lost. Note that networks with high packet loss can still experience announce receipt timeout under high-packet-loss conditions; however, the announce receipt timeout condition occurs only after at least announceReceiptTimeout Announce messages have been lost.

10.3.10.3 oldAnnounceInterval: The saved value of the previous announce interval, when a new announce interval is requested via a Signaling message that contains a message interval request TLV. The data type for oldAnnounceInterval is UScaledNs. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.4 infoIs: An Enumeration2 that takes the values Received, Mine, Aged, or Disabled to indicate the origin and state of the PTP Port's time-synchronization spanning tree information:

a) If infoIs is Received, the PTP Port has received current information (i.e., announce receipt timeout has not occurred and, if gmPresent is TRUE, sync receipt timeout also has not occurred) from the timeTransmitter PTP Instance for the attached gPTP communication path.
b) If infoIs is Mine, information for the PTP Port has been derived from the TimeReceiverPort for the PTP Instance (with the addition of TimeReceiverPort stepsRemoved). This includes the possibility that the TimeReceiverPort is the PTP Port whose portNumber is 0, i.e., the PTP Instance is the root of the gPTP domain.
c) If infoIs is Aged, announce receipt timeout or, when gmPresent is TRUE, sync receipt timeout has occurred.
d) If portOper, ptpPortEnabled, and asCapable are not all TRUE, infoIs is Disabled.

The variable infoIs is used only by the BTCA, i.e., not by the ~~explicit port state~~external port configuration option.

10.3.10.5 timeTransmitterPriority: The timeTransmitterPriorityVector for the PTP Port. The data type for timeTransmitterPriority is UInteger224 (see 10.3.4). This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.6 currentLogAnnounceInterval: The current value of the logarithm to base 2 of the mean time interval, in seconds, between the sending of successive Announce messages (see 10.7.2.2). This value is set in the AnnounceIntervalSetting state machine (see 10.3.17). The data type for currentLogAnnounceInterval is Integer8. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.7 initialLogAnnounceInterval: The initial value of the logarithm to base 2 of the mean time interval, in seconds, between the sending of successive Announce messages (see 10.7.2.2). The data type for initialLogAnnounceInterval is Integer8. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.8 announceInterval: A variable containing the mean Announce message transmission interval for the PTP Port. This value is set in the AnnounceIntervalSetting state machine (see 10.3.17). The data type for announceInterval is UScaledNs. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.9 messageStepsRemoved: The value of stepsRemoved contained in the received Announce information. The data type for messageStepsRemoved is UInteger16. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.10 newInfo: A Boolean variable that is set to cause a PTP Port to transmit Announce information; specifically, it is set when an announce interval has elapsed (see Figure), PTP Port states have been updated, and portPriority and portStepsRemoved information has been updated with newly determined timeTransmitterPriority and timeTransmitterStepsRemoved information. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.11 portPriority: The portPriorityVector for the PTP Port. The data type for portPriority is UInteger224 (see 10.3.4). This variable is used only by the BTCA, i.e., not by the ~~explicit port state~~external port configuration option.

10.3.10.12 portStepsRemoved: The value of stepsRemoved for the PTP Port. portStepsRemoved is set equal to timeTransmitterStepsRemoved (see 10.3.9.3) after timeTransmitterStepsRemoved is updated. The data type for portStepsRemoved is UInteger16. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.13 rcvdAnnouncePtr: A pointer to a structure that contains the fields of a received Announce message. This variable is used by both the BTCA and the explicit PTP Port state configuration option.

10.3.10.14 rcvdMsg: A Boolean variable that is TRUE if a received Announce message is qualified and FALSE if it is not qualified. This variable is used only by the BTCA, i.e., not by the ~~explicit port state~~external port configuration option.

10.3.10.15 updtInfo: A Boolean variable that is set to TRUE to indicate that the PortAnnounceInformation state machine (see 10.3.12) should copy the newly determined timeTransmitterPriority and timeTransmitterStepsRemoved to portPriority and portStepsRemoved, respectively. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option; however, its value does not impact the ~~explicit port state~~external port configuration option (see the NOTE in 10.3.16.3).

10.3.10.16 annLeap61: A global variable in which the leap61 flag (see 10.6.2.2.8) of a received Announce message is saved. The data type for annLeap61 is Boolean. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.17 annLeap59: A global variable in which the leap59 flag (see 10.6.2.2.8) of a received Announce message is saved. The data type for annLeap59 is Boolean. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.18 annCurrentUtcOffsetValid: A global variable in which the currentUtcOffsetValid flag (see 10.6.2.2.8) of a received Announce message is saved. The data type for annCurrentUtcOffsetValid is Boolean. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.19 annPtpTimescale: A global variable in which the ptpTimescale flag (see 10.6.2.2.8) of a received Announce message is saved. The data type for annPtpTimescale is Boolean. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.20 annTimeTraceable: A global variable in which the timeTraceable flag (see 10.6.2.2.8) of a received Announce message is saved. The data type for annTimeTraceable is Boolean. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.21 annFrequencyTraceable: A global variable in which the frequencyTraceable flag (see 10.6.2.2.8) of a received Announce message is saved. The data type for annFrequencyTraceable is Boolean. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.22 annCurrentUtcOffset: A global variable in which the currentUtcOffset field (see 10.6.3.2.1) of a received Announce message is saved. The data type for annCurrentUtcOffset is Integer16. This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.23 annTimeSource: A global variable in which the timeSource field (see 10.6.3.2.1) of a received Announce message is saved. The data type for annTimeSource is TimeSource (see 8.6.2.7). This variable is used by both the BTCA and the ~~explicit port state~~external port configuration option.

10.3.10.24 receivedPathTrace: An array in which the pathSequence array field of the path trace TLV of the most recently received Announce message is saved. The data type for receivedPathTrace is clockIdentity[N], where N is the number of entries in the pathSequence array field.

10.3.11 PortAnnounceReceive state machine

Change 10.3.11.3 as follows:

10.3.11.3 State diagram

The PortAnnounceReceive state machine shall implement the function specified by the state diagram in Figure 10-13, the local variable specified in 10.3.11.1, the function specified in 10.3.11.2, and the relevant global variables specified in 10.2.4, 10.2.5, 10.3.9, and 10.3.10~~, and 11.2.13~~. The state machine is not used if externalPortConfigurationEnabled is TRUE. The state machine receives Announce information from the MD entity of the same PTP Port, determines if the Announce message is qualified, and if so, sets the rcvdMsg variable.

10.3.12 PortAnnounceInformation state machine

Add the following definition to 10.3.12.1:

10.3.12.1 State machine variables

10.3.12.1.4 TEMP: A temporary variable used to reduce clutter in the state diagram (see Figure 10-14). The data type for TEMP is Integer16.

Change 10.3.12.3 as follows:

10.3.12.3 State diagram

The PortAnnounceInformation state machine shall implement the function specified by the state diagram in Figure 10-14, the local variables specified in 10.3.12.1, the functions specified in 10.3.12.2, and the relevant global variables specified in 10.2.4, 10.2.5, 10.3.9, and 10.3.10~~, and 11.2.13~~. This state machine is used only if externalPortConfigurationEnabled is FALSE (if this variable is TRUE, the PortAnnounceInformationExt state machine of 10.3.14.3 is used instead). The state machine receives new qualified Announce information from the PortAnnounceReceive state machine (see 10.3.11) of the same PTP Port and determines if the Announce information is better than the current best timeTransmitter information it knows. The state machine also updates the current best timeTransmitter information when it receives updated PTP Port state information from the PortStateSelection state machine (see 10.3.13) and when announce receipt timeout or, when gmPresent is TRUE, sync receipt timeout occurs.

10.3.13 PortStateSelection state machine

10.3.13.2 State machine functions

Change 10.3.13.2.4 as follows:

10.3.13.2.4 UpdtStatesTree(): Performs the following operations (see 10.3.4 and 10.3.5 for details on the priority vectors):

a) Computes the gmPathPriorityVector for each PTP Port that has a portPriorityVector and for which neither announce receipt timeout nor, if gmPresent is TRUE, sync receipt timeout have occurred,

b) Saves gmPriority (see 10.3.9.21) in lastGmPriority (see 10.3.9.22), computes the gmPriorityVector for the PTP Instance and saves it in gmPriority, chosen as the best of the set consisting of the systemPriorityVector (for this PTP Instance) and the gmPathPriorityVector for each PTP Port for which the clockIdentity of the timeTransmitter port is not equal to thisClock (see 10.2.4.22),

c) Sets the per PTP Instance global variables leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource as follows:

   1) If the gmPriorityVector was set to the gmPathPriorityVector of one of the ports, then leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource are set to annLeap61, annLeap59,

annCurrentUtcOffsetValid, annPtpTimescale, annTimeTraceable, annFrequencyTraceable, annCurrentUtcOffset, and annTimeSource, respectively, for that PTP Port.

2) If the gmPriorityVector was set to the systemPriorityVector, then leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource are set to sysLeap61, sysLeap59, sysCurrentUtcOffsetValid, sysPtpTimescale, sysTimeTraceable, sysFrequencyTraceable, sysCurrentUtcOffset, and sysTimeSource, respectively.

d) Updates the timePropertiesDS members leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource with the values of the corresponding global variables computed in item c) above.

e) Updates the parentDS members grandmasterIdentity, grandmasterClockQuality.clockClass, grandmasterClockQuality.clockAccuracy, grandmasterClockQuality.offsetScaledLogVariance, grandmasterPriority1, grandmasterPriority2 with the clockIdentity, clockClass, clockAccuracy, offsetScaledLogVariance, priority1, and priority2 attributes, respectively, of the systemIdentity component of the gmPrioirity computed in item b) above.

f) Updates the parentDS member parentPortIdentity with the sourcePortIdentity component of the gmPriority computed in item b) above.

g) Computes the timeTransmitterPriorityVector for each PTP Port.

h) Computes timeTransmitterStepsRemoved, which is equal to one of the following:

1) messageStepsRemoved (see 10.3.10.9) for the PTP Port associated with the gmPriorityVector, incremented by 1, if the gmPriorityVector is not the systemPriorityVector, or

2) 0 if the gmPriorityVector is the systemPriorityVector.

i) Sets currentDS.stepsRemoved equal to timeTransmitterStepsRemoved.

j) Assigns the PTP Port state for PTP Port j, and sets selectedState[j] equal to this PTP Port state, as follows, for j = 1, 2, ..., numberPorts:

1) If the PTP Port is disabled (infoIs == Disabled), then selectedState[j] is set to DisabledPort.

2) If asymmetryMeasurementMode is TRUE, then selectedState[j] is set to PassivePort, and updtInfo is set to FALSE.

3) If announce receipt timeout, or sync receipt timeout with gmPresent set to TRUE, has occurred (infoIs = Aged), then selectedState[j] is set to TimeTransmitterPort, and updtInfo is set to TRUE.

4) If the portPriorityVector was derived from another PTP Port on the PTP Instance or from the PTP Instance itself as the root (infoIs == Mine), then selectedState[j] is set to TimeTransmitterPort. In addition, updtInfo is set to TRUE if the portPriorityVector differs from the timeTransmitterPriorityVector or portStepsRemoved differs from timeTransmitterStepsRemoved.

5) If the portPriorityVector was received in an Announce message, announce receipt timeout, or sync receipt timeout with gmPresent TRUE, has not occurred (infoIs == Received), and the gmPriorityVector is now derived from the portPriorityVector, then selectedState[j] is set to TimeReceiverPort, and updtInfo is set to FALSE. The per port global variable receivedPathTrace, for this port, is copied to the per PTP Instance global array pathTrace, and, if it is not empty, thisClock is appended to pathTrace.

6) If the portPriorityVector was received in an Announce message, announce receipt timeout, or sync receipt timeout with gmPresent TRUE, has not occurred (infoIs == Received), the gmPriorityVector is not now derived from the portPriorityVector, the timeTransmitterPriorityVector is not better than the portPriorityVector, and the sourcePortIdentity component of the portPriorityVector *does not* reflect another PTP Port on the PTP Instance, then selectedState[j] is set to PassivePort, and updtInfo is set to FALSE.

7) If the portPriorityVector was received in an Announce message, announce receipt timeout, or sync receipt timeout with gmPresent TRUE, has not occurred (infoIs == Received), the gmPriorityVector is not now derived from the portPriorityVector, the timeTransmitterPriorityVector is not better than the portPriorityVector, and the sourcePortIdentity component of the portPriorityVector *does* reflect another PTP Port on the PTP Instance, then selectedState[j] set to PassivePort, and updtInfo is set to FALSE.

8) If the portPriorityVector was received in an Announce message, announce receipt timeout, or sync receipt timeout with gmPresent TRUE, has not occurred (infoIs == Received), the gmPriorityVector is not now derived from the portPriorityVector, and the timeTransmitterPriorityVector is better than the portPriorityVector, then selectedState[j]is set to TimeTransmitterPort, and updtInfo is set to TRUE.

k) Updates gmPresent as follows:

1) gmPresent is set to TRUE if the priority1 field of the rootSystemIdentity of the gmPriorityVector is less than 255.

2) gmPresent is set to FALSE if the priority1 field of the rootSystemIdentity of the gmPriorityVector is equal to 255.

l) Assigns the PTP Port state for PTP Port 0 (see 8.5.2.3), and sets selectedState[0] as follows:

1) if selectedState[j] is set to TimeReceiverPort for any PTP Port with portNumber j, j = 1, 2, ..., numberPorts, selectedState[0] is set to PassivePort.

2) if selectedState[j] is *not* set to TimeReceiverPort for any PTP Port with portNumber j, j = 1, 2, ..., numberPorts, selectedState[0] is set to TimeReceiverPort.

m) If the clockIdentity member of the systemIdentity (see 10.3.2) member of gmPriority (see 10.3.9.21) is equal to thisClock (see 10.2.4.22), i.e., if the current PTP Instance is the Grandmaster PTP Instance, the pathTrace array is set to contain the single element thisClock (see 10.2.4.22).

Change 10.3.13.3 as follows:

10.3.13.3 State diagram

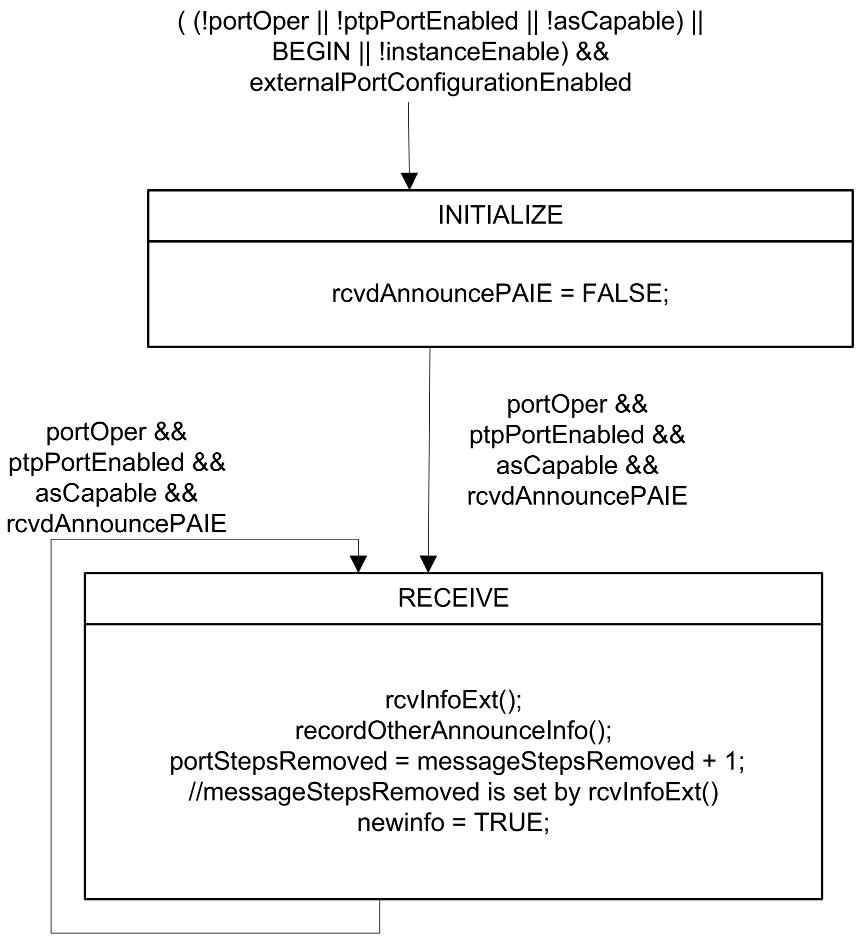
The PortStateSelection state machine shall implement the function specified by the state diagram in Figure 10-15, the functions specified in 10.3.13.1, and the relevant global variables specified in 10.2.4, 10.2.5, 10.3.9, and 10.3.10~~, and 11.2.13~~. This state machine is used only if externalPortConfigurationEnabled is FALSE (if this variable is TRUE, the PortStateSettingExt state machine is used instead). The state machine updates the gmPathPriority vector for each PTP Port of the PTP Instance, the gmPriorityVector for the PTP Instance, and the timeTransmitterPriorityVector for each PTP Port of the PTP Instance. The state machine determines the PTP Port state for each PTP Port and updates gmPresent.

10.3.14 PortAnnounceInformationExt state machine

Change 10.3.14.3 as follows:

10.3.14.3 State diagram

The PortAnnounceInformationExt state machine shall implement the function specified by the state diagram in Figure 10-16, the local variables specified in 10.3.14.1, the functions specified in 10.3.14.2, and the relevant global variables specified in 10.2.4, 10.2.5, 10.3.9, and 10.3.10~~, and 11.2.13~~. This state machine is used only if externalPortConfigurationEnabled is TRUE (if this variable is FALSE, the PortAnnounceInformation state machine of 10.3.12.3 is used instead). The state machine receives Announce information from the MD entity of the same PTP Port and saves the information..


((!portOper || !ptpPortEnabled || !asCapable) ||
BEGIN || !instanceEnable) &&
externalPortConfigurationEnabled
INITIALIZE
rcvdAnnouncePAIE = FALSE;
portOper &&
ptpPortEnabled &&
asCapable &&
rcvdAnnouncePAIE
portOper &&
ptpPortEnabled &&
asCapable &&
rcvdAnnouncePAIE
RECEIVE
rcvInfoExt();
recordOtherAnnounceInfo();
portStepsRemoved = messageStepsRemoved + 1;
//messageStepsRemoved is set by rcvInfoExt()
newinfo = TRUE;

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 and P802.1ASdr applied, of this standard in that the statement newinfo = TRUE; is added after the existing statements in the RECEIVE state.

Figure 10-16—PortAnnounceInformationExt state machine

10.3.15 PortStateSettingExt state machine

Change 10.3.15.2.2 as follows:

10.3.15.2.2 updtPortState(j): Performs the following operations for PTP Port j (see 10.3.4 and 10.3.5 for details on the priority vectors):

a) Sets the per PTP Instance global variables leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource as follows:
   1) If the PTP Port state of any PTP Port of this PTP Instance ~~other than PTP Port 0~~ (see 8.5.2.3) is TimeReceiverPort, then leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource are set to annLeap61, annLeap59, annCurrentUtcOffsetValid, annPtpTimescale, annTimeTraceable, annFrequencyTraceable, annCurrentUtcOffset, and annTimeSource, respectively, for that PTP Port.
   2) If no PTP Port of this PTP Instance ~~other than PTP Port 0~~ (see 8.5.2.3) has the PTP Port state TimeReceiverPort, then leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource are set to sysLeap61, sysLeap59, sysCurrentUtcOffsetValid, sysPtpTimescale, sysTimeTraceable, sysFrequencyTraceable, sysCurrentUtcOffset, and sysTimeSource, respectively.

b) Update the timePropertiesDS members leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource with the values of the corresponding global variables computed in item a) above.

c) Computes timeTransmitterStepsRemoved as follows:
   1) If the PTP Port state of any PTP Port of this PTP Instance ~~other than PTP Port 0~~ (see 8.5.2.3) is TimeReceiverPort, then timeTransmitterStepsRemoved is set equal to portStepsRemoved for that PTP Port.
   2) If no PTP Port of this PTP Instance ~~other than PTP Port 0~~ (see 8.5.2.3) has the PTP Port state TimeReceiverPort, then timeTransmitterStepsRemoved is set equal to 0.

d) Sets currentDS.stepsRemoved equal to timeTransmitterStepsRemoved.

e) Assigns the PTP Port state for PTP Port j, and sets selectedState[j] equal to this PTP Port state, as follows:
   1) If disabledExt is TRUE, selectedState[j] is set to DisabledPort, else
   2) If asymmetryMeasurementMode is TRUE, selectedState[j] is set to PassivePort, else
   3) selectedState[j] is set to portStateInd. If portStateInd is equal to TimeTransmitterPort, newInfo is set to TRUE.

f) Updates gmPresent as follows:
   1) If the PTP Port state of any PTP Port of this PTP Instance ~~other than PTP Port 0~~ (see 8.5.2.3)is TimeReceiverPort and the priority1 field of the rootSystemIdentity of the messagePriorityPAIE of the timeReceiver port is less than 255, gmPresent is set to TRUE, else
   2) If the PTP Port state of any PTP Port of this PTP Instance ~~other than PTP Port 0~~ (see 8.5.2.3)is TimeReceiverPort and the priority1 field of the rootSystemIdentity of the messagePriorityPAIE of the timeReceiver PTP Port is equal to 255, gmPresent is set to FALSE, else
   3) If no PTP Port of this PTP Instance ~~other than PTP Port 0~~ (see 8.5.2.3) has the PTP Port state TimeReceiverPort, gmPresent is set to TRUE if priority1 for this PTP Instance is less than 255 and FALSE if priority1 for this PTP Instance is equal to 255.

g) Assigns the PTP Port state for PTP Port 0, and sets selectedState[0] as follows:
   1) If selectedState[j] is set to TimeReceiverPort, selectedState[0] is set to PassivePort.

   2) If selectedState[j] is *not* set to TimeReceiverPort and selectedState[k] is not equal to TimeReceiverPort for every k not equal to 0 or j, selectedState[0] is set to TimeReceiverPort.

h) Computes the gmPriorityVector as follows:
   1) If selectedState[j] is set to TimeReceiverPort, the gmPriorityVector is set equal to messagePriorityPAIE for PTP Port j.
   2) If selectedState[j] is *not* set to TimeReceiverPort and selectedState[k] is not equal to TimeReceiverPort for every k not equal to 0 or j, the gmPriorityVector is set equal to the systemPriorityVector.

i) Update the parentDS members grandmasterIdentity, grandmasterClockQuality.clockClass, grandmasterClockQuality.clockAccuracy, grandmasterClockQuality.offsetScaledLogVariance, grandmasterPriority1, grandmasterPriority2 with the clockIdentity, clockClass, clockAccuracy, offsetScaledLogVariance, priority1, and priority2 attributes, respectively, of the systemIdentity component of the gmPrioirityVector computed in item i) above.

j) Update the parentDS member parentPortIdentity with the sourcePortIdentity component of the gmPriorityVector computed in item i) above.

k) Computes the timeTransmitterPriorityVector for PTP Port j.

l) If no PTP Port of this PTP Instance has the PTP Port state TimeReceiverPort, the pathTrace array is set to contain the single element thisClock (see 10.2.4.22).

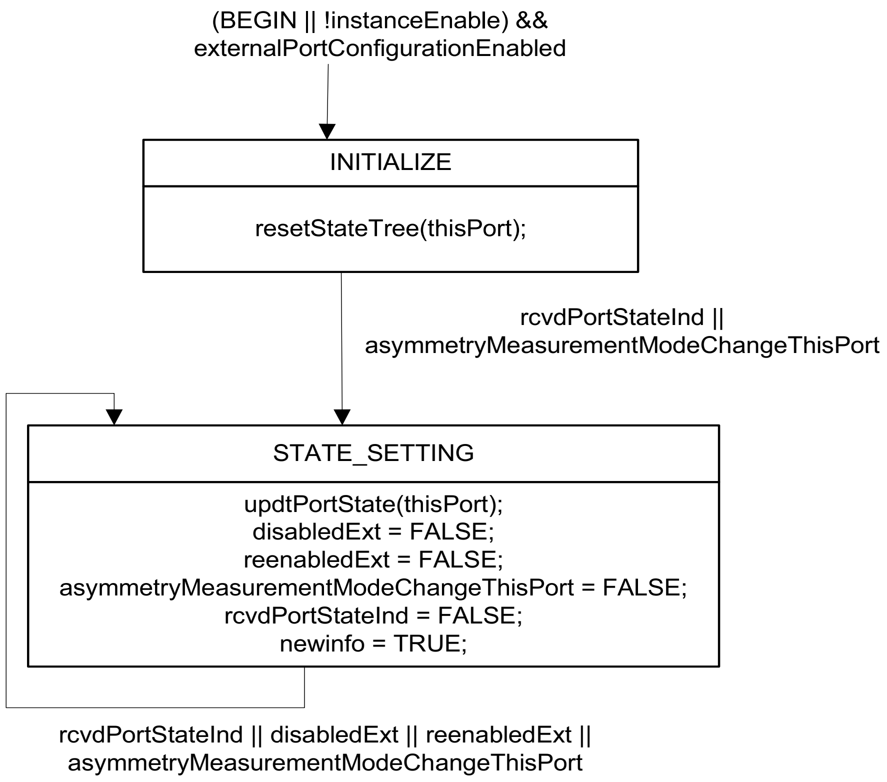
Change 10.3.15.3 as follows:

10.3.15.3 State diagram

The PortStateSettingExt state machine shall implement the function specified by the state diagram in Figure , the local variables specified in 10.3.15.1, the functions specified in 10.3.15.2, and the relevant global variables specified in 10.2.4, 10.2.5, 10.3.9, and 10.3.10~~, and 11.2.13~~. This state machine is used only if externalPortConfigurationEnabled is TRUE (if this variable is FALSE, the PortStateSelection state machine of 10.3.13.3 is used instead). A separate instance of this state machine runs on each PTP Port (unlike the PortStateSelection state machine, for which a single instance runs in the PTP Instance and performs operations on all the ports).

The state machine updates the gmPriorityVector for the PTP Instance and the timeTransmitterPriorityVector for each PTP Port of the PTP Instance. The state machine determines the PTP Port state for each PTP Port and updates gmPresent.

NOTE 1—It is possible to use the external port configuration mechanism to misconfigure the network, e.g., to produce a configuration where one or more PTP Instances have more than one timeReceiver port. Detecting and correcting misconfiguations is outside the scope of this standard..


(BEGIN || !instanceEnable) &&
externalPortConfigurationEnabled
INITIALIZE
resetStateTree(thisPort);
rcvdPortStateInd ||
asymmetryMeasurementModeChangeThisPort
STATE_SETTING
updtPortState(thisPort);
disabledExt = FALSE;
reenabledExt = FALSE;
asymmetryMeasurementModeChangeThisPort = FALSE;
rcvdPortStateInd = FALSE;
newinfo = TRUE;
rcvdPortStateInd || disabledExt || reenabledExt ||
asymmetryMeasurementModeChangeThisPort

NOTE 2—This figure differs from the 2020 edition, with Corrigendum 1 and P802.1ASdr applied, of this standard in that the statement newinfo = TRUE; is added after the existing statements in the STATE_SETTING state.

Figure 10-17—PortStateSettingExt state machine

10.3.16 PortAnnounceTransmit state machine

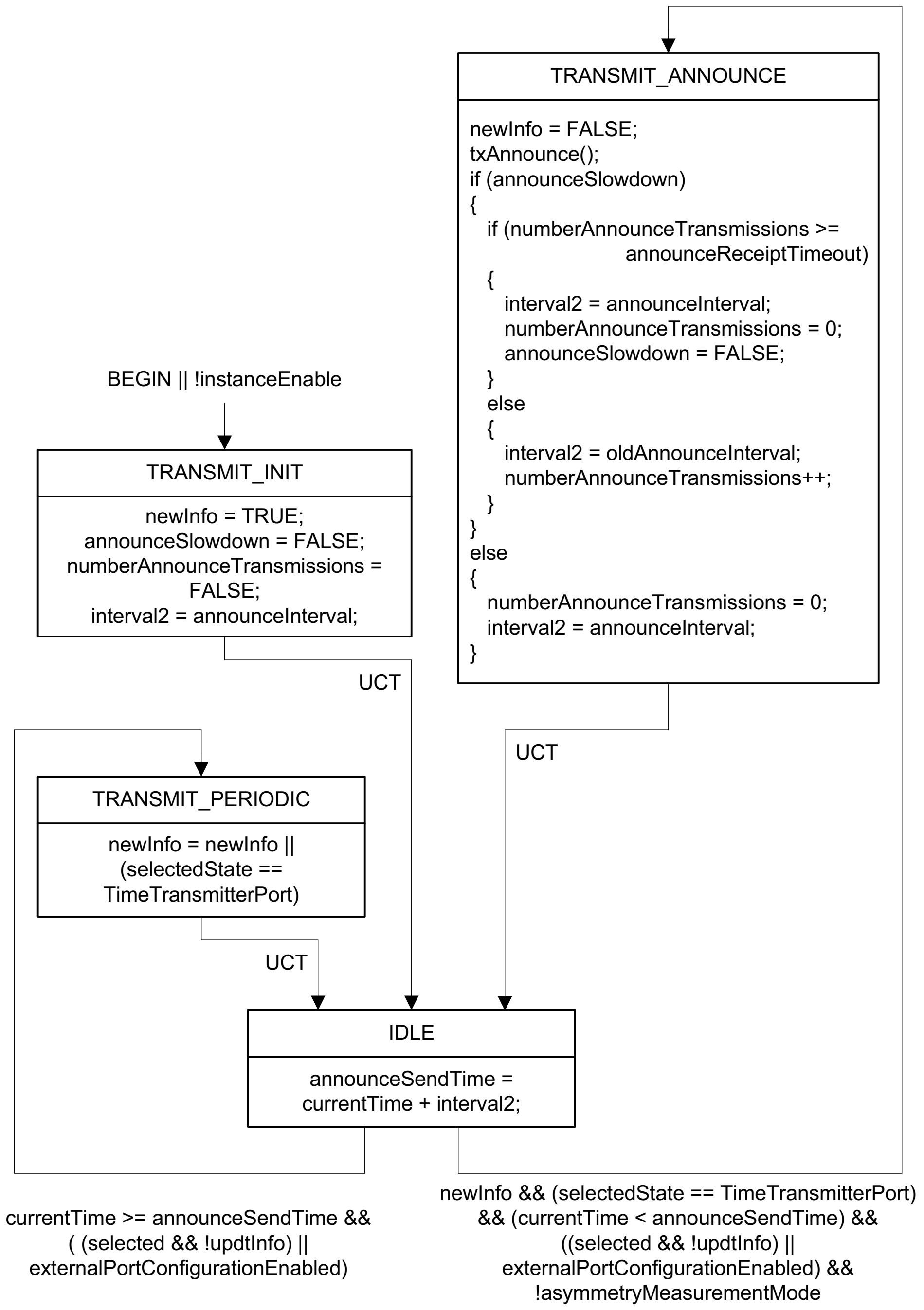
Change 10.3.16.3 as follows:

10.3.16.3 State diagram

The PortAnnounceTransmit state machine shall implement the function specified by the state diagram in Figure 10-17, the local variables specified in 10.3.16.1, the functions specified in 10.3.16.2, and the relevant global variables specified in 10.2.4, 10.2.5, 10.3.9, and 10.3.10~~, and 11.2.13~~. The state machine transmits Announce information to the MD entity when an announce interval has elapsed, PTP Port states have been updated, and portPriority and portStepsRemoved information has been updated with newly determined timeTransmitterPriority and timeTransmitterStepsRemoved information.

NOTE 1—When the external port configuration option is used (i.e., externalPortConfigurationEnabled is TRUE; see 10.3.9.24) the values of the variables updtInfo and selected do not affect the operation of the PortAnnounceTransmit state machine because the term of the conditions in which they appear, i.e., (selected && !updtInfo) || externalPortConfiguartionEnabled, evaluates to TRUE when externalPortConfigurationEnabled is TRUE.

Replace Figure 10-18 with the following:


TRANSMIT_ANNOUNCE
newInfo = FALSE;
txAnnounce();
if (announceSlowdown)
{
if (numberAnnounceTransmissions >= announceReceiptTimeout)
{
interval2 = announceInterval;
numberAnnounceTransmissions = 0;
announceSlowdown = FALSE;
}
else
{
interval2 = oldAnnounceInterval;
numberAnnounceTransmissions++;
}
}
else
{
numberAnnounceTransmissions = 0;
interval2 = announceInterval;
}
BEGIN || !instanceEnable
TRANSMIT_INIT
newInfo = TRUE;
announceSlowdown = FALSE;
numberAnnounceTransmissions = FALSE;
interval2 = announceInterval;
UCT
UCT
TRANSMIT_PERIODIC
newInfo = newInfo || (selectedState == TimeTransmitterPort)
UCT
IDLE
announceSendTime = currentTime + interval2;
currentTime >= announceSendTime && ((selected && !updtInfo) || externalPortConfigurationEnabled)
newInfo && (selectedState == TimeTransmitterPort) && (currentTime < announceSendTime) && ((selected && !updtInfo) || externalPortConfigurationEnabled) && !asymmetryMeasurementMode

NOTE 2—This figure differs from the 2020 edition, with Corrigendum 1 and P802.1ASdr applied, of this standard in that the statement interval2 = announceInterval is added after the existing statements in the TRANSMIT_INIT state.

Figure 10-18—PortAnnounceTransmit state machine

10.3.17 AnnounceIntervalSetting state machine

Add the following definition to 10.3.17.1:

10.3.17.1 State machine variables

10.3.17.1.6 TEMP: A temporary variable used to reduce clutter in the state diagram (see Figure 10-19). The data type for TEMP is Integer16.

10.3.17.2 State machine functions

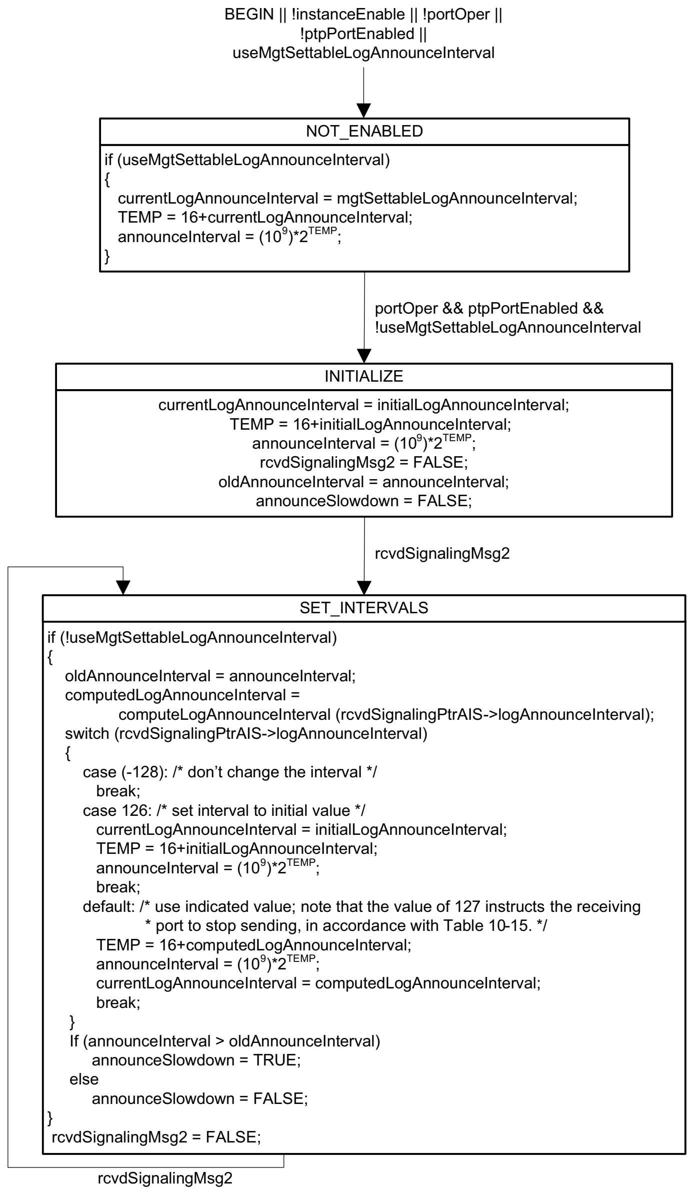
Change 10.3.17.2.2 as follows:

10.3.17.2.2 computeLogAnnounceInterval (logRequestedAnnounceInterval): An Integer8 function that computes and returns the logAnnounceInterval, based on the logRequestedAnnounceInterval. 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 computeLogAnnounceInterval (logRequestedAnnounceInterval)
Integer8 logRequestedAnnounceInterval;
{
        Integer8 logSupportedAnnounceIntervalMax,
                        logSupportedClosestLongerAnnounceInterval;
        if (isSupportedLogAnnounceInterval (logRequestedAnnounceInterval))
                // The requested Announce Interval is supported and returned
                return (logRequestedAnnounceInterval);
        else
        {
                if (logRequestedAnnounceInterval > logSupportedAnnounceIntervalMax)
                        // Return the ~~fasted~~largest supported ~~rate~~logAnnounceInterval, even if
~~faster~~smaller than the requested ~~rate~~interval
                        return (logSupportedAnnounceIntervalMax);
                else
                        // Return the ~~fastest~~smallest supported ~~rate~~logAnnounceInterval that is
still ~~slower~~larger than
                        // the requested ~~rate~~interval.
                        return (logSupportedClosestLongerAnnounceInterval);
        }
}
```

10.3.17.3 State diagram

Replace Figure 10-19 with the following:


BEGIN || !instanceEnable || !portOper || !ptpPortEnabled || useMgtSettableLogAnnounceInterval
NOT_ENABLED
portOper && ptpPortEnabled && !useMgtSettableLogAnnounceInterval
INITIALIZE
rcvdSignalingMsg2
SET_INTERVALS
rcvdSignalingMsg2

```
if (useMgtSettableLogAnnounceInterval)
{
   currentLogAnnounceInterval = mgtSettableLogAnnounceInterval;
   TEMP = 16+currentLogAnnounceInterval;
   announceInterval = (10^9)*2^TEMP;
}
```

```
currentLogAnnounceInterval = initialLogAnnounceInterval;
TEMP = 16+initialLogAnnounceInterval;
announceInterval = (10^9)*2^TEMP;
rcvdSignalingMsg2 = FALSE;
oldAnnounceInterval = announceInterval;
announceSlowdown = FALSE;
```

```
if (!useMgtSettableLogAnnounceInterval)
{
   oldAnnounceInterval = announceInterval;
   computedLogAnnounceInterval =
            computeLogAnnounceInterval (rcvdSignalingPtrAIS->logAnnounceInterval);
   switch (rcvdSignalingPtrAIS->logAnnounceInterval)
   {
      case (-128): /* don't change the interval */
         break;
      case 126: /* set interval to initial value */
         currentLogAnnounceInterval = initialLogAnnounceInterval;
         TEMP = 16+initialLogAnnounceInterval;
         announceInterval = (10^9)*2^TEMP;
         break;
      default: /* use indicated value; note that the value of 127 instructs the receiving
                * port to stop sending, in accordance with Table 10-15. */
         TEMP = 16+computedLogAnnounceInterval;
         announceInterval = (10^9)*2^TEMP;
         currentLogAnnounceInterval = computedLogAnnounceInterval;
         break;
   }
   If (announceInterval > oldAnnounceInterval)
      announceSlowdown = TRUE;
   else
      announceSlowdown = FALSE;
}
rcvdSignalingMsg2 = FALSE;
```

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the variable LogAnnounceInterval at the end of line 5 of the SET_INTERVALS state is changed to logAnnounceInterval (i.e., the capitalization is corrected), and the statement oldAnnounceInterval = announceInterval is added immediately after the first open curly brace in the SET_INTERVALS state.

Figure 10-19—AnnounceIntervalSetting state machine

10.3.18 SyncIntervalSetting state machine

Add the following definition to 10.3.18.1:

10.3.18.1 State machine variables

10.3.18.1.6 TEMP: A temporary variable used to reduce clutter in the state diagram (see Figure). The data type for TEMP is Integer16.

10.3.18.2 State machine functions

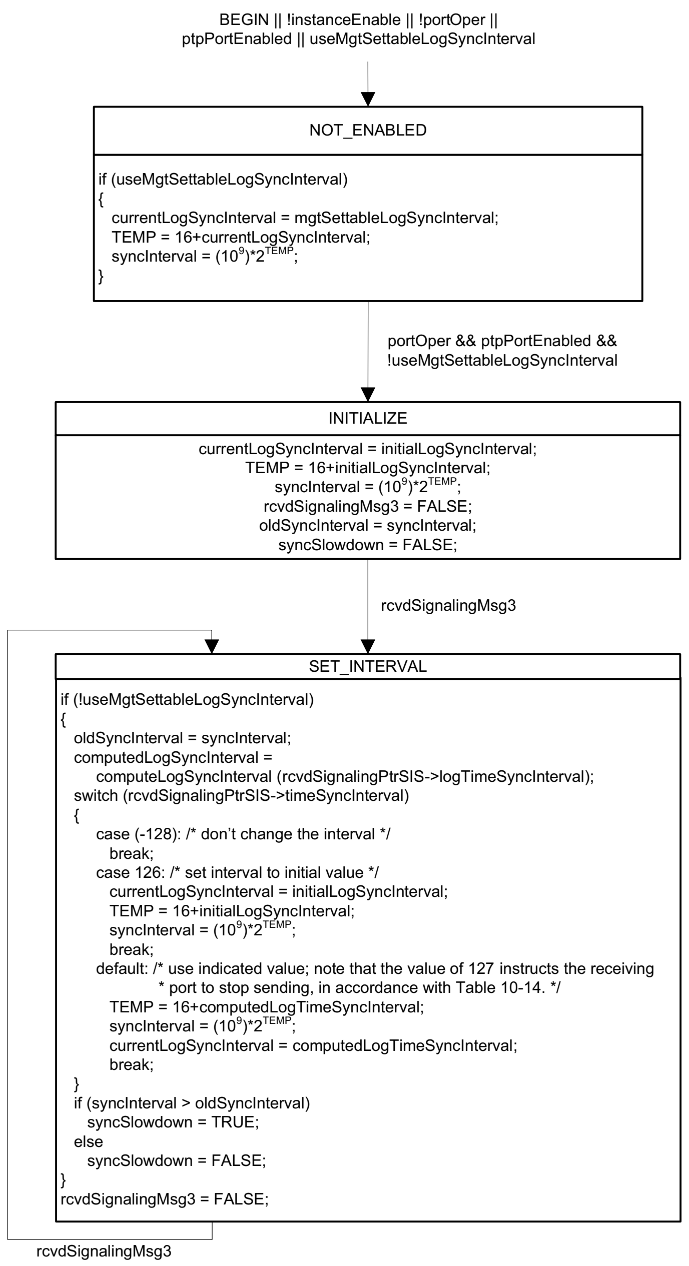
Change 10.3.18.2.2 as follows:

10.3.18.2.2 computeLogSyncInterval (logRequestedSyncInterval): An Integer8 function that computes and returns the logSyncInterval, based on the logRequestedSyncInterval. 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 computeLogSyncInterval (logRequestedSyncInterval)
Integer8 logRequestedSyncInterval;
{
    Integer8 logSupportedSyncIntervalMax, logSupportedClosestLongerSyncInterval;
    if (isSupportedLogSyncInterval (logRequestedSyncInterval))
        // The requested Sync Interval is supported and returned
        return (logRequestedSyncInterval);
    else
    {
        if (logRequestedSyncInterval > logSupportedSyncIntervalMax)
            // Return the ~~fasted~~largest supported ~~rate~~logSyncInterval, even if ~~faster~~ smaller than the requested ~~rate~~interval
            return (logSupportedSyncIntervalMax);
        else
            // Return the ~~fastest~~smallest supported ~~rate~~logSyncInterval that is still ~~slower~~larger than
            // the requested ~~rate~~interval.
            return (logSupportedClosestLongerSyncInterval);
    }
}
```

10.3.18.3 State diagram

Replace Figure 10-20 with the following:


BEGIN || !instanceEnable || !portOper ||
ptpPortEnabled || useMgtSettableLogSyncInterval
NOT_ENABLED
portOper && ptpPortEnabled &&
!useMgtSettableLogSyncInterval
INITIALIZE
rcvdSignalingMsg3
SET_INTERVAL
rcvdSignalingMsg3

NOT_ENABLED

```
if (useMgtSettableLogSyncInterval)
{
   currentLogSyncInterval = mgtSettableLogSyncInterval;
   TEMP = 16+currentLogSyncInterval;
   syncInterval = $(10^9)*2^{TEMP}$;
}
```

INITIALIZE

```
currentLogSyncInterval = initialLogSyncInterval;
TEMP = 16+initialLogSyncInterval;
syncInterval = $(10^9)*2^{TEMP}$;
rcvdSignalingMsg3 = FALSE;
oldSyncInterval = syncInterval;
syncSlowdown = FALSE;
```

SET_INTERVAL

```
if (!useMgtSettableLogSyncInterval)
{
   oldSyncInterval = syncInterval;
   computedLogSyncInterval =
        computeLogSyncInterval (rcvdSignalingPtrSIS->logTimeSyncInterval);
   switch (rcvdSignalingPtrSIS->timeSyncInterval)
   {
        case (-128): /* don't change the interval */
           break;
        case 126: /* set interval to initial value */
           currentLogSyncInterval = initialLogSyncInterval;
           TEMP = 16+initialLogSyncInterval;
           syncInterval = $(10^9)*2^{TEMP}$;
           break;
        default: /* use indicated value; note that the value of 127 instructs the receiving
                  * port to stop sending, in accordance with Table 10-14. */
           TEMP = 16+computedLogTimeSyncInterval;
           syncInterval = $(10^9)*2^{TEMP}$;
           currentLogSyncInterval = computedLogTimeSyncInterval;
           break;
   }
   if (syncInterval > oldSyncInterval)
      syncSlowdown = TRUE;
   else
      syncSlowdown = FALSE;
}
rcvdSignalingMsg3 = FALSE;
```

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the statement oldSyncInterval = syncInterval is added immediately after the first open curly brace in the SET_INTERVALS state

Figure 10-20—SyncIntervalSetting state machine

Delete the NOTE immediately after Figure 10-20.

10.4 State machines related to signaling gPTP capability

Insert 10.4.1, and renumber subsequent subclauses as appropriate:

10.4.1 Enabling and disabling the state machines

If the managed object gptpCapableStateMachinesEnabled (see 14.8.55) is TRUE, the GptpCapableTransmit and GptpCapableReceive state machines shall be enabled and shall function as specified in 10.4.2 and 10.4.3, respectively. The GptpCapableIntervalSetting state machine is enabled if it is implemented.

If the managed object gptpCapableStateMachinesEnabled is FALSE, the GptpCapableTransmit and GptpCapableReceive state machines shall be disabled. The GptpCapableIntervalSetting state machine shall be disabled if it is implemented.

If the managed object gptpCapableStateMachinesEnabled is FALSE, the global variable neighborGptpCapable for the port (see 10.2.5.16) shall be set to TRUE.

NOTE 1—The global variable neighborGptpCapable indicates to a PTP Port whether the PTP Port at the other end of the attached PTP Link is capable of invoking gPTP, and is used in the determination of asCapable (see 10.2.5.1, 11.2.2, 12.4, and 13.4). If gptpCapableStateMachinesEnabled is TRUE, the variable is set to TRUE or FALSE by the GptpCapableTransmit state machine. If gptpCapableStateMachinesEnabled is FALSE, the variable is automatically set to TRUE, i.e., it is assumed that the network is engineered such that the PTP Port at the other end of the attached link is capable of invoking gPTP. In this case, it is essential that the network be engineered to fulfill this condition; if neighborGptpCapable is TRUE and the PTP Port at the other end of the link is not capable of invoking gPTP, undesirable behavior can occur.

NOTE 2—If the GptpCapableTransmit and GptpCapableReceive state machines are disabled, the exchange of peer delay messages can occur immediately, i.e., it is not necessary to wait until gPTP capability is determined by the GptpCapableTransmit and GptpCapableReceive state machines.

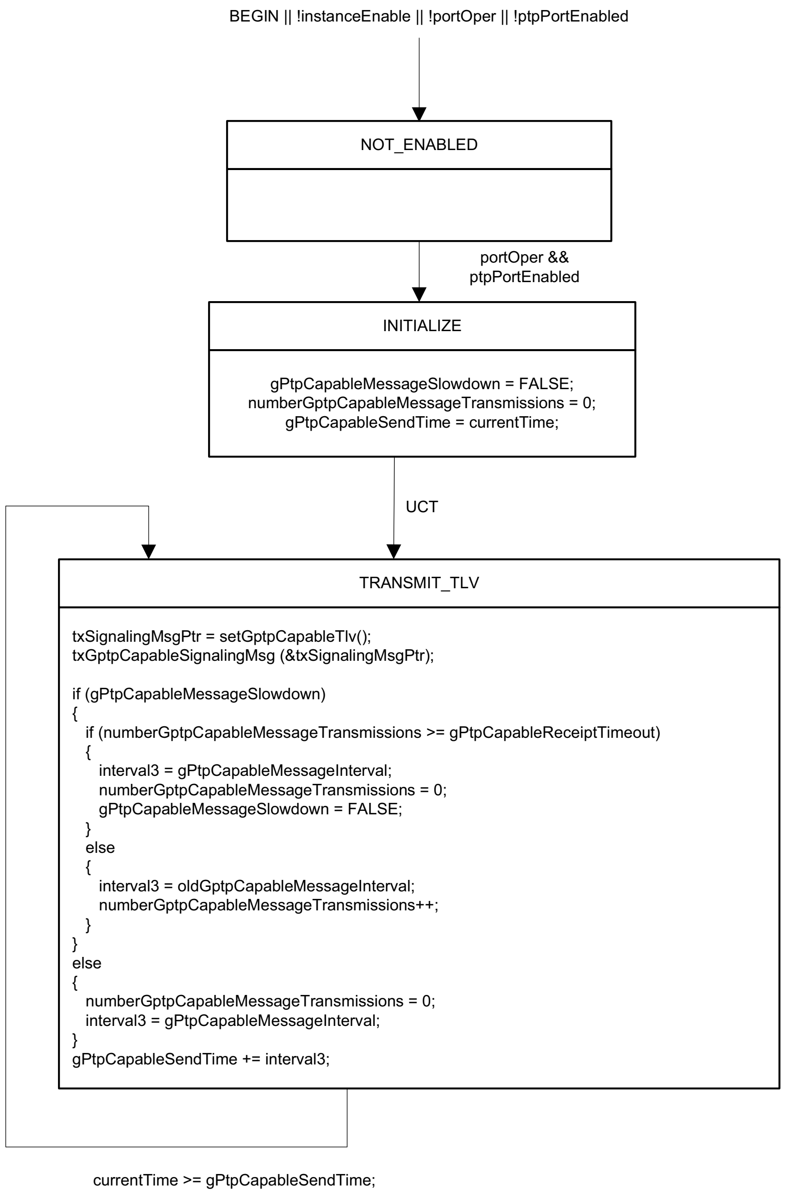
10.4.2 GptpCapableTransmit state machine

10.4.2.1 State machine variables

Change 10.4.2.1.1 as follows:

10.4.2.1.1 ~~intervalTimer~~gPtpCapableSendTime: ~~A variable used to save the~~The time, relative to the LocalClock entity, ~~at which~~when ~~the gPTP-capable message interval timer is set (see Figure 10-21). A~~a Signaling message containing a gPTP-capable TLV is next sent~~ when this timer expires~~. The data type for gPtpCapableSendTime~~intervalTimer~~ is UScaledNs.

Replace Figure 10-21 with the following:


BEGIN || !instanceEnable || !portOper || !ptpPortEnabled
NOT_ENABLED
portOper &&
ptpPortEnabled
INITIALIZE
UCT
TRANSMIT_TLV
currentTime >= gPtpCapableSendTime;

INITIALIZE:

```
gPtpCapableMessageSlowdown = FALSE;
numberGptpCapableMessageTransmissions = 0;
gPtpCapableSendTime = currentTime;
```

TRANSMIT_TLV:

```
txSignalingMsgPtr = setGptpCapableTlv();
txGptpCapableSignalingMsg (&txSignalingMsgPtr);

if (gPtpCapableMessageSlowdown)
{
  if (numberGptpCapableMessageTransmissions >= gPtpCapableReceiptTimeout)
  {
    interval3 = gPtpCapableMessageInterval;
    numberGptpCapableMessageTransmissions = 0;
    gPtpCapableMessageSlowdown = FALSE;
  }
  else
  {
    interval3 = oldGptpCapableMessageInterval;
    numberGptpCapableMessageTransmissions++;
  }
}
else
{
  numberGptpCapableMessageTransmissions = 0;
  interval3 = gPtpCapableMessageInterval;
}
gPtpCapableSendTime += interval3;
```

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the condition !domainEnabled is removed from the set of logical ORs at the ingress of the NOT_ENABLED state and the variable intervalTimer is renamed gPtpCapableSendTime.

Figure 10-21—GptpCapableTransmit state machine

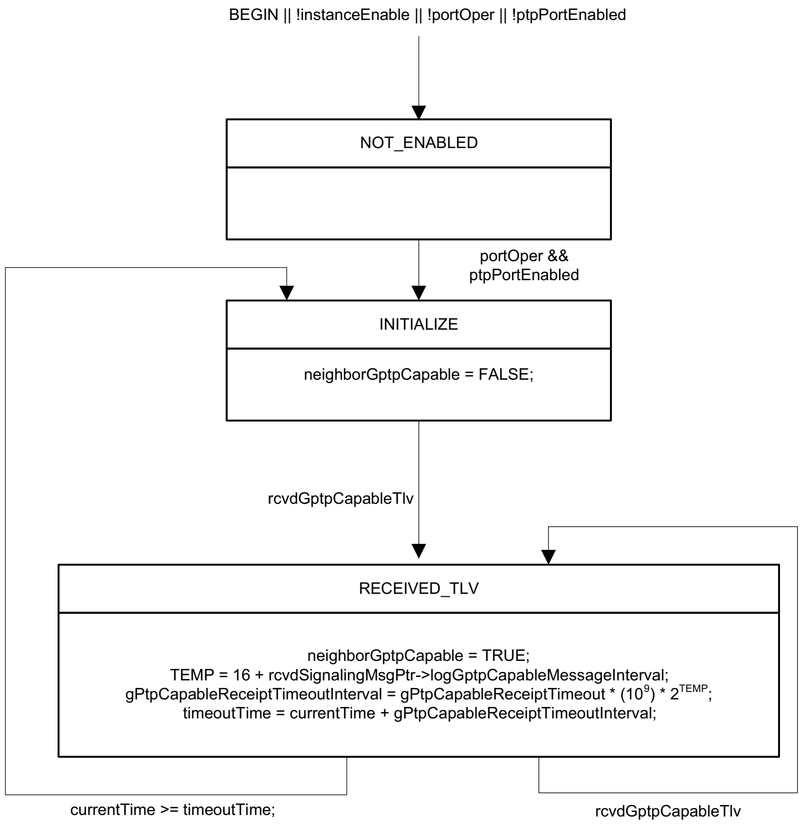
10.4.3 GptpCapableReceive sate machine

Add the following definition to 10.4.3.1:

10.4.3.1 State machine variables

10.4.3.1.5 TEMP: A temporary variable used to reduce clutter in the state diagram (see Figure). The data type for TEMP is Integer16.

Replace Figure 10-22 with the following:


BEGIN || !instanceEnable || !portOper || !ptpPortEnabled
NOT_ENABLED
portOper && ptpPortEnabled
INITIALIZE
neighborGptpCapable = FALSE;
rcvdGptpCapableTlv
RECEIVED_TLV
neighborGptpCapable = TRUE;
TEMP = 16 + rcvdSignalingMsgPtr->logGptpCapableMessageInterval;
gPtpCapableReceiptTimeoutInterval = gPtpCapableReceiptTimeout * (10^9) * 2^TEMP;
timeoutTime = currentTime + gPtpCapableReceiptTimeoutInterval;
currentTime >= timeoutTime;
rcvdGptpCapableTlv

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the condition !domainEnabled is removed from the set of logical ORs at the ingress of the NOT_ENABLED state.

Figure 10-22—GptpCapableReceive state machine

10.4.4 GptpCapableIntervalSetting state machine

Add the following definition to 10.4.4.1:

10.4.4.1 State machine variables

10.4.4.1.6 TEMP: A temporary variable used to reduce clutter in the state diagram (see Figure). The data type for TEMP is Integer16.

10.4.4.2 State machine functions

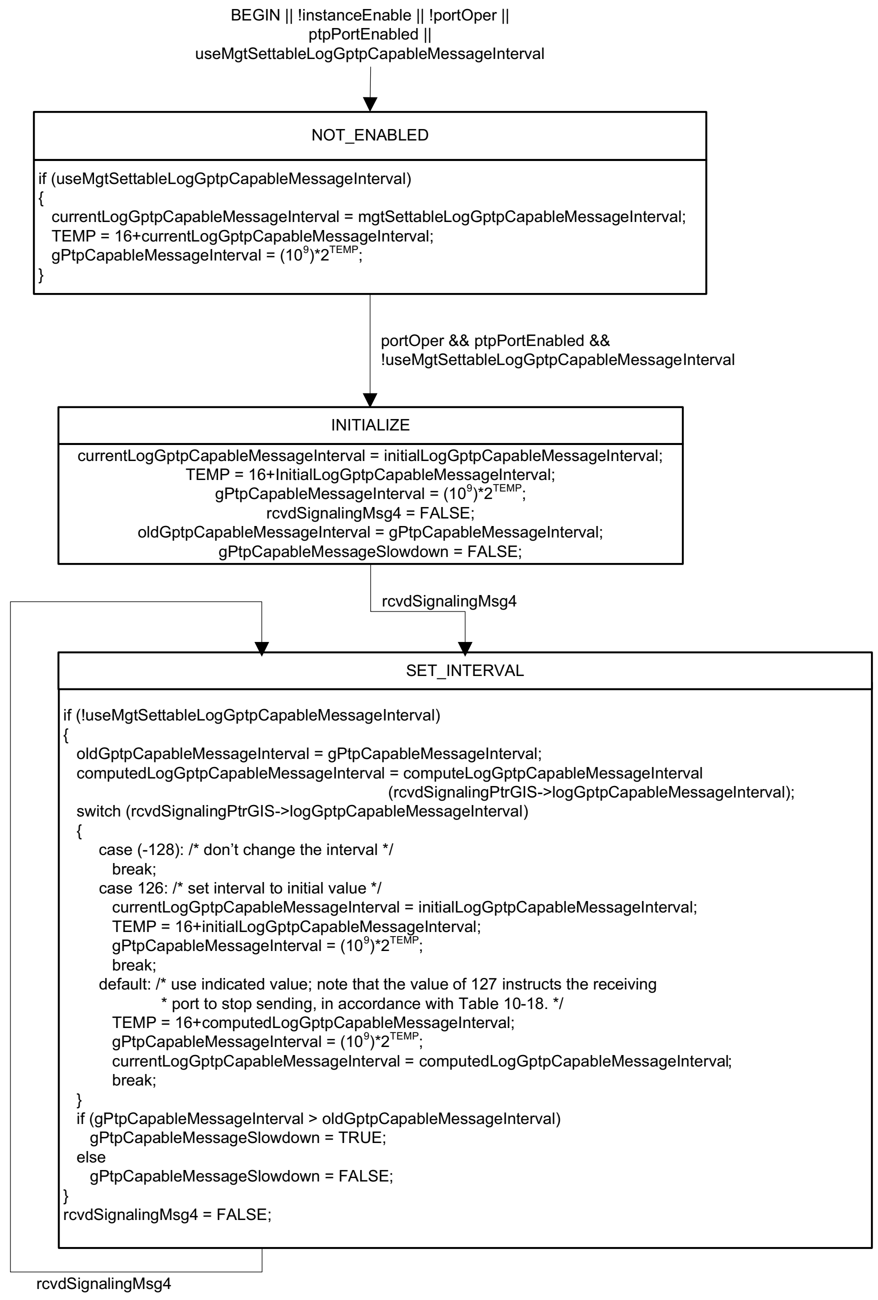
Change 10.4.4.2.2 as follows:

10.4.4.2.2 computeLogGptpCapableMessageInterval (logRequestedGptpCapableMessageInterval): An Integer8 function that computes and returns the logGptpCapableMessageInterval, based on the logRequestedGptpCapableMessageInterval. 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 computeLogGptpCapableMessageInterval (logRequestedGptpCapableMessageInterval)
Integer8 logRequestedGptpCapableMessageInterval;
{
        Integer8 logSupportedGptpCapableMessageIntervalMax,
                        logSupportedClosestLongerGptpCapableMessageInterval;
        if (isSupportedLogGptpCapableMessageInterval
                                (logRequestedGptpCapableMessageInterval))
                // The requested gPTP-capable Message Interval is supported and returned
                return (logRequestedGptpCapableMessageInterval);
        else
        {
                if (logRequestedGptpCapableMessageInterval >
                                logSupportedGptpCapableMessageIntervalMax)
                        // Return the ~~fasted~~largest supported ~~rate~~logGptpCapableMessageInterval, even if ~~faster~~smaller than the requested ~~rate~~interval
                        return (logSupportedGptpCapableMessageIntervalMax);
                else
                        // Return the ~~fastest~~smallest supported ~~rate~~logGptpCapableMessageInterval that is still ~~slower~~larger than
                        // the requested ~~rate~~interval.
                        return (logSupportedClosestLongerGptpCapableMessageInterval);
        }
}
```

10.4.4.3 State diagram

Replace Figure 10-23 with the following:


BEGIN || !instanceEnable || !portOper || ptpPortEnabled || useMgtSettableLogGptpCapableMessageInterval
NOT_ENABLED
portOper && ptpPortEnabled && !useMgtSettableLogGptpCapableMessageInterval
INITIALIZE
rcvdSignalingMsg4
SET_INTERVAL
rcvdSignalingMsg4

NOT_ENABLED

```
if (useMgtSettableLogGptpCapableMessageInterval)
{
   currentLogGptpCapableMessageInterval = mgtSettableLogGptpCapableMessageInterval;
   TEMP = 16+currentLogGptpCapableMessageInterval;
   gPtpCapableMessageInterval = (10^9)*2^TEMP;
}
```

INITIALIZE

```
currentLogGptpCapableMessageInterval = initialLogGptpCapableMessageInterval;
TEMP = 16+InitialLogGptpCapableMessageInterval;
gPtpCapableMessageInterval = (10^9)*2^TEMP;
rcvdSignalingMsg4 = FALSE;
oldGptpCapableMessageInterval = gPtpCapableMessageInterval;
gPtpCapableMessageSlowdown = FALSE;
```

SET_INTERVAL

```
if (!useMgtSettableLogGptpCapableMessageInterval)
{
   oldGptpCapableMessageInterval = gPtpCapableMessageInterval;
   computedLogGptpCapableMessageInterval = computeLogGptpCapableMessageInterval
                                              (rcvdSignalingPtrGIS->logGptpCapableMessageInterval);
   switch (rcvdSignalingPtrGIS->logGptpCapableMessageInterval)
   {
      case (-128): /* don't change the interval */
         break;
      case 126: /* set interval to initial value */
         currentLogGptpCapableMessageInterval = initialLogGptpCapableMessageInterval;
         TEMP = 16+initialLogGptpCapableMessageInterval;
         gPtpCapableMessageInterval = (10^9)*2^TEMP;
         break;
      default: /* use indicated value; note that the value of 127 instructs the receiving
                * port to stop sending, in accordance with Table 10-18. */
         TEMP = 16+computedLogGptpCapableMessageInterval;
         gPtpCapableMessageInterval = (10^9)*2^TEMP;
         currentLogGptpCapableMessageInterval = computedLogGptpCapableMessageInterval;
         break;
   }
   if (gPtpCapableMessageInterval > oldGptpCapableMessageInterval)
     gPtpCapableMessageSlowdown = TRUE;
   else
     gPtpCapableMessageSlowdown = FALSE;
}
rcvdSignalingMsg4 = FALSE;
```

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the statement oldGptpCapableMessageInterval = gPtpCapableMessageInterval is added immediately after the first open curly brace in the SET_INTERVALS state.

Figure 10-23—GptpCapableIntervalSetting state machine

Delete the NOTE immediately after Figure 10-23.

10.5 Message attributes

10.5.3 Addresses

Add the following sentence, as a new paragraph, after the NOTE in 10.5.3.

If the transport is full-duplex IEEE 802.3, all Announce and Signaling messages shall use the MAC address of the respective egress physical port as the source address.

Change 10.5.5 as follows:

10.5.5 Subtype

The subtype ~~of~~for the Announce and Signaling 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.~~

10.6 Message formats

10.6.2 Header

10.6.2.2 Header field specifications

Change 10.6.2.2.6 as follows:

10.6.2.2.6 domainNumber (UInteger8)

10.6.2 The value is the gPTP ~~domain number~~domainNumber specified in 8.1.

10.6.2.2.8 flags (Octet2)

Change Table 10-9 as follows:

Table 10-9—Values of flag bits

Octet	Bit	Message types	Name	Value
0	0	All	alternatetimeTransmitterFlag in Announce, Sync, Follow_Up, and Delay_Resp messages	Not used in this standard; transmitted as FALSE and ignored on reception
0	1	Sync, Pdelay_Resp	twoStepFlag	*For Sync messages:* a) For a one-step transmitting PTP Port (see 11.1.3 and 11.2.13.9), the value is FALSE. b) For a two-step transmitting PTP Port, the value is TRUE. *For Pdelay_Resp messages:* The value is transmitted as TRUE and ignored on reception.
0	2	All	unicastFlag	Not used in this standard; transmitted as FALSE and ignored on reception
0	3	All	Reserved	Not used by IEEE Std 1588-2019; reserved as FALSE and ignored on reception
0	4	All	Reserved	Not used by IEEE Std 1588-2019; reserved as FALSE and ignored on reception
0	5	All	PTP profileSpecific 1	Not used in this standard; transmitted as FALSE and ignored on reception
0	6	All	PTP profileSpecific 2	Not used in this standard; transmitted as FALSE and ignored on reception
0	7	All	Reserved	Not used in this standard; transmitted as FALSE and ignored on reception
1	0	Announce	leap61	The value of the global variable leap61 (see 10.3.9.4)
1	1	Announce	leap59	The value of the global variable leap59 (see 10.3.9.5)
1	2	Announce	currentUtcOffsetValid	The value of the global variable currentUtcOffsetValid (see 10.3.9.6)
1	3	Announce	ptpTimescale	For domain 0, transmitted as TRUE and ignored on receipt. For domains other than domain 0, ~~T~~the value of the global variable ptpTimescale (see 10.3.9.7)
1	4	Announce	timeTraceable	The value of the global variable timeTraceable (see 10.3.9.8)

Table 10-9—Values of flag bits *(continued)*

Octet	Bit	Message types	Name	Value
1	5	Announce	frequencyTraceable	The value of the global variable frequencyTraceable (see 10.3.9.9)
1	6	All	Reserved	Not used by IEEE Std 1588-2019; reserved as FALSE and ignored on reception
1	7	All	Reserved	Not used in this standard; reserved as FALSE and ignored on reception

10.6.2.2.10 messageTypeSpecific (Octet4)

Change Table 10-10 as follows:

Table 10-10—messageTypeSpecific semantics

Value of messageType	Description
Follow_Up, Pdelay_Resp_Follow_Up, Announce, Signaling, Management	For the General message class, this field is reserved; it is transmitted as 0 and ignored on reception.
Sync, ~~Delay_Req~~Pdelay_Req, Pdelay_Resp	For the Event message class, this field may be used for internal implementation as specified in this subclause.

10.6.4 Signaling message

Change 10.6.4.1 as follows:

10.6.4.1 General Signaling message specifications

The fields of the body of the Signaling message shall be as specified in .Table 10-13 and 10.6.4.2.

Change 10.6.4.2.2 as follows:

Table 10-13—Signaling message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 10.6.2)								34	0
targetPortIdentity								10	34
~~message interval request TLV, gPTP-capable TLV, or gPTP-capable message interval request TLV~~one or more TLVs								~~16~~N	44

10.6.4.2.2 ~~Message interval request TLV or gPTP-capable TLV~~TLVs carried in one Signaling message

The Signaling message carries either: ~~the message interval request TLV, defined in 10.6.4.3, or the gPTP-capable TLV, defined in 10.6.4.4, but not both. If it is desired to send both TLVs, two Signaling messages must be sent.~~

a) one or both interval request TLVs, defined in 10.6.4.3 and 10.6.4.5, or

b) the gPTP capable TLV, defined in 10.6.4.4.

Change 10.6.4.5.6 as follows:

10.6.4.5.6 logGptpCapableMessageInterval (Integer8)

The value is the logarithm to base 2 of the mean time interval, desired by the PTP Port that sends this TLV, between successive Signaling messages that contain the gPTP-capable TLV (see 10.6.4.4), sent by the PTP Port at the other end of the link. The format and allowed values of logGptpCapableMessageInterval are the same as the format and allowed values of initialLogGptpCapableMessageInterval (see 10.7.2.5).

The values 127, 126, and –128 are interpreted as defined in Table 10-14.

Table 10-14—Interpretation of special values of logGptpCapableMessageInterval

Value	Instruction to PTP Instance that receives this TLV
127	Instructs the PTP Port that receives this TLV to stop sending Signaling messages that contain the gPTP-capable TLV.
126	Instructs the PTP Port that receives this TLV to set currentlogGptpCapableMessageInterval to the value of initialLogGptpCapableMessageInterval (see 10.7.2.4).
–128	Instructs the PTP Port that receives this TLV not to change the mean time interval between successive Signaling messages that contain the gPTP-capable TLV.
All values in the ranges [–127, –25] and [25, 125] are reserved.	

When a Signaling message that contains this TLV is sent by a PTP Port, the value of gPtpCapableReceiptTimeoutTimeInterval for that PTP Port (see 10.3.10.1) shall be set equal to ~~p~~gPtpCapableReceiptTimeout (see 10.7.3.3) multiplied by the value of the interval, in seconds, reflected by logGptpCapableMessageInterval.

10.7 Protocol timing characterization

10.7.2 Message transmission intervals

Change 10.7.2.2 as follows:

10.7.2.2 Announce message transmission interval

The logarithm to the base 2 of the announce interval (in seconds) is carried in the logMessageInterval field of the Announce message.

When useMgtSettableLogAnnounceInterval (see 14.8.14) is FALSE, the initialLogAnnounceInterval specifies the announce interval when the PTP Port is initialized and the value to which the announce interval is set when a message interval request TLV is received with the logAnnounceInterval field set to 126 (see the AnnounceIntervalSetting state machine in 10.3.17). The currentLogAnnounceInterval specifies the current value of the announce interval. The default value of initialLogAnnounceInterval is 0. Every PTP Port supports the value 127; the PTP Port does not send Announce messages when currentLogAnnounceInterval has this value (see 10.3.17). A PTP Port may support other values, except for the reserved values indicated in Table 10-17. A PTP Port ignores requests for unsupported values (see 10.3.17). The initialLogAnnounceInterval and currentLogAnnounceInterval are per-PTP Port attributes.

When useMgtSettableLogAnnounceInterval is TRUE, currentLogAnnounceInterval is set equal to mgtSettableLogAnnounceInterval (see 14.8.15), and initialLogAnnounceInterval is ignored.

Announce messages shall be transmitted such that the value of the arithmetic mean of the intervals, in seconds, between message transmissions is within ± 30% of $2^{\text{currentLogAnnounceInterval}}$. In addition, a PTP Port shall transmit Announce messages such that at least 90% of the inter-message intervals are within ± 30% of the value of $2^{\text{currentLogAnnounceInterval}}$. The interval between successive Announce messages should not exceed twice the value of $2^{\cancel{\text{portDS.}}\text{currentL}\cancel{\text{l}}\text{ogAnnounceInterval}}$ in order to prevent causing an announceReceiptTimeout event. The PortAnnounceTransmit state machine (see 10.3.16) is consistent with these requirements, i.e., the requirements here and the requirements of the PortAnnounceTransmit state machine can be met simultaneously.

NOTE 1—A minimum number of inter-message intervals is necessary in order to verify that a PTP Port meets these requirements. The arithmetic mean is the sum of the inter-message interval samples divided by the number of samples. For more detailed discussion of statistical analyses, see Papoulis [B25].

NOTE 2—If useMgtSettableLogAnnounceInterval is FALSE, the value of initialLogAnnounceInterval is the value of the mean time interval between successive Announce messages when the PTP Port is initialized. The value of the mean time interval between successive Announce messages can be changed, e.g., if the PTP Port receives a Signaling message that carries a message interval request TLV (see 10.6.4.3) and the current value is stored in currentLogAnnounceInterval. The value of the mean time interval between successive Announce messages can be reset to the initial value, e.g., by a message interval request TLV for which the value of the field logAnnounceInterval is 126 (see 10.6.4.3.8).

NOTE 3—A PTP Port that requests (using a Signaling message that contains a message interval request TLV; see 10.6.4 and 10.3.17) that the PTP Port at the other end of the attached link set its currentLogAnnounceInterval to a specific value can determine if the request was honored by examining the logMessageInterval field of subsequent received Announce messages.

Change 10.7.2.3 as follows:

10.7.2.3 Time-synchronization event message transmission interval

The logarithm to the base 2 of the sync interval (in seconds) is carried in the logMessageInterval field of the time-synchronization messages.

When useMgtSettableLogSyncInterval (see 14.8.19) is FALSE, the initialLogSyncInterval specifies the sync interval when the PTP Port is initialized and the value to which the sync interval is set when a message interval request TLV is received with the logTimeSyncInterval field set to 126 (see the SyncIntervalSetting state machine in 10.3.18). The default value is media-dependent; the value is specified in the respective media-dependent clauses. The initialLogSyncInterval is a per-PTP Port attribute.

The currentLogSyncInterval specifies the current value of the sync interval and is a per-PTP Port attribute.

When useMgtSettableLogSyncInterval is TRUE, currentLogSyncInterval is set equal to mgtSettableLogSyncInterval (see 14.8.20), and initialLogSyncInterval is ignored.

When the value of syncLocked is FALSE, time-synchronization messages shall be transmitted such that the value of the arithmetic mean of the intervals, in seconds, between message transmissions is within ± 30% of $2^{\text{currentLogSyncInterval}}$. In addition, a PTP Port shall transmit time-synchronization messages such that at least 90% of the inter-message intervals are within ± 30% of the value of $2^{\text{currentLogSyncInterval}}$. The interval between successive time-synchronization messages should not exceed twice the value of $2^{\text{~~portDS.l~~currentLogSyncInterval}}$ in order to prevent causing a syncReceiptTimeout event. The PortSyncSyncSend state machine (see 10.2.12) is consistent with these requirements, i.e., the requirements here and the requirements of the PortSyncSyncSend state machine can be met simultaneously.

NOTE 1—A minimum number of inter-message intervals is necessary in order to verify that a PTP Port meets these requirements. The arithmetic mean is the sum of the inter-message interval samples divided by the number of samples. For more detailed discussion of statistical analyses, see Papoulis [B25].

NOTE 2—If useMgtSettableLogSyncInterval is FALSE the value of initialLogSyncInterval is the value of the sync interval when the PTP Port is initialized. The value of the sync interval can be changed, e.g., if the PTP Port receives a Signaling message that carries a message interval request TLV (see 10.6.4.3) and the current value is stored in currentLogSyncInterval. The value of the sync interval can be reset to the initial value, e.g., by a message interval request TLV for which the value of the field logTimeSyncInterval is 126 (see 10.6.4.3.7).

10.7.3 Timeouts

Change 10.7.3.1 as follows:

10.7.3.1 syncReceiptTimeout

The value of this attribute tells a timeReceiver port the number of sync intervals to wait without receiving synchronization information, before assuming that the timeTransmitter is no longer transmitting synchronization information and that the BTCA needs to be run, if appropriate. The condition of the timeReceiver port not receiving synchronization information for syncReceiptTimeout sync intervals is known as *sync receipt timeout.*

The default value shall be 3. The range shall be 2 through 255. The syncReceiptTimeout is a per-PTP Port attribute.

Change 10.7.3.2 as follows:

10.7.3.2 announceReceiptTimeout

The value of this attribute tells a timeReceiver port the number of announce intervals to wait without receiving an Announce message, before assuming that the timeTransmitter is no longer transmitting Announce messages, and that the BTCA needs to be run, if appropriate. The condition of the timeReceiver port not receiving an Announce message for announceReceiptTimeout announce intervals is known as *announce receipt timeout.*

The default value shall be 3. The range shall be 2 through 255. The announceReceiptTimeout is a per-PTP Port attribute.

Change 10.7.3.3 as follows:

10.7.3.3 gPtpCapableReceiptTimeout

The value of this attribute tells a PTP Port the number of gPTP-capable message intervals to wait without receiving from its neighbor a Signaling message containing a gPTP-capable TLV, before determining that its neighbor is no longer invoking gPTP.

NOTE—A determination that its neighbor is no longer invoking gPTP ~~will~~ causes the PTP Port to set asCapable to FALSE.

The default value shall be 9. The range shall be ~~1~~2 through 255.

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

11.1 Overview

11.1.1 Propagation delay measurement

Change the first paragraph of 11.1.1 as follows:

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 instance of the peer-to-peer delay mechanism that runs only on domain 0 is referred to as the transport-specific peer-to-peer delay mechanism. The instance of the peer-to-peer delay mechanism that runs as part of Common Mean Link Delay Service (CMLDS, see 11.2.17) is referred to as the CMLDS peer-to-peer delay mechansim. The CMLDS peer-to-peer delay mechanism is the same as the peer-to-peer delay mechanism described in IEEE Std 1588-2019, specialized to a two-step PTP Port[1] and sending the requestReceiptTimestamp and the responseOriginTimestamp separately [see item c) 8) of 11.4.2 in IEEE Std 1588-2019]. The transport-specific peer-to-peer delay mechanism is mathematically equivalent to the CMLDS peer-to-peer delay mechanism, though its computations are organized differently. The description in this subclause focuses on the transport-specific peer-to-peer delay mechanism. In addition, when it is not important in this standard to distinguish the transport-specific CMLDS peer-to-peer delay mechanisms, the mechanism is simply referred to as the "peer-to-peer delay mechansim," or more briefly as the "peer delay mechansim."

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.

Change the second paragraph of 11.1.1 as follows:

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 measurement ~~s~~ made using the peer-to-peer delay mechanism (i.e., nrrPdelay, see 11.2.13.13) 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.

Change the last paragraph of 11.1.1 as follows:

The rate ratio of the responder relative to the initiator is the quantity neighborRateRatio (see 10.2.5.7). When computed by the peer-to-peer delay mechansim, it~~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

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

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.

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

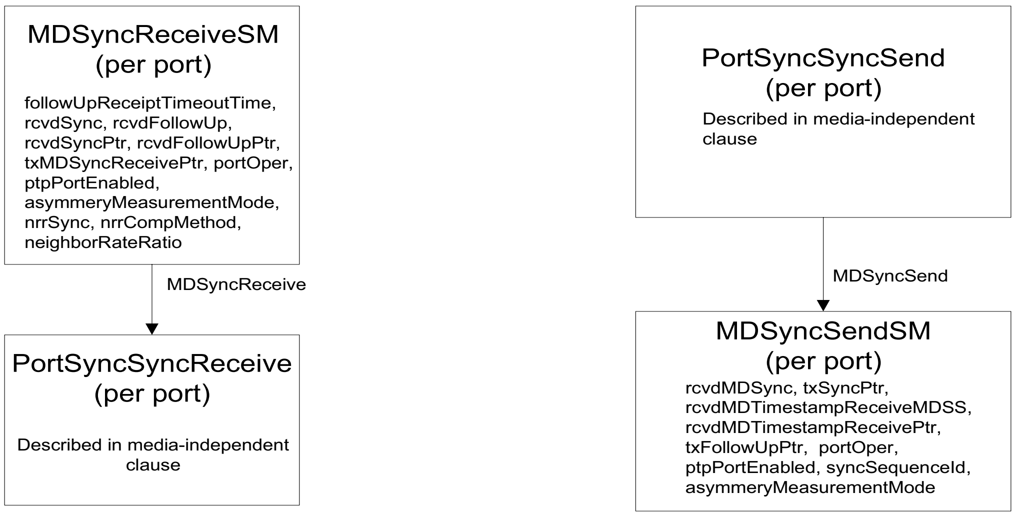
Change 11.2.1 as follows:

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. In addition, if the global variable driftTrackingTlvSupport is TRUE, this state machine computes 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 using the information contained in successive Drift_Tracking TLVs, and places the result in the global variable nrrSync (see 11.2.13.14). The global variable neighborRateRatio (see 10.2.5.7) is set equal to nrrSync or nrrPdelay (see 11.2.13.13) depending on whether the value of the global variable nrrCompMethod (see 11.2.13.15) is equal to Sync or Pdelay, respectively. 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.

Replace Figure 11-4 with the following:


MDSyncReceiveSM
(per port)
followUpReceiptTimeoutTime, rcvdSync, rcvdFollowUp, rcvdSyncPtr, rcvdFollowUpPtr, txMDSyncReceivePtr, portOper, ptpPortEnabled, asymmeryMeasurementMode, nrrSync, nrrCompMethod, neighborRateRatio
MDSyncReceive
PortSyncSyncReceive
(per port)
Described in media-independent clause
PortSyncSyncSend
(per port)
Described in media-independent clause
MDSyncSend
MDSyncSendSM
(per port)
rcvdMDSync, txSyncPtr, rcvdMDTimestampReceiveMDSS, rcvdMDTimestampReceivePtr, txFollowUpPtr, portOper, ptpPortEnabled, syncSequenceId, asymmeryMeasurementMode

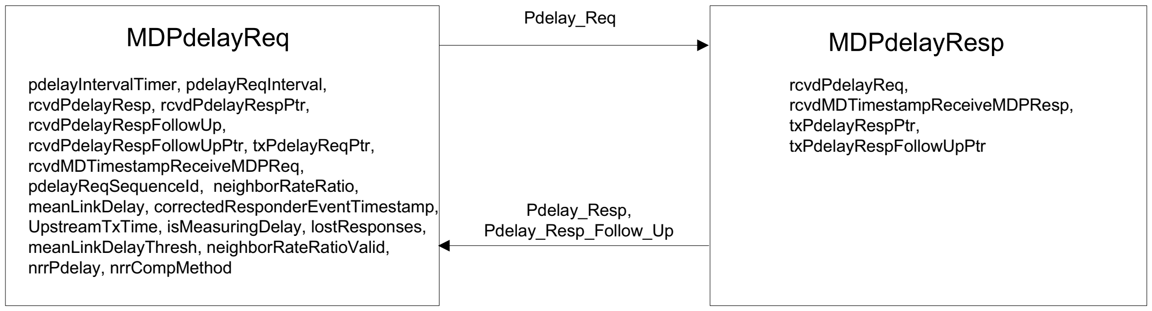
NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the the variables nrrSync, nrrCompMethod, and neighborRateRatio are added to the MDSyncReceiveSM block.

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, places the result in the global variable nrrPdelay (see 11.2.13.13), 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.

Replace Figure 11-5 with the following:


MDPdelayReq
pdelayIntervalTimer, pdelayReqInterval, rcvdPdelayResp, rcvdPdelayRespPtr, rcvdPdelayRespFollowUp, rcvdPdelayRespFollowUpPtr, txPdelayReqPtr, rcvdMDTimestampReceiveMDPReq, pdelayReqSequenceId, neighborRateRatio, meanLinkDelay, correctedResponderEventTimestamp, UpstreamTxTime, isMeasuringDelay, lostResponses, meanLinkDelayThresh, neighborRateRatioValid, nrrPdelay, nrrCompMethod
Pdelay_Req
Pdelay_Resp, Pdelay_Resp_Follow_Up
MDPdelayResp
rcvdPdelayReq, rcvdMDTimestampReceiveMDPResp, txPdelayRespPtr, txPdelayRespFollowUpPtr

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the the variables nrrPdelay and nrrCompMethod are added to the MDPdelayReq block.

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.

Change 11.2.2 as follows:

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 transport-specific or CMLDS 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. If only domain 0 is implemented, the state machines may implement the transport-specific peer-to-peer delay mechanism (see 11.2.17).

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	$\mathrm{FFFF\ FFFF\ FFFF\ FFFF\ FFFF\ FFFF}_{16}$
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.

Change 11.2.12 as follows:

11.2.12 Overview of MD entity global variables

Subclause 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~~Link Ports)
— Instances used by CMLDS, per ~~LinkPort~~Link Port

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

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~~Link Ports)	Instance used by CMLDS, per ~~LinkPort~~Link Port
currentLogPdelayReqInterval	11.2.13.1	No	Yes[1]	No	Yes
initialLogPdelayReqInterval	11.2.13.2	No	Yes[1]	No	Yes
pdelayReqInterval	11.2.13.3	No	Yes[1]	No	Yes
allowedLostResponses	11.2.13.4	No	Yes[1]	No	Yes
allowedFaults	11.2.13.5	No	Yes[1]	No	Yes
isMeasuringDelay	11.2.13.6	No	Yes[1]	No	Yes
meanLinkDelayThresh	11.2.13.7	No	Yes[1]	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
nrrPdelay	11.2.13.13	No	Yes	No	Yes~~No~~
nrrSync	11.2.13.14	No	Yes	No	No
nrrCompMethod	11.2.13.15	No	Yes	No	No

[1] 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

Change 11.2.13.1 as follows:

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, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

Change 11.2.13.2 as follows:

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, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

Change 11.2.13.3 as follows:

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, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

Change 11.2.13.4 as follows:

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, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

Change 11.2.13.5 as follows:

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~~nrrPdelay (see 11.2.13.13) 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, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

Change 11.2.13.6 as follows:

11.2.13.6 isMeasuringDelay: A Boolean that is TRUE if the port is ~~measuring~~receivingPdelay_Resp and Pdelay_Resp_Follow_Up messages from the port at the other end of the 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, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

Change 11.2.13.7 as follows:

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, i.e., all the PTP Instances (per port), and also one instance of this variable for domain 0 if domain 0 is implemented. The variable is accessible by all the domains.

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

Add the following definitions to 11.2.13:

11.2.13.13 nrrPdelay: The neighborRateRatio (see 10.2.5.7) computed using the transport-specific or CMLDS peer-to-peer delay mechanism (see 11.2.19). The data type for nrrPdelay is Float64.

11.2.13.14 nrrSync: The neighborRateRatio (see 10.2.5.7) computed using the Drift_Tracking TLV carried by successive Sync or, if two-step, Follow_Up messages (see 11.2.14). The data type for nrrSync is Float64.

11.2.13.15 nrrCompMethod: An Enumeration1 that takes on the values Sync and Pdelay to indicate the source of the value of neighborRateRatio (see 10.2.5.7):

a) If the value is Sync and driftTrackingTlvSupport (see 10.2.4.27) is TRUE, neighborRateRatio is populated with the value of nrrSync whenever a new value of nrrSync is computed.
b) If the value is Pdelay or if driftTrackingTlvSupport (10.2.4.27) is FALSE, neighborRateRatio is populated with the value of nrrPdelay whenever a new value of nrrPdelay is computed.

11.2.14 MDSyncReceiveSM state machine

Add the following definition to 11.2.14.1:

11.2.14.1 State machine variables

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

11.2.14.1.8 TEMP: A temporary variable used to reduce clutter in the state diagram (see Figure). The data type for TEMP is Integer16.

11.2.14.2 State machine functions

Insert 11.2.14.2.1 f), g), and h), and renumber subsequent list items as appropriate.

f) If the value of the global variable driftTrackingTlvSupport (see 10.2.4.27) is TRUE and the most recently received Sync message (if twoStepFlag is FALSE) or Follow_Up message (if twoStepFlag is TRUE) contains a Drift_Tracking TLV:
   1) syncGrandmasterIdentity (see 10.2.4.25) is set equal to the syncGrandmasterIdentity of the Drift_Tracking TLV.
   2) If the value of syncStepsRemoved of the Drift_Tracking TLV is not equal to 0xFFFF, syncStepsRemoved (see 10.2.4.26) is set equal to the value of syncStepsRemoved of the Drift_Tracking TLV plus one. If the value of syncStepsRemoved of the Drift_Tracking TLV is equal to 0xFFFF, syncStepsRemoved (see 10.2.4.26) is set equal to 0xFFFF.
   3) nrrSync (see 11.2.13.14) is computed using the following information conveyed by successive Sync and, if twoStepFlag is TRUE, Follow_Up messages; the specific algorithms are beyond the scope of this document:
      i) the syncEventIngressTimestamp values for the respective Sync messages.
      ii) The successive correctedResponderEventTimestamp values, whose data type is UScaledNs, obtained by adding (A) the seconds field of the syncEgressTimestamp of the

Drift_Tracking TLV, multiplied by 10^9, (B) the nanoseconds field of the syncEgressTimestamp of the Drift_Tracking TLV, and (C) CMLDS delayAsymmetry (i.e., cmldsLinkPort.delayAsymmetry, see 10.2.5.9) if CMLDS is provided, otherwise instance-specific delayAsymmetry (i.e., portDS.delayAsymmetry, see 14.8.10) divided by rateRatio (see 10.2.8.1.4)

iii) the rateRatioDrift field of the Drift_Tracking TLV.

NOTE 1—If delayAsymmetry does not change during the time interval over which nrrSync is computed, it is not necessary to add it in f)3)ii)(C) because in that case it will be canceled when computing the difference between earlier and later correctedResponderEventTimestamps.

NOTE 2—The methodology in f)3) for computing nrrSync follows the the methodology described in 11.2.19.3.3 for computing nrrPdelay using information contained in peer delay messages. See the NOTEs and description in 11.2.19.3.3 for more detail.

NOTE 3—The use of the rateRatioDrift, e.g., in compensating for frequency drift when computing rateRatio (see the RECEIVED_SYNC state of the PortSyncSyncReceive state machine of Figure 10-4), is implementation-specific or profile standard-specific.

g) If the value of the global variable driftTrackingTlvSupport (see 10.2.4.27) is TRUE and the most recently received Sync message (if twoStepFlag is FALSE) or Follow_Up message (if twoStepFlag is TRUE) does not contain a Drift_Tracking TLV:
   1) syncGrandmasterIdentity (see 10.2.4.25) is set equal to 0xFFFF FFFF FFFF FFFF.
   2) syncStepsRemoved (see 10.2.4.26) is set equal to 0xFFFF.

h) the value of neighborRateRatio (see 10.2.5.7) is set equal to the value of nrrSync (see 11.2.13.14) or nrrPdelay (see 11.2.13.13) based on the value of nrrCompMethod (see 11.2.13.15).

Change 11.2.14.2.1 f) as follows:

f) upstreamTxTime is set equal to the syncEventIngressTimestamp for the most recently received Sync message (see 11.4.3), minus the mean propagation time on the PTP Link attached to this PTP Port (meanLinkDelay; see 10.2.5.8) divided by neighborRateRatio (see 10.2.5.7), ~~and, if and only if the state machine is invoked by the instance-specific peer-to-peer delay mechanism,~~ minus delayAsymmetry (see 10.2.5.9) for this PTP Port divided by the sum of rateRatio [see item e) in this subclause] and the quantity neighborRateRatio − 1 ~~[see item e) in this subclause]~~. The syncEventIngressTimestamp is equal to the timestamp value measured relative to the timestamp measurement plane, minus any ingressLatency (see 8.4.3). The upstreamTxTime can be written as follows:

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

upstreamTxTime = syncEventIngressTimestamp − (meanLinkDelay/neighborRateRatio) − (delayAsymmetry/(rateRatio + neighborRateRatio − 1))

~~State machine invoked by CMLDS:~~

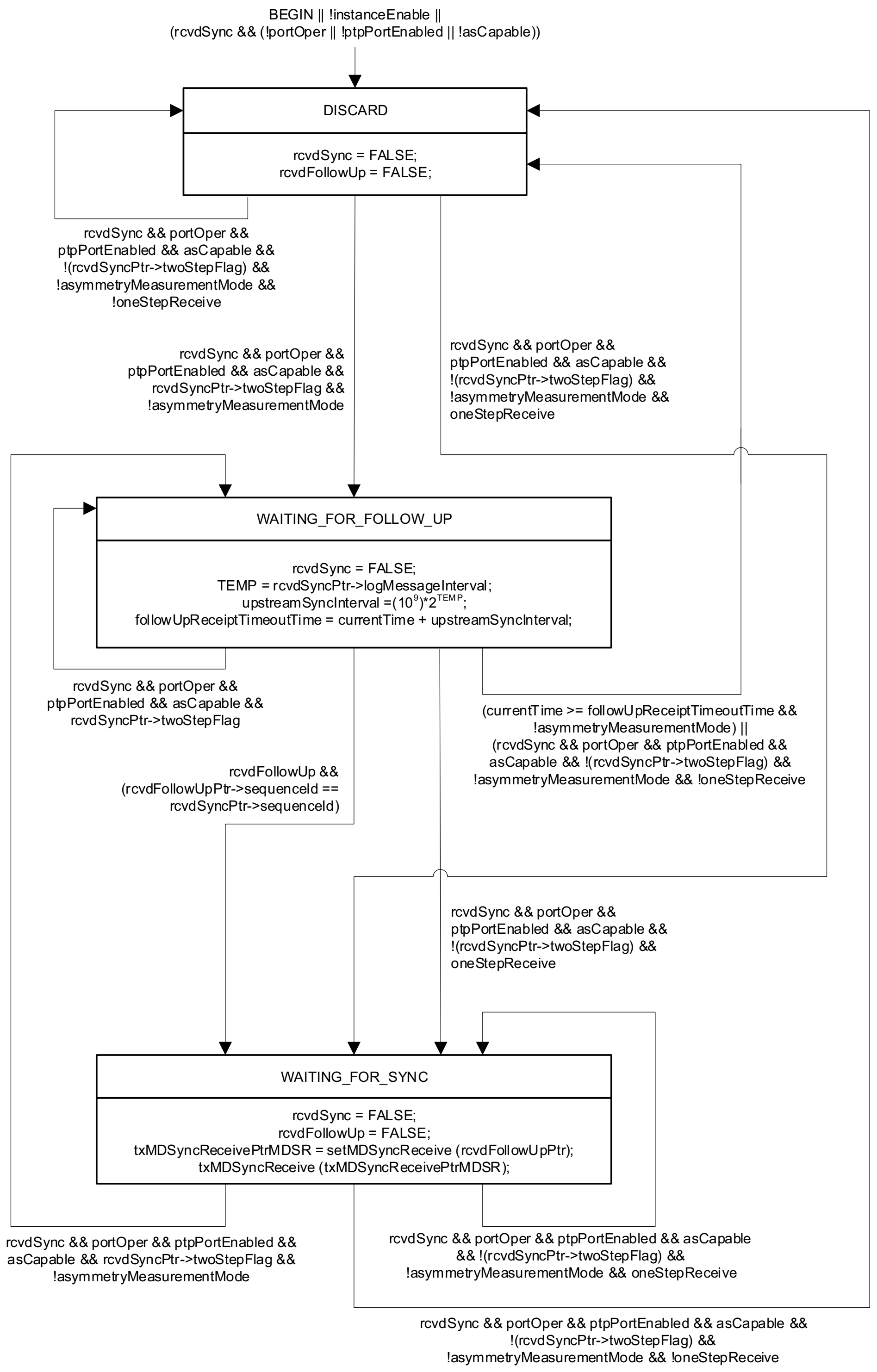
~~upstreamTxTime = syncEventIngressTimestamp − (meanLinkDelay/neighborRateRatio)~~

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 1—The syncEventIngressTimestamp is measured relative to the time base of the current PTP Instance. The mean propagation delay 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. delayAsymmetry (i.e., portDS.delayAsymmetry, see 14.8.10) is divided by the new rateRatio to convert it from the time base of the Grandmaster Clock to the time base of the current PTP Instance.

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).

11.2.14.3 State diagram

Replace Figure 11-6 with the following:


BEGIN || !instanceEnable || (rcvdSync && (!portOper || !ptpPortEnabled || !asCapable))
DISCARD
rcvdSync = FALSE;
rcvdFollowUp = FALSE;
rcvdSync && portOper && ptpPortEnabled && asCapable && !(rcvdSyncPtr->twoStepFlag) && !asymmetryMeasurementMode && !oneStepReceive
rcvdSync && portOper && ptpPortEnabled && asCapable && rcvdSyncPtr->twoStepFlag && !asymmetryMeasurementMode
rcvdSync && portOper && ptpPortEnabled && asCapable && !(rcvdSyncPtr->twoStepFlag) && !asymmetryMeasurementMode && oneStepReceive
WAITING_FOR_FOLLOW_UP
rcvdSync = FALSE;
TEMP = rcvdSyncPtr->logMessageInterval;
upstreamSyncInterval =(10^9)*2^TEMP;
followUpReceiptTimeoutTime = currentTime + upstreamSyncInterval;
rcvdSync && portOper && ptpPortEnabled && asCapable && rcvdSyncPtr->twoStepFlag
(currentTime >= followUpReceiptTimeoutTime && !asymmetryMeasurementMode) || (rcvdSync && portOper && ptpPortEnabled && asCapable && !(rcvdSyncPtr->twoStepFlag) && !asymmetryMeasurementMode && !oneStepReceive
rcvdFollowUp && (rcvdFollowUpPtr->sequenceId == rcvdSyncPtr->sequenceId)
rcvdSync && portOper && ptpPortEnabled && asCapable && !(rcvdSyncPtr->twoStepFlag) && oneStepReceive
WAITING_FOR_SYNC
rcvdSync = FALSE;
rcvdFollowUp = FALSE;
txMDSyncReceivePtrMDSR = setMDSyncReceive (rcvdFollowUpPtr);
txMDSyncReceive (txMDSyncReceivePtrMDSR);
rcvdSync && portOper && ptpPortEnabled && asCapable && rcvdSyncPtr->twoStepFlag && !asymmetryMeasurementMode
rcvdSync && portOper && ptpPortEnabled && asCapable && !(rcvdSyncPtr->twoStepFlag) && !asymmetryMeasurementMode && oneStepReceive
rcvdSync && portOper && ptpPortEnabled && asCapable && !(rcvdSyncPtr->twoStepFlag) && !asymmetryMeasurementMode && !oneStepReceive

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the variable portEnabledOper at the entry to the DISCARD state is changed to portOper.

Figure 11-6—MDSyncReceiveSM state machine

11.2.15 MDSyncSendSM state machine

Insert 11.2.15.2.3 g) and h), and renumber subsequent list items as appropriate.

g) If the value of the global variable driftTrackingTlvSupport (see 10.2.4.27) is TRUE and the PTP Instance is the Grandmaster PTP Instance, a Drift_Tracking TLV is attached to the transmitted Follow_Up message with the syncGrandmasterIdentity, syncStepsRemoved, and syncEgressTimestamp fields set as follows:

1) syncGrandmasterIdentity is set equal to the defaultDS.clockIdentity.
2) syncStepsRemoved is set equal to 0.
3) syncEgressTimestamp is set equal to the syncEventEgressTimestamp of the associated Sync message.
4) rateRatioDrift (see 11.4.4.4.9) is the measured estimate of the rate of change per second of the frequency of the Grandmaster Clock relative to the frequency of the local clock at the Grandmaster PTP Instance. If the Grandmaster Clock and the Local Clock at the Grandmaster PTP Instance are the same, rateRatioDrift is zero.

h) If the value of the global variable driftTrackingTlvSupport (see 10.2.4.27) is TRUE and the PTP Instance is not the Grandmaster PTP Instance, a Drift_Tracking TLV is attached to the transmitted Follow_Up message with the syncGrandmasterIdentity, syncStepsRemoved, and syncEgressTimestamp fields set as follows:

1) syncGrandmasterIdentity is set equal to the value of the global variable syncGrandmasterIdentity (see 10.2.4.25).
2) syncStepsRemoved is set equal to the value of the global variable syncStepsRemoved (see 10.2.4.26) .
3) syncEgressTimestamp is set equal to the syncEventEgressTimestamp of the associated Sync message.
4) rateRatioDrift (see 11.4.4.4.9) is the measured estimate of the rate of change per second of the frequency of the Grandmaster Clock relative to the frequency of the local clock of the PTP Instance that sends the message.

Change 11.2.15.2.3 i) as follows:

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

Insert 11.2.15.2.5 g) and h), and renumber subsequent list items as appropriate.

g) If the value of the global variable driftTrackingTlvSupport (see 10.2.4.27) is TRUE and the PTP Instance is the Grandmaster PTP Instance, a Drift_Tracking TLV is attached to the transmitted Sync message with the syncGrandmasterIdentity, syncStepsRemoved, and syncEgressTimestamp fields set as follows:

1) syncGrandmasterIdentity is set equal to the defaultDS.clockIdentity.
2) syncStepsRemoved is set equal to 0.
3) syncEgressTimestamp is set equal to the syncEventEgressTimestamp of the associated Sync message.
4) rateRatioDrift (see 11.4.4.4.9) is the measured estimate of the rate of change per second of the frequency of the Grandmaster Clock relative to the frequency of the local clock at the Grandmaster PTP Instance. If the Grandmaster Clock and the Local Clock at the Grandmaster PTP Instance are the same, rateRatioDrift is zero.

h) If the value of the global variable driftTrackingTlvSupport (see 10.2.4.27) is TRUE and the PTP Instance is not the Grandmaster PTP Instance, a Drift_Tracking TLV is attached to the transmitted

Sync message with the syncGrandmasterIdentity, syncStepsRemoved, and syncEgressTimestamp fields set as follows:

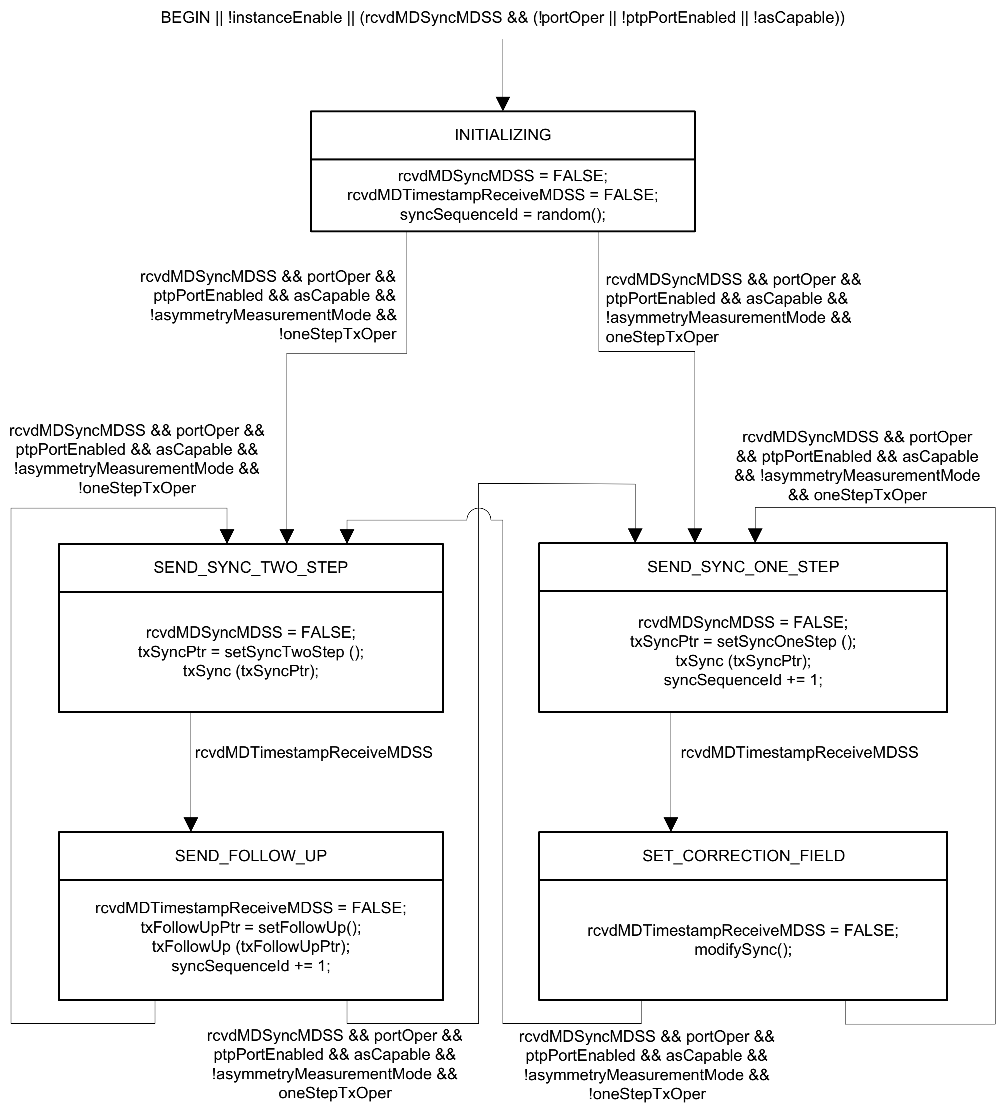
1) syncGrandmasterIdentity is set equal to the value of the global variable syncGrandmasterIdentity (see 10.2.4.25).
2) syncStepsRemoved is set equal to the value of the global variable syncStepsRemoved (see 10.2.4.26) .
3) syncEgressTimestamp is set equal to the syncEventEgressTimestamp of the associated Sync message.
4) rateRatioDrift (see 11.4.4.4.9) is the measured estimate of the rate of change per second of the frequency of the Grandmaster Clock relative to the frequency of the local clock of the PTP Instance that sends the message.

Change 11.2.15.2.5 i) as follows:

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

11.2.15.3 State diagram

Replace Figure 11-7 with the following:


BEGIN || !instanceEnable || (rcvdMDSyncMDSS && (!portOper || !ptpPortEnabled || !asCapable))
INITIALIZING
rcvdMDSyncMDSS = FALSE;
rcvdMDTimestampReceiveMDSS = FALSE;
syncSequenceId = random();
rcvdMDSyncMDSS && portOper && ptpPortEnabled && asCapable && !asymmetryMeasurementMode && !oneStepTxOper
rcvdMDSyncMDSS && portOper && ptpPortEnabled && asCapable && !asymmetryMeasurementMode && oneStepTxOper
rcvdMDSyncMDSS && portOper && ptpPortEnabled && asCapable && !asymmetryMeasurementMode && !oneStepTxOper
rcvdMDSyncMDSS && portOper && ptpPortEnabled && asCapable && !asymmetryMeasurementMode && oneStepTxOper
SEND_SYNC_TWO_STEP
rcvdMDSyncMDSS = FALSE;
txSyncPtr = setSyncTwoStep ();
txSync (txSyncPtr);
SEND_SYNC_ONE_STEP
rcvdMDSyncMDSS = FALSE;
txSyncPtr = setSyncOneStep ();
txSync (txSyncPtr);
syncSequenceId += 1;
rcvdMDTimestampReceiveMDSS
rcvdMDTimestampReceiveMDSS
SEND_FOLLOW_UP
rcvdMDTimestampReceiveMDSS = FALSE;
txFollowUpPtr = setFollowUp();
txFollowUp (txFollowUpPtr);
syncSequenceId += 1;
SET_CORRECTION_FIELD
rcvdMDTimestampReceiveMDSS = FALSE;
modifySync();
rcvdMDSyncMDSS && portOper && ptpPortEnabled && asCapable && !asymmetryMeasurementMode && oneStepTxOper
rcvdMDSyncMDSS && portOper && ptpPortEnabled && asCapable && !asymmetryMeasurementMode && !oneStepTxOper

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that (a) the variable portEnabledOper is changed to portOper on the links between the SEND_FOLLOW_UP and SEND_SYNC_TWO_STEP states, between the SEND_FOLLOW_UP and SEND_SYNC_ONE_STEP states, and between the SET_CORRECTION_FIELD and SEND_SYNC_TWO_STEP states; and (b) the statement 'syncSequenceId += ' in the last line of the SEND_SYNC_TWO_STEP state is moved to the last line of the SEND_FOLLOW_UP state.

Figure 11-7—MDSyncSendSM state machine

11.2.17 Common Mean Link Delay Service (CMLDS)

Change 11.2.17.1 as follows:

11.2.17.1 General

Each ~~port~~PTP Port or Link 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 or if domainNumber 0 is not present, 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, and if ~~(~~the domainNumber of this domain is 0~~;~~ (see 8.1), these two state machines may also provide the CMLDS; however, if ~~they~~the state machines do not provide the CMLDS (i.e., if only the ~~PTP Instance~~transport-specific peer delay mechanism is provided), they shall be invoked on domain 0. In other words, if the ~~domain number~~domainNumber is not 0, portDS.delayMechanism (see Table 14-8 in 14.8.5) must not be P2P. If CMLDS is used, the LocalClock entity for CMLDS and the LocalClock entity for each PTP Instance shall be the same LocalClock.

NOTE 1—In the above~~ sentence~~, the ~~condition that~~case where 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 another PTP profile~~s~~ (see IEEE Std 1588) besides IEEE Std 802.1AS is active on the port, the CMLDS provides the mean propagation delay and neighbor rate ratio to all of them.~~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~~transport-specific peer delay messages, are given in 8.1.

NOTE 3—The requirements of this subclause for CMLDS are consistent with the requirements of IEEE Std 1588-2019.

If a PTP Port on a physical interface that also supports CMLDS and a LinkPort ~~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~~transport-specific peer delay messages (i.e., the Pdelay_Resp and Pdelay_Resp_Follow_Up corresponding to this Pdelay_Req) using the ~~instance~~transport-specific peer-to-peer delay mechanism if domain 0 is enabled. These ~~instance~~transport-specific messages have majorSdoId value of 0x1, minorSdoId value of 0x00, and domainNumber value of 0.

NOTE ~~3~~4—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~~5—In general, a port can receive:

a) Peer delay messages of the CMLDS (with sdoId 0f 0x200),

b) ~~PTP Instance~~transport-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~~transport-specific peer delay, peer delay messages of those profiles (i.e., with an sdoId of neither 0x100 nor 0x200).

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.

Change 11.2.17.2 as follows:

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

The MDPdelayReq state machine (see 11.2.19)~~,~~ and the MDPdelay~~_~~Resp state machine (see 11.2.20)~~, and MDSyncReceiveSM state machine (see 11.2.14)~~ are invoked either by the transport-specific peer-to-peer delay mechansim or by the CMLDS peer-to-peer delay mechanism. ~~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 resulting values of meanLinkDelay and nrrPdelay are the same for both transport-specific peer delay and CMLDS peer delay; however, the organization of the computations for transport-specific peer delay and CMLDS peer delay are different. Some of the computations are done at the Initiator and some of the computations are done at the Responder. It is necessary for the Initiator and the Responder to both perform the transport-specific peer delay computations or both perform the CMLDS peer delay computations in order to obtain the correct results for meanLinkDelay and nrrPdelay. The differences are ~~described as follows~~:

a) Both ~~Instance~~transport-specific peer delay and CMLDS peer delay compute~~s~~ ~~mean link delay~~meanLinkDelay averaged over the two directions, and add~~s~~ delayAsymmetry separately when computing upstreamTxTime by the setMDSyncReceiveMDSR() function of the MDSyncReceive state machine. However, in the computation of meanLinkDelay, CMLDS subtracts delayAsymmetry from the correctionField when sending the pdelayReq message at the Initiator (see Figure 11-1) and adds delayAsymmetry back when computing the quantity t_3 in the function computePropTime() (at the Initiator), while transport-specific peer delay does not subtract and add back delayAsymmetry.~~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~~Transport-specific peer delay sets the correctionField of a transmitted Pdelay_Req message to 0, while CMLDS peer delay sets it to –delayAsymmetry (see 11.2.19.3.1).

c) ~~Instance~~Transport-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 peer delay 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~~Transport-specific peer delay sets the correctionField of Pdelay_Resp_Follow_Up equal to the fractional nanoseconds portion of the pdelayRespEventEgressTimestamp, while CMLDS peer delay 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 the quantity t_2~~mean link delay~~ for the meanLinkDelay computation [i.e., the quantity D in Equation (11-5), see 11.2.19.3.4], the correctionField of the Pdelay_Resp message, divided by 2^{16}, is added ~~when computing the quantity~~ ~~t_2~~ if ~~instance~~transport-specific peer delay is used, while it is subtracted if CMLDS peer delay is used (see 11.2.19.3.4), and the quantity -delayAsymmetry, divided by 2^{16}, is added if CMLDS peer delay is used (i.e., delayAsymmetry is subtracted in the case of CMLDS).

f) When computing ~~neighborRateRatio~~nrrPdelay, the computation of the correctedResponderEventTimestamp ~~must be~~is corrected for delayAsymmetry if, and only if, CMLDS peer delay is used. The reason for this correction is that, with CMLDS peer delay, 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~~transport-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~~transport-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 peer delay ~~must~~need to be consistent with IEEE Std 1588-2019 because CMLDS peer delay 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~~transport-specific peer-to-peer delay mechanism and CMLDS peer delay are different (i.e., are organized differently), even though they produce the same results.

11.2.19 MDPdelayReq state machine

Change 11.2.19.1 as follows:

11.2.19.1 General

This state machine is either invoked as part of the Common Mean Link Delay Service (CMLDS) or by the transport-specific peer-to-peer delay mechanism. 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, , 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

Change 11.2.19.2.10 as follows:

11.2.19.2.10 ~~neighborRateRatio~~nrrPdelayValid: A Boolean variable that indicates whether the function computePdelayRateRatio() (see 11.2.19.3.3) successfully computed ~~neighborRateRatio~~ nrrPdelay (see 11.2.13.13).

Change 11.2.19.2.12 as follows:

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~~transport-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 peer-to-peer delay mechanism.

Change 11.2.19.2.13 as follows:

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

11.2.19.2.14

11.2.19.3 State machine functions

Change 11.2.19.3.1 as follows:

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~~transport-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.

Change 11.2.19.3.3 as follows:

11.2.19.3.3 computePdelayRateRatio(): Computes ~~neighborRateRatio~~nrrPdelay (see 11.2.13.13) 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~~field
   3) The correctionField, divided by 2^{16}
   4) delayAsymmetry (see 10.2.5.9), 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~~nrrPdelay is computed, it is not necessary to ~~subtract~~add it if this state machine is invoked by CMLDS because in that case it will be canceled when computing the difference between earlier and later correctedResponderEventTimestamps.

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~~nrrPdelay is acceptable as long as the performance requirements specified in B.2.4 are met. If ~~neighborRateRatio~~nrrPdelay is successfully computed, the Boolean ~~neighborRateRatio~~nrrPdelayValid (see 11.2.19.2.10) is set to TRUE. If ~~neighborRateRatio~~nrrPdelay 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~~nrrPdelayValid is set to FALSE.

NOTE 2—As one example, ~~neighborRateRatio~~nrrPdelay 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.

Change 11.2.19.3.4 as follows:

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, and (4) -delayAsymmetry divided by 2^{16} if this state machine is invoked by CMLDS (i.e., delayAsymmetry is subtracted in the case of CMLDS).

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~~ nrrPdelay for this MD entity (see 10.2.5.7)

When CMLDS is used, Eq. (11-5) can be rewritten as shown in Eq. (11-6) (see NOTE 3 below) using the definitions of t_2 and t_3 above for the CMLDS case.

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~~nrrPdelay (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)~~meanLinkDelay is organized differently from the organization used for transport-specific peer delay 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) - $ (responseOriginTimestamp − requestReceiptTimestamp)~~
~~+ (correctionField of Pdelay_Resp) − (correctionField of Pdelay_Resp_Follow_Up)] / 2~~ (11-6)

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

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~~ the present standard for transport-specific peer delay [see 11.2.20.3.1 d)1]; however, the correctionField of Pdelay_Resp is then subtracted in Equation (11-6) rather than added [in Eq. (11-6) for transport-specific peer delay, where s=1], and the two minus signs ~~cancel each other~~result in the same sign for the fractional nanoseconds portionof t_2 and the seconds and nanoseconds portion of t_2 (requestReceiptTimestamp). The computations of D in this standard and IEEE Std 1588-2019 are mathematically equivalent. The organization of the computation with CMLDS must be used in the present standard for interoperability with~~used in~~ IEEE Std 1588-2019 (see 11.2.17.2). This organization ~~must be used with CMLDS~~ in the present standard, for the case where CMLDS is used, is consistent with the organization in~~for interoperability with~~ IEEE Std 1588-2019 ~~(see 11.2.17.2)~~.

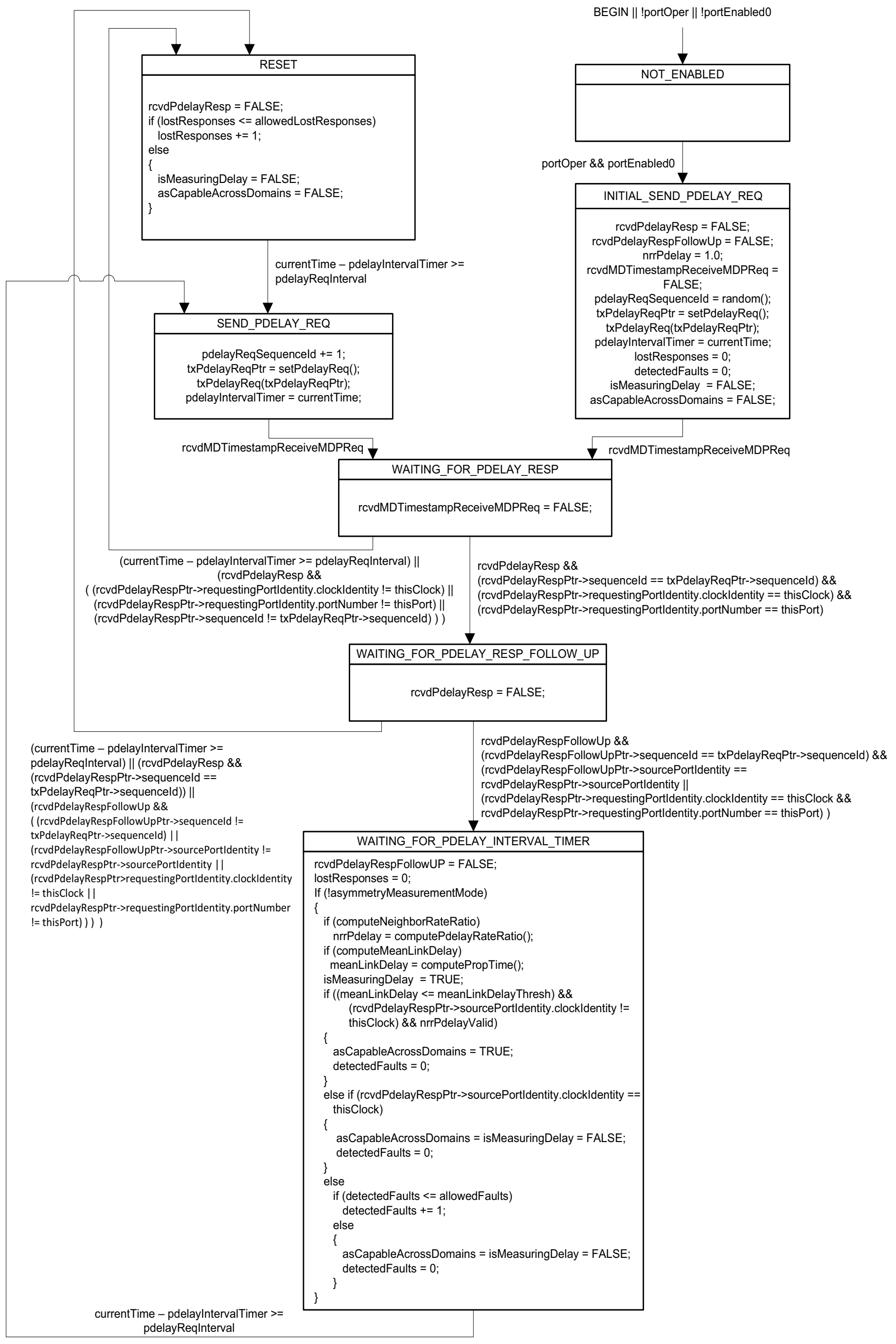
11.2.19.4 State diagram

Change NOTE 1 of 11.2.19.4 as follows:

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~~nrrPdelay, retains its most recent value when a Pdelay_Resp and/or Pdelay_Resp_Follow_Up message is lost.

Replace Figure 11-9 with the following:

Delete the NOTE immediately after Figure 11-9.


BEGIN || !portOper || !portEnabled0
NOT_ENABLED
portOper && portEnabled0
INITIAL_SEND_PDELAY_REQ
rcvdPdelayResp = FALSE;
rcvdPdelayRespFollowUp = FALSE;
nrrPdelay = 1.0;
rcvdMDTimestampReceiveMDPReq = FALSE;
pdelayReqSequenceId = random();
txPdelayReqPtr = setPdelayReq();
txPdelayReq(txPdelayReqPtr);
pdelayIntervalTimer = currentTime;
lostResponses = 0;
detectedFaults = 0;
isMeasuringDelay = FALSE;
asCapableAcrossDomains = FALSE;
RESET
rcvdPdelayResp = FALSE;
if (lostResponses <= allowedLostResponses)
lostResponses += 1;
else
{
isMeasuringDelay = FALSE;
asCapableAcrossDomains = FALSE;
}
currentTime – pdelayIntervalTimer >= pdelayReqInterval
SEND_PDELAY_REQ
pdelayReqSequenceId += 1;
txPdelayReqPtr = setPdelayReq();
txPdelayReq(txPdelayReqPtr);
pdelayIntervalTimer = currentTime;
rcvdMDTimestampReceiveMDPReq
rcvdMDTimestampReceiveMDPReq
WAITING_FOR_PDELAY_RESP
rcvdMDTimestampReceiveMDPReq = FALSE;
(currentTime – pdelayIntervalTimer >= pdelayReqInterval) || (rcvdPdelayResp && ((rcvdPdelayRespPtr->requestingPortIdentity.clockIdentity != thisClock) || (rcvdPdelayRespPtr->requestingPortIdentity.portNumber != thisPort) || (rcvdPdelayRespPtr->sequenceId != txPdelayReqPtr->sequenceId)))
rcvdPdelayResp && (rcvdPdelayRespPtr->sequenceId == txPdelayReqPtr->sequenceId) && (rcvdPdelayRespPtr->requestingPortIdentity.clockIdentity == thisClock) && (rcvdPdelayRespPtr->requestingPortIdentity.portNumber == thisPort)
WAITING_FOR_PDELAY_RESP_FOLLOW_UP
rcvdPdelayResp = FALSE;
(currentTime – pdelayIntervalTimer >= pdelayReqInterval) || (rcvdPdelayResp && (rcvdPdelayRespPtr->sequenceId == txPdelayReqPtr->sequenceId)) || (rcvdPdelayRespFollowUp && ((rcvdPdelayRespFollowUpPtr->sequenceId != txPdelayReqPtr->sequenceId) | | (rcvdPdelayRespFollowUpPtr->sourcePortIdentity != rcvdPdelayRespPtr->sourcePortIdentity | | (rcvdPdelayRespPtr>requestingPortIdentity.clockIdentity != thisClock | | rcvdPdelayRespPtr->requestingPortIdentity.portNumber != thisPort))))
rcvdPdelayRespFollowUp && (rcvdPdelayRespFollowUpPtr->sequenceId == txPdelayReqPtr->sequenceId) && (rcvdPdelayRespFollowUpPtr->sourcePortIdentity == rcvdPdelayRespPtr->sourcePortIdentity || (rcvdPdelayRespPtr->requestingPortIdentity.clockIdentity == thisClock && rcvdPdelayRespPtr->requestingPortIdentity.portNumber == thisPort))
WAITING_FOR_PDELAY_INTERVAL_TIMER
rcvdPdelayRespFollowUP = FALSE;
lostResponses = 0;
If (!asymmetryMeasurementMode)
{
if (computeNeighborRateRatio)
nrrPdelay = computePdelayRateRatio();
if (computeMeanLinkDelay)
meanLinkDelay = computePropTime();
isMeasuringDelay = TRUE;
if ((meanLinkDelay <= meanLinkDelayThresh) && (rcvdPdelayRespPtr->sourcePortIdentity.clockIdentity != thisClock) && nrrPdelayValid)
{
asCapableAcrossDomains = TRUE;
detectedFaults = 0;
}
else if (rcvdPdelayRespPtr->sourcePortIdentity.clockIdentity == thisClock)
{
asCapableAcrossDomains = isMeasuringDelay = FALSE;
detectedFaults = 0;
}
else
if (detectedFaults <= allowedFaults)
detectedFaults += 1;
else
{
asCapableAcrossDomains = isMeasuringDelay = FALSE;
detectedFaults = 0;
}
}
currentTime – pdelayIntervalTimer >= pdelayReqInterval

NOTE—This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that (a) the condition "else if (rcvdPdelayRespPtr->sourcePortIdentity.clockIdentity != thisClock)" in the state WAITING_FOR_PDELAY_INTERVAL_TIMER is changed to "else if (rcvdPdelayRespPtr->sourcePortIdentity.clockIdentity == thisClock)" (i.e., "!=" is changed to "=="), and (b) the variables neighborRateRatio and neighborRateRatioValid are replaced by nrrPdelay and nrrPdelayValid, respectively.

Figure 11-9—MDPdelayReq state machine

11.2.20 MDPdelayResp state machine

Change as follows:

11.2.20.1 General

This state machine is either invoked as part of the Common Mean Link Delay Service (CMLDS) or by the transport-specific peer-to-peer delay mechanism. 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, , 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

Change 11.2.20.2.5 as follows:

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~~transport-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 peer delay mechanism.

11.2.20.3 State machine functions

Change 11.2.20.3.1 as follows:

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~~transport-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 peer-to-peer delay mechanism.

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.

Change 11.2.20.3.3 as follows:

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~~transport-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 peer-to-peer delay mechanism~~,~~.

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.

11.2.21 LinkDelayIntervalSetting state machine

11.2.21.2 State machine variables

Change 11.2.21.2.3 as follows:

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~~transport-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 peer-to-peer delay mechanism.

Add the following definition to 11.2.21.2:

11.2.21.2 State machine variables

11.2.21.2.7 TEMP: A temporary variable used to reduce clutter in the state diagram (see Figure 11-11). The data type for TEMP is Integer16.

11.2.21.3 State machine functions

Change 11.2.21.3.2 as follows:

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~~largest supported ~~rate~~logPdelayReqInterval, even if ~~faster~~smaller than the requested ~~rate~~interval
                        return (logSupportedPdelayReqIntervalMax);
                else
                        // Return the ~~fastest~~smallest supported ~~rate~~logPdelayReqInterval that is still ~~slower~~larger than
                        // the requested ~~rate~~interval.
                        return (logSupportedClosestLongerPdelayReqInterval);
        }
}
```

Change the first paragraph of 11.2.21.4 as follows:

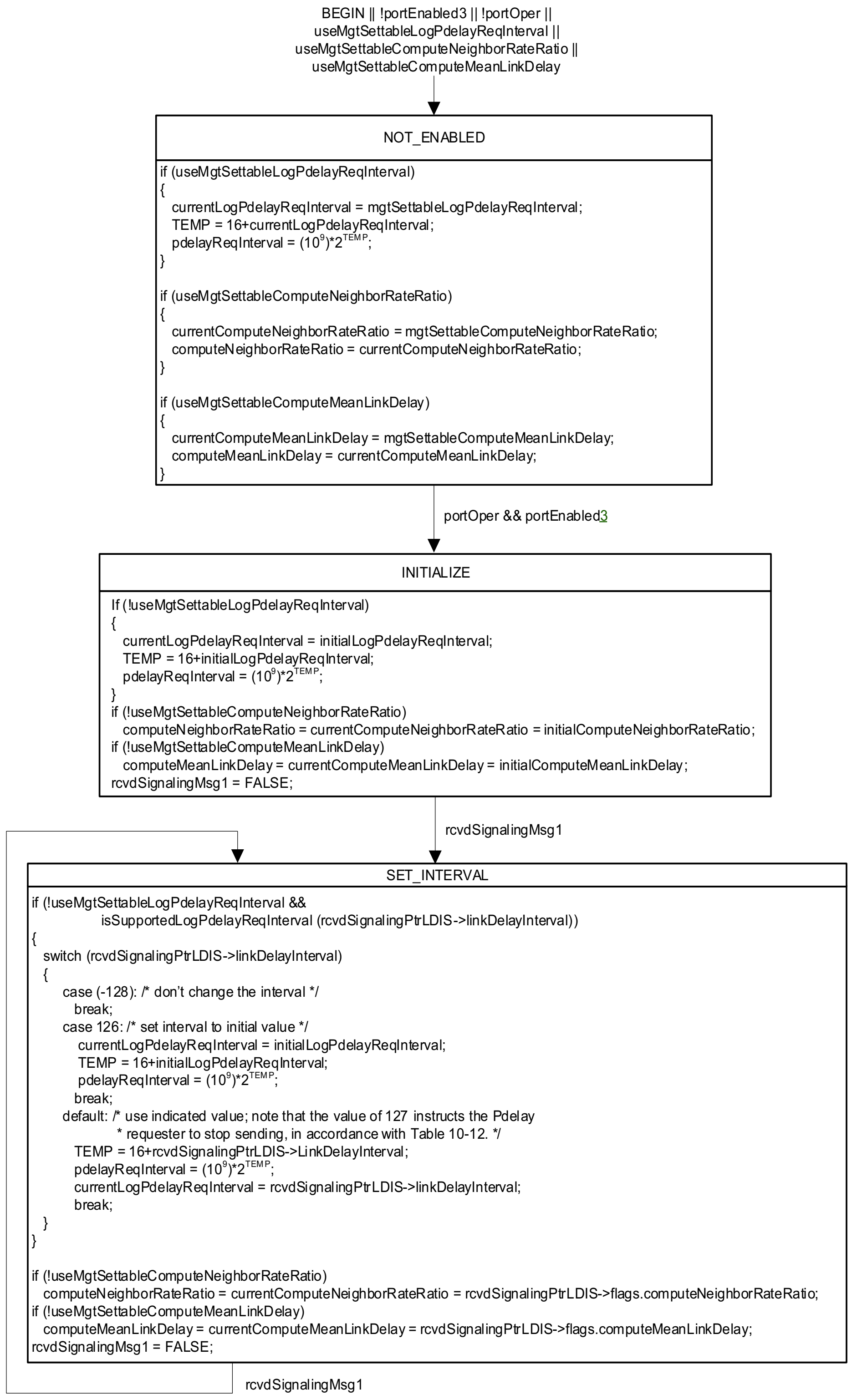
11.2.21.4 State diagram

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, , 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~~nrrPdelay 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).

11.3 Message attributes

11.3.6 Subtype

The subtype ~~of~~for 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).


BEGIN || !portEnabled3 || !portOper ||
useMgtSettableLogPdelayReqInterval ||
useMgtSettableComputeNeighborRateRatio ||
useMgtSettableComputeMeanLinkDelay
NOT_ENABLED
portOper && portEnabled3
INITIALIZE
rcvdSignalingMsg1
SET_INTERVAL
rcvdSignalingMsg1

NOT_ENABLED:

```
if (useMgtSettableLogPdelayReqInterval)
{
   currentLogPdelayReqInterval = mgtSettableLogPdelayReqInterval;
   TEMP = 16+currentLogPdelayReqInterval;
   pdelayReqInterval = $(10^9)*2^{TEMP}$;
}

if (useMgtSettableComputeNeighborRateRatio)
{
   currentComputeNeighborRateRatio = mgtSettableComputeNeighborRateRatio;
   computeNeighborRateRatio = currentComputeNeighborRateRatio;
}

if (useMgtSettableComputeMeanLinkDelay)
{
   currentComputeMeanLinkDelay = mgtSettableComputeMeanLinkDelay;
   computeMeanLinkDelay = currentComputeMeanLinkDelay;
}
```

INITIALIZE:

```
If (!useMgtSettableLogPdelayReqInterval)
{
   currentLogPdelayReqInterval = initialLogPdelayReqInterval;
   TEMP = 16+initialLogPdelayReqInterval;
   pdelayReqInterval = $(10^9)*2^{TEMP}$;
}
if (!useMgtSettableComputeNeighborRateRatio)
   computeNeighborRateRatio = currentComputeNeighborRateRatio = initialComputeNeighborRateRatio;
if (!useMgtSettableComputeMeanLinkDelay)
   computeMeanLinkDelay = currentComputeMeanLinkDelay = initialComputeMeanLinkDelay;
rcvdSignalingMsg1 = FALSE;
```

SET_INTERVAL:

```
if (!useMgtSettableLogPdelayReqInterval &&
            isSupportedLogPdelayReqInterval (rcvdSignalingPtrLDIS->linkDelayInterval))
{
   switch (rcvdSignalingPtrLDIS->linkDelayInterval)
   {
      case (-128): /* don't change the interval */
         break;
      case 126: /* set interval to initial value */
         currentLogPdelayReqInterval = initialLogPdelayReqInterval;
         TEMP = 16+initialLogPdelayReqInterval;
         pdelayReqInterval = $(10^9)*2^{TEMP}$;
         break;
      default: /* use indicated value; note that the value of 127 instructs the Pdelay
               * requester to stop sending, in accordance with Table 10-12. */
         TEMP = 16+rcvdSignalingPtrLDIS->LinkDelayInterval;
         pdelayReqInterval = $(10^9)*2^{TEMP}$;
         currentLogPdelayReqInterval = rcvdSignalingPtrLDIS->linkDelayInterval;
         break;
   }
}

if (!useMgtSettableComputeNeighborRateRatio)
   computeNeighborRateRatio = currentComputeNeighborRateRatio = rcvdSignalingPtrLDIS->flags.computeNeighborRateRatio;
if (!useMgtSettableComputeMeanLinkDelay)
   computeMeanLinkDelay = currentComputeMeanLinkDelay = rcvdSignalingPtrLDIS->flags.computeMeanLinkDelay;
rcvdSignalingMsg1 = FALSE;
```

Figure 11-11—LinkDelayIntervalSetting state machine

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

11.4 Message formats

11.4.2 Header

11.4.2.6 correctionField (Integer64)

Change Table 11-6 as follows:

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), and asymmetry ~~corrections~~ (see 8.3, 11.2.14.2.1, and 11.2.15.2.3.~~;~~ T~~t~~he 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~~transport-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 peer-to-peer delay mechanism.
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~~transport-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 peer-to-peer delay mechanism.
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.	

11.4.3 Sync message

Change 11.4.3.1 as follows:

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. Carrying the Drift_Tracking TLV is optional.

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
Drift_Tracking TLV (optional)	30	76

11.4.3.2 Sync message field specifications if twoStep flag is FALSE

Insert 11.4.3.2.2, and renumber subsequent subclauses as appropriate:

11.4.3.2.2 Drift_Tracking TLV

The Sync message may carry the Drift_Tracking TLV, defined in 11.4.4.4.

11.4.4 Follow_Up message

Change 11.4.4.1 as follows:

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. Carrying the Drift_Tracking TLV is optional.

11.4.4.2 Follow_Up message field specifications

Insert 11.4.4.2.3, and renumber subsequent subclauses as appropriate:

Table 11-10—Follow_Up message fields

Bits 7 6 5 4 3 2 1 0	Octets	Offset
header (see 11.4.2)	34	0
preciseOriginTimestamp	10	34
Follow_Up information TLV	32	44
<u>Drift_Tracking TLV (optional)</u>	<u>30</u>	<u>76</u>

11.4.4.2.3 Drift_Tracking TLV

The Follow_Up message may carry the Drift_Tracking TLV, defined in 11.4.4.4.

Insert 11.4.4.4, and renumber subsequent subclauses and as appropriate:

11.4.4.4 Drift_Tracking TLV definition

11.4.4.4.1 General

The fields of the Drift_Tracking TLV shall be as specified in Table 11-12 and in 11.4.4.4.2 through 11.4.4.4.9. This TLV is a standard organization extension TLV for the Sync or Follow_Up message, as specified in 14.3 of IEEE Std 1588-2019.

Insert Table 11-12, and renumber subsequent tables as appropriate:

Table 11-12—Drift_Tracking TLV

Bits 7 6 5 4 3 2 1 0	Octets	Offset
tlvType	2	0
lengthField	2	2
organizationId	3	4
organizationSubType	3	7
syncEgressTimestamp	12	10
syncGrandmasterIdentity	8	22
syncStepsRemoved	2	30
rateRatioDrift	4	32

11.4.4.4.2 tlvType (Enumeration16)

The value of the tlvType 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.

11.4.4.4.3 lengthField (UInteger16)

The value of lengthField is 26.

11.4.4.4.4 organizationId (Octet3)

The value of organizationId is 00-80-C2.

11.4.4.4.5 organizationSubType (Enumeration24)

The value of organizationSubType is 6.

11.4.4.4.6 syncEgressTimestamp (ExtendedTimestamp)

The value of the syncEgressTimestamp field is the timestamp, based on the local clock, when the Sync message was sent by that PTP Relay Instance~~is the seconds and nanoseconds portion of the associated Sync message~~ (see 11.4.3.2).

11.4.4.4.7 syncGrandmasterIdentity (ClockIdentity)

The value is the value of the clockIdentity component of the rootSystemIdentity of the gmPriorityVector (see 10.3.5) of the PTP Instance that transmits the Sync message.

11.4.4.4.8 syncStepsRemoved (UInteger16)

The value is the value of syncTimeTransmitterStepsRemoved (see 10.3.9.3) for the PTP Instance that transmits the Sync message.

11.4.4.4.9 rateRatioDrift (Integer32)

The value of rateRatioDrift is equal to (RRdrift – 1.0) x (2^{41}), truncated to the next smaller signed integer, where RRdrift is the measured estimate of the rate of change per second of the ratio of the frequency of the Grandmaster Clock to the frequency of the Local Clock entity in the PTP Instance that sends the message.

NOTE—The above scaling allows the representation of rates of change of fractional frequency offset in the range $[-(2^{-10} - 2^{-41}), 2^{-10} - 2^{-41}]\ s^{-1}$, with granularity of 2^{-41}. This range is approximately $[-9.766 \times 10^{-4}, 9.766 \times 10^{-4}]\ s^{-1}$.

11.5 Protocol timing characterization

11.5.2 Message transmission intervals

11.5.2.2 Pdelay_Req message transmission interval

Change 11.5.2.2 as follows:

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, currentLog~~Sync~~PdelayReqInterval is set equal to mgtSettableLogPdelayReqInterval (see 14.16.13), and initialLogPdelayReqInterval is ignored.

Pdelay_Req messages shall be transmitted such that the value of the arithmetic mean of the intervals, in seconds, between Pdelay_Req message transmissions is not less than the value of $0.9 \times 2^{\text{currentLogPdelayReqInterval}}$.

NOTE 1—A minimum number of inter-message intervals is necessary to verify that a PTP Port meets these requirements. The arithmetic mean is the sum of the inter-message interval samples divided by the number of samples. For more detailed discussion of statistical analyses, see Papoulis [B25].

NOTE ~~1~~2—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~~3—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~~4—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 the requirements of the current subclause and IEEE Std 1588-2019, which require~~s~~ that the ~~logarithm to the base 2 of the mean~~ value of the arithmetic mean of the intervals, in seconds, between Pdelay_Req message transmissions is not less than the value of $0.9 \times 2^{\text{currentLogPdelayReqInterval}}$. ~~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 $2^{\text{currentLogPdelayReqInterval}}$ when measured relative to the synchronized time. The factor 0.9 allow a margin of 10% for the arithmetic mean of the successive intervals between Pdelay_Req messages.

Change 11.5.4 as follows:

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~~nrrPdelay 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.

Change 11.6 as follows:

11.6 Control of computation of neighborRateRatio

If the variable driftTrackingTlvSupport (see 10.2.4.27) is TRUE and nrrCompMethod (see 11.2.13.15) is equal to Sync, the value of neighborRateRatio (see 10.2.5.7) is set equal to nrrSync (see 11.2.13.13).

If the variable driftTrackingTlvSupport (see 10.2.4.27) is FALSE or nrrCompMethod (see 11.2.13.15) is equal to Pdelay, the value of neighborRateRatio (see 10.2.5.7) is set equal to nrrPdelay (see 11.2.13.13). The computation of nrrPdelay is controlled as described below. The description below applies only to the computation of nrrPdelay, and not to the computation of nrrSync.

The variable computeNeighborRateRatio (see 10.2.5.10) indicates whether ~~neighborRateRatio~~nrrPdelay 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~~nrrPdelay 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.

13. Media-dependent layer specification for interface to IEEE 802.3 Ethernet passive optical network link

13.3 Message format

13.3.1 TIMESYNC message

13.3.1.2 TIMESYNC message field specifications

Change 13.3.1.2.15 as follows:

13.3.1.2.15 domainNumber (UInteger8)

This field is specified as the gPTP ~~domain number~~domainNumber (see 8.1).

14. Timing and synchronization management

14.1 General

Change 14.1.1 as follows:

14.1.1 Data set hierarchy

This clause defines the set of managed objects, and their functionality, that allow administrative configuration of clock parameters and timing and synchronization protocols.

Management data models typically represent data for the physical device (i.e., time-aware system). The specifications for discovery, management address, and security for the physical device are typically covered by standards of the management mechanism, which are outside the scope of this standard. For the management information model of this standard, the scope of work is the data contained within a time-aware system. From a management perspective, the time-aware system contains a list of one or more PTP Instances. Each entry in the list is a set of managed data sets for the respective PTP Instance.

Conformance for each managed object is optional. This standard operates correctly using default values; therefore, management is not essential. Since the management mechanism is not limited to remote protocols (e.g., SNMP, NETCONF), management can use a local mechanism with a simple interface (e.g., DIP switches). Therefore, each product can determine the support of managed objects as appropriate for its management mechanism.

The following hierarchy summarizes the managed data sets within a gPTP Node:

a) instanceList[]
   1) defaultDS
   2) currentDS
   3) parentDS
   4) timePropertiesDS
   5) pathTraceDS
   6) acceptableTimeTransmitterTableDS
   7) ptpInstanceSyncDS
   8) driftTrackingDS
   9) portList[]
      i) portDS
      ii) descriptionPortDS
      iii) portStatisticsDS
      iv) acceptableTimeTransmitterPortDS
      v) externalPortConfigurationPortDS
      vi) asymmetryMeasurementModeDS
      vii) commonServicesPortDS
b) commonServices
   1) commonMeanLinkDelayService
      i) cmldsDefaultDS
      ii) cmldsLinkPortList[]
         — cmldsLinkPortDS
         — cmldsLinkPortStatisticsDS
         — cmldsAsymmetryMeasurementModeDS

2) hotStandbyService
   i) hotStandbySystemList[]
      — hotStandbySystemDS
      — hotStandbySystemDescriptionDS
3) Future common services can follow.

The instanceList is indexed using a number that is unique per PTP Instance within the time-aware system, applicable to the management context only (i.e., not used in PTP messages). The domainNumber of the PTP Instance must not be used as the index to instanceList since it is possible for a time-aware system to contain multiple PTP Instances using the same domainNumber. The portList is indexed using a number that is unique per logical port (i.e., PTP Port) in the PTP Instance (see 8.5.1). Since the portNumber of a logical port can have any value in the range 1, 2, 3, ..., 0xFFFE (see 8.5.2.3), the portList index and portNumber values for a logical port ~~will~~are not necessarily ~~be~~ the same. PTP Instances and logical ports may be created or deleted dynamically in implementations that support dynamic create/delete of devices. Unless otherwise indicated, the data sets and managed objects under the instanceList[] are maintained separately for each PTP Instance supported by the time-aware system.

Following the instanceList[] and all the data sets of each instanceList[] member is an overall structure for common services. That structure contains one sub-structure for each common service. At present there ~~is~~are ~~only one~~two common services, namely the Common Mean Link Delay Service (CMLDS) and the Hot Standby Service. ~~, and the~~The corresponding sub-structures ~~is~~are the commonMeanLinkDelayService structure and the hotStandbyService structure, respectively. The item "~~f~~Future common services can follow" is a placeholder for any common services that might be defined in the future. The commonMeanLinkDelayService structure and the hotStandbyService structure contain~~s~~ the data sets and lists that are needed by the Common Mean Link Delay Service and the Hot Standby Service, respectively.

The commonMeanLinkDelayService structure contains the cmldsLinkPortList, which is a list of CMLDS logical ports, i.e., Link Ports (see 11.2.17), of the time-aware system that ~~will~~ runs the common service. The CMLDS must be implemented (i.e., a CMLDS executable must be present) on every physical port for which there is a PTP Port of a PTP Instance that can use the CMLDS (i.e., where portDS.delayMechanism of that PTP Instance can have the value COMMON_P2P). Therefore, the cmldsLinkPortList[] must include Link Ports that correspond to all such physical ports. As is the case for the portList of a PTP Instance, the cmldsLinkPortList is indexed using a number that is unique per Link Port that invokes the CMLDS (see 8.5.1). Since the portNumber of a logical port (i.e., PTP Port or CMLDS Link Port) can have any value in the range 1, 2, 3, ..., 0xFFFE (see 8.5.2.3), the cmldsLinkPortList index and cmldsLinkPortDS.portIdentity.portNumber values for a logical port of the Common Mean Link Delay Service ~~will~~are not necessarily ~~be~~ the same. CMLDS Link Ports may be created or deleted dynamically in implementations that support dynamic create/delete of devices.

The Common Mean Link Delay Service Data Sets are not maintained separately for each PTP Instance. Rather, a single copy of the commonServices.cmldsDefaultDS is maintained for the time-aware system, and a single copy of each data set under the cmldsLinkPortList[] is maintained per Link Port of the time-aware system.

A PTP Instance can use the commonServicesPortDS to determine which Link Port it must use when it obtains information provided by the Common Mean Link Delay Service (see 14.14).

The hotStandbyService structure contains the hotStandbySystemList, which is a list of instances of the Hot Standby Service. Since a Hot Standby Service Instance is associated with two PTP Instances, but a time-aware system can have more than two PTP Instances, there can, in general, be multiple instances of the Hot Standby Service (i.e., one Hot Standby Service Instance associated with each set of two distinct PTP Instances). A hotStandbySystemDS and a hotStandbySystemDescriptionDS are maintained for each instance of the Hot Standby Service.

Each instance of the Hot Standby Service (i.e., each hotStandbySystemList member) is associated with two PTP Instances, i.e., the primary PTP Instance and the secondary PTP Instance (see 18.1 and 18.2). The indices of these PTP Instances in the instanceList are contained in the members primaryPtpInstanceIndex and secondaryPtpInstanceIndex, respectively, of the hotStandbySystemDS for this instance of the Hot Standby Service.

NOTE—This hierarchy is intended to support a wide variety of time-aware system implementations. Examples include the following:

a) A time-aware system containing four PTP Relay Instances, each of which use the same physical ports, but different domainNumber values.~~-~~

b) A time-aware system containing four PTP Relay Instances and two hotStandbySystem entities, with two PTP Instances associated with one of the hotStandbySystem entities and the other two PTP Instances associated with the other hotStandbySystem entity.

~~b~~c) A time-aware system that represents a chassis with slots for switch/router cards, where each switch/router card is represented as a PTP Instance using distinct physical ports and all PTP Instances can use the same domainNumber.

14.1.2 Data set descriptions

Insert 14.1.2 g), and renumber the list as appropriate:

g) The PTP Instance Synchronization Parameter Data Set (ptpInstanceSyncDS in 14.1.1; see Table 14-7), which represents time synchronization status information for the PTP Instance.

Insert 14.1.2 h), and renumber the list as appropriate.

h) The Drift Tracking Parameter Data Set (driftTrackingDS in 14.1.1; see Table xxx), which contains a manged object used to enable or disable the Drift_Tracking TLV.

Insert 14.1.2 r), and renumber the list as appropriate:

r) The Hot Standby Service Hot Standby System Parameter Data Set (hotStandbySystemDS in 14.1.1; see Table 14-20), which describes the attributes of the respective instance of the Hot Standby Service.

Insert 14.1.2 s), and renumber the list as appropriate:

s) The Hot Standby Service Hot Standby System Description Parameter Data Set (hotStandbySystemDescriptionDS, see Table 14-21), which represents descriptive information for the instance of the Hot Standby Service.

14.2 Default Parameter Data Set (defaultDS)

Change 14.2.16 as follows:

14.2.16 domainNumber

The value is the ~~domain number~~domainNumber of the gPTP domain for this instance of gPTP supported by the time-aware system (see 8.1).

NOTE—The PTP Instance for which domainNumber is 0 has constraints applied to it, e.g., timescale (see 8.2.1).

14.3 Current Parameter Data Set (currentDS)

Change 14.3.3 as follows:

14.3.3 offsetFromTimeTransmitter

The value is an implementation-specific or profile standard-specific representation of the current value of the time difference between a timeReceiver and the Grandmaster Clock, as computed by the timeReceiver, and as specified in 10.2.10. The value is computed by an algorithm that is implementation-specific or profile standard-specific. The data type shall be TimeInterval. The default value is implementation specific.

NOTE—For example, the inputs to this implementation-specific algorithm could be the successive values of clockSourcePhaseOffset (see 10.2.4.7) of the ClockTimeTransmitterSyncOffset state machine (see 10.2.10 and Figure 10-6).

14.4 Parent Parameter Data Set (parentDS)

Insert 14.4.8, and renumber subsequent subclauses as appropriate:

14.4.8 gmPresent

The value is the value of the per PTP Instance global variable gmPresent (see 10.2.4.13).

Change Table 14-3 as follows:.

Table 14-3—parentDS table

Name	Data type	Operations supported[1]	References
parentPortIdentity	PortIdentity (see 6.4.3.7)	R	14.4.2
cumulativeRateRatio	Integer32	R	14.4.3
grand~~M~~masterIdentity	ClockIdentity	R	14.4.4
grandmasterClockQuality.clock Class	UInteger8	R	14.4.5.2; 7.6.2.5 of IEEE Std 1588-2019
grandmasterClockQuality.clock Accuracy	Enumeration8	R	14.4.5.3; 7.6.2.6 of IEEE Std 1588-2019
grandmasterClockQuality.offset ScaledLogVariance	UInteger16	R	14.4.5.4
grandmasterPriority1	UInteger8	R	14.4.6
grandmasterPriority2	UInteger8	R	14.4.7
gmpresent	Boolean	R	14.4.8

[1] R = Read only access; RW = Read/write access.

Insert 14.8 and Table 14-7, and renumber subsequent subclauses and tables as appropriate:

14.8 PTP Instance Synchronization Parameter Data Set (ptpInstanceSyncDS)

14.8.1 General

The ptpInstanceSyncDS describes the synchronization status of the PTP Instance. The ptpInstanceSyncDS shall be implemented if the optional hot standby feature (see Clause 18) is implemented and may be implemented otherwise.

14.8.2 isSynced

The value of the global variable isSynced (see 18.4.1.1).

14.8.3 offsetFromTimeTransmitterMax

The value is the threshold for offsetFromTimeTransmitter (see 18.4.1.2), below which the PTP Instance is considered to be synchronized. For values less than or equal to zero, the PTP Instance is considered synchronized if and only if offsetFromTimeTransmitter is zero. For values greater than zero, the PTP Instance is considered synchronized if and only if the equation

$$-\mathrm{offsetFromTimeTransmitterThreshold} \le \mathrm{offsetFromTimeTransmitter} \le \mathrm{offsetFromTimeTransmitterThreshold}$$

is satisfied.

14.8.4 rxSyncCountTimeReceiverPThresh

The value of rxSyncCountTimeReceiverPThresh is the threshold for rxSyncCountTimeReceiverP (see 18.4.1.4), above which the PTP Instance is considered to be synchronized.

14.8.5 offsetMaxExceededCountThresh

The value of offsetMaxExceededCountThresh (see 18.4.1.7) is the threshold for the number of consecutive exceedances of offsetFromTimeTransmitterMax (see 18.4.1.3) by offsetFromTimeTransmitter (see 18.4.1.2), at which isSynced (see 18.4.1.1) is no longer TRUE.

14.8.6 offsetMaxMetCountThresh

The value of offsetMaxMetCountThresh (18.4.1.9) is the threshold for the number of consecutive occurrences of offsetFromTimeTransmitter (see 18.4.1.2) being within offsetFromTimeTransmitterMax (see 18.4.1.3), at which isSynced (see 18.4.1.1) is changed to TRUE if it currently is FALSE.

14.8.7 ptpInstanceSyncDS table

There is one ptpInstanceSyncDS table per PTP Instance, as detailed in Table 14-7.

Table 14-7—ptpInstanceSyncDS table

Name	Data type	Operations supported[1]	References
isSynced	Boolean	R	14.8.2
offsetFromTimeTransmitterMax	TimeInterval	RW	14.8.3
rxSyncCountTimeReceiverPThres h	UInteger32	RW	14.8.4
offsetMaxExceededCountThresh	UInteger32	RW	14.8.5
offsetMaxMetCountThresh	UInteger32	RW	14.8.6

[1] R = Read only access; RW = Read/write access.

Insert 14.9 and Table 14-7, and renumber subsequent subclauses as appropriate.

14.9 Drift Tracking Parameter Data Set (driftTrackingDS)

14.9.1 General

The driftTrackingDS contains a managed object that is used to enable or disable the optional Drift_Tracking TLV.

14.9.2 driftTrackingTlvSupport

The value of driftTrackingTlvSupport indicates whether the Drift_Tracking TLV is enabled or disabled. If the value is TRUE, the TLV is enabled, i.e., the global variable driftTrackingTlvSupport (see 10.2.4.27) is set to TRUE. If the value is FALSE, the TLV is disabled, i.e., the global variable driftTrackingTlvSupport is set to FALSE.

14.9.3 driftTrackingDS table

There is one driftTrackingDS table per PTP Instance, as detailed inTable 14-8.

Table 14-8—driftTrackingDS table

Name	Data type	Operations supported[1]	References
driftTrackingTlvSupport	Boolean	RW	14.9.2

[1] R = Read only access; RW = Read/write access.

14.8 Port Parameter Data Set (portDS)

Change 14.8.5 as follows:

14.8.5 delayMechanism

The value of delayMechanism indicates the mechanism for measuring mean propagation delay and neighbor rate ratio on the link attached to this PTP Port and is taken from the enumeration in ~~Table 14-9~~Table 14-9. If the ~~domain number~~domainNumber is not 0, portDS.delay mechanism must not be P2P (see 11.2.17).

Table 14-9—delayMechanism enumeration

Delay mechanism	Value	Specification
P2P	02	The PTP Port uses the transport-specific peer-to-peer delay mechanism
COMMON_P2P	03	The PTP Port uses the CMLDS peer-to-peer delay mechanism
SPECIAL	04	The PTP Port uses a transport that has a native time transfer mechanism and, therefore, does not use the peer-to-peer delay mechanism (e.g., IEEE 802.11, IEEE 802.3 EPON)
	All other values reserved	
NOTE—The enumeration values are consistent with Table 21 in IEEE Std 1588-2019.		

Change 14.8.6 as follows:

14.8.6 isMeasuringDelay

The value is equal to the value of the per-port, per PTP Instance global variable~~Boolean~~ isMeasuringDelay (see 11.2.13.6 and 16.4.3.3).

Insert 14.8.55, and renumber subsequent subclauses as appropriate:

14.8.55 gptpCapableStateMachinesEnabled

A Boolean that is used to enable or disable the GptpCapableTransmit, GptpCapableReceive, and GptpCapableIntervalSetting state machines. If the value is TRUE, the GptpCapableTransmit and GptpCapableReceive state machines are enabled, and the GptpCapableIntervalSetting state machine is enabled if it is implemented. If the value is FALSE, the GptpCapableTransmit and GptpCapableReceive state machines are disabled, and the GptpCapableIntervalSetting state machine is disabled if it is implemented.. The default value is TRUE.

14.8.56 nrrPdelay

The value of the global variable nrrPdelay (see 11.2.13.13).

The value is an estimate of the ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this Link Port, to the frequency of the LocalClock entity of this time-aware system (see 10.2.5.7). nrrPdelay is expressed as the fractional frequency offset stored in the global variable nrrPdelay (see 11.2.13.13) multiplied by 2^{41}, i.e., the quantity (nrrPdelay – 1.0)(2^{41}).

14.8.57 nrrSync

The value of the global variable nrrSync (see 11.2.13.14).

The value is an estimate of the ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this Link Port, to the frequency of the LocalClock entity of this time-aware system (see 10.2.5.7). nrrSync is expressed as the fractional frequency offset stored in the global variable nrrSync (see 11.2.13.13) multiplied by 2^{41}, i.e., the quantity (nrrSync – 1.0)(2^{41}).

14.8.58 nrrCompMethod

The value of the global variable nrrCompMethod (see 11.2.13.15).

14.8.55 portDS table

Insert the following items after the final item of Table 14-10:

.

Table 14-10—portDS table

Name	Data type	Operations supported[1]	References
gptpCapableStateMachinesEnabled	Boolean	RW	14.8.55
nrrPdelay	Integer32	R	14.8.56
nrrSync	Integer32	R	14.8.57
nrrCompMethod	Enumeration2	RW	14.8.58

[1] R = Read only access; RW = Read/write access.

14.10 Port Parameter Statistics Data Set (portStatisticsDS)

Change the heading of 14.10.9 as follows:

14.10.9 rxP~~TP~~tpPacketDiscardCount

Insert 14.10.20, and renumber subsequent subclauses as appropriate:

14.10.20 rxSyncCountTimeReceiverP

This counter increments ~~every~~whenever time synchronization information is received on ~~the~~a PTP Port ~~whose~~when its port state is TimeReceiverPort. The receipt of time synchronization information is denoted by one of the following events:

— A transition to TRUE from FALSE of the rcvdSync variable of the MDSyncReceiveSM state machine (see 11.2.14.1.2 and Figure 11-6) when in the DISCARD, WAITING_FOR_SYNC, or WAITING_FOR_FOLLOW_UP states; or
— rcvdIndication transitions to TRUE (see Figure 12-7).

This counter is initialized to zero, and resets to zero when the port state is not TimeReceiverPort.

Change Table 14-12 as follows:

.

Table 14-12—portStatisticsDS table

Name	Data type	Operations supported[1]	References
rxSyncCount	UInteger32	R	14.10.2
rxOneStepSyncCount	UInteger32	R	14.10.3
rxFollowUpCount	UInteger32	R	14.10.4
rxPdelayRequestCount	UInteger32	R	14.10.5
rxPdelayResponseCount	UInteger32	R	14.10.6
rxPdelayResponseFollowUpCount	UInteger32	R	14.10.7
rxAnnounceCount	UInteger32	R	14.10.8
rxP~~TP~~tpPacketDiscardCount	UInteger32	R	14.10.9
syncReceiptTimeoutCount	UInteger32	R	14.10.10
announceReceiptTimeoutCount	UInteger32	R	14.10.11
pdelayAllowedLostResponsesExceededCount	UInteger32	R	14.10.12
txSyncCount	UInteger32	R	14.10.13
txOneStepSyncCount	UInteger32	R	14.10.14
txFollowUpCount	UInteger32	R	14.10.15
txPdelayRequestCount	UInteger32	R	14.10.16
txPdelayResponseCount	UInteger32	R	14.10.17
txPdelayResponseFollowUpCount	UInteger32	R	14.10.18
txAnnounceCount	UInteger32	R	14.10.19
rxSyncCountTimeReceiverP	UInteger32	R	14.10.20

[1]R= Read only access.

14.13 Asymmetry Measurement Mode Parameter Data Set (asymmetryMeasurementModeDS)

Change 14.13.3 as follows:

14.13.3 asymmetryMeasurementModeDS table

There is one asymmetryMeasurementModeDS table for the single PTP Instance whose ~~domain number~~domainNumber is 0, per PTP Port, as detailed in Table 14-15. This data set is used only when there is a single gPTP domain and CMLDS is not used.

Table 14-15—asymmetryMeasurementModeDS table

Name	Data type	Operations supported[1]	References
asymmetryMeasurementMode	Boolean	RW	14.13.2

[1] R = Read only access; RW = Read/write access.

14.16 Common Mean Link Delay Service Link Port Parameter Data Set (cmldsLinkPortDS)

Change 14.16.4 as follows:

14.16.4 isMeasuringDelay

The value is equal to the value of the instance of the Boolean isMeasuringDelay (see 11.2.13.6 and 16.4.3.3) that is per LinkPort and across all domains.

Change 14.16.9 as follows:

14.16.9 ~~neighborRateRatio~~nrrPdelay

The value is an estimate of the ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this Link Port, to the frequency of the LocalClock entity of this time-aware system (see 10.2.5.7). ~~neighborRateRatio~~nrrPdelay is expressed as the fractional frequency offset stored in the global variable nrrPdelay (see 11.2.13.13) multiplied by 2^{41}, i.e., the quantity (~~neighborRateRatio~~nrrPdelay – 1.0)(2^{41}).

NOTE—This data set member corresponds to the scaledNeighborRateRatio member of the CommonMeanLinkDelayInformation Structure in 16.6.3.2 of IEEE Std 1588-2019.

14.16.27 cmldsLinkPortDS table

Change the eighth row of Table 14-18 (not including the header row) as follows:

.

Table 14-18—cmldsLinkPortDS table

Name	Data type	Operations supported[1]	References
~~neighborRateRatio~~nrrPdelay	Integer32	R	14.16.9

[1] R = Read only access; RW = Read/write access.

14.17 Common Mean Link Delay Service Link Port Parameter Statistics Data Set (cmldsLinkPortStatisticsDS)

Change the heading of 14.17.5 as follows

14.17.5 rxP~~TP~~tpPacketDiscardCount

Change Table 14-19 as follows:.

Table 14-19—cmldsLinkPortStatisticsDS table

Name	Data type	Operations supported[1]	References
rxPdelayRequestCount	UInteger32	R	14.17.2
rxPdelayResponseCount	UInteger32	R	14.17.3
rxPdelayResponseFollowUpCount	UInteger32	R	14.17.4
rxP~~TP~~tpPacketDiscardCount	UInteger32	R	14.17.5
pdelayAllowedLostResponsesExceededCount	UInteger32	R	14.17.6
txPdelayRequestCount	UInteger32	R	14.17.7
txPdelayResponseCount	UInteger32	R	14.17.8
txPdelayResponseFollowUpCount	UInteger32	R	14.17.9

[1] R= Read only access.

Insert 14.19, and renumber any subsequent subclauses as appropriate:

14.19 Hot Standby System Parameter Data Set (hotStandbySystemDS)

14.19.1 General

The hotStandbySystemDS describes the attributes of the respective instance of the Hot Standby Service.

14.19.2 primaryPtpInstanceIndex

The value of primaryPtpInstanceIndex is the index (see 14.1.1) of the primary PTP Instance associated with this hotStandbySystem instance.

14.19.3 secondaryPtpInstanceIndex.

The value of secondaryPtpInstanceIndex is the index (see 14.1.1) of the secondaryPTP Instance associated with this hotStandbySystem instance.

14.19.4 hotStandbySystemEnable

The value is the hotStandbySystemEnable attribute of the HotStandbySystem entity (see 18.5.1.2).

14.19.5 hotStandbySystemState

The value of hotStandbySystemState is the state of the hotStandbySystem, i.e., the value of the global variable hotStandbySystemState (see 18.5.1.1).

14.19.6 hotStandbySystemSplitFunctionality

If the value is TRUE, the optional split functionality (see 18.5.3.4) is used. If the value is FALSE, the optional split functionality is not used.

14.19.7 primarySecondaryOffset

The absolute value of the difference between the clockTimeReceiverTimes (see 10.2.4.3) of the primary and secondary PTP Instances.

14.19.8 primarySecondaryOffsetThresh

The threshold for hotStandbySystemDS.primarySecondaryOffset (see 14.19.7), above which the hotStandbySytemState transitions from REDUNDANT to NOT_REDUNDANT, or does not transition from NOT_REDUNDANT or OUT_OF_SYNC to REDUNDANT even if other conditions for these transitions are satisfied.

14.19.9 hotStandbySystemLogSyncTimeThresh

The value of hotStandbySystemLogSyncTimeThresh is the logarithm to base 2 of the time interval, in seconds, after which the hotStandbySystem transitions from the OUT_OF_SYNC state to either the NOT-REDUNDANT or REDUNDANT state, or from the NOT_REDUNDANT to the REDUNDANT state, if all other conditions for the respective transition are met. The value -128 means that the transition time is zero, i.e., the transition occurs immediately.

14.19.10 hotStandbySystemDS table

There is one hotStandbyDS table per time-aware system, as detailed in Table 14-20.

Table 14-20—hotStandbySystemDS table

Name	Data type	Operations supported[1]	References
primaryPtpInstanceIndex	UInteger32	RW	14.19.2
secondaryPtpInstanceIndex	UInteger32	RW	14.19.3
hotStandbySystemEnable	Boolean	RW	14.19.4
hotStandbySystemState	Enumeration8	R	14.19.5
hotStandbySystemSplitFunctionality	Boolean	RW	14.19.6
primarySecondaryOffset	ScaledNs	R	14.19.7
primarySecondaryOffsetThresh	ScaledNs	RW	14.19.8
hotStandbySystemLogSyncTimeThresh	Integer8	RW	14.19.9

[1] R = Read only access; RW = Read/write access.

Insert 14.20, and renumber any subsequent subclauses as appropriate:

14.20 Hot Standby System Description Parameter Data Set (hotStandbySystemDescriptionDS)

14.20.1 General

The hotStandbySystemDescriptionDS contains descriptive information for the respective instance of the Hot Standby Service.

14.20.2 userDescription

The user description is a character string whose maximum length is 128.

14.20.3 hotStandbySystemDescriptionDS table

There is one hotStandbySystemDescriptionDS table per hotStandbySystem instance, as detailed in Table 14-21.

Table 14-21—hotStandbySystemDescriptionDS table

Name	Data type	Operations supported[1]	References
userDescription	Octet128	RW	14.20.2

[1] RW= Read/write access.

16. Media-dependent layer specification for CSN

16.4 Path delay measurement over a CSN backbone

16.4.3 Path delay measurement between CSN nodes

16.4.3.2 Path delay measurement without network clock reference

Change the first paragraph of 16.4.3.2 as follows:

Each CSN node has a free-running local clock. The path delay measurement uses the peer-to-peer delay mechanism protocol, messages, and state machines described in Clause 11 for full-duplex point-to-point links, as illustrated by Figure 16-5. The criteria of 11.2.17 for determining whether the peer-to-peer delay mechanism is ~~instance~~ the transport-specific peer-to-peer delay mechanism or ~~is provided by~~the CMLDS peer-to-peer delay mechansim apply here.

Change 16.4.3.3 as follows:

16.4.3.3 Intrinsic CSN path delay measurement

Some CSN technologies feature a native mechanism that provides a path delay measurement with accuracy similar to the accuracy the peer delay protocol provides. For these CSNs, the path delay can be provided using the native measurement method rather than using the Pdelay protocol defined in 11.2.19 and 11.2.20. Such a situation is described in more detail as follows. The CSN MD entity populates the following per-PTP Port and MD-entity global variables (described respectively in 10.2.5 and 11.2.13) as indicated:

— asCapable (10.2.5.1) is set to TRUE.
— neighborRateRatio (10.2.5.7) is set to the value provided by the native CSN measurement.
— meanLinkDelay (10.2.5.8) is set to the value provided by the native CSN measurement.
— computeNeighborRateRatio (10.2.5.10) is set to FALSE.
— computeMeanLinkDelay (10.2.5.11) is set to FALSE.
— isMeasuringDelay (11.2.13.6) is set to TRUE to indicate that the CSN MD entity is measuring path delay (in this case, using its internal mechanism).
— domainNumber (8.1) is set to the ~~domain number~~domainNumber of this gPTP domain.

16.5 Synchronization messages

16.5.3 Synchronization message propagation on a CSN with network reference clock

16.5.3.2 CSN ingress node

16.5.3.2.2 CSN TLV

Change 16.5.3.2.2.10 as follows:

16.5.3.2.2.10 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of this gPTP domain.

17. YANG Data Model

17.1 YANG framework

Change 17.1.1 as follows:

17.1.1 Relationship to the IEEE Std 1588 data model

The YANG data models specified in this standard are based on, and augment, those specified in IEEE Std 1588. In particular the ieee802-dot1as-ptp.yang module imports the ieee1588-ptp module as a whole, augmenting that module as necessary to meet the requirements of this standard. In addition, the ieee802-dot1as-hs.yang module imports the ieee1588-ptp and ieee802-dot1as-ptp modules as a whole, augmenting those modules as necessary to to meet the requirements of this standard. Th~~is~~ese imports make~~s~~ existing and new IEEE Std 1588 YANG capabilities not specifically addressed by the present standard available to its implementors without delay, without the need to revise or amend IEEE Std 802.1AS.

Some of the data sets in Clause 14 (e.g., defaultDS) are derived from IEEE Std 1588, and some of the data sets are unique to IEEE Std 802.1AS (i.e., not derived from IEEE Std 1588). For each data set in Clause 14 that is derived from IEEE Std 1588, a portion of the members are derived from IEEE Std 1588, and the remaining members are unique to IEEE Std 802.1AS. For the members that are derived from IEEE Std 1588, the specifications in both standards are analogous (i.e., same name, data type, semantics, etc).

The YANG data model for IEEE Std 1588-2019 is published as amendment IEEE Std 1588e. The YANG module of IEEE Std 1588e (ieee1588-ptp.yang) contains the hierarchy (tree) of data sets and their members.

The YANG modules of this clause (ieee802-dot1as-ptp.yang and ieee802-dot1as-hs.yang) use~~s~~ the YANG "import" statement to import the YANG module of IEEE Std 1588e. This effectively uses the IEEE Std 1588 YANG tree as the foundation of the IEEE Std 802.1AS YANG tree. By importing the tree and its data set containers, all members from Clause 14 that are derived from IEEE Std 1588 are also imported.

The core of the YANG modules for IEEE Std 802.1AS consists of YANG "augment" statements, used to add members to the tree that are unique for IEEE Std 802.1AS.

NOTE 2 - IETF RFC8575 [B48] is the standard YANG data model for IEEE Std 1588-2008. The YANG data model of IEEE Std 1588e is effectively a newer version of RFC8575. Therefore, the YANG module of RFC8575 is not imported by the YANG module of this clause.

Change 17.2 as follows:

17.2 IEEE 802.1AS YANG models

This clause uses a UML-like representation to provide an overview of the hierarchy of the IEEE Std 802.1AS YANG data model.

A representation of the management model is provided in Figures 17-1 through 17-4. The purpose of the diagram is to express the model design in a concise manner. The structure of the representation shows the name of the object followed by a list of properties for the object. The properties indicate their type and accessibility. It should be noted that the representation is meant to express simplified semantics for the properties. It is not meant to provide the specific datatype used to encode the object in either MIB or YANG. In the representation, a box with a white background represents information that comes from sources outside of this IEEE standard. A box with a gray background represents objects that are defined by this IEEE standard.

NOTE 1 - OMG UML 2.5 [B49] conventions together with C++ language constructs are used in this clause as a representation to convey model structure and relationships.

NOTE 2 - This standard specifies YANG for Clause 14 of this standard. There are optional features in the YANG module of IEEE Std 1588 that are not specified in Clause 14, and therefore not shown in the figures of this subclause. If optional IEEE Std 1588 YANG features are implemented, conformance is specified by IEEE Std 1588.

For all figures, Clause 14 data that is imported from the ieee1588-ptp.yang module is shown in white, and Clause 14 data in augments of ieee802-dot1as-ptp.yang is shown in gray.

Figure 17-1 provides an overview of the IEEE Std 802.1AS YANG tree. The top level instance-list provides the list of one or more PTP Instances, each with data sets. For each PTP Instance, port-ds-list provides the list of one or more PTP Ports, each with data sets. The common-services apply to all PTP Instances, including the Common Mean Link Delay Service (cmlds).

Figure provides detail for the data sets of each PTP Instance, including each data set member.

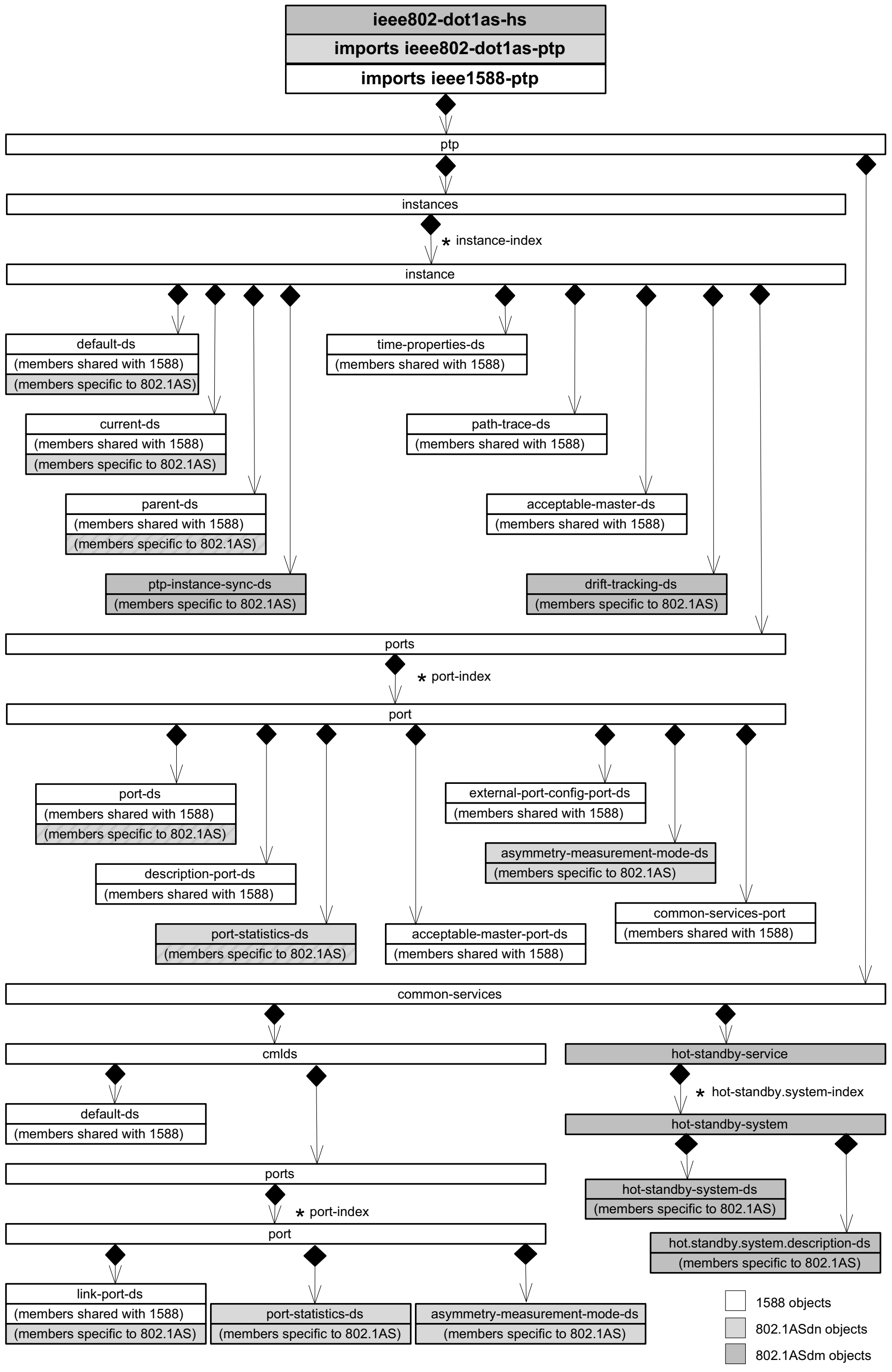
Figure provides detail for the data sets of each PTP Port, including each data set member.

NOTE 2 - 14.8.4 specifies ptpPortEnabled (ptp-port-enabled), which is provided in YANG as the semantically equivalent node in ieee1588-ptp named port-enable (in port-ds of Figure). 14.8.15 specifies mgtSettableLogAnnounceInterval (mgt-settable-log-announce-interval), which is provided in YANG as the semantically equivalent node in ieee1588-ptp named log-announce-interval (in port-ds of Figure). 14.8.20 specifies mgtSettableLogSyncInterval (mgt-settable-log-sync-interval), which is provided in YANG as the semantically equivalent node in ieee1588-ptp named log-sync-interval (in port-ds of Figure).

Figure provides detail for the common services, including each data set member. The Common Mean Link Delay Service (cmlds) has a data sets for the service itself (e.g., default-ds), and data sets for each ~~PTP~~ Link Port. The Hot Standby Service has data sets for each HotStandbySystem.

NOTE 3 - 14.16.9 specifies neighborRateRatio (neighbor-rate-ratio), which is provided in YANG as the semantically equivalent node in ieee1588-ptp named scaled-neighbor-rate-ratio (in link-port-ds of Figure).

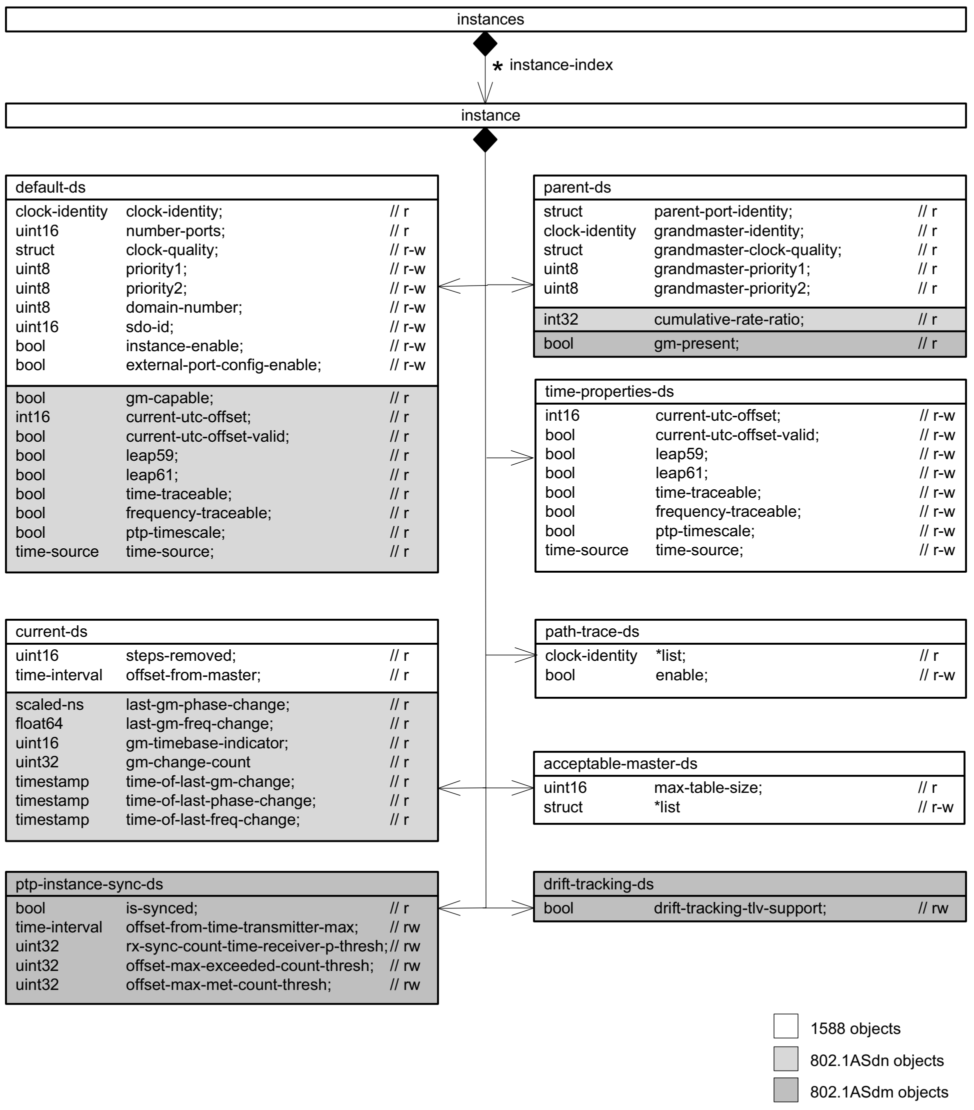
Replace Figure 17-1 with the following:


ieee802-dot1as-hs
imports ieee802-dot1as-ptp
imports ieee1588-ptp
ptp
instances
* instance-index
instance
default-ds
(members shared with 1588)
(members specific to 802.1AS)
current-ds
(members shared with 1588)
(members specific to 802.1AS)
parent-ds
(members shared with 1588)
(members specific to 802.1AS)
ptp-instance-sync-ds
(members specific to 802.1AS)
time-properties-ds
(members shared with 1588)
path-trace-ds
(members shared with 1588)
acceptable-master-ds
(members shared with 1588)
drift-tracking-ds
(members specific to 802.1AS)
ports
* port-index
port
port-ds
(members shared with 1588)
(members specific to 802.1AS)
description-port-ds
(members shared with 1588)
port-statistics-ds
(members specific to 802.1AS)
acceptable-master-port-ds
(members shared with 1588)
external-port-config-port-ds
(members shared with 1588)
asymmetry-measurement-mode-ds
(members specific to 802.1AS)
common-services-port
(members shared with 1588)
common-services
cmlds
default-ds
(members shared with 1588)
ports
* port-index
port
link-port-ds
(members shared with 1588)
(members specific to 802.1AS)
port-statistics-ds
(members specific to 802.1AS)
asymmetry-measurement-mode-ds
(members specific to 802.1AS)
hot-standby-service
* hot-standby.system-index
hot-standby-system
hot-standby-system-ds
(members specific to 802.1AS)
hot.standby.system.description-ds
(members specific to 802.1AS)
1588 objects
802.1ASdn objects
802.1ASdm objects

NOTE 1—This figure differs from the 2020 edition of this standard with Corrigendum 1 applied and as amended by IEEE Std 802.1ASdn in that managed objects needed for the hot standby and Drift_Tracking TLV features are added.

Figure 17-1—Overview of YANG tree

Replace Figure with the following:



```
default-ds
clock-identity   clock-identity;                // r
uint16           number-ports;                  // r
struct           clock-quality;                 // r-w
uint8            priority1;                     // r-w
uint8            priority2;                     // r-w
uint8            domain-number;                 // r-w
uint16           sdo-id;                        // r-w
bool             instance-enable;               // r-w
bool             external-port-config-enable;   // r-w
bool             gm-capable;                    // r
int16            current-utc-offset;            // r
bool             current-utc-offset-valid;      // r
bool             leap59;                        // r
bool             leap61;                        // r
bool             time-traceable;                // r
bool             frequency-traceable;           // r
bool             ptp-timescale;                 // r
time-source      time-source;                   // r
```

```
parent-ds
struct           parent-port-identity;          // r
clock-identity   grandmaster-identity;          // r
struct           grandmaster-clock-quality;     // r
uint8            grandmaster-priority1;         // r
uint8            grandmaster-priority2;         // r
int32            cumulative-rate-ratio;         // r
bool             gm-present;                    // r
```

```
time-properties-ds
int16            current-utc-offset;            // r-w
bool             current-utc-offset-valid;      // r-w
bool             leap59;                        // r-w
bool             leap61;                        // r-w
bool             time-traceable;                // r-w
bool             frequency-traceable;           // r-w
bool             ptp-timescale;                 // r-w
time-source      time-source;                   // r-w
```

```
current-ds
uint16           steps-removed;                 // r
time-interval    offset-from-master;            // r
scaled-ns        last-gm-phase-change;          // r
float64          last-gm-freq-change;           // r
uint16           gm-timebase-indicator;         // r
uint32           gm-change-count                // r
timestamp        time-of-last-gm-change;        // r
timestamp        time-of-last-phase-change;     // r
timestamp        time-of-last-freq-change;      // r
```

```
path-trace-ds
clock-identity   *list;                         // r
bool             enable;                        // r-w
```

```
acceptable-master-ds
uint16           max-table-size;                // r
struct           *list                          // r-w
```

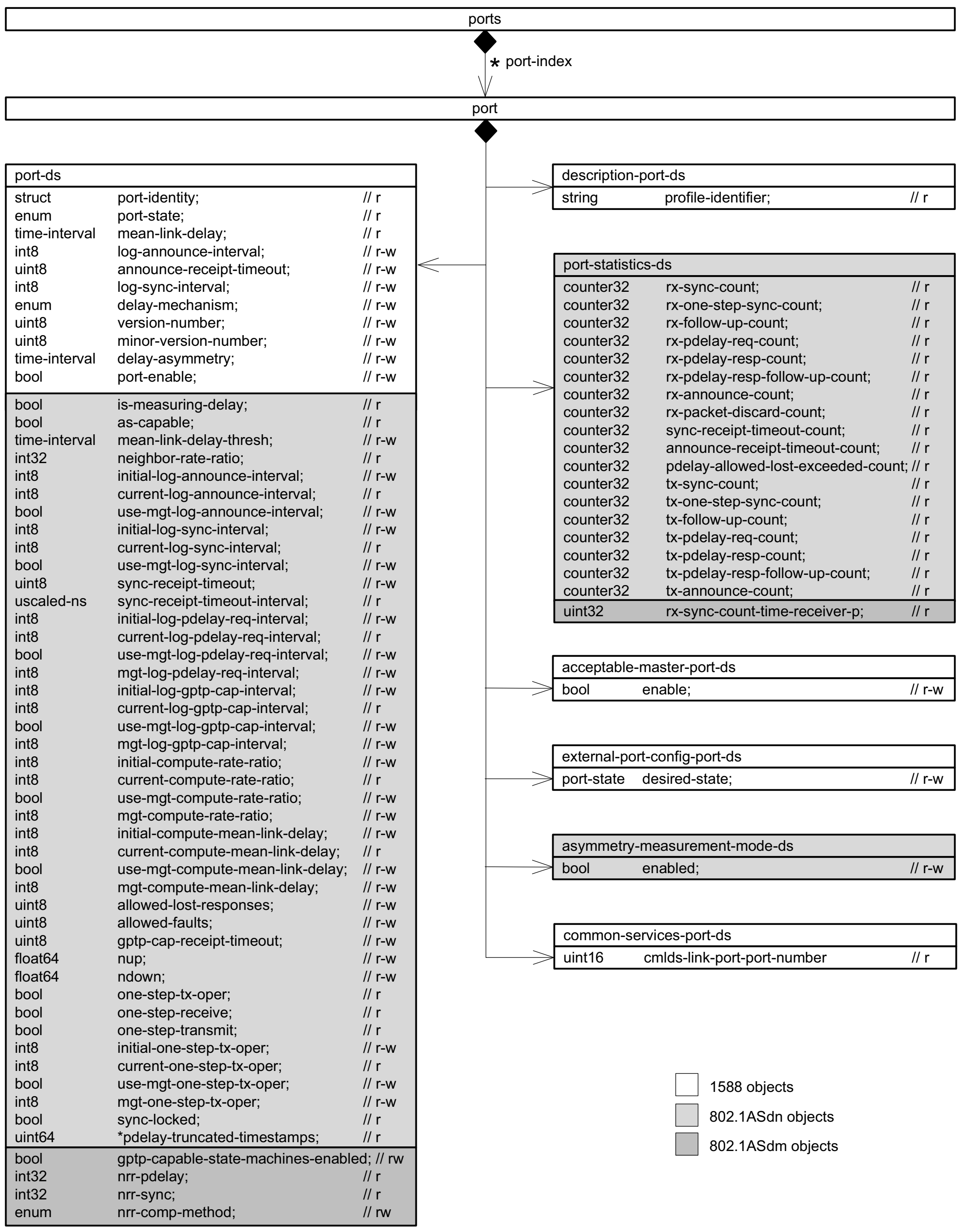
```
ptp-instance-sync-ds
bool             is-synced;                                 // r
time-interval    offset-from-time-transmitter-max;          // rw
uint32           rx-sync-count-time-receiver-p-thresh;      // rw
uint32           offset-max-exceeded-count-thresh;          // rw
uint32           offset-max-met-count-thresh;               // rw
```

```
drift-tracking-ds
bool             drift-tracking-tlv-support;    // rw
```

NOTE 2—This figure differs from the 2020 edition of this standard with Corrigendum 1 applied and as amended by IEEE Std 802.1ASdn in that managed objects needed for the hot standby and Drift_Tracking TLV features are added.

Figure 17-2—PTP Instance detail

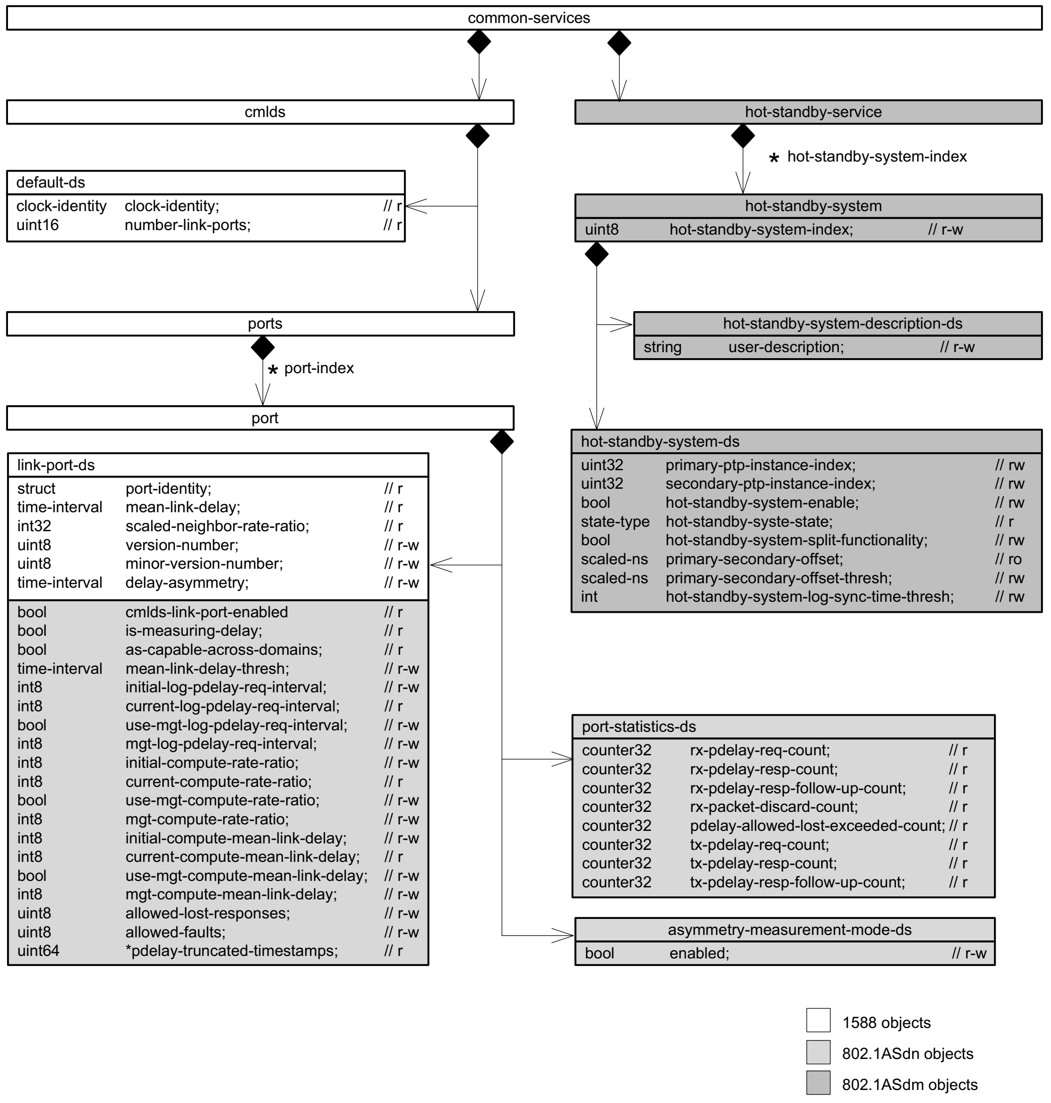
Replace Figure with the following:


ports
* port-index
port
port-ds
struct port-identity; // r
enum port-state; // r
time-interval mean-link-delay; // r
int8 log-announce-interval; // r-w
uint8 announce-receipt-timeout; // r-w
int8 log-sync-interval; // r-w
enum delay-mechanism; // r-w
uint8 version-number; // r-w
uint8 minor-version-number; // r-w
time-interval delay-asymmetry; // r-w
bool port-enable; // r-w
bool is-measuring-delay; // r
bool as-capable; // r
time-interval mean-link-delay-thresh; // r-w
int32 neighbor-rate-ratio; // r
int8 initial-log-announce-interval; // r-w
int8 current-log-announce-interval; // r
bool use-mgt-log-announce-interval; // r-w
int8 initial-log-sync-interval; // r-w
int8 current-log-sync-interval; // r
bool use-mgt-log-sync-interval; // r-w
uint8 sync-receipt-timeout; // r-w
uscaled-ns sync-receipt-timeout-interval; // r
int8 initial-log-pdelay-req-interval; // r-w
int8 current-log-pdelay-req-interval; // r
bool use-mgt-log-pdelay-req-interval; // r-w
int8 mgt-log-pdelay-req-interval; // r-w
int8 initial-log-gptp-cap-interval; // r-w
int8 current-log-gptp-cap-interval; // r
bool use-mgt-log-gptp-cap-interval; // r-w
int8 mgt-log-gptp-cap-interval; // r-w
int8 initial-compute-rate-ratio; // r-w
int8 current-compute-rate-ratio; // r
bool use-mgt-compute-rate-ratio; // r-w
int8 mgt-compute-rate-ratio; // r-w
int8 initial-compute-mean-link-delay; // r-w
int8 current-compute-mean-link-delay; // r
bool use-mgt-compute-mean-link-delay; // r-w
int8 mgt-compute-mean-link-delay; // r-w
uint8 allowed-lost-responses; // r-w
uint8 allowed-faults; // r-w
uint8 gptp-cap-receipt-timeout; // r-w
float64 nup; // r-w
float64 ndown; // r-w
bool one-step-tx-oper; // r
bool one-step-receive; // r
bool one-step-transmit; // r
int8 initial-one-step-tx-oper; // r-w
int8 current-one-step-tx-oper; // r
bool use-mgt-one-step-tx-oper; // r-w
int8 mgt-one-step-tx-oper; // r-w
bool sync-locked; // r
uint64 *pdelay-truncated-timestamps; // r
bool gptp-capable-state-machines-enabled; // rw
int32 nrr-pdelay; // r
int32 nrr-sync; // r
enum nrr-comp-method; // rw
description-port-ds
string profile-identifier; // r
port-statistics-ds
counter32 rx-sync-count; // r
counter32 rx-one-step-sync-count; // r
counter32 rx-follow-up-count; // r
counter32 rx-pdelay-req-count; // r
counter32 rx-pdelay-resp-count; // r
counter32 rx-pdelay-resp-follow-up-count; // r
counter32 rx-announce-count; // r
counter32 rx-packet-discard-count; // r
counter32 sync-receipt-timeout-count; // r
counter32 announce-receipt-timeout-count; // r
counter32 pdelay-allowed-lost-exceeded-count; // r
counter32 tx-sync-count; // r
counter32 tx-one-step-sync-count; // r
counter32 tx-follow-up-count; // r
counter32 tx-pdelay-req-count; // r
counter32 tx-pdelay-resp-count; // r
counter32 tx-pdelay-resp-follow-up-count; // r
counter32 tx-announce-count; // r
uint32 rx-sync-count-time-receiver-p; // r
acceptable-master-port-ds
bool enable; // r-w
external-port-config-port-ds
port-state desired-state; // r-w
asymmetry-measurement-mode-ds
bool enabled; // r-w
common-services-port-ds
uint16 cmlds-link-port-port-number // r
1588 objects
802.1ASdn objects
802.1ASdm objects

NOTE 3—This figure differs from the 2020 edition of this standard with Corrigendum 1 applied and as amended by IEEE Std 802.1ASdn in that managed objects needed for the hot standby and Drift_Tracking TLV features are added.

Figure 17-3—PTP Port detail

Replace Figure with the following:


common-services
cmlds
hot-standby-service
* hot-standby-system-index
default-ds
clock-identity clock-identity; // r
uint16 number-link-ports; // r
hot-standby-system
uint8 hot-standby-system-index; // r-w
hot-standby-system-description-ds
string user-description; // r-w
ports
* port-index
port
hot-standby-system-ds
uint32 primary-ptp-instance-index; // rw
uint32 secondary-ptp-instance-index; // rw
bool hot-standby-system-enable; // rw
state-type hot-standby-syste-state; // r
bool hot-standby-system-split-functionality; // rw
scaled-ns primary-secondary-offset; // ro
scaled-ns primary-secondary-offset-thresh; // rw
int hot-standby-system-log-sync-time-thresh; // rw
link-port-ds
struct port-identity; // r
time-interval mean-link-delay; // r
int32 scaled-neighbor-rate-ratio; // r
uint8 version-number; // r-w
uint8 minor-version-number; // r-w
time-interval delay-asymmetry; // r-w
bool cmlds-link-port-enabled // r
bool is-measuring-delay; // r
bool as-capable-across-domains; // r
time-interval mean-link-delay-thresh; // r-w
int8 initial-log-pdelay-req-interval; // r-w
int8 current-log-pdelay-req-interval; // r
bool use-mgt-log-pdelay-req-interval; // r-w
int8 mgt-log-pdelay-req-interval; // r-w
int8 initial-compute-rate-ratio; // r-w
int8 current-compute-rate-ratio; // r
bool use-mgt-compute-rate-ratio; // r-w
int8 mgt-compute-rate-ratio; // r-w
int8 initial-compute-mean-link-delay; // r-w
int8 current-compute-mean-link-delay; // r
bool use-mgt-compute-mean-link-delay; // r-w
int8 mgt-compute-mean-link-delay; // r-w
uint8 allowed-lost-responses; // r-w
uint8 allowed-faults; // r-w
uint64 *pdelay-truncated-timestamps; // r
port-statistics-ds
counter32 rx-pdelay-req-count; // r
counter32 rx-pdelay-resp-count; // r
counter32 rx-pdelay-resp-follow-up-count; // r
counter32 rx-packet-discard-count; // r
counter32 pdelay-allowed-lost-exceeded-count; // r
counter32 tx-pdelay-req-count; // r
counter32 tx-pdelay-resp-count; // r
counter32 tx-pdelay-resp-follow-up-count; // r
asymmetry-measurement-mode-ds
bool enabled; // r-w
1588 objects
802.1ASdn objects
802.1ASdm objects

NOTE 4—This figure differs from the 2020 edition of this standard with Corrigendum 1 applied and as amended by IEEE Std 802.1ASdn in that managed objects needed for the hot standby and Drift_Tracking TLV features are added.

Figure 17-4—Common services detail

17.3 Structure of YANG models

The YANG model specified by this standard uses the YANG modules summarized in Table 17-1.[1]

In the YANG module definitions, if any discrepancy between the “description” text and the corresponding definition in any other part of this standard occur, the definitions outside this clause (Clause 17) take precedence.

Change Table 17-1 as follows:.

Table 17-1—Summary of the YANG modules

Module	Managed functionality	YANG specification notes
ietf-yang-types	Type definitions	IETF RFC 6991 - Common YANG Data Types.
ieee1588-ptp	Clause 14	IEEE Std 1588e - MIB and YANG Data Models. IEEE Std 802.1ASdn imports this YANG module as its foundational tree, including a subset of members from Clause 14.
ieee802-dot1as-ptp	Clause 14	IEEE Std 802.1ASdn - YANG Data Model. The YANG module of this clause uses YANG augments to add members from Clause 14 that are unique to IEEE Std 802.1AS.
ieee802-dot1as-hs	Clause 14	IEEE Std 802.1ASdm - YANG Data Model. The YANG module of this clause uses YANG augments to add members from Clause 14 that are unique to IEEE Std 802.1ASdm.

17.5 YANG schema tree definitions

Insert 17.5.2 and renumber subsequent subclauses as appropriate:

17.5.2 Tree diagram for ieee802-dot1as-ptp.yang

```
module: ieee802-dot1as-hs

  augment /ptp:ptp/ptp:instances/ptp:instance/ptp:parent-ds:
    +--ro gm-present?   boolean
  augment /ptp:ptp/ptp:instances/ptp:instance:
    +--rw ptp-instance-sync-ds
    |  +--ro is-synced?                           boolean
    |  +--rw offset-from-time-transmitter-max?    ptp:time-interval
    |  +--rw rx_sync-count-time-receiver-p-thresh?   uint32
    |  +--rw offset-max-exceeded-count-thresh?    uint32
    |  +--rw offset-max-met-count-thresh?         uint32
    +--rw drift-tracking-ds
       +--rw drift-tracking-tlv-support?   boolean
  augment /ptp:ptp/ptp:instances/ptp:instance/ptp:ports/ptp:port/ptp:port-ds:
    +--rw gptp-capable-state-machines-enabled?   boolean
```

[1]An amendment’s designation is often used to refer to functionality in an IEEE standard after the amendment has been incorporated in a revision of the standard, even if the functionality has been revised. The amendment that added each YANG module is identified to help locate the relevant provisions of this standard.

```
    +--ro nrr-pdelay?                            int32
    +--ro nrr-sync?                              int32
    +--rw nrr-comp-method?                       nrr-comp-method-type
        augment    /ptp:ptp/ptp:instances/ptp:instance/ptp:ports/ptp:port/dot1as-ptp:port-
statistics-ds:
    +--ro rx-sync-count-time-receiver-p?   uint32
  augment /ptp:ptp/ptp:common-services:
    +--rw hot-standby-service {hot-standby}?
       +--rw hot-standby-system* [hot-standby-system-index]
          +--rw hot-standby-system-index             uint8
          +--rw hot-standby-system-ds
          |  +--rw primary-ptp-instance-index?                uint32
          |  +--rw secondary-ptp-instance-index?              uint32
          |  +--rw hot-standby-system-enable?                 boolean
          |  +--ro hot-standby-system-state?                 hot-standby-system-state-type
          |  +--rw hot-standby-system-split-functionality?    boolean
          |  +--ro primary-secondary-offset?                  dot1as-ptp:scaled-ns
          |  +--rw primary-secondary-offset-thresh?           dot1as-ptp:scaled-ns
          |  +--rw hot-standby-system-log-sync-time-thresh?   int8
          +--rw hot-standby-system-description-ds
             +--rw user-description?   string
```

17.6 YANG modules[1] [2]

Insert 17.6.2 and renumber subsequent subclauses as appropriate:

17.6.2 Module ieee802-dot1as-hs.yang

```
module ieee802-dot1as-hs {
  yang-version 1.1;
  namespace "urn:ieee:std:802.1AS:yang:ieee802-dot1as-hs";
  prefix dot1as-hs;

  import ieee1588-ptp {
    prefix ptp;
  }
  import ieee802-dot1as-ptp {
    prefix dot1as-ptp;
  }

  organization
    "IEEE 802.1 Working Group";
  contact
    "WG-URL: http://ieee802.org/1/
     WG-EMail: stds-802-1-l@ieee.org

     Contact: IEEE 802.1 Working Group Chair
              Postal: C/O IEEE 802.1 Working Group
              IEEE Standards Association
              445 Hoes Lane
              Piscataway, NJ 08854
              USA
```

[1]Copyright release for YANG modules: Users of this standard may freely reproduce the YANG modules contained in this subclause so that they can be used for their intended purpose.
[2]An ASCII version of the YANG modules are attached to the PDF version of this standard, and can be obtained by Web browser from the IEEE 802.1 Website at https://1.ieee802.org/yang-modules/.

```
       E-mail: stds-802-1-chairs@ieee.org";
    description
      "Management objects that control hot standby systems as specified in
       IEEE Std 802.1ASdm-d1.3.

       References in this YANG module to IEEE Std 802.1ASdm are to IEEE Std
802.1AS-2020 as modified by
       IEEE Std 802.1AS-2020/Cor-1-2021, and amended by IEEE Std 802.1ASdr,
IEEE Std 802.1ASdn, and IEEE Std 802.1ASdm.

       Copyright (C) IEEE (2023).
       This version of this YANG module is part of IEEE Std 802.1ASdm;
       see the standard itself for full legal notices.";

    revision 2023-10-11 {
      description
        "Published as part of IEEE Std 802.1ASdm-XXXX.
         Initial version.";
      reference
        "IEEE Std 802.1ASdm - YANG Data Model";
    }

    // The year (XXXX) will be replaced during publication.
    // This is the 1st balloted draft D0.1
    // of the YANG module for amendment IEEE P802.1ASdm.

    feature hot-standby {
      description
         "This feature indicates that the device supports the hot-standby
functionality.";
    }

    typedef hot-standby-system-state-type {
      type enumeration {
        enum init {
          value 0;
          description
              "Initialization after the HotStandbySystem powers on and is
enabled. In this state, the system is
             waiting for both PTP Instances to synchronize.";
        }
        enum redundant {
          value 1;
          description
                 "Both PTP Instances are synchronized according to the
requirements of the
              respective application or profile standard (see 3.24). Time
synchronization is redundant.";
        }
        enum not-redundant {
          value 2;
          description
             "One PTP Instance is synchronized, and the other PTP Instance
is faulted (not
              synchronized). Time synchronization continues to meet the
requirements of the respective
                    application or profile standard (see 3.24). Time
synchronization is not redundant.";
      }
      enum out-of-sync {
        value 3;
        description
          "The HotStandbySystem is adjusting phase/frequency of its local
time using the
               data stored while the system was in the REDUNDANT or
NOT_REDUNDANT state, but the local
              time will eventually drift relative to other time-aware
systems. During OUT_OF_SYNC state, time
               synchronization might not meet the requirements of the
respective application or profile standard
           (see 3.24 of IEEE Std 802.1ASdm).";
      }
    }
  }

  typedef nrr-comp-method-type {
    type enumeration {
      enum sync {
        value 0;
        description
              "If the value is Sync and driftTrackingTlvSupport (see
10.2.4.27) is TRUE, neighborRateRatio is
           populated with the value of nrrSync whenever a new value of
nrrSync is computed.";
      }
      enum pdelay {
        value 1;
        description
              "If the value is Pdelay or if driftTrackingTlvSupport
(10.2.4.27) is FALSE, neighborRateRatio is
           populated with the value of nrrPdelay whenever a new value of
nrrPdelay is computed.";
      }
    }
  }

  augment "/ptp:ptp/ptp:instances/ptp:instance/ptp:parent-ds" {
    description
      "Augment IEEE Std 1588 parentDS.";
    leaf gm-present {
      type boolean;
      config false;
      description
        "The value of gmPresent is set equal to the value of the global
variable gmPresent (see 10.2.4.13).
              This parameter indicates to the ClockTarget whether a
Grandmaster PTP Instance is present.";
      reference
```

```
          "14.4.8 of IEEE Std 802.1ASdm";
      }
    }

    augment "/ptp:ptp/ptp:instances/ptp:instance" {
      description
        "Augment IEEE Std 1588 instanceList.";
      container ptp-instance-sync-ds {
        description
            "The ptpInstanceSyncDS describes the synchronization status of
the PTP Instance.";
        reference
          "14.8 of IEEE Std 802.1ASdm";
        leaf is-synced {
          type boolean;
          config false;
          description
            "The value of the global variable isSynced (see 18.4.1.1)";
          reference
            "14.8.2 of IEEE Std 802.1ASdm";
        }
        leaf offset-from-time-transmitter-max {
          type ptp:time-interval;
          config true;
          description
            "The value is the threshold for offsetFromTimeTransmitter (see
18.4.1.2), below which the PTP Instance is
             considered to be synchronized.";
          reference
            "14.8.3 of IEEE Std 802.1ASdm";
        }
        leaf rx_sync-count-time-receiver-p-thresh {
          type uint32;
          config true;
          description
             "The value of rxSyncCountTimeReceiverPThresh is the threshold
for rxSyncCountTimeReceiverP (see 18.4.1.4),
                     above which the PTP Instance is considered to be
synchronized.";
          reference
            "14.8.4 of IEEE Std 802.1ASdm";
        }
        leaf offset-max-exceeded-count-thresh {
          type uint32;
          config true;
          description
             "The value of offsetMaxExceededCountThresh (see 18.4.1.7) is
the threshold for the number of consecutive
             exceedances of offsetFromTimeTransmitterMax (see 18.4.1.3) by
offsetFromTimeTransmitter (see 18.4.1.2),
             at which isSynced (see 18.4.1.1) is no longer TRUE.";
          reference
            "14.8.5 of IEEE Std 802.1ASdm";
        }
        leaf offset-max-met-count-thresh {
          type uint32;
          config true;
          description
                "The value of offsetMaxMetCountThresh (18.4.1.9) is the
threshold for the number of consecutive
             occurrences of offsetFromTimeTransmitter (see 18.4.1.2) being
within offsetFromTimeTransmitterMax
             (see 18.4.1.3), at which isSynced (see 18.4.1.1) is changed to
TRUE if it currently is FALSE.";
          reference
            "14.8.6 of IEEE Std 802.1ASdm";
        }
      }
      container drift-tracking-ds {
        description
           "The driftTrackingDS contains a managed object that is used to
enable or disable the optional Drift_Tracking
           TLV.";
        reference
          "14.9 of IEEE Std 802.1ASdm";
        leaf drift-tracking-tlv-support {
          type boolean;
          config true;
          description
              "The value of driftTrackingTlvSupport indicates whether the
Drift_Tracking TLV is enabled or disabled.";
          reference
            "14.9.2 of IEEE Std 802.1ASdm";
        }
      }
    }

        augment   "/ptp:ptp/ptp:instances/ptp:instance/ptp:ports/ptp:port/
ptp:port-ds" {
      description
        "Augment IEEE Std 1588 commonServices.";
      leaf gptp-capable-state-machines-enabled {
        type boolean;
        config true;
        description
                "A Boolean that is used to enable or disable the
GptpCapableTransmit, GptpCapableReceive, and
           GptpCapableIntervalSetting state machines.";
        reference
          "14.8.55 of IEEE Std 802.1ASdm";
      }
      leaf nrr-pdelay {
        type int32;
        config false;
        description
           "The value is an estimate of the ratio of the frequency of the
LocalClock entity of the time-aware system
```

```
          at the other end of the link attached to this Link Port, to the
frequency of the LocalClock entity of this
          time-aware system (see 10.2.5.7). nrrPdelay is expressed as the
fractional frequency offset stored in the
          global variable nrrPdelay (see 11.2.13.13) multiplied by 2^41,
i.e., the quantity (nrrPdelay - 1.0)(2^41).";
      reference
        "14.8.56 of IEEE Std 802.1ASdm";
    }
    leaf nrr-sync {
      type int32;
      config false;
      description
        "The value is an estimate of the ratio of the frequency of the
LocalClock entity of the time-aware system at
          the other end of the link attached to this Link Port, to the
frequency of the LocalClock entity of this
          time-aware system (see 10.2.5.7). nrrSync is expressed as the
fractional frequency offset stored in the global
          variable nrrSync (see 11.2.13.13) multiplied by 2^41, i.e., the
quantity (nrrSync - 1.0)(2^41).";
      reference
        "14.8.57 of IEEE Std 802.1ASdm";
    }
    leaf nrr-comp-method {
      type nrr-comp-method-type;
      config true;
      description
          "An Enumeration that takes on the values sync and pdelay to
indicate the source of the value of
          neighborRateRatio (see 10.2.5.7)";
      reference
        "14.8.58 of IEEE Std 802.1ASdm";
    }
  }

      augment   "/ptp:ptp/ptp:instances/ptp:instance/ptp:ports/ptp:port/
dot1as-ptp:port-statistics-ds" {
    description
      "Augment IEEE Std 802.1AS PortStatisticsDS.";
    leaf rx-sync-count-time-receiver-p {
      type uint32;
      config false;
      description
            "This counter increments whenever time synchronization
information is received on a PTP Port
          when its port state is TimeReceiverPort.";
      reference
        "14.10.20 of IEEE Std 802.1ASdm";
    }
  }

  augment "/ptp:ptp/ptp:common-services" {
    description
```

```
        "Augment IEEE Std 1588 commonServices.

         14.16.9 of IEEE Std 802.1ASdm specifies nrrPdelay
         (nrr-pdelay), which is provided in YANG as the
         semantically equivalent node in ieee1588-ptp named
         scaled-neighbor-rate-ratio (in link-port-ds).";
      container hot-standby-service {
        if-feature "hot-standby";
        description
                       "The   hotStandbyService   structure   contains   the
hotStandbySystemList, which is a list of instances of the Hot
           Standby Service.";
        reference
          "14.19 of IEEE Std 802.1ASdm";
        list hot-standby-system {
          description
            "List of instances of the Hot Standby Service";
          key "hot-standby-system-index";
          leaf hot-standby-system-index {
            type uint8;
            config true;
          }
          container hot-standby-system-ds {
            description
                 "The hotStandbySystemDS describes the attributes of the
respective instance of the Hot Standby Service.";
            reference
              "14.19 of IEEE Std 802.1ASdm";
            leaf primary-ptp-instance-index {
              type uint32;
              config true;
              description
                  "The value of primaryPtpInstanceIndex is the index (see
14.1.1) of the primary PTP Instance associated with
                 this hotStandbySystem instance.";
              reference
                "14.19.2 of IEEE Std 802.1ASdm";
            }
            leaf secondary-ptp-instance-index {
              type uint32;
              config true;
              description
                 "The value of secondaryPtpInstanceIndex is the index (see
14.1.1) of the secondaryPTP Instance associated
                 with this hotStandbySystem instance.";
              reference
                "14.19.3 of IEEE Std 802.1ASdm";
            }
            leaf hot-standby-system-enable {
              type boolean;
              config true;
              description
                 "The value is the hotStandbySystemEnable attribute of the
HotStandbySystem entity (see 18.5.1.2).";
```

```
              reference
                "14.19.4 of IEEE Std 802.1ASdm";
            }
            leaf hot-standby-system-state {
              type hot-standby-system-state-type;
              config false;
              description
                  "The value of hotStandbySystemState is the state of the
hotStandbySystem, i.e., the value of the global
                 variable hotStandbySystemState (see 18.5.1.1).";
              reference
                "14.19.5 of IEEE Std 802.1ASdm";
            }
            leaf hot-standby-system-split-functionality {
              type boolean;
              config true;
              description
               "If the value is TRUE, the optional split functionality (see
18.5.3.4) is used. If the value is FALSE, the
                 optional split functionality is not used.";
              reference
                "14.19.6 of IEEE Std 802.1ASdm";
            }
            leaf primary-secondary-offset {
              type dot1as-ptp:scaled-ns;
              config false;
              description
                      "The absolute value of the difference between the
clockTimeTransmitterTimes (see 10.2.4.3) of the primary and
                 secondary PTP Instances.";
              reference
                "14.19.7 of IEEE Std 802.1ASdm";
            }
            leaf primary-secondary-offset-thresh {
              type dot1as-ptp:scaled-ns;
              config true;
              description
                                                       "The   threshold   for
hotStandbySystemDS.primarySecondaryOffset (see 14.19.7), above which the
                    hotStandbySytemState transitions from REDUNDANT to
NOT_REDUNDANT, or does not transition from
                  NOT_REDUNDANT or OUT_OF_SYNC to REDUNDANT even if other
conditions for these transitions
                 are satisfied.";
              reference
                "14.19.8 of IEEE Std 802.1ASdm";
            }
            leaf hot-standby-system-log-sync-time-thresh {
              type int8;
              config true;
              description
                   "The value of hotStandbySystemLogSyncTimeThresh is the
logarithm to base 2 of the time interval, in
```

```
              seconds, after which the hotStandbySystem transitions from
the OUT_OF_SYNC state to either the NOT-
              REDUNDANT or REDUNDANT state, or from the NOT_REDUNDANT to
the REDUNDANT state, if all
             other conditions for the respective transition are met. The
value -128 means that the transition time
               is zero, i.e., the transition occurs immediately.";
            reference
              "14.19.9 of IEEE Std 802.1ASdm";
          }
        }
        container hot-standby-system-description-ds {
          description
               "The hotStandbySystemDescriptionDS contains descriptive
information for the respective instance of the Hot Standby Service.";
          reference
            "14.20 of IEEE Std 802.1ASdm";
          leaf user-description {
            type string {
              length "0..128";
            }
            config true;
            description
              "Configurable description of the hot standby system.";
            reference
              "14.20.3 of IEEE Std 802.1ASdm";
          }
        }
      }
    }
  }
}
```

Insert the following new Clause 18:

18. Hot Standby

18.1 General

Hot standby includes:

— A function that transforms the synchronized times of two generalized Precision Time Protocol (gPTP) domains into one synchronized time for use by applications;
— A function that directs the synchronized time of one gPTP domain into a different gPTP domain; and
— Mechanisms that determine whether a gPTP domain has sufficient quality to be used for hot standby.

For time synchronization using hot standby, two distinct domains are statically configured in the network and the Best TimeTransmitter Clock Algorithm (BTCA) is disabled for these domains. When hot standby is used for redundancy, a time-aware system operates two PTP Instances simultaneously, each in its own domain. One of the domains is considered the primary domain, and the other domain is considered the secondary domain. A time-aware system that has a primary domain and a secondary domain available uses the primary domain via the associated PTP instance. If the primary domain becomes unavailable (e.g., due to temporary or permanent failure of a physical link) and the secondary domain is still available to the time-aware system, the time aware system begins using the secondary domain.

Examples of hot standby are shown in 7.2.4 of this standard.

18.2 Overview

Figure 18-1 provides a model of hot standby for time synchronization.

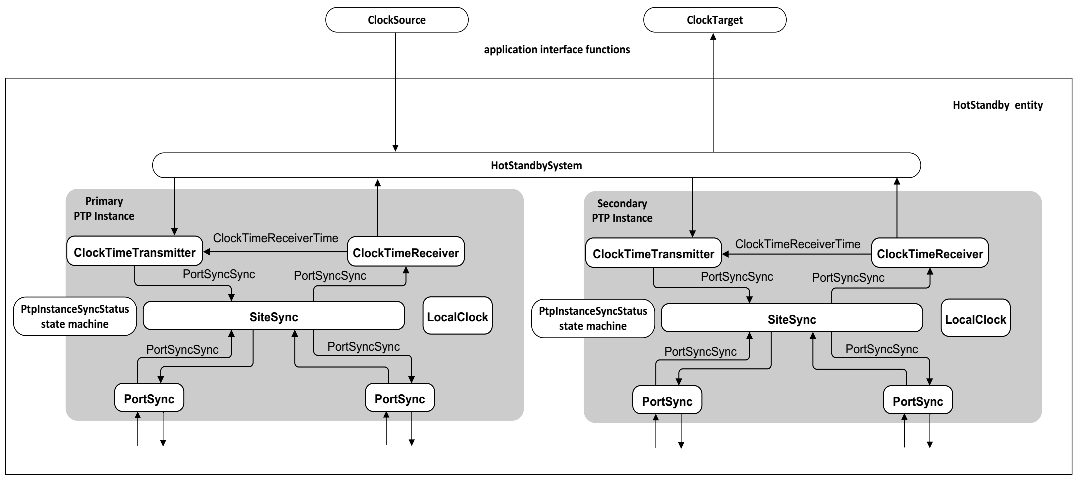


Figure 18-1—Model for hot standby entity in a time-aware system, and its interfaces to higher-layer applications

Each PTP Instance has a corresponding PtpInstanceSyncStatus state machine (see 18.4). The PtpInstanceSyncStatus state machine monitors its PTP Instance to determine whether it is synchronized according to the requirements of the respective application or profile standard.

There is one HotStandbySystem entity, which interacts with the primary and secondary PTP Instances via the HotStandbySystem state machine (see 17.6) in order to provide a single value of time to the application (ClockTarget). This single value of time is based on the redundant values of time provided by either the primary PTP Instance, or the secondary PTP Instance, or both. There is one HotStandbySystem for both PTP Instances that use hot standby. The primary and secondary PTP Instances either both use the PTP timescale or both use the ARB timescale.

The ClockTarget entity represents the application that uses the synchronized time. The ClockTarget and its application interfaces are analogous to the ClockTarget specified in Clause 9. In the case of PTP Relay Instances, the ClockTarget is present if it is needed by the application.

The ClockSource entity is used to transfer the source of time when at least one of the PTP Instances is grandmaster-capable. The ClockSource and its application interfaces are analogous to the ClockSource specified in Clause 9. The ClockSource might be present if the PTP Instance is capable of becoming a Grandmaster PTP Instance.

There is one LocalClock entity for each PTP Instance. Therefore, Figure 18-1 shows two LocalClock entities.

NOTE—The LocalClock entities can be traceable to the same oscillator or to different oscillators.

18.3 PTP Instance configuration

PTP Instances that support hot standby are configured as follows:

a) The two PTP Instances are enabled using domainNumbers as specified by the respective application or profile standard;
b) For both PTP Instances, externalPortConfigurationEnabled is set to TRUE;
c) If a PTP Instance (primary or secondary) is grandmaster, externalPortConfigurationPortDS.desiredState is configured to TimeTransmitterPort or PassivePort for all PTP Ports (portNumber 1 and higher); otherwise, externalPortConfiguration.desiredState is configured to TimeReceiverPort for one PTP Port, and is configured to TimeTransmitterPort or PassivePort for other PTP Ports; and
d) Each PTP Instance shall have a corresponding PtpInstanceSyncStatus state machine (see 18.4).

18.4 PtpInstanceSyncStatus state machine

This state machine operates within the PTP Instance, to determine whether it is synchronized according to the requirements of the respective application or profile standard (see 3.24).

18.4.1 PtpInstanceSyncStatus state machine global variables

The following variables are used in the state diagram in Figure 18-2 (in 18.4.4):

18.4.1.1 isSynced: The synchronization status of the PTP Instance. The variable is a Boolean, and is TRUE if synchronization is sufficient according to the operation of the PtpInstanceSyncStatus state machine and

FALSE otherwise. An application or profile standard can set the offsetFromTimeTransmitterMax (see 18.4.1.3), rxSyncCountTimeReceiverPThresh (see 18.4.1.5), offsetMaxExceededCountThresh (see 18.4.1.7), and offsetMaxMetCountThresh (see 18.4.1.9) according to its requirements.

18.4.1.2 offsetFromTimeTransmitter: The value of the managed object currentDS.offsetFromTimeTransmitter (see 14.3.3).

18.4.1.3 offsetFromTimeTransmitterMax: The value of the managed object hotstandbyDS.offsetFromTimeTransmitterMax (see 14.8.3)

18.4.1.4 rxSyncCountTimeReceiverP: The value of the managed object portStatisticsDS.rxSyncCountTimeReceiverP (see 14.10.20).

18.4.1.5 rxSyncCountTimeReceiverPThresh: The value of the managed object hotstandbyDS.rxSyncCountTimeReceiverPThresh (see 14.8.4)

18.4.1.6 offsetMaxExceededCount: The current number of consecutive exceedances of offsetFromTimeTransmitterMax (see 18.4.1.3) by offsetFromTimeTransmitter (see 18.4.1.2). The data type of offsetMaxExceededCount is UInteger32.

18.4.1.7 offsetMaxExceededCountThresh: The threshold (see 14.8.5) for the number of consecutive exceedances of offsetFromTimeTransmitterMax (see 18.4.1.3) by offsetFromTimeTransmitter (see 18.4.1.2), at which isSynced for the PTP Instance is no longer TRUE (see 18.4.3.3).

18.4.1.8 offsetMaxMetCount: The current number of consecutive occurrences of offsetFromTimeTransmitter (see 18.4.1.2) being within offsetFromTimeTransmitterMax (see 18.4.1.3). The data type of offsetMaxMetCount is UInteger32.

18.4.1.9 offsetMaxMetCountThresh: The threshold (see 14.8.6) for the number of consecutive occurrences of offsetFromTimeTransmitter (see 18.4.1.2) being within offsetFromTimeTransmitterMax (see 18.4.1.3), at which isSynced for the PTP Instance is changed from FALSE to TRUE (see 18.4.3.3).

18.4.2 PtpInstanceSyncStatus state machine local variables

18.4.2.1 lastRxSyncCountTimeReceiverP: Holds the last value of rxSyncCountTimeReceiverP (see 18.4.1.4) that the state machine read for the TimeReceiverPort, as part of monitoring for a sync event message timeout.

18.4.3 PtpInstanceSyncStatus state machine functions

18.4.3.1 isGptpCapable(): This function returns a boolean value that is TRUE when ptpPortEnabled (see 10.2.5.13) and asCapable (see 10.2.5.1) are TRUE for at least one PTP Port (i.e., port for which portNumber is not zero).

```
isGptpCapable()
{
        if (!instanceEnable)
                return (FALSE);
        for (int i = 1; i <= numberPorts; i++)
        // see 8.6.2.8 for numberPorts
        {
                if (portIsCapable (i))
                        return (TRUE);
        }

        return (FALSE)
}
```

18.4.3.2 isGm(): This function determines if the PTP Instance is a Grandmaster PTP Instance (TRUE) or not (FALSE) by searching the selectedState array (see 10.2.4.20) for absence or presence of at least one value equal to TimeReceiverPort (see Table 10-2).

```
isGm()
{
        for (int i = 1; i <= numberPorts; i++)
        // see 8.6.2.8 for numberPorts
        {
          if (selectedState[i] == TimeReceiverPort)
            return (FALSE);
        }
        return (TRUE);
}
```

18.4.3.3 portIsCapable(j): This function returns a boolean value that is TRUE if PTP Port j is enabled and asCapable is TRUE, and FALSE otherwise.

```
portIsCapable (j)
{
        if (ptpPortEnabled && asCapable)
                        return (TRUE);
                else
                        return (FALSE);
}
```

18.4.4 State diagram

The PtpInstanceSyncStatus state machine shall implement the function specified by the state diagram in Figure 18-2, the variables specified in 18.4.1 and 18.4.2, and the functions specified in 18.4.3.

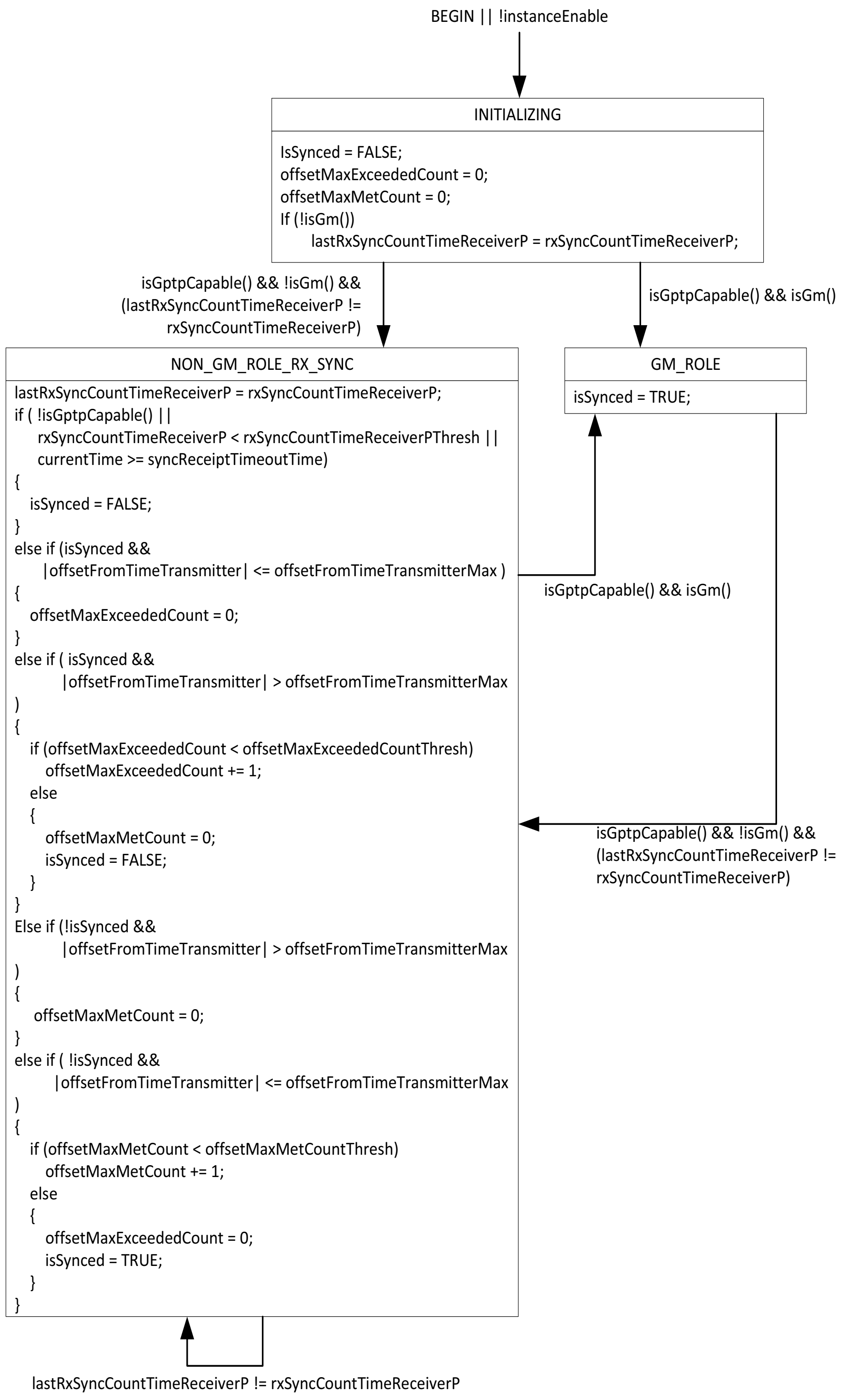

BEGIN || !instanceEnable
INITIALIZING
IsSynced = FALSE;
offsetMaxExceededCount = 0;
offsetMaxMetCount = 0;
If (!isGm())
lastRxSyncCountTimeReceiverP = rxSyncCountTimeReceiverP;
isGptpCapable() && !isGm() && (lastRxSyncCountTimeReceiverP != rxSyncCountTimeReceiverP)
isGptpCapable() && isGm()
NON_GM_ROLE_RX_SYNC
lastRxSyncCountTimeReceiverP = rxSyncCountTimeReceiverP;
if (!isGptpCapable() || rxSyncCountTimeReceiverP < rxSyncCountTimeReceiverPThresh || currentTime >= syncReceiptTimeoutTime)
{ isSynced = FALSE; }
else if (isSynced && |offsetFromTimeTransmitter| <= offsetFromTimeTransmitterMax)
{ offsetMaxExceededCount = 0; }
else if (isSynced && |offsetFromTimeTransmitter| > offsetFromTimeTransmitterMax)
{ if (offsetMaxExceededCount < offsetMaxExceededCountThresh) offsetMaxExceededCount += 1; else { offsetMaxMetCount = 0; isSynced = FALSE; } }
Else if (!isSynced && |offsetFromTimeTransmitter| > offsetFromTimeTransmitterMax)
{ offsetMaxMetCount = 0; }
else if (!isSynced && |offsetFromTimeTransmitter| <= offsetFromTimeTransmitterMax)
{ if (offsetMaxMetCount < offsetMaxMetCountThresh) offsetMaxMetCount += 1; else { offsetMaxExceededCount = 0; isSynced = TRUE; } }
GM_ROLE
isSynced = TRUE;
isGptpCapable() && isGm()
isGptpCapable() && !isGm() && (lastRxSyncCountTimeReceiverP != rxSyncCountTimeReceiverP)
lastRxSyncCountTimeReceiverP != rxSyncCountTimeReceiverP

Figure 18-2—PtpInstanceSyncStatus State Machine

18.5 HotStandbySystem state machine

This state machine interacts with the primary and secondary PTP Instances in order to provide a single redundant time to the application.

18.5.1 HotStandbySystem state machine global variables

18.5.1.1 hotStandbySystemState: State of the HotStandbySystem. The variable is an enumeration that takes one of the following values:

a) INIT: Initialization after the HotStandbySystem powers on and is enabled. In this state, the system is waiting for both PTP Instances to synchronize.
b) REDUNDANT: Both PTP Instances are synchronized according to the requirements of the respective application or profile standard (see 3.24). Time synchronization is redundant.

NOTE—A PTP Instance is implicitly considered to be synchronized if it is a Grandmaster PTP Instance (see 18.4.3).

c) NOT_REDUNDANT: One PTP Instance is synchronized, and the other PTP Instance is faulted (not synchronized). Time synchronization continues to meet the requirements of the respective application or profile standard (see 3.24). Time synchronization is not redundant.
d) OUT_OF_SYNC: The HotStandbySystem may adjust phase/frequency of its local time using the data stored while the system was in the REDUNDANT or NOT_REDUNDANT state, but the local time will eventually drift relative to other time-aware systems. During OUT_OF_SYNC state, time synchronization might not meet the requirements of the respective application or profile standard (see 3.24).

18.5.1.2 hotStandbySystemEnable: A Boolean variable used to enable/disable the HotStandbySystem.

18.5.1.3 hotStandbySystemLogSyncTimeThresh: The value of the managed object hotStandbySystemLogSyncTimeThreshold (see 14.19.6) of the hotStandbySystemDS. The data type for hotStandbySystemLogSyncTimeThreshold is Integer8.

18.5.1.4 primarySecondaryOffset: The value of the managed object hotStandbySystemDS.primarySecondaryOffset (see 14.19.7), which is the absolute value of the difference between the clockTimeReceiverTime variables (see 10.2.4.3) of the primary and secondary PTP Instances.

18.5.2 HotStandbySystem state machine local variables

18.5.2.1 primary: Reference to the data sets of the PTP Instance configured to use the primary domainNumber as specified by the respective application or profile standard (see 3.24). This references an element of the instanceList specified in 14.1.

18.5.2.2 secondary: Reference to the data sets of the PTP Instance configured to use the secondary domainNumber as specified by the respective application or profile standard (see 3.24). This references an element of the instanceList specified in 14.1.

18.5.2.3 TEMP: A temporaray variable used to reduce clutter in the state diagram (see Figure 18-3). The data type for TEMP is Integer16.

18.5.2.4 transitionTime: The value of currentTime (see 10.2.4.12) after which the hotStandbySystem transitions from the OUT_OF_SYNC state to either the NOT_REDUNDANT or REDUNDANT state, or from the NOT_REDUNDANT to the REDUNDANT state, if all other conditions for the respective transition is met. The data type for transitionTime is UScaledNs.

18.5.2.5 primarySecondaryOffsetThresh: The value of the managed object hotStandbySystemDS.primarySecondaryOffsetThresh, which is the threshold for primarySecondaryOffset (see 18.5.1.4), above which the hotStandbySytemState transitions from REDUNDANT to NOT_REDUNDANT, or stays in NOT_REDUNDANT or OUT_OF_SYNC even if other conditions for transitioning to REDUNDANT are satisfied.

18.5.3 HotStandbySystem requirements

18.5.3.1 Primary grandmaster

When the primary PTP Instance is a Grandmaster PTP Instance (i.e., no PTP Port in TimeReceiverPort state), the HotStandbySystem shall transfer phase, frequency, clockSourceTimeBaseIndicator (see 10.2.4.8), clockSourceLastGmPhaseChange (see 10.2.4.10), and clockSourceLastGmFreqChange (see 10.2.4.11) from the ClockSource to the ClockTimeTransmitter of the primary PTP Instance (see Figure 18-1). If no external source of time is implemented, the ClockSource is equivalent to the LocalClock of this PTP Instance.

When the primary PTP Instance is grandmaster, there is no requirement for the HotStandbySystem to receive time from the ClockTimeReceiver of the secondary PTP Instance.

18.5.3.2 Secondary grandmaster

18.5.3.2.1 HotStandbySystem in REDUNDANT state

When the secondary PTP Instance is a Grandmaster PTP Instance (i.e., no PTP Port in TimeReceiverPort state), and the hotStandbySystemState is REDUNDANT, the HotStandbySystem shall transfer phase, frequency, clockSourceTimeBaseIndicator (see 10.2.4.8), clockSourceLastGmPhaseChange (see 10.2.4.10), and clockSourceLastGmFreqChange (see 10.2.4.11) from the ClockTimeReceiver of the primary PTP Instance to the ClockTimeTransmitter of the secondary PTP Instance (see Figure 18-1). By using phase from the primary PTP Instance, the secondary grandmaster can maintain continuity in the event of a fault in the primary domain.

18.5.3.2.2 HotStandbySystem in NOT_REDUNDANT state

When the secondary PTP Instance is a Grandmaster PTP Instance (i.e., no PTP Port in TimeReceiverPort state), and the hotStandbySystemState is NOT_REDUNDANT, the HotStandbySystem shall transfer phase, frequency, clockSourceTimeBaseIndicator (see 10.2.4.8), clockSourceLastGmPhaseChange (see 10.2.4.10), and clockSourceLastGmFreqChange (see 10.2.4.11) from the ClockSource to the ClockTimeTransmitter of the secondary PTP Instance (see Figure 18-1). If no external source of time is implemented, the ClockSource is equivalent to the LocalClock of this PTP Instance.

The secondary PTP Instance is considered synchronized because it is a Grandmaster PTP Instance (see 18.4.4 and Figure 18-2). Since the hotStandbySystemState is NOT_REDUNDANT, this means that either the primary PTP Instance is not synchronized or primarySecondaryOffset exceeds primarySecondaryOffsetThresh. However, in the former case primarySecondaryOffset can exceed primarySecondaryOffsetThresh because in this case the primary and secondary PTP Instances receive timing from different sources. The result is that the HotStandbySystem remains in the NOT_REDUNDANT state, even if the primary PTP Instance subsequently becomes synchronized again by the primary Grandmaster PTP Instance, because primarySecondaryOffset continues to exceed primarySecondaryOffsetThresh. This situation would not occur if it were the primary PTP Instance that was synchronized and the secondary PTP Instance that was not synchronized due to some fault condition. When the fault condition is cleared, the secondary Grandmaster PTP Instance ClockTimeTransmitter would receive timing from the primary PTP Instance ClockTimeReceiver (see 18.5.3.2.1) and, as a result, it would become synchronized again and the hotStandbySystemState would change to REDUNDANT.

The situation described in the previous paragraph is avoided if the optional split functionality (see 18.5.3.4) is used. In this case, timing is transferred directly from the secondary SiteSync entity to the primary SiteSync entity. When this happens, primarySecondaryOffset changes and, since now the primary PTP Instance is synchronized to the secondary PTP Instance, is within primarySecondaryOffsetThresh. The HotStandbySystem state changes to REDUNDANT and the secondary Grandmaster PTP Instance begins receiving timing from the primary PTP Instance. However, even if the optinal split functionality is not used, an application can choose to re-initialize the secondary PTP Instance after the primary PTP Instance becomes synchronized. If this is done, the secondary Grandmaster PTP Instance receives timing from the primary PTP Instance ClockTimeReceiver after initialization and hotStandbySystemState is REDUNDANT.

On initialization, if the primary PTP Instance is synchronized, the secondary PTP Instance shall receive timing from the primary PTP Instance. When the conditions as specified the HotStandbySystem state machine (see Figure 18-3) are satisfied, the hotStandbySystemState machine transitions to the REDUNDANT state.

18.5.3.2.3 Transition of hotStandbySystemState from REDUNDANT to NOT_REDUNDANT

The secondary grandmaster shall conform to the time synchronization requirements of the respective application or profile standard (see 3.24) during a transition of HotStandbySystemState from REDUNDANT to NOT_REDUNDANT due to a fault in its primary PTP Instance (i.e., primary.isSynced == FALSE).

NOTE—The secondary grandmaster is responsible for maintaining continuity (no "jump" in time) when a fault occurs in the primary grandmaster.

NOTE—The primary domain global variables lastGmPhaseChange and lastGmFrequencyChange are propagated throughout the secondary domain, and can be used to compensate for any phase or frequency jump that occurs when the HotStandbySystem state becomes NOT_REDUNDANT.

18.5.3.3 TimeReceiver HotStandbySystem

18.5.3.3.1 General

A HotStandbySystem is said to be a TimeReceiver HotStandbySystem if and only if neither PTP Instance is a Grandmaster PTP Instance.

18.5.3.3.2 HotStandbySystem in REDUNDANT state

When the HotStandbySystem is a timeReceiver (i.e., neither PTP Instance is a grandmaster), and hotStandbySystemState is REDUNDANT, the HotStandbySystem shall transfer phase, frequency, gmTimeBaseIndicator (see 10.2.4.15), lastGmPhaseChange (see 10.2.4.16), and lastGmFrequencyChange (see 10.2.4.17) from the ClockTimeReceiver of the primary PTP Instance to the ClockTarget (application) if a ClockTarget is present.

18.5.3.3.3 HotStandbySystem in NOT_REDUNDANT state

18.5.3.3.3.1 Transfer of synchronized time to the ClockTarget

When the HotStandbySystem is a timeReceiver, and the HotStandbySystem is in the NOT_REDUNDANT state, the HotStandbySystem shall transfer phase, frequency, gmTimeBaseIndicator (see 10.2.4.15), lastGmPhaseChange (see 10.2.4.16), and lastGmFrequencyChange (see 10.2.4.17) from the

ClockTimeReceiver of the synchronized PTP Instance (i.e. isSynced == TRUE) to the ClockTarget if a ClockTarget is present.

18.5.3.4 Split functionality

The HotStandbySystem may provide an optional split functionality. If the managed object hotStandbySystemDS.hotstandbySystemSplitFunctionality is set to TRUE, the split functionaliy is invoked. If the managed object hotStandbySystemDS.hotstandbySystemSplitFunctionality is set to FALSE, the split functionaliy is not invoked. The split functionality shall provide an interworking function (IWF) that transfers time synchronization information from the primary PTP Instance to the secondary PTP Instance when isSynced is FASLE for the secondary PTP Instance and isSynced is TRUE for the primary PTP Instance, or from the secondary PTP Instance to the primary PTP Instance when isSynced is FASLE for the primary PTP Instance and isSynced is TRUE for the secondary PTP Instance. The IWF provides the most recently received PortSyncSync structure at the timeReceiver port of the synchronized PTP Instance (isSynced == TRUE, see 18.4.1.1) to the unsynchronized PTP Instance (isSynced == FALSE) SiteSyncSync entity, in order to generate a PortSyncSync structure that is used by the timeTransmitter ports of the unsynchronized PTP Instance, as follows:

a) The domainNumber of the received PortSyncSync structure is changed by the IWF from the synchronized PTP Instance domainNumber to the unsynchronized PTP Instance domainNumber;
b) The localPortNumber of the received PortSyncSync structure is changed by the IWF to the portNumber of the unsynchronized PTP Instance timeReceiver port; and
c) All other members (i.e., syncReceiptTimeoutTime, followUpCorrectionField, sourcePortIdentity, logMessageInterval, preciseOriginTimestamp, upstreamTxTime, rateRatio, gmTimeBaseIndicator, lastGmPhaseChange and lastGmFreqChange) of the received PortSyncSync structure of the synchronized PTP Instance are provided to the unsynchronized PTP Instance SiteSync entity unchanged.
d) The flags leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, and frequencyTraceable, and the fields currentUtcOffset and timeSource of received Announce messages on the timeReceiver port of the unsynchronized PTP Instance are ignored. The IWF provides the values of these corresponding members of the timePropertiesDS of the synchronized PTP Instance to the unsynchronized PTP Instance. These values are then copied to:
    i) the respective members of the timePropertiesDS of the unsynchronized PTP Instance;
    ii) the unsynchronized PTP Instance global variables leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, and frequencyTraceable, and the fields currentUtcOffset and timeSource, respectively; and
    iii) the unsynchronized PTP Instance global variables annLeap61, annLeap59, annCurrentUtcOffsetValid, annPtpTimescale, annTimeTraceable, and annFrequencyTraceable, and the fields annCurrentUtcOffset and annTimeSource, respectively.
e) The messagePriorityVector (see 10.3.4 and 10.3.5) information of received Announce messages on the timeReceiver port of the unsynchronized PTP Instance is ignored. The IWF provides the values of the corresponding members of the synchronized PTP Instance parentDS, i.e., grandmasterIdentity, grandmasterClockQuality.clockClass, grandmasterClockQuality.clockAccuracy, grandmasterClockQuality.offsetScaledLogVariance, grandmasterPriority1, grandmasterPriority2 , which are copied to:
    i) the corresponding members of the parentDS of the unsynchronized PTP Instance; and
    ii) the gmPriorityVector of the unsynchronized PTP Instance.

    The IWF also provides the value of the stepsRemoved member of the synchronized PTP Instance currentDS, which is copied to:

    iii) the stepsRemoved member of the currentDS of the unsynchronized PTP Instance; and

iv) the gmPriorityVector of the unsynchronized PTP Instance.

NOTE 1—With the above, the secondary/primary PTP Instance state machines operate as though the time synchronization information had been received from the secondary/primary PTP Instance timeReceiver port. The SiteSync entity of the secondary/primary PTP Instance transfers the timing information to the PortSync entity of each timeTransmitter port of the secondary/primary PTP Instance. Each PortSyncSync state machine computes rateRatio, which now is relative to the primary/secondary PTP Instance GM. Each MDSyncSend state machine computes the fields of transmitted Sync and, in the two-step case, Follow_Up messages. The copied syncReceiptTimeout time is less than currentTime because sync receipt timeout has not occurred at the primary/secondary PTP Instance.

NOTE 2—The split functionality is used to transfer time synchronization information from the PTP Instance for which isSynced is TRUE to the PTP Instance for which isSynced is FALSE. It is not meant to cover the case where isSynced is FALSE for both the primary and secondary PTP Instances.

18.5.3.5 Both PTP Instances have isSynced FALSE

When isSynced is FALSE for both the primary and secondary PTP Instances (i.e., both are not synchronized)the HotStandbySystemState is OUT_OF_SYNC and the HotStandbySystem shall transfer phase, frequency, gmTimeBaseIndicator (see 10.2.4.15), lastGmPhaseChange (see 10.2.4.16), and lastGmFrequencyChange (see 10.2.4.17) from the ClockTimeReceiver of the PTP Instance for which isSynced was TRUE, before the HotStandbySystem transitioned to the OUT_OF_SYNC state, to the ClockTarget if a ClockTarget is present.

When the HotStandbySystem state is OUT_OF_SYNC, time synchronization performance is not required to meet the respective application or profile standard (see 3.24) requirements. Nevertheless, in order to mitigate LocalClock frequency drift, the PTP Instance should adjust the phase/frequency of its LocalClock using the data stored while isSynced for that PTP Instance was TRUE and the HotStandbySystem was in the REDUNDANT or NOT_REDUNDANT state.

18.5.4 State diagram

The HotStandbySystem state machine shall implement the function specified by the state diagram in Figure 17-3, the variables specified in 18.5.1 and 18.5.2, and the state requirements specified in 18.5.3.

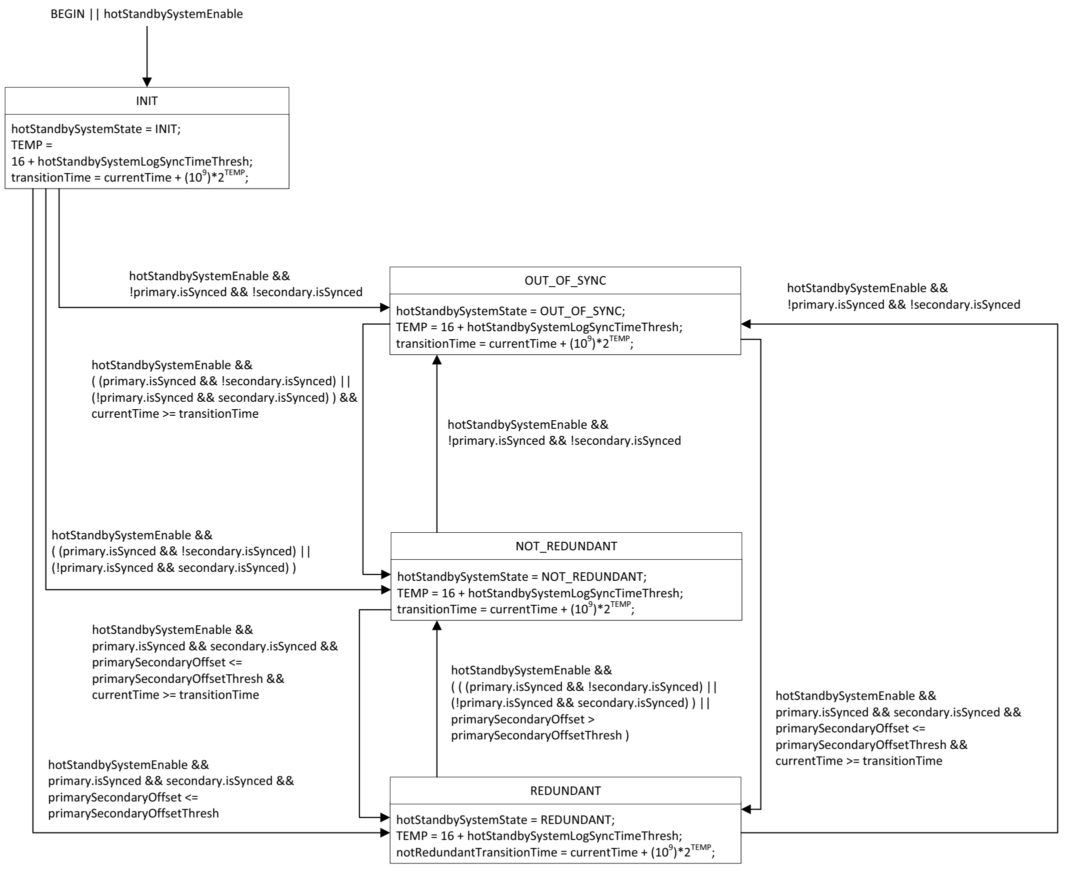

BEGIN || hotStandbySystemEnable
INIT
hotStandbySystemState = INIT;
TEMP =
16 + hotStandbySystemLogSyncTimeThresh;
transitionTime = currentTime + (10^9)*2^TEMP;
hotStandbySystemEnable && !primary.isSynced && !secondary.isSynced
OUT_OF_SYNC
hotStandbySystemState = OUT_OF_SYNC;
TEMP = 16 + hotStandbySystemLogSyncTimeThresh;
transitionTime = currentTime + (10^9)*2^TEMP;
hotStandbySystemEnable && !primary.isSynced && !secondary.isSynced
hotStandbySystemEnable && ((primary.isSynced && !secondary.isSynced) || (!primary.isSynced && secondary.isSynced)) && currentTime >= transitionTime
hotStandbySystemEnable && !primary.isSynced && !secondary.isSynced
hotStandbySystemEnable && ((primary.isSynced && !secondary.isSynced) || (!primary.isSynced && secondary.isSynced))
NOT_REDUNDANT
hotStandbySystemState = NOT_REDUNDANT;
TEMP = 16 + hotStandbySystemLogSyncTimeThresh;
transitionTime = currentTime + (10^9)*2^TEMP;
hotStandbySystemEnable && primary.isSynced && secondary.isSynced && primarySecondaryOffset <= primarySecondaryOffsetThresh && currentTime >= transitionTime
hotStandbySystemEnable && (((primary.isSynced && !secondary.isSynced) || (!primary.isSynced && secondary.isSynced)) || primarySecondaryOffset > primarySecondaryOffsetThresh)
hotStandbySystemEnable && primary.isSynced && secondary.isSynced && primarySecondaryOffset <= primarySecondaryOffsetThresh && currentTime >= transitionTime
hotStandbySystemEnable && primary.isSynced && secondary.isSynced && primarySecondaryOffset <= primarySecondaryOffsetThresh
REDUNDANT
hotStandbySystemState = REDUNDANT;
TEMP = 16 + hotStandbySystemLogSyncTimeThresh;
notRedundantTransitionTime = currentTime + (10^9)*2^TEMP;

Figure 18-3—HotStandbySystem State Diagram

18.6 PrimarySecondaryOffset state machine

This state machine updates the value of the global variable primarySecondaryOffset (see 18.5.1.4) whenever a new value of clockTimeReceiverTime (see 10.2.4.3) is computed for either the primary or secondary PTP Instance. A new value of clockTimeReceiverTime is computed when the global variable rcvdPSSyncCSS (see 10.2.4.28) or the global variable rcvdLocalClockTickCSS (see 10.2.4.29) is set to TRUE.

18.6.1 State diagram

The PrimarySecondaryOffset state machine shall implement the function specified by the state diagram in Figure 18-4..

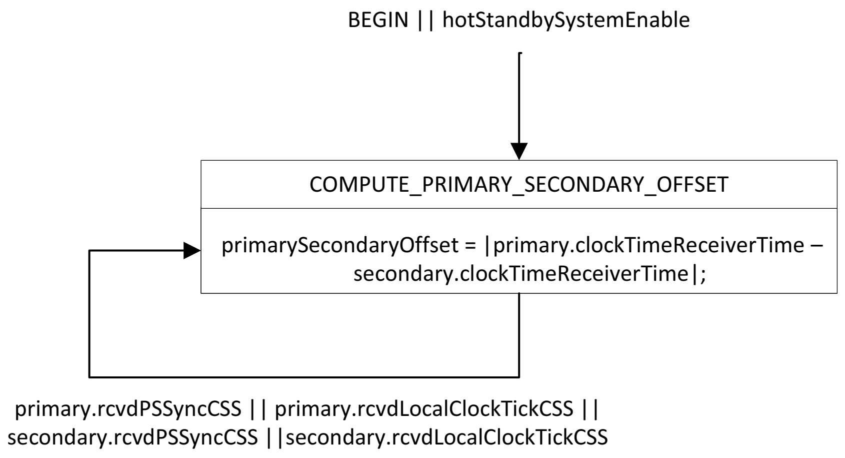

BEGIN || hotStandbySystemEnable
COMPUTE_PRIMARY_SECONDARY_OFFSET
primarySecondaryOffset = |primary.clockTimeReceiverTime – secondary.clockTimeReceiverTime|;
primary.rcvdPSSyncCSS || primary.rcvdLocalClockTickCSS || secondary.rcvdPSSyncCSS ||secondary.rcvdLocalClockTickCSS

Figure 18-4—PrimarySecondaryOffset State Diagram

Annex A

(normative)

Protocol Implementation Conformance Statement (PICS) proforma[3]

A.5 Major capabilities

Change the table of A.5 as follows:

Item	Feature	Status	References	Support
~~DOM0~~	~~Does the time-aware system support a PTP Instance with domain number 0, in accordance with the requirements of 8.1?~~	~~M~~	~~item a) of 5.4, 8.1~~	~~Yes [ ]~~
DOMADD	Does the time-aware system support one or more PTP Instances with ~~domain number~~domainNumber in the range 1 to 127?	O	item f) of 5.4.2, 8.1	Yes [ ] No [ ]
SIG	Does the PTP Instance transmit Signaling messages?	~~O~~M	~~10.2.13, item e) of 5.4.2,~~ item i) of 5.4.1, 10.4, 10.6.4, A.8	Yes [ ] ~~No [ ]~~
APPL	Does the PTP Instance support one or more of the application interfaces?	O	item i) of 5.4.2, Clause 9, A.20	Yes [ ] No [ ]
HOTSTDBY	Does the time-aware system support hot standby as specified in Clause 17?	O	Clause 17, 14.8, item l) of 5.4.2, 9.3.3.4, 9.4.3.4, 9.5.3.4, 9.6.2.6	Yes [ ] No [ ]
DRFTRK	Does the PTP Instance support the Drift_Tracking TLV as specified in 10.2.4.25, 10.2.4.26, 10.2.4.27, 11.2.14, 11.2.15, and 11.4.4?	O	item n) of 5.4.2, 10.2.4.25, 10.2.4.26, 10.2.4.27, 11.2.14, 11.2.15, and 11.4.4.	Yes [ ] No [ ]

[3] *Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

A.7 Minimal time-aware system

Change the table of A.7 as follows:

Item	Feature	Status	References	Support
MINTA-6	Is the ~~domain number~~domainNumber for all transmitted messages in the range 0 through 127, in compliance with the requirements of 8.1?	M	8.1	Yes [ ]
MINTA-8	Is the ~~domain number~~domainNumber for at least one of the gPTP domains supported by the time-aware system, in compliance with the requirements of 8.1?	M	8.1	Yes [ ]
MINTA-10	If path delay asymmetry is modeled by this PTP Instance, does it comply with the requirements of 8.3?	O	8.3	Yes [ ] No [ ] N/A [ ]
MINTA-15	Does the PTP Instance support the state machines related to signaling gPTP capability?	M	item h) of 5.4.1, 10.4.2	Yes [ ]
MINTA-16	For receive of all messages and for transmit of all messages except Announce ~~and Signaling~~, does the PTP Instance support the message requirements?	M	item i) of 5.4.1, 10.5, 10.6, 10.7	Yes [ ]

Insert the following items after MINTA-20 in the table of A.7:

Item	Feature	Status	References	Support
MINTA-21	Do the PTP Ports of this PTP Instance implement the functionality of the GptpCapableIntervalSetting state machine in compliance with the requirements of 10.4.4 and Figure 10-23?	MINTA-15:O	10.4.4, item i) of 5.4.2, item i) of 5.4.1	Yes [ ] No [ ]
MINTA-22	Does the PTP Instance implement the PtpInstanceSyncStatus state machine of 18.4 and the PtpInstanceStatusDS of 14.8?	¬HOTSTDBY:O	18.4, 14.8	Yes [ ] No [ ]

A.8 Signaling

Change SIG-7 of the table of A.8 as follows:

Item	Feature	Status	References	Support
SIG-7	Does the message interval request TLV for Signaling messages comply with the requirements in 10.6.4.3?	~~SIG:M~~ BMC-23:M, MIMSTR-16:M, MDFDPP-6:M	10.6.4.3	Yes [ ]

A.9 Best timeTransmitter clock

Change BMC-11 of the table of A.9 as follows:

Item	Feature	Status	References	Support
BMC-11	Is the PTP Port number equal to 1 in compliance with the requirements of 8.5.2.3?	~~¬~~¬BRDG:M	8.5.2.3	Yes [ ] N/A [ ]

Insert the following item after BMC-22 in the table of A.9:

Item	Feature	Status	References	Support
BMC-23	Do the TimeTransmitterPorts of this PTP Instance implement the functionality of the AnnounceIntervalSetting state machine in compliance with the requirements of 10.3.17 and Figure 10-19?	BMC:O	10.3.17, item h) of 5.4.2, item i) of 5.4.1	Yes [ ] No [ ]

A.10 Grandmaster-capable PTP Instance

Change GMCAP-2 of the table of A.10 as follows:

Item	Feature	Status	References	Support
GMCAP-2	Does the PTP Instance implement the functionality specified by the Clock~~TimeTransmitter~~SyncOffset state machine in compliance with the requirements of 10.2.10 and Figure 10-6?	GMCAP:M	10.2.10	Yes [ ]

A.11 Media-independent timeTransmitter

Insert the following item after MIMSTR-15 in the table of A.11:

Item	Feature	Status	References	Support
MIMSTR-16	Do the TimeTransmitterPorts of this PTP Instance implement the functionality of the SyncIntervalSetting state machine in compliance with the requirements of 10.3.18 and Figure 10-20?	MIMSTR:O	10.3.18, item b)3) of 5.4.2, item i) of 5.4.1	Yes [ ] No [ ]

A.13 Media-dependent, full-duplex point-to-point link

Delete MDFDPP-5 from the table of A.13, and renumber subsequent items as appropriate

Change MDFDPP-6 in the table of A.13 as follows:

Item	Feature	Status	References	Support
MDFDPP-6	Does this port implement the functionality of the LinkDelayIntervalSetting state machine in compliance with the requirements of 11.2.21 and Figure 11-11?	MDFDPP:~~M~~O	11.2.21, item i) of 5.5, item i) of 5.4.1	Yes [ ] No [ ]
MDFDPP-36	Does this port support two-step capability on receive?	MDFDPP:M	11.2.14, item d) of 5.5	Yes [ ] No [ ]
MDFDPP-37	Does this port support two-step capability on transmit?	MDFDPP:M	11.2.15, item e) of 5.5	Yes [ ] No [ ]

Change the table of A.19 as follows:

A.19 Remote management

Item	Feature	Status	References	Support
	If item RMGT is not supported, mark N/A.			N/A[ ]
RMGT-1	What management protocol standard(s) or specification(s) are supported?	RMGT:M	item k) 1) of 5.4.2	
RMGT-2	What standard(s) or specification(s) for managed object definitions and encodings are supported for use by the remote management protocol?	RMGT:M	item k) 2) of 5.4.2	
RMGT-3	If the Simple Network Management Protocol (SNMP) is listed in RMGT-2, is the IEEE 8021-AS-MIB module fully supported (per its MODULE-COMPLIANCE)?	RMGT:O	item k) 3) of 5.4.2, Clause 15	Yes [ ] No [ ] N/A [ ]
RMGT-4	If a remote management protocol that supports YANG is listed in RMGT-2, is the YANG data model ieee802-dot1as-ptp of Clause 17 supported?	RMGT:O	item k) 4) of 5.4.2, Clause 17	Yes [ ] No [ ]
RMGT-5	If a remote management protocol that supports YANG is listed in RMGT-2, and if hot standby is supported, is the YANG data model ieee802-dot1as-hs of Clause 17 supported?	RMGT:O	item k)5) of 5.4.2, Clause 17, Clause 18	Yes [ ] No [ ] N/A [ ]

Annex F

(informative)

PTP profile included in this standard

F.3 PTP attribute values

Change F.3 a) as follows:

a) A domain whose ~~domain number~~domainNumber is 0 is present. A domain whose ~~domain number~~domainNumber is in the range 1 through 127 can be present (see 8.1).

Annex G

(informative)

The asymmetry compensation measurement procedure based on line-swapping

G.3 Measurement procedure

Change the second paragraph of G.3 as follows:

The criteria of 11.2.17 for determining whether the peer-to-peer delay mechanism is ~~instance~~the transport-specific peer-to-peer delay mechanism or ~~is provided by~~the CMLDS peer-to-peer delay mechansim apply here.

Change item f) as follows:

f) The NMS reads and saves the multiple sets of (t3', t4') from ACC2 through the MIB. Then the NMS can compute the delay asymmetry, in units of time, as [(t4' – t4) × neighborRateRatio – (t3' – t3)]/2. The NMS can use multiple sets of (t3, t4, t3', t4') to compute average values to get a more accurate result. The averaging method can be decided as required and is outside the scope of gPTP. If ACC1 and ACC2 are frequency synchronized, then neighborRateRatio is 1, and the delay asymmetry is [(t4' – t3') – (t4 – t3)]/2.

Annex H

(informative)

Bibliography

Insert the following bibliography reference after the last bibliography reference:

[B30] "Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)," IEEE Registration Authority (https://standards.ieee.org/products-programs/regauth/).