P802.1ASdm/D1.1
May 29, 2023
(Amendment to
IEEE Std 802.1AS™-2020
as amended by
IEEE 802.1AS-2020/Cor 1-2021)

Draft Standard for
Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications

Amendment: Hot Standby

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Time-Sensitive Networking Task Group of IEEE 802.1

IEEE Standards Association
445 Hoes Lane
Piscataway, NJ 08854, USA

The text proper of this draft begins with the 2023 Title page. The cover pages (a), (b), (c) etc. are for 802.1 WG information, and will be removed prior to Sponsor Ballot.

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

Editors' Foreword

This draft standard is an amendment. The scope of changes to the base standard is thus strictly limited, as detailed in the PAR.

Information on participation in this project, and in the IEEE 802.1 Working Group can be found here.

Participation in 802.1 standards development

Comments on this draft are encouraged. **NOTE: All issues related to IEEE standards presentation style, formatting, spelling, etc. are routinely handled between the 802.1 Editor and the IEEE Staff Editors prior to publication, after balloting and the process of achieving agreement on the technical content of the standard is complete.** Readers are urged to devote their valuable time and energy only to comments that materially affect either the technical content of the document or the clarity of that technical content. Comments should not simply state what is wrong, but also what might be done to fix the problem.

Full participation in the work of IEEE 802.1 requires attendance at IEEE 802 meetings. Information on 802.1 activities, working papers, and email distribution lists etc. can be found on the 802.1 Website:

http://ieee802.org/1/

Use of the email distribution list is not presently restricted to 802.1 members, and the working group has a policy of considering ballot comments from all who are interested and willing to contribute to the development of the draft. Individuals not attending meetings have helped to identify sources of misunderstanding and ambiguity in past projects. The email lists exist primarily to allow the members of the working group to develop standards, and are not a general forum. All contributors to the work of 802.1 should familiarize themselves with the IEEE patent policy and anyone using the mail distribution will be assumed to have done so. Information can be found at http://standards.ieee.org/about/sasb/patcom/materials.html/

Comments on this document may be sent to the 802.1 email exploder, to the Editor, or to the Chair of the 802.1 Working Group.

Geoffrey Garner
Editor, P802.1AS
Email:gmgarner@alum.mit.edu

Glenn Parsons
Chair, 802.1 Working Group
Email: glenn.parsons@ericsson.com

NOTE: Comments whose distribution is restricted in any way cannot be considered, and may not be acknowledged.

All participants in IEEE standards development have responsibilities under the IEEE patent policy and should familiarize themselves with that policy, see
http://standards.ieee.org/about/sasb/patcom/materials.html

As part of our IEEE 802 process, the text of the PAR and CSD (Criteria for Standards Development, formerly referred to as the 5 Criteria or 5C's) is reviewed on a regular basis in order to ensure their continued validity. A vote of "Approve" on this draft is also an affirmation by the balloter that the PAR is still valid.

Project Authorization Request, Scope, Purpose, and Criteria for Standards Development (CSD)

The complete amendment PAR, as approved by IEEE NesCom 3rd June 2020, can be found at:

https://development.standards.ieee.org/myproject-web/public/view.html#pardetail/8208

<<Editor's Note: The above date of approval of the amendment PAR and link will be updated when available. The text of the Scope of the Proposed changes and the Need for the Project reflect the updated PAR text approved by 802.1.>>

The 'Scope of the Proposed changes' and the 'Need for the Project' specify the changes to be made by this amendment (see below).

Scope of the Proposed changes:

This amendment specifies protocols, procedures, and managed objects for hot standby without use of the Best Master Clock Algorithm (BMCA), for time-aware systems, including:
- 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. This amendment specifies a Type-Length-Value (TLV) to enable 1 μs time synchronization accuracy over 100 network hops. This amendment also addresses errors and omissions in the description of existing functionality.

Need for the Project:

Hot standby is needed in some applications that use time synchronization (e.g., industrial automation and automotive in-vehicle networks). A TLV is needed by IEC/ IEEE 60802 Time-Sensitive Networking Profile for Industrial Automation to achieve the goal of 1 μs time synchronization accuracy over 100 network hops while using existing silicon and low-cost crystal oscillators, i.e., not, for example, temperature compensated crystal oscillators.

Maintenance items included in this amendment

This amendment includes 802.1 maintenance items 263, 278, 279, 280, 281, 283, 289, 292, 293, 294, 297, 307, 308, 327, 328, 331, 332, 334, and 335. These maintenance items are documented at https://www.802-1.org/items?utf8=?&search=802.1AS&open=1&commit=Search. The maintenance items do the following:

a) Maintenance item 263 clarifies that a Signaling message can contain more than one message interval request TLV, but that the gPTP capable TLV is sent in a separate Signaling message.
b) Maintenance item 278 makes it clear that the message interval request TLVs and state machines are optional (but the gPTP capable TLV and state machine is mandatory), and fixes inconsistencies between clause5 and the PICS related to this.
c) Maintenance item 279 fixes a typo, by changing externalPortConfiguration to externalPortConfigurationEnabled in one place.
d) Maintenance item 280 updates some terminology in MIB descriptions.

<<Editor's Note: It was decided at the July 11, 2020 TSN TG meeting that the MIB will not be updated for 802.1ASdm (see Z.1.6, item b)). This means that maintenance item 280 will not be done for 802.1ASdm. However, the terminology corrections identitied for the MIB in maintenance item 280 will be reflected in corresponding YANG descriptions.>>

e) Maintenance item 281 fixes a typo, by changing Delay_Req to Pdelay_Req in one place.

f) Maintenance item 283 changes “domain number” to “domainNumber.”
g) Maintenance item 289 removes several incorrect references to 11.2.13.
h) Maintenance items 292 and 293 fix some capitalization.
i) Maintenance item 294 corrects the description of the ptpTimescale flag in Table 10-9 for the case of domain 0.
j) Maintenance item 297 makes editorial corrections to Table 11-6 and PICS item MINTA-6.
k) Maintenance item 307 fixes a type by changing portEnabledOper to portOper in Figure and Figure .
l) Maintenance item 308 fixes incorrect logic in the MDSyncSendSM state machine.
m) Maintenance item 327 removes the undefined variable domainEnabled from the GptpCapableTransmit and GptpCapableReceive state machines.
n) Maintenance item 328 corrects a comment that is incorrect in functions used by the GptpCapableIntervalSetting, AnnounceIntervalSetting, SyncIntervalSetting, and LinkDelayIntervalSetting state machines.
o) Maintenance item 331 corrects the computation of meanLinkDelay for the case where CMLDS is used.
p) Maintenance item 332 corrects the setting of the variable newInfo in the PortStateSettingExt state machine so that an Announce message is sent immediately when the port state is set to MasterPort.
q) Maintenance item 334 corrects the error in the computation of clockSourcePhaseOffset in Figure 10-6.
r) Maintenance item 335 clarifies the difference between currentTime and localTime used in 10.2.9.2.1.
s) Maintenance item 360 adds ranges and defaults for syncReceiptTimeout and announceReceiptTimeout in 10.7.3.1 and 10.7.3.2, respectively, and fixes the range for gPtpCapableReceiptTimeout in 10.7.3.3.

This page intentionally left blank.

This page intentionally left blank.

P802.1ASdm/D1.1
May 29, 2023
(Amendment to
IEEE Std 802.1AS™-2020)

Draft Standard for Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications

Amendment: Hot Standby

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 master, 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

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents." They can also be obtained on request from IEEE or viewed at http://standards.ieee.org/ipr/disclaimers.html.

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

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at https://ieeexplore.ieee.org or contact IEEE at the address listed previously. For more information about the IEEE SA or IEEE's standards development process, visit the IEEE-SA Website at https://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: https://standards.ieee.org/standard/index.html. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at https://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

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>>

When the IEEE-SA Standards Board approved this standard on <dd> <month> <year>, it had the following membership:

Jean-Philippe Faure, *Chair*
Vacant Position, *Vice-Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Chuck Adams
Masayuki Ariyoshi
Ted Burse
Stephen Dukes
Doug Edwards
J. Travis Griffith
Gary Hoffman
Michael Janezic
Thomas Koshy
Joseph L. Koepfinger1
Kevin Lu
Daleep Mohla
Damir Novosel
Ronald C. Petersen
Annette D. Reilly
Robby Robson
Dorothy Stanley
Adrian Stephens
Mehmet Ulema
Phil Wennblom
Howard Wolfman
Yu Yuan

*Member Emeritus

<<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

List of Figures

List of Tables

Draft IEEE Standard for Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications

Amendment: Hot Standby

[This amendment is based on IEEE Std 802.1AS™-2020 as modified by P802.AS-2020/Cor1.]

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 SlavePort 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.

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

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 ClockSlaveSync state machine (10.2.13).
f) ~~Support the following best master clock algorithm (BMCA) requirements:~~
    1) ~~Implement the BMCA (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 BMCA 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 BMCA 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:

k) Support the following for Grandmaster PTP Instance capability:
    1) Support the media-independent master 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 ClockMasterSyncSend state machine (10.2.9).

4) Implement the Clock~~Master~~SyncOffset state machine (10.2.10).
5) Implement the ClockMasterReceive 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.

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

m) Implement hot standby as specified in Clause 17, 14.8, 9.3.3.4, 9.4.3.4, 9.5.3.3, 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. Where not further qualified, by reference to the use of other capabilities or by profile standards, these defaults are intended to facilitate deployment in target environments and applications with time-aware system implementations conformant to prior revisions of this standard, including IEEE Std 802.1AS-2020 and IEEE Std 802.1AS-2011. Mandatory requirements that this standard does not identify as defaults, or otherwise qualify, 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 master clock algorithm (BMCA) requirements:

a) Implement the BMCA (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 BMCA 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 BMCA 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 master 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. The level of fault tolerance required in a time-aware network can be very different for differing applications.

7.2.2 Time-aware network consisting of a single gPTP domain

Change the first sentence of the second paragraph of 7.2.2 as follows:

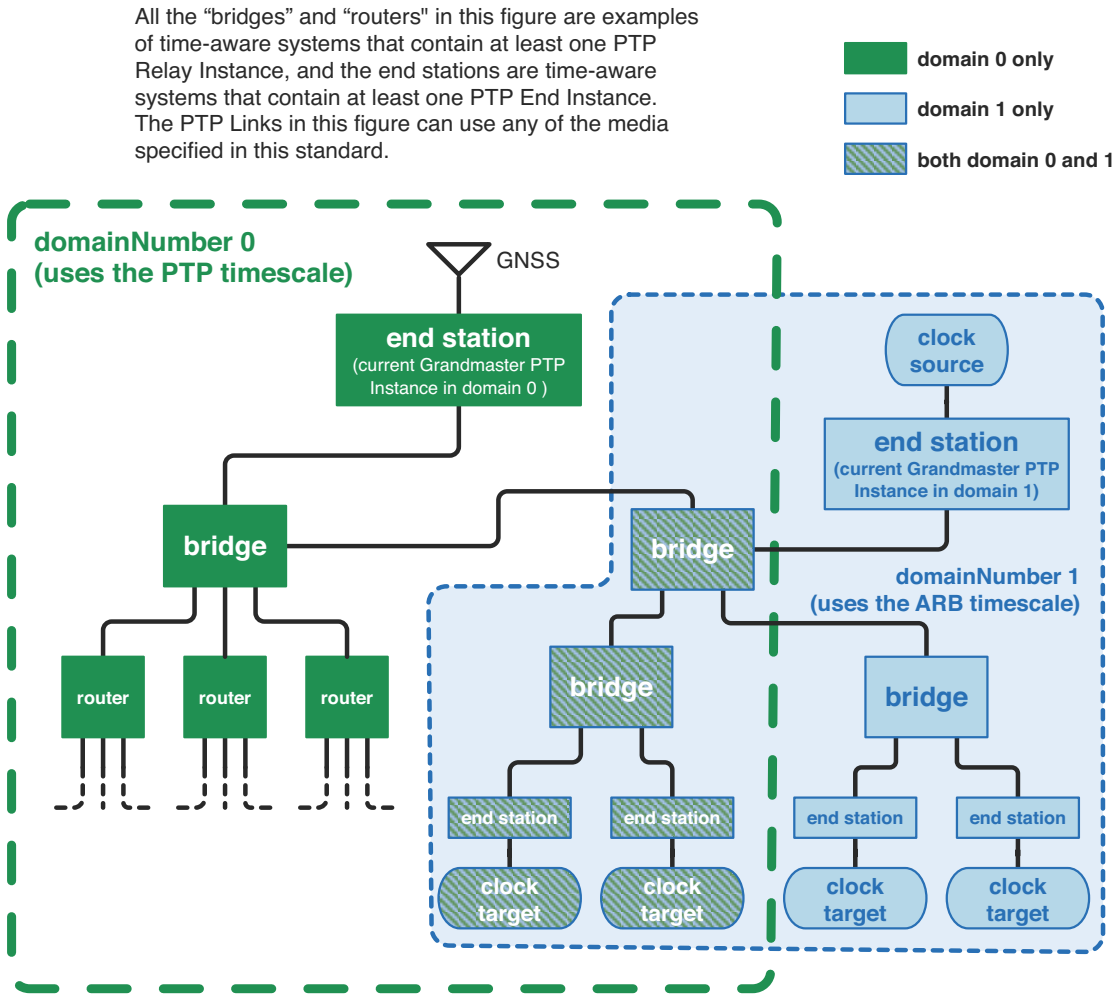
Any PTP Instance with clock sourcing capabilities can be a potential Grandmaster PTP Instance, and a selection method (the *best master clock algorithm*, or BMCA) ensures that all of the PTP Instances in ~~a~~ the gPTP domain use the same Grandmaster PTP Instance.[2]

Remove the last sentence of 7.2.2.

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

7.2.3 Time-aware network consisting of multiple gPTP domains

Replace Figure 7-3 with the following:



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.

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

Change 7.2.4 as follows:

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

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 (BMCA)~~

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 Master Clock Algorithm (BMCA) 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~~.

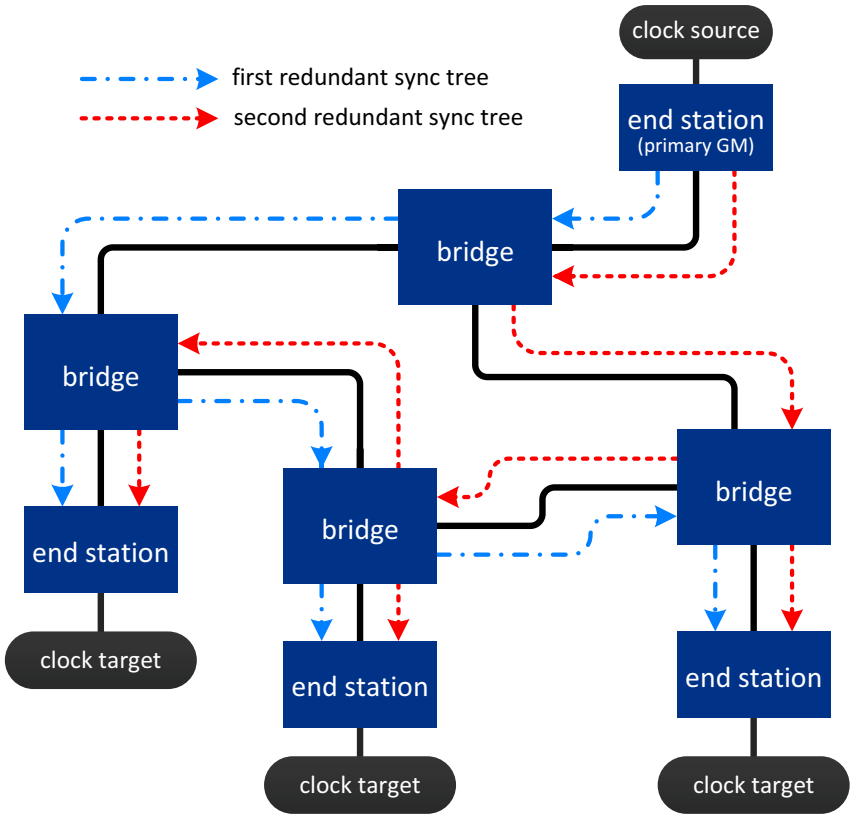
7.2.4.3 ~~Redundacy not fully specified in this standard~~

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.

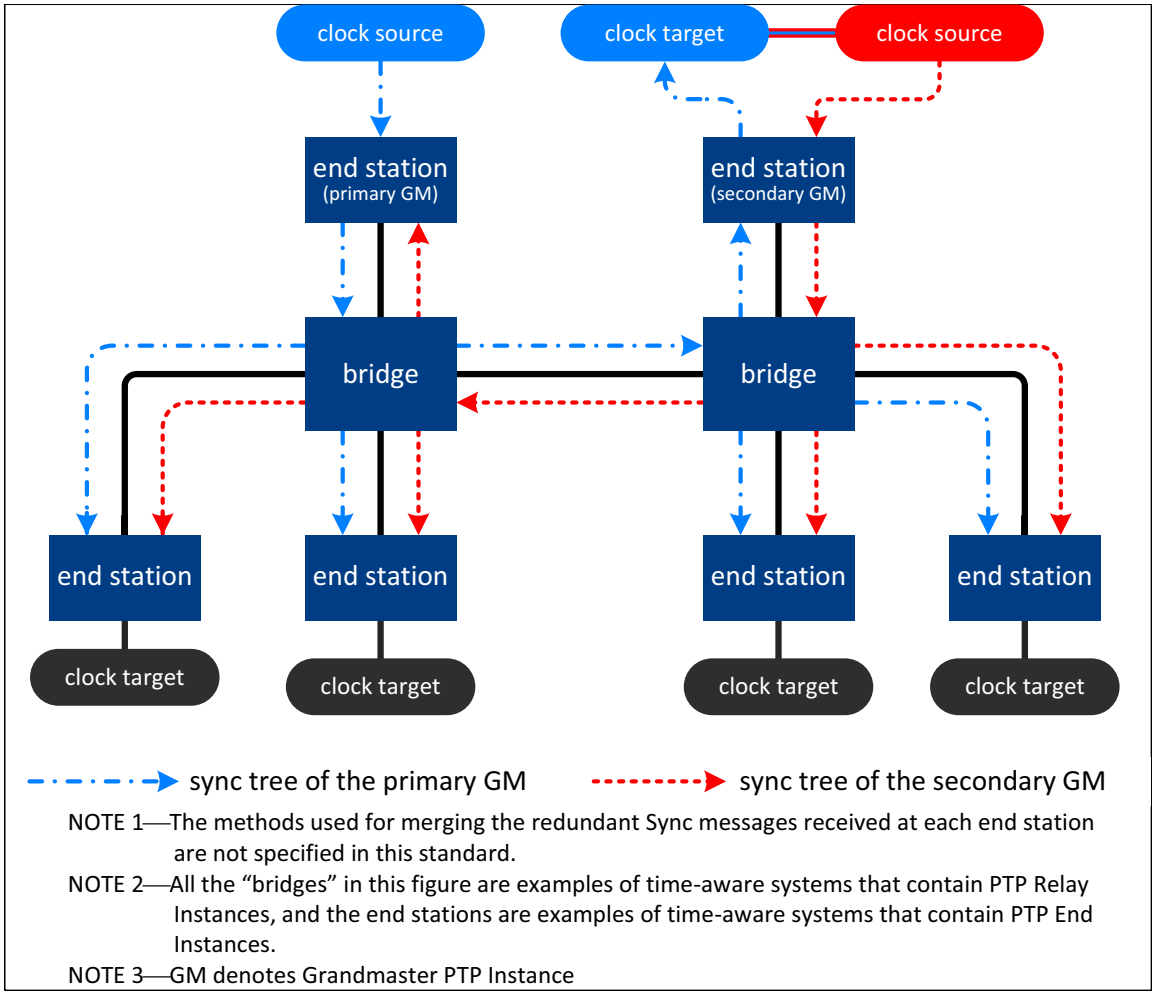
Figure 7-4 shows an example network realizing two redundant synchronization trees from a single GM, with each synchronization tree in a different gPTP domain (there are a total of two gPTP domains).



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.

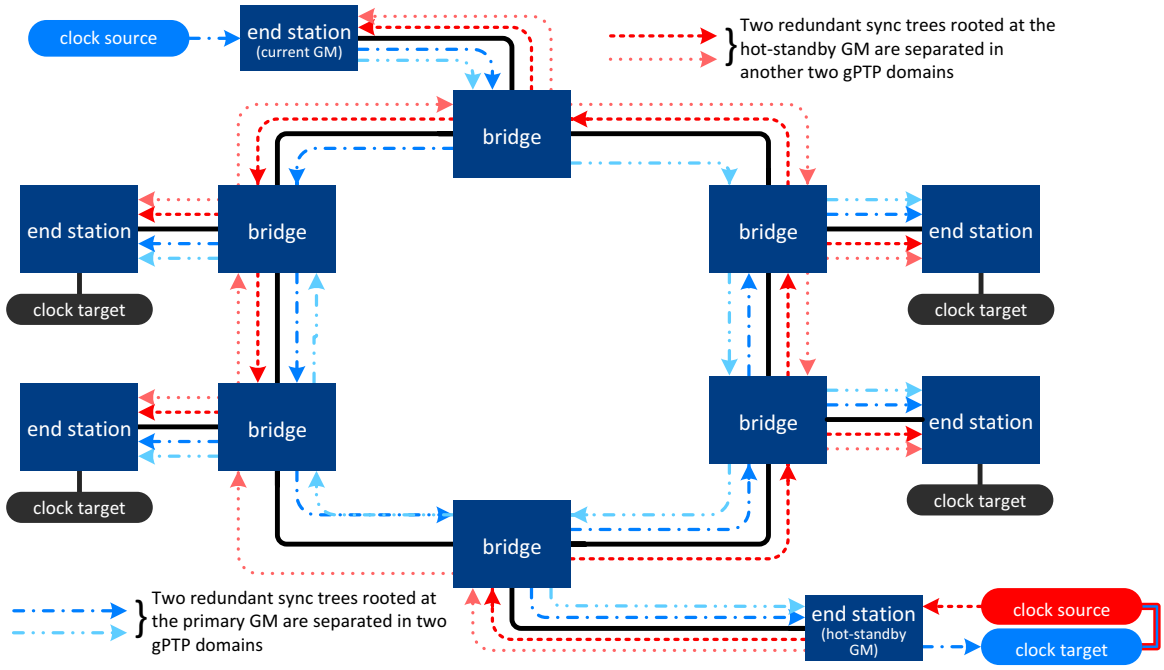
NOTE 3—GM denotes Grandmaster PTP Instance

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 and~~,~~ 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.



Note 1: The methods used for merging the redundant sync msgs 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 7-6—~~Time-aware network example for GM+synchronziation path redundancy, with one primary and one hot standby GM. Each GM establishes two sync trees, resulting in a total of four Sync trees that are separated in four gPTP domains~~

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

7.2.5 Time-aware network with hot standby

Figure 7-7 shows an example of a time-aware network with hot standby (see Clause 17). 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 17.6.3.4) is assumed to be disabled.

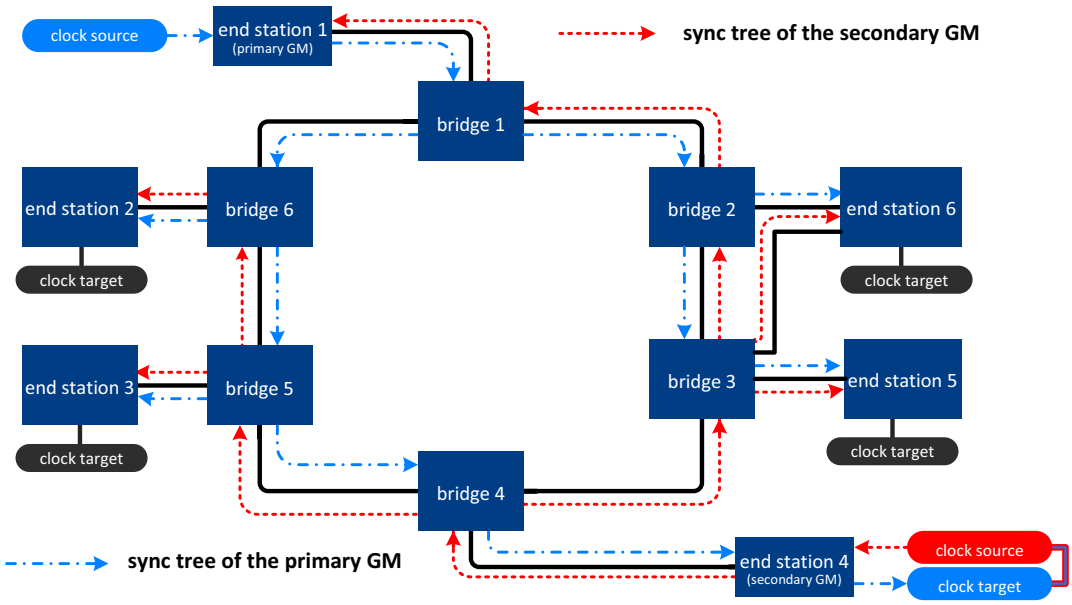
End station 1 is the primary GM (see Clause 17), and its synchronization spanning tree is illustrated by the blue arrows. End station 4 is the secondary GM, and its synchronization spanning tree is illustrated by the red arrows. Except for the links to end stations 2, 3, and 5, the network can tolerate 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 its ptpInstanceState changes to NOT_SYNCED; the hotStandbySystemState of end station 4 is then change to NOT_REDUNDANT (see Clause 17). 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 primarySecondaryOffsetThreshold (see Clause 17) 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 7-7—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 7-8. The interfaces are those specified inClause 9.

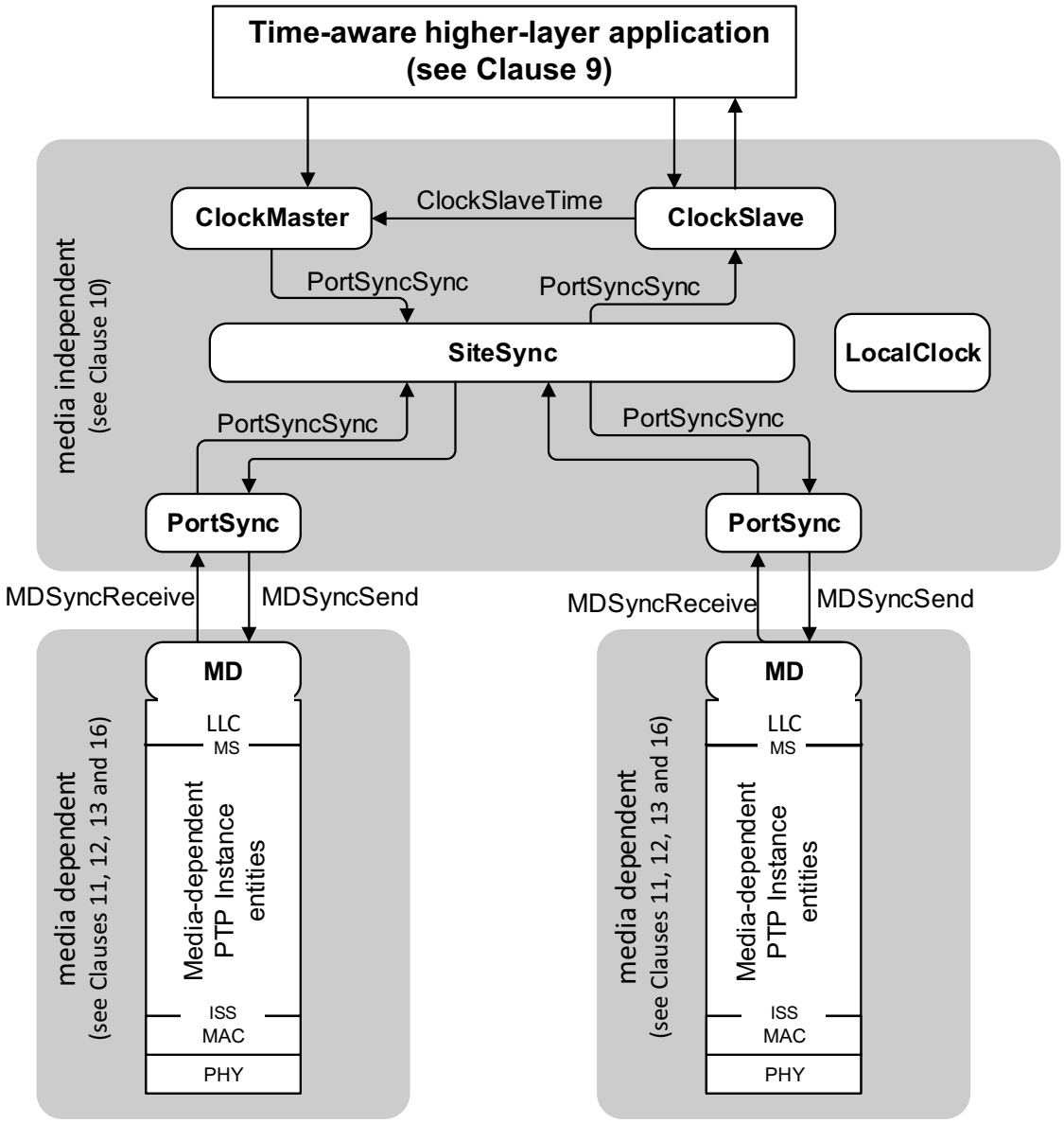


Figure 7-8—Model for PTP Instance and its interfaces to higher-layer applications~~model~~

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 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.

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 ClockMaster 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 ClockSlave 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,
        slaveTimeCallback,
        gmPresent,
        ptpInstanceState,
        grandmasterIdentity
}
```

9.3.3.1 domainNumber (UInteger8)

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

9.3.3.2 slaveTimeCallback (ExtendedTimestamp)

The value of slaveTimeCallback 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 ClockSlave 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 ptpInstanceState

The value of ptpInstanceState is the state of the ptpInstance (see 14.8.2), i.e., the value of the global variable ptpInstanceState (see 17.5.1.1). This parameter shall be present if the optional hot standby feature is implemented (see Clause 17) and may be present otherwise.

9.3.3.5 grandmasterIdentity

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 ClockSlave 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,
        ptpInstanceState,
        grandmasterIdentity
}
```

9.4.3.1 domainNumber (UInteger8)

This parameter is the ~~domain number~~domainNumber of the ClockSlave 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 slaveTimeCallback 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 slaveTimeCallback 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.

9.4.3.4 ptpInstanceState

The value of ptpInstanceState is the state of the ptpInstance (see 14.8.2), i.e., the value of the global variable ptpInstanceState (see 17.5.1.1). This parameter shall be present if the optional hot standby feature is implemented (see Clause 17) and may be present otherwise.

9.4.3.5 grandmasterIdentity

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

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 ClockSlave 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,
        slaveTimeCallback,
        ptpInstanceState,
        grandmasterIdentity
}
```

9.5.3.1 domainNumber (UInteger8)

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

9.5.3.2 slaveTimeCallback (ExtendedTimestamp)

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

9.5.3.3 ptpInstanceState

The value of ptpInstanceState is the state of the ptpInstance (see 14.8.2), i.e., the value of the global variable ptpInstanceState (see 17.5.1.1). This parameter shall be present if the optional hot standby feature is implemented (see Clause 17) and may be present otherwise.

9.5.3.4 grandmasterIdentity

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

9.6 ClockTargetPhaseDiscontinuity interface

Change 9.6.2 as follows:

9.6.2 ClockTargetPhaseDiscontinuity.result parameters

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

9.6.2.1 domainNumber (UInteger8)

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

9.6.2.2 gmIdentity (ClockIdentity)

If gmPresent (see 10.2.4.13) is TRUE, the value of gmIdentity is the ClockIdentity of the current Grandmaster PTP Instance. If gmPresent is FALSE, the value of gmIdentity 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 ptpInstanceState

The value of ptpInstanceState is the state of the ptpInstance (see 14.8.2), i.e., the value of the global variable ptpInstanceState (see 17.5.1.1). This parameter shall be present if the optional hot standby feature is implemented (see Clause 17) and may be present otherwise.

9.6.2.7 grandmasterIdentity

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

10. Media-independent layer specification

10.2 Time-synchronization state machines

Change 10.2.1 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 ClockMasterSyncReceive, Clock~~Master~~SyncOffset, and ClockMasterSyncSend 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 ClockMasterSyncSend 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) ClockMasterSyncReceive (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 masterTime, and provides masterTime to Clock~~Master~~SyncOffset and ClockMasterSyncSend state machines.

b) Clock~~Master~~SyncOffset (one instance per PTP Instance): receives syncReceiptTime from the ClockSlave entity and masterTime from the ClockMasterSyncReceive state machine, computes phase offset and frequency offset between masterTime and syncReceiptTime if the PTP Instance is not the Grandmaster PTP Instance, and provides the frequency and phase offsets to the ClockMasterSyncSend state machine.

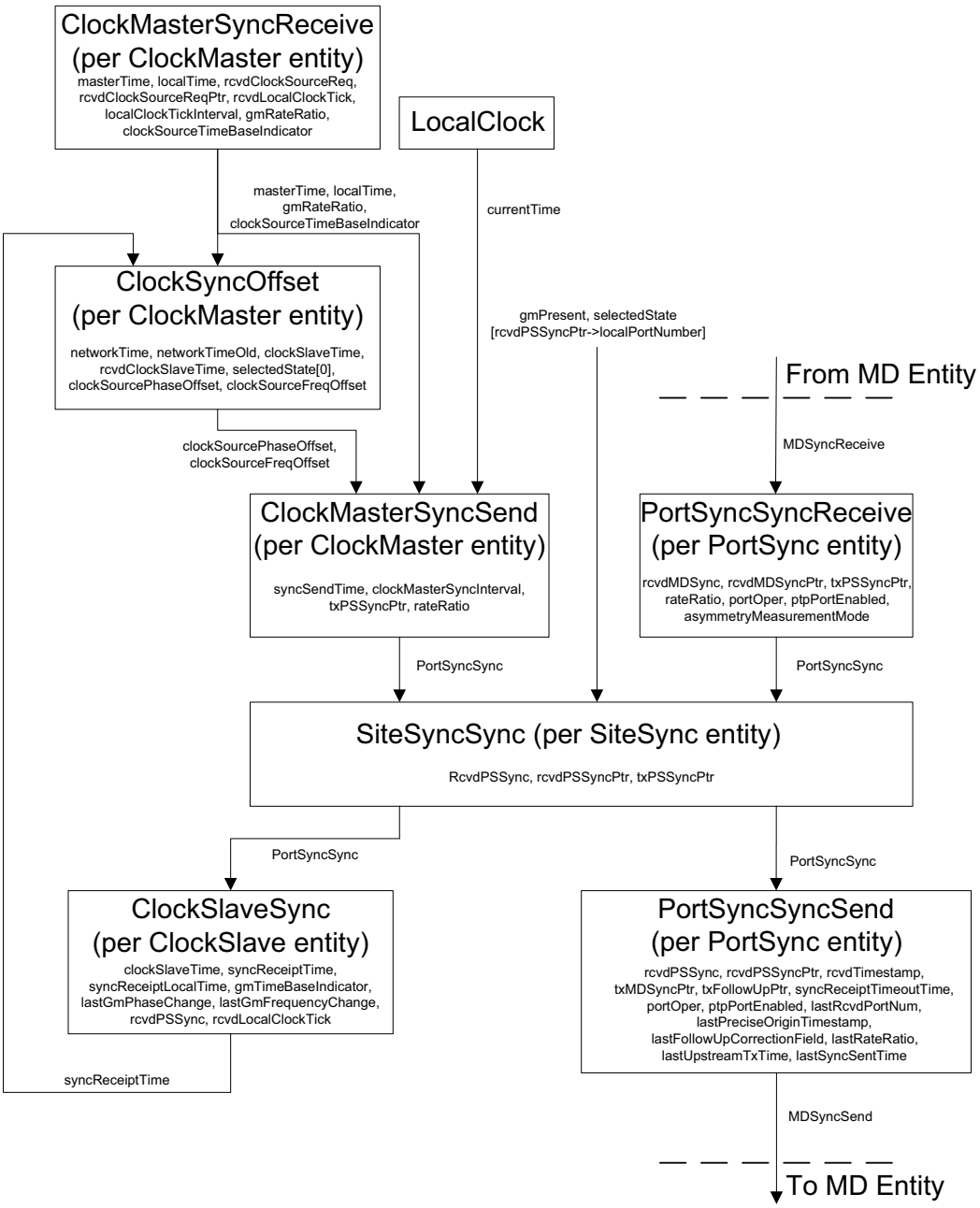
c) ClockMasterSyncSend (one instance per PTP Instance): receives masterTime from the ClockMasterSyncReceive state machine, receives phase and frequency offset between masterTime and syncReceiptTime from the Clock~~Master~~SyncOffset state machine, and provides masterTime (i.e., synchronized time) and 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 slave port or from the ClockMaster entity; and sends the information to the PortSync entities of all the ports and to the ClockSlave 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.

g) ClockSlaveSync (one instance per PTP Instance): receives time-synchronization information from the SiteSync entity; computes clockSlaveTime and syncReceiptTime; sets syncReceiptLocalTime,



NOTE 1—selectedState for each port and gmPresent are set by Port State Selection state machine (see 10.3.12)

NOTE 2—currentTime is a global variable that is always equal to the current time relative to the local oscillator

NOTE 3—application interfaces to higher layers are not shown

NOTE 4—The ClockMasterSyncReceive, ClockMasterSyncSend, and ClockSyncOffset state machines are optional for PTP Instances that are not grandmaster-capable.

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

GmTimeBaseIndicator, lastGmPhaseChange, and lastGmFreqChange; sends clockSlaveTime to the ClockMaster 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

Insert the following items after the final item in Table 10.1: .

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)	Instance used by CMLDS, per LinkPort
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		Yes	No	No	No
rcvdLocalClockTick		Yes	No	No	No

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, and 10.2.4.29, 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 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.

10.2.4.29 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.

10.2.5 Per port global variables

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 ClockMasterSyncSend 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 NOTE 1 to 10.2.9.2.1, just after item c)2), and renumber subsequent NOTES as needed:

NOTE 1—The definition of currentTime (see 10.2.4.12) and localTime (see 10.2.4.19), and the statement localTime = currentTime in the RECEIVE_SOURCE_TIME state of the ClockMasterSyncReceive state machine, imply that currentTime and localTime are always equal and the quantity gmRateRatio × (currentTime - localTime) is always zero. Both localTime and masterTime are updated by the ClockMasterSyncReceive state machine. The updates occur both when sourceTime is received from the ClockSource (indicated by the variable rcvdClockSourcReq being TRUE) and when the LocalClock entity itself updates by one tick (indicated by the variable rcvdLocalClockTickCMSR being TRUE). masterTime is either updated to the sourceTime (if the update occurs due to receipt of sourceTime from the ClockSource) or incremented by one LocalClock tick interval multiplied by gmRateRatio. localTime is updated by setting it equal to currentTime. The result is that localTime is always equal to the value of currentTime when masterTime was most recently updated. localTime, currentTime, and masterTime are per PTP Instance global variables. When they are updated, their values are known to all state machines. In setting (i) preciseOriginTimestamp to masterTime with fractional ns truncated, and followUpCorrectionField to (ii) the fractional ns part of masterTime, plus (iii) gmRateRatio × (currentTime - localTime), the contributions (i) plus (ii) give the estimate of the ClockSource time when masterTime was most recently updated, because this latest computed masterTime includes any updates due to LocalClock ticks. The quantity (iii) is zero in the model here, because currentTime and localTime are equal in this model. However, in practice there could be situations where currentTime and localTime are not equal. For example, if the ClockMasterSyncReceive state machine is implemented as a different module than the ClockMasterSyncSend state machine, with masterTime and localTime being sent from the ClockMasterSyncReceive state machine to the ClockMasterSyncSend state machine, then currentTime and localTime will not be equal. In this case, currentTime in c)2) is the value of the LocalClock time when the ClockMasterSyncSend state machine receives masterTime and localTime from the ClockMasterSyncReceive 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 ClockMasterSyncSend 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 masterTime and clockSourceTimeBaseIndicator from the ClockMasterSyncReceive state machine, and phase and frequency offset between masterTime and syncReceiptTime from the Clock~~Master~~SyncOffset state machine. It provides masterTime (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~~Master~~SyncOffset state machine

Change 10.2.10.3 as follows:

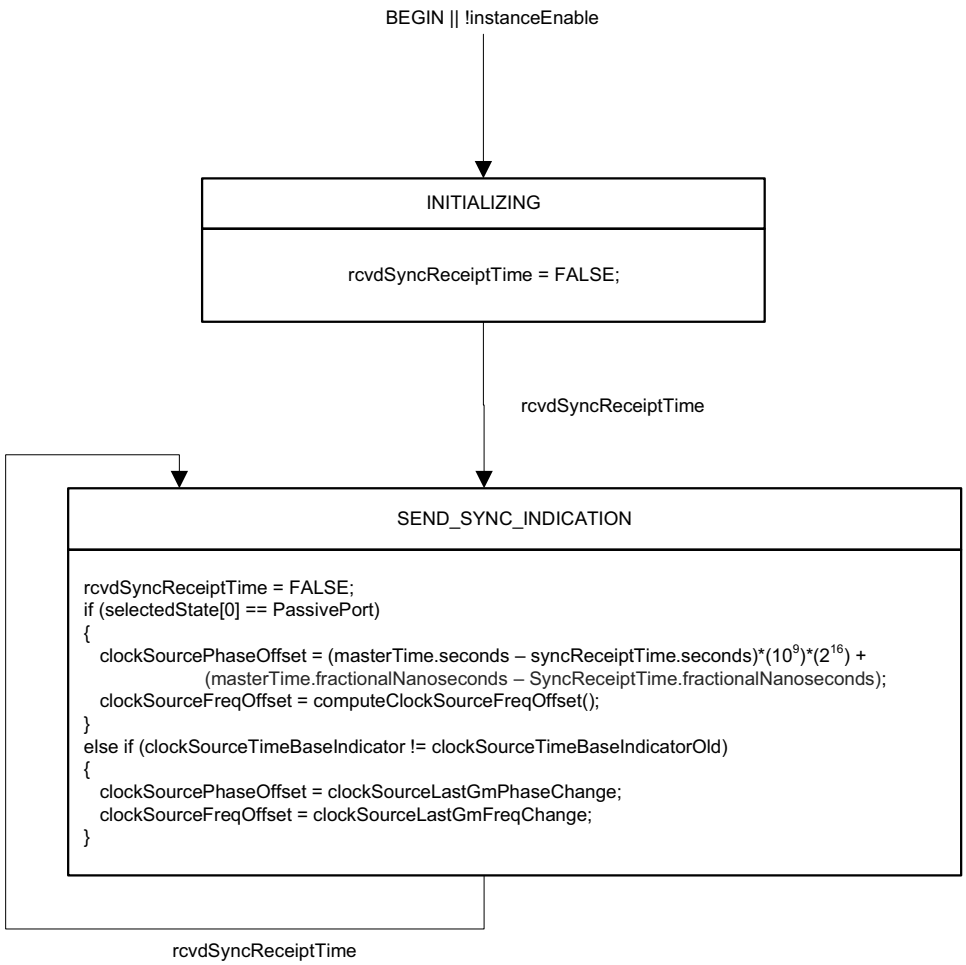
10.2.10.3 State diagram

The Clock~~Master~~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 ClockSlaveSync state machine and masterTime from the ClockMasterSyncReceive 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~~Master~~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 ClockMasterSyncSend 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 with the following figure:

.



This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that the statement

clockSourcePhaseOffset = masterTime.seconds – syncReceiptTime;

in the SEND_SYNC_INDICATION state is replaced by

clockSourcePhaseOffset = (masterTime.seconds – syncReceiptTime.seconds)*(10^9)*(2^{16}) +
(masterTime.fractionalNanoseconds – SyncReceiptTime.fractionalNanoseconds;

Figure 10-6—Clock~~Master~~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 ClockSlaveSync state machine

Change 10.2.13.1 as follows:

10.2.13.1 State machine variables

The following variables are 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.

10.2.13.2 State diagram

Change the first paragraph of 10.2.13.2 as follows:

The ClockSlaveSync 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 clockSlaveTime, and sets syncReceiptLocalTime (i.e., the time relative to the LocalClock entity corresponding to syncReceiptTime), GmTimeBaseIndicator, lastGmPhaseChange, and lastGmFreqChange. It provides clockSlaveTime to the Clock~~Master~~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 master clock selection, external port configuration, and announce interval setting state machines

10.3.1 Best master 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 BMCA, 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.

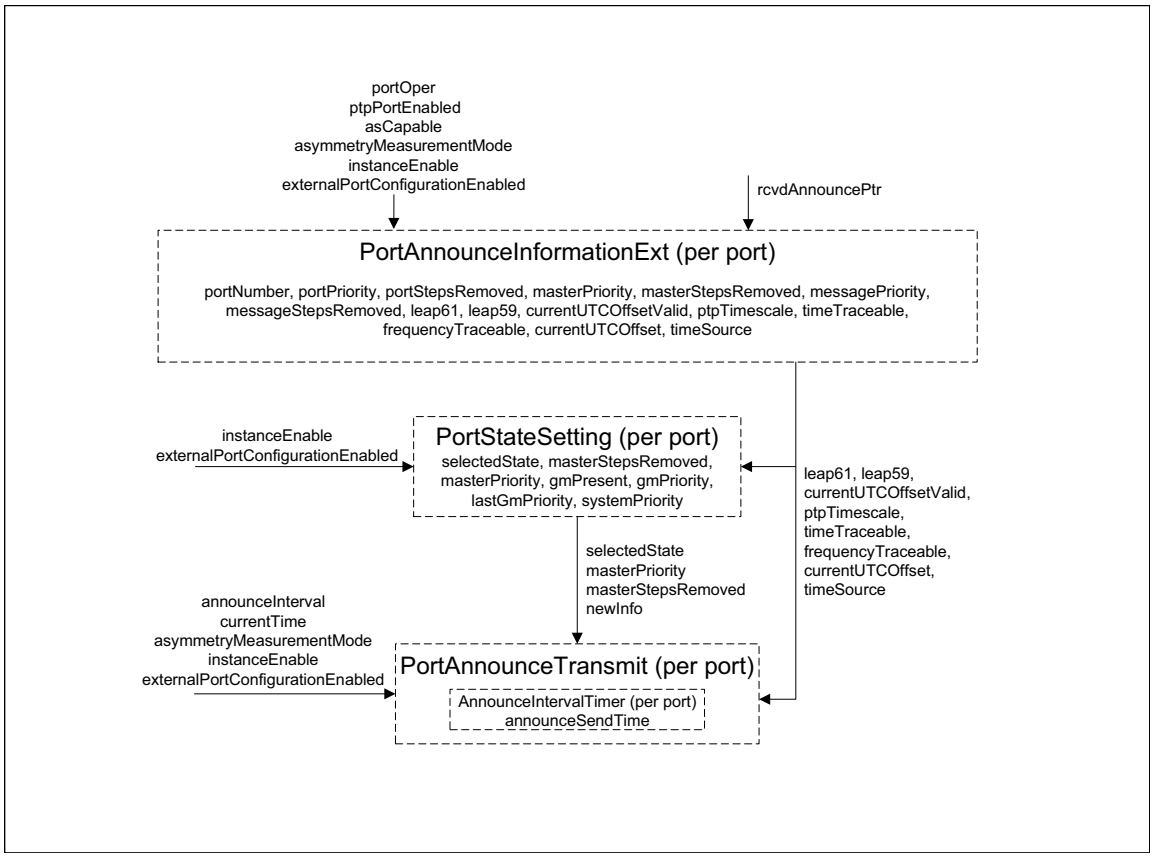
In the case of external port configuration, the time-synchronization spanning tree and the desired PTP Port states are controlled by an external entity, but the Grandmaster PTP Instance might change. If the slave port of a PTP Instance that is gmCapable (priority1 < 255, see 8.6.2.1) is no longer asCapable, the state of this PTP Port changes from SlavePort 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 slave port of a PTP Instance that is not gmCapable is no longer asCapable, the state of this PTP Port changes from SlavePort 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 the sync receipt timeout condition (see 10.7.3.1) and the announce receipt timeout condition (see 10.7.3.2).

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

10.3.7.2 External port configuration state machines overview

Replace Figure with the following figure:



This figure differs from the 2020 edition, with Corrigendum 1 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

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 $0 \leq j \leq$ 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 BMCA, 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 $0 \leq j \leq$ 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 BMCA 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 masterStepsRemoved: 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 masterStepsRemoved). The data type for masterStepsRemoved is UInteger16. This variable is used by both the BMCA 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 BMCA 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 BMCA 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 BMCA 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 BMCA and the ~~explicit port state~~external port configuration option.

10.3.9.8 timeTraceable: A Boolean variable whose value is TRUE if both clockSlaveTime [i.e., the synchronized time maintained at the slave (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 BMCA 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 clockSlaveTime, 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 BMCA 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 BMCA 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 BMCA 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.

10.3.9.12 sysLeap61: A Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the ClockMaster 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 BMCA 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 ClockMaster 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 BMCA 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 ClockMaster 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 BMCA 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 ClockMaster entity of this PTP Instance is PTP (see 8.2.1) and FALSE if the timescale of the ClockMaster entity of this PTP Instance is ARB. This variable is used by both the BMCA and the ~~explicit port state~~external port configuration option.

10.3.9.16 sysTimeTraceable: A Boolean variable whose value is TRUE if both masterTime [i.e., the time maintained by the ClockMaster entity of this PTP Instance (see 10.2.4.21)] and currentUtcOffset (see 10.3.9.10), relative to the ClockMaster 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 BMCA 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 masterTime of the ClockMaster entity of this PTP Instance, i.e., the frequency of the LocalClockEntity multiplied by the most recently computed gmRateRatio by the ClockMasterSyncReceive 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 BMCA and the ~~explicit port state~~external port configuration option.

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 ClockMaster 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 BMCA 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 ClockMaster 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 BMCA, 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 BMCA 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 BMCA. 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 BMCA. This variable is used by both the BMCA 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 BMCA. 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 BMCA, 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 BMCA 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 BMCA 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 master PTP Instance for the attached gPTP communication path.

b) If infoIs is Mine, information for the PTP Port has been derived from the SlavePort for the PTP Instance (with the addition of SlavePort stepsRemoved). This includes the possibility that the SlavePort 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 BMCA, i.e., not by the ~~explicit port state~~external port configuration option.

10.3.10.5 masterPriority: The masterPriorityVector for the PTP Port. The data type for masterPriority is UInteger224 (see 10.3.4). This variable is used by both the BMCA 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 BMCA 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 BMCA 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 BMCA 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 BMCA 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 masterPriority and masterStepsRemoved information. This variable is used by both the BMCA 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 BMCA, 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 masterStepsRemoved (see 10.3.9.3) after masterStepsRemoved is updated. The data type for portStepsRemoved is UInteger16. This variable is used by both the BMCA 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 BMCA 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 BMCA, 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 masterPriority and masterStepsRemoved to portPriority and portStepsRemoved, respectively. This variable is used by both the BMCA 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 BMCA 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 BMCA 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 BMCA 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 BMCA 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 BMCA 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 BMCA 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 BMCA 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 BMCA 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 master information it knows. The state machine also updates the current best master 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 master 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 masterPriorityVector for each PTP Port.

h) Computes masterStepsRemoved, 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 masterStepsRemoved.

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 MasterPort, 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 MasterPort. In addition, updtInfo is set to TRUE if the portPriorityVector differs from the masterPriorityVector or portStepsRemoved differs from masterStepsRemoved.

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 SlavePort, 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 masterPriorityVector 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 masterPriorityVector 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 masterPriorityVector is better than the portPriorityVector, then selectedState[j]is set to MasterPort, 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 SlavePort 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 SlavePort for any PTP Port with portNumber j, j = 1, 2, ..., numberPorts, selectedState[0] is set to SlavePort.

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

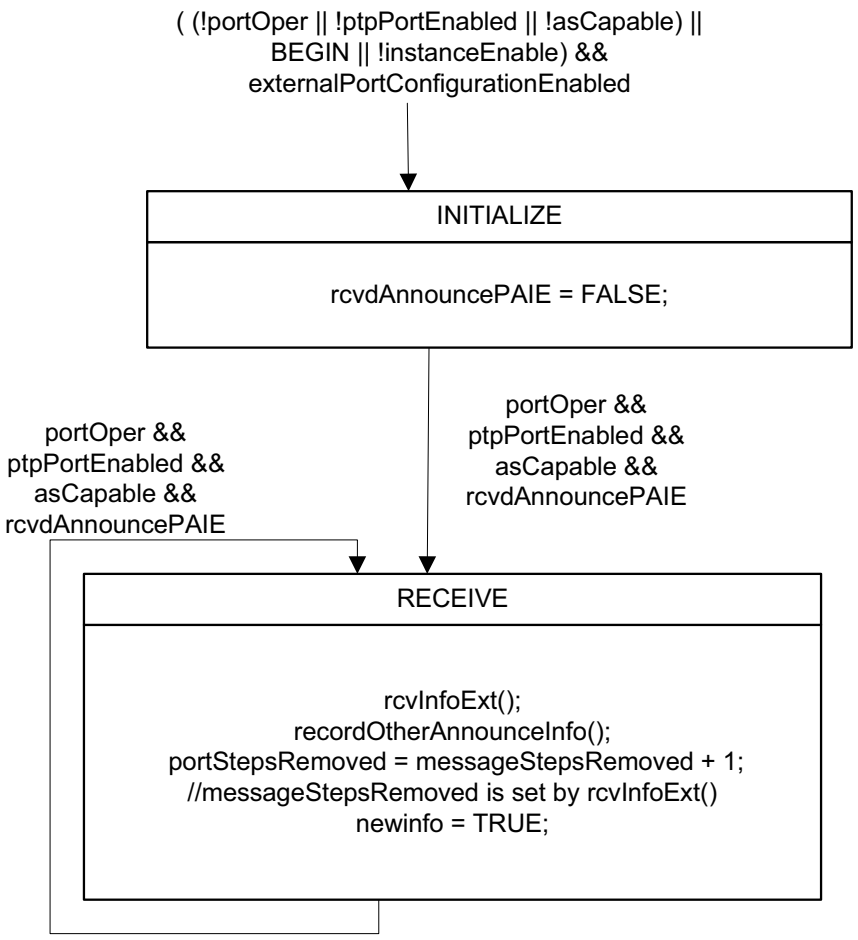
The PortStateSelection state machine shall implement the function specified by the state diagram in Figure 10-15, the functions specified in 10.3.13.1, and the relevant global variables specified in 10.2.4, 10.2.5, 10.3.9, and 10.3.10~~, and 11.2.13~~. This state machine is used only if externalPortConfigurationEnabled 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 masterPriorityVector 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..



This figure differs from the 2020 edition, with Corrigendum 1 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 SlavePort, 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 SlavePort, 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 masterStepsRemoved 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 SlavePort, then masterStepsRemoved 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 SlavePort, then masterStepsRemoved is set equal to 0.

d) Sets currentDS.stepsRemoved equal to masterStepsRemoved.

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 MasterPort, 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 SlavePort and the priority1 field of the rootSystemIdentity of the messagePriorityPAIE of the slave 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 SlavePort and the priority1 field of the rootSystemIdentity of the messagePriorityPAIE of the slave 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 SlavePort, 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 SlavePort, selectedState[0] is set to PassivePort.

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

h) Computes the gmPriorityVector as follows:

1) If selectedState[j] is set to SlavePort, the gmPriorityVector is set equal to messagePriorityPAIE for PTP Port j.

2) If selectedState[j] is *not* set to SlavePort and selectedState[k] is not equal to SlavePort 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 masterPriorityVector for PTP Port j.

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

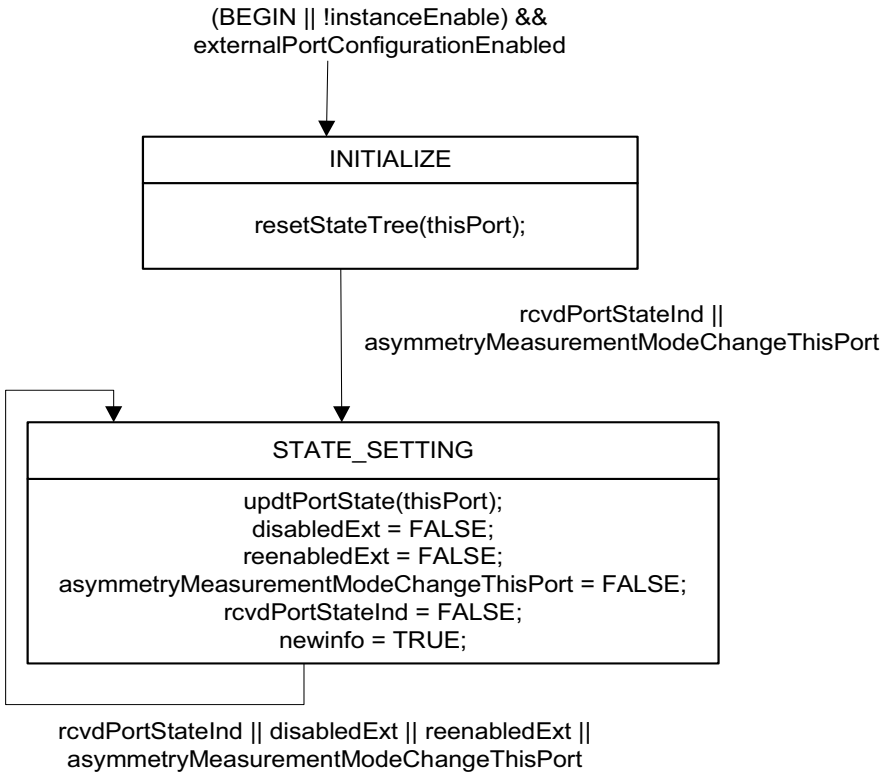
Change 10.3.15.3 as follows:

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

NOTE—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 slave port. Detecting and correcting misconfiguations is outside the scope of this standard..



This figure differs from the 2020 edition, with Corrigendum 1 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

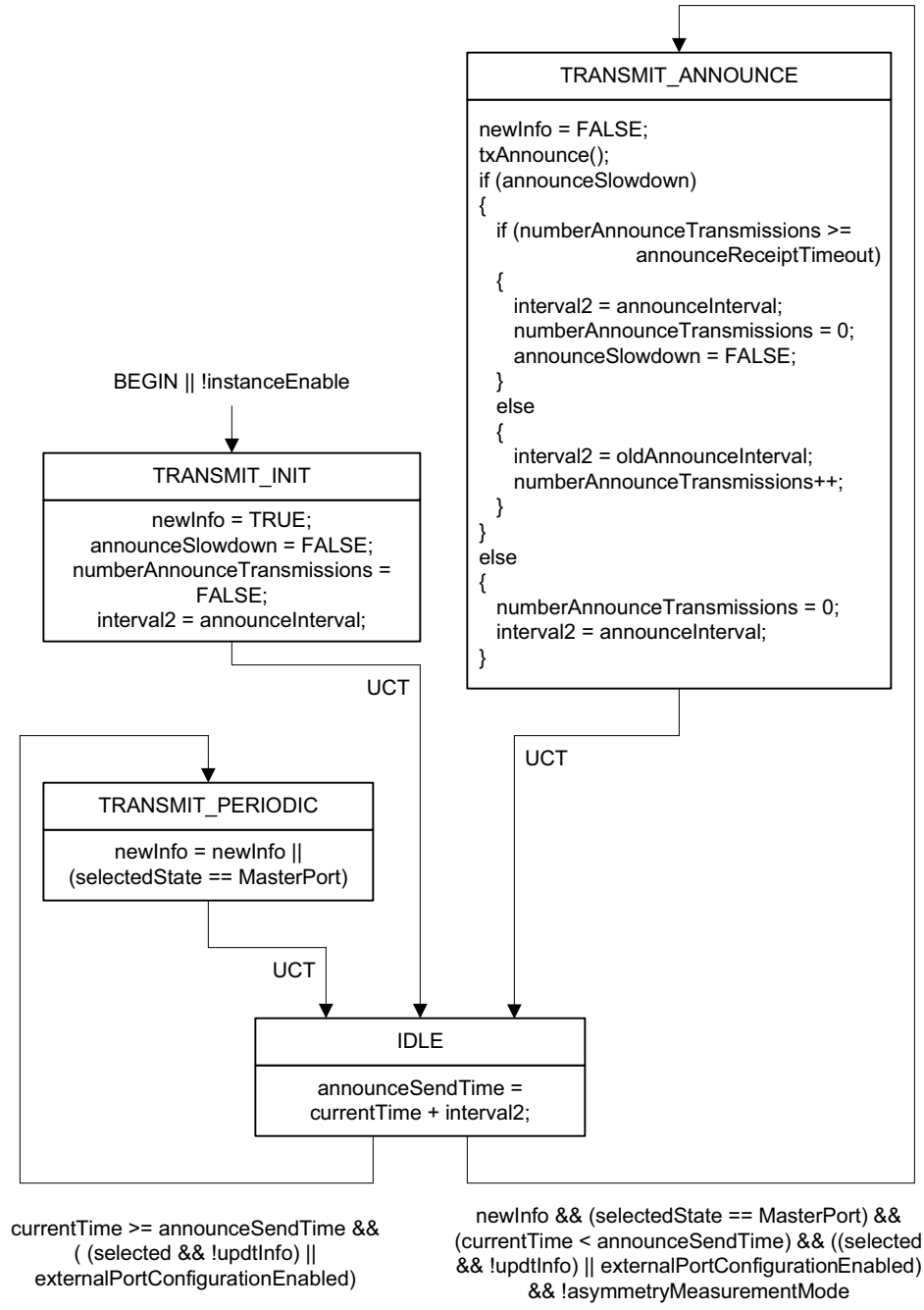
Change 10.3.16.3 as follows:

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 masterPriority and masterStepsRemoved information.

NOTE—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 with the following figure:



This figure differs from the 2020 edition, with Corrigendum 1 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). The data type for TEMP is Integer16.

10.3.17.2 State machine functions

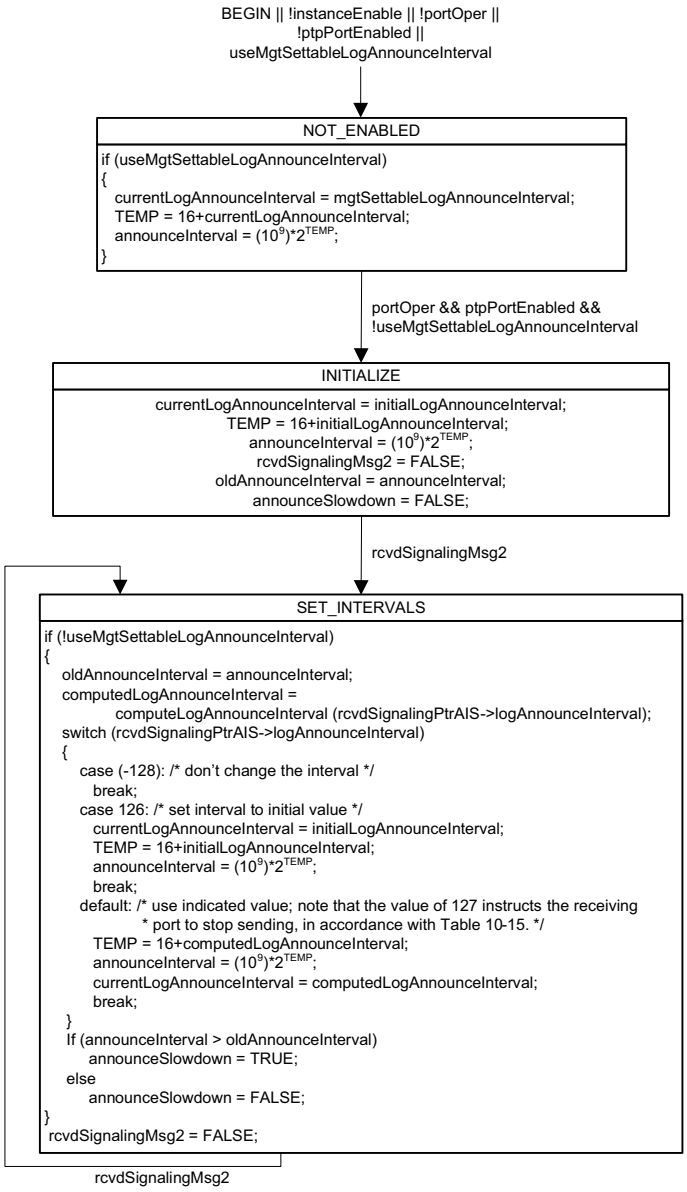
Change 10.3.17.2.2 as follows:

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 fastedslowest supported rate, even if faster than the
requested rate
                        return (logSupportedAnnounceIntervalMax);
                else
                        // Return the fastest supported rate that is still slower than
                        // the requested rate.
                        return (logSupportedClosestLongerAnnounceInterval);
        }
}
```

10.3.17.3 State diagram

Replace Figure with the following figure:



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

Delete the NOTE immediately after figure.

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:

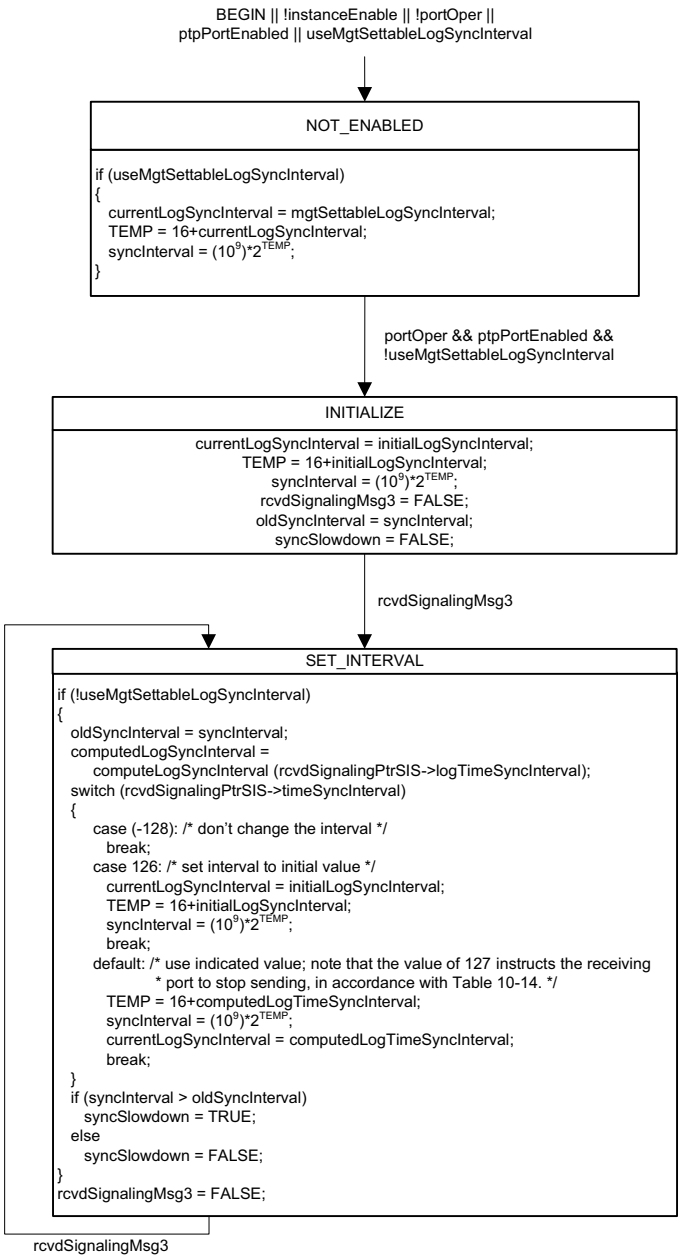
10.3.18.2.2 computeLogSyncInterval (logRequestedSyncInterval): An Integer8 function that computes and returns the logSyncInterval, based on the logRequestedSyncInterval. This function is defined as indicated below. It is defined here so that the detailed code that it invokes does not need to be placed 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 fastedslowest supported rate, even if faster than the
requested rate
                        return (logSupportedSyncIntervalMax);
                else
                        // Return the fastest supported rate that is still slower than
                        // the requested rate.
                        return (logSupportedClosestLongerSyncInterval);
        }
}
```

10.3.18.3 State diagram

Replace Figure with the following figure:

Delete the NOTE immediately after Figure .



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

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, can occur immediately, i.e., it is not necessary to wait until gPTP capability is determined by the GptpCapableTransmit and GptpCapableReceive state machines.

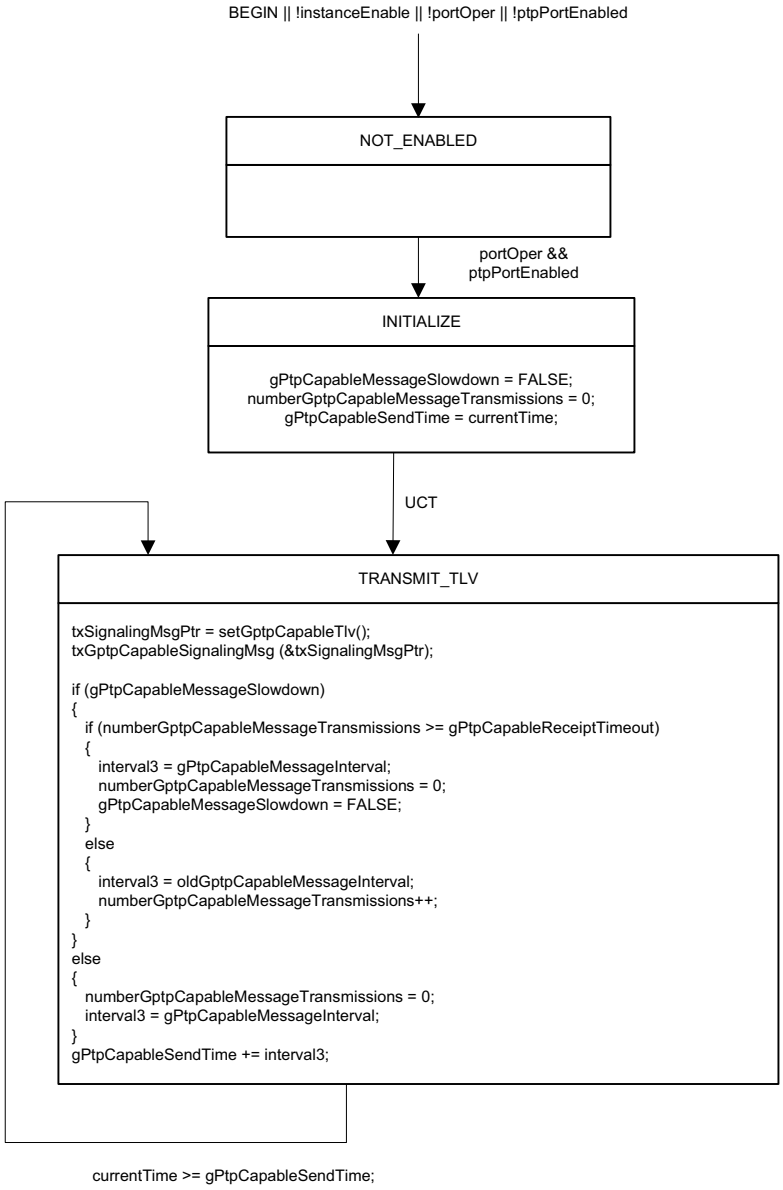
10.4.2 GptpCapableTransmit state machine

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 with the following figure:



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

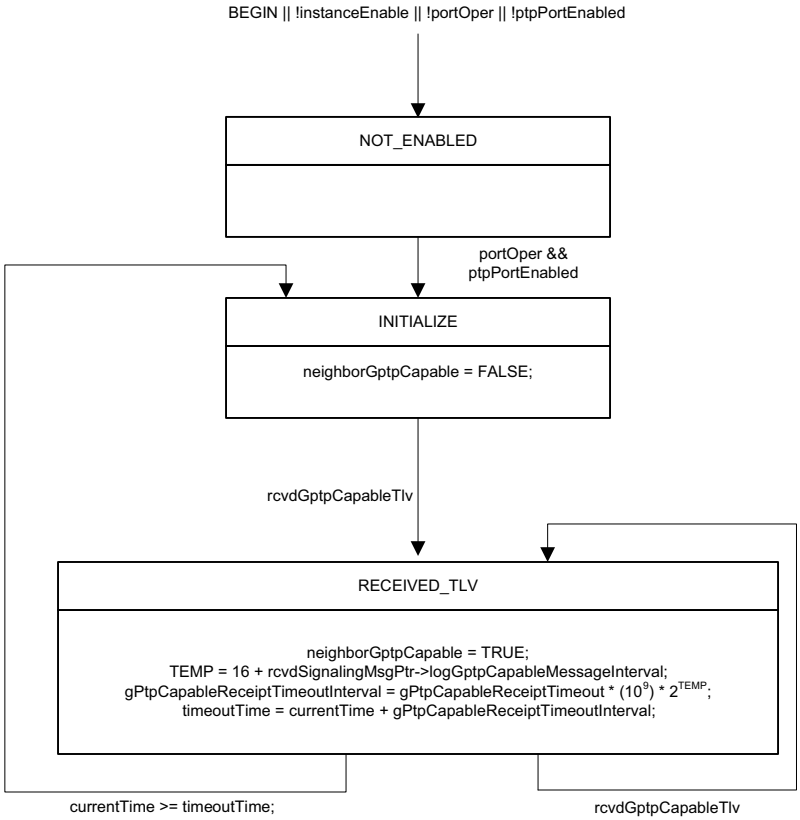
10.4.3 GptpCapableReceive sate machine

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 with the following figure:



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:

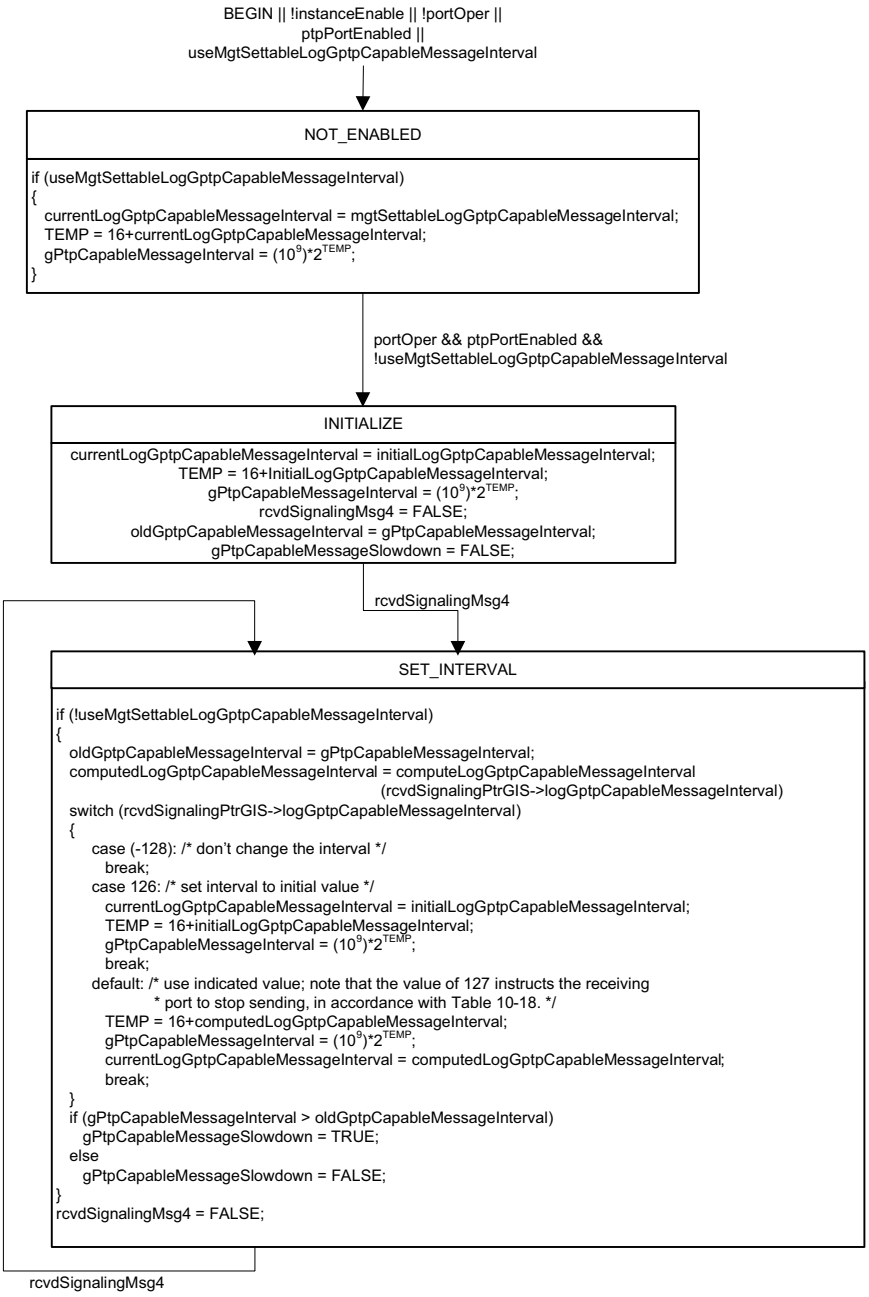
10.4.4.2.2 computeLogGptpCapableMessageInterval (logRequestedGptpCapableMessageInterval): An Integer8 function that computes and returns the logGptpCapableMessageInterval, based on the logRequestedGptpCapableMessageInterval. This function is defined as indicated below. It is defined here so that the 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 fastedslowest supported rate, even if faster than the
requested rate
                        return (logSupportedGptpCapableMessageIntervalMax);
                else
                        // Return the fastest supported rate that is still slower than
                        // the requested rate.
                        return (logSupportedClosestLongerGptpCapableMessageInterval);
        }
}
```

10.4.4.3 State diagram

Replace Figure with the following figure:

Delete the NOTE immediately after Figure .



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

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.

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	alternateMasterFlag 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

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

Octet	Bit	Message types	Name	Value
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)
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.

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

Change 10.6.4.2.2 as follows:

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.

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

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.	

~~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 $2^{\text{~~portDS.~~currentL~~l~~ogAnnounceInterval}}$ 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 $2^{\text{~~portDS.l~~currentL}\text{ogSyncInterval}}$ 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 slave port the number of sync intervals to wait without receiving synchronization information, before assuming that the master is no longer transmitting synchronization information and that the BMCA needs to be run, if appropriate. The condition of the slave 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 slave port the number of announce intervals to wait without receiving an Announce message, before assuming that the master is no longer transmitting Announce messages, and that the BMCA needs to be run, if appropriate. The condition of the slave 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 cause 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

11.2.14 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 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

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)~~

Insert g g), 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 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) neighborRateRatio (see 10.2.5.7) is computed using the following information conveyed by successive Sync and, if twoStepFlag is TRUE, Follow_Up messages:
   i) the syncEventIngressTimestamp values for the respective Sync messages.
   ii) The 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 syncEgressTimestampTimestamp of the Drift_Tracking TLV, and (C) delayAsymmetry (see 10.2.5.9) if and only if this state machine is invoked by CMLDS.

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

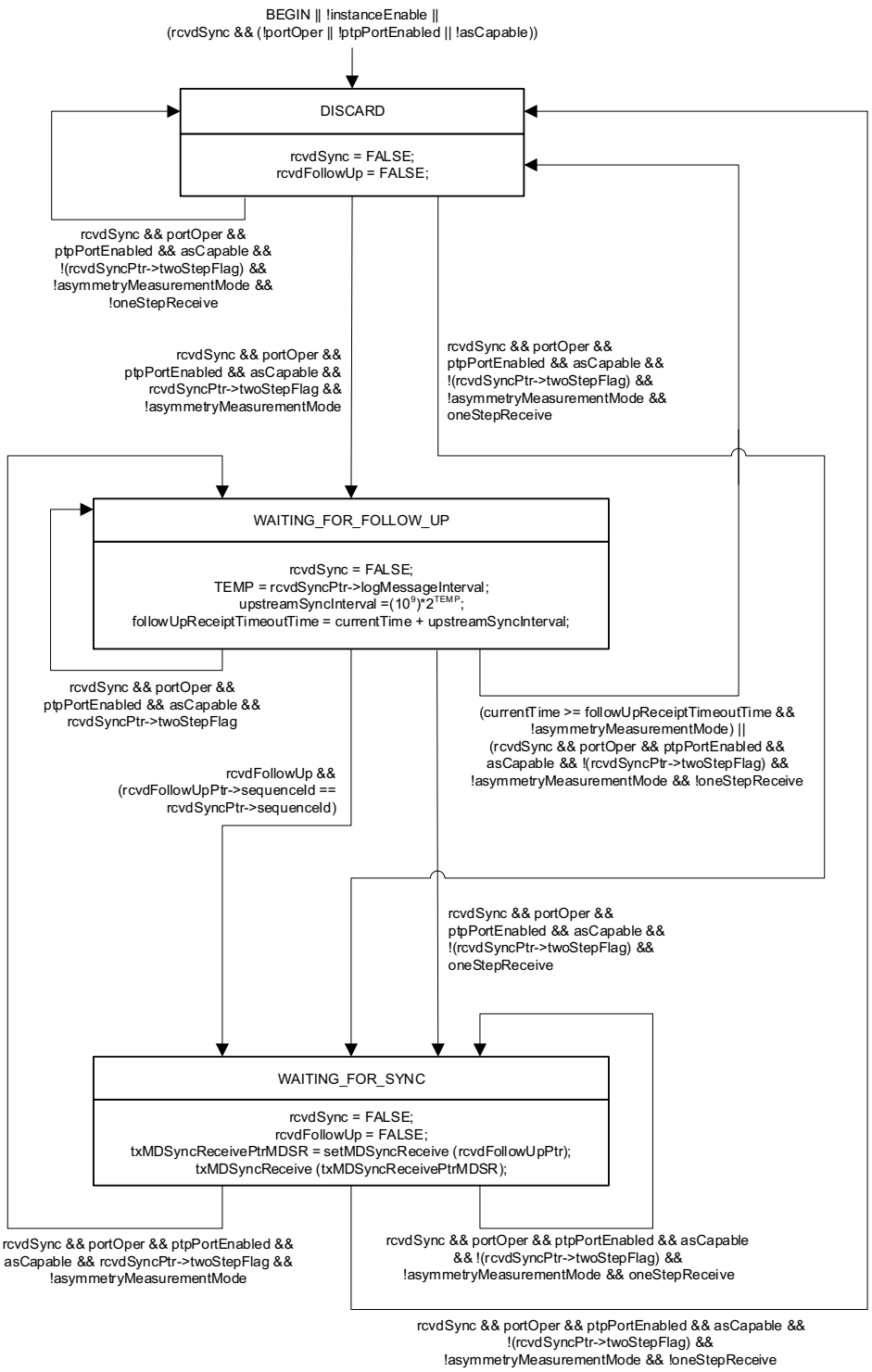
<<Editor's Note: If neighborRateRatio is computed using information in the Drift_Tracking TLV, should the computation using information in peer delay messages be disabled? If so, this needs to be added to the MDPdelayReq state machine in 11.2.19.3.3. If both are computed, there needs to be a way to distinguish which information neighborRateRatio was computed from (e.g., using different global variables for each computation).>>

h) 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.

11.2.14.3 State diagram

Replace Figure with the following figure:



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 IMDSyncSendSM 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.

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.

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:

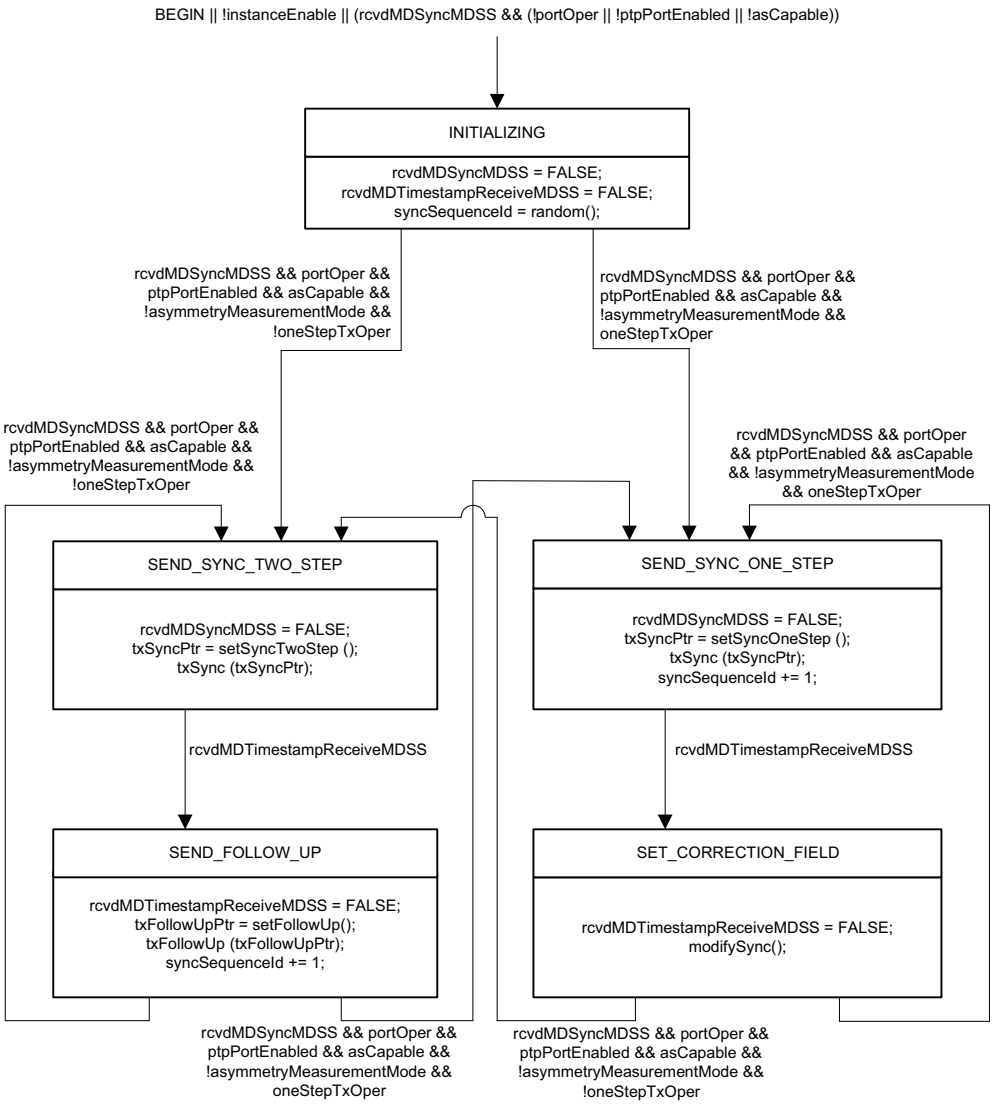
1) syncGrandmasterIdentity is set equal to the defaultDS.clockIdentity.
2) syncStepsRemoved is set equal to 0.
3) syncEgressTimestamp is set equal to the syncEventEgressTimestamp of the associated Sync message.

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.

11.2.15.3 State diagram

Replace Figure with the following figure:



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 LinkPort of a time-aware system invokes a single instance of the MDPdelayReq state machine (see 11.2.19) and the MDPdelayResp state machine (see 11.2.20). If the time-aware system implements more than one domain, these two state machines shall provide a Common Mean Link Delay Service (CMLDS), as described in this subclause, that measures mean propagation delay on the PTP Link attached to the port and the neighbor rate ratio for the port (i.e., the ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the PTP Link attached to this port, to the frequency of the LocalClock entity of this time-aware system). The CMLDS makes the mean propagation delay and neighbor rate ratio available to all active domains. If the time-aware system implements one domain, 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 also provide the CMLDS (i.e., if only the PTP Instance-specific peer delay mechanism is provided), they shall be invoked on domain 0. In other words, if the ~~domain number~~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 other PTP profiles that use the CMLDS are present on the port, the CMLDS must be provided.

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

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

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

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

NOTE 3—The above requirement ensures:

a) Backward compatibility with time-aware systems that comply with the 2011 version of this standard if domain 0 is implemented, and

b) Compatibility with time-aware systems that implement only one domain and invoke the MDPdelayReq and MDPdelayResp state machines on domain 0.

NOTE 4—In general, a port can receive:

a) Peer delay messages of the CMLDS,

b) PTP Instance-specific peer delay messages of domain 0 (with sdoId of 0x100), and

c) If there are other PTP profiles on the neighbor port that use instance-specific peer delay, peer delay messages of those profiles.

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

The CMLDS shall be enabled on a Link Port if the value of portDS.delayMechanism (see 14.8.5) is COMMON_P2P for at least one PTP Port that is enabled (i.e., for which portOper and ptpPortEnabled are both TRUE) and corresponds to the same physical port as the Link Port (see 14.1). The value of cmldsLinkPortEnabled is TRUE if the CMLDS is enabled on the Link Port and FALSE if the CMLDS is not enabled on the Link Port.

Change 11.2.17.2 as follows:

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

The MDPdelayReq state machine (see 11.2.19)~~,~~ and MDPdelay_Resp state machine (see 11.2.20)~~, and MDSyncReceiveSM state machine (see 11.2.14)~~ perform various computations of ~~mean link delay~~meanLinkDelay and ~~of the effect of delayAsymmetry~~neighborRateRatio 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 neighborRateRatio are the same for both instance-specific peer delay and CMLDS; however, the organization of the computations for instance-specific peer delay and CMLDS 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 instance-specific peer delay computations or both perform the CMLDS computations in order to obtain the correct results for meanLinkDelay and neighborRateRatio. The differences are ~~described as follows~~:

a) Both ~~I~~instance-specific peer delay and CMLDS 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 instance-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-specific peer delay sets the correctionField of a transmitted Pdelay_Req message to 0, while CMLDS sets it to –delayAsymmetry (see 11.2.19.3.1).

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

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

e) When computing ~~mean link delay~~meanLinkDelay [i.e., the quantity D in Equation (11-5), see 11.2.19.3.4]:

   4) In computing the quantity t_2, the correctionField of the Pdelay_Resp message, divided by 2^{16}, is added ~~when computing the quantity t_2~~ if instance-specific peer delay is used, while it is subtracted if CMLDS is used (see 11.2.19.3.4).

   5) In computing the quantity t_3, the quantity delayAsymmetry, divided by 2^{16}, is added if CMLDS is used.

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

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

11.2.19 MDPdelayReq state machine

Change NOTE 1 of 11.2.19.3.3 as follows:

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

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

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, and (4) delayAsymmetry if this state machine is invoked by CMLDS.

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

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

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

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

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

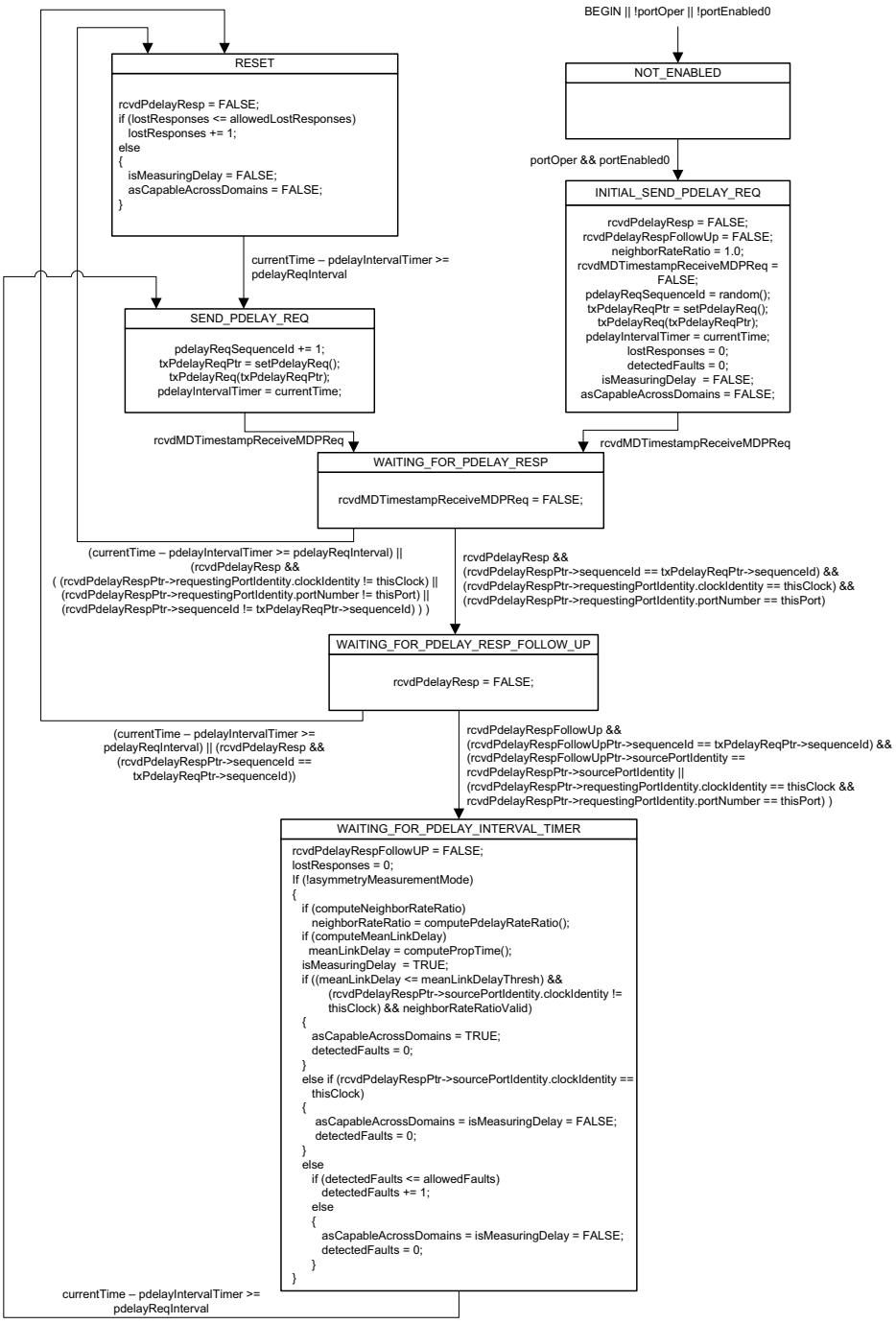
NOTE 3—In IEEE Std 1588-2019, the computation of ~~Equation (11-5)~~meanLinkDelay is organized differently from the organization used for instance-specific peer delay in the present standard. Using the definitions of t_2 and t_3 above, Equation (11-5) can be rewritten as shown in Equation (11-6).

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

where each term is expressed in units of nanoseconds as described in the definitions of t_1, t_2, t_3, and t_4 above. In IEEE Std 1588-2019, the fractional nanoseconds portion of t_2 is subtracted from the correctionField of Pdelay_Resp, rather than added as in ~~this~~ the present standard for instance-specific peer delay [see 11.2.20.3.1 d)1]; however, the correctionField of Pdelay_Resp is then subtracted in Equation (11-6) rather than added [in Eq. (11-6) for instance-specific peer delay, where s=1], and the two minus signs ~~cancel each other~~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

Replace Figure 11-9 with the following figure:



This figure differs from the 2020 edition, with Corrigendum 1 applied, of this standard in that 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 “==”).

Figure 11-9—MDPdelayReq state machine

Delete the NOTE immediately after figure.

11.2.21 LinkDelayIntervalSetting state machine

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~~slowest supported rate, even if faster than the
requested rate
                        return (logSupportedPdelayReqIntervalMax);
                else
                        // Return the fastest supported rate that is still slower than
                        // the requested rate.
                        return (logSupportedClosestLongerPdelayReqInterval);
        }
}
```

11.4 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-specific peer-to-peer delay mechanism; or For Pdelay_Resp, minus the corrections for fractional nanoseconds (see 11.2.20.3.1, Figure 11-9, and Figure 11-10), and for Pdelay_Resp_Follow_Up, the sum of the correctionField of the corresponding Pdelay_Req message and the fractional nanoseconds portion of the pdelayRespEventEgressTimestamp of the corresponding Pdelay_Resp message, if this state machine is invoked by CMLDS.
Sync (sent by a two-step PTP Port), Pdelay_Req, Announce, Signaling	The value is 0 (see 10.6.2.2.9) if the message is sent by the instance-specific peer-to-peer delay mechanism, or The value is 0 for Sync (sent by a two-step PTP Port), Announce, and Signaling, and –delayAsymmetry for Pdelay_Req (see 11.2.19.3.1), if the message is sent by CMLDS.
NOTE—IEEE Std 1588-2019 describes asymmetry corrections for the Pdelay_Req and Pdelay_Resp messages. However, the peer-to-peer delay mechanism computes the mean propagation delay. Here, where the gPTP communication path is a full-duplex point-to-point PTP Link, these corrections cancel in the mean propagation delay computation and therefore are not needed.	

11.4.3 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.

11.4.3.2 Sync message field specifications if twoStep flag is FALSE

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

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.

Table 11-10—Follow_Up message fields

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

11.4.4.2 Follow_Up message field specifications

Insert 11.4.4.2.3, and renumber subsequent subclauses as appropriate:

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 11.4.4.4.8. 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								Octets	Offset
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
organizationId								3	4
organizationSubType								3	7
syncEgressTimestamp								10	10
syncGrandmasterIdentity								8	20
syncStepsRemoved								2	28

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 (Timestamp)

The value 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 sent the Sync message.

11.4.4.4.8 syncStepsRemoved (UInteger16)

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

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 interval~~s~~, 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.

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) acceptableMasterTableDS
    7) ptpInstanceSyncDS
    8) driftTrackingDS
    9) portList[]
        i) portDS
        ii) descriptionPortDS
        iii) portStatisticsDS
        iv) acceptableMasterPortDS
        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 service~~s~~, namely the Common Mean Link Delay Service (CMLDS) and the Hot Standby Service. ~~, and the~~The corresponding sub-structure~~s~~ ~~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~~ run~~s~~ 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 17.1 and 17.3). 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 14-6), 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 therespective 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 offsetFromMaster

The value is an implementation-specific or profile standard-specific representation of the current value of the time difference between a slave and the Grandmaster Clock, as computed by the slave, 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 ClockMasterSyncOffset 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.clockClass	UInteger8	R	14.4.5.2; 7.6.2.5 of IEEE Std 1588-2019
grandmasterClockQuality.clockAccuracy	Enumeration8	R	14.4.5.3; 7.6.2.6 of IEEE Std 1588-2019
grandmasterClockQuality.offsetScaledLogVariance	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 17) is implemented and may be implemented otherwise.

14.8.2 ptpInstanceState

The value of the global variable ptpInstanceState is the state of the ptpInstance (see 17.5.1.1).

14.8.3 offsetFromMasterThreshold

The value is the threshold for offsetFromMaster (see 17.5.1.2), below which the PTP Instance is considered to be synchronized.

14.8.4 rxSlavePortSyncCountThreshold

The value of rxSlavePortSyncCountThreshold is the threshold for rxSyncCountSlaveP (see 17.5.1.4), above which the PTP Instance is considered to be synchronized.

14.8.5 threshExceedance

The value of threshExceedance (see 17.5.1.7) is the threshold for the number of consecutive exceedances of offsetFromMasterThreshold (see 17.5.1.3) by offsetFromMaster (see 17.5.1.2), at which the PTP Instance is no longer in the SYNCED state (see 17.5.3.3).

14.8.6 threshInRanges

The value of threshInRanges (17.5.1.9) is the threshold for the number of consecutive occurrences of offsetFromMaster (see 17.5.1.2) being within offsetFromMasterThreshold (see 17.5.1.3), at which the PTP Instance is placed in the SYNCED state (see 17.5.3.3).

14.8.7 ptpInstanceSyncDS table

There is one ptpInstanceSyncDS table per PTP Instance, as detailed in Table 14-6.

Table 14-6—ptpInstanceSyncDS table

Name	Data type	Operations supported[1]	References
ptpInstanceState	Enumeration8	R	14.8.2
offsetFromMasterThreshold	TimeInterval	RW	14.8.3
rxSlavePortSyncCountThreshold	UInteger32	RW	14.8.4
threshExceedance	UInteger32	RW	14.8.5
threshInRanges	UInteger32	RW	14.8.6

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

Insert 14.9 and Table xxx, 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 14-6.

Table 14-7—driftTrackingDS table

Name	Data type	Operations supported[1]	References
driftTrackingTlvSupport	Enumeration8	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 14-8~~Table 14-8. If the ~~domain number~~domainNumber is not 0, portDS.delay mechanism must not be P2P (see 11.2.17).

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

Table 14-8—delayMechanism enumeration

Delay mechanism	Value	Specification
P2P	02	The PTP Port uses the peer-to-peer delay mechanism
COMMON_P2P	03	The PTP Port uses the CMLDS
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.		

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.

14.8.55 portDS table

Insert the following item 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

[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 rxSyncCountSlaveP

This counter increments ~~every~~whenever time synchronization information is received on ~~the~~a PTP Port ~~whose~~when its port state is SlavePort. 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 SlavePort.

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
rxSyncCountSlaveP	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.

14.17 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 17.6.1.2).

14.19.5 hotStandbySystemState

The value of hotStandbySystemState is the state of the hotStandbySystem, i.e., the value of the global variable hotStandbySystemState (see 17.6.1.1).

14.19.6 hotStandbySystemLogSyncTimeThreshold1

The value of hotStandbySystemLogSyncTimeThreshold1 is the logarithm to base 2 of the time interval, in seconds, after which a hotStandbySystem in the INIT state transitions from the INIT state to the NOT_REDUNDANT state if the conditions to transition to the REDUNDANT state are not met (see 17.6.4). The value -128 means that the transition time is zero, i.e., the transition occurs immediately.

NOTE—If hotStandbySystemLogSyncTimeThreshold1 is too small, the hotStandbySystem transitions to the NOT_REDUNDANT state before both PTP Instance have had time to transition to the SYNCED state.

14.19.7 hotStandbySystemLogSyncTimeThreshold2

The value of hotStandbySystemLogSyncTimeThreshold is the logarithm to base 2 of the time interval, in seconds, after which:

a) a hotStandbySystem in the OUT_OF_SYNC state transitions to the NOT_REDUNDANT state if (i) the primary PTP Instance is in the SYNCED state and the secondary PTP Instance is in the NOT_SYNCED state, or (ii) the primary PTP Instance is in the NOT_SYNCED state and the secondary PTP Instance is in the SYNCED state (see 17.6.4);
b) a hotStandbySystem in the OUT_OF_SYNC state transitions to the REDUNDANT state if both the primary and secondary PTP Instances are in the SYNCED state (see 17.6.4) and primarySecondaryOffset (see 14.19.9) is less than primarySecondaryOffsetThresh (see 14.19.10); or
c) a hotStandbySystem in the NOT_REDUNDANT state transitions to the REDUNDANT state if both the primary and secondary PTP Instances are in the SYNCED state (see 17.6.4) and primarySecondaryOffset (see 14.19.9) is less than primarySecondaryOffsetThresh (see 14.19.10).

The value -128 means that the transition time is zero, i.e., the transition occurs immediately.

14.19.8 hotStandbySystemSplitFunctionality

If the value is TRUE, the optional split functionality (see 17.6.3.4) is used. If the value is FALSE, the optional split functionality is not used.

14.19.9 primarySecondaryOffset

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

14.19.10 primarySecondaryOffsetThresh

The threshold for hotStandbySystemDS.primarySecondaryOffset (see 14.19.9), 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.

14.19.11 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
hotStandbySystemLogSyncTimeThreshold1	Integer8	RW	14.19.6
hotStandbySystemLogSyncTimeThreshold2	Integer8	RW	14.19.6
hotStandbySystemSplitFunctionality	Boolean	RW	14.19.8
primarySecondaryOffset	ScaledNs	R	14.19.9
primarySecondaryOffsetThreshold	ScaledNs	RW	14.19.10

[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 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

[1] R= Read only access.

<Editor's Note: Contributions are requested to supply members of this data set.>>

16. Media-dependent layer specification for CSN

16.4 Path delay measurement over a CSN backbone

16.4.3 TPath delay measurement between CSN nodes

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.

Insert the following new Clause 17:

17. Hot Standby

17.1 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 Master Clock Algorithm (BMCA) 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.

17.2 Assumptions for hot standby

The following list contains detailed assumptions and goals for the design of hot standby:

a) Hot standby requires and relies on a priori selected redundant resources, such as grandmasters and paths through the network. The goal is to allow to mitigate temporary or permanent failures (i.e., loss of Sync messages) that affect either the primary or secondary domain only.
b) Restoration of failed resources is performed by an entity outside of gPTP, such as a remote management client, is based on a profile-specific rule set, or is performed by a higher-layer entity.
c) Two domains are used so that two physical grandmasters can operate simultaneously, and disjoint physical paths can be used simultaneously.
d) A grandmaster is not required to have an external source of global time (e.g., GPS or other GNSSs) that is within any specific performance requirements. As a result, hot standby cannot assume that two grandmasters are in sync with one another when their PTP Instances begin operation. For example, consider two grandmasters that begin PTP operation when they power on, with the local timestamp starting at zero. If one grandmaster in this example powers on 101 ms after the other grandmaster, the time of the two domains can differ by 101 ms. Hot standby resolves this by using the designation of a primary and secondary grandmaster (and domain). When the secondary grandmaster begins PTP operation, it does not transmit time into its domain until it is fully synchronized with the primary domain. When the secondary domain is fully synchronized with the primary domain, it starts transmitting the synchronized time. The result is that both domains contain the same time (within the performance requirements for IEEE Std 802.1AS synchronization or within the performance requirements specified by a profile standard (see 3.24)).

Since the secondary Grandmaster PTP Instance can, in general, be collocated with any PTP Instance of the primary domain, this means that arbitrary primary and secondary domain PTP Instances can have times that differ by more than the maximum time error between any two PTP Instances of a single domain.

e) Since both domains contain the same time (see item d) above) to within a specified error bound, the HotStandbySystem entity can use either domain for the time that is transferred up to the application.

f) When two physical grandmasters are configured along with two disjoint trees, hot standby provides continuous synchronization when any one grandmaster fails or any one link fails. The assumption of disjoint trees means that the PTP Links that carry time synchronization to PTP End Instances of the same time-aware system are carried over disjoint physical links.

g) The Split function provides enhanced mitigation to the baseline hot standby design of item f), such that some topologies provide continuous synchronization for more than one failure (grandmaster or link). Formal fault analysis is applied to the implemented topology in order to determine the extent of mitigation.

h) The isSynced function is intended to detect some transient faults (e.g., temporary quality problems) in addition to permanent faults (e.g., timeout).

NOTE—While the time in both domains is expected to be the same, there is no comparison performed within gPTP, and a potential deviation will not be detected within gPTP.

<<Editor's note: "This subclause is intended to be removed from the next revision of the draft unless a contribution is provided to make a major revision with proper structure and without technically unclear statements (e.g., see comment #136 against D1.0)">>

17.3 Overview

Figure 17-1 provides a model of hot standby for time synchronization.

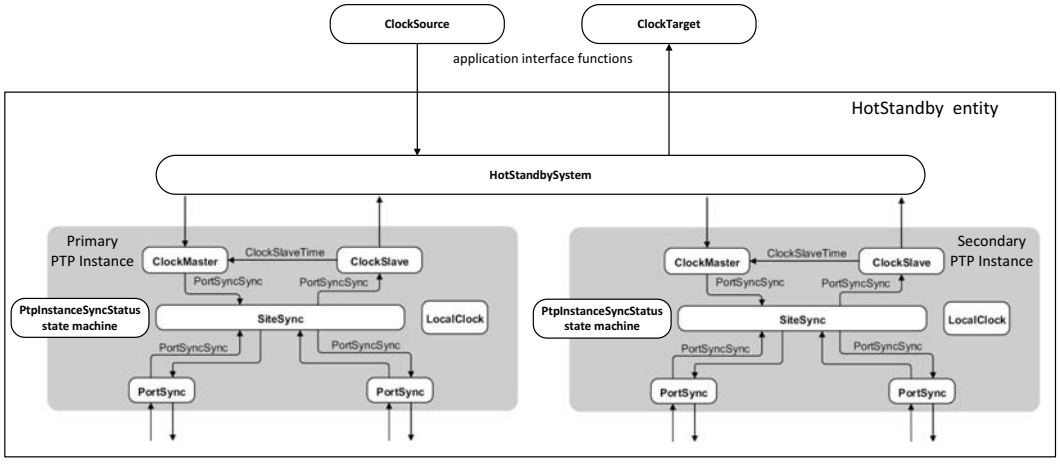


Figure 17-1—Model for hot standby entity in a time-aware system, and its interfaces to higher-layer applications

Each PTP Instance has a corresponding PtpInstanceSyncStatus state machine (see 17.5). The

PtpInstanceSyncStatus state machine monitors its PTP Instance to determine whether it is synchronized according to the requirements of the respective application or TSN profile.

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 17-1 shows two LocalClock entities.

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

17.4 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 MasterPort or PassivePort for all PTP Ports (portNumber 1 and higher); otherwise, externalPortConfiguration.desiredState is configured to SlavePort for one PTP Port, and is configured to MasterPort or PassivePort for other PTP Ports; and
d) Each PTP Instance shall have a corresponding PtpInstanceSyncStatus state machine (see 17.5).

17.5 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).

17.5.1 PtpInstanceSyncStatus state machine global variables

17.5.1.1 ptpInstanceState: State of the PTP Instance with respect to requirements of the respective application or profile standard (see 3.24). The variable is an enumeration that takes one of the following values:

a) NOT_CAPABLE: For all enabled PTP ports, asCapable is FALSE, i.e., the neighbor is not exchanging the messages that are required for conformance to this standard.

b) SYNCED: Time is synchronized to the requirements of this standard (see 17.5.3 and 17.5.4), or to the requirements specified by a profile standard (see 3.24.)

c) NOT_SYNCED: On the PTP Port in SlavePort state, the port failed to receive time synchronization event messages or is not synchronized according to the requirements of this standard (see 17.5.3 and 17.5.4), or to the requirements specified by a profile standard (see 3.24.).

d) INITIALIZING: Time synchronization does not conform to the requirements of this standard (see 17.5.3 and 17.5.4).

17.5.1.2 offsetFromMaster: The value of the managed object currentDS.offsetFromMaster (see 14.3.3).

17.5.1.3 offsetFromMasterThreshold: The value of the managed object hotstandbyDS.offsetFromMasterThreshold (see 14.8.3)

17.5.1.4 rxSyncCountSlaveP: The value of the managed object portStatisticsDS.rxSyncCountSlaveP (see 14.10.20).

17.5.1.5 rxSlavePortSyncCountThreshold: The value of the managed object hotstandbyDS.rxSlavePortSyncCountThreshold (see 14.8.4)

17.5.1.6 detectedExceedances: The current number of consecutive exceedances of offsetFromMasterThreshold (see 17.5.1.3) by offsetFromMaster (see 17.5.1.2).

17.5.1.7 threshExceedance: The threshold for the number of consecutive exceedances of offsetFromMasterThreshold (see 17.5.1.3) by offsetFromMaster (see 17.5.1.2), at which the PTP Instance is no longer in the SYNCED state (see 17.5.3.3).

17.5.1.8 detectedInRanges: The current number of consecutive occurrences of offsetFromMaster (see 17.5.1.2) being within offsetFromMasterThreshold (see 17.5.1.3).

17.5.1.9 threshInRanges: The threshold for the number of consecutive occurrences of offsetFromMaster (see 17.5.1.2) being within offsetFromMasterThreshold (see 17.5.1.3), at which the PTP Instance is placed in the SYNCED state (see 17.5.3.3).

17.5.2 PtpInstanceSyncStatus state machine local variables

17.5.2.1 lastRxSyncCountSlaveP: Holds the last value of rxSyncCountSlaveP (see 17.5.1.4) that the state machine read for the SlavePort, as part of monitoring for a sync event message timeout.

17.5.2.2 slaveP: Holds return value from slavePort() function.

17.5.3 PtpInstanceSyncStatus state machine functions

17.5.3.1 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)

}
```

17.5.3.2 slavePort(): This function returns the index of the selectedState array element (see 10.2.4.20) whose value is equal to SlavePort (see Table 10-2), i.e., the index into portDS of the port with portState of SlavePort. If the PTP Instance is grandmaster, the function returns 0.

```
slavePort()
{
        for (int i = 1; i <= numberPorts; i++)
        // see 8.6.2.8 for numberPorts
        {
          if (selectedState[i] == SlavePort)
            return (i);
        }
        return (0);
}
```

17.5.3.3 isSynced(): This function returns a boolean value that is TRUE when the PTP Instance is determined to be synchronized to the master port that is the immediate upstream port of the slave port of this PTP Instance, or the PTP Instance is a Grandmaster PTP Instance. If the PTP Instance is not synchronized, isSynced() returns FALSE. The PTP Instance is determined to be synchronized to the master port that is the immediate upstream port of the slave port of this PTP Instance when the following conditions hold:

a) The PTP Instance has a port whose PTP Port State is SlavePort;

b) offsetFromMaster (see 17.5.1.2 and 14.3.3) does not exceed the configurable threshold offsetFromMasterThreshold (see 17.5.1.3);

c) The number of Sync messages received on the slave port since that PTP Port most recently became the slave port, rxSyncCountSlaveP (see 17.5.1.4), is greater than or equal to the configurable threshold rxSlavePortSyncCountThreshold (see 17.5.1.5).

```
isSynced()
{
        slaveP = slavePort();
        if (slaveP <= 0)
                return TRUE;
        if (rxSyncCountSlaveP < rxSlavePortSyncCountThreshold)
                return FALSE;

        switch (ptpInstanceState)
        {
                case SYNCED:
                        if (offsetFromMaster > offsetFromMasterThreshold)
                                detectedExceedances += 1;
                        if (detectedExceedances >= threshExceedance)
                        {
                                detectedExceedances = threshExceedance;
                                detectedInRanges = 0;
                                return (FALSE);
                        }
                        else
                                return (TRUE);
                        break;
```

```
                case NOT_SYNCED:
                case NOT_CAPABLE:
                case INITIALIZING:



                        if (offsetFromMaster <= offsetFromMasterThreshold)
                                detectedInRanges += 1;
                        if (detectedInRanges >= threshInRanges)
                        {
                                detectedInRanges = threshInRanges;
                                detectedExceedances = 0;
                                return (TRUE);
                        }
                        else
                                return (FALSE);
                        break;
        }
}
```

<<Editor's note: It has been suggested that whether a PTP Instance is synchronized also depends on the GM attributes, e.g., clockClass, clockAccuracy, offsetScaledLogVariance, etc. Contributions are requested to further discuss the details of this and make detailed proposals. In addition, whether a PTP Instance is synchronized also depends on the receipt of Follow_Up, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages. Contributions are invited.>>

17.5.3.4 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);
}
```

17.5.4 State diagram

The PtpInstanceSyncStatus state machine shall implement the function specified by the state diagram in Figure 17-2, the variables specified in 17.5.1 and 17.5.2, and the functions specified in 17.5.3.

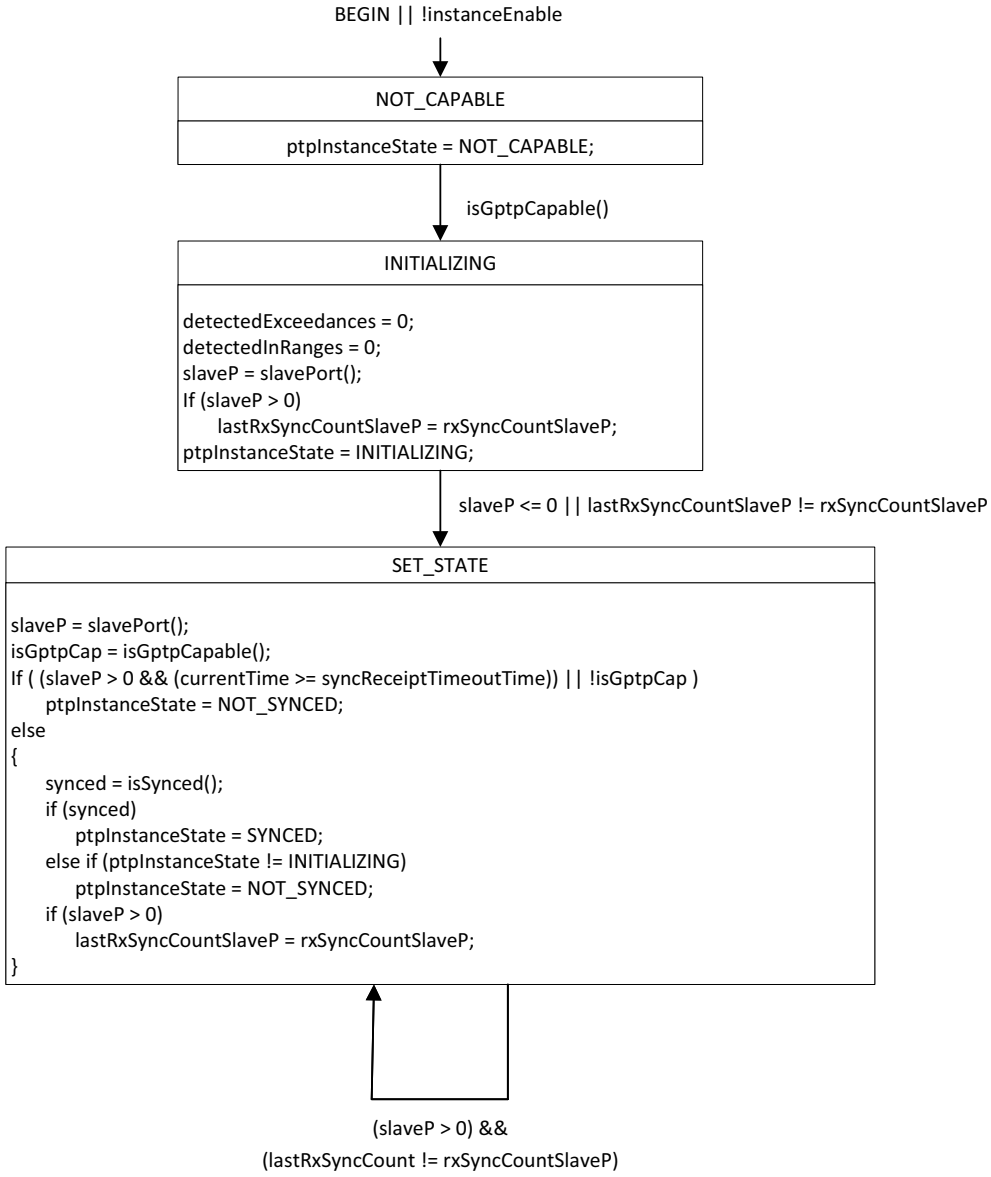


Figure 17-2—PtpInstanceSyncStatus State Machine

17.6 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.

The HotStandbySystem state machine's methodology assumes that when one of the two PTP Instances encounters a fault, time continues to be synchronized, but a higher-layer entity is notified that the fault exists (via remote management or other diagnostics protocols). If the fault in the PTP Instance corrects itself, the state machine does not attempt to restore use of that PTP Instance, and therefore time synchronization

remains non-redundant (i.e. based on only one PTP Instance). The rationale is that when transient faults occur, re-integration of a previously faulted PTP Instance is not specified by this standard. When a fault does occur, keeping the faulted PTP Instance out of service helps the higher-layer entity to locate and diagnose the cause of the fault. In the respective applications, this higher-layer entity can decide when and how to replace the faulty equipment, and then reset the network back to its original redundant state.

17.6.1 HotStandbySystem state machine global variables

17.6.1.1 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 17.5.3).

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 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).

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

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

17.6.1.4 primarySecondaryOffset: The value of the managed object hotStandbySystemDS.primarySecondaryOffset (see 14.19.9), which is the absolute value of the difference between the clockSlaveTime variables (see 10.2.4.3) of the primary and secondary PTP Instances.

17.6.2 HotStandbySystem state machine local variables

17.6.2.1 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.

17.6.2.2 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.

17.6.2.3 TEMP: A temporaray variable used to reduce clutter in the state diagram (see Figure 17-3). The data type for TEMP is Integer16.

17.6.2.4 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 ismet. The data type for notRedundantTransitionTime is UScaledNs.

17.6.2.5 primarySecondaryOffsetThresh: The value of the managed object hotStandbySystemDS.primarySecondaryOffsetThresh, which is the threshold for primarySecondaryOffset (see 17.6.1.4), above which the hotStandbySytemState transitions from REDUNDANT to NOT_REDUNDANT, or stays in NOT_REDUNDANT or OUT_OF_SYNC to REDUNDANT even if other conditions for transitioning to REDUNDANT are satisfied.

17.6.3 HotStandbySystem requirements

17.6.3.1 Primary grandmaster

When the primary PTP Instance is a Grandmaster PTP Instance (i.e., no PTP Port in SlavePort 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 ClockMaster of the primary PTP Instance (see Figure 17-1). 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 ClockSlave of the secondary PTP Instance.

17.6.3.2 Secondary grandmaster

17.6.3.2.1 HotStandbySystem in REDUNDANT state

When the secondary PTP Instance is a Grandmaster PTP Instance (i.e., no PTP Port in SlavePort 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 ClockSlave of the primary PTP Instance to the ClockMaster of the secondary PTP Instance (see Figure 17-1). By using phase from the primary PTP Instance, the secondary grandmaster can maintain continuity in the event of a fault in the primary grandmaster.

17.6.3.2.2 HotStandbySystem in NOT_REDUNDANT state

When the secondary PTP Instance is a grandmaster PTP Instance (i.e., no PTP Port in SlavePort 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 ClockMaster of the secondary PTP Instance (see Figure 17-1). If no external source of time is implemented, the ClockSource is equivalent to the LocalClock of this PTP Instance.

17.6.3.2.3 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.ptpInstanceState != SYNCED).

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

NOTE—The seconday 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.

17.6.3.3 Slave HotStandbySystem

17.6.3.3.1 General

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

17.6.3.3.2 HotStandbySystem in REDUNDANT state

When the HotStandbySystem is a slave (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 ClockSlave of the primary PTP Instance to the ClockTarget (application) if a ClockTarget is present.

17.6.3.3.3 HotStandbySystem in NOT_REDUNDANT state

17.6.3.3.3.1 Transfer of synchronized time to the ClockTarget

When the HotStandbySystem is a slave, 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 ClockSlave of the synchronized PTP Instance (i.e. ptpInstanceState == SYNCED) to the ClockTarget if a ClockTarget is present.

17.6.3.4 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 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 the secondary PTP Instance is in the NOT_SYNCED state and the primary PTP Instance is in the SYNCED state, or from the secondary PTP Instance to the primary PTP Instance when the primary PTP Instance is in the NOT_SYNCED state and the secondary PTP Instance is in the SYNCED state. The IWF provides the most recently received PortSyncSync structure of the primary/secondary PTP Instance SiteSync entity to the secondary/primary PTP Instance SiteSync entity, as follows:

a) The domainNumber is changed from the primary/secondary PTP Instance domainNumber to the secondary/primary PTP Instance domainNumber;
b) localPortNumber is changed to the portNumber of the secondary/primary PTP Instance slave port; and
c) All other members of the primary/secondary PTP Instance PortSyncSync structure are provided to the secondary/primary 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 slave port of the secondary/primary PTP Instance are ignored. The IWF provides the values of these corresponding members of the timePropertiesDS of the primary/secondary PTP Instance to the secondary/primary PTP Instance. These values are then copied to:
    i) the respective members of the timePropertiesDS of the secondary/primary PTP Instance;
    ii) the secondary/primary PTP Instance global variables leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, and frequencyTraceable, and the fields currentUtcOffset and timeSource, respectively; and

iii) the secondary/primary 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 slave port of the secondary/primary PTP Instance is ignored. The IWF provides the values of the corresponding members of the primmary/secondary 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 secondary/primary; and
ii) the gmPriorityVector of the secondary/primary.

The IWF also provides the value of the stepsRemoved member of the primary/secondary currentDS, which is copied to:

iii) the stepsRemoved member of the currentDS of the secondary/primary; and
iv) the gmPriorityVector of the secondary/primary.

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 slave port. The SiteSync entity of the secondary/primary PTP Instance transfers the timing information to the PortSync entity of each master 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 that is in the SYNCED state to the PTP Instance that is in the NOT_SYNCED state. It is not meant to cover the case where the primary and secondary PTP Instances are both in the NOT_SYNCED state.

17.6.3.5 Both PTP Instances not in the SYNCED state

When the primary and secondary PTP Instances are both not in the SYNCED state, 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 ClockSlave of the PTP Instance that was in the SYNCED state, before the HotStandbySystem left 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 drift, the PTP Instance should adjust the phase/frequency of its local time using the data stored while the PTP Instance was in the SYNCED state and the HotStandbySystem was in the REDUNDANT or NOT_REDUNDANT state.

17.6.4 State diagram

The HotStandbySystem state machine shall implement the function specified by the state diagram in Figure 17-3, the variables specified in 17.6.1 and 17.6.2, and the state requirements specified in 17.6.3.

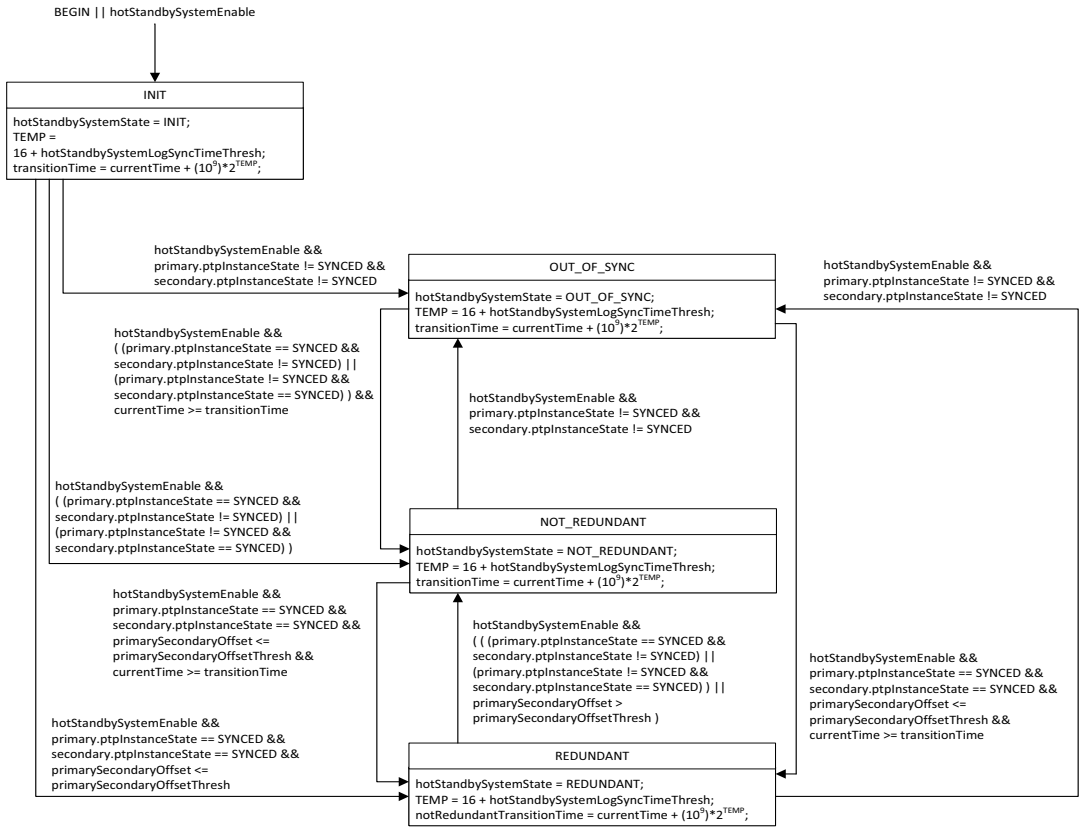


Figure 17-3—HotStandbySystem State Diagram

17.7 PrimarySecondaryOffset state machine

This state machine updates the value of the global variable primarySecondaryOffset (see 17.6.1.4) whenever a new value of clockSlaveTime (see 10.2.4.3) is computed for either the primary or secondary PTP Instance. A new value of clockSlaveTime is computed when the ClockSlaveSync when the global variable rcvdPSSyncCSS (see 10.2.4.28) or the global variable rcvdLocalClockTickCSS (see 10.2.4.29) is set to TRUE.

17.7.1 State diagram

The PrimarySecondaryOffset state machine shall implement the function specified by the state diagram in Figure 17-4..

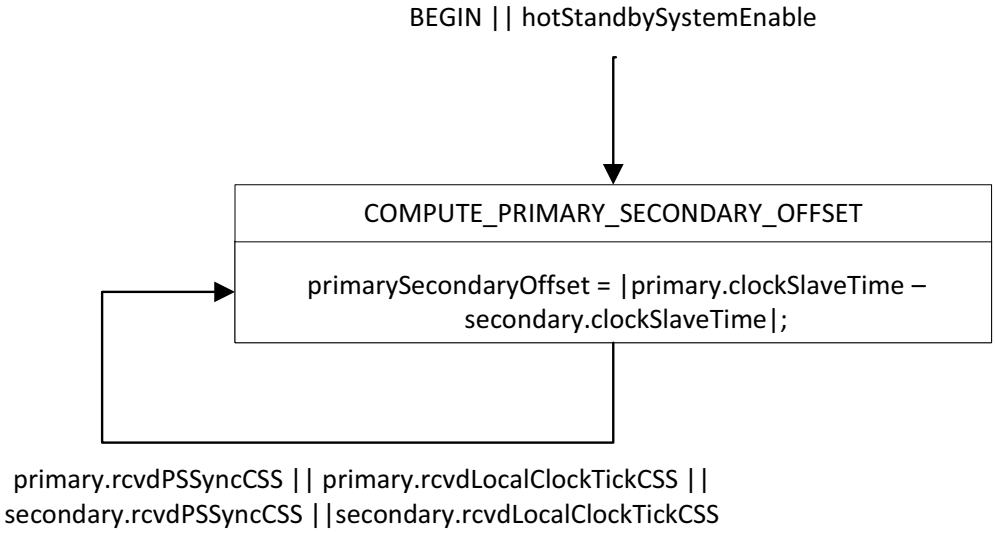


Figure 17-4—PrimarySecondaryOffset State Diagram

Annex A

(normative)

Protocol Implementation Conformance Statement (PICS) proforma[2]

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, 9.5.3.3, 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 [ ]

[2] *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 17.5 and the PtpInstanceStatusDS of 14.8?	¬HOTSTDBY:O	17.5, 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 BBest master 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 MasterPorts 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~~Master~~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 master

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

Item	Feature	Status	References	Support
MIMSTR-16	Do the MasterPorts 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 RMGT-3 in the table of A.19 as follows:

A.19 Remote management

Item	Feature	Status	References	Support
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 [ ]

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 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/).

Annex Z

COMMENTARY

<<Editor's Note: This is a temporary Annex intended to record issues/resolutions thereof as the project proceeds. It also documents the revision history. It will be removed prior to Sponsor ballot, and should be ignored for the purposs of TG/WG ballot.>>

Z.1 Revision history

Z.1.1 Revision 0.1

This is the initial version, prepared by the editor. It includes, in Clause 17, Annex Z.1 of IEC/IEEE 60802/D.1.2, which is the description of the hot standby feature and is taken from material previously contributed by Rodney Cummings. The Annex Z.1 material is modified as described in Z.1.1.1(a)(5) below. This initial version also includes maintenance items 263, 279, 281, 289, 292, 293, 294, 297, 307, and 308.

Clause 17 is a brand new clause. In Clause 17, track changes are used to show edits relative to Annex Z of 60802/D1.2, which is where this text was taken from. This is for the convenience of the reader, so that the reader can see what changes the editor has made. Various changes were necessary because the text in Annex Z of 60802 was written from the standpoint that this text would be part of 60802, not 802.1AS. It is intended that these track changes will be removed prior to SA ballot.

This initial draft is incomplete. It is planned to complete it before the initial TG ballot.

Z.1.2 Revision 0.2

This is a minor revision of the initial version, prepared by the editor. As with D0.1, it includes, in Clause 17, Annex Z.1 of IEC/IEEE 60802/D.1.2, which is the description of the hot standby feature and is taken from material previously contributed by Rodney Cummings. The Annex Z.1 material is modified as described in Z.1.2.1(a)(5) below. This initial version also includes maintenance items 263, 279, 281, 289, 292, 293, 294, 297, 307, and 308. However, the track changes of Clause 17 that were in D0.1 have been accepted, and several other minor edits were made (e.g., inserting the title line of Clause 11, replacing reference to “IEEE STD 802.1AS-2020” by “this standard”).

Z.1.2.1 Material needed for revision 0.2:

a) Description and incorporation of the split functionality, and other relevant material, described in the following references:
   1) Feng Chen, *Segment Protection*, IEEE 802.1 presentation, March 2014 (available at https://www.ieee802.org/1/files/public/docs2014/as-chen-segment-protection-0314-v01.pdf).
   2) Rodney Cummings, *802.1AS Hot Standby Amendment: Scope Discussion*, Revision 1, IEEE 802.1 presentation, January 2020 (available at https://www.ieee802.org/1/files/public/docs2020/new-cummings-as-hot-standby-scope-0120-v02.pdf).
   3) Mike Potts, *IEEE 802.1AS Addendum for “Hot Standby” with multiple domain definition*, IEEE 802.1 presentation, 18 February 2020 (available at https://www.ieee802.org/1/files/public/docs2020/dm-potts-as-hot-standby-multiple-domains-0220-v01.pdf)
   4) Rodney Cummings, *802.1AS Hot Standby Amendment: 4 Domains*, Revision 3, IEEE 802.1 presentation, March 2020 (available at https://www.ieee802.org/1/files/public/docs2020/dm-cummings-as-4-domains-0320-v04.pdf).

5) Geoffrey M. Garner, *Initial Plans and Assumptions for 802.1ASdm*, IEEE 802.1 presentation, November 2, 2020 (available at https://www.ieee802.org/1/files/public/docs2020/dm-garner-plans-1120-v00.pdf).

6) Max Turner, *802.1ASdm - Redundancy*, IEEE 802.1 presentation, July 2021 (available at https://www.ieee802.org/1/files/public/docs2021/dm-turner-ASredundancy-0721-v02.pdf).

b) Definition and specification of the NOT_QUALITY state, along with any related functions.

c) Addressing of all Editor's Notes.

d) The editor must review the full Clause for completeness, consistency, and correctness.

e) The following maintenance items must still be added: 278, 280, 283. Details are as follows:

1) Discussion is needed for maintenance item #278. This maintenance item is to make the PICS and Clause 5 conformance statements consistent with respect to the interval setting state machines. These state machines were mandatory in 802.1AS-2011, but in 802.1AS-2020 the PICS and clause 5 are inconsistent on whether they are mandatory or optional. In initial discussion of the maintenance item, several participants indicated that the state machines are not needed by some profile standards (see 3.24) and applications, and that those applications and profiles that don't need them should not be required to include them, as the cost will be greater. The feature apparently is mainly needed for AVB. It was suggested that the feature could be optional in 802.1AS and mandatory in 802.1BA. However, during the presentation of *Insights and observations on TSN applied across ecosystems* on Monday, March 8, 2021 (available at https://www.ieee802.org/1/files/public/docs2021/new-Kaltheuner-et-al-TSN-across-ecosystems-0321-v01.pdf), it was indicated that if a feature that was previously mandatory, and needed by AVB, is now made optional, vendors may choose not to provide the feature in the future because the AVB market is smaller than, e.g., Industrial and Automotive. It also was mentioned, in the 802.1 maintance meeting ion Tuesday, March 9, 2021, that there is some chance that 802.1BA could be withdrawn if an editor for a revision is not found quickly because 802.1BA is approaching the deadline for re-affirmation. In any case, discussion is needed on how to proceed on this maintanance item.

2) Maintenance item 280 is to change "time-aware Bridge" to "PTP Relay Instance" and "time-aware end station" to "PTP End Instance" in a MIB description field on p.363 of 802.1AS-2020.

3) Maintenance item 283 is the following:

"IEEE Std 802.1AS-2020 uses both "domain number" and "domainNumber". The former is the concept, while the latter is the attribute, parameter, or name of the respective field. There are 38 instances of the former, and 66 instances of the latter. In contrast, IEEE Std 1588-2019 uses "domainNumber" almost exclusively; there is one instance of "domain numbers" and 106 instances of "domainNumber". It appears that at least some of the instances of "domain number" in 802.1AS-2020 should be "domainNumber. It would be useful to go through the document, check the usage of "domain number" and "domainNumber", and change one to the other where necessary."

The editor has reviewed the instances of "domain number" and "domainNumber." In the opinion of the editor, the instances of "domainNumber" are correct and need not be changed. Also, it is the editor's opinion that all the instances of "domain number" can be changed to "domainNumber" because all these uses pertain to the attribute "domainNumber" rather than the general concept "domain number." Can this change be made by a single editing instruction, or must there be a separate editing instruction (and change indicated by track changes) for each clause or subclause where "domain number" is used? If a single editing instruction is sufficient, where should it go? Note that two of the instances of "domain number" are in the MIB, on pp. 305 and 320 (in description fields). If a single editing instruction is used, how are the changes in the MIB handled (because, if the MIB is changed, the MIB revision and date also must change,

as well as the MIB file that will be attached). Note also that Maintenance item 280 also changes the MIB.

Z.1.3 Revision 0.3

This revision version, prepared by the editor, addressess all comments against D0.2, except the following:

a) Comment 95: Improvements to the formatting of Figure 17-1, Figure 17-2, and Figure 17-3, e.g., fixing the font size and making each figure fill the page as much as possible, are deferred to the next draft. However, all technical changes to the figures, required by comments against D0.2, are made.
b) Comment 49: Updates to the MIB, and Clause 15, are deferred to working group ballot, when updates to managed objects will have been made. This is done so that all MIB updates can be made at once. Note that changing “domain number” to “domainNumber” in two places in the MIB also are deferred. However, “domain number” is changed to “domainNumber” in all other places, in accordance with comment 50.
c) Comment 148: In deferring MIB changes, the part of comment 148 that requires MIB changes, i.e., addressing maintenance item 280 and the part of 283 that affects the MIB, is deferred.

Except for the maintenance items indicated above, all maintenance items listed in the frontmatter are included.

Z.1.3.1 Requests for contributions after D0.3

Contributions are requested on the following items:

a) Contributions are needed on what new managed objects related to hot standby should be defined (see the editor’s note at the end of 17.6.1.3).
b) Contributions are needed on how the split functionality (see Z.1.2.1) or equivalent should operate, and how it should be incorporated into the draft (see also the editor’s note at the beginning of 17.6).
c) It has been suggested that whether a PTP Instance is synchronized also depends on the GM attributes, e.g., clockClass, clockAccuracy, offsetScaledLogVariance, etc. Contributions are requested to further discuss the details of this and make detailed proposals (see the editor’s note at the end of 17.5.3.3).

Z.1.4 Revision 0.4

This revision version, prepared by the editor, addressess all comments against D0.3. The following comments against Revision D0.2 are still unaddressed:

a) Comment 95: Improvements to the formatting of Figure 17-1, Figure 17-2, and Figure 17-3, e.g., fixing the font size and making each figure fill the page as much as possible, are deferred to the next draft. However, all technical changes to the figures, required by comments against D0.2, are made.
b) Comment 49: Updates to the MIB, and Clause 15, are deferred to working group ballot, when updates to managed objects will have been made. This is done so that all MIB updates can be made at once. Note that changing “domain number” to “domainNumber” in two places in the MIB also are deferred. However, “domain number” is changed to “domainNumber” in all other places, in accordance with comment 50.
c) Comment 148: In deferring MIB changes, the part of comment 148 that requires MIB changes, i.e., addressing maintenance item 280 and the part of 283 that affects the MIB, is deferred.

Except for the maintenance items indicated above, all maintenance items listed in the frontmatter are included.

Z.1.4.1 Requests for contributions after D0.4

Contributions are requested on the following items:

a) Contributions are needed on what additional new managed objects related to hot standby should be defined (see the editor's note at the end of 17.6.1.3).

b) Contributions are requested on the split functionality (see 17.6.3.4)

c) The split functionality and, in general, the transfer of information from the primary and to the secondary PTP Instance do not address the transfer of time properties information (e.g., offsetFromMaster, leap59, leap61, etc.), nor of ClockQuality information). Should this be considered? (For that matter, the transfer of time properties and ClockQuality information from the ClockSource to the ClockMaster, either in Clause 17 or Clause 9, is not considered.) See the editor's note at the end of 17.6.3.4.

d) It has been suggested that whether a PTP Instance is synchronized also depends on the GM attributes, e.g., clockClass, clockAccuracy, offsetScaledLogVariance, etc. Contributions are requested to further discuss the details of this and make detailed proposals (see the editor's note at the end of 17.5.3.3).

Z.1.5 Revision 0.5

This revision version, prepared by the editor, is the same as D0.4, except that track changes in Clause 17 have been accepted, the present subclause (Annex 1.5) has been added, and the draft number has been updated. The track changes were included in D0.4 for the convenience of readers. Since Clause 17 is a new clause, the only editing instruction for Clause 17 is the instruction at the very beginning to add the clause. Therefore, there was no danger of track changes in Clause 17 being confused with amendment editing instructions. However, D0.5 will be balloted (D0.4 was not balloted), and the practice is to not include track changes in balloted drafts; instead, changed text is indicated by change bars in the left margin. D0.5 does include change bars (as did D0.4, including Clause 17).

Z.1.6 Revision 0.6

This revision, prepared by the editor, addresses all comments against the current and previous revisions except for:

a) Comment 95 against D0.2: Improvements to the formatting of Figure 17-1, Figure 17-2, and Figure 17-3, e.g., fixing the font size and making each figure fill the page as much as possible, are deferred to the next draft. However, all technical changes to the figures, required by comments against D0.2, are made.

b) All comments related to the MIB. It was decided at the July 11, 2020 TSN TG meeting that revised MIB will not be included in P802.1ASdm. After repeated requests for a volunteer MIB editor, to which there were no responses, it was decided that an updated MIB will not be necessary in view of the fact that a YANG module will be included. As a result, Clause 15 has been removed from the draft.

c) Comment 56 against D0.5. This comment indicates that the split functionality of 17.6.3.3.2.2 effectively duplicates the functionality of 17.6.3.2 Secondary grandmaster - both ensure (by different internal means) that the master-ports of both the primary and secondary PTP Instance communicate effectively the same time synchronization information to downstream slave ports.

   Note that it is true that both the split functionality (17.6.3.3.2.2) and the Secondary GM section (17.6.3.2) both transfer timing from the primary to secondary PTP Instance. However, the two cases

are different. The split functionality is used when the secondary PTP Instance is not GM, and the transfer of the PortSyncSync structure, with translation, from primary to secondary allows the secondary domain state machines to work exactly as they do now in 802.1AS-2020. In contrast, in 17.6.3.2 the secondary is GM and, consequently, is expecting the information to come from the ClockMaster entity; that is why in this case the transfer occurs via the primary ClockSlave and secondary ClockMaster.

The currently formulation was chosen so that the existing 802.1AS-2020 state machines could be used as is, without modification. If a different formulation is chosen (e.g., a formulation that does not use the split functionality, does not use the secondary GM, or something entirely different, it needs to be determined exactly what changes are needed for the existing 802.1AS-2020 state machines.

More discussion of this topic is needed. Comments and/or contributions are requested.

Z.1.7 Revision 0.7

This revision version, prepared by the editor, is the same as D0.6, except that track changes in Clause 17 have been accepted, the present subclause (Z.1.7) has been added, and the draft number has been updated. The track changes were included in D0.6 for the convenience of readers. Since Clause 17 is a new clause, the only editing instruction for Clause 17 is the instruction at the very beginning to add the clause. Therefore, there was no danger of track changes in Clause 17 being confused with amendment editing instructions. However, D0.7 will be balloted (D0.6 was not balloted), and the practice is to not include track changes in balloted drafts; instead, changed text is indicated by change bars in the left margin. D0.7 does include change bars (as did D0.6, including Clause 17).

Z.1.8 Revision 0.8

This revision, prepared by the editor, addresses all comments against the current and previous revisions except for items a), b), and c) in Z.1.6 above. Note that itemb) is no longer relevant as it has been decided not to include a MIB, and item c) is superseded by TG ballot comments against 0.7 related to the split functionality.

This revision has all changes to Clause 17 shown via track changes. This was possible because Clause 17 is a brand new clause and has no editing instructions other than to add the clause. This was done for the convenience of readers. Revision 0.8 will not be balloted; instead, a subsequent revision will be balloted.

All comments against D0.7 have been addressed, except for comment #69, which relates to adding a cross-reference to the cover page so that the title of the standard appears in the PDF bookmarks. This comment will be addressed in the next draft. Note that the resolutions to comments 3, 27, and 92 resulted in major changes to the HotStandbySystem state machine, and the resolution to comments 13, 51, 53, 54, and 55 resulted in major changes to the PtpInstanceSyncStatus state machine and addition of the ptp.

Z.1.9 Revision 1.0

This revision version, prepared by the editor, is the same as D0.8, except that track changes in Clause 17 have been accepted, the present subclause (Z.1.9) has been added, and the draft number has been updated. The track changes were included in D0.8 for the convenience of readers. Since Clause 17 is a new clause, the only editing instruction for Clause 17 is the instruction at the very beginning to add the clause. Therefore, there was no danger of track changes in Clause 17 being confused with amendment editing instructions. However, D1.0 will be balloted (D0.8 was not balloted), and the practice is to not include track

changes in balloted drafts; instead, changed text is indicated by change bars in the left margin. D1.0 does include change bars (as did D0.8, including Clause 17).

Z.1.10 Revision 1.1

This version, prepared by the editor, incorporates all comments against D1.0, but with the following exceptions:

a) For comments 1 and 141, the approval date of and the link to the modified PAR cannot be inserted until the PAR approved by the IEEE SASB and placed on the IEEE SA web site. As of the date of completion of this draft, the information provided to the editor indicates that the modified PAR has been approved by NesCom but is awaiting approval by the SASB via letter ballot.

b) Comments 3 and 144 ask that 802.1ASdr be incorporated into the draft. This will be done after P802.1ASdr completes SA ballot. In addition, comments 58 and 59 requests changes that depend on P802.1ASdr; these changes will be made after P802.1ASdr completes SA ballot.

c) Comments 47 and 48 ask that permission be obtained for two definitions that were copied from IEEE Std 1588-2019. It is the understanding of the editor that respective forms related to this must be submitted to the IEEE SA by the 802.1 Chair. As of the date of completion of this draft, it is expected that the forms will be submitted. The editor has not yet been informed what additions, if any, are needed for the draft; however, any needed additions will be made when the editor is informed of this.

In addition, comment 95 against D0.2, Improvements to the formatting of Figure 17-1, Figure 17-2, and Figure 17-3, e.g., fixing the font size and making each figure fill the page as much as possible, is deferred to the next draft. Finally, a YANG module is not yet available; however, it is expected that this will be available, and will be incorporated in the draft along with any other material needed related to P802.1ASdn.