IEEE P802.1AS-Rev/D5.0

Draft Standard for Local and Metropolitan Area Networks—

Timing and Synchronization for Time-Sensitive Applications ~~in Bridged Local Area Networks—Amendment 1: Enhancements and performance improvements~~

Sponsor
LAN/MAN Standards Committee of the IEEE Computer Society

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

Abstract: ~~This corrigendum to IEEE Std 802.1AS provides technical and editorial corrections.~~This standard defines a protocol and procedures for the transport of timing over ~~bridged and virtual bridged~~ local area networks. It includes the transport of synchronized time, the selection of the timing source (i.e., best master), and the indication of the occurrence and magnitude of timing impairments (i.e., phase and frequency discontinuities).
Keywords: best master, frequency offset, grandmaster, IEEE 802.1AS, phase offset, synchronization, syntonization, time-aware system.

DRAFT STATUS:

This draft was prepared by the editor for the ~~November 2016~~July 2017 802.1 TSN TG meeting. It will also be the first WG ballot draft (it is expected that the ballot will be in progress at the time of the July 2017 meeting). Th~~is~~e draft is prepared in Revision format, as the P802.1AS-Rev PAR was approved February 16, 2015. Track changes show changes relative to 802.1AS-2011 with 802.1AS-Cor1 folded in. Change bars are ~~relative to D4.0~~5. ~~(Change bars in D4.0 are relative to D3.0,~~ which was the previous balloted draft~~)~~.

Items still to be incorported into the draft are listed below, on pp. (ii)-(v).

The items listed below are comments that must still be incorporated into the draft. These comments were made either against D4.5, or against eariler drafts and referred to by comments against D4.5 or frontmatter in D4.5.

1) Comment 49 against D2.0, comment 39 against D3.0, and comment 20 against D4.5, which indicatge that it must be ensured that all the normative requirements are given in clause 5, and that all the clause 5 requirements are referenced in the PICS (this is also mentioned in comment 11 against D4.5)(.
2) Comment 21 against D4.5, which indicates that the MIB must be updated (including multiple domain/multiple PTP instance aspects, which is also indicated in comment 46 against D3.0).
3) Part of comment 12 against D4.5, which indicates that global variables that are common across PTP instances (i.e., across domains) must be moved to their own subclause (rather than being interspersed with instance-specific global variables). This must be done for both per port and per time-aware system global variables.
4) Part of comment 12 against D4.5, which indicates that the variables computeNeighborRateRatio and computeNeighborPropDelay must be set and initialized, for the case where the Pdelay interval is set via management rather than by the link delay interval setting state machine (see 11.2.20 and Figure 11-11.
5) Comments 136, 137, and 138 against D4.5, which indicate that certain aspects of the MTIE masks and related tables in Annex B must be supplied (i.e., are missing).
6) Comments 54 and 55 against D4.5, which will be addressed by updating the material on the formation of clockIdentities to be consistent with the latest revised RAC tutorial (the updating will follow updating of analogous material in IEEE P1588-Rev).
7) Comments 107 - 109, 112, which relate to separating 10.3.1 into 2 subclauses, one that describes BMCA and one that describes external port state configuration. Along with this, comment 122, which reflects a comment against an earlier draft that asked that material analogous to the 10.3.7 (with a figure analogous to Figure 10-11) be prepared for external port state configuration (but the new material should be in its own section).
8) Comments 76, 79, 86, 88, 89, 103, 104, 113, 114, 116, 120 against D4.5, which ask that various figures be modified (mainly resized) to make them easier to read.
9) Comments 80, 82, 83, 87, 118, 119 against D4.5, which ask that statements in state machines that involve exponents be modified so that the superscripted text is larger and easier to read.
10) Comment 85 against D4.5 against D4.5, which asks that various tables where one border has a thicker line than the other borders be modified so that all the line thicknesses are the same.

The title of this draft standard is in accordance with the Revision PAR. The above abstract was copied from the abstract in IEEE Std 802.1AS-2011.

Except for this front matter and Annex Z, diffmarks (i.e., strikethroughs and underlines, in different colors) indicate changes relative to IEEE Std 802.1AS-2011, with the changes of IEEE Std 802.1AS-2011/Cor 1-2013 applied. For convenience, an informal, clean copy of IEEE Std 802.1AS-2011, with the changes of IEEE Std 802.1AS-2011/Cor 1-2013 applied is available at:

http://www.ieee802.org/1/files/private/asbt-drafts/2011-with-cor1/802-1AS-2011-with-cor1-clean.pdf,

and an informal, diffmarked copy showing the changes of IEEE Std 802.1AS-2011/Cor 1-2013 applied to IEEE Std 802.1AS-2011 is available at:

http://www.ieee802.org/1/files/private/asbt-drafts/2011-with-cor1/802-1AS-2011-with-cor1-diff.pdf.

For Clauses 2 - 16 and Annexes A - H, change bars (i.e., in the left margins) show changes relative to P802.1AS-Rev/D4.5.

The base standard to which this is an amendment, IEEE Std 802.1AS-2011,and the corrigendum, IEEE Std 802.1AS-2011/Cor 1-2013, can be obtained free of charge from:

http://standards.ieee.org/about/get/

Introduction to IEEE P802.1AS~~bt~~-Rev/D~~0~~5.~~1~~0™

(This introduction is not part of P802.1AS~~bt~~-Rev/D5.0~~0.1~~, Draft Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications~~ in Bridged Local Area Networks—Amendment 1: Enhancements and performance improvements~~.)

This standard contains state-of-the-art material. The area covered by this standard is undergoing evolution. Revisions are anticipated within the next few years to clarify existing material, to correct possible errors, and to incorporate new related material. Information on the current revision state of this and other IEEE 802 standards may be obtained from

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Participants

The following is a list of participants in the ~~Interworking~~Time-Sensitive Networking activities of the IEEE 802.1 Working Group during the development of P802.1AS-Rev~~Cor-1~~. Voting members at the time of publication are marked with an asterisk (*).

When the IEEE 802.1 Working Group approved IEEE Std 802.1~~-Cor-1~~AS-Rev, it had the following membership:

The following members of the balloting committee voted on P802.1AS-Rev~~-Cor-1~~. Balloters may have voted for approval, disapproval, or abstention.

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

Contents

List of figures

List of tables

IEEE Standard for
Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications~~ in Bridged Local Area Networks~~

1. Overview

1.1 Scope

This standard specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications, such as audio, video, and ~~video~~time-sensitive control, across ~~bridged and virtual bridged local area~~ networks ~~consisting of local area network (LAN) media where the transmission delays are fixed and symmetrical~~; for example, IEEE 802~~.3™~~ ~~full-duplex~~and similar ~~links~~media. This includes the maintenance of ~~synchronized time~~synchronized time during normal operation and following addition, removal, or failure of network components and network reconfiguration. It specifies the use of IEEE 1588™ specifications where applicable in the context of IEEE Std 802~~.1D™-2004 and IEEE Std 802~~.1Q~~™-2005~~.[1] Synchronization to an externally provided timing signal (e.g., a recognized timing standard such as UTC or TAI) is not part of this standard but is not precluded.

[1]Information on references can be found in Clause 2.

1.2 Purpose

This standard enables ~~stations attached to bridged LANs~~systems to meet the respective jitter, wander, and ~~time synchronization~~time-synchronization requirements for time-sensitive applications. This includes applications that involve multiple streams delivered to multiple endpoints. To facilitate the widespread use of ~~bridged LANs~~packet~~ ~~networks for these applications, synchronization information is one of the components needed at each network element where time-sensitive application data are mapped or demapped or a time-sensitive function is performed. This standard leverages the work of the IEEE 1588 Working Group by developing the additional specifications needed to address these requirements.

2. Normative references

The following referenced documents are indispensable for the application of this standard (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 802.1AC™-2012, IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Service Definition.[2], [3]

IEEE Std 802.1AX™-2014, IEEE Standard for Local and metropolitan area networks—Link Aggregation.

~~IEEE Std 802.1D™-2004, IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Bridges.~~[4], [5]

IEEE Std 802.1Q™- 201~~1~~4, IEEE Standard for Local and metropolitan area networks—~~Media Access Control (MAC)~~ Bridges and ~~Virtual~~ Bridged ~~Local Area~~ Networks.

~~IEEE Std 802.1ag™-2007, IEEE Standard for Local and metropolitan area networks—Virtual Bridged Local Area Networks—Amendment 5: Connectivity Fault Management.~~

IEEE Std 802.3™-20~~08~~12, IEEE Standard for ~~Information technology—Telecommunications and information exchange between systems—Local and metropolitan area network—Specific requirements, Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications~~Ethernet.

~~IEEE Std 802.3av™-2009, IEEE Standard for Information technology—Part 3: Amendment 1: Physical Layer Specifications and Management Parameters for 10 Gb/s Passive Optical Networks.~~

IEEE Std 802.11™-2012, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

IEEE P802.11-REVmc/D6.0, IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, April 2016.

<<Editor's note: The above is the latest draft of P802.11-REVmc, as of D4.0 of the current document. This reference will be replaced by a reference to an approved, published standard prior to sponsor ballot.>>

IEEE Std 1588™-2008, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.

[2]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org).

[3]The IEEE standards or products referred to in Clause 2 are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.

[4]~~IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org).~~

[5]~~The IEEE standards or products referred to in Clause 2 are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.~~

[Editor's note: It is expected that IEEE 1588-Rev will be approved and published prior to sponsor ballot of 802.1AS-Rev. The above reference to IEEE 1588-2008 will be updated when IEEE 1588-Rev is published.]

IETF RFC 3410 (December 2002), Introduction and Applicability Statements for Internet Standard Management Framework, Case, J., Mundy, R., Partain, D, and Stewart, B.[6]

ITU-T Recommendation G.9960 (ex. G.hn), Unified high-speed wire-line based home networking tranceivers—System architecture and physical layer specification, June 2010.[7]

ITU-T Recommendation G.9961, Data link layer (DLL) for unified high-speed wire-line based home networking transceivers, June 2010.

ITU-T Recommendation G.984.3, Amendment 2 (2009-11) Gigabit-capable Passive Optical Networks (G-PON): Transmission convergence layer specification—Time-of-day distribution and maintenance updates and clarifications, November 2009.

MoCA® MAC/PHY Specification v2.0, MoCA-M/P-SPEC-V2.0-20100507, Multimedia over Coax Alliance (MoCA).[8]

[6]IETF RFCs are available from the Internet Engineering Task Force Web site at http://www.ietf.org/rfc.html.

[7]ITU-T publications are available from the International Telecommunications Union, Place des Nations, CH-1211, Geneva 20, Switzerland/Suisse (http://www.itu.int/).

[8]MoCA specifications are available from the Multimedia over Coax Alliance at http://www.mocalliance.org/specs.

3. Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary: Glossary of Terms & Definitions* should be consulted for terms not defined in this clause.[9]

3.1 accuracy: The mean of the time or frequency error between the clock under test and a perfect reference clock over an ensemble of measurements.

3.2 Bridge: Either a MAC Bridge~~, as specified in Clause 5 of IEEE Std 802.1D-2004,~~ or a VLAN-aware Bridge, as specified in Clause 5 of IEEE Std 802.1Q-2014~~05~~.

3.3 clock: A physical device that is capable of providing a measurement of the passage of time since a defined epoch.

3.4 direct communication: A communication of IEEE 802.1AS information between two time-aware systems with no intervening time-aware system.

3.5 end station: A device attached to a local area network (LAN) or metropolitan area network (MAN), which acts as a source of, and/or destination for, traffic carried on the LAN or MAN.

NOTE—In this standard, an *end station* is sometimes referred to as a *station.*[10]

3.6 event message: A message that is timestamped on egress from a time-aware system and ingress to a time-aware system.

NOTE—See 8.4.3.

3.7 fractional frequency offset: The fractional frequency offset, y, between a measured clock and a reference clock is defined as:

$$y = \frac{f_m - f_r}{f_r}$$

where f_m is the frequency of the measured clock and f_r is the frequency of the reference clock. The measurement units of f_m and f_r are the same.

3.8 general message: A message that is not timestamped.

3.9 gPTP communication path: A segment of a generalized precision time protocol (gPTP) domain that enables direct communication between two time-aware systems.

NOTE—See 8.1.

3.10 grandmaster: The time-aware system that contains the best clock, as determined by the best master clock algorithm (BMCA), in the generalized precision time protocol (gPTP) domain.

3.11 local area network (LAN): A network of devices, whether indoors or outdoors, covering a limited geographic area, e.g., a building or campus.

3.12 message timestamp point: A point within an event message serving as a reference point for when a timestamp is taken.

[9]*The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at http://shop.ieee.org/.

[10]Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

3.13 maximum absolute value relative time error: The maximum absolute value relative time error, max|RTE|, between two clocks over a measurement interval of duration T, is defined as:

$$\max|\mathrm{RTE}| = \max_{t_0 \le t \le t_0+T} |x_2(t) - x_1(t)|$$

where $x_1(t)$ and $x_2(t)$ are the time errors, relative to the timescale in use, of the two clocks, respectively, as a function of time, and t_0 is the start time of the measurement interval.

3.14 message type: The message type of a message is the name of the respective message, e.g., Sync, Announce, Timing Measurement Action Frame.

3.15 precision: A measure of the deviation from the mean of the time or frequency error between the clock under test and a perfect reference clock.

3.16 primary reference: A source of time and/or frequency that is traceable to international standards. *See also:* **traceability.**

3.17 recognized standard source of time: A recognized standard time source is a source external to IEEE 1588 precision time protocol (PTP) that provides time that is traceable to the international standards laboratories maintaining clocks that form the basis for the *temps atomique international* (international atomic time) (TAI) and coordinated universal time (UTC) timescales. Examples of these are National Institute of Standards and Technology (NIST) timeservers and global ~~positioning (satellite)~~navigation~~-~~satellite systems (G~~PS~~NSSs).

3.18 reference plane: The boundary between a port of a time-aware system and the network physical medium. Timestamp events occur as frames cross this interface.

3.19 residence time: The duration of the time interval between the receipt of a ~~time synchronization~~time-synchronization event message by a time-aware system, and the sending of the next subsequent ~~time synchronization~~time-synchronization event message on another port of that time-aware system. The residence time can be different for different ports.

NOTE—If a port of a time-aware system sends a ~~time synchronization~~time-synchronization event message without having received a ~~time synchronization~~time-synchronization event message, i.e., if sync receipt timeout occurs (see 10.7.3.1), the duration of the interval between the most recently received ~~time synchronization~~time-synchronization event message and the sent ~~time synchronization~~time-synchronization event message is mathematically equivalent to residence time; however, this interval is not normally referred to as a *residence time*.

3.20 stability: A measure of how the mean of the time or frequency error between the clock under test and a perfect reference clock varies with respect to variables such as time, temperature, etc.

3.21 ~~synchronized time~~synchronized time: The ~~synchronized time~~synchronized time of an event is the time of that event relative to the grandmaster.

NOTE—If there is a change in the grandmaster or grandmaster time base, the *~~synchronized time~~synchronized time* can experience a phase and/or frequency step.

3.22 ~~synchronized time~~synchronized time-aware systems: Two time-aware systems are synchronized to a specified uncertainty if they have the same epoch and their measurements of the time of a single event at an arbitrary time differ by no more than that uncertainty.

NOTE—See 8.2.2.

3.23 syntonized time-aware systems: Two time-aware systems are syntonized if the duration of the second is the same on both, which means the time measured by each advances at the same rate. They can but need not share the same epoch.

3.24 time-aware ~~Bridge~~relay: A ~~Bridge~~system that is capable of communicating ~~synchronized time~~synchronized time received on one port to other ports, using the IEEE 802.1AS protocol.

NOTE—A time-aware relay could be, for example, a bridge, a router, or a multi-port end station.

3.25 time-aware end station: An end station that is capable of acting as the source of ~~synchronized time~~synchronized time on the network, or destination of ~~synchronized time~~synchronized time using the IEEE 802.1AS protocol, or both.

3.26 time-aware system: A time-aware ~~Bridge~~relay or a time-aware end station.

3.27 timestamp measurement plane: The plane at which timestamps are captured. If the timestamp measurement plane is different from the reference plane, the timestamp is corrected for ingressLatency and/ or egressLatency. *See:* **reference plane.**

NOTE—For timestamp on egress and ingress, see 8.4.3.

3.28 traceability: See IEEE Std 1588-2008, 3.1.44.

4. Acronyms and abbreviations

ACK	acknowledgement
ADEV	Allan deviation
AP	(wireless LAN) access point
AV	audio/video
AVB	audio/video bridging
AVB network	audio/video bridged network
BC	boundary clock
BMC	best master clock
BMCA	best master clock algorithm
CID	Company ID
CMLDS	Common Mean Link Delay Service
CSN	coordinated shared network
CTC	channel time clock
EPON	IEEE 802.3 Ethernet passive optical network, as specified in IEEE Std 802.3~~-2008 and IEEE Std 802.3av-2009~~
ESS	extended service set
G.hn	ITU-T G.9960 and ITU-T G.9961
GM	grandmaster
GMT	Greenwich mean time
GNSS	global navigation satellite system
GPS	global positioning (satellite) system
gPTP	generalized precision time protocol
IP	Internet protocol
IS	integration service
ISO	International Organization for Standardization[11]
ISS	Internal Sublayer Service
LAN	local area network
LCI	location configuration information
LLC	logical link control
MAC	media access control
MACsec	media access control security

[11]Information available at www.iso.org.

MA-L	MAC addresses large
MA-M	MAC addresses medium
MAN	metropolitan area network
MA-S	MAC addresses small
MLME	IEEE 802.11 MAC layer management entity
MPCP	IEEE 802.3 multipoint control protocol
MPDPDU	IEEE 802.3 MPCP data unit
MII	media-independent interface
MD	media-dependent

NTP	network time protocol[12]
OC	ordinary clock
OLT	IEEE 802.3 optical line terminal
ONU	IEEE 802.3 optical network unit
OSSP	organization-specific slow protocol
P2P	peer-to-peer
PAR	project authorization request
PICS	Protocol Implementation Conformance Statement
PLL	phased-lock loop
POSIX®	portable operating system interface (see ISO/IEC 9945:2003 [B10][13])
PTP	IEEE 1588 precision time protocol
PTPDEV	PTP deviation
RTT	round-trip time
SI	international system of units
SM	state machine
TAI	temps atomique international (international atomic time)
TC	transparent clock
TDEV	time deviation
TDM	time division multiplexing
TDMA	time division multiple access
TG	task group
TS	timestamp
UCT	unconditional transfer
UTC	coordinated universal time
VLAN	virtual local area network
WG	Working Group
WLAN	wireless local area network

[12]Information available at www.ietf.org/rfc/rfc1305.txt.

[13]The numbers in brackets correspond to those of the bibliography in Annex G.

5. Conformance

This clause specifies the mandatory and optional capabilities provided by conformant implementations of this standard. An implementation can:

a) Compose all of part of the functionality of a system;
b) Provide, as specified by this standard, one or more instances of the MAC Service to other functional entities whose specification is outside the scope of this standard;
c) Provide, as specified by this standard, one or more instances of the MAC Internal Sublayer Service (ISS) to other implementations or instances of the same implementation that conform to this standard.

Accordingly, and as detailed in 5.3, this clause specifies conformance requirements for common systems and for functional components within systems, possibly connected to other system components with interfaces that are not otherwise accessible.

5.1 Requirements terminology

For consistency with existing IEEE and IEEE 802.1 standards, requirements placed upon conformant implementations of this standard are expressed using the following terminology:

a) ***shall*** is used for mandatory requirements;
b) ***may*** is used to describe implementation or administrative choices (“may” means “is permitted to,” and hence, “may” and “may not” mean precisely the same thing);
c) ***should*** is used for recommended choices (the behaviors described by “should” and “should not” are both permissible but not equally desirable choices).

The Protocol Implementation Conformance Statement (PICS) proforma (see Annex A) reflects the occurrences of the words shall, may, and should within the standard.

The standard avoids needless repetition and apparent duplication of its formal requirements by using ***is***, ***is not***, ***are***, and ***are not*** for definitions and the logical consequences of conformant behavior. Behavior that is permitted but is neither always required nor directly controlled by an implementer or administrator, or whose conformance requirement is detailed elsewhere, is described by ***can***. Behavior that never occurs in a conformant implementation or system of conformant implementations is described by ***can not***. The word ***allow*** is used as a replacement for the cliché “support the ability for,” and the word ***capability*** means “can be configured to.”

5.2 Protocol Implementation Conformance Statement (PICS)

The supplier of an implementation that is claimed to conform to this standard shall complete a copy of the PICS proforma provided in Annex A and shall provide the information necessary to identify both the supplier and the implementation.

5.3 Time-aware ~~Bridge and end station requirements~~systems

An implementation of timing and synchronization in ~~Bridges~~ a system shall:

a) ~~Conform to the requirements of IEEE Std 802.1Q;~~
b) Implement the generalized precision time protocol (gPTP) requirements specified in 8.2, 8.4, 8.5, and 8.6;

c) Support the media-independent slave clock at least on one port (10.2.12), and on each supported port, implement PortSyncSyncReceive function (10.2.7.3) and ClockSlaveSync function (10.2.12.3);
d) Support the requirements where no best master is present in the domain (10.2.12.2);
e) Support the following best master clock algorithm (BMCA) requirements (10.3):
   1) Support the ~~T~~time-aware system attributes and requirements (8.6.2);
   2) Implement the BMCA (10.3.2);
   3) Implement PortAnnounceReceive function (10.3.10);
   4) Implement PortAnnounceInformation function (10.3.11);
   5) Implement Port~~Role~~StateSelection function (10.3.12).
f) Implement SiteSyncSync function (10.2.6).

~~5.3.1 Time-aware Bridge and end station options~~

An implementation of ~~Time-aware~~ timing and synchronization in a ~~Bridge~~system may:

g) Support the following Grandmaster Capability (8.6.2.1 and 10.1.2):
   1) Implement ClockMasterSyncSend function (10.2.8);
   2) Implement ClockMasterSyncOffset function (10.2.9);
   3) Implement ClockMasterReceive function (10.2.10).
h) Support the media-independent slave clock on more than one port (10.2.6);
i) Support the following Media Independent Master Capability on at least one port (10.2.11):
   1) Implement PortSyncSyncSend function (10.2.11);
   2) Implement PortAnnounceTransmit function (10.3.15);
   3) Implement AnnounceIntervalSetting function (10.3.16);
   4) Conform to Announce message requirements (10.5.3);
   5) Support the Announce sequence number requirements (10.5.7);
   6) Support the Announce message PDU requirements (10.6).
j) Support more than one domain.

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

An implementation of ~~T~~time-aware ~~Bridges~~systems with IEEE 802.3 MAC services to physical ports shall support full-duplex operation, as specified in 11.2 and IEEE Std 802.3-2012, 4.2 and Annex 4A.

~~:~~

a) ~~Conform to the requirements of IEEE 802.1Q MAC-specific bridging methods;~~
b) ~~Support full-duplex operation, as specified in 11.2 and IEEE Std 802.3-2008, 4.2 and Annex 4.~~

5.5 MAC-specific timing and synchronization methods for IEEE Std 802.11~~-2007~~

An implementation of ~~T~~time-aware ~~Bridges~~systems with IEEE 802.11 MAC services to physical ports shall:

a) ~~Conform to the requirements of IEEE 802.1Q MAC-specific bridging methods;~~
b) Conform to the requirements of TIMINGMSMT as specified in IEEE Std 802.11 if the time-aware system is a bridge;
c) Optionally support FINETIMINGMSMT as specified in IEEE Std 802.11REVmc if the time-aware system is a bridge;
d) Support at least one of TIMINGMSMT as specified in IEEE Std 802.11 or FINETIMINGMSMT as specified in IEEE Std 802.11REVmc if the time-aware system is an end station;

e) Support the requirements as specified in 12.2;
f) ~~Implement MDSync Message protocol, its message parameters and defaults (12.4).~~Support the requirements of the master port (12.5.1) or the slave port (12.5.2).

5.6 MAC-specific timing and synchronization methods for IEEE 802.3 EPON

An implementation of ~~Timing~~time-aware ~~Bridges~~systems with IEEE 802.3 EPON MAC services to physical ports shall:

a) ~~Conform to the requirements of IEEE 802.1Q MAC-specific bridging methods;~~
b) Support the requirements as specified in IEEE Std 802.3-20~~08~~12, Multipoint MAC Control (64.2 and 64.3) and Multipoint PCS and PMA extensions (65);
c) Implement TIMESYNC Message protocol, and its message parameters and defaults (13.3.1);
d) Implement requester and responder functions (13.8.1 and 13.8.2).

5.7 MAC-specific timing and synchronization methods for coordinated shared network (CSN)

An implementation of ~~Timing~~time-aware ~~Bridges~~systems with CSN MAC services to physical ports shall:

a) ~~Conform to the requirements of IEEE 802.1Q MAC-specific bridging methods;~~
b) Implement path delay calculation, as specified in 16.4;
c) Implement functionality of MDSyncSendSM and MDSyncReceiveSM state machines (16.3);
d) Implement CSN specific Grandmaster Capability (16.7).

6. Conventions

6.1 General

This clause defines various conventions and notation used in the standard, i.e., naming conventions, service specification method and notation, and data type definitions.

6.2 Service specification method and notation

The method and notation for specifying service interfaces is described in Clause 7 ~~6.1~~of IEEE Std 802.1~~ag-2007~~AC-2012.

6.3 Lexical form syntax

A lexical form refers to:

a) A name
b) A data type

The conventions illustrated in the following list regarding lexical forms are used in this standard:

1) Type names: e.g., ClockQuality (no word separation, initial letter of each word capitalized);
2) Enumeration members and global constants: e.g., ATOMIC_CLOCK (underscore word separation, all letters capitalized);
3) Fields within PTP messages, instances of structures, and variables: e.g., secondsField, clockQuality, clockIdentity (two-word field names at a minimum, no word separation, initial word not capitalized, initial letter capitalization on subsequent words);
4) Members of a structure: e.g., clockQuality.clockClass (structure name followed by a period followed by the member name);
5) Data set name: e.g., defaultDS, parentDS, portDS, currentDS, timePropertiesDS (no word separation, initial letter of each word not capitalized followed by the letters DS;
6) Data set members: e.g., defaultDS.clockQuality.clockClass (Data set name followed by a period followed a type name followed by a period followed by the variable name; and
7) PTP message names: e.g., Sync, Delay_Req (underscore word separation, initial letter of each word capitalized.

When a lexical form appears in text, as opposed to in a type, or a format definition, the form is to be interpreted as singular, plural, or possessive as appropriate to the context of the text.

6.4 Data types and on-the-wire formats

6.4.1 General

The data types specified for the various variables and message fields define logical properties that are necessary for correct operation of the protocol or interpretation of IEEE 1588 precision time protocol (PTP) or IEEE802.11 message content.

NOTE—Implementations are free to use any internal representation of data types if the internal representation does not change the semantics of any quantity visible via communications using the IEEE 802.1AS protocol or in the specified operations of the protocol.

6.4.2 Primitive data types specifications

All non-primitive data types are derived from the primitive types in Table 6-1. Signed integers are represented in two's complement form.

Table 6-1—Primitive data types

Data type	Definition
Boolean	TRUE or FALSE
EnumerationN	N-bit enumerated value
UIntegerN	N-bit unsigned integer
IntegerN	N-bit signed integer
Nibble	4-bit field not interpreted as a number
Octet	8-bit field not interpreted as a number
OctetN	N-octet field not interpreted as a number, with N > 1
Double	Double precision (64-bit) floating-point value

6.4.3 Derived data type specifications

6.4.3.1 ScaledNs

The ScaledNs type represents signed values of time and time interval in units of 2^{-16} ns.

typedef Integer96 ScaledNs;

For example: –2.5 ns is expressed as:

0xFFFF FFFF FFFF FFFF FFFD 8000

Positive or negative values of time or time interval outside the maximum range of this data type are encoded as the largest positive or negative value of the data type, respectively.

6.4.3.2 UScaledNs

The UScaledNs type represents unsigned values of time and time interval in units of 2^{-16} ns.

typedef UInteger96 UScaledNs;

For example: 2.5 ns is expressed as:

0x0000 0000 0000 0000 0002 8000

Values of time or time interval greater than the maximum value of this data type are encoded as the largest positive value of the data type, respectively.

6.4.3.3 TimeInterval

The TimeInterval type represents time intervals, in units of 2^{-16} ns.

```
struct TimeInterval
{
        Integer64 scaledNanoseconds;
};
```

For example: 2.5 ns is expressed as:

0x0000 0000 0002 8000

Positive or negative time intervals outside the maximum range of this data type are encoded as the largest positive and negative values of the data type, respectively.

6.4.3.4 Timestamp

The Timestamp type represents a positive time with respect to the epoch.

```
struct Timestamp
{
        UInteger48 seconds;
        UInteger32 nanoseconds;
};
```

The seconds member is the integer portion of the timestamp in units of seconds.

The nanoseconds member is the fractional portion of the timestamp in units of nanoseconds.

The nanoseconds member is always less than 10^9.

For example:

+2.000000001 seconds is represented by seconds = 0x0000 0000 0002 and nanoseconds= 0x0000 0001

6.4.3.5 ExtendedTimestamp

The ExtendTimestamp type represents a positive time with respect to the epoch.

```
struct ExtendedTimestamp
{
        UInteger48 seconds;
        UInteger48 fractionalNanoseconds;
};
```

The seconds member is the integer portion of the timestamp in units of seconds.

The fractionalNanoseconds member is the fractional portion of the timestamp in units of 2^{-16} ns.

The fractionalNanoseconds member is always less than $(2^{16})(10^9)$.

For example:

+2.000000001 seconds is represented by seconds = 0x0000 0000 0002 and fractionalN~~n~~anoseconds = 0x0000 0001 0000

6.4.3.6 ClockIdentity

The ClockIdentity type identifies a time-aware system.

typedef Octet8 ClockIdentity;

6.4.3.7 PortIdentity

The PortIdentity type identifies a port of a time-aware system.

```
struct PortIdentity
{
        ClockIdentity clockIdentity;
        UInteger16 portNumber;
};
```

6.4.3.8 ClockQuality

The ClockQuality represents the quality of a clock.

```
struct ClockQuality
{
        UInteger8 clockClass;
        Enumeration8 clockAccuracy;
        UInteger16 offsetScaledLogVariance;
};
```

6.4.4 Protocol data unit (PDU) formats

6.4.4.1 General

The data types defined in 6.4.2 and 6.4.3 shall be mapped onto the wire according to the mapping rules for the respective medium, e.g., IEEE Std 802.3~~-2008~~ and IEEE Std 802.11, and the terms of 6.4.4.

IEEE 802.1AS PDUs consist of the messages defined or referenced in Clause 10, Clause 11, Clause 12, and Clause 13, based on the data types defined in 6.4.2 and 6.4.3. The internal ordering of the fields of the IEEE 802.1AS PDUs is specified in 6.4.4.3 to 6.4.4.5.

6.4.4.2 Numbering of bits within an octet

Bits are numbered with the most significant bit being ~~8~~7 and the least significant bit being ~~1~~0.

NOTE—The numbering and ordering of bits within an octet of a PDU, described here, is independent of and unrelated to the order of transmission of the bits on the underlying physical layer.

6.4.4.3 Primitive data types

Numeric primitive data types defined in 6.4.2 shall be formatted with the most significant octet nearest the beginning of the PDU followed in order by octets of decreasing significance.

The Boolean data type TRUE shall be formatted as a single bit equal to 1 and FALSE as a single bit equal to 0.

Enumerations of whatever length shall be formatted as though the assigned values are unsigned integers of the same length, e.g., Enumeration16 shall be formatted as though the value had type UInteger16.

6.4.4.4 Arrays of primitive types

All arrays shall be formatted with the member having the lowest numerical index closest to the start of the PDU followed by successively higher numbered members, without any padding. In octet arrays, the octet with the lowest numerical index is termed the most significant octet.

When a field containing more than one octet is used to represent a numeric value, the most significant octet shall be nearest the start of the PDU, followed by successively less significant octets.

When a single octet contains multiple fields of primitive data types, the bit positions within the octet of each of the primitive types as defined in the message field specification shall be preserved. For example, the first field of the header of PTP messages is a single octet composed of two fields, one of type Nibble bits ~~5–8~~4–7, and one of type Enumeration4 bits ~~1–4~~0–3, see 11.4.2 and 10.6.2.

6.4.4.5 Derived data types

Derived data types defined as structs shall be formatted with the first member of the struct closest to the beginning of the PDU followed by each succeeding member, without any padding. Each member shall be formatted according to its data type.

Derived data types defined as typedefs shall be formatted according to its referenced data type.

7. F~~Time synchronization~~Time-synchronization model for a ~~bridged local area~~ network

7.1 General

This clause provides a model for understanding the operation of the generalized precision time protocol (gPTP), which specifies the operation of time-aware systems on a ~~bridged~~ packet-switched ~~LAN~~network. Although this standard is based on the precision time protocol (PTP) described in IEEE Std 1588-2008 (and, indeed, is a proper profile of IEEE Std 1588 in particular configurations) there are differences, which are summarized in 7.5.

Although this standard has been written as a stand-alone document, it is useful to understand the IEEE 1588 architecture as described in Clause 6 of that document.

7.2 Architecture of a time-aware ~~bridged local area~~ network

7.2.1 General

A time-aware ~~bridged LAN~~network consists of a number of interconnected time-aware systems ~~interconnected by LANs~~ 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 ~~LANs~~ network elements is called a *gPTP network*. Each instance of gPTP that the time-aware systems support is in one *gPTP domain*, and the time-aware systems 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. There are two types of time-aware systems, as follows:

a) Time-aware end station, which if not grandmaster, is a recipient of time information, and
b) Time-aware ~~Bridge~~relay, which if not grandmaster, receives time information from the grandmaster (perhaps indirectly through other time-aware ~~Bridges~~relays), applies corrections to compensate for delays in the LAN and the ~~Bridge~~time-aware relay 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 (Clause 11)
d) IEEE 802.3 Ethernet using passive optical network (EPON) links (Clause 13)
e) IEEE 802.11 wireless (Clause 12)
f) Generic coordinated shared networks (CSNs, e.g., MoCA and G.hn) (Clause~~Annex~~ 16)

7.2.2 Time-aware network consisting of a single gPTP domain

Figure 7-1 illustrates an example time-aware network consisting of a single gPTP domain, using all ~~those~~the above network technologies (i.e., (c) - (f) of 7.2.1), where end stations on several local networks are connected to a grandmaster on a backbone network via an EPON access network.

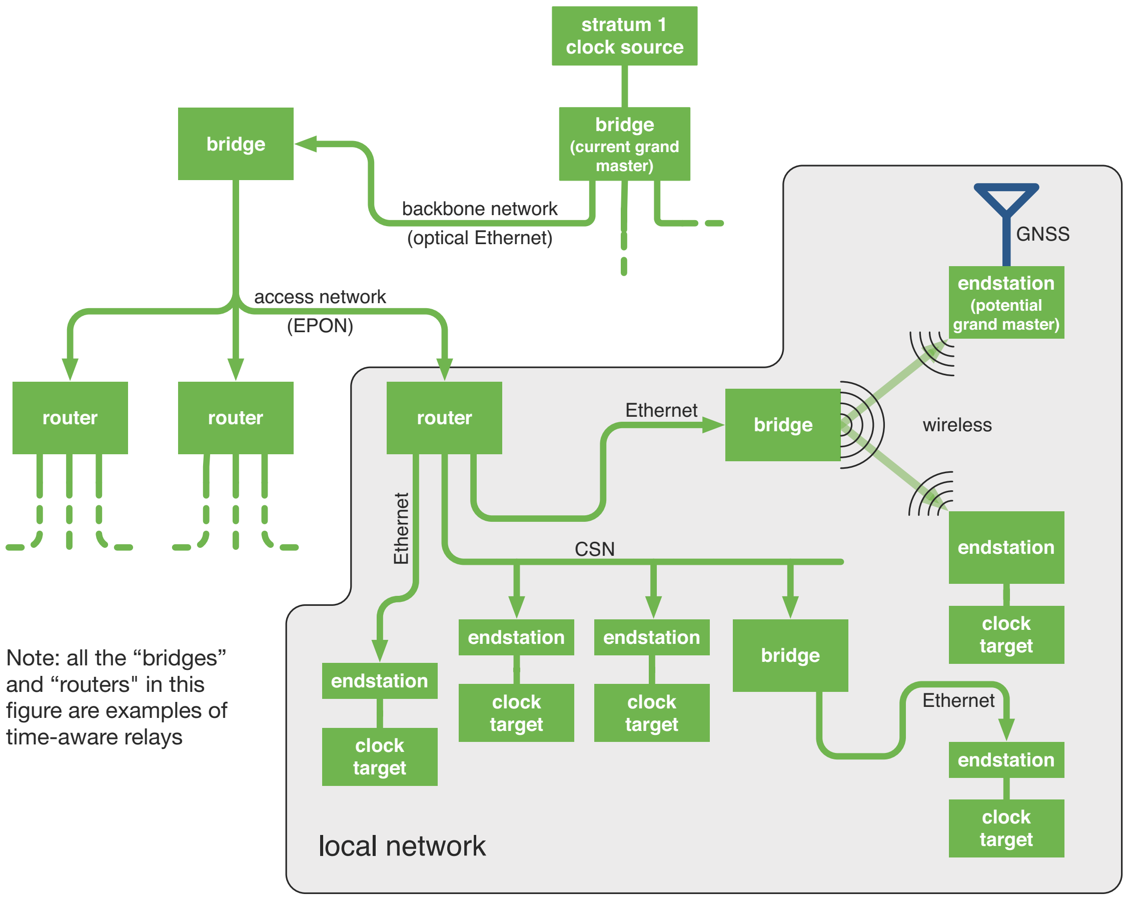


Figure 7-1—Time-aware network example

Any time-aware system with clock sourcing capabilities can be a potential grandmaster, so there is a selection method (the *best master clock algorithm*, or BMCA) that ensures that all of the time-aware systems in a gPTP domain use the same grandmaster.[14] The BMCA is largely identical to that used in IEEE Std 1588-2008, but somewhat simplified. In Figure 7-1 the BMCA process has resulted in the grandmaster being on the network backbone. If, however, the access network fails, the systems on a local network automatically switch over to one of the potential grandmasters on the local network that is as least as "good" as any other. For example, in Figure 7-2, the access network link has failed, so a potential grandmaster that has a G~~PS~~NSS reference source has become the active grandmaster, and there are now two gPTP domains where there used to be one. Finally, note that in the case where a time-aware system supports more than one domain, one of the domains supported must be domain 0 for backward compatibility with the 2011 edition of this standard, though domain 0 is not necessarily active in a time-aware system.

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

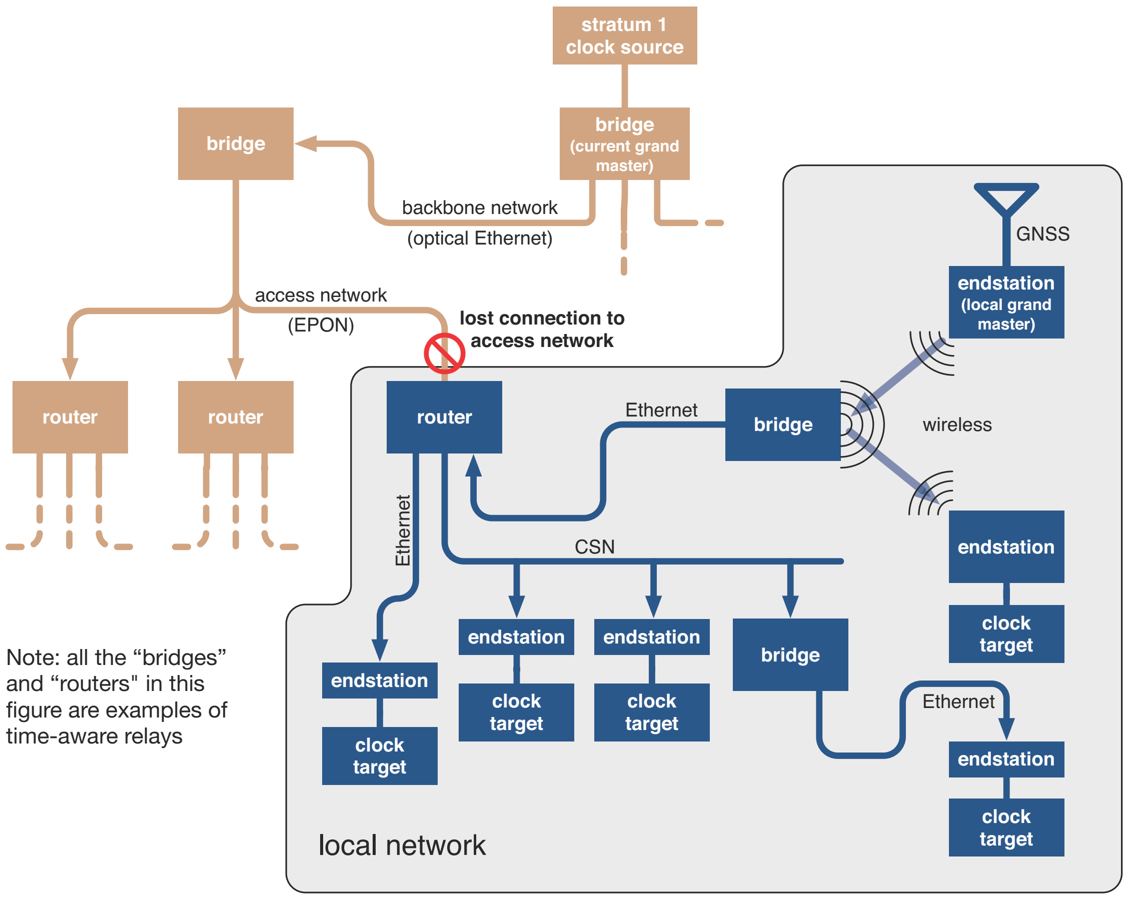


Figure 7-2—Time-aware network of Figure 7-1 after an access network link failure

7.2.3 Time-aware network consisting of multiple gPTP domains

Figure 7-3 illustrates an example time-aware network consisting of multiple gPTP domains that could be used in an industrial application. Specifically, in this example the network has two timescales/domains, where domain 0 uses the PTP timescale and domain 1 uses the ARB timescale (see 8.2). Notice that not all stations in domain 1 (within the blue dashed rectangle) have domain 0 active in this example, even though every time-aware system supports domain 0 for backward compatibility with the 2011 edition of this standard. In addition, it is required that all stations belonging to the same domain have direct connections among them in their physical topology (e.g., time cannot be transported from one time-aware system in domain 0 to another time-aware system in domain 0 via a time-aware system in domain 1 if the time-aware system in domain 1 does not also have domain 0 active) . In addition, those stations for which both domains are active are depicted in double layers, representing two independent, active PTP instances.

As in the single-domain case, any of the network technologies of 7.2.1 can be used. The grandmaster of each domain is selected by the BMCA; now, a separate, independent instance of the BMCA is invoked in each domain.

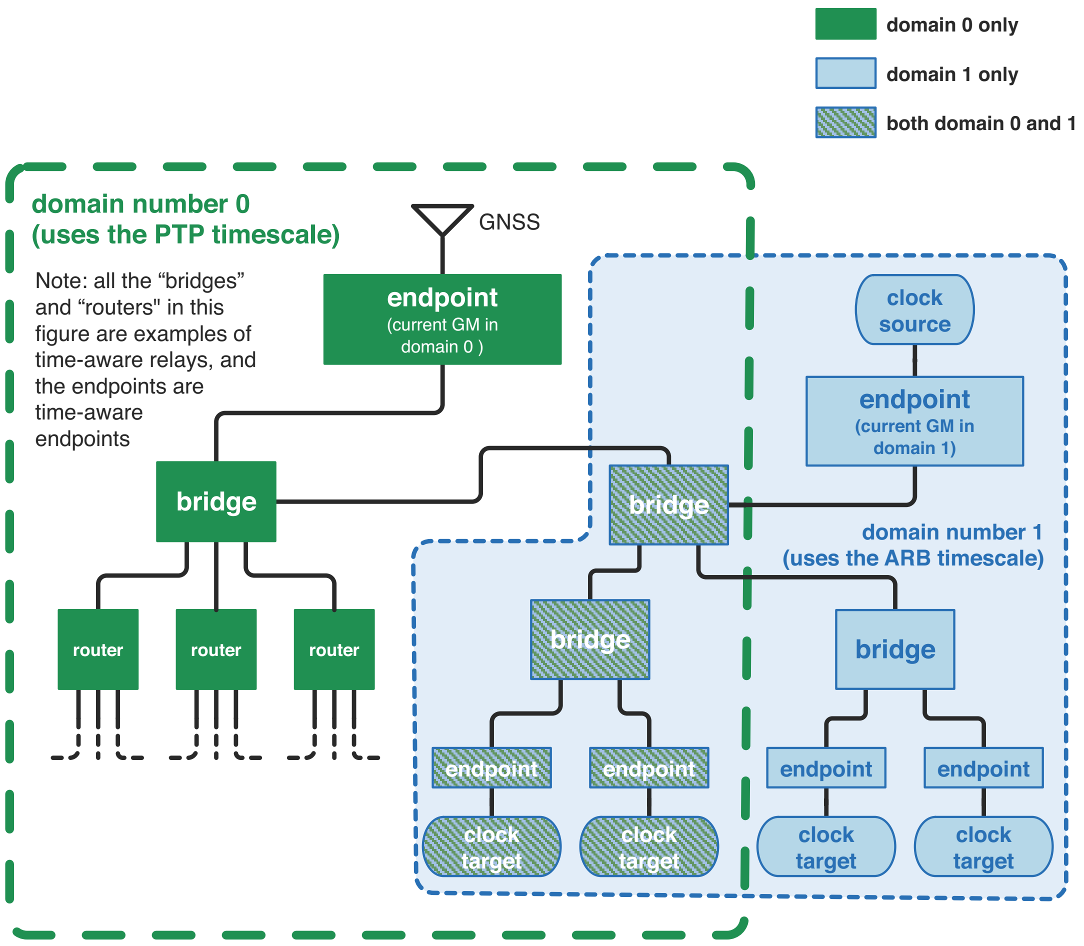


Figure 7-3—Time-aware network example for the case of multiple gPTP domains

7.2.4 Time-aware networks with redundant grandmasters and/or redundant paths

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.

This standard provides a basic level of redundancy as follows:

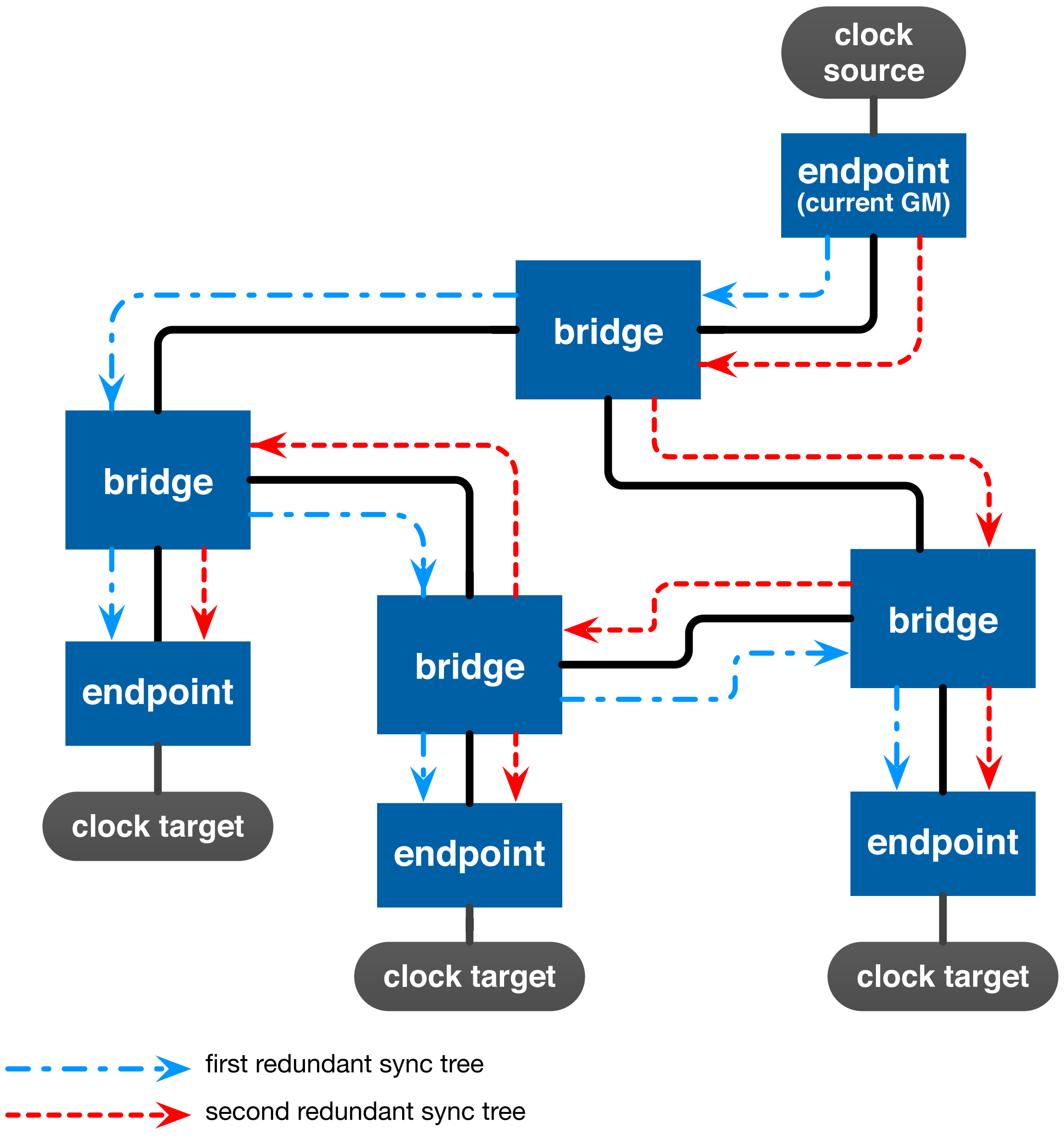
- A detection component that triggers when the current grandmaster stops working (i.e., loss of Sync messages and Announce messages for a period of time) or if the link to the grandmaster goes down (i.e., immediate loss of Sync messages and Announce messages).
- A correction component that triggers the Best Master Clock Algorithm and the sending of Announce messages, so that a new grandmaster can be elected.
- An action component, where the winning grandmaster starts sending Announce and Sync messages and all the time-aware systems listen to this new grandmaster.

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 and clock path redundancy. Figures 7-4 through 7-6 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 (i.e., the action component of) 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 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 outside the scope of 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.

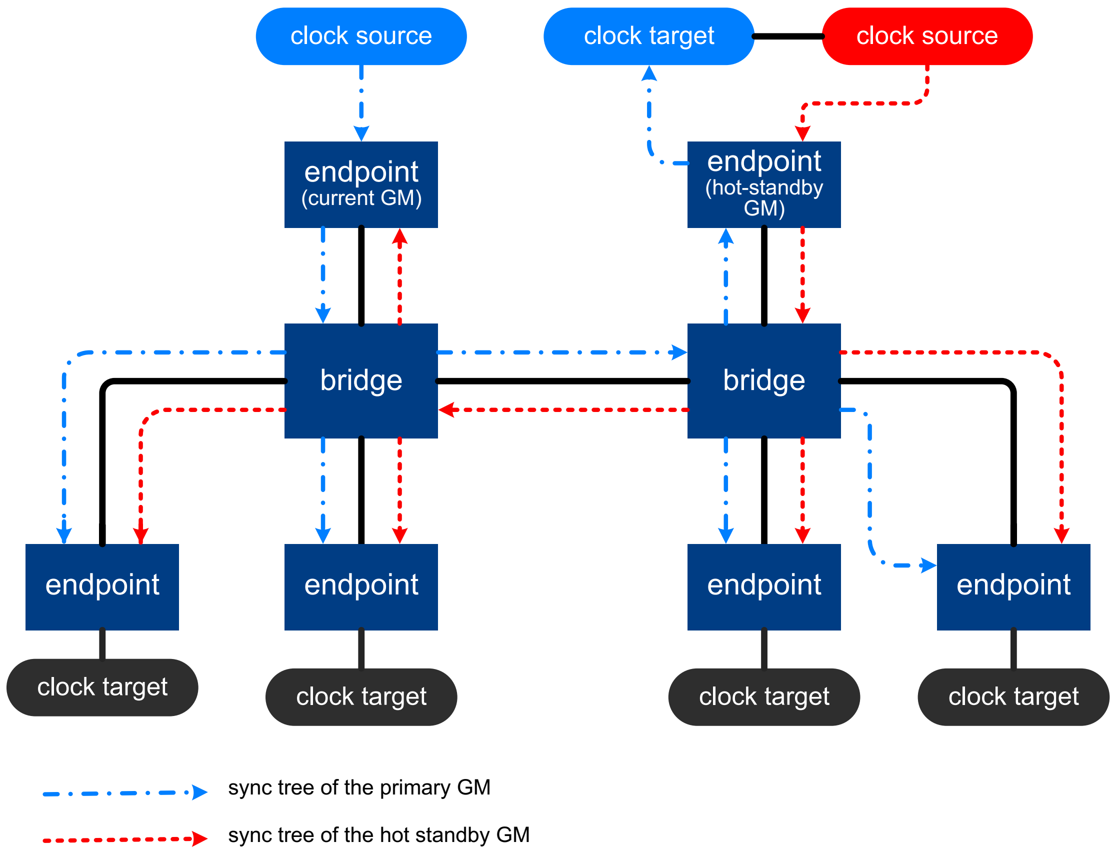
Figure 7-4 shows an example network realizing two redundant synchronization trees from a single GM, with each synchronization tree in a different gPTP domain (there are a total of two gPTP domains).



Note 1: The methods used for merging the redundant sync msgs received at each endpoint are not specified in this standard.
Note 2: All the "bridges" in this figure are examples of time-aware relays, and the endpoints are time-aware.

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

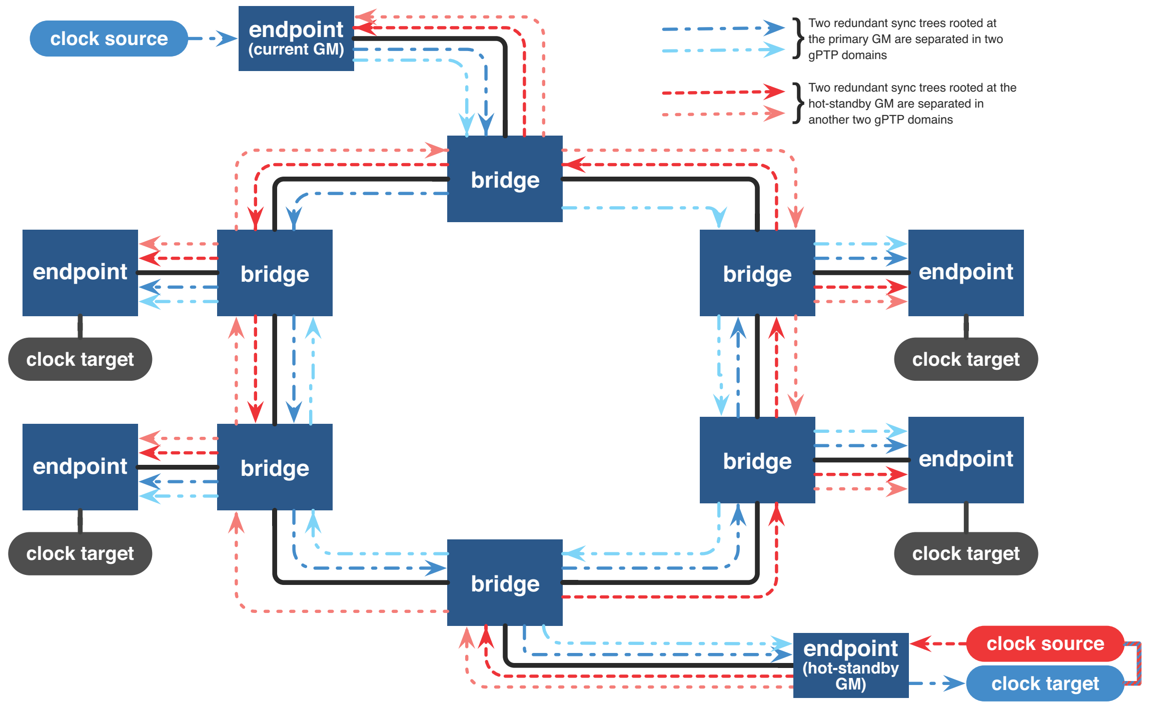
Figure 7-5 shows an example network with two redundant GMs, one as primary GM and the other as secondary GM, where each GM has one of the two redundant synchronization trees originating from it. This example supports hot-standby operating mode. In this mode, the seconday GM has to be synchronized to 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 msgs received at each endpoint are not specified in this standard.
Note 2: The endpoint operating as the hot-standby GM might need the additional clock target attached to it if the hot-standby GM is required to be synchronized to the primary GM.
Note 3: All the “bridges” in this figure are examples of time-aware relays, and the endpoints are time-aware.

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

Figure 7-6 shows another case of a ring topology, using both the redundancy features of both Figure 7-4 and Figure 7-5.


clock source
endpoint (current GM)
Two redundant sync trees rooted at the primary GM are separated in two gPTP domains
Two redundant sync trees rooted at the hot-standby GM are separated in another two gPTP domains
bridge
endpoint
bridge
bridge
endpoint
clock target
clock target
endpoint
bridge
bridge
endpoint
clock target
clock target
bridge
endpoint (hot-standby GM)
clock source
clock target

Note 1: The methods used for merging the redundant sync msgs received at each endpoint are not specified in this standard.
Note 2: The endpoint operating as the hot-standby GM may need the additional clock target attached to it if the hot-standby GM is required to be synchronized to the primary GM.
Note 3: All the "bridges" in this figure are examples of time-aware relays, and the endpoints are time-aware.

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

7.3 Time synchronization

7.3.1 General

Time synchronization in gPTP is done the same way (in the abstract) as is done in IEEE Std 1588-2008: a grandmaster sends information including the current ~~synchronized time~~synchronized time to all directly attached time-aware systems. Each of these time-aware systems must correct the received ~~synchronized time~~synchronized time by adding the propagation time needed for the information to transit the communication path from the grandmaster. If the time-aware system is a time-aware ~~Bridge~~relay, then it must forward the corrected time information (including additional corrections for delays in the forwarding process) to all the other attached time-aware systems.

To make this all work, there are two time intervals that must be precisely known: the forwarding delay (called the *residence time*), and the time taken for the ~~synchronized time~~synchronized time information to transit the communication path between two time-aware systems. The residence time measurement is local to a ~~Bridge~~ time-aware relay and easy to compute, while the communication path delay is dependent on many things including media- dependent properties and the length of the path.

7.3.2 Delay measurement

Each type of LAN or communication path has different methods for measuring propagation time, but they are all based on the same principal: measuring the time that a well-known part of a message is transmitted from one device and the time that the same part of the same message is received by the other device, then sending another message in the opposite direction and doing the same measurement as shown in Figure 7-7.

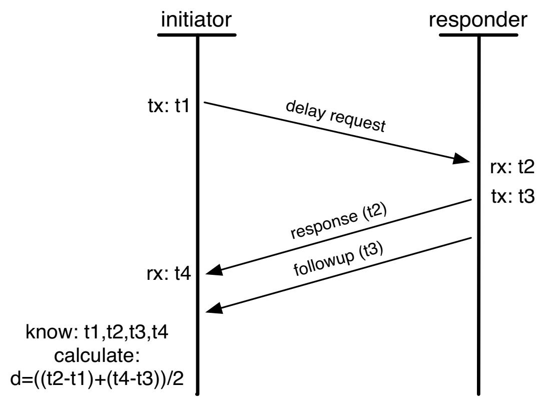


Figure 7-7— Conceptual ~~Example~~ medium delay measurement

This basic mechanism is used in the various LANs in the following ways:

a) Full-duplex Ethernet LANs use the two step peer-to-peer (P2P) path delay algorithm as defined in IEEE Std 1588-2008, where the messages are called Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up (see Figure 11-1).
b) IEEE 802.11 wireless LANs use the Timing measurement procedure defined in IEEE Std 802.11, where the messages are the “timing measurement action frame” and its corresponding “ACK~~.~~” (see Figure 12-1).
c) EPON LANs use the discovery process, where the messages are “GATE” and “REGISTER_REQ~~.~~” (see Clauses 64 and 77 of IEEE Std 802.3-2012).
d) CSNs either use the same mechanism as full-duplex Ethernet, or use a method native to the particular CSN (similar to the way native methods are used by IEEE 802.11 and EPON) (see Figure 16-5).

7.3.3 Logical syntonization

The ~~time synchronization~~time-synchronization correction previously described is dependent on the accuracy of the delay and residence time measurements. If the clock used for this purpose is frequency locked (syntonized) to the grandmaster, then all the time interval measurements use the same time base. Since actually adjusting the frequency of an oscillator (e.g., using a PLL) is slow and prone to gain peaking effects, time-aware ~~Bridges~~ relays can correct time interval measurements using the grandmaster frequency ratio.

Each time-aware system measures, at each port, the ratio of the frequency of the time-aware system at the other end of the link attached to that port to the frequency of its own clock. The cumulative ratio of the grandmaster frequency to the local clock frequency is accumulated in a standard organizational type, length, value (TLV) attached to the Follow_Up message (or the Sync message if the optional one-step processing is enabled). The frequency ratio of the grandmaster relative to the local clock is used in computing

~~synchronized time~~synchronized time, and the frequency ratio of the neighbor relative to the local clock is used in correcting the propagation time measurement.

The grandmaster frequency ratio is measured by accumulating neighbor frequency ratios for two main reasons. First, if there is a network reconfiguration and a new grandmaster is elected, the nearest neighbor frequency ratios do not have to be newly measured as they are constantly measured using the Pdelay messages. This results in the frequency offset relative to the new grandmaster being known when the first Follow_Up message (or first Sync message if the optional one-step processing is enabled) is received, which reduces the duration of any transient error in ~~synchronized time~~synchronized time during the reconfiguration. This is beneficial to many high-end audio applications. Second, there are no gain peaking effects because an error in frequency offset at one ~~node~~time-aware relay, and resulting residence time error, does not directly affect the frequency offset at a downstream ~~node~~time-aware relay.

7.3.4 Grandmaster (best master) selection and network establishment

All time-aware systems participate in best master selection so that the IEEE 802.1AS protocol can determine the synchronization spanning tree. This synchronization spanning tree ~~may~~can be different from the forwarding spanning tree determined by ~~IEEE 802.1D and~~ IEEE 802.1Q Rapid Spanning Tree Protocol (RSTP) since the spanning tree determined by RSTP can be suboptimal~~,~~ or even inadequate for synchronization, or for a different topology of nodes than the synchronization spanning tree.

gPTP requires that all ~~Bridges and end-stations~~systems in the gPTP domain be time-aware ~~-~~systems, i.e., the protocol does not transfer timing over ~~“ordinary Bridges”~~systems that are not time-aware (e.g., those that meet the requirements of ~~IEEE Std 802.1D or~~ IEEE Std 802.1Q, but do NOT meet the requirements of this standard). A time-aware system uses the peer delay mechanism on each port to determine if a non-time-aware system ~~n “ordinary Bridge”~~ is at the other end of the link or in between itself and the Pdelay responder. If, on sending Pdelay_Req

a) no response is received,
b) multiple responses are received, or
c) the measured propagation delay exceeds a specified threshold, then

the protocol concludes that a non-time-aware system ~~n “ordinary Bridge”~~ or end-to-end TC (see IEEE Std 1588-2008) is present. In this case, the link attached to the port is deemed not capable of running gPTP, and BMCA ignores it. However, the port continues to attempt the measurement of propagation delay using the peer delay mechanism (for full-duplex IEEE 802.3 links), MPCP messages (for EPON), or IEEE802.11 messages (for IEEE 802.11 links), and periodically checks whether the link is or is not capable of running IEEE 802.1AS.

7.3.5 Energy efficiency

Sending PTP messages at relatively high rates when there is otherwise little or no traffic ~~is counter to~~conflicts with the goal of reducing energy consumption. This standard specifies a way to request that a neighbor port reduce the rate of sending Sync (and Follow_Up if optional one-step processing is not enabled), peer delay, and Announce messages, and also to inform the neighbor not to compute neighbor rate ratio and/or propagation delay on this link. A time-aware system could do this when it enters low-power mode, but this standard does not specify the conditions under which this is done; it specifies only the actions a time-aware system takes.

7.4 Time-aware system architecture

The model of a time-aware system is shown in Figure 7-8.

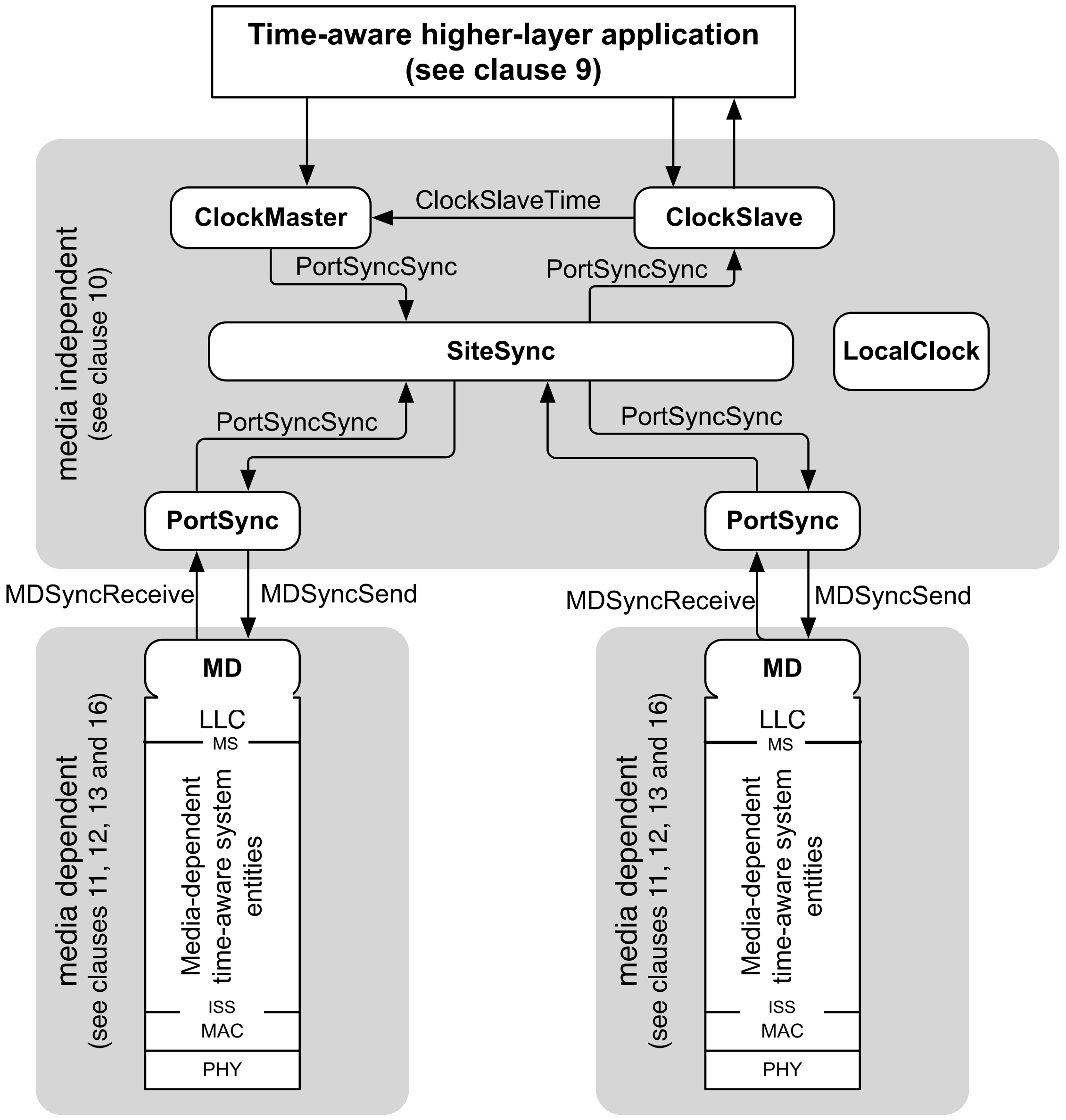


Figure 7-8—Time-aware system model

A time-aware system consists of the following major parts:

a) If the time-aware system includes application(s) that either use or source time information, then they interface with the gPTP information using the application interfaces specified in Clause 9.
b) A single media-independent part that consists of ClockMaster, ClockSlave, and SiteSync logical entities, one or more PortSync entities, and a LocalClock entity. The BMCA and forwarding of time information between logical ports and the ClockSlave and ClockMaster is done by the SiteSync

entity, while the computation of port-specific delays needed for ~~time synchronization~~time-synchronization correction is done by the PortSync entities.

c) Media-dependent ports, which translate the abstract "MDSyncSend" and "MDSyncReceive" structures received from or sent to the media-independent layer and corresponding methods used for the particular LAN attached to the port.

In the case of full-duplex Ethernet ports, IEEE 1588 Sync and Follow_Up~~-~~ (or just Sync if the optional one-step processing is enabled) messages are used, with an additional TLV in the Follow_up~~-~~ (or the Sync if the optional one-step processing is enabled) used for communication of rate ratio and information on phase and frequency change when there is a change in grandmaster. The path delay is measured using the two-step IEEE 1588 peer delay mechanism. This is defined in Clause 11.

For IEEE 802.11 ports, timing information is communicated using the MAC Layer Management Entity to request a "timing measurement" [as defined in IEEE Std 802.11], which also sends everything that would be included in the Follow_up message for full-duplex Ethernet. The timing measurement result includes all the information to determine the path delay. This is defined in Clause 12.

For EPON, timing information is communicated using a "slow protocol" as defined in Clause 13. CSNs use the same communication system used by Ethernet full-duplex, as is defined in ~~Annex~~Clause 16.

7.5 Differences between gPTP and PTP

a) gPTP assumes all communication between time-aware systems is done only using IEEE 802 MAC PDUs and addressing, while IEEE 1588 supports various layer 2 and layer 3-4 communication methods.

b) gPTP specifies a media-independent sublayer that simplifies the integration within a single timing domain of multiple different networking technologies with radically different media access protocols. gPTP specifies a media-dependent sublayer for each medium. The information exchanged between time-aware systems has been generalized to support different packet formats and management schemes appropriate to the particular networking technology. IEEE 1588-20xx, on the other hand, has inroduced a new architecture based on media-independent and media-dependent sublayers; however, this architecture is optional. The architecture of IEEE 1588-2011, which is not based on media-independent and media-dependent layers, has been retained for ~~is fully specified only for~~ IP version 4, IP version 6, Ethernet LANs, and several industrial automation control protocols.~~-~~The intent in IEEE 1588-20xx is that the new architecture, based on media-independent and media-dependent layers, will be used for IEEE 802.11, IEEE 802.3 EPON, CSN using the specifications of IEEE 802.1AS, and that the architecture must be used for transports that define native timing mechanisms if those native timing mechanisms are used. The new architecture in IEEE 1588-20xx can optionally be used for the transports described in IEEE 1588-2011.

c) In gPTP there are only two types of time-aware systems: time-aware end stations and ~~Bridges~~time-aware relays, while IEEE 1588 has ordinary clocks, boundary clocks, end-to-end transparent clocks, and P2P transparent clocks. A time-aware end station corresponds to an IEEE 1588 ordinary clock, and a time-aware ~~Bridge~~relay is a type of IEEE 1588 boundary clock where its operation is very tightly defined, so much so that a time-aware ~~Bridge~~relay with Ethernet ports can be shown to be mathematically equivalent to a P2P transparent clock in terms of how synchronization is performed, as shown in 11.1.3. In addition, a time-aware relay can optionally operate in a mode (i.e., the mode where the variable syncLocked is TRUE, see 10.2.4.15) where the time-aware relay is equivalent to a P2P transparent clock in terms of when time-synchronization messages are sent. A time-aware system measures link delay and residence time, and communicates these in a correction field.

d) Time-aware systems only communicate gPTP information directly with other time-aware systems. That is, a gPTP domain consists ONLY of time-aware systems. Non-time-aware ~~Bridges~~relays cannot be used to relay gPTP information. In IEEE 1588 it is possible to use non-IEEE-1588-aware

~~Bridges~~relays in an IEEE 1588 domain, although this slows timing convergence and introduce extra jitter and wander that must be filtered by any IEEE 1588 clock.

e) For Ethernet full-duplex links, gPTP requires the use of the peer delay mechanism, while IEEE 1588 also allows the use of end-to-end delay measurement.

f) For Ethernet full-duplex links, gPTP requires the use of two-step processing (use of Follow_Up and Pdelay_Resp_Follow_Up messages to communicate timestamps) with an optional one-step processing mode that embeds timestamps in the Sync "on the fly" as they are being transmitted, while IEEE 1588 allows either two-step or one-step processing~~-~~to be required depending on a specific profile.~~(embedding timestamps in messages "on the fly" as they are being transmitted)~~.

g) ~~In steady state, there is only a single active grandmaster in a time-aware network. That is, there is only a single gPTP domain, whereas IEEE 1588 allows multiple overlapping timing domains.~~

h) All time-aware systems in a gPTP domain are logically syntonized, meaning that they all measure time intervals using the same frequency. This is done by the process described in 7.3.3, and is mandatory. Syntonization in IEEE 1588 is optional~~, and the method used is not as direct and takes longer to converge~~. The syntonization method used by gPTP is supported in is supported as an option in IEEE 1588-20xx, but uses a TLV standardized as part of PTP (this is a new feature for IEEE 1588-20xx), while gPTP uses the ORGANIZATION_EXTENSION TLV specified in 11.4.4.3.

i) The BMCA used in gPTP is the same as that used in IEEE 1588, except that ~~with the following exceptions: (1) Announce messages received on a slave port that were not sent by the receiving time-aware system are used immediately, i.e., there is no foreign-master qualification, (2) a port that the BMCA determines should be a master port enters the master state immediately, i.e., there is no pre-master state, (3) the uncalibrated state is not needed and, therefore, not used, and (4)~~ all time-aware systems are required to participate in best master selection (even if a time-aware system~~it~~ is not grandmaster capable). For this reason, the BMCA of this standard is an alternate BMCA according to the specifications of IEEE 1588. The BMCA used in gPTP uses the new optional feature of IEEE 1588-20xx that allows foreign master qualification, pre-master qualification and the pre-master state, and the uncalibrated state to not be used.

j) Finally, this standard includes formal interface definitions, including primitives, for the time-aware applications. (See Clause 9.) IEEE Std 1588-20xx~~08~~ ~~does not define how an application provides or obtains time information.~~ describes external interfaces less formally, i.e., without describing specific interface primitives.

8. IEEE 802.1AS concepts and terminology

8.1 gPTP domain

A gPTP domain, hereafter referred to as simply a *domain,* consists of one or more time-aware systems 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.

<<Editor's note: Much of the text below this editor's note is taken, with some edits, from the latest draft of P1588 (P1588_V_20, June 26, 2016). It describes the sdoId attribute, which is needed for identifying messages of the common pdelay service, and also for the profile isolation feature.

Note, however, that the profile isolation feature is not explicitly used in 802.1AS-Rev, at least, not in the current draft. In the latest version of the profile isolation feature, the Qualified Standards Development Organization (QSDO; see 20.3.2 of P1588_V_20 for what constitutes a QSDO) that devlops the PTP profile would apply to the IEEE RA for an sdoId (i.e., once 1588-Rev is approved (assuming it is approved with this feature as described in the latest draft) and the mechanism for assigning sdoId's is established). The QSDO would then specify one or more domain numbers for use with this profile. The profile would be isolated from other PTP profiles because only this profile would be allowed to use those domain numbers and sdoId. However, IEEE 802.1AS-Rev (at least, this draft) is not a new profile relative to the profile specified in the 2011 edition of 802.1AS; rather, it is a revision of the profile that adds some new features while preserving backward compatibility for existing 2011 features. In particular, for gPTP domains transportSpecific, i.e., majorSdoId, remains 0x1, and minorSdoId is 0x00 to match the reserved field of the 2011 edition (see below for specification of these values). It is true that sdoId will have a different value (i.e., 0x200) for the Common Mean Link Delay Service; however, this feature is not used in the 2011 edition (see XXX for a description of how backward compatibility is ensured when using this service). Isolation from other PTP profiles will be ensured in the same manner that it was ensured for networks conformant to the 2011 edition, i.e., by the fact that only PTP profiles developed by IEEE 802.1 can use transportSpecific, or majorSdoId, of 0x1. If 802.1 develops an additional PTP profile(s) in the future, it could choose to use the profile isolation feature (i.e., it could request an sdoId value from the IEEE RA and then assign one or more domain numbers for use with that profile.

The TSN TG should decide if any of the description in this editor's note would be useful to maintain as a NOTE for readers of this standard.>>

<<Editor's note: Here and elsewhere in this document, a distinction is made between the domain number, which refers to the number given to the domain, and the domainNumber, which is the name of the field, variable, object, etc. (depending on the usage). However, in the case of sdoId, the same name is given to the attribute and the object, variable, etc. The distinction is not made here because it is not made in the latest draft of 1588-Rev (and 1588-Rev is followed.>>

A domain is identified by two attributes: domain number 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.

The domain number of a gPTP domain shall be in the range 0-127. A time-aware system shall support a domain whose domain number is 0. A time-aware system may support one or more additional domains each with a distinct domain number in the range 1-127. Unless otherwise specified in this standard, the operation of the gPTP protocol 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 are given in 11.2.15.

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 time-aware systems in a gPTP domain are traceable to a single grandmaster.

8.2 Timescale

8.2.1 Introduction

The timescale for a gPTP domain is established by the grandmaster. There are two types of timescales supported by gPTP:

- The timescale PTP: The epoch is the PTP epoch (see 8.2.2), and the timescale is continuous. The unit of measure of time is the SI second as realized on the rotating geoid. The timescale of domain 0 should be PTP. See IEEE Std 1588 for more details.
- The timescale ARB (arbitrary): The epoch is the domain startup time, and can be set by an administrative procedure. Between invocations of the administrative procedure, the timescale is continuous. Additional invocations of the administrative procedure can introduce discontinuities in the overall timescale.

~~, referred to as the *PTP timescale,* is established by the grandmaster. The IEEE 802.1AS timescale is continuous and can be traceable to TAI. The value of the second is the international second, SI, within the accuracy supported by the gPTP domain. The epoch is the PTP epoch (see 8.2.2).~~

8.2.2 Epoch

The epoch is the origin of the timescale of a gPTP domain.

The PTP epoch is 1 January 1970 00:00:00 TAI, which is 31 December 1969 23:59:51.999918 UTC.

~~NOTE—The PTP epoch is set such that a direct application of the POSIX algorithm to a PTP timescale timestamp yields the ISO 8601:2004 [B8] printed representation of TAI.~~NOTE—The common Portable Operating System Interface (POSIX) algorithms can be used for converting elapsed seconds since the PTP epoch to the ISO 8601:2004 printed representation of time of day on the TAI timescale; see ISO/IEC 9945:2003 [B10] and ISO 8601:2004 [B8].

See Annex C for information on converting between common timescales.

8.2.3 UTC Offset

~~It is possible to calculate UTC time using the currentUtcOffset field value of the time properties data set, if the time source is traceable to TAI and the currentUtcOffset field is valid. The value of currentUtcOffset shall be: currentUtcOffset = TAI – UTC, where TAI is the TAI time and UTC is the UTC time.~~ When the timescale is PTP, it is possible to calculate UTC time using the value of currentUtcOffset. The value of currentUtcOffset is given by

currentUtcOffset = TAI – UTC.

The value of currentUtcOffset for the current grandmaster is maintained in the currentUtcOffset member of the time properties data set (see 14.5.1).

NOTE—As of 0 hours 1 ~~January~~January 2017~~09~~ UTC, UTC was behind TAI by 3~~4~~7 s, i.e., TAI - UTC = +3~~4~~7 s. At that moment the IEEE 802.1AS defined value of currentUtcOffset became +3~~4~~7 s (see *Service de la Rotation Terrestre* [B17] and U.S. Naval Observatory [B19]).

~~It~~When~~-~~the timescale is PTP, it is possible to calculate local time from the time provided by a gPTP domain using the currentUtcOffset field value of the time properties data set and knowledge of the local time zone and whether and when daylight savings time is observed.

When the timescale is ARB, the value of currentUtcOffset cannot be used to compute UTC.

The mechanism for computing UTC does not change the synchronized time (see 3.21) of a time-aware system.

8.2.4 Measurement of time within a gPTP domain

Time in a gPTP domain shall be measured as elapsed time since the ~~PTP~~ epoch of the timescale of that domain.

8.3 Communication path asymmetry

This standard requires the measurement of the mean propagation time (also referred to as the *mean propagation delay*) between the ~~two endpoint~~ time-aware systems that comprise the two endpoints of a link. The measurement is performed by one of the time-aware systems~~,~~ (the initiator time-aware system), sending a message to the other time-aware system~~,~~ (the responder time-aware system). The responder then sends a message back to the initiator at a later time. The departure of the message sent by the initiator time-aware system is timestamped, and the timestamp value is retained by that system. The arrival of this message at the responder time-aware system is timestamped; the timestamp value is conveyed to the initiator time-aware system in a subsequent message. The departure of the response message sent by the responder time-aware system (in response to the message it receives from the initiator time-aware system) is timestamped, and the timestamp value is conveyed to the initiator time-aware system in a subsequent message. The arrival of this response message at the initiator time-aware system is timestamped, and the timestamp value is retained by that system. The mean propagation time is computed by the initiator time-aware system after receiving the response message, from the four timestamp values it has at this point.

Typically, the propagation time is not exactly the same in both directions, and the degree to which it differs in the two directions is characterized by the delay asymmetry. The relation between the individual propagation times in the two directions, the mean propagation time, and the delay asymmetry is as follows. Let t_{ir} be the propagation time from the initiator to the responder, t_{ri} be the propagation time from the responder to the initiator, meanPathDelay be the mean propagation time, and delayAsymmetry be the delay asymmetry. The propagation times in the two directions are illustrated in Figure 8-1.

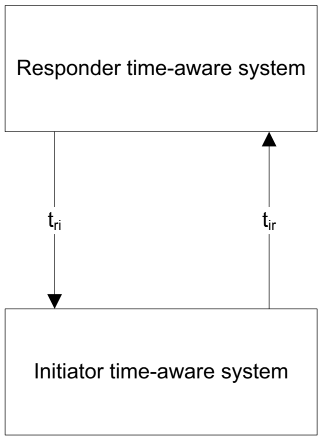


Figure 8-1—Propagation asymmetry

The meanPathDelay is the mean value of t_{ir} and t_{ri}, i.e., meanPathDelay = $(t_{ir} + t_{ri}) / 2$. The delayAsymmetry is defined as

t_{ir} = meanPathDelay – delayAsymmetry, and

t_{ri} = meanPathDelay + delayAsymmetry.

In other words, delayAsymmetry is defined to be positive when the responder to initiator propagation time is longer than the initiator to responder propagation time.

This standard does not explicitly require the measurement of delayAsymmetry; however, if delayAsymmetry is modeled, it shall be modeled as specified in this clause.

NOTE 1—A time-aware system can change the value of delayAsymmetry during operation using methods ~~not~~ specified in this standard.

<<Editor's note: In accordance with the resolution of comment #67 against D4.5, the word "not" is deleted from the above NOTE. It is requested that reviewers/balloters review this to ensure this was intended (e.g., that it was not a typo on the part of the editor).>>

NOTE 2—A time aware system port cannot measure the value of delayAsymmetry during live operation of the system (i.e., asymmetryMeasurementMode, see 10.2.4.2, is FALSE), so the value of delayAsymmetry must be defined separately using information from the supplier or additional testing before running the live system. The methods for measuring Asymmetry are not specified in this standard. These values can be added to the system configuration to improve the accuracy of time synchronization. The inaccuracy caused by asymmetry is half the value of the difference between t_{ri} and t_{ir}, and these inaccuracies can either accumulate over successive hops or, if the successive asymmetries have different signs, can cancel each other over the successive hops.

8.4 Messages

8.4.1 General

All communications occur via PTP messages and/or media-specific messages.

8.4.2 Message attributes

8.4.2.1 General

All messages used in this standard have the following attributes:

a) Message class
b) Message type

The message class attribute is defined in this clause. The message type attribute is defined in 3.14**.** Some messages have additional attributes; these are defined in the subclauses where the respective messages are defined.

8.4.2.2 Message class

There are two message classes, the event message class and the general message class. Event messages are timestamped on egress from a time-aware system and ingress to a time-aware system. General messages are not timestamped. Every message is either an event message or a general message.

8.4.3 Generation of event message timestamps

All event messages are timestamped on egress and ingress. The timestamp shall be the time, relative to the LocalClock entity (see 10.1) at which the message timestamp point passes the reference plane marking the boundary between the time-aware system and the network media.

The definition of the timestamp measurement plane (see 3.24), along with the corrections defined as follows, allows transmission delays to be measured in such a way (at such a low layer) that they appear fixed and symmetrical to gPTP even though the MAC client might otherwise observe substantial asymmetry and transmission variation. For example, the timestamp measurement plane is located below any retransmission and queuing performed by the MAC.

NOTE 1—If an implementation generates event message timestamps using a point other than the message timestamp point, then the generated timestamps should be appropriately corrected by the time interval (fixed or otherwise) between the actual time of detection and the time the message timestamp point passed the reference plane. Failure to make these corrections results in a time offset between time-aware systems.

NOTE 2—In general, the timestamps ~~may~~can be generated at a timestamp measurement plane that is removed from the reference plane. Furthermore, the timestamp measurement plane, and therefore the time offset of this plane from the reference plane, is likely to be different for inbound and outbound event messages. To meet the requirement of this clause, the generated timestamps should be corrected for these offsets. Figure 8-2 illustrates these offsets. Based on this model the appropriate corrections are as follows:

~~<~~egressTimestamp~~>~~ = ~~<~~egressMeasuredTimestamp~~>~~ + egressLatency

~~<~~ingressTimestamp~~>~~ = ~~<~~ingressMeasuredTimestamp~~>~~ − ingressLatency

where the timestamps relative to the reference plane, ~~<~~egressTimestamp~~>~~ and ~~<~~ingressTimestamp~~>~~, are computed from the timestamps relative to the timestamp measurement plane, ~~<~~egressMeasuredTimestamp~~>~~ and ~~<~~ingressMeasuredTimestamp~~>~~, respectively, using their respective latencies, egressLatency and ingressLatency. Failure to make these corrections results in a time offset between the slave and master clocks.

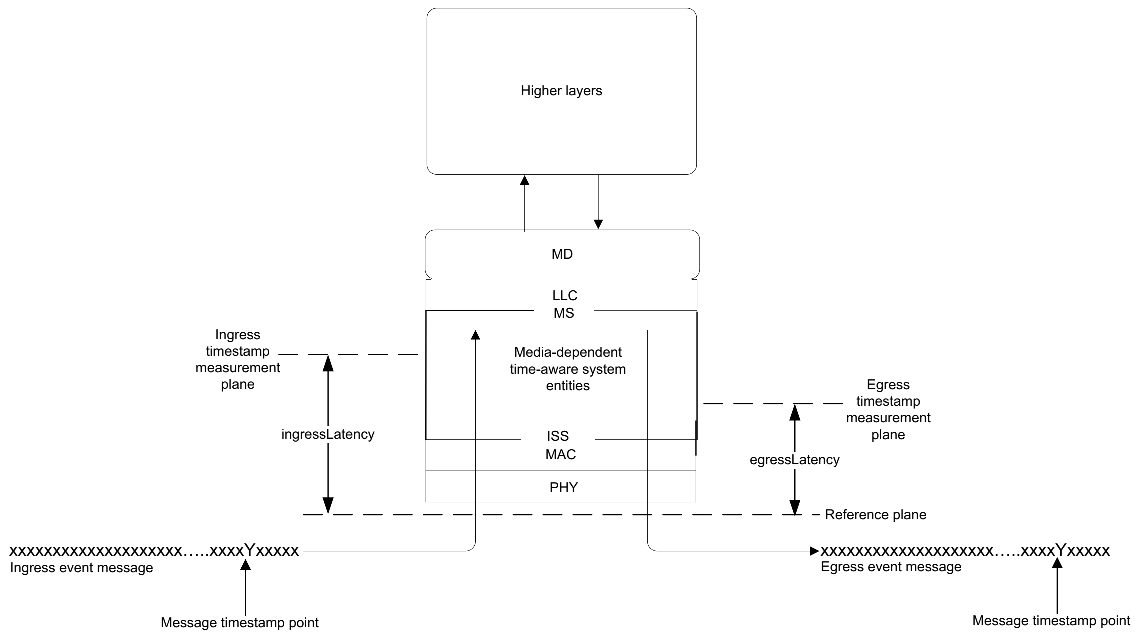

Higher layers
MD
LLC
MS
Media-dependent time-aware system entities
ISS
MAC
PHY
Ingress timestamp measurement plane
ingressLatency
Egress timestamp measurement plane
egressLatency
Reference plane
xxxxxxxxxxxxxxxxxxxx…..xxxxYxxxxx
Ingress event message
Message timestamp point
xxxxxxxxxxxxxxxxxxxx…..xxxxYxxxxx
Egress event message
Message timestamp point

Figure 8-2—Definition of message timestamp point, reference plane, timestamp measurement plane, and latency constants

8.4.4 Priorities

The ISS priority for frames carrying IEEE 802.1AS messages shall be 6.

NOTE—Frames carrying IEEE 802.1AS messages are neither VLAN-tagged nor priority-tagged, i.e., they are untagged, see 11.3.3.

8.5 Ports

8.5.1 General

The time-aware systems in a gPTP domain interface with the network media via physical ports. gPTP defines a logical port in such a way that communication between time-aware systems is point-to-point even over physical ports that are attached to shared media. One logical port, consisting of one PortSync entity and one MD entity, is instantiated for each time-aware system with which the time-aware system communicates. In the case of shared media, multiple logical ports can be associated with a single physical port.

Unless otherwise qualified, each instance of the term *port* refers to a *logical port.*

8.5.2 Port identity

8.5.2.1 General

A port is identified by a port identity of type PortIdentity, see 6.4.3.7. The value is maintained in the portIdentity member of the port parameter data set, see 14.8.2. A port identity consists of the following two attributes:

a) portIdentity.clockIdentity
b) portIdentity.portNumber

8.5.2.2 clockIdentity

8.5.2.2.1 General

The clockIdentity shall be an IEEE EUI-64 individual assigned number, as specified in this subclause. It is an 8-octet array, and is constructed from an MA-L (OUI), MA-M, or MA-S assignment from the IEEE Registration Authority (see [B2]).

NOTE 1—Implementers must consult the most recent IEEE rules on the use of OUIs (MA-Ls), MA-Ms, and MA-Ss to ensure conformance to IEEE policy (see tutorial (a) of [B2]).

When forming a clockIdentity value:

a) The OUI (MA-L), MA-M, or MA-S from which the clockIdentity is formed shall be owned by the organization creating the clockIdentity under the terms of this subclause.
b) If the clockIdentity is formed from an OUI, the organization owning the OUI shall ensure that the remaining 5 octets (i.e. 40 bits) of the EUI-64 are unique within the scope of clockIdentity values assigned by the organization.
c) If the clockIdentity is formed from an MA-M, the organization owning the MA-M shall ensure that the remaining 4 ½ octets (i.e., 36 bits) of the EUI-64 are unique within the scope of clockIdentity values assigned by the organization.
d) If the clockIdentity is formed from an MA-S, the organization owning the MA-S shall ensure that the remaining 3 ½ octets (i.e., 28 bits) of the EUI-64 are unique within the scope of clockIdentity values assigned by the organization.
e) The 8 octets of the IEEE EUI-64 shall be assigned in order to the 8-octet array clockIdentity with the most significant octet of the IEEE EUI-64 assigned to the clockIdentity octet array member with index 0.

Informative example (see tutorial (b) of [B2]):

The OUI for Company X is 0xACDE48. If Company X wished to generate a EUI-64 clockIdentity, a legal value would be as follows: 0xACDE48234567ABCD, where the 5-octet array $234567ABCD_{16}$ would be guaranteed by Company X to be unique among all Company X EUI-64 assigned numbers used as clockIdentity values. The byte and bit representations of the clockIdentity are illustrated below.

Table 8-1—Illustration of formation of clockIdentity from EUI-64

OUI			extension identifier				
octet[0] (most significant)	octet[1]	octet[2]	octet[3]	octet[4]	octet[5]	octet[6]	octet[7] (least significant)
0xAC	0xDE	0x48	0x23	0x45	0x67	0xAB	0xCD
10101100 (leftmost bit is most significant)	11011110b	01001000b	00100011b	01000101b	01100111b	10101011b	11001101b (rightmost bit is least significant

Implementers may alternatively use an EUI-48 to create the an EUI-64 clockIdentity by mapping an EUI-48 to an EUI-64.

NOTE 2—The IEEE has deprecated this mapping, i.e., the IEEE has indicated that this mapping should not be used in new applications (see tutorials (a) and (c) of [B2]). However, the mapping can continue to be used in existing standards that already use the mapping.

In this case:

f) The EUI-48 shall be an Ethernet MAC address owned by the organization creating the instance of a clockIdentity under the terms of this subclause. The organization owning the MAC address shall guarantee that the MAC address is used in generating only a single instance of a clockIdentity, for example, by requiring that the MAC address be a MAC address embedded in the device identified by the clockIdentity.
g) The mapping rules for constructing the EUI-64 from the EUI-48 shall be those specified by the IEEE Registration Authority (see tutorials (a) and (c) of [B2]).
h) The 8 octets of the created IEEE EUI-64 shall be assigned in order to the 8-octet array clockIdentity with the most significant octet of the IEEE EUI-64 assigned to the clockIdentity octet array member with index 0.

NOTE 3—When using an EUI-48, the first 3 octets of the EUI-48 are assigned in order to the first 3 octets of the clockIdentity, with the most significant octet of the EUI-64 assigned to the clockIdentity octet array member with index 0. Octets with index 3 and 4 have hexidecimal values FF and FE, respectively. The remaining 3 octets of the IEEE EUI-48 are assigned in order to the last 3 octets of the clockIdentity.

NOTE 4—The term MAC-48 identifier is now obsolete. The IEEE Registration Authority now uses the term EUI-48 to include the uses that were historically referred to by the term MAC-48, in addition to the uses that were already referred to by the term EUI-48 (see tutorial (a) of [B2]).

Informative example (see [B2]):

If Company X wished to use an EUI-48, based on the MAC address AC-DE-48-23-45-67 in an owned device, to form a clockIdentity, the resulting clockIdentity would be : 0xACDE48FFFE234567, where the 3 octet array 0x234567 would be ensured by Company X to be unique among all Company X assigned EUI-48 numbers. The byte and bit representations of the clockIdentity are illustrated in Table 8-2 below, based on the mapping as described by the IEEE Registration Authority (see [B2]).

~~The clockIdentity is an 8-octet array formed by mapping an IEEE EUI-48 assigned to the time-aware system to IEEE EUI-64 format (i.e., to an array of 8 octets). The EUI-48 shall be an Ethernet MAC address owned by the organization creating the instance of a clockIdentity under the terms of this subclause. The organization owning the MAC address shall ensure that the MAC address is used in generating only a single instance of a clockIdentity, for example by requiring that the MAC address be a MAC address embedded in the device identified by the clockIdentity. The mapping rules for constructing the EUI-64 from the EUI-48 shall be those specified by the IEEE (see [B2]). The 8 octets of the created IEEE EUI-64 shall be assigned in order to the 8-octet array clockIdentity with most significant octet of the IEEE EUI-64 assigned to the clockIdentity octet array member with index 0.~~

~~NOTE 1—When using an EUI-48, the first 3 octets, i.e., the OUI portion, of the IEEE EUI-48 are assigned in order to the first 3 octets of the clockIdentity with most significant octet of the IEEE EUI-64, i.e., the most significant octet of the OUI portion, assigned to the clockIdentity octet array member with index 0. Octets with index 3 and 4 have hex values 0xFF and 0xFE respectively. The remaining 3 octets of the IEEE EUI-48 are assigned in order to the last 3 octets of the clockIdentity (see [B2]).~~

~~NOTE 2—The least and next to least significant bits of the most significant octet of the OUI indicate respectively: whether the address is an individual or group address, and whether the address is administered universally by the IEEE or locally.~~

~~NOTE 3—The clockIdentity is used by this standard as a unique identifier and not as a network address.~~

~~NOTE 4—Informative example (see [B2]): The OUI for Company X is 0xACDE48. If Company X wished to use an EUI-48 assigned number of 0xACDE48234567 as part of the clockIdentity, the resulting clockIdentity would be: 0xACDE48FFFE234567, where the 3-octet array 0x234567 would be ensured by Company X to be unique among all Company X assigned EUI-48 numbers. The byte and bit representations are illustrated in Table 8-2, see [B2].~~

Table 8-2—Illustration of formation of clockIdentity from EUI-48

OUI			EUI-64~~48~~ assigned values		extension identifier		
octet[0] (most significant)	octet[1]	octet[2]	octet[3]	octet[4]	octet[5]	octet[6]	octet[7] (least significant)
0xAC	0xDE	0x48	0xFF	0xFE	0x23	0x45	0x67
10101100 (leftmost bit~~1~~ is most significant)	11011110b	01001000b	11111111b	11111110b	00100011b	01000101b	01100111b (rightmost bit~~1~~ is least significant)

Interpretations or functional behaviors dependent on a clockIdentity value, except for those interpretations and functional behaviors defined in this standard, shall not be permitted.

NOTE 5—The clockIdentity value is used by gPTP as a unique identifier and not as a network address. Although a network address might be inferred from the clockIdentity for technologies using the underlying IEEE assigned numbers for that purpose, such an interpretation is out of the scope of this standard.

8.5.2.2.2 Reserved clockIdentity value

The clockIdentity value consisting of all ones shall be reserved for designating all clocks in the gPTP domain.

8.5.2.3 Port number

The portNumber value for a port on a time-aware end station (i.e., a time-aware system supporting a single port) shall be in the range 1, 2, ..., 0xFFFE. The portNumber values for the ports on a time-aware ~~Bridge~~relay supporting N ports shall be distinct in the range 1, 2, …, ~~N~~0xFFFE~~, respectively~~.

The portNumber value 0 is assigned to the interface between the ClockMaster and ClockSource entities (see 10.1 and Figure 10-1.

8.5.2.4 Ordering of clockIdentiy and portIdentity values

Two clockIdentity values X and Y are compared as follows. Let *x* be the unsigned integer formed by concatenating octets 0 through 7 of X such that octet *j*+1 follows octet *j* (i.e., is less significant than octet *j*) in *x* (*j* = 0, 1, ..., 6~~7~~). Let *y* be the unsigned integer formed by concatenating octets 0 through 7 of Y such that octet *j*+1 follows octet *j* (i.e., is less significant than octet *j*) in *y* (*j* = 0, 1, ..., 6~~7~~). Then

a) X = Y if and only if $x = y$,
b) X > Y if and only if $x > y$, and
c) X < Y if and only if $x < y$.

Two portIdentity values A and B with members clockIdentity and portNumber are compared as follows. Let *a* be the unsigned integer formed by concatenating octets 0 through 7 of A.clockIdentity, such that octet j+1 follows octet j (i.e., is less significant than octet j) in *a* (j = 0, 1, ..., 6~~7~~), followed by octet 0 of A.portNumber, followed by octet 1 of A.portNumber. Let *b* be the unsigned integer formed by concatenating octets 0 through 7 of B.clockIdentity, such that octet j+1 follows octet j (i.e., is less significant than octet j) in *b* (j = 0, 1, ..., 6~~7~~), followed by octet 0 of B.portNumber, followed by octet 1 of B.portNumber. Then

d) A = B if and only if $a = b$,
e) A > B if and only if $a > b$, and
f) A < B if and only if $a < b$.

A portIdentity A with members clockIdentity and portNumber and a clockIdentity B are compared as follows. The unsigned integer *a* is formed from portIdentity A as described above. The unsigned integer *b* is formed by first forming a portIdentity B' whose clockIdentity is B and portNumber is 0. *b* is then formed from B' as described above. A and B are then compared as described in items d) through f) above.

8.6 Time-aware system characterization

8.6.1 Time-aware system type

There are two types of time-aware systems used in a gPTP domain, as follows:

a) time-aware end station
b) time-aware ~~Bridge~~relay

All time-aware systems are identified by clockIdentity.

In addition, time-aware systems are characterized by the following attributes:

c) priority1
d) clockClass
e) clockAccuracy
f) offsetScaledLogVariance
g) priority2
h) clockIdentity
i) timeSource
j) numberPorts

NOTE—Attributes c) through i) can be considered to be associated with the ClockMaster entity of the time-aware system.

8.6.2 Time-aware system attributes

8.6.2.1 priority1

priority1 is used in the execution of the BMCA, see 10.3. The value of priority1 is an integer selected from the range 0 through 255. The ordering of priority1 in the operation of the BMCA, see 10.3.4 and 10.3.5, is specified as follows. A ClockMaster A shall be deemed better than a ClockMaster B if the value of priority1 of A is numerically less than that of B.

The value of priority1 shall be 255 for a time-aware system that is not grandmaster-capable. The value of priority1 shall be less than 255 for a time-aware system that is grandmaster-capable. The value 0 shall be reserved for management use, i.e., the value of priority1 shall be set to 0 only via management action and shall not be specified as a default value by a user of this standard. In the absence of a default value set by a user of this standard, the default value ~~shall~~should be set as indicated in Table 8-3. These default values are

Table 8-3—Default values for priority1, for the respective media

System type	Default value for priority1
~~Network infrastructure time-aware system~~Devices that are a central/critical part of the network and are not expected to ever be turned off during normal network operation. For example, these include Bridges, Wireless Access Points, and grandmaster-specific end nodes. (End node grandmasters are not a 'central' part of the network, but they are 'critical' to this gPTP operation)	246
~~Portable time-aware system~~Devices that can be turned off at any time, so they cannot be considered a central/critical part of the network. When these devices are turned off, they only affect the function(s) they support and do not affect any other functions of the network. For example, these include desktop computers, fixed (heavy or otherwise) end nodes, speakers, receivers, amplifiers, and televisions.	~~250~~248
~~Other time-aware systems~~Devices that can go away (physically or otherwise) at any time. This includes devices that are designed to be transient to the network. For example, these include laptop computers, cell phones, and battery powered speakers.	~~248~~250
Time-aware system that is not grandmaster-capable	255

NOTE 1—Care must be applied to multi-function device design, specifically for an end station that also contains a Bridge. The Bridge function inside multi-function devices must not be powered down when the end station function is powered down, or else the data and controls (like gPTP) passing through the Bridge function to other network devices will cease. Example devices of this are a television and/or amplifier/receiver that also contains a Bridge.

NOTE 2—The BMCA, see 10.3, considers priority1 before other attributes; the priority1 attribute can therefore be used to force a desired ordering of time-aware systems for best master selection.

NOTE 3—The settings for priority1 in Table 8-3 guarantee that a time-aware system that is grandmaster-capable is always preferred by the BMCA over a time-aware system that is not grandmaster-capable.

NOTE 4—These values are assigned so that devices with priority1 value of 246 are selected as grandmaster over devices with priority1 values of 248 or 250 (devices with priority1 value of 255 are never selected as grandmaster).

suitable for applications in which the availability of the grandmaster is the most important criterion for grandmaster selection. A device built for a specific application for which this is not the case should be capable of having priority1 changed via management.

~~NOTE 1—The BMCA, see 10.3, considers priority1 before other attributes; the priority1 attribute can therefore be used to force a desired ordering of time-aware systems for best master selection.~~

~~NOTE 2—The previous settings for priority1 guarantee that a time-aware system that is grandmaster-capable is always preferred by the BMCA to a time-aware system that is not grandmaster-capable.~~

~~NOTE 3—These values are assigned so that fixed devices that are available are selected as grandmaster over those devices that are more likely to be removed or powered down.~~

8.6.2.2 clockClass

The clockClass attribute denotes the traceability of the ~~synchronized time~~synchronized time distributed by a ClockMaster when it is the grandmaster.

The value shall be selected as follows:

a) If the Default Parameter Data Set member gmCapable is TRUE, then
   1) clockClass is set to the value that reflects the combination of the LocalClock and ClockSource entities; else
   2) if the value that reflects the LocalClock and ClockSource entities is not specified or not known, clockClass is set to 248;
b) If the Default Parameter Data Set member gmCapable is FALSE (see 8.6.2.1), clockClass is set to 255.

See 7.6.2.4 of IEEE Std 1588-2008 for a more detailed description of clockClass.

NOTE—The time-aware system has a LocalClock entity, which ~~may~~can be the free-running quartz crystal that just meets the IEEE 802.3 requirements, but could also be better. There can be a ClockSource entity, e.g., timing taken from ~~GPS~~a GNSS, available in the local system that provides timing to the ClockSource entity. The time provided by the time-aware system, if it is the grandmaster, is reflected by the combination of these two entities, and the clockClass ~~should~~ reflects this combination as specified in 7.6.2.4 of IEEE Std 1588-2008. For example, when the LocalClock entity uses a quartz oscillator that meets the requirements of IEEE Std 802.3~~-2008~~ and B.1 of this standard, clockClass ~~may be~~is set to 248. But, if a ~~GPS~~GNSS receiver is present and synchronizes the time-aware system, then the clockClass ~~may be~~is set to the value 6, indicating traceability to a primary reference time source (see 7.6.2.4 of IEEE Std 1588-2008).

8.6.2.3 clockAccuracy

The clockAccuracy attribute indicates the expected time accuracy of a ClockMaster.

The value shall be selected as follows:

a) clockAccuracy is set to the value that reflects the combination of the LocalClock and ClockSource entities; else
b) if the value that reflects the LocalClock and ClockSource entities is not specified or unknown, clockAccuracy is set to 254 (FE_{16}).

See 7.6.2.5 of IEEE Std 1588-2008 for more detailed description of clockAccuracy.

8.6.2.4 offsetScaledLogVariance

The offsetScaledLogVariance is a scaled, offset representation of an estimate of the PTP variance. The PTP variance characterizes the precision and frequency stability of the ClockMaster. The PTP variance is the square of PTPDEV (see B.1.3.2).

The value shall be selected as follows:

a) offsetScaledLogVariance is set to the value that reflects the combination of the LocalClock and ClockSource entities; else
b) if the value that reflects these entities is not specified or not known, offsetScaledLogVariance is set to 17258 ($436A_{16}$). This value corresponds to the value of PTPDEV for observation interval equal to the default Sync message transmission interval (i.e., observation interval of 0.125 s, see 11.5.2.3 and B.1.3.2).

See 7.6.3 of IEEE Std 1588-2008 for more detailed description of PTP variance and offsetScaledLogVariance (7.6.3.3 of IEEE Std 1588-2008 provides a detailed description of the computation of offsetScaledLogVariance from PTP variance, along with an example).

8.6.2.5 priority2

priority2 is used in the execution of the BMCA, see 10.3. The value of priority2 shall be an integer selected from the range 0 through 255. The ordering of priority2 in the operation of the BMCA is the same as the ordering of priority1, see 8.6.2.1.

The default value of priority2 shall be 248. See 7.6.2.3 of IEEE Std 1588-2008 for a more detailed description of priority2.

8.6.2.6 clockIdentity

The clockIdentity value for a time-aware system shall be as specified in 8.5.2.2.

8.6.2.7 timeSource

The timeSource is an information only attribute indicating the type of source of time used by a ClockMaster. The value is not used in the selection of the grandmaster. The values shall be as specified in Table 8-4. These represent categories. For example, the GPS entry ~~would~~ includes not only the GPS system of the U.S. Department of Defense but the European Galileo system and other present and future ~~satellite-based timing systems~~GNSSs.

All unused values are reserved.

See 7.6.2.6 of IEEE Std 1588-2008 for a more detailed description of timeSource.

The initialization value is selected as follows:

a) If the timeSource (8.6.2.7 and Table 8-4) is known at the time of initialization, the value is derived from the table, else
b) The value is set to $A0_{16}$ (INTERNAL_OSCILLATOR).

8.6.2.8 numberPorts

The numberPorts indicates the number of ports on the time-aware system.

Table 8-4—timeSource enumeration

Value	Time source	Description
0x10	ATOMIC_CLOCK	Any device, or device directly connected to such a device, that is based on atomic resonance for frequency and that has been calibrated against international standards for frequency and time
0x20	GPS NOTE—In this standard, this value refers to any GNSS (i.e., not only GPS).	Any device synchronized to any of the satellite systems that distribute time and frequency tied to international standards
0x30	TERRESTRIAL_RADIO	Any device synchronized via any of the radio distribution systems that distribute time and frequency tied to international standards
0x40	PTP	Any device synchronized to an IEEE 1588 PTP-based source of time external to the gPTP domain. See NOTE.
0x50	NTP	Any device synchronized via NTP to servers that distribute time and frequency tied to international standards
0x60	HAND_SET	Used in all cases for any device whose time has been set by means of a human interface based on observation of an international standards source of time to within the claimed clock accuracy
0x90	OTHER	Any source of time and/or frequency not covered by other values, or for which the source is not known
0xA0	INTERNAL_OSCILLATOR	Any device whose frequency is not based on atomic resonance nor calibrated against international standards for frequency, and whose time is based on a free-running oscillator with epoch determined in an arbitrary or unknown manner

NOTE—For example, a clock that implements both ~~the~~a gPTP domain and ~~another~~a separate IEEE 1588 (i.e., PTP) domain, and is synchronized by the ~~other~~separate IEEE 1588 domain, would have time source of PTP in the gPTP domain.

9. Application interfaces

9.1 Overview of the interfaces

Unless otherwise stated, information in this clause is per domain.

The following subclauses define one application interface between the ClockSource entity and ClockMaster entity (see 10.1.1) and four application interfaces between the ClockTarget entity and ClockSlave entity (see 10.1.1). The ClockSource is an entity that can be used as an external timing source for the gPTP domain. The ClockSource entity either contains or has access to a clock (see 3.3). The ClockTarget entity represents any application that uses information provided by the ClockSlave entity via any of the application interfaces.

NOTE—The manner in which the ClockSource entity obtains time from a clock is outside the scope of this standard. The manner in which the ClockTarget uses the information provided by application interfaces is outside the scope of this standard.

The five interfaces are illustrated in Figure 9-1. They include the following:

a) the ClockSourceTime interface, which provides external timing to a time-aware system,
b) the ClockTargetEventCapture interface, which returns the synchronized time of an event signaled by a ClockTarget entity,
c) the ClockTargetTriggerGenerate interface, which causes an event to be signaled at a synchronized time specified by a ClockTarget entity,
d) the ClockTargetClockGenerator interface, which causes a periodic sequence of results to be generated, with a phase and rate specified by a ClockTarget entity, and
e) the ClockTargetPhaseDiscontinuity interface, which supplies information that an application can use to determine if a discontinuity in grandmaster phase or frequency has occurred.

NOTE—The application interfaces described in this clause are models for behavior and not application program interfaces. Other application interfaces besides items a) through e) above are possible, but are not described here. In addition, there can be multiple instances of a particular interface.

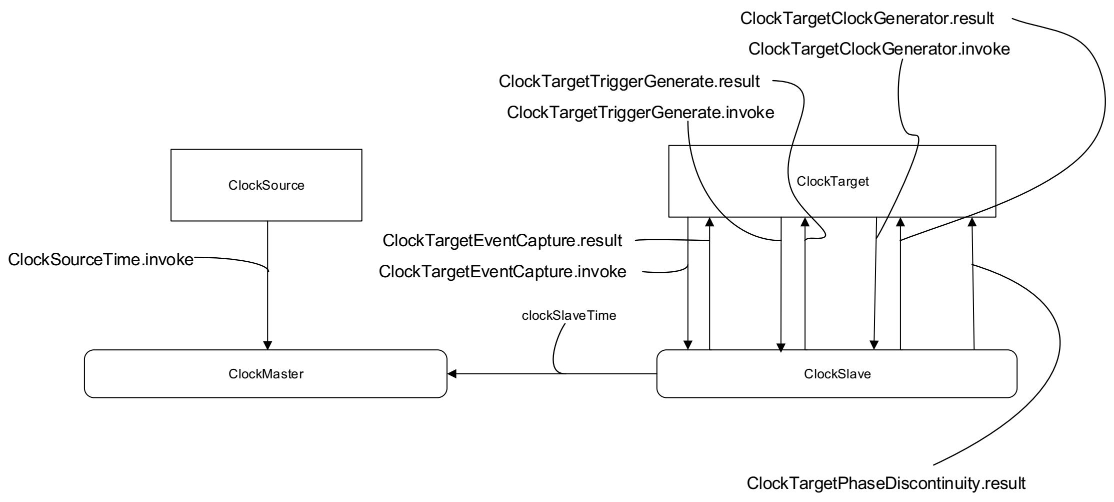

ClockTargetClockGenerator.result
ClockTargetClockGenerator.invoke
ClockTargetTriggerGenerate.result
ClockTargetTriggerGenerate.invoke
ClockSource
ClockTarget
ClockTargetEventCapture.result
ClockTargetEventCapture.invoke
ClockSourceTime.invoke
clockSlaveTime
ClockMaster
ClockSlave
ClockTargetPhaseDiscontinuity.result

Figure 9-1—Application interfaces

9.2 ClockSourceTime interface

9.2.1 General

This interface is used by the ClockSource entity to provide time to the ClockMaster entity of a time-aware system. The ClockSource entity invokes the ClockSourceTime.invoke function to provide the time, relative to the ClockSource, that this function is invoked.

9.2.2 ClockSourceTime.invoke function parameters

```
ClockSourceTime.invoke {
        domainNumber,
        sourceTime,
        timeBaseIndicator,
        lastGmPhaseChange,
        lastGmFreqChange
}
```

The parameter definitions are as follows:

9.2.2.1 domainNumber (UInteger8)

The domain number of the gPTP domain that this ClockSource entity is providing time to.

9.2.2.2 sourceTime (ExtendedTimestamp)

The value of sourceTime is the time this function is invoked by the ClockSource entity.

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 time-aware system can enter holdover, the ClockSource entity itself is not lost. The ClockSource entity ensures that timeBaseIndicator changes if the source of time is lost.

9.2.2.4 lastGmPhaseChange (ScaledNs)

The value of lastGmPhaseChange is the phase change (i.e., change in sourceTime) that occurred on the most recent change in timeBaseIndicator. The value is initialized to 0.

9.2.2.5 lastGmFreqChange (Double)

The value of lastGmFreqChange is the fractional frequency change (i.e., frequency change expressed as a pure fraction) that occurred on the most recent change in timeBaseIndicator. The value is initialized to 0.

9.3 ClockTargetEventCapture interface

9.3.1 General

This interface is used by the ClockTarget entity to request the synchronized time of an event that it signals to the ClockSlave entity of a time-aware system. The ClockTarget entity invokes the

ClockTargetEventCapture.invoke function to signal an event to the ClockSlave entity. The ClockSlave entity invokes the ClockTargetEventCapture.result function to return the time of the event relative to the current grandmaster or, if no time-aware system is grandmaster-capable, the LocalClock. The ClockTargetEventCapture.result function also returns gmPresent, to indicate to the ClockTarget whether or not a grandmaster is present.

9.3.2 ClockTargetEventCapture.invoke parameters

```
ClockTargetEventCapture.invoke {
        domainNumber
}
```

The~~ function contains no~~ parameter~~s~~ definitions are:~~.~~

9.3.2.1 domainNumber (UInteger8)

The domain number of the ClockSlave entity that is requested to provide the synchronized time of the signaled event.

9.3.3 ClockTargetEventCapture.result parameters

```
ClockTargetEventCapture.result {
        domainNumber,
        slaveTimeCallback,
        gmPresent
}
```

The parameter definitions are:

9.3.3.1 domainNumber (UInteger8)

The domain number of the ClockSlave entity that is requested to provide the synchronized time of the signaled event.

9.3.3.2 slaveTimeCallback (ExtendedTimestamp)

The value of slaveTimeCallback is the time, relative to the grandmaster, 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 10.2.3.13). This parameter indicates to the ClockTarget whether or not a grandmaster is present.

9.4 ClockTargetTriggerGenerate interface

9.4.1 General

This interface is used by the ClockTarget entity to request that the ClockSlave entity send a result at a specified time relative to the grandmaster. The ClockTarget entity invokes the

ClockTargetTriggerGenerate.invoke function to indicate the synchronized time of the event. The ClockSlave entity invokes the ClockTargetTriggerGenerate.result function to either signal the event at the requested synchronized time or indicate an error condition.

9.4.2 ClockTargetTriggerGenerate.invoke parameters

```
ClockTargetTriggerGenerate.invoke {
        domainNumber,
        slaveTimeCallback
}
```

The parameter definition is:

9.4.2.1 domainNumber (UInteger8)

The domain number of the ClockSlave entity that is requested to signal an event at the specified time.

9.4.2.2 slaveTimeCallback (ExtendedTimestamp)

If slaveTimeCallback is nonzero, its value is the synchronized time the corresponding ClockTargetTriggerGenerate.result function, i.e., the trigger, is to be invoked. If slaveTimeCallback is zero, any previous ClockTargetTriggerGenerate.invoke function for which a ClockTargetTriggerGenerate.result function has not yet been issued is canceled.

9.4.3 ClockTargetTriggerGenerate.result parameters

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

The parameter definitions are:

9.4.3.1 domainNumber (UInteger8)

The domain number of the ClockSlave entity that is requested to signal 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, 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 10.2.3.13). This parameter indicates to the ClockTarget whether or not a grandmaster is present.

9.4.4 ClockTargetTriggerGenerate interface definition

The invocation of the ClockTargetTriggerGenerate.invoke function causes the ClockSlave entity to store the value of the slaveTimeCallback parameter in an internal variable (replacing any previous value of that variable) until the synchronized time, or LocalClock time if gmPresent is FALSE, equals the value of that variable, at which time the ClockTargetTriggerGenerate.result function is invoked with errorCondition = FALSE. If it is not possible to invoke the ClockTargetTriggerGenerate.result function at slaveTimeCallback, e.g., if slaveTimeCallback is earlier than the synchronized time (or LocalClock time if gmPresent is FALSE) when the ClockTargetTriggerGenerate.invoke function is invoked, the ClockTargetTriggerGenerate.result function is invoked with errorCondition = TRUE. Invocation of the ClockTargetTriggerGenerate.invoke function with slaveTimeCallback = 0 (which is earlier than any synchronized time) is used to cancel a pending request.

9.5 ClockTargetClockGenerator interface

9.5.1 General

This interface is used by the ClockTarget entity to request that the ClockSlave entity deliver a periodic clock signal of specified period and phase. The ClockTarget entity invokes the ClockTargetClockGenerator.invoke function to request that the ClockSlave entity generate the periodic clock signal. The ClockSlave entity invokes the ClockTargetClockGenerator.result function at significant instants of the desired clock signal.

9.5.2 ClockTargetClockGenerator.invoke parameters

```
ClockTargetClockGenerator.invoke {
        domainNumber,
        clockPeriod,
        slaveTimeCallbackPhase
}
```

The parameter definitions are:

9.5.2.1 domainNumber (UInteger8)

The domain number of the ClockSlave entity that is requested to deliver a periodic clock signal.

9.5.2.2 clockPeriod (TimeInterval)

The value of clockPeriod is the period between successive invocations of the ClockTargetClockGenerator.result function. A value that is zero or negative causes any existing periodic clock signal generated via this application interface to be terminated.

9.5.2.3 slaveTimeCallbackPhase (ExtendedTimestamp)

The value of slaveTimeCallbackPhase describes phase of the generated clock signal by specifying a point on the ~~PTP~~ timescale in use such that ClockTargetClockGenerator.result invocations occur at s that differ from slaveTimeCallbackPhase by $n \times$ clockPeriod, where n is an integer.

NOTE—The value of slaveTimeCallbackPhase can be earlier or later than the synchronized time the ClockTargetClockGenerator.invoke function is invoked; use of a slaveTimeCallbackPhase value in the future does not imply that the initiation of the periodic clock signal is suppressed until that synchronized time.

9.5.3 ClockTargetClockGenerator.result parameters

The semantics of the function are:

```
ClockTargetClockGenerator.result {
        domainNumber,
        slaveTimeCallback,
}
```

The parameter definition is:

9.5.3.1 domainNumber (UInteger8)

The domain number of the ClockSlave entity that is requested to deliver a periodic clock signal.

9.5.3.2 slaveTimeCallback (ExtendedTimestamp)

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

9.6 ClockTargetPhaseDiscontinuity interface

9.6.1 General

This interface provides discontinuity information, sent from the grandmaster, to an application within a station. It is used by the ClockSlave entity to supply sufficient information to the ClockTarget entity to enable the ClockTarget entity to determine whether a phase or frequency discontinuity has occurred. The ClockSlave invokes the ClockTargetPhaseDiscontinuity.result function in the SEND_SYNC_INDICATION block of the ClockSlaveSync state machine (see 10.2.12 and Figure 10-9). The invocation occurs when a PortSyncSync structure is received, after the needed information has been computed by the ClockSlaveSync state machine.

9.6.2 ClockTargetPhaseDiscontinuity.result parameters

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

The parameter definitions are:

9.6.2.1 domainNumber (UInteger8)

The domain number of the ClockSlave entity that is providing discontinuity information.

9.6.2.2 gmIdentity (ClockIdentity)

If gmPresent (see 10.2.3.13) is TRUE, the value of gmIdentity is the ClockIdentity of the current grandmaster. 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.

9.6.2.4 lastGmPhaseChange (ScaledNs)

The value of the global lastGmPhaseChange parameter (see 10.2.3.16) received from the grandmaster.

9.6.2.5 lastGmFreqChange (Double)

The value of lastGmFreqChange parameter (see 10.2.3.17) received from the grandmaster.

10. Media-independent layer specification

10.1 Overview

Unless otherwise stated, information in this clause is per domain.

10.1.1 Model of operation

A time-aware system contains a best master selection function and a synchronization function. These functions include port-specific aspects and aspects associated with the time-aware system as a whole. The functions are distributed among a number of entities, which together describe the behavior of a compliant implementation. The functions are specified by a number of state machines.

The model for the media-independent layer of a time-aware system is shown in Figure 10-1. It includes a single SiteSync entity, ClockMaster entity, and ClockSlave entity for the time-aware system as a whole, plus one PortSync and one MD entity for each port. The MD entity performs media-dependent functions, which are described in the clauses for the respective media. In addition to the entities, Figure 10-1 shows the information that flows between the entities via the PortSyncSync, MDSyncSend, and MDSyncReceive structures (see 10.2.2.3, 10.2.2.1, and 10.2.2.2, respectively).

The SiteSync, ClockMaster, ClockSlave, and PortSync entities each contain a number of cooperating state machines, which are described later in this clause (the MD entity state machines are described in the respective media-dependent clauses). The ClockMaster entity receives information from an external time source, referred to as a ClockSource entity (see 9.2), via an application interface, and provides the information to the SiteSync entity. The ClockSlave entity receives grandmaster time-synchronization and current grandmaster information from the SiteSync entity, and makes the information available to an external application, referred to as a clockTarget entity (see 9.3 through 9.6), via one or more application service interfaces. The SiteSync entity executes the portion of best master clock selection associated with the time-aware system as a whole, i.e., it uses the best master information received on each port to determine which port has received the best information, and updates the ~~role~~states of all the ports (see 10.3.1 for a discussion of port ~~role~~states). It also distributes synchronization information received on the SlavePort to all the ports whose ~~role~~state is MasterPort (see 10.3.1). The PortSync entity for a SlavePort receives best master selection information from the time-aware system at the other end of the associated link, compares this to the current best master information that it has, and forwards the result of the comparison to the Site Sync entity. The PortSync entity for a SlavePort also receives time-synchronization information from the MD entity associated with the port, and forwards it to the SiteSync entity. The PortSync entity for a MasterPort sends best master selection and time-synchronization information to the MD entity for the port, which in turn sends the respective messages.

NOTE—This clause does not require a one-to-one correspondence between the PortSync entities of time-aware systems attached to the same gPTP communication path (see 3.9), i.e., more than two time-aware systems can be attached to a gPTP communication path that uses a shared medium and meet the requirements of this clause. However, it is possible for a media-dependent clause to have additional requirements that limit the gPTP communication paths to point-to-point links for that medium; in this case, each link has exactly two PortSync entities, which can be considered to be in one-to-one correspondence. One example of this is the full-duplex, point-to-point media-dependent layer specified in Clause 11. In addition, one or more gPTP communication paths can be logically point-to-point but traverse the same shared medium.

The LocalClock entity is a free-running clock (see 3.3) that provides a common time to the time-aware system, relative to an arbitrary epoch. A time-aware system contains a LocalClock entity. The requirements for the LocalClock entity are specified in B.1. All timestamps are taken relative to the LocalClock entity (see 8.4.3). The LocalClock entity also provides the value of currentTime (see 10.2.3.12), which is used in the state machines to specify the various timers.

NOTE—The epoch for the LocalClock entity ~~may~~can be the time that the time-aware system is powered on.

The time-synchronization state machines are described in 10.2. The best master clock selection state machines are described in 10.3. The attributes and format of the Announce message are described in 10.5 and 10.6. The timing characterization of the protocol is described in 10.7.

10.1.2 Grandmaster-capable time-aware system

A time-aware system may be grandmaster-capable. An implementation may ~~optionally~~ provide the ability to configure a time-aware system as grandmaster-capable via a management interface.

NOTE—The managed object gmCapable is read only (see Table 14-1). gmCapable is configured by setting the value of the managed object priority1, which is read/write (see Table 14-1). If the value of priority1 is 255, then:

a) gmCapable is set to FALSE (see 8.6.2.1), and
b) the value of the managed object clockClass, which is read only, is set to 255 (see 8.6.2.2).

NOTE—While a time-aware system that is not grandmaster-capable can never be the grandmaster of the gPTP domain, such a time-aware system contains a best master selection function, invokes the best master selection algorithm, and conveys synchronization information received from the current grandmaster.

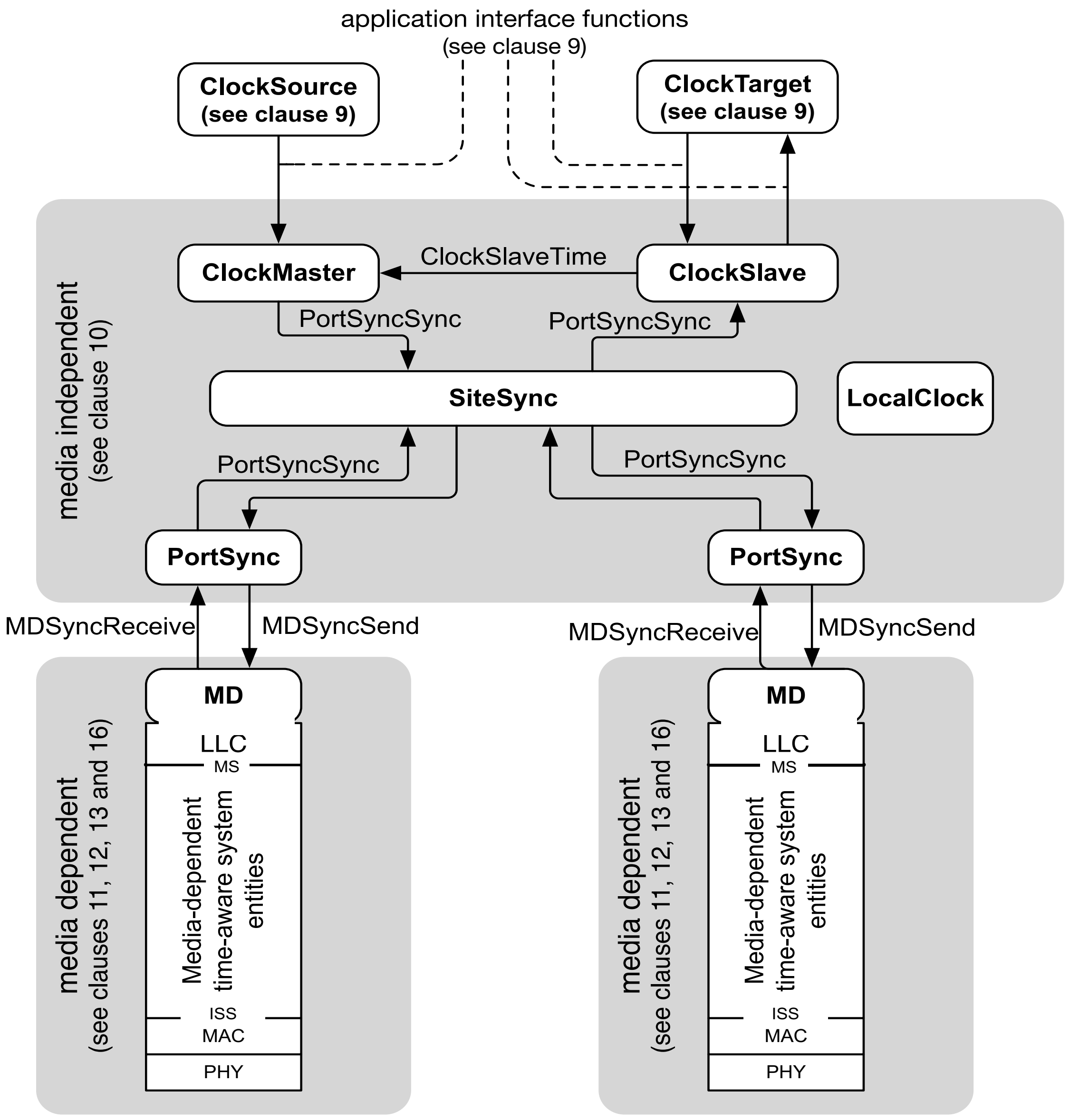

application interface functions
(see clause 9)
ClockSource (see clause 9)
ClockTarget (see clause 9)
ClockMaster
ClockSlaveTime
ClockSlave
PortSyncSync
PortSyncSync
SiteSync
LocalClock
PortSyncSync
PortSyncSync
media independent (see clause 10)
PortSync
PortSync
MDSyncReceive
MDSyncSend
MDSyncReceive
MDSyncSend
MD
LLC
MS
Media-dependent time-aware system entities
ISS
MAC
PHY
media dependent (see clauses 11, 12, 13 and 16)
MD
LLC
MS
Media-dependent time-aware system entities
ISS
MAC
PHY
media dependent (see clauses 11, 12, 13 and 16)

Figure 10-1—Model for media-independent layer of time-aware system

10.2 Time-synchronization state machines

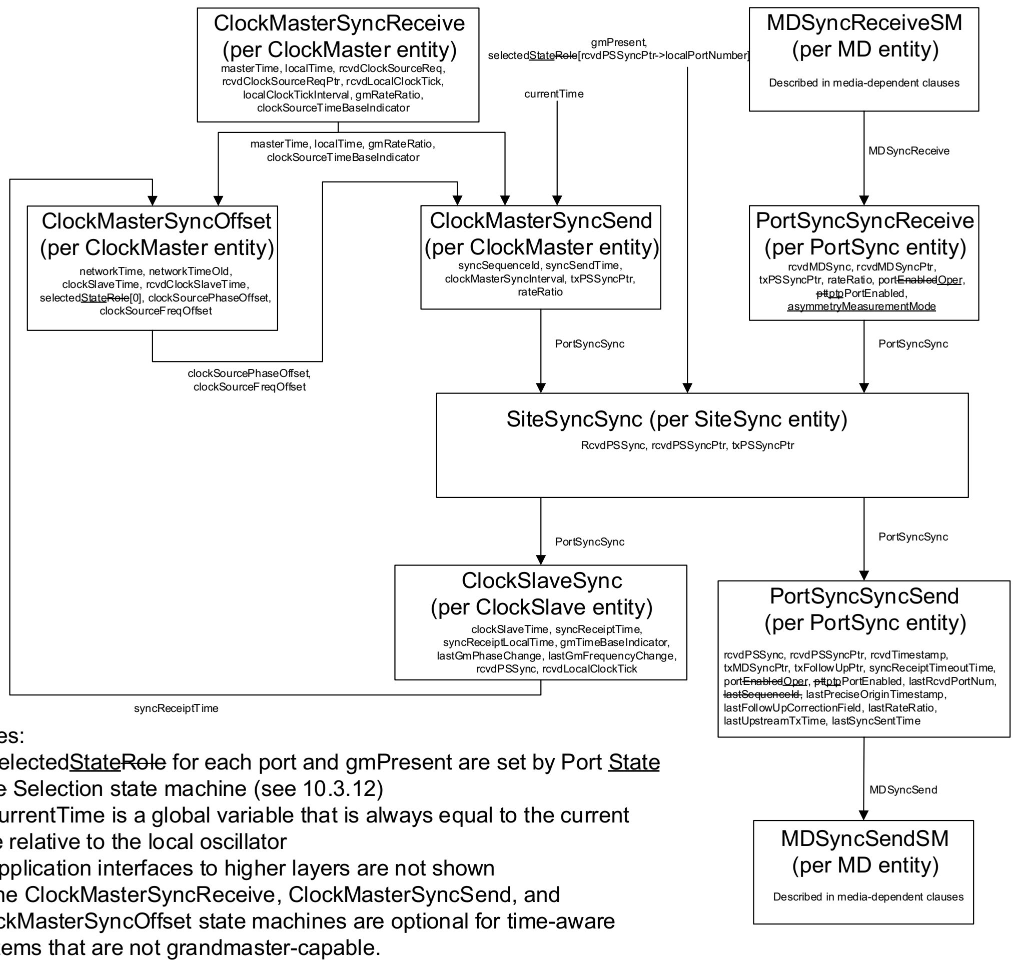
10.2.1 Overview

The time-synchronization function in a time-aware system is specified by a number of cooperating state machines. Figure 10-2 is not itself a state machine, but illustrates the machines, their interrelationships, the principle variables and structures used to communicate between them, their local variables, and performance parameters. Figure 10-2 includes state machines that are described in the media-dependent clauses of this standard, in order to illustrate the interrelationships between these state machines and the media-independent layer state machines described in this clause. Figure 10-2 does not show the application interface functions described in Clause 9, nor the service interface primitives between the media-dependent layer and the LLC.

The ClockMasterSyncReceive, ClockMasterSyncOffset, and ClockMasterSyncSend state machines are optional for time-aware systems that are not grandmaster-capable (see 8.6.2.1 and 10.1.2). These state machines may be present in a time-aware system 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 time-aware system is not grandmaster-capable.

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

a) ClockMasterSyncReceive (one instance per time-aware system): receives ClockSourceTime.invoke functions from the ClockSource entity and notifications of LocalClock entity ticks (see 10.2.3.18), updates masterTime, and provides masterTime to ClockMasterSyncOffset and ClockMasterSyncSend state machines. This state machine is optional for time-aware systems that are not grandmaster-capable~~ (see 8.6.2.1 and 10.1.2)~~.
b) ClockMasterSyncOffset (one instance per time-aware system): 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 time-aware system is not the grandmaster, and provides the frequency and phase offsets to the ClockMasterSyncSend state machine. This state machine is optional for time-aware systems that are not grandmaster-capable~~ (see 8.6.2.1 and 10.1.2)~~.
c) ClockMasterSyncSend (one instance per time-aware system): receives masterTime from the ClockMasterSyncReceive state machine, receives phase and frequency offset between masterTime and syncReceiptTime from the ClockMasterSyncOffset state machine, and provides masterTime (i.e., synchronized time) and the phase and frequency offset to the SiteSync entity using a PortSyncSync structure. This state machine is optional for time-aware systems that are not grandmaster-capable~~ (see 8.6.2.1 and 10.1.2)~~.
d) PortSyncSyncReceive (one instance per port): receives time-synchronization information from the MD entity of the corresponding port, computes accumulated rateRatio, computes syncReceiptTimeoutTime, and sends the information to the SiteSync entity.
e) SiteSyncSync (one instance per time-aware system): 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 port): receives time-synchronization information from the SiteSync entity, requests that the MD entity of the corresponding 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 time-aware system): receives time-synchronization information from the SiteSync entity, computes clockSlaveTime and syncReceiptTime, sets syncReceiptLocalTime, GmTimeBaseIndicator, lastGmPhaseChange, 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.
h) MDSyncReceiveSM (one instance per port): See 11.2.1, item (a) for a brief description of this state machine.
i) MDSyncSendSM (one instance per port): See 11.2.1, item (b) for a brief description of this state machine.


ClockMasterSyncReceive (per ClockMaster entity)
masterTime, localTime, rcvdClockSourceReq, rcvdClockSourceReqPtr, rcvdLocalClockTick, localClockTickInterval, gmRateRatio, clockSourceTimeBaseIndicator
gmPresent, selectedState~~Role~~[rcvdPSSyncPtr->localPortNumber]
currentTime
MDSyncReceiveSM (per MD entity)
Described in media-dependent clauses
masterTime, localTime, gmRateRatio, clockSourceTimeBaseIndicator
MDSyncReceive
ClockMasterSyncOffset (per ClockMaster entity)
networkTime, networkTimeOld, clockSlaveTime, rcvdClockSlaveTime, selectedState~~Role~~[0], clockSourcePhaseOffset, clockSourceFreqOffset
ClockMasterSyncSend (per ClockMaster entity)
syncSequenceId, syncSendTime, clockMasterSyncInterval, txPSSyncPtr, rateRatio
PortSyncSyncReceive (per PortSync entity)
rcvdMDSync, rcvdMDSyncPtr, txPSSyncPtr, rateRatio, port~~Enabled~~Oper, ~~ptt~~ptpPortEnabled, asymmetryMeasurementMode
PortSyncSync
PortSyncSync
clockSourcePhaseOffset, clockSourceFreqOffset
SiteSyncSync (per SiteSync entity)
RcvdPSSync, rcvdPSSyncPtr, txPSSyncPtr
PortSyncSync
PortSyncSync
ClockSlaveSync (per ClockSlave entity)
clockSlaveTime, syncReceiptTime, syncReceiptLocalTime, gmTimeBaseIndicator, lastGmPhaseChange, lastGmFrequencyChange, rcvdPSSync, rcvdLocalClockTick
PortSyncSyncSend (per PortSync entity)
rcvdPSSync, rcvdPSSyncPtr, rcvdTimestamp, txMDSyncPtr, txFollowUpPtr, syncReceiptTimeoutTime, port~~Enabled~~Oper, ~~ptt~~ptpPortEnabled, lastRcvdPortNum, ~~lastSequenceId,~~ lastPreciseOriginTimestamp, lastFollowUpCorrectionField, lastRateRatio, lastUpstreamTxTime, lastSyncSentTime
syncReceiptTime
MDSyncSend
MDSyncSendSM (per MD entity)
Described in media-dependent clauses

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

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

<<Editor's note: In Figure 10-2, portEnabled is changed to portOper, and pttPortEnabled is changed to ptpPortEnabled.>>

10.2.2 Data structures communicated between state machines

The following subclauses describe the data structures communicated between the time-synchronization state machines.

10.2.2.1 MDSyncSend

10.2.2.1.1 General

This structure contains information that is sent by the PortSync entity of a port to the MD entity of that port when requesting that the MD entity cause time-synchronization information to be sent. The structure contains information that reflects the most recent time-synchronization information received by this time-aware system, and is used to determine the contents of the time-synchronization event message and possibly separate general message that will be sent by this port.

The structure is:

```
MDSyncSend {
        domainNumber,
        followUpCorrectionField,
        sourcePortIdentity,
        logMessageInterval,
        preciseOriginTimestamp,
        upstreamTxTime,
        rateRatio,
        gmTimeBaseIndicator,
        lastGmPhaseChange,
        lastGmFreqChange
}
```

The members of the structure are defined in the following subclauses.

10.2.2.1.2 domainNumber (UInteger8)

The domainNumber of the gPTP domain in which this structure is sent.

10.2.2.1.3 followUpCorrectionField (ScaledNs)

The followUpCorrectionField contains the accumulated time since the preciseOriginTimestamp was captured by the grandmaster. This is equal to the elapsed time, relative to the grandmaster, between the time the grandmaster sent the received time-synchronization event message, truncated to the nearest nanosecond, and the time at which that event message was sent by the upstream time-aware system. The followUpCorrectionField is equal to the value of the followUpCorrectionField member of the most recently received PortSyncSync structure from the PortSync entity of this port (see 10.2.2.3.5).

10.2.2.1.4 sourcePortIdentity (PortIdentity)

The sourcePortIdentity is the portIdentity of this port (see 8.5.2).

10.2.2.1.5 logMessageInterval (Integer8)

The logMessageInterval is the value of currentLogSyncInterval for this port (see 10.7.2.3).

10.2.2.1.6 preciseOriginTimestamp (Timestamp)

The preciseOriginTimestamp is the sourceTime of the ClockMaster entity of the grandmaster, with any fractional nanoseconds truncated, when the received time-synchronization information was sent by the grandmaster. The preciseOriginTimestamp is the value of the preciseOriginTimestamp member of the most recently received PortSyncSync structure from the PortSync entity of this port (see 10.2.2.3.8).

10.2.2.1.7 upstreamTxTime (UScaledNs)

The upstreamTxTime is given by:

$$\text{upstreamTxTime} = \text{syncEventIngressTimestamp} - \frac{\text{neighborPropDelay}}{\text{neighborRateRatio}}$$

where the value of ~~the~~ ~~<~~syncEventIngressTimestamp~~>~~ correspond~~ing~~s to the receipt of the time-synchronization information at the slave port of this PTP instance, ~~minus~~neighborPropDelay is defined in 10.2.4.8), and ~~the mean propagation time on the link attached to this port divided by~~ neighborRateRatio ~~(see~~is defined in 10.2.4.7). The upstreamTxTime is the value of the upstreamTxTime member of the most recently received PortSyncSync structure from the PortSync entity of this port (see 10.2.2.3.9).

10.2.2.1.8 rateRatio (Double)

The rateRatio is the value of the rateRatio member of the most recently received PortSyncSync structure from the PortSync entity of this port (see 10.2.2.3.10). It is equal to the ratio of the frequency of the grandmaster to the frequency of the LocalClock entity of this time-aware system (see 10.2.7.1.4).

10.2.2.1.9 gmTimeBaseIndicator (UInteger16)

The gmTimeBaseIndicator is the timeBaseIndicator of the ClockSource entity of the current grandmaster. It is set equal to the gmTimeBaseIndicator of the received time-synchronization information. The gmTimeBaseIndicator is the value of the gmTimeBaseIndicator member of the most recently received PortSyncSync structure from the PortSync entity of this port (see 10.2.2.3.11).

10.2.2.1.10 lastGmPhaseChange (ScaledNs)

The lastGmPhaseChange is the time of the current grandmaster minus the time of the previous grandmaster, at the time that the current grandmaster became grandmaster, or the step change in the time of the current grandmaster at the time of the most recent gmTimeBaseIndicator change. It is set equal to the lastGmPhaseChange of the received time-synchronization information. The lastGmPhaseChange is the value of the lastGmPhaseChange member of the most recently received PortSyncSync structure from the PortSync entity of this port (see 10.2.2.3.12).

10.2.2.1.11 lastGmFreqChange (Double)

The lastGmFreqChange is the fractional frequency offset of the current grandmaster relative to the previous grandmaster, at the time that the current grandmaster became grandmaster, or relative to itself prior to the last change in gmTimeBaseIndicator. It is set equal to the lastGmFreqChange of the received time-synchronization information. The lastGmFreqChange is the value of the lastGmFreqChange member of the most recently received PortSyncSync structure from the PortSync entity of this port (see 10.2.2.3.13).

10.2.2.2 MDSyncReceive

10.2.2.2.1 General

This structure contains information that is sent by the MD entity of a port to the PortSync entity of that port. It provides the PortSync entity with master clock timing information and timestamp of receipt of a time-synchronization event message compensated for propagation time on the upstream link. The information is sent to the PortSync entity upon receipt of time-synchronization information by the MD entity of the port. The information is in turn provided by the PortSync entity to the SiteSync entity. The information is used by the PortSyncSyncReceive state machine of the PortSync entity to compute the rate ratio of the local oscillator relative to the master and is communicated to the SiteSync entity, and then by the SiteSync entity to the other PortSync entities for use in computing master clock timing information.

The structure is:

```
MDSyncReceive {
        domainNumber,
        followUpCorrectionField,
        sourcePortIdentity,
        logMessageInterval,
        preciseOriginTimestamp,
        upstreamTxTime,
        rateRatio,
        gmTimeBaseIndicator,
        lastGmPhaseChange,
        lastGmFreqChange
}
```

The members of the structure are defined in the following subclauses.

10.2.2.2.2 domainNumber (UInteger8)

The domainNumber is the domain number of the gPTP domain in which this structure is sent.

10.2.2.2.3 followUpCorrectionField (ScaledNs)

The followUpCorrectionField contains the elapsed time, relative to the grandmaster, between the time the grandmaster sent the received time-synchronization information, truncated to the nearest nanosecond, and the time at which this information was sent by the upstream time-aware system.

NOTE 1—The sum of followUpCorrectionField and preciseOriginTimestamp is the synchronized time that corresponds to the time the most recently received time-synchronization event message was sent by the upstream time-aware system.

NOTE 2—For a medium that uses separate event and general messages (for example, full-duplex, point-to-point media described in Clause 11), the event message corresponding to the most recently received network synchronization information is the event message that corresponds to the most recently received general message. For a medium that places synchronization information based on the event message timestamp in the next event message (for example, IEEE 802.11 media described in Clause 12), the event message corresponding to the most recently received network synchronization information is the previous event message; in this case, the time-synchronization information in the current event message refers to the previous event message.

10.2.2.2.4 sourcePortIdentity (PortIdentity)

The sourcePortIdentity is the value of the sourcePortIdentity of the time-synchronization event message received by this port. It is the portIdentity of the upstream MasterPort that sent the event message.

10.2.2.2.5 logMessageInterval (Integer8)

The logMessageInterval is the value of the logMessageInterval of the time-synchronization event message received by this port. It is the currentLogSyncInterval (see 10.7.2.3) of the upstream MasterPort that sent the event message.

10.2.2.2.6 preciseOriginTimestamp (Timestamp)

The preciseOriginTimestamp is the sourceTime of the ClockMaster entity of the grandmaster, with any fractional nanoseconds truncated, when the time-synchronization event message was sent by the grandmaster.

10.2.2.2.7 upstreamTxTime (UScaledNs)

The upstreamTxTime is given by:

$$\text{upstreamTxTime} = \text{syncEventIngressTimestamp} - \frac{\text{neighborPropDelay}}{\text{neighborRateRatio}}$$

where the value of syncEventIngressTimestamp corresponds to the receipt of the time-synchronization information at the slave port of this PTP instance (i.e., at this port), neighborPropDelay is defined in 10.2.4.8), and neighborRateRatio is defined in 10.2.4.7).~~The upstreamTxTime is the value of the <syncEventIngressTimestamp> corresponding to the receipt of the time-synchronization event message, minus the mean propagation time on the link attached to this port divided by neighborRateRatio (see 10.2.4.7).~~

10.2.2.2.8 rateRatio (Double)

The rateRatio is the value of rateRatio of the received ~~time synchronization~~time-synchronization information. It is equal to the ratio of the frequency of the grandmaster to the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this port, i.e., the time-aware system that sent the most recently received time-synchronization event message (see 10.2.7.1.4).

10.2.2.2.9 gmTimeBaseIndicator (UInteger16)

The gmTimeBaseIndicator is the timeBaseIndicator of the ClockSource entity of the current grandmaster. It is set equal to the gmTimeBaseIndicator of the received time-synchronization information.

10.2.2.2.10 lastGmPhaseChange (ScaledNs)

The lastGmPhaseChange is the time of the current grandmaster minus the time of the previous grandmaster, at the time that the current grandmaster became grandmaster, or the step change in the time of the current grandmaster at the time of the most recent gmTimeBaseIndicator change. It is set equal to the lastGmPhaseChange of the received time-synchronization information.

10.2.2.2.11 lastGmFreqChange (Double)

The lastGmFreqChange is the fractional frequency offset of the current grandmaster relative to the previous grandmaster, at the time that the current grandmaster became grandmaster, or relative to itself prior to the last change in gmTimeBaseIndicator. It is set equal to the lastGmFreqChange of the received time-synchronization information.

10.2.2.3 PortSyncSync

10.2.2.3.1 General

This structure is sent by the PortSync and ClockMaster entities to the SiteSync entity, and also from the SiteSync entity to the PortSync and ClockSlave entities.

When sent from the PortSync or ClockMaster entity, it provides the SiteSync entity with master clock timing information, timestamp of receipt of a time-synchronization event message compensated for propagation time on the upstream link, and the time at which sync receipt timeout occurs if a subsequent Sync message is not received by then. The information is used by the SiteSync entity to compute the rate ratio of the local oscillator relative to the master and is communicated to the other PortSync entities for use in computing master clock timing information.

When sent from the SiteSync entity to the PortSync or ClockMaster entity, the structure contains information needed to compute the synchronization information that will be included in respective fields of the time-synchronization event and general messages that will be sent, and also to compute the synchronized time that the ClockSlave entity will supply to the ClockTarget entity.

The structure is:

```
PortSyncSync {
        domainNumber,
        localPortNumber,
        syncReceiptTimeoutTime,
        followUpCorrectionField,
        sourcePortIdentity,
        logMessageInterval,
        preciseOriginTimestamp,
        upstreamTxTime,
        rateRatio,
        gmTimeBaseIndicator,
        lastGmPhaseChange,
        lastGmFreqChange
}
```

The parameters of the PortSyncSync structure are defined in the following subclauses for the case where the structure is sent from the PortSync or ClockMaster entity to the SiteSync entity. If the structure is sent from the SiteSync entity to the PortSync or ClockSlave entity, the member values are copied from the most recently received PortSyncSync structure where the port that received this structure has port ~~role~~state of SlavePort.

10.2.2.3.2 domainNumber (UInteger8)

The domainNumber is the domain number of the gPTP domain in which this structure is sent.

10.2.2.3.3 localPortNumber (UInteger16)

If the structure is sent by a PortSync entity, the localPortNumber is the port number of the port whose PortSync entity sent this structure. If the structure is sent by a ClockMaster entity, the localPortNumber is zero.

10.2.2.3.4 syncReceiptTimeoutTime (UScaledNs)

If the structure is sent by a PortSync entity, the syncReceiptTimeoutTime is the value of the local time (i.e., the free-running, local oscillator time) at which sync receipt timeout occurs if a subsequent time-synchronization event message is not received by that time. If the structure is sent by a ClockMaster entity, the syncReceiptTimeoutTime is FFFFFFFFFFFFFFFF$_{16}$ [see 10.2.8.2.1, item h)].

10.2.2.3.5 followUpCorrectionField (ScaledNs)

If the structure is sent by a PortSync entity, the followUpCorrectionField is the value of the followUpcorrectionField member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.2). If the structure is sent by a ClockMaster entity, the followUpCorrectionField is the sub-nanosecond portion of the ClockMaster time.

10.2.2.3.6 sourcePortIdentity (PortIdentity)

If the structure is sent by a PortSync entity, the sourcePortIdentity is the value of the sourcePortIdentity member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.4). If the structure is sent by a ClockMaster entity, the clockIdentity member of the sourcePortIdentity is the clockIdentity of this time-aware system, and the portNumber member of the sourcePortIdentity is 0.

10.2.2.3.7 logMessageInterval (Integer8)

If the structure is sent by a PortSync entity, the logMessageInterval is the value of the logMessageInterval member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.5). If the structure is sent by a ClockMaster entity, the logMessageInterval is the value of clockMasterLogSyncInterval (see 10.7.2.4).

10.2.2.3.8 preciseOriginTimestamp (Timestamp)

If the structure is sent by a PortSync entity, the preciseOriginTimestamp is the value of the preciseOriginTimestamp member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.6). If the structure is sent by a ClockMaster entity, the preciseOriginTimestamp is the ClockMaster time truncated to the next lower nanosecond.

10.2.2.3.9 upstreamTxTime (UScaledNs)

If the structure is sent by a PortSync entity, the upstreamTxTime is the value of the upstreamTxTime member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.7). If the structure is sent by a ClockMaster entity, the upstreamTxTime is the local oscillator time corresponding to the ClockMaster time.

10.2.2.3.10 rateRatio (Double)

If the structure is sent by a PortSync entity, the rateRatio is the value of the rateRatio member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.8). It is equal to the ratio of the frequency of the grandmaster to the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this port, i.e., the time-aware system that sent the most recently-received time-synchronization event message (see 10.2.7.1.4). If the structure is sent by a ClockMaster entity, the rateRatio is equal to gmRateRatio (see 10.2.3.14).

10.2.2.3.11 gmTimeBaseIndicator (UInteger16)

If the structure is sent by a PortSync entity, the gmTimeBaseIndicator is the value of the gmTimeBaseIndicator member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.9). If the structure is sent by a ClockMaster entity, the gmTimeBaseIndicator is equal to clockSourceTimeBaseIndicator (see 10.2.3.8).

10.2.2.3.12 lastGmPhaseChange (ScaledNs)

If the structure is sent by a PortSync entity, the lastGmPhaseChange is the value of the lastGmPhaseChange member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.9). If the structure is sent by a ClockMaster entity, the lastGmPhaseChange is equal to clockSourcePhaseOffset (see 10.2.3.7).

10.2.2.3.13 lastGmFreqChange (Double)

If the structure is sent by a PortSync entity, the lastGmFreqChange is the value of the lastGmFreqChange member of the MDSyncReceive structure whose receipt caused the sending of this structure (see 10.2.2.2.9). If the structure is sent by a ClockMaster entity, the lastGmFreqChange is equal to clockSourceFreqOffset (see 10.2.3.6).

10.2.3 Per-time-aware-system global variables

10.2.3.1 BEGIN: a Boolean controlled by the system initialization. If BEGIN is true, all state machines, including per-port state machines, continuously execute their initial state.

10.2.3.2 clockMasterSyncInterval: a variable containing the mean time interval between successive messages providing time-synchronization information by the ClockMaster entity to the SiteSync entity. This value is given by $1000000000 \times 2^{\text{clockMasterLogSyncInterval}}$, where clockMasterLogSyncInterval is the logarithm to base 2 of the mean time between the successive providing of time-synchronization information by the ClockMaster entity (see 10.7.2.4). The data type for clockMasterSyncInterval is UScaledNs.

10.2.3.3 clockSlaveTime: the synchronized time maintained, at the slave, at the granularity of the LocalClock entity [i.e., a new value is computed every localClockTickInterval (see 10.2.3.18) by the ClockSlave entity]. The data type for clockSlaveTime is ExtendedTimestamp.

10.2.3.4 syncReceiptTime: the synchronized time computed by the ClockSlave entity at the instant time-synchronization information, contained in a PortSyncSync structure, is received. The data type for syncReceiptTime is ExtendedTimestamp.

10.2.3.5 syncReceiptLocalTime: the value of currentTime (i.e., the time relative to the LocalClock entity) corresponding to syncReceiptTime. The data type for syncReceiptLocalTime is UScaledNs.

10.2.3.6 clockSourceFreqOffset: the fractional frequency offset of the ClockSource entity frequency relative to the current grandmaster frequency. The data type for clockSourceFreqOffset is Double.

10.2.3.7 clockSourcePhaseOffset: the time provided by the ClockSource entity, minus the synchronized time. The data type for clockSourcePhaseOffset is ScaledNs.

10.2.3.8 clockSourceTimeBaseIndicator: a global variable that is set equal to the timeBaseIndicator parameter of the ClockSourceTime.invoke application interface function (see 9.2.2.3). That parameter is set by the ClockSource entity and is changed by that entity whenever the time base changes. The data type for clockSourceTimeBaseIndicator is UInteger16.

10.2.3.9 clockSourceTimeBaseIndicatorOld: a global variable that is set equal to the previous value of clockSourceTimeBaseIndicator. The data type for clockSourceTimeBaseIndicatorOld is UInteger16.

10.2.3.10 clockSourceLastGmPhaseChange: a global variable that is set equal to the lastGmPhaseChange parameter of the ClockSourceTime.invoke application interface function (see 9.2.2.4). That parameter is set by the ClockSource entity and is changed by that entity whenever the time base changes. The data type for clockSourceLastGmPhaseChange is ScaledNs.

10.2.3.11 clockSourceLastGmFreqChange: a global variable that is set equal to the lastGmFreqChange parameter of the ClockSourceTime.invoke application interface function (see 9.2.2.5). That parameter is set by the ClockSource entity and is changed by that entity whenever the time base changes. The data type for clockSourceLastGmFreqChange is Double.

10.2.3.12 currentTime: the current value of time relative to the LocalClock entity clock. The data type for currentTime is UScaledNs.

10.2.3.13 gmPresent: a Boolean that indicates whether a grandmaster-capable time-aware system is present in the domain. If TRUE, a grandmaster-capable time-aware system is present; if FALSE, a grandmaster-capable time-aware system is not present.

10.2.3.14 gmRateRatio: the measured ratio of the frequency of the ClockSource entity to the frequency of the LocalClock entity. The data type for gmRateRatio is Double.

10.2.3.15 gmTimeBaseIndicator: the most recent value of gmTimeBaseIndicator provided to the ClockSlaveSync state machine via a PortSyncSync structure. The data type for gmTimeBaseIndicator is UInteger16.

10.2.3.16 lastGmPhaseChange: the most recent value of lastGmPhaseChange provided to the ClockSlaveSync state machine via a PortSyncSync structure. The data type for lastGmPhaseChange is ScaledNs.

10.2.3.17 lastGmFreqChange: the most recent value of lastGmFreqChange provided to the ClockSlaveSync state machine via a PortSyncSync structure. The data type for lastGmFreqChange is Double.

10.2.3.18 localClockTickInterval: the time interval between two successive significant instants (i.e., “ticks”) of the LocalClock entity. The data type for localClockTickInterval is TimeInterval.

10.2.3.19 localTime: the value of currentTime when the most recent ClockSourceTime.invoke function (see 9.2) was received from the ClockSource entity, or when the LocalClock entity most recently updated its time. The data type for localTime is UScaledNs.

10.2.3.20 selected~~Role~~State: an Enumeration2 array of length numberPorts+1 (see 8.6.2.8). selected~~Role~~State[j] is set equal to the port ~~role~~state (see Table 10-1) of port whose portNumber is j.

10.2.3.21 masterTime: the time maintained by the ClockMaster entity, based on information received from the ClockSource and LocalClock entities. The data type for masterTime is ExtendedTimestamp.

10.2.3.22 thisClock: the clockIdentity of the current time-aware system. The data type for thisClock is ClockIdentity.

10.2.3.23 parentLogSyncInterval: the most recent logMessageInterval value received on the slave port. If this time-aware system is the grandmaster, then this is the clockMasterLogSyncInterval (see 10.7.2.4. Thedata type for parentLogSyncInterval is Integer8.

10.2.3.24 instanceEnable: a per-domain Boolean used to enable gPTP on all ports that are enabled for that domain (i.e., ports for which portOper and ptpPortEnabled are both TRUE). Setting instanceEnable to FALSE causes all per-domain state machines to go to the initial state.

NOTE—instanceEnable has no effect on the operation of the MDPdelayReq (see 11.2.15) and MDPdelayResp (see 11.2.18) state machines, because those state machines are not per domain (i.e., there is a single instance of each of those state machines, per link, for all domains).

10.2.4 Per-port global variables

10.2.4.1 asCapable: a Boolean that is TRUE if and only if it is determined that this time-aware system and the time-aware system at the other end of the link attached to this port can interoperate with each other via the IEEE 802.1AS protocol. This means that

a) this time-aware system is capable of executing the IEEE 802.1AS protocol,
b) the time-aware system at the other end of the link is capable of executing the IEEE 802.1AS protocol, and
c) there are no non-IEEE-802.1AS systems in between this time-aware system and the time-aware system at the other end of the link that introduce sufficient impairments that the end-to-end time-synchronization performance of B.3 cannot be met.

The determination of asCapable is different for each medium, and is described in the respective media-dependent clauses.

There is one instance of this variable per port, per domain.

NOTE—The per-port global variable asCapableAcrossDomains (see 11.2.12.12) is common across and accessible by all the domains. It is computed by the MDPdelayReq state machine (see 11.2.15). In the case of full-duplex, point-to-point links (see Clause 11), asCapableAcrossDomains is used when setting the instance of asCapable for each domain (for the link in question).

10.2.4.2 asymmetryMeasurementMode: a Boolean that contains the value of the managed object asymmetryMeasurementMode (see 14.8.35). For full-duplex IEEE 802.3 media, the value is TRUE if an asymmetry measurement is being performed for the link attached to this port, and FALSE otherwise. For all other media, the value is FALSE. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

10.2.4.3 syncReceiptTimeoutTimeInterval: the time interval after which sync receipt timeout occurs if ~~time synchronization~~time-synchronization information has not been received during the interval. The value of syncReceiptTimeoutTimeInterval is equal to syncReceiptTimeout (see 10.7.3.1) multiplied by the syncInterval (see 10.2.4.6) for the port at the other end of the link to which this port is attached. The value of syncInterval for the port at the other end of the link is computed from logMessageInterval of the received Sync message (see 10.6.2.2.14). The data type for syncReceiptTimeoutTimeInterval is UScaledNs.

10.2.4.4 currentLogSyncInterval: the current value of the logarithm to base 2 of the mean time interval, in seconds, between the sending of successive time-synchronization event messages (see 10.7.2.3). This value is set in the ~~LinkDelay~~SyncIntervalSetting state machine (see 11.2.19). The data type for currentLogSyncInterval is Integer8.

10.2.4.5 initialLogSyncInterval: the initial value of the logarithm to base 2 of the mean time interval, in seconds, between the sending of successive time-synchronization event messages (see 10.7.2.3). The data type for initialLogSyncInterval is Integer8.

10.2.4.6 syncInterval: a variable containing the mean time-synchronization event message transmission interval for the port. This value is set in the ~~LinkDelay~~SyncIntervalSetting state machine (see 11.2.19). The data type for syncInterval is UScaledNs.

10.2.4.7 neighborRateRatio: the measured ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this port, to the frequency of the LocalClock entity of this time-aware system. The data type for neighborRateRatio is Double. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

10.2.4.8 neighborPropDelay: the measured propagation delay on the link attached to this port, relative to the LocalClock entity of the time-aware system at the other end of the link (i.e., expressed in the time base of the time-aware system at the other end of the link). The data type for neighborPropDelay is UScaledNs. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

10.2.4.9 delayAsymmetry: the asymmetry in the propagation delay on the link attached to this port relative to the grandmaster time base, as defined in 8.3. If propagation delay asymmetry is not modeled, then delayAsymmetry is zero. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

10.2.4.10 computeNeighborRateRatio: a Boolean, set by the LinkDelay~~Sync~~IntervalSetting state machine (see 11.2.19), that indicates whether neighborRateRatio is to be computed by this port. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

10.2.4.11 computeNeighborPropDelay: a Boolean, set by the ~~LinkDelaySyncInterval~~LinkDelayIntervalSetting state machine (see 11.2.19), that indicates whether neighborPropDelay is to be computed by this port. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

10.2.4.12 portOper(the term *port* in this definition is a physical port): a Boolean that is TRUE if and only if the port is up and able to send and receive messages. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

NOTE 1—portOper is an indicator, and not a control, and reflects the operational status of the underlying medium. It is not administratively set by gPTP.

NOTE 2—The variable portOper corresponds to the variable portEnabled in the previous edition of this standard. The change is reflected in many state machines.

NOTE 3—portOper is the same as MAC_Operational (see IEEE Std 802.1AC).

~~**portEnabled [the term *port* in the following items a) through c) is a physical port]:** a Boolean that is set if the time-aware system's MAC Relay Entity and Spanning Tree Protocol Entity can use the MAC Service provided by the Port's MAC entity to transmit and receive frames to and from the attached LAN, i.e., portEnabled is TRUE if and only if~~ ~~the port is up and able to send and receive gPTP messages:~~

~~a) MAC_Operational (see 6.4.2 of IEEE Std 802.1D-2004) is TRUE; and~~
~~b) Administrative Bridge Port State (see 14.8.2.2 of IEEE Sd 802.1D-2004) for the Port is Enabled; and~~
~~c) AuthControlledPortStatus is Authorized (if the port is a network access port; see IEEE Std 802.1X™-2010 [B5]).~~

~~NOTE—portEnabled is an indicator, and not a control. It is not administratively set.~~

10.2.4.13 ~~pttPortEnabled~~ptpPortEnabled: a Boolean that is administratively set if~~ the~~ time-synchronization ~~and best master selection functions of the~~is to be supported on this port~~ are enabled~~.

NOTE 1—It is expected that the value of ~~pttPortEnabled~~ptpPortEnabled will be set via the management interface (see 14.8.4). ~~By having both portEnabled and pttPortEnabled variables, a~~A port can be enabled for data transport but not for synchronization transport.

NOTE 2—The variable ptpPortEnabled was named pttPortEnabled in the previous edition of this standard. This is only a name change; the definition and function of this variable is the same as in the previous edition of this standard. The name change is reflected in many state machines.

10.2.4.14 thisPort: the portNumber of the current port. The data type for thisPort is UInteger16.

10.2.4.15 **syncLocked:** a Boolean, set by the PortSyncSyncSend state machine (see 10.2.11.3), that indicates that this port, when operating as a Master port, shall transmit a Sync as soon as possible after the Slave port received a Sync (ignoring syncInterval). If FALSE, the port shall use the timing set by syncInterval.

10.2.4.16 **neighborGptpCapable:** a Boolean, set by the gPtpCapableReceive state machine (see 10.4.2), that indicates that the neighbor of this port (i.e, the port at the other end of the link attached to this port) is capable of invoking the gPTP protocol.

10.2.4.17 **syncSlowdown:** a Boolean that is set to TRUE if the SyncIntervalSetting state machine (see Figure 11-10) receives a TLV that requests a larger Sync message transmission interval (see 11.5.2.3), and FALSE otherwise. When syncSlowdown is set to TRUE, the PortSyncSyncSend state machine (see Figure 10-8) continues to send Sync messages (and Follow_Up messages if the port is two-step) at the old (i.e., faster) rate until the number of Sync messages equal to syncReceiptTimeout (see 10.7.3.1) have been sent, but with the logMessageInterval field of the PTP common header set equal to the new sync interval (i.e., corresponding to the slower rate). After syncReceiptTimeout Sync messages have been sent, subsequent Sync messages are sent at the new (i.e., slower) rate, and with the logMessageInterval field of the PTP common header set to the new sync interval.

NOTE—If a receiver of Sync messages requests a slower rate, the receiver will continue to use the upstream syncInterval value, which it obtains from the logMessageInterval field of received Sync messages, until it receives a Sync message where that value has changed. If, immediately after requesting a slower Sync message rate, up to syncReceiptTimeout consecutive Sync messages sent to the receiver are lost, sync receipt timout could occur if the sender had changed to the slower rate immediately. Delaying the slowing down of the sending rate of Sync messages for syncReceiptTimeout messages prevents this from happening.

10.2.4.18 **oldAnnounceInterval:** the saved value of the previous announce interval, when a new announce is requested via a Signaling message that contains a message interval request TLV. The data type for oldAnnounceInterval is UScaledNs.

10.2.5 Functions used by multiple state machines

10.2.5.1 random(): returns a uniformly-distributed pseudo-random number whose data type is UInteger16 (i.e., the function returns a uniformly distributed, pseudo-random integer in the range [0, $2^{16} – 1$]).

10.2.6 SiteSyncSync state machine

10.2.6.1 State machine variables

10.2.6.1.1 rcvdPSSync: a Boolean variable that notifies the current state machine when a PortSyncSync structure (see 10.2.2.3) is received from the PortSyncSyncReceive state machine of a PortSync entity or from the ClockMasterSyncSend state machine of the ClockMaster entity. This variable is reset by this state machine.

10.2.6.1.2 rcvdPSSyncPtr: a pointer to the received PortSyncSync structure indicated by rcvdPSSync.

10.2.6.1.3 txPSSyncPtr: a pointer to the PortSyncSync structure transmitted by the state machine.

10.2.6.2 State machine functions

10.2.6.2.1 setPSSyncSend (rcvdPSSyncPtr): creates a PortSyncSync structure to be transmitted, and returns a pointer to this structure. The members are copied from the received PortSyncSync structure pointed to by rcvdPSSyncPtr.

10.2.6.2.2 txPSSync (txPSSyncPtr): transmits a copy of the PortSyncSync structure pointed to by txPSSyncPtr to the PortSyncSyncSend state machine of each PortSync entity and the ClockSlaveSync state machine of the ClockSlave entity of this time-aware system.

10.2.6.3 State diagram

The SiteSyncSync state machine shall implement the function specified by the state diagram in Figure 10-3, the local variables specified in 10.2.6.1, the functions specified in 10.2.6.2, the structure specified in 10.2.2.3, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. The state machine receives time-synchronization information, accumulated rateRatio, and syncReceiptTimeoutTime from the PortSync entity (PortSyncSyncReceive state machine) of the current slave port or from the ClockMaster entity (ClockMasterSyncSend state machine). If the information was sent by a PortSync entity, the state machine also receives the portIdentity of the port on the upstream time-aware system that sent the information to this time-aware system (if the information was sent by the ClockMaster entity, this portIdentity is zero). The state machine sends a PortSyncSync structure to the PortSync entities of all the ports and to the ClockSlave entity.

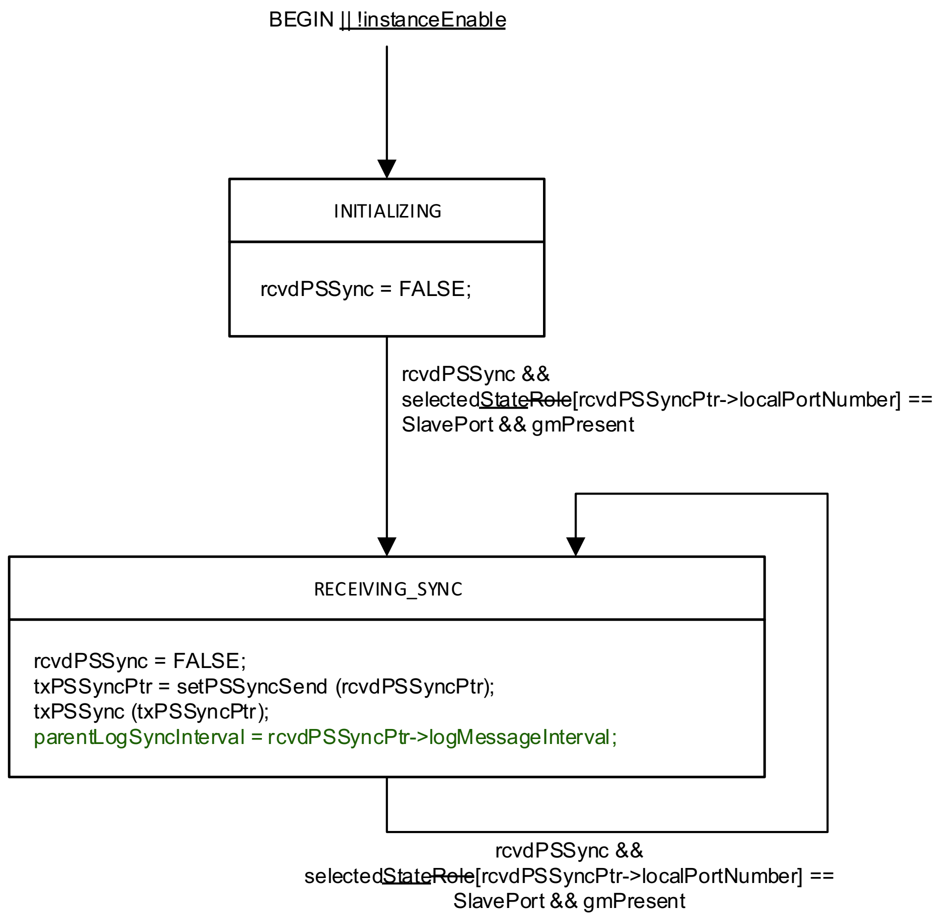

BEGIN || !instanceEnable
INITIALIZING
rcvdPSSync = FALSE;
rcvdPSSync && selected~~Role~~State[rcvdPSSyncPtr->localPortNumber] == SlavePort && gmPresent
RECEIVING_SYNC
rcvdPSSync = FALSE;
txPSSyncPtr = setPSSyncSend (rcvdPSSyncPtr);
txPSSync (txPSSyncPtr);
parentLogSyncInterval = rcvdPSSyncPtr->logMessageInterval;
rcvdPSSync && selected~~Role~~State[rcvdPSSyncPtr->localPortNumber] == SlavePort && gmPresent

Figure 10-3—SiteSyncSync state machine

10.2.7 PortSyncSyncReceive state machine

10.2.7.1 State machine variables

10.2.7.1.1 rcvdMDSync: a Boolean variable that notifies the current state machine when an MDSyncReceive structure is received from the MDSyncReceiveSM state machine of an MD entity of the same port (see 10.2.2.1). This variable is reset by this state machine.

10.2.7.1.2 rcvdMDSyncPtr: a pointer to the received MDSyncReceive structure indicated by rcvdMDSync.

10.2.7.1.3 txPSSyncPtr: a pointer to the PortSyncSync structure transmitted by the state machine.

10.2.7.1.4 rateRatio: a Double variable that holds the ratio of the frequency of the grandmaster to the frequency of the LocalClock entity. This frequency ratio is computed by (a) measuring the ratio of the grandmaster frequency to the LocalClock frequency at the grandmaster time-aware system and initializing rateRatio to this value in the ClockMasterSend state machine of the grandmaster ~~node~~time-aware system, and (b) accumulating, in the PortSyncSyncReceive state machine of each time-aware system, the frequency offset of the LocalClock entity of the time-aware system at the remote end of the link attached to that port to the frequency of the LocalClock entity of this time-aware system.

10.2.7.2 State machine functions

10.2.7.2.1 setPSSyncPSSR (rcvdMDSyncPtr, syncReceiptTimeoutTimeInterval, rateRatio): creates a PortSyncSync structure to be transmitted, and returns a pointer to this structure. The members are set as follows:

a) localPortNumber is set equal to thisPort,
b) domainNumber, followUpCorrectionField, sourcePortIdentity, logMessageInterval, and preciseOriginTimestamp are copied from the received MDSyncReceive structure,
c) upstreamTxTime is set equal to the upstreamTxTime member of the MDSyncReceive structure pointed to by rcvdMDSyncPtr,
d) syncReceiptTimeoutTime is set equal to currentTime plus syncReceiptTimeoutTimeInterval (see 10.2.4.3), and
e) the function argument rateRatio is set equal to the local variable rateRatio (computed just prior to invoking setPSSyncReceive (see Figure 10-4). The rateRatio member of the PortSyncSync structure is then set equal to the function argument rateRatio.

10.2.7.2.2 txPSSyncPSSR (txPSSyncPtr): transmits a copy of the PortSyncSync structure pointed to by txPSSyncPtr to the SiteSyncSync state machine of this time-aware system.

10.2.7.3 State diagram

The PortSyncSyncReceive state machine shall implement the function specified by the state diagram in Figure 10-4, the local variables specified in 10.2.7.1, the functions specified in 10.2.7.2, the structures specified in 10.2.2.1 and 10.2.2.3, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. The state machine receives time-synchronization information, accumulated rateRatio, and syncReceiptTimeoutTime from the MD entity (MDSyncReceiveSM state machine) of the same port. The state machine adds, to rateRatio, the fractional frequency offset of the LocalClock entity relative to the LocalClock entity of the upstream time-aware system at the remote end of the link attached to this port. The state machine computes syncReceiptTimeoutTime. The state machine sends this information to the SiteSync entity (SiteSyncSync state machine).

<<Editor's note: In Figure 10-4, the condition !asymmetryMeasurementMode has been added to the transition from the state RECEIVED_SYNC back to itself. In addition, portEnabled is changed to portOper and pttPortEnabled is changed to ptpPortEnabled.>>

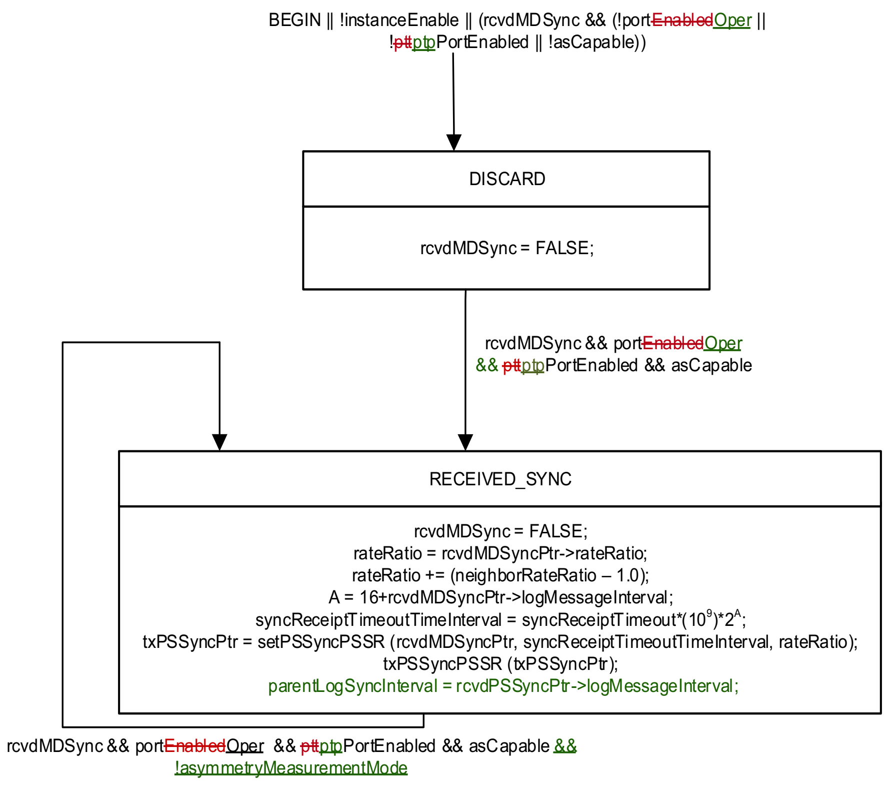

BEGIN || !instanceEnable || (rcvdMDSync && (!port~~Enabled~~Oper ||
!~~ptt~~ptpPortEnabled || !asCapable))
DISCARD
rcvdMDSync = FALSE;
rcvdMDSync && port~~Enabled~~Oper
&& ~~ptt~~ptpPortEnabled && asCapable
RECEIVED_SYNC
rcvdMDSync = FALSE;
rateRatio = rcvdMDSyncPtr->rateRatio;
rateRatio += (neighborRateRatio – 1.0);
A = 16+rcvdMDSyncPtr->logMessageInterval;
syncReceiptTimeoutTimeInterval = syncReceiptTimeout*(10^9)*2^A;
txPSSyncPtr = setPSSyncPSSR (rcvdMDSyncPtr, syncReceiptTimeoutTimeInterval, rateRatio);
txPSSyncPSSR (txPSSyncPtr);
parentLogSyncInterval = rcvdPSSyncPtr->logMessageInterval;
rcvdMDSync && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && asCapable &&
!asymmetryMeasurementMode

Figure 10-4—PortSyncSyncReceive state machine

10.2.8 ClockMasterSyncSend state machine

10.2.8.1 State machine variables

10.2.8.1.1 syncSendTime: the time in seconds, relative to the LocalClock entity, when synchronization information will next be sent to the SiteSync entity, via a PortSyncSync structure. The data type for syncSendTime is UScaledNs.

10.2.8.1.2 txPSSyncPtr: a pointer to the PortSyncSync structure transmitted by the state machine.

10.2.8.2 State machine functions

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

a) localPortNumber is set to 0
b) preciseOriginTimestamp is set equal to the masterTime, with any fractional nanoseconds truncated
c) followUpCorrectionField is set equal to the sum of
    1) the fractional nanoseconds portion of masterTime.fractionalNanoseconds,

   2) the quantity gmRateRatio × (currentTime – localTime)

d) the clockIdentity member of sourcePortIdentity is set equal to the clockIdentity of this time-aware system
e) the portNumber member of the sourcePortIdentity is set to 0

NOTE—This quantity and localPortNumber are redundant; both are retained so that the SiteSync entity can process PortSyncSync structures received from a PortSync entity or the ClockMaster entity in the same manner.

f) logMessageInterval is set to clockMasterLogSyncInterval
g) upstreamTxTime is set equal to localTime
h) syncReceiptTimeoutTime is set equal to $\text{FFFFFFFFFFFFFFFF}_{16}$, which indicates that there is no sync receipt timeout.

NOTE—A ClockMaster entity does not receive Sync messages, and there is no notion of sync receipt timeout. ~~syncReceiptTimeoutTime is set to the largest possible value, approximately 5.84 × 10^6 years.~~

i) rateRatio is set equal to gmRateRatio
j) gmTimeBaseIndicator is set equal to clockSourceTimeBaseIndicator
k) lastGmPhaseChange is set equal to clockSourcePhaseOffset~~, and~~
l) lastGmFreqChange is set equal to clockSourceFreqOffset, and
m) domainNumber is set equal to the domain number of this gPTP domain.

10.2.8.2.2 txPSSyncCMSS (txPSSyncPtr): transmits a copy of the PortSyncSync structure pointed to by txPSSyncPtr to the SiteSync state machine.

10.2.8.3 State diagram

The ClockMasterSyncSend state machine shall implement the function specified by the state diagram in Figure 10-5, the local variables specified in 10.2.8.1, the functions specified in 10.2.8.2, the structure specified in 10.2.2.3, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. The state machine receives masterTime and clockSourceTimeBaseIndicator from the ClockMasterSyncReceive state machine, and phase and frequency offset between masterTime and syncReceiptTime from the ClockMasterSyncOffset state machine. It provides masterTime (i.e., synchronized time) and the phase and frequency offset to the SiteSync entity via a PortSyncSync structure.

The ClockMasterSyncSend state machine is optional for time-aware systems that are not grandmaster-capable (see 8.6.2.1, 10.1.2, and 10.2.1). This state machine may be present in a time-aware system that is not grandmaster-capable; however, any information supplied by it to the SiteSyncSync state machine is not used by the SiteSyncSync state machine if the time-aware system is not grandmaster-capable.

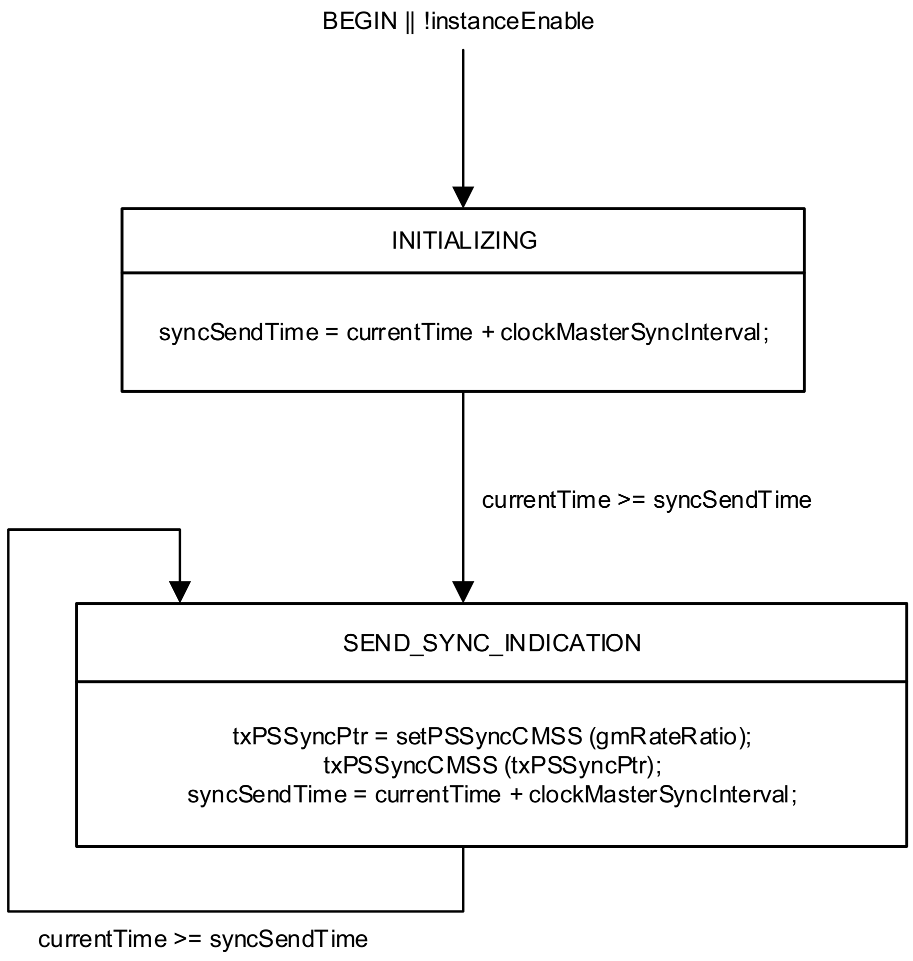

BEGIN || !instanceEnable
INITIALIZING
syncSendTime = currentTime + clockMasterSyncInterval;
currentTime >= syncSendTime
SEND_SYNC_INDICATION
txPSSyncPtr = setPSSyncCMSS (gmRateRatio);
txPSSyncCMSS (txPSSyncPtr);
syncSendTime = currentTime + clockMasterSyncInterval;
currentTime >= syncSendTime

Figure 10-5—ClockMasterSyncSend state machine

10.2.9 ClockMasterSyncOffset state machine

10.2.9.1 State machine variables

10.2.9.1.1 rcvdSyncReceiptTime: a Boolean variable that notifies the current state machine when syncReceiptTime is received from the ClockSlave entity. This variable is reset by this state machine.

10.2.9.2 State machine functions

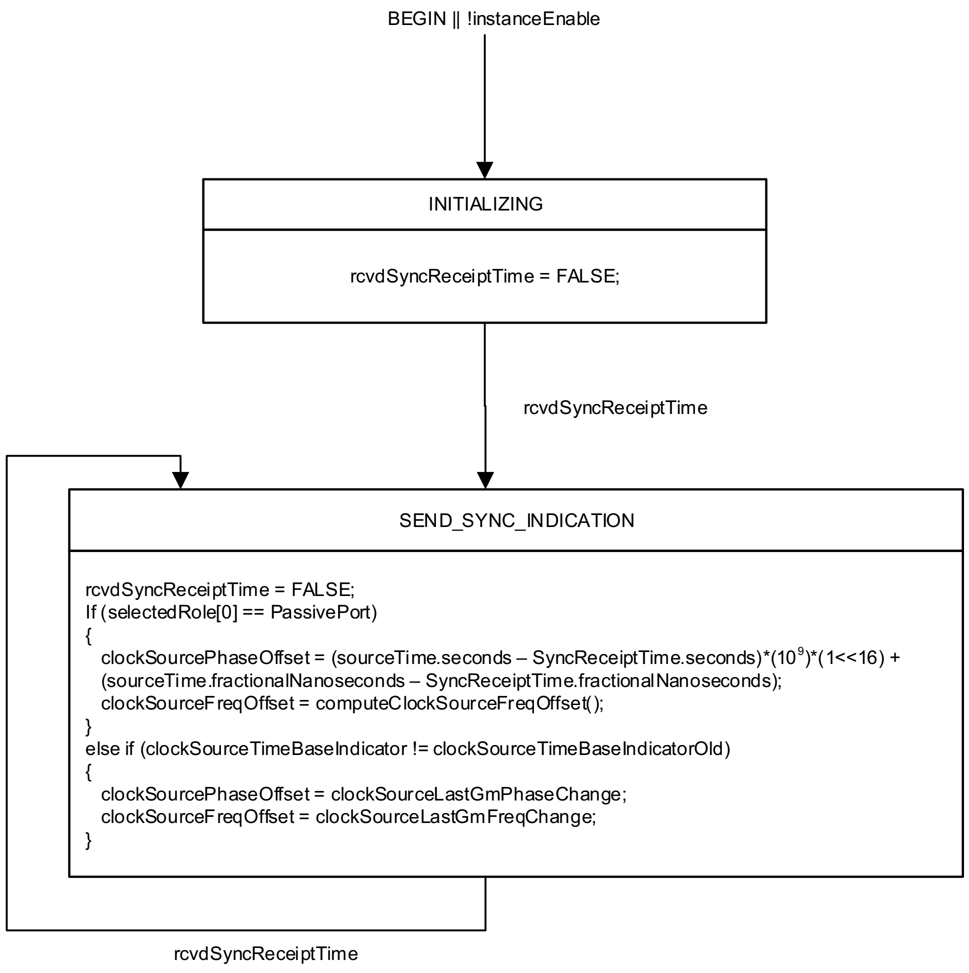
10.2.9.2.1 computeClockSourceFreqOffset(): computes clockSourceFreqOffset (see 10.2.3.6), using successive values of masterTime computed by the ClockMasterSyncReceive state machine (see 10.2.10) and successive values of syncReceiptTime computed by the ClockSlaveSync state machine (see 10.2.12). Any scheme that uses this information to compute clockSourceFreqOffset is acceptable as long as the performance requirements specified in B.2.4 are met.

NOTE—As one example, clockSourceFreqOffset can be estimated as the ratio of the duration of a time interval measured by the ClockSource entity to the duration of the same time interval computed from networkTime values, minus 1.

10.2.9.3 State diagram

The ClockMasterSyncOffset state machine shall implement the function specified by the state diagram in Figure 10-6, the local variables specified in 10.2.9.1, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. 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 time-aware system is not currently the grandmaster, i.e., if selected~~Role~~State[0] is equal to PassivePort.

The ClockMasterSyncOffset state machine is optional for time-aware systems that are not grandmaster-capable (see 8.6.2.1, 10.1.2, and 10.2.1). This state machine may be present in a time-aware system 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 time-aware system is not grandmaster-capable.



```
rcvdSyncReceiptTime = FALSE;
If (selectedRole[0] == PassivePort)
{
  clockSourcePhaseOffset = (sourceTime.seconds – SyncReceiptTime.seconds)*(10^9)*(1<<16) +
  (sourceTime.fractionalNanoseconds – SyncReceiptTime.fractionalNanoseconds);
  clockSourceFreqOffset = computeClockSourceFreqOffset( );
}
else if (clockSourceTimeBaseIndicator != clockSourceTimeBaseIndicatorOld)
{
  clockSourcePhaseOffset = clockSourceLastGmPhaseChange;
  clockSourceFreqOffset = clockSourceLastGmFreqChange;
}
```

Figure 10-6—ClockMasterSyncOffset state machine

10.2.10 ClockMasterSyncReceive state machine

10.2.10.1 State machine variables

10.2.10.1.1 rcvdClockSourceReq: a Boolean variable that notifies the current state machine when ClockSourceTime.invoke function is received from the Clock source entity. This variable is reset by this state machine.

10.2.10.1.2 rcvdClockSourceReqPtr: a pointer to the received ClockSourceTime.invoke function parameters.

10.2.10.1.3 rcvdLocalClockTick: a Boolean variable that notifies the current state machine when the LocalClock entity updates its time. This variable is reset by this state machine.

10.2.10.2 State machine functions

10.2.10.2.1 computeGmRateRatio(): computes gmRateRatio(see 10.2.3.14), using values of sourceTime conveyed by successive ClockSourceTime.invoke functions (see 9.2.2.1), and corresponding values of localTime (see 10.2.3.19). Any scheme that uses this information, along with any other information conveyed by the successive ClockSourceTime.invoke functions and corresponding values of localTime, to compute gmRateRatio is acceptable as long as the performance requirements specified in B.2.4 are met.

NOTE—As one example, gmRateRatio can be estimated as the ratio of the elapsed time of the ClockSource entity that supplies time to this time-aware system, to the elapsed time of the LocalClock entity of this time-aware system. This ratio can be computed for the time interval between a received ClockSourceTime.invoke function and a second received ClockSourceTime.invoke function some number of ClockSourceTime.invoke functions later, i.e.,

$$\frac{\text{ClockSource.invoke.sourceTime}_N - \text{ClockSource.invoke.sourceTime}_0}{\text{localTime}_N - \text{localTime}_0}$$

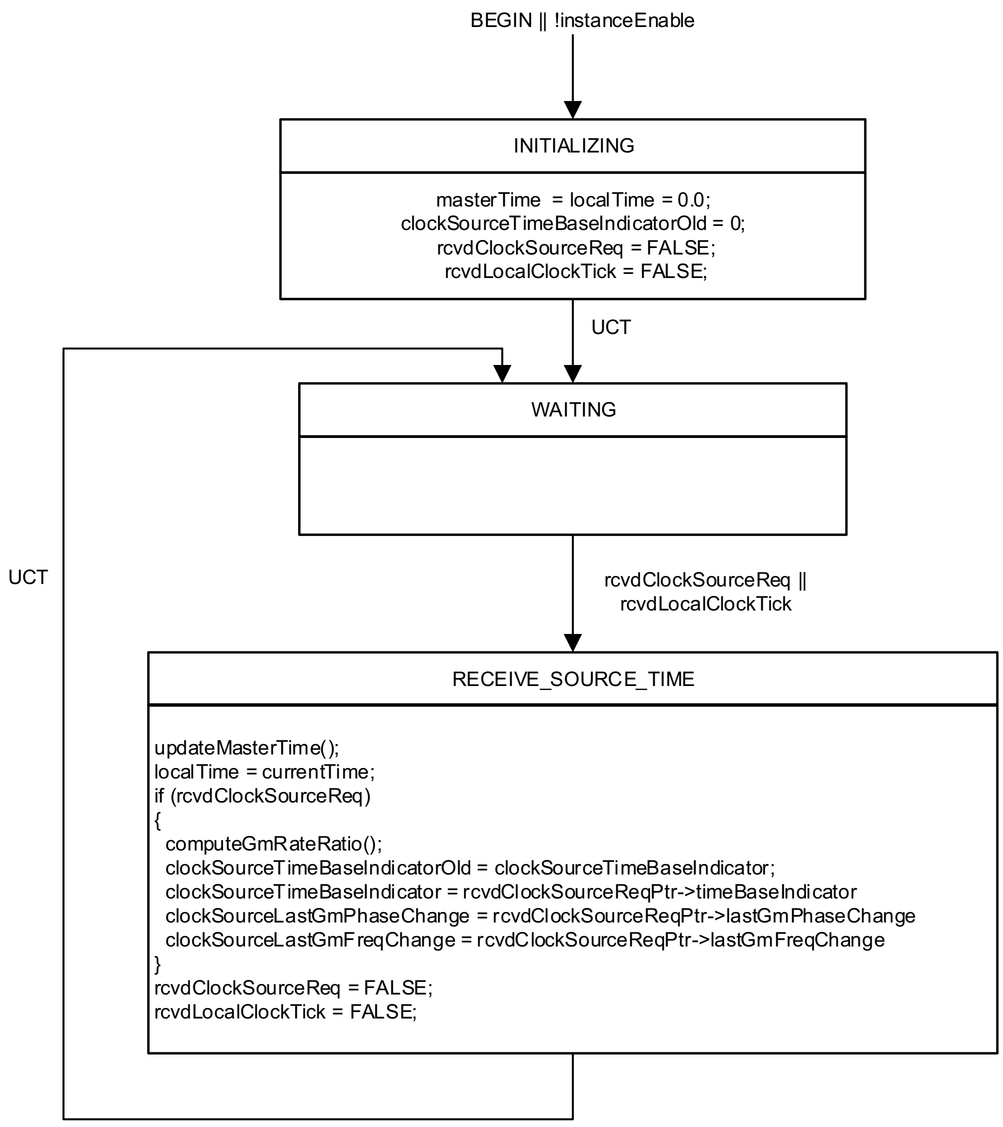
where the successive received ClockSourceTime.invoke functions are indexed from 0 to *N*, with the first such function indexed as 0, and localTime_j is the value of localTime when the ClockSourceTime.invoke function whose index is *j* is received.

10.2.10.2.2 updateMasterTime(): updates the global variable masterTime (see 10.2.3.21), based on information received from the ClockSource and LocalClock entities. It is the responsibility of the application to filter master times appropriately. As one example, masterTime can be set equal to the sourceTime member of the ClockSourceTime.invoke function when this function is received, and can be incremented by localClockTickInterval (see 10.2.3.18) multiplied by gmRateRatio (see 10.2.3.14) when rcvdLocalClockTick is TRUE.

10.2.10.3 State diagram

The ClockMasterSyncReceive state machine shall implement the function specified by the state diagram in Figure 10-7, the local variables specified in 10.2.10.1, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. The state machine updates the global variable masterTime with information received from the ClockSource entity via the ClockSourceTime.invoke function and information received from the LocalClock entity. It also computes gmRateRatio, i.e., the ratio of the ClockSource entity frequency and the LocalClock entity frequency.

The ClockMasterSyncReceive state machine is optional for time-aware systems that are not grandmaster-capable (see 8.6.2.1, 10.1.2, and 10.2.1). This state machine may be present in a time-aware system 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 time-aware system is not grandmaster-capable.


BEGIN || !instanceEnable
INITIALIZING
masterTime = localTime = 0.0;
clockSourceTimeBaseIndicatorOld = 0;
rcvdClockSourceReq = FALSE;
rcvdLocalClockTick = FALSE;
UCT
WAITING
UCT
rcvdClockSourceReq ||
rcvdLocalClockTick
RECEIVE_SOURCE_TIME

```
updateMasterTime();
localTime = currentTime;
if (rcvdClockSourceReq)
{
  computeGmRateRatio();
  clockSourceTimeBaseIndicatorOld = clockSourceTimeBaseIndicator;
  clockSourceTimeBaseIndicator = rcvdClockSourceReqPtr->timeBaseIndicator
  clockSourceLastGmPhaseChange = rcvdClockSourceReqPtr->lastGmPhaseChange
  clockSourceLastGmFreqChange = rcvdClockSourceReqPtr->lastGmFreqChange
}
rcvdClockSourceReq = FALSE;
rcvdLocalClockTick = FALSE;
```

Figure 10-7—ClockMasterSyncReceive state machine

10.2.11 PortSyncSyncSend state machine

10.2.11.1 State machine variables

10.2.11.1.1 rcvdPSSync: a Boolean variable that notifies the current state machine when a PortSyncSync structure is received from the SiteSyncSync state machine of the SiteSync entity of the time-aware system (see 10.2.2.3). This variable is reset by this state machine.

10.2.11.1.2 rcvdPSSyncPtr: a pointer to the received PortSyncSync structure indicated by rcvdPSSync.

10.2.11.1.3 lastPreciseOriginTimestamp: the preciseOriginTimestamp member of the most recently received PortSyncSync structure. The data type for lastPreciseOriginTimestamp is Timestamp.

10.2.11.1.4 lastFollowUpCorrectionField: the followUpCorrectionField member of the most recently received PortSyncSync member. The data type for lastFollowUpCorrectionField is ScaledNs.

10.2.11.1.5 lastRateRatio: the rateRatio member of the most recently received PortSyncSync structure. The data type for lastRateRatio is Double.

10.2.11.1.6 lastUpstreamTxTime: the upstreamTxTime of the most recently received PortSyncSync member. The data type for lastUpstreamTxTime is UScaledNs.

10.2.11.1.7 lastSyncSentTime: the value of currentTime (i.e., the time relative to the LocalClock entity) when the most recent MDSyncSend structure was sent. The data type for lastSyncSentTime is UScaledNs.

NOTE—lastSyncSentTime is the time the abstract MDSyncSend structure was sent, NOT the time the corresponding Sync message (or equivalent) was sent on a physical link.

10.2.11.1.8 lastRcvdPortNum: the portNumber of the port on which time-synchronization information was most recently received. The data type for lastReceivedPortNum is UInteger16.

10.2.11.1.9 lastGmTimeBaseIndicator: the gmTimeBaseIndicator of the most recently received PortSyncSync member. The data type for lastGmTimeBaseIndicator is UInteger16.

10.2.11.1.10 lastGmPhaseChange: the lastGmPhaseChange of the most recently received PortSyncSync member. The data type for lastGmPhaseChange is ScaledNs.

10.2.11.1.11 lastGmFreqChange: the lastGmFreqChange of the most recently received PortSyncSync member. The data type for lastGmPhaseChange is Double.

10.2.11.1.12 txMDSyncSendPtr: a pointer to the MDSyncSend structure sent to the MD entity of this port.

10.2.11.1.13 syncReceiptTimeoutTime: the value of the syncReceiptTimeoutTime member of the most recently received PortSyncSync structure. The data type for syncReceiptTimeoutTime is UScaledNs.

10.2.11.1.14 numberSyncTransmissions: a count of the number of consecutive Sync message transmissions after the SyncIntervalSetting state machine (see Figure 11-10) has set syncSlowdown (see 10.2.4.17) to TRUE. The data type for numberSyncTransmissions is UInteger8.

10.2.11.1.15 interval1: a local variable that holds either syncInterval or oldSyncInterval. The data type for interval1 is UScaledNs.

10.2.11.2 State machine functions

10.2.11.2.1 setMDSync(): creates an MDSyncSend structure, and returns a pointer to this structure. The members are set as follows:

a) sourcePortIdentity is set to the portIdentity of this port(see 8.5.2)
b) logMessageInterval is set equal to the value of currentLogSyncInterval for this port (see 10.7.2.3),
c) preciseOriginTimestamp is set equal to lastPreciseOriginTimestamp (see 10.2.11.1.3),
d) rateRatio is set equal to lastRateRatio (see 10.2.11.1.5),
e) followUpCorrectionField is set equal to lastFollowUpCorrectionField (see 10.2.11.1.4),
f) upstreamTxTime is set equal to lastUpstreamTxTime (see 10.2.11.1.6),

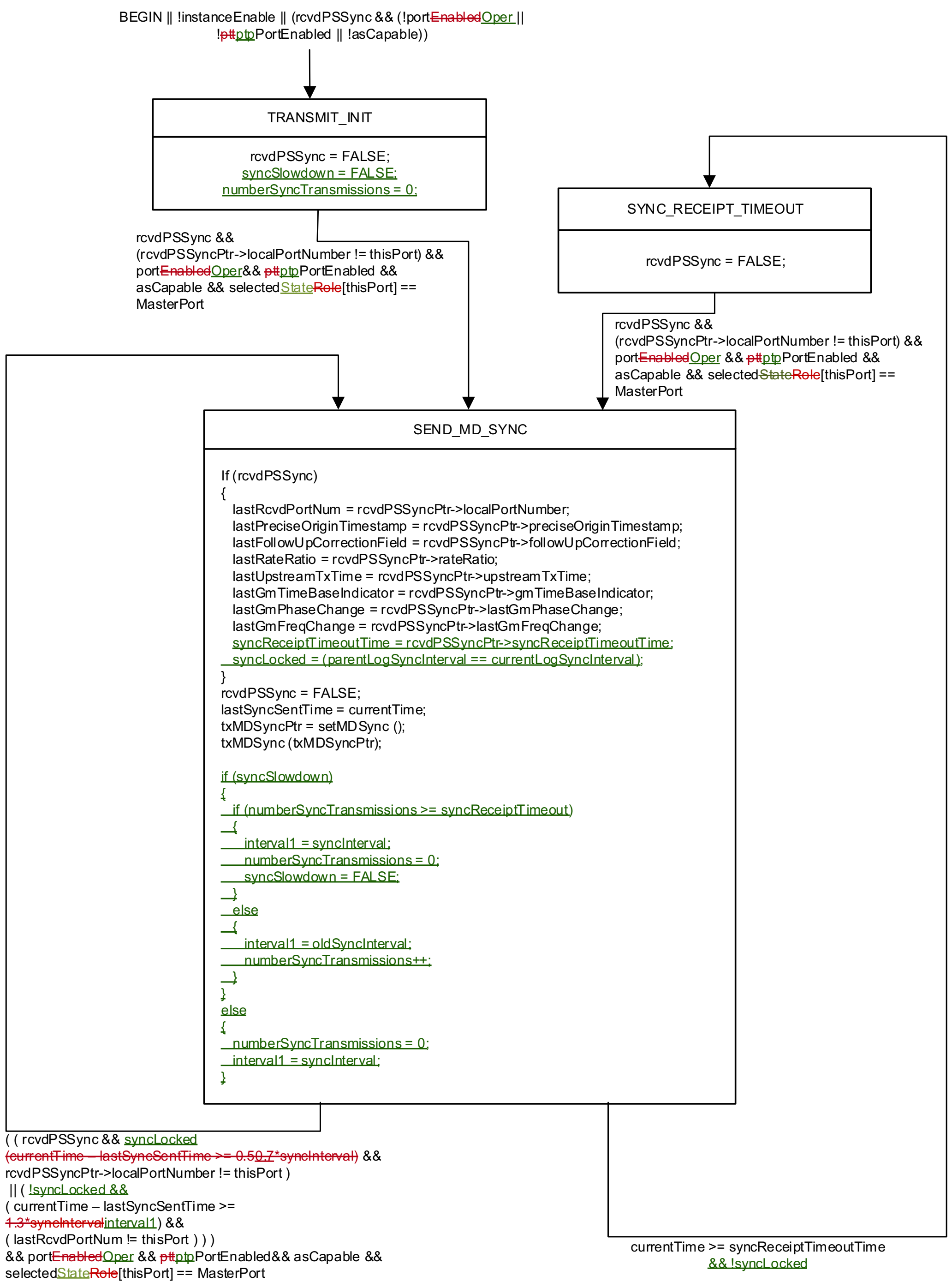
g) gmTimeBaseIndicator is set to lastGmTimeBaseIndicator (see 10.2.11.1.9),
h) lastGmPhaseChange (structure member) is set to lastGmPhaseChange (see 10.2.11.1.10),
i) lastGmFreqChange (structure member) is set to lastGmFreqChange (see 10.2.11.1.11), and
j) domainNumber is set equal to the domain number of this gPTP domain (see 8.1).

10.2.11.2.2 txMDSync(txMDSyncPtr): transmits the MDSyncSend structure pointed to by txMDSyncSendPtr, to the MDSyncSendSM state machine of the MD entity of this port.

10.2.11.3 State diagram

The PortSyncSyncSend state machine shall implement the function specified by the state diagram in Figure 10-8, the local variables specified in 10.2.11.1, the functions specified in 10.2.11.2, the structures specified in 10.2.2.1 through 10.2.2.3, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. The state machine receives time-synchronization information from the SiteSyncSync state machine, corresponding to the receipt of the most recent synchronization information on either the slave port, if this time-aware system is not the grandmaster, or from the ClockMasterSyncSend state machine, if this time-aware system is the grandmaster. The state machine causes time-synchronization information to be sen~~d~~t to the MDSyncSendSM state machine if this port is a MasterPort.

<<Editor's note: The changes to Figure 10-8 are (a) the elimination of the SET_SYNC_RECEIPT_TIMEOUT_TIME state, (b) modifications to the transition condition from SEND_MD_SYNC back to itself, indicated via strikethrough and underlines, to implement the syncLocked vs non-syncLocked modes, and (c) changing portEnabled to portOper and pttPortEnabled to ptpPortEnabled. >>


BEGIN || !instanceEnable || (rcvdPSSync && (!port~~Enabled~~Oper || !~~ptt~~ptpPortEnabled || !asCapable))
TRANSMIT_INIT
rcvdPSSync = FALSE;
syncSlowdown = FALSE;
numberSyncTransmissions = 0;
rcvdPSSync && (rcvdPSSyncPtr->localPortNumber != thisPort) && port~~Enabled~~Oper&& ~~ptt~~ptpPortEnabled && asCapable && selectedState~~Role~~[thisPort] == MasterPort
SYNC_RECEIPT_TIMEOUT
rcvdPSSync = FALSE;
rcvdPSSync && (rcvdPSSyncPtr->localPortNumber != thisPort) && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && asCapable && selected~~State~~~~Role~~[thisPort] == MasterPort
SEND_MD_SYNC
((rcvdPSSync && syncLocked ~~(currentTime – lastSyncSentTime >= 0.50.7*syncInterval)~~ && rcvdPSSyncPtr->localPortNumber != thisPort) || (!syncLocked && (currentTime – lastSyncSentTime >= ~~1.3*syncInterval~~interval1) && (lastRcvdPortNum != thisPort))) && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled&& asCapable && selectedState~~Role~~[thisPort] == MasterPort
currentTime >= syncReceiptTimeoutTime && !syncLocked

```
If (rcvdPSSync)
{
  lastRcvdPortNum = rcvdPSSyncPtr->localPortNumber;
  lastPreciseOriginTimestamp = rcvdPSSyncPtr->preciseOriginTimestamp;
  lastFollowUpCorrectionField = rcvdPSSyncPtr->followUpCorrectionField;
  lastRateRatio = rcvdPSSyncPtr->rateRatio;
  lastUpstreamTxTime = rcvdPSSyncPtr->upstreamTxTime;
  lastGmTimeBaseIndicator = rcvdPSSyncPtr->gmTimeBaseIndicator;
  lastGmPhaseChange = rcvdPSSyncPtr->lastGmPhaseChange;
  lastGmFreqChange = rcvdPSSyncPtr->lastGmFreqChange;
  syncReceiptTimeoutTime = rcvdPSSyncPtr->syncReceiptTimeoutTime;
  syncLocked = (parentLogSyncInterval == currentLogSyncInterval);
}
rcvdPSSync = FALSE;
lastSyncSentTime = currentTime;
txMDSyncPtr = setMDSync ();
txMDSync (txMDSyncPtr);

if (syncSlowdown)
{
  if (numberSyncTransmissions >= syncReceiptTimeout)
  {
    interval1 = syncInterval;
    numberSyncTransmissions = 0;
    syncSlowdown = FALSE;
  }
  else
  {
    interval1 = oldSyncInterval;
    numberSyncTransmissions++;
  }
}
else
{
  numberSyncTransmissions = 0;
  interval1 = syncInterval;
}
```

Figure 10-8—PortSyncSyncSend state machine

10.2.12 ClockSlaveSync state machine

10.2.12.1 State machine variables

10.2.12.1.1 rcvdPSSync: 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.

10.2.12.1.2 rcvdLocalClockTick: a Boolean variable that notifies the current state machine when the LocalClock entity updates its time. This variable is reset by this state machine.

10.2.12.1.3 rcvdPSSyncPtr: a pointer to the received PortSyncSync structure.

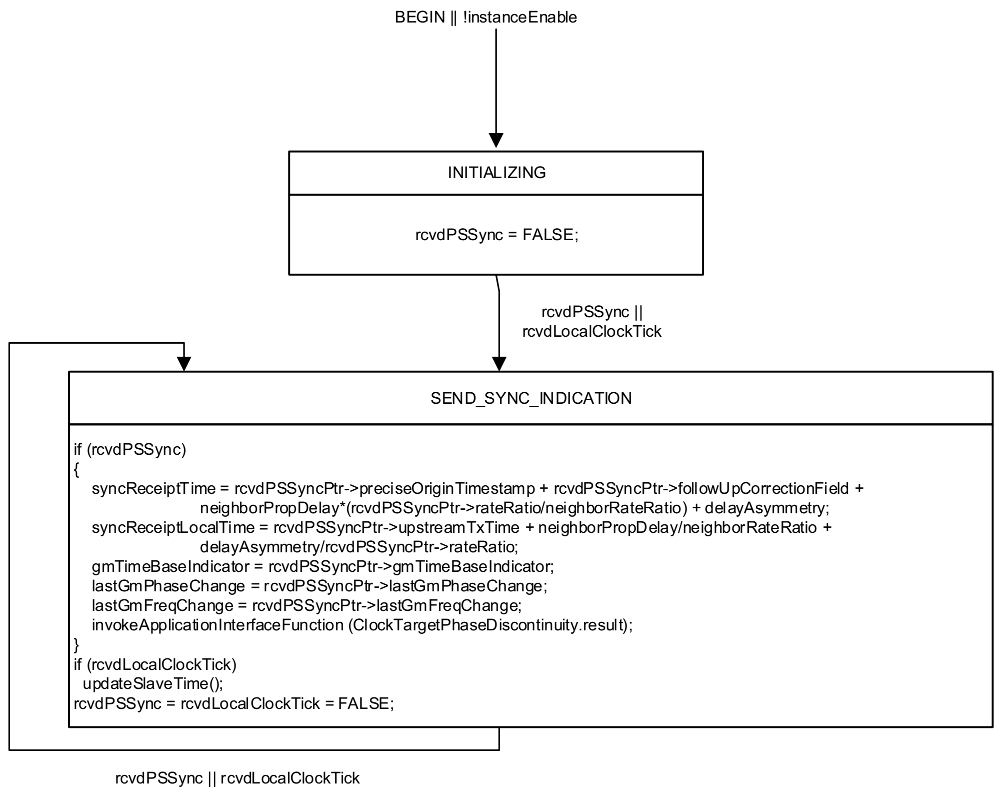
10.2.12.2 State machine functions

10.2.12.2.1 updateSlaveTime(): updates the global variable clockSlaveTime (see 10.2.3.3), based on information received from the SiteSync and LocalClock entities. It is the responsibility of the application to filter slave times appropriately (see B.3 and B.4 for examples). As one example, clockSlaveTime can be incremented by localClockTickInterval (see 10.2.3.18) multiplied by rateRatio member of the received PortSyncSync structure. If no time-aware system is grandmaster-capable, i.e., gmPresent is FALSE, then clockSlaveTime is set to the time provided by the LocalClock. This function is invoked when rcvdLocalClockTick is TRUE.

10.2.12.2.2 invokeApplicationInterfaceFunction (functionName): invokes the application interface function whose name is functionName. In the case here, functionName is clockTargetPhaseDiscontinuity.result (see 9.6.2).

10.2.12.3 State diagram

The ClockSlaveSync state machine shall implement the function specified by the state diagram in Figure 10-9, the local variables specified in 10.2.12.1, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. 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 ClockMasterSyncOffset 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.



```
if (rcvdPSSync)
{
    syncReceiptTime = rcvdPSSyncPtr->preciseOriginTimestamp + rcvdPSSyncPtr->followUpCorrectionField +
                        neighborPropDelay*(rcvdPSSyncPtr->rateRatio/neighborRateRatio) + delayAsymmetry;
    syncReceiptLocalTime = rcvdPSSyncPtr->upstreamTxTime + neighborPropDelay/neighborRateRatio +
                        delayAsymmetry/rcvdPSSyncPtr->rateRatio;
    gmTimeBaseIndicator = rcvdPSSyncPtr->gmTimeBaseIndicator;
    lastGmPhaseChange = rcvdPSSyncPtr->lastGmPhaseChange;
    lastGmFreqChange = rcvdPSSyncPtr->lastGmFreqChange;
    invokeApplicationInterfaceFunction (ClockTargetPhaseDiscontinuity.result);
}
if (rcvdLocalClockTick)
  updateSlaveTime();
rcvdPSSync = rcvdLocalClockTick = FALSE;
```

Figure 10-9—ClockSlaveSync state machine

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

10.3.1 Best master clock ~~algorithm~~selection overview

There are two methods for setting the grandmaster and time-synchronization spanning tree for a gPTP domain:

a) The BMCA is used to determine the grandmaster for a gPTP domain and construct the time-synchronization spanning tree with that grandmaster as the root. In this case, the network is configured automatically, using the results of the BMCA.

b) The port states are configured to force a desired grandmaster, and to construct a desired time-synchronization spanning tree with the grandmaster as the root.

The per-time-aware-system global variable externalPortConfiguration indicates whether (a) or (b) is used; a value of ENABLED indicates (b) and a value of DISABLED indicates (a) (see 10.3.8.24). Method (a) shall be implemented, and shall be the default mode of operation (i.e., externalPortConfiguration is DISABLED), on domain 0 to maintain backward compatibility. For domains other than domain 0:

c) at least one of the possibilities (a) or (b) shall be implemented,
d) both possibilities may be implemented, and
e) if both possibilities are implemented, the default value of externalPortConfiguration shall be DISABLED.

[Editor's Note: It is believed that the above accommodates both applications that will not use the BMCA, applications that will use only the BMCA, and applications that will use the BMCA and external port state configuration.]

~~The BMCA determines the grandmaster for a gPTP domain and constructs a time-synchronization spanning tree with the grandmaster as the root.~~ In both (a) and (b), s~~S~~ynchronized time is transported from the grandmaster to other time-aware systems via the time-synchronization spanning tree. In (a), b~~B~~est master selection information is exchanged between time-aware systems via Announce messages (see 10.5 and 10.6). In (b), Announce messages are used to transport information on the time-synchronization spanning tree and grandmaster time properties information from one time-aware system to the next in the tree. Each Announce message contains time-synchronization spanning tree vector information that identifies one time-aware system as the root of the time-synchronization spanning tree and, if the time-aware system is grandmaster-capable, the grandmaster. Each time-aware system in turn uses the information contained in the Announce messages it receives, along with its knowledge of itself, to: (i) in (a), compute which of the time-aware systems that it has knowledge of ~~should~~ought to be the root and, if grandmaster-capable, the grandmaster; and (ii) in (b), set the information on the spanning tree connectivity and grandmaster time properties.

Once an Announce message is transmitted by a port, subsequent timing information (see 7.4) transmitted by that port shall be derived from the grandmaster indicated in that Announce message.

As part of constructing the time-synchronization spanning tree, each port of each time-aware system is assigned a port ~~role~~state from Table 10-1 by state machines associated with the ports and with the time-aware system as a whole.

Table 10-1—Port ~~role~~state definitions

Port ~~role~~state	Description
MasterPort	Any port, P, of the time-aware system that is closer to the root than any other port of the gPTP communication path connected to P
SlavePort	The one port of the time-aware system that is closest to the root time-aware system. If the root is grandmaster-capable, the SlavePort is also closest to the grandmaster. The time-aware system does not transmit Sync or Announce messages on the SlavePort.
PassivePort	Any port of the time-aware system whose port ~~role~~state is not MasterPort, SlavePort, or DisabledPort.
DisabledPort	Any port of the time-aware system for which ~~portEnabled~~ portOper, ~~pttPortEnabled~~ptpPortEnabled, and asCapable are not all TRUE.
NOTE—Port states are per port and per domain (i.e., per PTP instance), see 8.1.	

NOTE 1—Information contained in Sync and associated Follow_Up messages received on ports whose port ~~role~~state is PassivePort is discarded; the SiteSyncSync state machine (see 10.2.6) uses only information received from a port whose port ~~role~~state is SlavePort.

An example master/slave hierarchy of time-aware systems is shown in Figure 10-10. The grandmaster ports all have port ~~role~~state of MasterPort. All the other time-aware systems have exactly one slave port. The time-synchronization spanning tree is ~~formed by~~composed of the time-aware systems and the links that do not have an endpoint port whose port ~~role~~state is PassivePort.

NOTE 2—The BMCA described in this standard is equivalent to a subset of the BMCA described in IEEE Std 1588-2008. It is also equivalent to a subset of the Rapid Spanning Tree Protocol (RSTP) described in IEEE Std 802.1Q~~D~~ (though the full RSTP described in the latter standard is not equivalent to the full BMCA described in the former standard). The BMCA description here uses the formalism of the RSTP description in the latter standard.

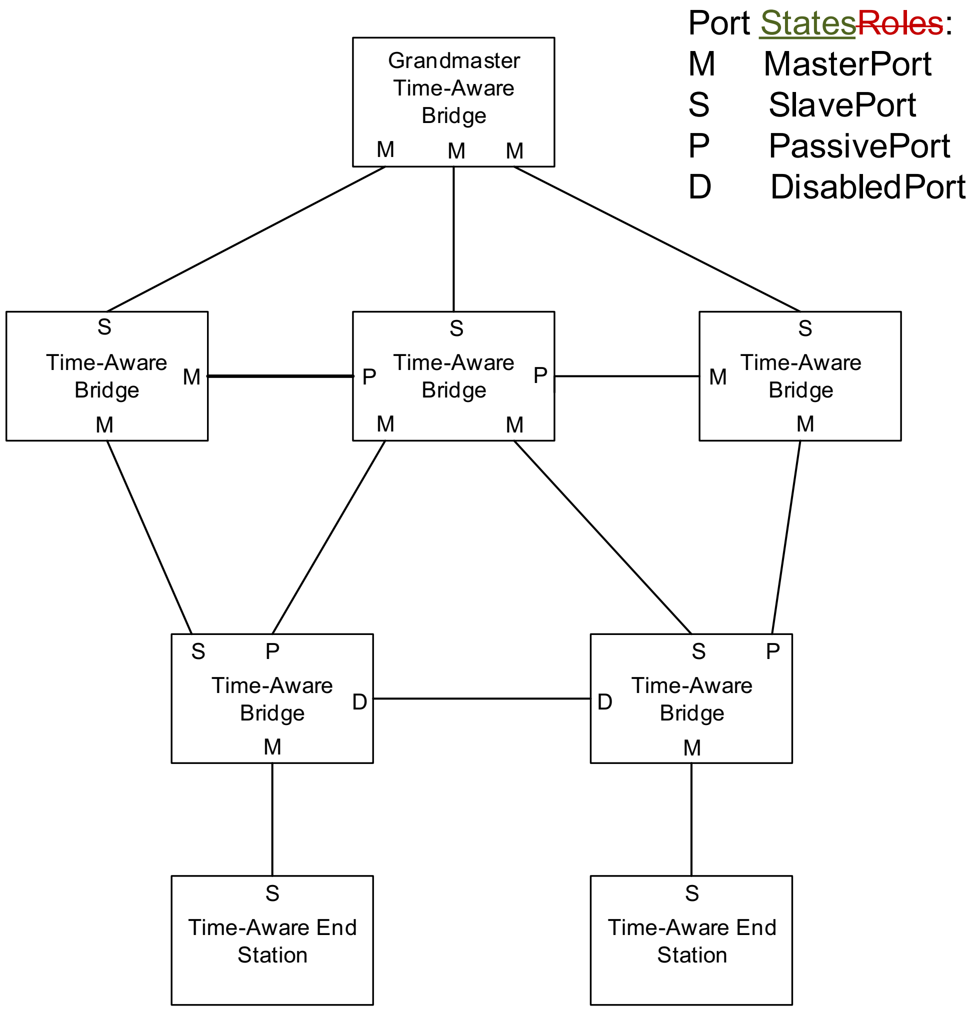


Figure 10-10—Example master/slave hierarchy of time-aware systems

10.3.2 systemIdentity

The systemIdentity attribute of a time-aware system is a UInteger112 (i.e., a 14-byte, unsigned integer) formed by concatenating the following attributes, in the following order, from most significant to least significant bit:

a) priority1 (1 octet, see 8.6.2.1)
b) clockClass (1 octet, see 8.6.2.2 and 6.4.3.8)
c) clockAccuracy (1 octet, see 8.6.2.3 and 6.4.3.8)
d) offsetScaledLogVariance (2 octets, see 8.6.2.4 and 6.4.3.8)
e) priority2 (1 octet, see 8.6.2.5)
f) clockIdentity (8 octets, see 8.5.2.2 and 6.4.3.6)

The systemIdentity attribute is defined for convenience when comparing two time-aware systems to determine, when using the BMCA (i.e., (a) of 10.3.1) which is a better candidate for root and if the time-aware system is grandmaster-capable (i.e., the value of priority1 is less than 255, see 8.6.2.1). Two time-aware systems are compared as follows. Let the systemIdentity of time-aware system A be S_A and the systemIdentity of time-aware system B be S_B. Let the clockIdentity of A be C_A and the clockIdentity of B be C_B. Then, if $C_A \neq C_B$, i.e., A and B represent different time-aware systems:

g) A is better than B if and only if $S_A < S_B$, and
h) B is better than A if and only if $S_B < S_A$.

If $C_A = C_B$, i.e., A and B represent the same time-aware system:

i) $S_A < S_B$ means that A represents an upgrading of the time-aware system compared to B or, equivalently, B represents a downgrading of the time-aware system compared to A,
j) $S_B < S_A$ means that B represents an upgrading of the time-aware system compared to A or, equivalently, A represents a downgrading of the time-aware system compared to B, and
k) $S_A = S_B$ means that A and B represent the same time-aware system that has not changed.

Comparisons g) and h) above imply that, with the ordering of attributes in the systemIdentity, the clockIdentity is a tie-breaker for the case where two different time-aware systems that have identical attributes a) through e) are compared.

Comparisons g) and h) also imply that a time-aware system that is grandmaster-capable is always better than another time-aware system that is not grandmaster-capable, because the priority1 is less than 255 if the time-aware system is grandmaster-capable and 255 if it is not grandmaster-capable (see 8.6.2.1).

The cases where A and B represent different time-aware systems and represent the same time-aware system are handled separately in the BMCA. When comparing two different time-aware systems, the better time-aware system is selected as the grandmaster candidate. However, if A and B represent the same time-aware system with attributes that have changed, the time-aware system is considered as having the most recent attributes when doing subsequent comparisons with other time-aware systems.

10.3.3 stepsRemoved

Every time-aware system has a stepsRemoved associated with it. For the root time-aware system, and therefore the grandmaster in the case where the root is grandmaster-capable, it is zero. For all other time-aware systems, it is the number of links in the path from the root to the respective time-aware system.

The stepsRemoved attributes of different ports of a time-aware system are compared after comparisons of other attributes that take precedence (i.e., priority1, clockClass, clockAccuracy, offsetScaledLogVariance, priority2) do not result in one port being declared better than the other. Among those ports whose stepsRemoved attributes are compared, t~~T~~he port on ~~each~~the time-aware system with the lowest stepsRemoved is assigned the ~~role~~state of SlavePort for that time-aware system (the root time-aware system does not have a SlavePort). This lowest stepsRemoved is also considered the stepsRemoved for the time-aware system. If a time-aware system has two or more ports with the same stepsRemoved, then the port with the smallest portNumber is selected as the SlavePort.

~~Each gPTP communication path in the network also has an associated stepsRemoved. This is the stepsRemoved of the lowest stepsRemoved time-aware system with a port attached to that communication path. That time-aware system is selected as the master time-aware system for that communication path. Every communication path has exactly one port whose role~~state ~~is MasterPort; the time-aware system of that port is the master time-aware system for that communication path. If there are two or more time-aware systems attached to the communication path with the same stepsRemoved, then the better of the two time-aware systems as determined by the systemIdentity is selected as the master time-aware system.~~

[Editor's note: The above text remains deleted, and text in the previous paragraph is modified. This is in accordance with the resolution of comment #129 against D4.5.>>

10.3.4 time-synchronization spanning tree priority vectors

Time-aware systems send best master selection information to each other in Announce messages. The information ~~may be~~is structured in a time-synchronization spanning tree priority vector. T~~t~~ime-synchronization spanning tree priority vectors provide the basis for a concise specification of the BMCA's determination of the time-synchronization spanning tree and grandmaster. A priority vector is formed by concatenating the following attributes, in the following order, from most significant to least significant bit:

a) rootSystemIdentity (14 octets, see 10.3.2)
b) stepsRemoved (2 octets, see 10.3.3)
c) sourcePortIdentity (i.e., portIdentity of the transmitting time-aware system; 10 octets, see 8.5.2 and 10.6.2)
d) portNumber of the receiving port (2 octets, see 8.5.2.3)

The first two components of a priority vector are significant throughout the gPTP domain; they are propagated via Announce messages and updated through invocation of BMCA state machines. The next component is assigned hop-by-hop for each communication path or time-aware system, and thus is of local significance only. It is used as a tie-breaker in decisions between time-synchronization spanning tree priority vectors that are otherwise equal. ~~The next component is locally significant; it is assigned hop by hop for each communication path or time-aware system and used as a tie-breaker in decisions between time-synchronization spanning tree priority vectors that are otherwise equal.~~ The fourth component is not conveyed in Announce messages, but is used as a tie-breaker within a time-aware system.

The set of all time-synchronization spanning tree priority vectors is totally ordered. For all components, a lesser numerical value is better, and earlier components in the preceding list are more significant. In addition, as mentioned earlier, a priority vector that reflects a root time-aware system that is grandmaster-capable is always better th~~e~~an a priority vector that reflects a root time-aware system that is not grandmaster-capable. As each time-aware system port receives a priority vector, via an Announce message, from ports closer to the root, additions are made to one or more components to yield a worse priority vector. This process of receiving information, adding to it, and passing it on, can be described in terms of the message priority vector received and a set of priority vectors used to facilitate the computation of a priority vector for each port, to be transmitted in further Announce Messages to time-aware systems further from the root.

10.3.5 Priority vector calculations

The portPriorityVector is the time-synchronization spanning tree priority vector held for the port when the reception of Announce messages and any pending update of information has been completed:

portPriorityVector = {rootSystemIdentity : stepsRemoved : sourcePortIdentity : portNumber}

A messagePriorityVector is the time-synchronization spanning tree priority vector conveyed in a received Announce Message. For a time-aware system S receiving an Announce Message on Port P_S with portNumber PN_S, from a MasterPort with portIdentity P_M on time-aware system M claiming a rootSystemIdentity of R_M and a stepsRemoved of SR_M:

messagePriorityVector = $\{R_M : SR_M : P_M : PN_S\}$

This messagePriorityVector is superior to the portPriorityVector and will replace it if, and only if, the messagePriorityVector is better than the portPriorityVector, or the Announce message has been transmitted from the same master time-aware system and MasterPort as the portPriorityVector, i.e., if the following is true:

((R_M < rootSystemIdentity)) ||

((R_M == rootSystemIdentity) && (SR_M < stepsRemoved)) ||

((R_M == rootSystemIdentity) && (SR_M == stepsRemoved) && (P_M < sourcePortIdentity (of current master time-aware system))) ||

((R_M == rootSystemIdentity) && (SR_M == stepsRemoved)
&& (P_M == sourcePortIdentity (of current master time-aware system)) && (PN_S < portNumber)) ||

((P_M.clockIdentity == sourcePortIdentity.clockIdentity (of current master time-aware system)) &&
(P_M.portNumber == sourcePortIdentity.PortNumber (of the current master time-aware system)))

A gmPathPriorityVector can be calculated from a received portPriorityVector by adding one to the stepsRemoved component:

gmPathPriorityVector = $\{R_M : SR_M + 1 : P_M : PN_S\}$

The systemPriorityVector for a time-aware system S with systemIdentity S_S and clockIdentity C_S is the priority vector that would, with the portIdentity of the SlavePort set equal to the portIdentity of the transmitting port, be used as the message priority vector in Announce Messages transmitted on S's ports whose ~~role~~state is MasterPort if S was selected as the root:

systemPriorityVector = $\{S_S : 0 : \{C_S : 0\} : 0\}$

The gmPriorityVector for S is the best of the set comprising the systemPriorityVector vector plus every gmPathPriorityVector for which the clockIdentity of the master time-aware system portIdentity is not the clockIdentity of S. If the systemPriorityVector is best, S has been selected as the root. For the case where the best gmPathPriorityVector is that of port PN_S above, then:

gmPriorityVector = $\{S_S : 0 : \{C_S : 0\} : 0\}$if S is better than R_M, or

gmPriorityVector = $\{R_M : SR_M + 1 : P_M : PN_S\}$ if S is worse than R_M.

The masterPriorityVector for a port Q on time-aware system S is the gmPriorityVector with S's clockIdentity C_S substituted for the clockIdentity of the master portIdentity, and Q's portNumber PN_Q substituted for the portNumber of the master portIdentity and for the portNumber of the receiving port:

masterPriorityVector = $\{S_S : 0 : \{C_S : PN_Q\} : PN_Q\}$if S is better than R_M, or
masterPriorityVector = $\{\{R_M : SR_M + 1 : \{C_S : PN_Q\} : PN_Q\}$ if S is worse than R_M.

If the masterPriorityVector is better than the portPriorityVector, the port will be the MasterPort for the attached communication path and the portPriorityVector will be updated. The messagePriorityVector information in Announce messages transmitted by a port always includes the first three components of the masterPriorityVector of the port.

NOTE—The consistent use of lower numerical values to indicate better information is deliberate as the MasterPort that is closest to the root, i.e., has a numerically lowest path cost component, is selected from amongst potential alternatives for any given communication path. Adopting the conventions that lower numerical values indicate better information, that where possible more significant priority components are encoded earlier in the octet sequence of an Announce message, and that earlier octets in the encoding of individual components are more significant, allow concatenated octets that compose a priority vector to be compared as if they were a multiple octet encoding of a single number, without regard to the boundaries between the encoded components. To reduce the confusion that naturally arises from having the lesser of two numerical values represent the better of the two, i.e., the one to be chosen all other factors being equal, this clause uses the following consistent terminology. Relative numeric values are described as “least,” “lesser,” “equal,” and “greater,” and their comparisons as “less than,” “equal to,” or “greater than,” while relative time-synchronization spanning tree priorities are described as “best,” “better,” “the same,” “different,” and “worse” and their comparisons as “better than,” “the same as,” “different from,” and “worse than.” The operators “<” and “==” represent less than and equal to respectively. The terms “superior” and “inferior” are used for comparisons that are not simply based on priority but can include the fact that the priority vector of a MasterPort can replace an earlier vector transmitted in an Announce message by the same port.

10.3.6 Port ~~role~~state assignments

The BMCA assigns one of the following port ~~role~~states to each time-aware system Port: MasterPort, SlavePort, PassivePort, or DisabledPort.

The DisabledPort ~~role~~state is assigned if ~~portEnabled~~portOper is FALSE (see 10.2.4.12), ~~pttPortEnabled~~ptpPortEnabled is FALSE (see 10.2.4.13), or asCapable is FALSE (see 10.2.4.1).

A port for which ~~portEnabled~~portOper, ~~pttPortEnabled~~ptpPortEnabled, and asCapable are all TRUE has its port ~~role~~state assigned according to the source and relative priority of the time-synchronization spanning tree portPriorityVector (see 10.3.4 and 10.3.5) as follows:

a) If the time-aware system is not the root, the source of the gmPriorityVector is the SlavePort.
b) Each port whose portPriorityVector is its masterPriorityVector is a MasterPort.
c) Each Port, other than the SlavePort, whose portPriorityVector has been received from another time-aware system or another port on this time-aware system is a PassivePort.

10.3.7 Overview of best master clock selection and announce interval setting state machines

[Editor’s note: This section could be expanded to also give an overview of external port state configuration. This would require preparing an analogous figure to Figure 10-11 below. Input is requested on whether this would be useful.]

The best master clock selection function in a time-aware system is specified by a number of cooperating state machines. Figure 10-11 is not itself a state machine, but illustrates the machines, their interrelationships, the principle variables and structures used to communicate between them, their local variables, and performance parameters.

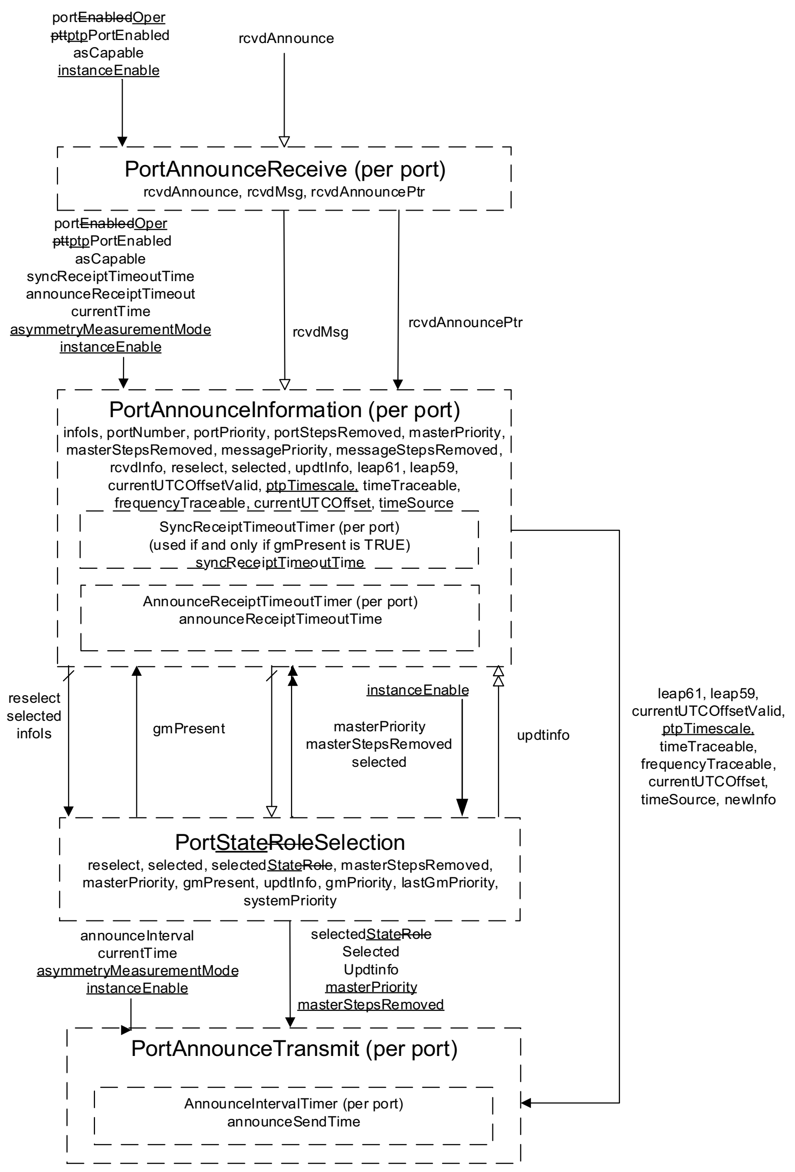
NOTE—The BMCA state machines are all invoked by the media-independent layer, i.e., by the SiteSync and PortSync entities. The media-dependent layer, i.e., the MD entity, simply takes an Announce message received from the PortSync entity of the same port and gives it to the next lower layer (e.g., IEEE 802.3, IEEE 802.11). It is the PortSync entity that generates and consumes Announce messages.

The media-independent layer state machines in Figure 10-11 are:

a) PortAnnounceReceive (one instance per port): receives Announce information from the MD entity of the same port, determines if the Announce message is qualified and, if so, sets the rcvdMsg variable. This state machine is invoked by the PortSync entity of the port.
b) PortAnnounceInformation (one instance per port): receives new qualified Announce information from the PortAnnounceReceive state machine, determines if the Announce information is better than the current best master information it knows about, and updates the current best master information when it receives updated port ~~role~~state information from the Port~~Role~~StateSelection state machine and when announce receipt timeout or, in the case where gmPresent is TRUE, sync receipt timeout occurs. This state machine is invoked by the PortSync entity of the port.
c) Port~~Role~~StateSelection (one instance per time-aware system): updates the gmPathPriority vector for each port of the time-aware system, the gmPriorityVector for the time-aware system, and the masterPriorityVector for each port of the time-aware system; determines the port ~~role~~state for each port; and updates gmPresent. This state machine is invoked by the SiteSync entity of the time-aware system.
d) PortAnnounceTransmit (one instance per port): transmits Announce information to the MD entity when an announce interval has elapsed, port ~~role~~states have been updated, and portPriority and portStepsRemoved information has been updated with newly determined masterPriority and masterStepsRemoved information. This state machine is invoked by the PortSync entity of the port.

An additional state machine, the AnnounceIntervalSetting state machine, receives a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3), and sets the global variables that give the duration of the mean interval between successive Announce messages.

<<Editor's note: In Figure 10-11, portEnabled is changed to portOper, and pttPortEnabled is changed to ptpPortEnabled.>


port~~Enabled~~Oper
~~ptt~~ptpPortEnabled
asCapable
instanceEnable
rcvdAnnounce
PortAnnounceReceive (per port)
rcvdAnnounce, rcvdMsg, rcvdAnnouncePtr
port~~Enabled~~Oper
~~ptt~~ptpPortEnabled
asCapable
syncReceiptTimeoutTime
announceReceiptTimeout
currentTime
asymmetryMeasurementMode
instanceEnable
rcvdMsg
rcvdAnnouncePtr
PortAnnounceInformation (per port)
infoIs, portNumber, portPriority, portStepsRemoved, masterPriority, masterStepsRemoved, messagePriority, messageStepsRemoved, rcvdInfo, reselect, selected, updtInfo, leap61, leap59, currentUTCOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUTCOffset, timeSource
SyncReceiptTimeoutTimer (per port)
(used if and only if gmPresent is TRUE)
syncReceiptTimeoutTime
AnnounceReceiptTimeoutTimer (per port)
announceReceiptTimeoutTime
reselect
selected
infoIs
gmPresent
instanceEnable
masterPriority
masterStepsRemoved
selected
updtinfo
leap61, leap59,
currentUTCOffsetValid,
ptpTimescale,
timeTraceable,
frequencyTraceable,
currentUTCOffset,
timeSource, newInfo
Port~~Role~~StateSelection
reselect, selected, selected~~Role~~State, masterStepsRemoved, masterPriority, gmPresent, updtInfo, gmPriority, lastGmPriority, systemPriority
announceInterval
currentTime
asymmetryMeasurementMode
instanceEnable
selected~~Role~~State
Selected
Updtinfo
masterPriority
masterStepsRemoved
PortAnnounceTransmit (per port)
AnnounceIntervalTimer (per port)
announceSendTime

Notation:
Variables are shown both within the machine where they are principally used and between machines where they are used to communicate information. In the latter case a variety of arrow styles, running from one machine to another, show how each is typically used:

→ Not changed by the target machine. Where the machines are both per ~~P~~port, this variable communicates between instances for the same port

▷ Set (or cleared) by the originating machine, cleared (or set) by the target machine. Where the machines are both per ~~P~~port, this communicates between instances for the same port.

↠ ▷▷ As above, except that the originating per-port machine instance communicates with multiple port machine instances (by setting or clearing variables owned by those ports).

As above, except that multiple per-~~P~~port instances communicate with (an)other instance(s) (by setting or clearing variables owned by the originating ports).

Figure 10-11—Best master clock selection state machines—overview and interrelationships

10.3.8 Per-time-aware-system global variables

10.3.8.1 reselect: a Boolean array of length numberPorts+1 (see 8.6.2.8). Setting reselect[j], where 0 ≤ j ≤ numberPorts, to TRUE causes the ~~ROLE~~STATE_SELECTION block of the Port~~Role~~StateSelection state machine (see 10.3.12) to be re-entered, which in turn causes the port ~~role~~state of each port of the time-aware system to be updated (via the function updt~~Role~~StatesTree(), see 10.3.12.2.4). This variable is used only by the BMCA, i.e., not by the explicit port state configuration option.

10.3.8.2 selected: a Boolean array of length numberPorts+1 (see 8.6.2.8). selected[j], where 0 ≤ j ≤ numberPorts, is set to TRUE immediately after the port ~~role~~states of all the ports are updated. This indicates to the PortAnnounceInformation state machine (see 10.3.11) that it can update the portPriorityVector and other variables for each port. This variable is used both both the BMCA and the explicit port state configuration option; however, its value does not impact the explicit port state configuration option (see the NOTE in 10.3.15.3, just after Figure 10-17).

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 is done only for convenience, so that array element j can correspond to port j. Note also that, in contrast, selectedState[0] is used (see 10.2.3.20).

10.3.8.3 masterStepsRemoved: the value of stepsRemoved for the time-aware system, after the port ~~role~~states of all the ports have been updated (see 10.3.12.2.4 for details on the computation of masterStepsRemoved). This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.4 leap61: a Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the current grandmaster, contains 61 s, and FALSE if the last minute of the current UTC day does not contain 61 s. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.5 leap59: a Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the current grandmaster, contains 59 s, and FALSE if the last minute of the current UTC day does not contain 59 s. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.6 currentUtcOffsetValid: a Boolean variable whose value is TRUE if currentUtcOffset (see 10.3.8.10), relative to the current grandmaster, is known to be correct, and FALSE if currentUtcOffset is not known to be correct. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.7 ptpTimescale: a Boolean variable whose value is TRUE if the timescale of the current grandmaster is PTP (see 8.2.1) and FALSE if the timescale is ARB. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.8 timeTraceable: a Boolean variable whose value is TRUE if both clockSlaveTime (i.e., the synchronized time maintained at the slave~~,~~ (see 10.2.3.3)) and currentUtcOffset (see 10.3.8.10) relative to the current grandmaster are traceable to a primary reference, and FALSE if one or both are not traceable to a primary reference. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.9 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 10.2.7.1.4), is traceable to a primary reference, and FALSE if this frequency is not traceable to a primary reference. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.10 currentUtcOffset: the difference between TAI time and UTC time, i.e., TAI time minus UTC time, expressed in seconds, and relative to the current grandmaster. The data type for currentUtcOffset is Integer16. This variable is used both both the BMCA and the explicit port state configuration option.

<<Editor's note: In accordance with the resolution of comment #65 against D4.5, white space was was added here to force the NOTE to the next page so that it will be on the same page as its footnote 15. This seems to be a Framemaker problem. This editor's note and any white space will be removed prior to sponsor ballot, though the white space will be retained if needed to keep the NOTE and footnote on the same page.

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.[15]

10.3.8.11 timeSource: the value of the timeSource attribute of the current grandmaster (see 8.6.2.7). The data type for timeSource is Enumeration8. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.12 sysLeap61: a Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the ClockMaster entity of this time-aware system, contains 61 s, and FALSE if the last minute of the current UTC day does not contain 61 s. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.13 sysLeap59: a Boolean variable whose value is TRUE if the last minute of the current UTC day, relative to the ClockMaster entity of this time-aware system, contains 59 s, and FALSE if the last minute of the current UTC day does not contain 59 s. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.14 sysCurrentUTCOffsetValid: a Boolean variable whose value is TRUE if currentUtcOffset (see 10.3.8.10), relative to the ClockMaster entity of this time-aware system, is known to be correct, and FALSE if currentUtcOffset is not known to be correct. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.15 sysPtpTimescale: a Boolean variable whose value is TRUE if the timescale of the ClockMaster entity of this time-aware system is PTP (see 8.2.1), and FALSE if the timescale of the ClockMaster entity of this time-aware system is ARB. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.16 sysTimeTraceable: a Boolean variable whose value is TRUE if both masterTime (i.e., the time maintained by the ClockMaster entity of this time-aware system~~,~~ (see 10.2.3.21)) and currentUtcOffset (see 10.3.8.10) relative to the ClockMaster entity of this time-aware system, are traceable to a primary reference, and FALSE if one or both are not traceable to a primary reference. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.17 sysFrequencyTraceable: a Boolean variable whose value is TRUE if the frequency that determines masterTime of the ClockMaster entity of this time-aware system, i.e., the frequency of the LocalClockEntity multiplied by the most recently computed gmRateRatio by the ClockMasterSyncReceive state machine (see 10.2.3.14 and 10.2.10), is traceable to a primary reference, and FALSE if this frequency is not traceable to a primary reference. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.18 sysCurrentUtcOffset: the difference between TAI time and UTC time, i.e., TAI time minus UTC time, expressed in seconds, and relative to the ClockMaster entity of this time-aware system. The data type

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

for sysC~~c~~urrentUtcOffset is Integer16. This variable is used both both the BMCA and the explicit port state configuration option.

NOTE—See the NOTE in 10.3.8.10 for more detail on the sign convention.

10.3.8.19 sysTimeSource: the value of the timeSource attribute of the ClockMaster entity of this time-aware system (see 8.6.2.7). The data type for sysT~~t~~imeSource is Enumeration8.

10.3.8.20 systemPriority: the systemPriority vector for this time-aware system. The data type for ~~master~~systemPriority is UInteger224 (see 10.3.4).

10.3.8.21 gmPriority: the current gmPriorityVector for the time-aware system. The data type for ~~master~~gmPriority is UInteger224 (see 10.3.4).

10.3.8.22 lastGmPriority: the previous gmPriorityVector for the time-aware system, prior to the most recent invocation of the Port~~Role~~StateSelection state machine. The data type for ~~master~~lastGmPriority is UInteger224 (see 10.3.4). lastGmPriority is used only by the BMCA, i.e., not by the explicit port state configuration option.

10.3.8.23 pathTrace: an array that contains the clockIdentities of the successive time-aware systems that receive, process, and send Announce messages. The data type for pathTrace is ClockIdentity[N], where N is the number of time-aware systems, including the grandmaster, that the Announce information has traversed. This variable is used both both the BMCA and the explicit port state 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 maxium 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 entires in the pathTrace array (see 10.6.3.1 and Table 10-8) 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 10.3.10.2.1 and 10.3.15.2.1, and behaves as follows:

— Item (c) of the description of the qualifyAnnounce() function of the PortAnnounceReceive state machine (see 10.3.10.2.1) 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 the description of the 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 here) 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 the description of the txAnnounce() function of the PortAnnounceTransmit state machine (see 10.3.15.2.1) 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 any 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.
— These behaviors of the qualifyAnnounce() and txAnnounce() functions result in the path trace feature not being 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 10.3.10.2.1).

10.3.8.24 externalPortConfiguration: a variable whose value indicates whether port states are externally configured or determined by the BMCA. The data type shall be Enumeration8. The value ENABLED indicates that the port states are externally configured; the value DISABLED indicates that the port states are determined by the BMCA. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.8.25 lastAnnouncePort: the port number of the port on which the most recent Announce message was received. This variable is used by the PortAnnounceInformationExt and PortStateSettingExt state machines, for the external port state configuration option. This variable is not used by the BMCA. The data type for this variable is UInteger32.

[Editor's Note: The data type for lastAnnouncePort should be the same as for numberPorts. In Table 14.1, the data type for numberPorts is Integer8, while in the MIB it is UInteger32. See the Editor's note right after Table 14.1 and in Clause 15. It must be decided whether the data type should be UInteger16 or UInteger32. The data type is UInteger16 in the latest P1588-Rev draft.]

10.3.9 Per-port global variables

10.3.9.1 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 10.3.9.8) for the port at the other end of the link to which this port is attached. The value of announceInterval for the 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. The variable infoIs is used only by the BMCA, i.e., not by the explicit port state configuration option.

10.3.9.2 announceSlowdown: a Boolean that is set to TRUE if the AnnounceIntervalSetting state machine (see Figure 10-18) 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 PortAnnouncedTransmit state machine (see Figure 10-17) 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 both both the BMCA and the explicit port state configuration option.

NOTE—If a receiver of Announce messages requests a slower rate, the receiver will continue 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 consecutive Announce messages sent to the receiver are lost, announce receipt timout 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 this from happening.

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

10.3.9.4 infoIs: an Enumeration2 that takes the values Received, Mine, Aged, or Disabled, to indicate the origin and state of the port's time-synchronization spanning tree information:

a) If infoIs is Received, the 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 time-aware system for the attached communication path.

b) If infoIs is Mine, information for the port has been derived from the SlavePort for the time-aware system (with the addition of SlavePort stepsRemoved). This includes the possibility that the SlavePort is the port whose portNumber is 0, i.e., the time-aware system is the root of the gPTP domain.
c) If infoIs is Aged, announce receipt timeout or, in the case where gmPresent is TRUE, sync receipt timeout have occurred.
d) If ~~portEnabled~~portOper, ~~pttPortEnabled~~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 configuration option.

10.3.9.5 masterPriority: the masterPriorityVector for the port. The data type for masterPriority is UInteger224 (see 10.3.4). This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.6 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 10.3.16). The data type for currentLogAnnounceInterval is Integer8. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.7 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 both both the BMCA and the explicit port state configuration option.

10.3.9.8 announceInterval: a variable containing the mean Announce message transmission interval for the port. This value is set in the AnnounceIntervalSetting state machine (see 10.3.16). The data type for announceInterval is UScaledNs. This variable is used both both the BMCA and the explicit port state configuration option.

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

10.3.9.10 newInfo: a Boolean variable that is set to cause a port to transmit Announce information; specifically, it is set when an announce interval has elapsed (see Figure 10-17), port ~~role~~states have been updated, and portPriority and portStepsRemoved information has been updated with newly determined masterPriority and masterStepsRemoved information. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.11 portPriority: the portPriorityVector for the 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 configuration option.

10.3.9.12 portStepsRemoved: the value of stepsRemoved for the port. portStepsRemoved is set equal to masterStepsRemoved (see 10.3.8.3) after masterStepsRemoved is updated. The data type for portStepsRemoved is UInteger16. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.13 rcvdAnnouncePtr: a pointer to a structure that contains the fields of a received Announce message. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.14 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 configuration option.

10.3.9.15 updtInfo: a Boolean variable that is set to TRUE to indicate that the PortAnnounceInformation state machine (see 10.3.11) should copy the newly determined masterPriority and masterStepsRemoved to portPriority and portStepsRemoved, respectively. This variable is used both both the BMCA and the explicit port state configuration option; however, its value does not impact the explicit port state configuration option (see the NOTE in 10.3.15.3, just after Figure 10-17).

10.3.9.16 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 annL~~l~~eap61 is Boolean. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.17 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 annL~~l~~eap59 is Boolean. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.18 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 annC~~c~~urrentUtcOffsetValid is Boolean. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.19 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 both both the BMCA and the explicit port state configuration option.

10.3.9.20 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 annT~~t~~imeTraceable is Boolean. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.21 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 annF~~f~~requencyTraceable is Boolean. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.22 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 annC~~c~~urrentUtcOffset is Integer16. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.9.23 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 annT~~t~~imeSource is Enumeration8. This variable is used both both the BMCA and the explicit port state configuration option.

10.3.10 PortAnnounceReceive state machine

10.3.10.1 State machine variables

10.3.10.1.1 rcvdAnnounce: a Boolean variable that notifies the current state machine when Announce message information is received from the MD entity of the same port. This variable is reset by this state machine.

10.3.10.2 State machine functions

10.3.10.2.1 qualifyAnnounce (rcvdAnnouncePtr): qualifies the received Announce message pointed to by rcvdAnnouncePtr as follows:

a) If the Announce message was sent by the current time-aware system, i.e., if sourcePortIdentity.clockIdentity (see 10.6.2.2.11 and 8.5.2) is equal to thisClock (see 10.2.3.22), the Announce message is not qualified and FALSE is returned;

b) If the stepsRemoved field is greater than or equal to 255, the Announce message is not qualified and FALSE is returned;

c) If a path trace TLV is present and one of the elements of the pathSequence array field of the path trace TLV is equal to thisClock (i.e., the clockIdentity of the current time-aware system, see 10.2.3.22), the Announce message is not qualified and FALSE is returned;

d) Otherwise, the Announce message is qualified and TRUE is returned. If a path trace TLV is present and the port~~Role~~State of the port is SlavePort, the pathSequence array field of the TLV is copied to the global array pathTrace, and thisClock is appended to pathTrace (i.e., is added to the end of the array). If a path trace TLV is not present, the pathTrace array is set to the empty array (i.e., an array of zero elements). See 10.3.8.23 for a description of the path trace feature.

10.3.10.3 State diagram

The PortAnnounceReceive state machine shall implement the function specified by the state diagram in Figure 10-12, the local variables specified in 10.3.10.1, the functions specified in 10.3.10.2, and the relevant global variables specified in 10.2.3, 10.2.4, 10.3.9, and 11.2.12. The state machine is not used if externalPortConfiguration is ENABLED. The state machine receives Announce information from the MD entity of the same port, determines if the Announce message is qualified, and if so, sets the rcvdMsg variable.

[Editor's note: The fact that this state machine is not used when the external port state configuration option is used means that incoming Announce messages to the SlavePort are not qualified; rather, the information contained in the most recent Announce message is used to update the spanning tree connectivity information and GM time properties information. This is consistent with the latest draft of IEEE P1588 (July 28, 2016). Both stepsRemoved and path trace information are not checked (though these are maintained, also consistent with 1588 at least in the case of stepsRemoved).]

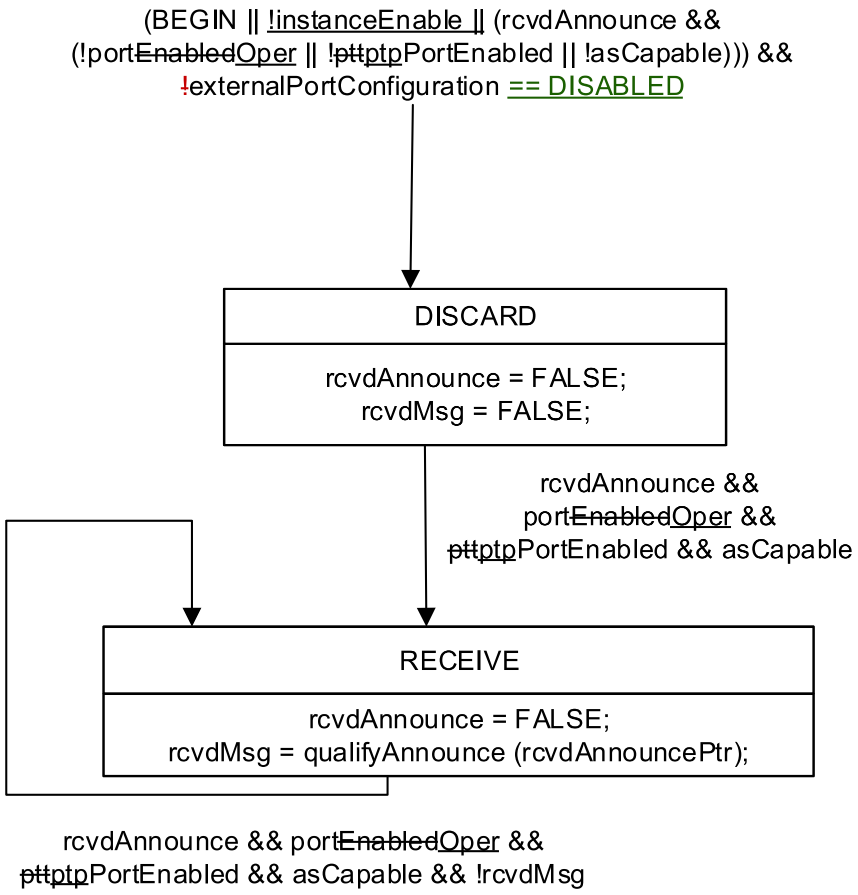

(BEGIN || !instanceEnable || (rcvdAnnounce &&
(!port~~Enabled~~Oper || !~~ptt~~ptpPortEnabled || !asCapable))) &&
~~!~~externalPortConfiguration == DISABLED
DISCARD
rcvdAnnounce = FALSE;
rcvdMsg = FALSE;
rcvdAnnounce &&
port~~Enabled~~Oper &&
~~ptt~~ptpPortEnabled && asCapable
RECEIVE
rcvdAnnounce = FALSE;
rcvdMsg = qualifyAnnounce (rcvdAnnouncePtr);
rcvdAnnounce && port~~Enabled~~Oper &&
~~ptt~~ptpPortEnabled && asCapable && !rcvdMsg

Figure 10-12—PortAnnounceReceive state machine

<<Editor's note: In Figure 10-12, portEnabled is changed to portOper, and pttPortEnabled is changed to ptpPortEnabled.>>

10.3.11 PortAnnounceInformation state machine

10.3.11.1 State machine variables

10.3.11.1.1 announceReceiptTimeoutTime: a variable used to save the time at which announce receipt timeout occurs. The data type for announceReceiptTimeoutTime is UScaledNs.

10.3.11.1.2 messagePriority: the messagePriorityVector corresponding to the received Announce information. The data type for messagePriority is UInteger224 (see 10.3.4).

10.3.11.1.3 rcvdInfo: an Enumeration2 that holds the value returned by rcvInfo() (see 10.3.11.2.1).

10.3.11.2 State machine functions

10.3.11.2.1 rcvInfo (rcvdAnnouncePtr): decodes the messagePriorityVector (see 10.3.4 and 10.3.5) and stepsRemoved 10.6.3.2.6) field from the Announce information pointed to by rcvdAnnouncePtr (see 10.3.9.13), and then does the following:

a) Stores the messagePriorityVector and stepsRemoved field value in messagePriority and messageStepsRemoved, respectively, and then
   3) ~~Returns SuperiorMasterInfo if the received message conveys the port role~~state~~ MasterPort, and the messagePriorityVector is superior to the portPriorityVector of the port,~~
   1) Returns RepeatedMasterInfo if the received message conveys the port ~~role~~state MasterPort, and the messagePriorityVector is the same as the portPriorityVector of the port, else
   2) Returns SuperiorMasterInfo if the received message conveys the port state MasterPort, and the messagePriorityVector is superior to the portPriorityVector of the port, else
   3) Returns InferiorMasterInfo if the received message conveys the port ~~role~~state MasterPort, and the messagePriorityVector is worse than the portPriorityVector of the port, else
   4) ~~Otherwise, r~~Returns OtherInfo.

NOTE—In accordance with 10.3.5, the message priority vector is superior to the portPriorityVector of the port if, and only if, the messagePriorityVector is better than the portPriorityVector, or the Announce message has been transmitted from the same master time-aware system and MasterPort as the portPriorityVector. In (1) - (4) above, rcvInfo() first checks whether the messagePriorityVector and portPriorityVector are the same (and he received message conveys the port state MasterPort), before checking whether the messagePriorityVector is superior to the portPriorityVector. This is because RepeatedMasterInfo needs to be returned if the messagePriorityVector and portPriorityVector are the same, while SuperiorMasterInfo needs to be returned in other instances where the Announce message has been transmitted from the same master time-aware system and MasterPort as the portPriorityVector (if the test for SuperiorMasterInfo were done before the test for RepeatedMasterInfo, SuperiorMasterInfo would be returned for the cases where RepeatedMasterInfo is desired).

10.3.11.2.2 recordOtherAnnounceInfo(): saves the flags leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, and frequencyTraceable, and the fields currentUtcOffset and timeSource, of the received Announce message for this port. The values are saved in the per-port global variables annLeap61, annLeap59, annCurrentUTCOffsetValid,~~ ~~annPtpTimescale, annTimeTraceable, annFrequencyTraceable, annCurrentUTCOffset, and annTimeSource (see 10.3.9.16 through 10.3.9.23).

10.3.11.3 State diagram

The PortAnnounceInformation state machine shall implement the function specified by the state diagram in Figure 10-13, the local variables specified in 10.3.11.1, the functions specified in 10.3.11.2, and the relevant global variables specified in 10.2.3, 10.2.4, 10.3.8, 10.3.9, and 11.2.12.~~ ~~This state machine is used only if externalPortConfiguration is DISABLED (if this variable is ENABLED, the PortAnnounceInformationExt state machine is used instead). The state machine receives new qualified Announce information from the PortAnnounceReceive state machine (see 10.3.10) of the same port and determines if the Announce information is better than the current best master information it knows about. The state machine also updates the current best master information when it receives updated port ~~role~~state information from the

Port~~Role~~StateSelection state machine (see 10.3.12), and when announce receipt timeout or, in the case where gmPresent is TRUE, sync receipt timeout occurs.

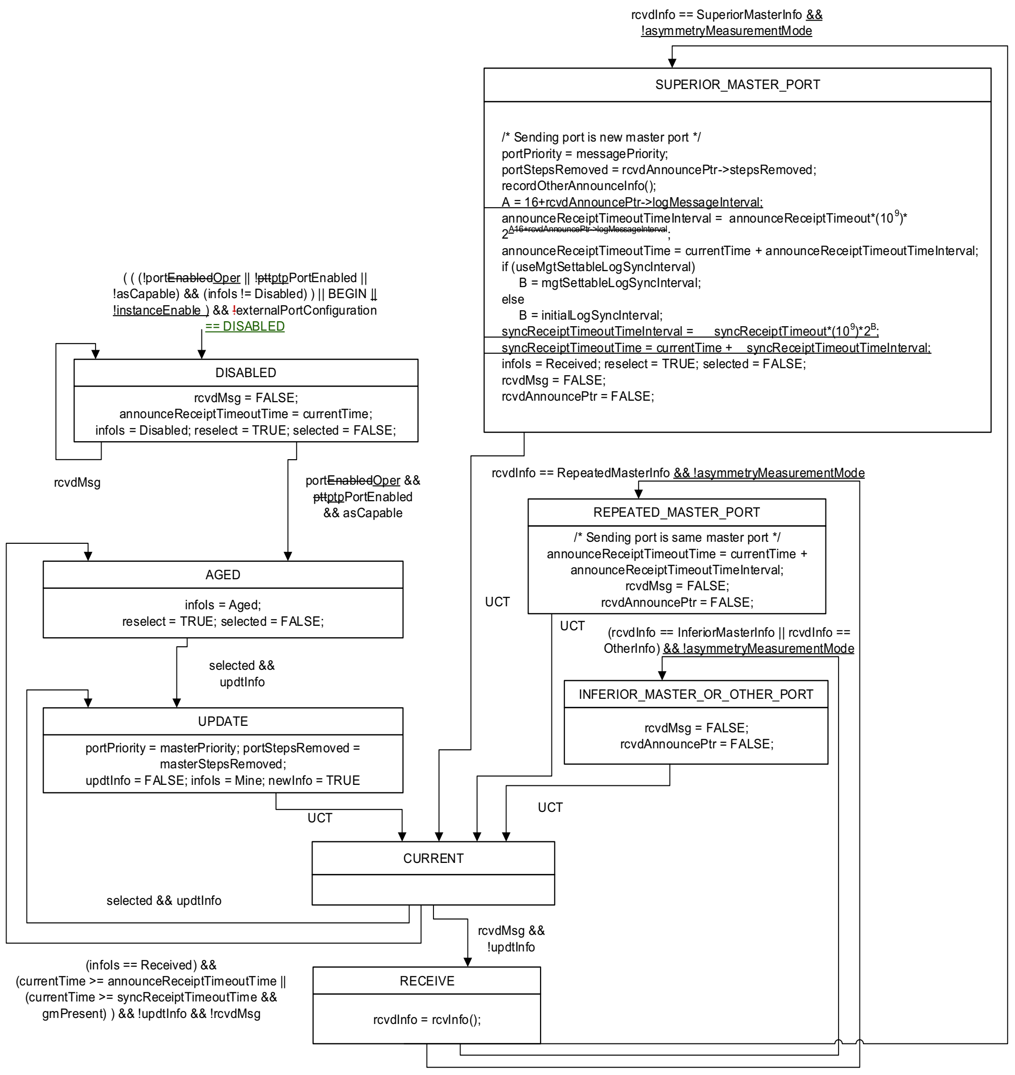

rcvdInfo == SuperiorMasterInfo && !asymmetryMeasurementMode
SUPERIOR_MASTER_PORT
/* Sending port is new master port */
portPriority = messagePriority;
portStepsRemoved = rcvdAnnouncePtr->stepsRemoved;
recordOtherAnnounceInfo();
A = 16+rcvdAnnouncePtr->logMessageInterval;
announceReceiptTimeoutTimeInterval = announceReceiptTimeout*(10^9)* 2^A~~16+rcvdAnnouncePtr->logMessageInterval~~;
announceReceiptTimeoutTime = currentTime + announceReceiptTimeoutTimeInterval;
if (useMgtSettableLogSyncInterval)
B = mgtSettableLogSyncInterval;
else
B = initialLogSyncInterval;
syncReceiptTimeoutTimeInterval = syncReceiptTimeout*(10^9)*2^B;
syncReceiptTimeoutTime = currentTime + syncReceiptTimeoutTimeInterval;
infoIs = Received; reselect = TRUE; selected = FALSE;
rcvdMsg = FALSE;
rcvdAnnouncePtr = FALSE;
(((!port~~Enabled~~Oper || !~~ptt~~ptpPortEnabled || !asCapable) && (infoIs != Disabled)) || BEGIN || !instanceEnable) && ~~!~~externalPortConfiguration == DISABLED
DISABLED
rcvdMsg = FALSE;
announceReceiptTimeoutTime = currentTime;
infoIs = Disabled; reselect = TRUE; selected = FALSE;
rcvdMsg
port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && asCapable
rcvdInfo == RepeatedMasterInfo && !asymmetryMeasurementMode
REPEATED_MASTER_PORT
/* Sending port is same master port */
announceReceiptTimeoutTime = currentTime + announceReceiptTimeoutTimeInterval;
rcvdMsg = FALSE;
rcvdAnnouncePtr = FALSE;
UCT
AGED
infoIs = Aged;
reselect = TRUE; selected = FALSE;
UCT
(rcvdInfo == InferiorMasterInfo || rcvdInfo == OtherInfo) && !asymmetryMeasurementMode
selected && updtInfo
INFERIOR_MASTER_OR_OTHER_PORT
rcvdMsg = FALSE;
rcvdAnnouncePtr = FALSE;
UPDATE
portPriority = masterPriority; portStepsRemoved = masterStepsRemoved;
updtInfo = FALSE; infoIs = Mine; newInfo = TRUE
UCT
UCT
CURRENT
selected && updtInfo
rcvdMsg && !updtInfo
(infoIs == Received) && (currentTime >= announceReceiptTimeoutTime || (currentTime >= syncReceiptTimeoutTime && gmPresent)) && !updtInfo && !rcvdMsg
RECEIVE
rcvdInfo = rcvInfo();

Figure 10-13—PortAnnounceInformation state machine

<<Editor's note: The change to Figure 10-13 is: (a) the introduction of the variable A in the SUPERIOR_MASTER_PORT state and the setting of A equal to the exponent of 2, and then the raising of 2 to this power, to make the figure more readable, (b) the addition of !asymmetryMeasurement Mode as a condition to transfer out ot the RECEIVE state, and (c) the replacing of portEnabled by portOper and pttPortEnabled by ptpPortEnabled.>>

10.3.12 Port~~Role~~StateSelection state machine

10.3.12.1 State machine variables

10.3.12.1.1 systemIdentityChange: a Boolean variable that notifies the current state machine when the systemIdentity (see 10.3.2) of this time-aware system has changed. This variable is reset by this state machine.

The systemIdentity changes when at least one of the attributes priority1, clockClass, clockAccuracy, offsetScaledLogVariance, and priority2 changes, e.g., due to management action, degradation or loss of the ClockSource, etc. The systemIdentity also includes the attribute clockIdentity, but this attribute does not change.

10.3.12.1.2 asymmetryMeasurementModeChange: a Boolean variable that notifies the current state machine when the per-port variable asymmetryMeasurementMode (see 10.2.4.2) changes on any port. This variable is reset by this state machine. There is one instance of asymmetryMeasurementModeChange for all the domains (per port). The variable is accessible by all the domains.

10.3.12.2 State machine functions

10.3.12.2.1 updt~~Role~~StateDisabledTree(): sets all the elements of the selected~~Role~~State array (see 10.2.3.20) to DisabledPort. Sets lastGmPriority to all ones. Sets the pathTrace array (see 10.3.8.23) to contain the single element thisClock (see 10.2.3.22).

10.3.12.2.2 clearReselectTree(): sets all the elements of the reselect array (see 10.3.8.1) to FALSE.

10.3.12.2.3 setSelectedTree(): sets all the elements of the selected array (see 10.3.8.2) to TRUE.

10.3.12.2.4 updt~~Role~~StatesTree(): performs the following operations (see 10.3.4 and 10.3.5 for details on the priority vectors):

a) Computes the gmPathPriorityVector for each 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 10.3.8.21) in lastGmPriority (see 10.3.8.22), computes the gmPriorityVector for the time-aware system and saves it in gmPriority, chosen as the best of the set consisting of the systemPriorityVector (for this time-aware system) and the gmPathPriorityVector for each port for which the clockIdentity of the master port is not equal to thisClock (see 10.2.3.22),
c) Sets the per-time-aware-system 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 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) Computes the masterPriorityVector for each port.
e) Computes masterStepsRemoved, which is equal to:
    3) messageStepsRemoved (see 10.3.9.9) for the port associated with the gmPriorityVector, incremented by 1, if the gmPriorityVector is not the systemPriorityVector, or
    4) 0 if the gmPriorityVector is the systemPriorityVector,

f) assigns the port ~~role~~state for port j and sets selected~~Role~~State[j] equal to this port ~~role~~state, as follows, for j = 1, 2, ..., numberPorts:
   5) If the port is disabled (infoIs == Disabled), selected~~Role~~State[j] is set to DisabledPort.
   6) If asymmetryMeasurementMode is TRUE, selectedState[j] is set to PassivePort and updtInfo is set to FALSE.
   7) If announce receipt timeout, or sync receipt timeout with gmPresent set to TRUE, have occurred (infoIs = Aged), updtInfo is set to TRUE and selected~~Role~~State[j] is set to MasterPort.
   8) If the portPriorityVector was derived from another port on the time-aware system or from the time-aware system itself as the root (infoIs == Mine), selected~~Role~~State[j] is set to MasterPort. In addition, updtInfo is set to TRUE if the portPriorityVector differs from the masterPriorityVector or portStepsRemoved differs from masterStepsRemoved.
   9) If the portPriorityVector was received in an Announce message and announce receipt timeout, or sync receipt timeout with gmPresent TRUE, have not occurred (infoIs == Received), and the gmPriorityVector is now derived from the portPriorityVector, selected~~Role~~State[j] is set to SlavePort and updtInfo is set to FALSE.
   10) If the portPriorityVector was received in an Announce message and announce receipt timeout, or sync receipt timeout with gmPresent TRUE, have 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 port on the time-aware system, selected~~Role~~State[j] is set to PassivePort and updtInfo is set to FALSE.
   11) If the portPriorityVector was received in an Announce message and announce receipt timeout, or sync receipt timeout with gmPresent TRUE, have 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 port on the time-aware system, selected~~Role~~State[j] set to PassivePort and updtInfo is set to FALSE.
   12) If the portPriorityVector was received in an Announce message and announce receipt timeout, or sync receipt timeout with gmPresent TRUE, have not occurred (infoIs == Received), the gmPriorityVector is not now derived from the portPriorityVector, and the masterPriorityVector is better than the portPriorityVector, selected~~Role~~State[j]is set to MasterPort and updtInfo is set to TRUE.

g) Updates gmPresent as follows:
   13) gmPresent is set to TRUE if the priority1 field of the rootSystemIdentity of the gmPriorityVector is less than 255.
   14) gmPresent is set to FALSE if the priority1 field of the rootSystemIdentity of the gmPriorityVector is equal to 255.

h) Assigns the port ~~role~~state for port 0 (see 8.5.2.3), and sets selected~~Role~~State[0], as follows:
   15) if selected~~Role~~State[j] is set to SlavePort for any port with portNumber j, j = 1, 2, ..., numberPorts, selected~~Role~~State[0] is set to PassivePort.
   16) if selected~~Role~~State[j] is *not* set to SlavePort for any port with portNumber j, j = 1, 2, ..., numberPorts, selected~~Role~~State[0] is set to SlavePort.

i) If the clockIentity member of the systemIdentity (see 10.3.2) member of gmPriority (see 10.3.8.21) is equal to thisClock (see 10.2.3.22), i.e., if the current time-aware system is the grandmaster, the pathTrace array is set to contain the single element thisClock (see 10.2.3.22).

10.3.12.3 State diagram

The Port~~Role~~StateSelection state machine shall implement the function specified by the state diagram in Figure 10-14, the functions specified in 10.3.12.1, and the relevant global variables specified in 10.2.3, 10.2.4, 10.3.8, 10.3.9, and 11.2.12. This state machine is used only if externalPortConfiguration is DISABLED (if this variable is ENABLED, the PortStateSettingExt state machine is used instead). The state machine updates the gmPathPriority vector for each port of the time-aware system, the gmPriorityVector for the time-aware system, and the masterPriorityVector for each port of the time-aware system. The state machine determines the port ~~role~~state for each port and updates gmPresent.

<<Editor's note: In Figure 10-14, the following changes have been made: (a) the initial "R" in

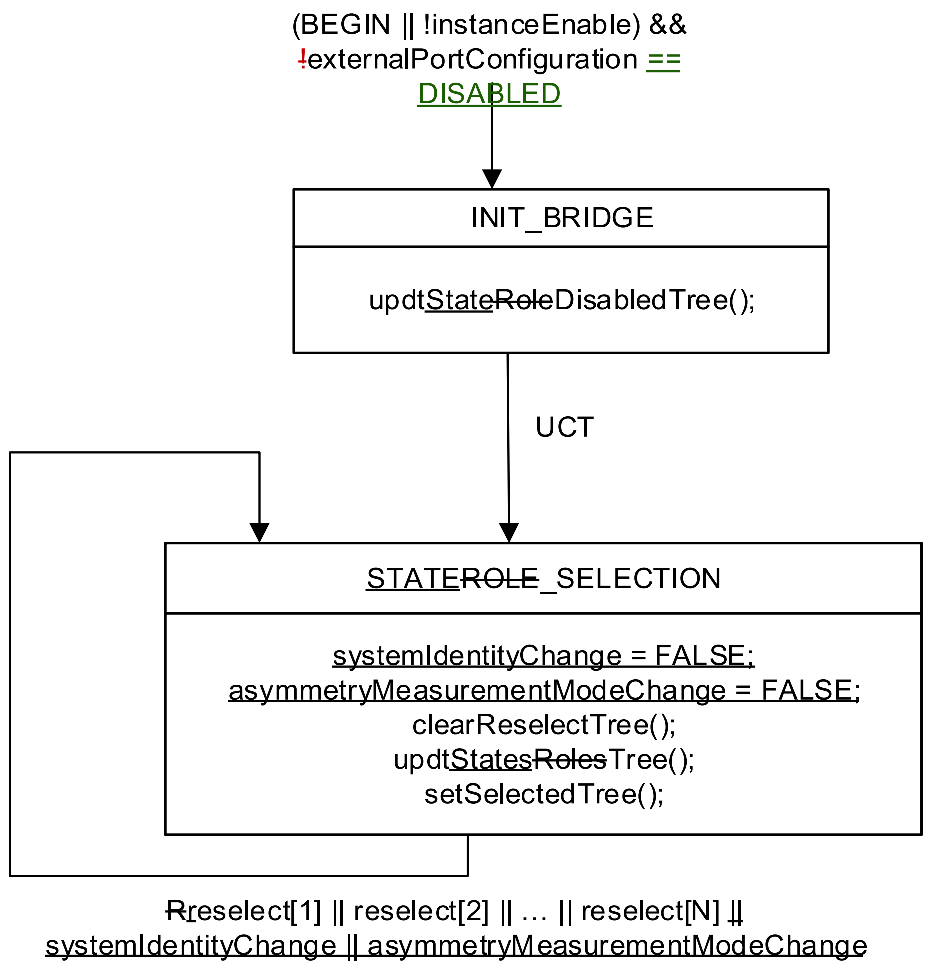

(BEGIN || !instanceEnable) && ~~!~~externalPortConfiguration == DISABLED
INIT_BRIDGE
updtState~~Role~~DisabledTree();
UCT
STATE~~ROLE~~_SELECTION
systemIdentityChange = FALSE;
asymmetryMeasurementModeChange = FALSE;
clearReselectTree();
updtStates~~Roles~~Tree();
setSelectedTree();
~~R~~reselect[1] || reselect[2] || … || reselect[N] || systemIdentityChange || asymmetryMeasurementModeChange

Figure 10-14—Port~~Role~~StateSelection state machine

Reselect[1] is changed to lower case "r", in the condition for the transition from STATE_SELECTION back to itself (the "R" was a typo), (b) the condition that the variable systemIdentityChange is TRUE is added to the transition from the state STATE_SELECTION back to itself, and (c) systemIdentityChange and asymmetryMeasurementModeChange are reset to FALSE in the STATE_SELECTION state. The effect of (b) and (c) is to cause updtStatesTree() to be invoked, i.e.,to invoke BMCA, whenever one of the time-aware system attributes priority1, clockClass, clock Accuracy, offsetScaledLogVariance, or priority2 changes, or if a port is put in or taken out of asymmetryMeasurement mode. Note that when (b) and (c) were discussed in the Novemeber, 2014 TSN TG meeting, it was indicated that an alternative fix to (b) and (c) would be to trigger updtStatesTree() at least one per Announce interval; this is analogous to the triggering of a state decision event at least once per Announce interval, as required by IEEE 1588-2008. It was also indicted that this could be added as a condition, in addition to (b) and (c). However, it was noted that this is problematic if the Announce interval is not the same on all ports, e.g., should the smallest Announce interval be used (IEEE 1588 does not indicate what to do regarding the timing of state

decision events in this case). Since the problem identified in the November, 2014 meeting is solved by simply triggering updtStatesTree() if an attribute changes, this solution was used.

The problem addressed by (b) and (c), namely the failure of a time-aware system to stop sending Sync and Follow_Up when it is no longer gmCapable if it was the only gmCapable system in the network was not officially added to the 802.1 maintenance database; should this be added?>>

10.3.13 PortAnnounceInformationExt state machine

10.3.13.1 State machine variables

10.3.13.1.1 **rcvdAnnounce:** a Boolean variable that notifies the current state machine when Announce message information is received from the MD entity of the same port. This variable is reset by this state machine.

10.3.13.1.2 **messagePriority:** the messagePriorityVector corresponding to the received Announce information. The data type for messagePriority is UInteger224 (see 10.3.4).

10.3.13.2 State machine functions

10.3.13.2.1 **rcvInfoExt (rcvdAnnouncePtr):** decodes the messagePriorityVector (see 10.3.4 and 10.3.5) and stepsRemoved 10.6.3.2.6) field from the Announce information pointed to by rcvdAnnouncePtr (see 10.3.9.13), and then stores the messagePriorityVector and stepsRemoved field value in messagePriority and messageStepsRemoved, respectively. If a path trace TLV is present in the Announce message and the portState of the port is SlavePort, the pathSequence array field of the TLV is copied to the global array pathTrace, and thisClock is appended to pathTrace (i.e., is added to the end of the array).

10.3.13.2.2 **recordOtherAnnounceInfo():** saves the flags leap61, leap59, currentUtcOffsetValid,ptpTimescale, timeTraceable, and frequencyTraceable, and the fields currentUtcOffset and timeSource, of the received Announce message for this port. The values are saved in the per-port global variables annLeap61, annLeap59, annCurrentUtcOffsetValid, annPtpTimescale, annTimeTraceable, annFrequencyTraceable, annCurrentUtcOffset, and annTimeSource (see 10.3.9.16 through 10.3.9.23).

10.3.13.3 State diagram

The PortAnnounceInformationExt state machine shall implement the function specified by the state diagram in Figure 10-15, the local variables specified in 10.3.13.1, the functions specified in 10.3.13.2, and the relevant global variables specified in 10.2.3, 10.2.4, 10.3.8, 10.3.9, and 11.2.12. This state machine is used only if externalPortConfiguration is ENABLED (if this variable is DISABLED, the

PortAnnounceInformation state machine of 10.2.11.3 is used instead). The state machine receives Announce information from the MD entity of the same port and saves the information.

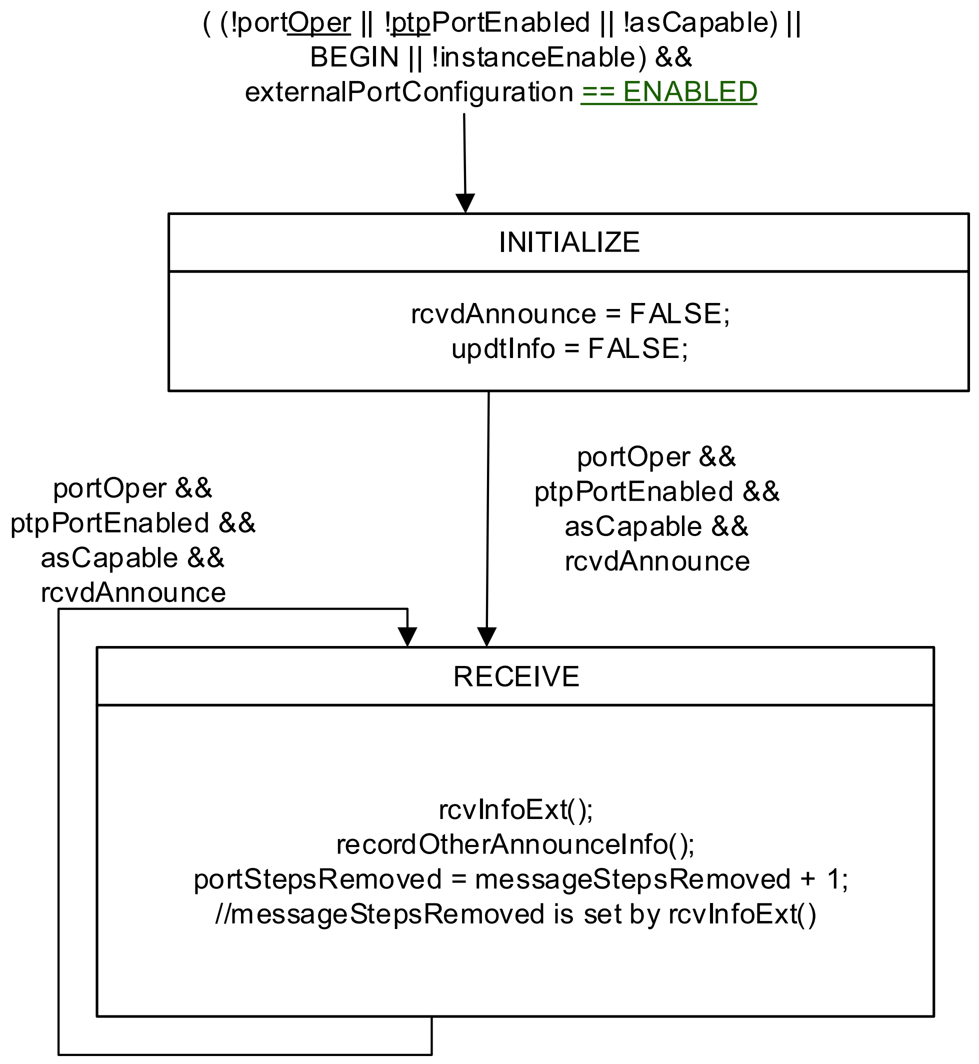

((!portOper || !ptpPortEnabled || !asCapable) || BEGIN || !instanceEnable) && externalPortConfiguration == ENABLED
INITIALIZE
rcvdAnnounce = FALSE;
updtInfo = FALSE;
portOper && ptpPortEnabled && asCapable && rcvdAnnounce
portOper && ptpPortEnabled && asCapable && rcvdAnnounce
RECEIVE
rcvInfoExt();
recordOtherAnnounceInfo();
portStepsRemoved = messageStepsRemoved + 1;
//messageStepsRemoved is set by rcvInfoExt()

Figure 10-15—PortAnnounceInformationExt state machine

10.3.14 PortStateSettingExt state machine

10.3.14.1 State machine variables

10.3.14.1.1 **disabledExt:** a Boolean variable that notifies the current state machine (i.e., is set to TRUE) when at least one of the variables portOper, ptpPortEnabled, or asCapable, for this port, has changed from TRUE to FALSE. This variable is reset by this state machine.

10.3.14.1.2 asymmetryMeasurementModeChangeThisPort: a Boolean variable that notifies the current state machine when the per-port variable asymmetryMeasurementMode (see 10.2.4.2) changes on this port. This variable is reset by this state machine. There is one instance of asymmetryMeasurementModeChangeThisPort for all the domains (per port). The variable is accessible by all the domains.

10.3.14.1.3 rcvdPortStateInd: a Boolean variable that notifies the current state machine (i.e., set to TRUE) when the port state of this port has been externally set. This variable is reset by this state machine.

10.3.14.1.4 portStateInd: an Enumeration2 that indicates the port state that has been set. The values are MasterPort, SlavePort, PassivePort.

NOTE—The port state can be externally set to DisabledPort by setting portOper or ptpPortEnabled to FALSE. The port state is set to DisabledPort when asCapable becomes FALSE.

10.3.14.2 State machine functions

10.3.14.2.1 resetStateDisabledTree(j): sets selectedState[j] (see 10.2.3.20) to DisabledPort. Sets the pathTrace array (see 10.3.8.23) to contain the single element thisClock (see 10.2.3.22) if no port of the time-aware system has the port state SlavePort.

10.3.14.2.2 updtPortState(j): performs the following operations for port j (see 10.3.4 and 10.3.5 for details on the priority vectors):

a) Sets the per-time-aware-system global variables leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource as follows:
   1) If the port state of any port of this time-aware system other than 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 port.
   2) If no port of this time-aware system other than port 0 has the 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) Computes masterStepsRemoved as follows:
   3) If the port state of any port of this time-aware system other than port 0 is SlavePort, then masterStepsRemoved is set equal to portStepsRemoved for that port.
   4) If no port of this time-aware system other than port 0 has the port state SlavePort, then masterStepsRemoved is set equal to 0.

c) Assigns the port state for port j and sets selectedState[j] equal to this port state, as follows:
   5) If disabledExt is TRUE, selectedState[j] is set to DisabledPort, else
   6) If asymmetryMeasurementMode is TRUE, selectedState[j] is set to PassivePort, else
   7) selectedState[j] is set to portStateInd.

d) Updates gmPresent as follows:
   8) If the port state of any port of this time-aware system other than port 0 is SlavePort and the priority1 field of the rootSystemIdentity of the messagePriority of the slave port is less than 255, gmPresent is set to TRUE, else
   9) If the port state of any port of this time-aware system other than port 0 is SlavePort and the priority1 field of the rootSystemIdentity of the messagePriority of the slave port is equal to 255, gmPresent is set to FALSE, else
   10) If no port of this time-aware system other than port 0 has the port state SlavePort, gmPresent is set to TRUE if priority1 for this time-aware system is less than 255 and FALSE if priority1 for this time-aware system is equal to 255.

e) Assigns the port state for port 0, and sets selectedState[0], as follows:
   11) if selectedState[j] is set to SlavePort, selectedState[0] is set to PassivePort.
   12) 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.
f) Computes the gmPriorityVector as follows:
   13) if selectedState[j] is set to SlavePort, the gmPriorityVector is set equal to messagePriority for port j.
   14) 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.
g) Computes the masterPriorityVector for port j.
h) If no port of this time-aware system has the port state SlavePort, the pathTrace array is set to contain the single element thisClock (see 10.2.3.22).

10.3.14.3 State diagram

The PortStateSettingExt state machine shall implement the function specified by the state diagram in Figure 10-16, the functions specified in 10.3.14.2, and the relevant global variables specified in 10.2.3, 10.2.4, 10.3.8, 10.3.9, and 11.2.12. This state machine is used only if externalPortConfiguration is ENABLED (if this variable is DISABLED, the PortStateSelection state machine of 10.3.12.3 is used instead). A separate instance of this state machine runs on each port (unlike the PortStateSelection state machine, for which a single instance runs in the time-aware system and performs operations on all the ports).

The state machine updates the gmPriorityVector for the time-aware system and the masterPriorityVector for each port of the time-aware system. The state machine determines the port state for each port and updates gmPresent.

NOTE—It is possible, using the external port state configuration mechanism, to misconfigure the network, e.g., to

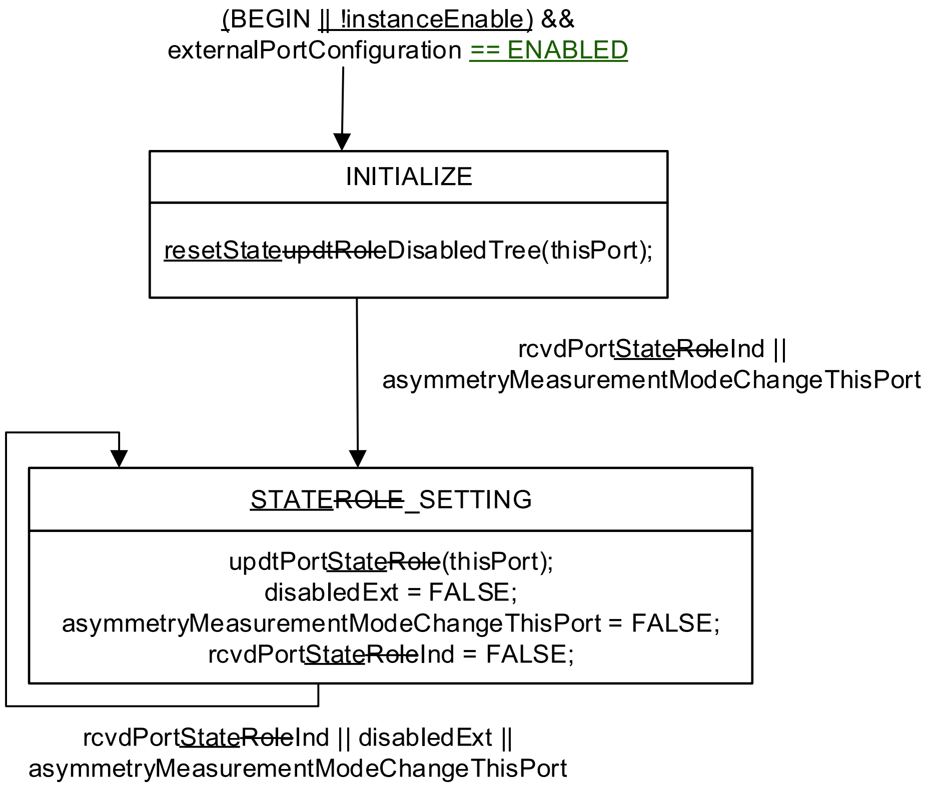

(BEGIN || !instanceEnable) && externalPortConfiguration == ENABLED
INITIALIZE
resetState~~updtRole~~DisabledTree(thisPort);
rcvdPortState~~Role~~Ind || asymmetryMeasurementModeChangeThisPort
STATE~~ROLE~~_SETTING
updtPortState~~Role~~(thisPort);
disabledExt = FALSE;
asymmetryMeasurementModeChangeThisPort = FALSE;
rcvdPortState~~Role~~Ind = FALSE;
rcvdPortState~~Role~~Ind || disabledExt || asymmetryMeasurementModeChangeThisPort

Figure 10-16—PortStateSettingExt state machine

produce a configuration where one or more time-aware systems has more than one slave port. Detecting and correcting misconfiguations is outside the scope of this standard.

10.3.15 PortAnnounceTransmit state machine

10.3.15.1 State machine variables

10.3.15.1.1 announceSendTime: the time, relative to the LocalClock, at which the next transmission of Announce information is to occur. The data type for announceSendTime is UScaledNs.

10.3.15.1.2 numberAnnounceTransmissions: a count of the number of consecutive Announce message transmissions after the AnnounceIntervalSetting state machine (see Figure 10-18) has set announceSlowdown (see 10.3.9.2) to TRUE. The data type for numberAnnounceTransmissions is UInteger8.

10.3.15.1.3 interval2: a local variable that holds either announceInterval or oldAnnounceInterval. The data type for interval2 is UScaledNs.

10.3.15.2 State machine functions

10.3.15.2.1 txAnnounce (): transmits Announce information to the MD entity of this port. The Announce information is set as follows:

a) The components of the messagePriorityVector are set to the values of the respective components of the masterPriorityVector of this port.
b) The grandmasterIdentity, grandmasterClockQuality, grandmasterPriority1, and grandmasterPriority2 fields of the Announce message are set equal to the corresponding components of the messagePriorityVector.
c) The value of the stepsRemoved field of the Announce message is set equal to masterStepsRemoved.
d) The Announce message flags leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, and frequencyTraceable, and the Announce message fields currentUtcOffset and timeSource, are set equal to the values of the global variables leap61, leap59, currentUtcOffsetValid, ptpTimescale, timeTraceable, frequencyTraceable, currentUtcOffset, and timeSource, respectively (see 10.3.8.4 through 10.3.8.11).
e) The sequenceId field of the Announce message is set in accordance with 10.5.7.
f) A path trace TLV (see 10.6.3.3) is constructed, with its pathSequence field (see 10.6.3.3.4) set equal to the pathTrace array (see 10.3.8.23). If appending the path trace TLV to the Announce message does not cause the media-dependent layer frame to exceed any respective maximum size, the path trace TLV is appended to the Announce message; otherwise, it is not appended. If the pathTrace array is empty, the path trace TLV is not appended. See 10.3.8.23 for a description of the path trace feature.

10.3.15.3 State diagram

The PortAnnounceTransmit state machine shall implement the function specified by the state diagram in Figure 10-17, the local variables specified in 10.3.15.1, the function specified in 10.3.15.2, and the relevant global variables specified in 10.2.3, 10.2.4, 10.3.9, and 11.2.12. The state machine transmits Announce information to the MD entity when an announce interval has elapsed, port ~~role~~states have been updated, and portPriority and portStepsRemoved information has been updated with newly determined masterPriority and masterStepsRemoved information.

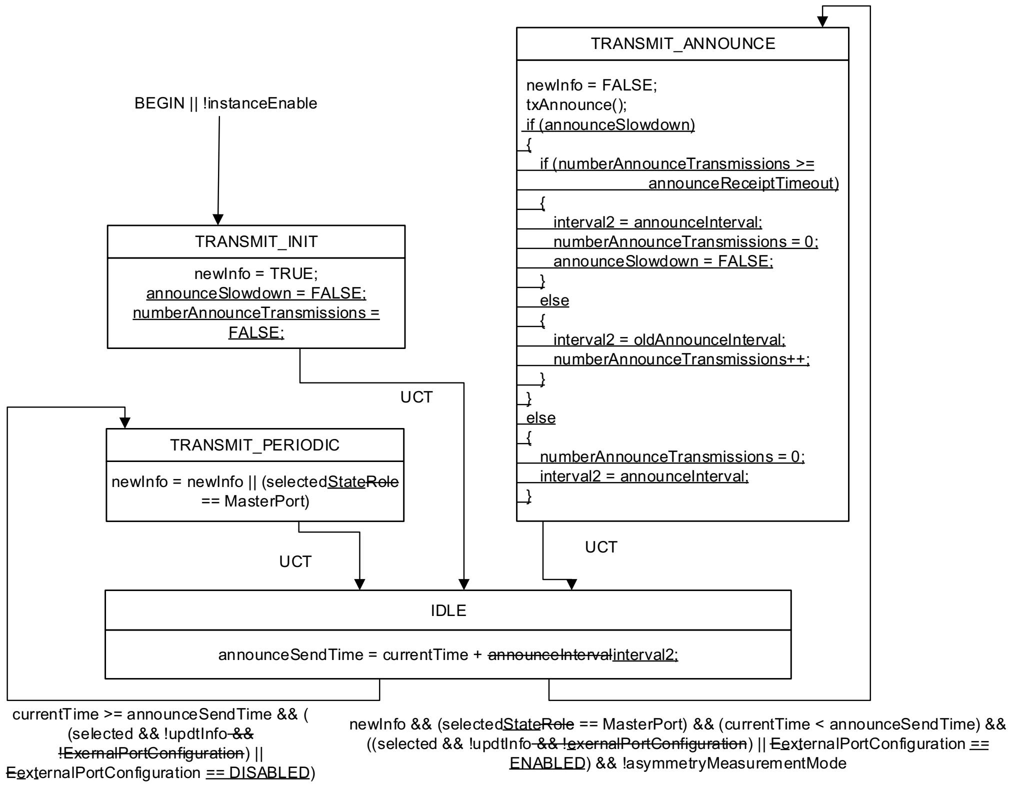

BEGIN || !instanceEnable
TRANSMIT_INIT
newInfo = TRUE;
announceSlowdown = FALSE;
numberAnnounceTransmissions = FALSE;
UCT
TRANSMIT_PERIODIC
newInfo = newInfo || (selected~~StateRole~~ == MasterPort)
UCT
TRANSMIT_ANNOUNCE
newInfo = FALSE;
txAnnounce();
if (announceSlowdown)
{
if (numberAnnounceTransmissions >= announceReceiptTimeout)
{
interval2 = announceInterval;
numberAnnounceTransmissions = 0;
announceSlowdown = FALSE;
}
else
{
interval2 = oldAnnounceInterval;
numberAnnounceTransmissions++;
}
}
else
{
numberAnnounceTransmissions = 0;
interval2 = announceInterval;
}
UCT
IDLE
announceSendTime = currentTime + ~~announceInterval~~interval2;
currentTime >= announceSendTime && ((selected && !updtInfo ~~&& !ExernalPortConfiguration~~) || ~~E~~externalPortConfiguration == DISABLED)
newInfo && (selected~~StateRole~~ == MasterPort) && (currentTime < announceSendTime) && ((selected && !updtInfo ~~&& !exernalPortConfiguration~~) || ~~E~~externalPortConfiguration == ENABLED) && !asymmetryMeasurementMode

Figure 10-17—PortAnnounceTransmit state machine

NOTE—When the external port state configuration option is used (i.e., externalPortConfiguration is ENABLED, see 10.3.8.24) the values of the variables updtInfo and selected do not affect the operation of the PortAnnounceTransmit state machine because the terms of the conditions in which they appear, i.e., (selected && !updtInfo) || externalPortConfiguartion == ENABLED, evaluate to TRUE when externalPortConfiguration is TRUE.

<<Editor's note: The change to Figure 10-17 is the addition of the condition !asymmetry MeasurementMode to the transition from IDLE to TRANSMIT_ANNOUNCE, and checks for whether or not externalPortConfiguration is ENABLED or DISABLED in the transitions from IDLE to TRANSMIT_PERIODIC or TRANSMIT_ANNOUNCE.>>

10.3.16 AnnounceIntervalSetting state machine

10.3.16.1 State machine variables

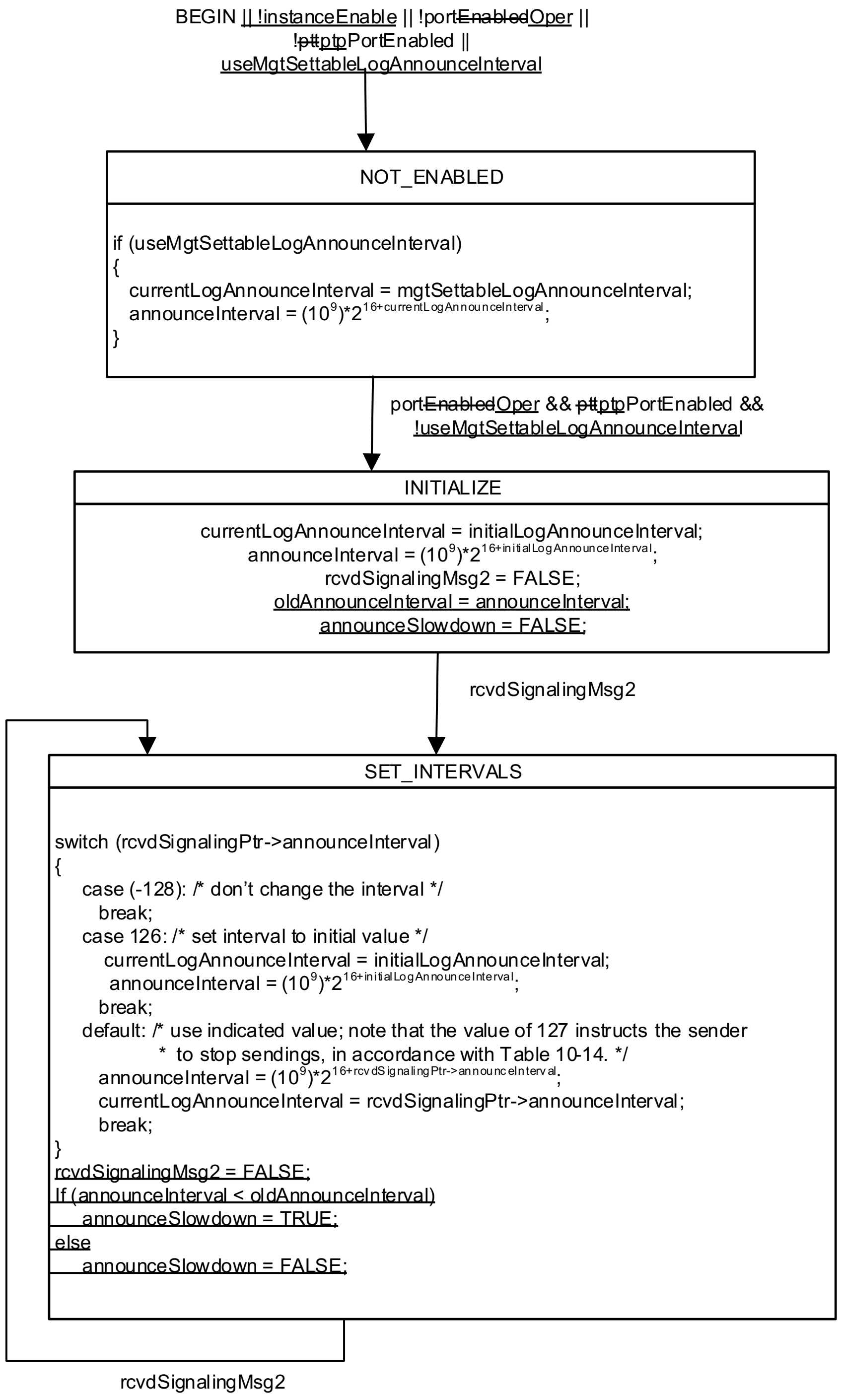
The following variables are used in the state diagram of 10.3.16.2:

10.3.16.1.1 rcvdSignalingMsg2: a Boolean variable that notifies the current state machine when a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3) is received. This variable is reset by the current state machine.

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

10.3.16.2 State diagram

The AnnounceIntervalSetting state machine shall implement the function specified by the state diagram in Figure 10-18, the local variables specified in 10.3.16.1, the messages specified in 10.6, the relevant global variables specified in 10.2.4, the relevant managed objects specified in 14.8, and the relevant timing attributes specified in 10.7. This state machine is responsible for setting the global variables that give the duration of the mean interval between successive Announce messages, both at initialization and in response to the receipt of a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3).


BEGIN || !instanceEnable || !port~~Enabled~~Oper || !~~ptt~~ptpPortEnabled || useMgtSettableLogAnnounceInterval
NOT_ENABLED
port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && !useMgtSettableLogAnnounceInterval
INITIALIZE
rcvdSignalingMsg2
SET_INTERVALS
rcvdSignalingMsg2

NOT_ENABLED:

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

INITIALIZE:

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

SET_INTERVALS:

```
switch (rcvdSignalingPtr->announceInterval)
{
    case (-128): /* don't change the interval */
      break;
    case 126: /* set interval to initial value */
       currentLogAnnounceInterval = initialLogAnnounceInterval;
        announceInterval = $(10^9)*2^{16+initialLogAnnounceInterval}$;
      break;
    default: /* use indicated value; note that the value of 127 instructs the sender
                 * to stop sendings, in accordance with Table 10-14. */
      announceInterval = $(10^9)*2^{16+rcvdSignalingPtr->announceInterval}$;
      currentLogAnnounceInterval = rcvdSignalingPtr->announceInterval;
      break;
}
rcvdSignalingMsg2 = FALSE;
If (announceInterval < oldAnnounceInterval)
    announceSlowdown = TRUE;
else
    announceSlowdown = FALSE;
```

Figure 10-18—AnnounceIntervalSetting state machine

<<Editor's note: In Figure 10-16, portEnabled is changed to portOper and pttPortEnabled is changed to ptpPortEnabled.>>

10.4 State machines related to signaling gPTP protocol capability

10.4.1 gPtpCapableTransmit state machine

10.4.1.1 State machine variables

The following variables are used in the state diagram of 10.4.1.3:

10.4.1.1.1 signalingMsgTimeInterval: a variable whose value is the mean time, in s, between successive Signaling messages that carry the gPTP capable TLV (see 10.7.2.5 and 10.6.4.4). The data type for signalingMsgTimeInterval is UScaledNs.

10.4.1.1.2 intervalTimer: a variable used to save the time at which the gPTP capable TLV interval timer is set (see Figure 10-19). A Signaling message containing a gPTP capable TLV is sent when this timer expires. The data type for intervalTimer is UScaledNs.

10.4.1.1.3 txSignalingMsgPtr: a pointer to a structure whose members contain the values of the fields of a Signaling message to be transmitted, which contains a gPTP capable TLV (see 10.6.4.4).

10.4.1.2 State machine functions

10.4.1.2.1 setGptpCapableTlv(): creates a structure containing the parameters of a Signaling message that contains a gPTP capable TLV, to be transmitted (see 10.6.4), and returns a pointer to this structure. The parameters are set as follows:

a) logGptpCapableMessageInterval is set to the value of the managed object logGptpCapableMessageInterval (see 14.8.29), and
b) remaining parameters are set as specified in 10.6.4 and 10.6.4.4.

10.4.1.3 State diagram

The gPtpCapableTransmit state machine shall implement the function specified by the state diagram in Figure 10-19, the local variables specified in 10.4.1.1, the functions specified in 10.4.1.2, the relevant parameters specified in 10.6.4 and 10.6.4.4, and the relevant timing attributes specified in 10.7. This state machine is responsible for setting the parameters of each Signaling message that contains the gPTP capable TLV, and causing these Signaling messages to be transmitted at a regular rate.

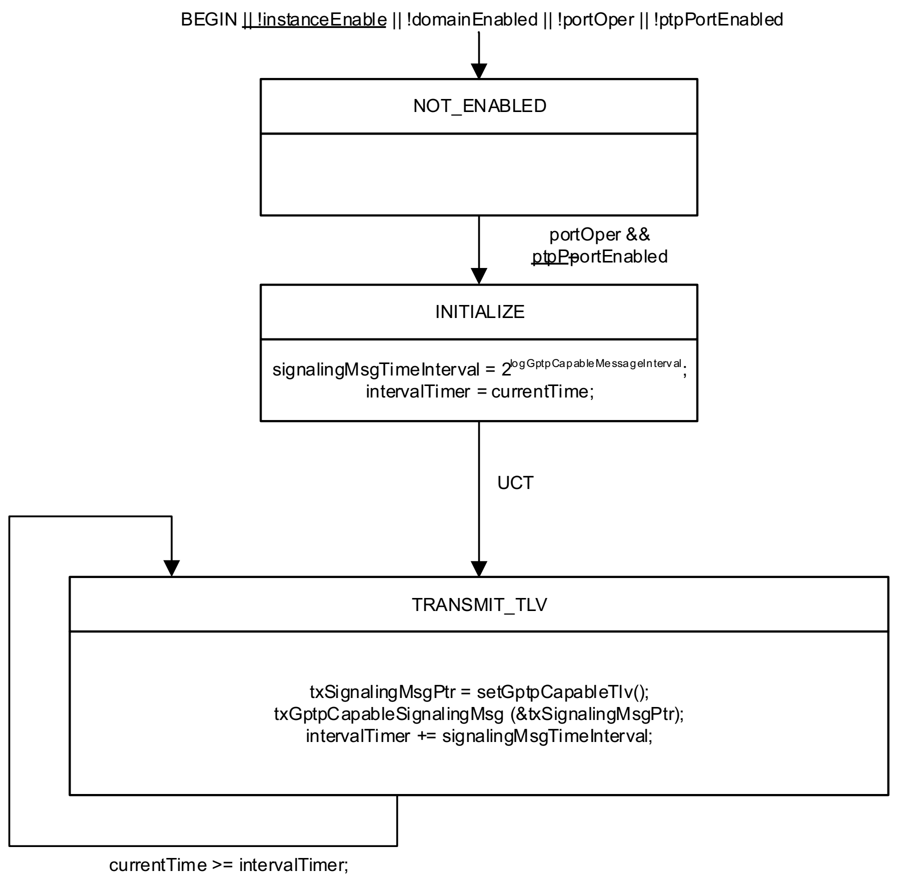

BEGIN || !instanceEnable || !domainEnabled || !portOper || !ptpPortEnabled
NOT_ENABLED
portOper && ptpPportEnabled
INITIALIZE
signalingMsgTimeInterval = 2^logGptpCapableMessageInterval;
intervalTimer = currentTime;
UCT
TRANSMIT_TLV
txSignalingMsgPtr = setGptpCapableTlv();
txGptpCapableSignalingMsg (&txSignalingMsgPtr);
intervalTimer += signalingMsgTimeInterval;
currentTime >= intervalTimer;

Figure 10-19—gPtpCapableTransmit state machine

10.4.2 gPtpCapableReceive state machine

10.4.2.1 State machine variables

The following variables are used in the state diagram of 10.4.1.3:

10.4.2.1.1 rcvdGptpCapableTlv: a Boolean variable that notifies the current state machine when a Signaling message containing a gPTP capable TLV is received. This variable is reset by the current state machine.

10.4.2.1.2 rcvdSignalingMsgPtr: a pointer to a structure whose members contain the values of the fields of a Signaling message whose receipt is indicated by rcvdGptpCapableTlv (see 10.4.2.1.2).

10.4.2.1.3 gPtpCapableReceiptTimeoutInterval: the time interval after which, if a Signaling message containing a gPTP capable TLV is not received, the neighbor of this port is considered to no longer be invoking the gPTP protocol. The data type for gPtpCapableReceiptTimeoutInterval is UScaledNs.

10.4.2.1.4 timeoutTime: a variable used to save the time at which the neighbor of this port is considered to no longer be invoking the gPTP protocol if a Signaling message containing a gPTP capable TLV is not received. The data type for timeoutTime is UScaledNs.

10.4.2.2 State diagram

The gPtpCapableTransmit state machine shall implement the function specified by the state diagram in Figure 10-20, the local variables specified in 10.4.2.1, the relevant parameters specified in 10.6.4 and 10.6.4.4, and the relevant timing attributes specified in 10.7. This state machine is responsible for setting the parameters of each Signaling message that contains the gPTP capable TLV, and causing these Signaling messages to be transmitted at a regular rate.

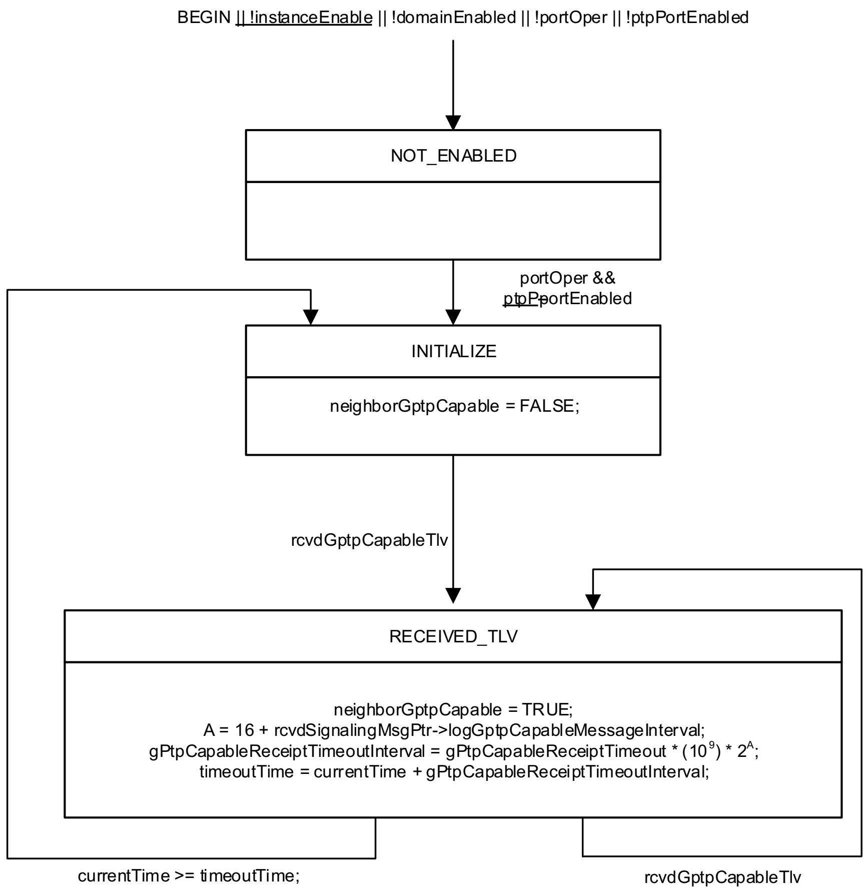

BEGIN || !instanceEnable || !domainEnabled || !portOper || !ptpPortEnabled
NOT_ENABLED
portOper &&
ptpP~~p~~ortEnabled
INITIALIZE
neighborGptpCapable = FALSE;
rcvdGptpCapableTlv
RECEIVED_TLV
neighborGptpCapable = TRUE;
A = 16 + rcvdSignalingMsgPtr->logGptpCapableMessageInterval;
gPtpCapableReceiptTimeoutInterval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

Figure 10-20—gPtpCapableReceive state machine

10.5 Message attributes

10.5.1 General

This subclause describes media-independent attributes of the Announce message and the Signaling message that are not described in 8.4.2, and whose descriptions are not generic to all messages used in this standard. This subclause also describes media-independent attributes of all time-synchronization event messages.

10.5.2 Message class

The Announce message is a general message, i.e., it is not timestamped. An Announce message provides status and characterization information of the time-aware system that transmitted the message and the grandmaster. This information is used by the receiving time-aware system when executing the BMCA.

The Signaling message is a general message, i.e., it is not timestamped. A Signaling message carries information, requests, and/or commands between time-aware systems, via one or more TLVs.

NOTE—In this standard, the Signaling message is used by a port of a time-aware system to request that the port at the other end of the link send time-synchronization event messages, link delay measurement messages, or Announce messages at desired intervals,~~ and~~ to indicate whether the port at the other end of the link should compute neighborRateRatio and/or neighborPropDelay, and to indicate whether a port can receive and correctly process one-step Syncs.. ~~A single TLV, termed t~~The *message interval request TLV,* is defined to carry this information (see 10.6.4.3). One usage of this functionality is to allow a time-aware system in power-saving mode to remain connected to a gPTP domain via the port on which the Signaling message is sent.

10.5.3 Addresses

The destination address of the Announce and Signaling messages shall be the reserved multicast address given in Table 10-2 unless otherwise specified in a media-dependent clause (see 16.2 and 12.2).

Table 10-2—Destination address for Announce and Signaling messages

Destination address
01-80-C2-00-00-0E
NOTE—This address is taken from Table 8-1,Table 8-2, and Table 8-3 of IEEE Std 802.1Q-20144~~1~~.

NOTE—Frames whose destination address is the address of Table 10-2 are never forwarded, according to IEEE 802.1Q~~-2005~~. Use of this address is shared by IEEE 802.1AS and other IEEE 802.1 protocols.

10.5.4 Ethertype

The Ethertype of the Announce and Signaling messages shall be the Ethertype given in Table 10-3.

Table 10-3—Ethertype for Announce and Signaling messages

Ethertype
$88F7_{16}$

NOTE—This Ethertype is used for all PTP messages.

10.5.5 Subtype

The subtype of the Announce and Signaling messages is indicated by the ~~transportSpecific~~majorSdoId field (see 10.6.2.2.1).

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

10.5.6 Source port identity

The Announce message, Signaling message, and all time-synchronization ~~event~~ messages contain a sourcePortIdentity field, see 10.6.2.2.11, ~~that~~which identifies the_egress port (see 8.5) on which the respective message is sent.

10.5.7 Sequence number

Each PortSync entity of a time-aware system maintains a separate sequenceId pool for each of the message types Announce and Signaling, respectively, transmitted by the MD entity of the port.

Each Announce and Signaling message contains a sequenceId field, see 10.6.2.2.12, that carries the message sequence number. The sequenceId of an Announce message shall be one greater than the sequenceId of the previous Announce message sent by the transmitting port, subject to the constraints of the rollover of the UInteger16 data type used for the sequenceId field. The sequenceId of a Signaling message shall be one greater than the sequenceId of the previous Signaling message sent by the transmitting port, subject to the constraints of the rollover of the UInteger16 data type used for the sequenceId field.

10.6 Message formats

10.6.1 General

The PTP messages Announce and Signaling each have a header, body, and, if present, a suffix that contains one or more TLVs (see 10.6.2, 10.6.3, 10.6.4, and Clause 14 of IEEE Std 1588-2008). Reserved fields shall be transmitted with all bits of the field 0 and ignored by the receiver, unless otherwise specified. The data type of the field shall be the type indicated in brackets in the title of each subclause.

This subclause defines the path trace TLV, which is carried by the Announce message (see 10.6.3.2.8), and the message interval request TLV, which is carried by the Signaling message (see 10.6.4.3). If a time-aware system cannot parse a TLV, it shall ignore it and attempt to parse the next TLV (see 14.1 of IEEE Std 1588-2008).

NOTE—Any overhead specific to the respective medium is added to each message.

10.6.2 Header

10.6.2.1 General header specifications

The common header for all PTP messages shall be as specified in Table 10-4 and 10.6.2.2 and its subclauses.

Table 10-4—PTP message header

Bits		Octets	Offset
~~8~~7 ~~7~~6 ~~6~~5 ~~5~~4	~~4~~3 ~~3~~2 ~~2~~1 ~~1~~0		
~~transportSpecific~~majorSdoId	messageType	1	0
minorVersionPTP~~reserved~~	versionPTP	1	1
messageLength		2	2
domainNumber		1	4
~~reserved~~minorSdoId		1	5
flags		2	6
correctionField		8	8
~~reserved~~messageTypeSpecific		4	16
sourcePortIdentity		10	20
sequenceId		2	30
control		1	32
logMessageInterval		1	33

10.6.2.2 Header field specifications

10.6.2.2.1 ~~transportSpecific~~majorSdoId (Nibble)

The value is ~~0x1~~specified in 8.1 for all transmitted PTP messages of a gPTP domain. The value is specified in 11.2.15 for all transmitted PTP messages of the Common Mean Link Delay Service. Any PTP message received for which the value is not one of the values specified in those subclauses shall be ~~not 0x1 is~~ ignored.

10.6.2.2.2 messageType (Enumeration4)

The value indicates the type of the message, as defined in Table 10-5.

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

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

Table 10-5—Values for messageType field

Message type	Message class	Value
Announce	General	0xB
Signaling	General	0xC
NOTE—Values for the messageType field for PTP other messages that are used only for specific media are defined in the respective media-dependent clause(s).		

10.6.2.2.3 minorVersionPTP (UInteger4)

[Editor's note: The references to IEEE Std 1588 in this subclause and the next one are to the latest draft of IEEE 1588-Rev. It is expected that this will be published prior to sponsor ballot of 802.1AS-Rev.]

For transmitted messages, the value shall be 1 (see 7.5.5 and 13.3.2.5 of IEEE Std 1588). For received messages, the value is ignored.

NOTE—minorVersionPTP indicates the minor version number of IEEE 1588 PTP used in the PTP profile contained in this standard for information only.

10.6.2.2.4 versionPTP (UInteger4)

For transmitted messages, t~~T~~he value ~~is~~shall be 2 (see 7.5.5 and 13.3.2.4 of IEEE Std 1588). For received messages, if the value is not 2, the entire message shall be ignored.

NOTE—VersionPTP indicates the version number of IEEE 1588 PTP used in the PTP profile contained in this standard.

10.6.2.2.5 messageLength (UInteger16)

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

NOTE—For example, the Follow_Up message (see) contains a PTP header (34 octets), preciseOriginTimestamp (10 octets), and Follow_Up information TLV (32 octets). The value of the messageLength field is 34+10+32 = 76.

10.6.2.2.6 domainNumber (UInteger8)

The value is the gPTP domain number specified in 8.1.

10.6.2.2.7 minorSdoId (UInteger8)

The value is specified in 8.1 for all transmitted PTP messages of a gPTP domain. The value is specified in 11.2.15 for all transmitted PTP messages of the Common Mean Link Delay Service. Any PTP message received for which the value is not one of the values specified in those subclauses shall be ignored.

10.6.2.2.8 Flags (Octet2)

The value of the bits of the array are defined in Table 10-6. For message types where the bit is not defined in Table 10-6, the value of the bit is set to FALSE.

10.6.2.2.9 correctionField (Integer64)

The value is 0.

10.6.2.2.10 messageTypeSpecific

<<Editor’s Note: The text below that describes messageTypeSpecific is copied from IEEE P1588_V_022 (July 28, 2016) with minor editing.>>

The value of the messageTypeSpecific field varies, based on the value of the messageType field, as described in Table 10-7.

For Event messages only, the four octets of the messageTypeSpecific field may be used for internal implementation of a PTP clock and its ports. For example, if the clock consists of multiple hardware components that are not synchronized, messageTypeSpecific can be used to transfer an internal timestamp between components (e.g. a physical layer chip and the clock's processor).

The messageTypeSpecific field is not used for features of this standard, and it has no meaning from one clock to another. In the on-the-wire format at each PTP port, for all messageType values, the messageTypeSpecific field is transmitted with all bits of the field 0, and ignored on receive.

Table 10-6—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, reserved as FALSE and ignored on reception
0	1	Sync, Pdelay_Resp	twoStepFlag	*For Sync messages:* a) For a one-step transmitting port (see 11.1.3 and 11.2.12.9), the value is FALSE. b) For a two-step transmitting 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, reserved as FALSE and ignored on reception
0	3	All	Reserved	Not used by IEEE Std 1588 -2008, reserved as FALSE and ignored on reception
0	4	All	Reserved	Not used by IEEE Std 1588 -2008, reserved as FALSE and ignored on reception
0	5	All	PTP profileSpecific 1	Not used in this standard, reserved as FALSE and ignored on reception

Table 10-6—Values of flag bits

Octet	Bit	Message Types	Name	Value
0	6	All	PTP profileSpecific 2	Not used in this standard, reserved as FALSE and ignored on reception
0	7	All	Reserved	Not used in this standard, reserved as FALSE and ignored on reception
1	~~1~~0	Announce	leap61	The value of the global variable leap61 (see 10.3.8.4)
1	~~2~~1	Announce	leap59	The value of the global variable leap59 (see 10.3.8.5)
1	~~3~~2	Announce	currentUtcOffsetValid	The value of the global variable currentUtcOffsetValid (see 10.3.8.6)
1	~~4~~3	~~All~~Announce	ptpTimescale	~~Reserved as TRUE, ignored on reception~~The value of the global variable ptpTimescale (see 10.3.8.7)
1	~~5~~4	Announce	timeTraceable	The value of the global variable timeTraceable (see 10.3.8.8)
1	~~6~~5	Announce	frequencyTraceable	The value of the global variable frequencyTraceable (see 10.3.8.9)
1	6	All	Reserved	Not used by IEEE Std 1588 - 2008, reserved as FALSE and ignored on reception
1	7	All	Reserved	Not used by IEEE Std 1588 - 2008, reserved as FALSE and ignored on reception

Table 10-7 — 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_Resp	For the Event message class, this field may be used for internal implementation as specified in this subclause.

10.6.2.2.11 sourcePortIdentity (PortIdentity)

The value is the port identity attribute, see 8.5.2, of the port that transmits the PTP message.

10.6.2.2.12 sequenceId (UInteger16)

The sequenceId field is assigned as specified in 10.5.7.

10.6.2.2.13 control (UInteger8)

The value is 0x5.

10.6.2.2.14 logMessageInterval (Integer8)

For an Announce message, the value is the value of currentLogAnnounceInterval, see 10.3.9.6, for the port that transmits the Announce message. For a Signaling message, the value is ~~reserved~~transmitted as 0x7F and ignored on reception.

10.6.3 Announce message

10.6.3.1 General Announce message specifications

The fields of the body of the Announce message shall be as specified in Table 10-8 and 10.6.3.2 and its subclauses.

Table 10-8—Announce message fields

Bits								Octets	Offset
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
header (see 10.6.2)								34	0
reserved								10	34
currentUtcOffset								2	44
reserved								1	46
grandmasterPriority1								1	47
grandmasterClockQuality								4	48
grandmasterPriority2								1	52
grandmasterIdentity								8	53
stepsRemoved								2	61
timeSource								1	63
path trace TLV								4+8N	64

10.6.3.2 Announce message field specifications

10.6.3.2.1 currentUtcOffset (Integer16)

The value is the value of currentUtcOffset (see 10.3.8.10) for the time-aware system that transmits the Announce message.

10.6.3.2.2 grandmasterPriority1 (UInteger8)

The value is the value of the priority1 component of the rootSystemIdentity of the gmPriorityVector (see 10.3.5) of the time-aware system that transmits the Announce message.

10.6.3.2.3 grandmasterClockQuality (ClockQuality)

The value is the clockQuality formed by the clockClass, clockAccuracy, and offsetScaledLogVariance of the rootSystemIdentity of the gmPriorityVector (see 10.3.5) of the time-aware system that transmits the Announce message.

10.6.3.2.4 grandmasterPriority2 (UInteger8)

The value is the value of the priority2 component of the rootSystemIdentity of the gmPriorityVector (see 10.3.5) of the time-aware system that transmits the Announce message.

10.6.3.2.5 grandmasterIdentity (ClockIdentity)

The value is the value of the clockIdentity component of the rootSystemIdentity of the gmPriorityVector (see 10.3.5) of the time-aware system that transmits the Announce message.

10.6.3.2.6 stepsRemoved (UInteger16)

The value is the value of masterStepsRemoved (see 10.3.8.3) for the time-aware system that transmits the Announce message.

10.6.3.2.7 timeSource (Enumeration8)

The value is the value of timeSource (see 10.3.8.11) for the time-aware system that transmits the Announce message.

10.6.3.2.8 Path trace TLV

The Announce message carries the path trace TLV, defined in 10.6.3.3.

10.6.3.3 Path trace TLV definition

10.6.3.3.1 General

The fields of the path-trace TLV shall be as specified in Table 10-9 and in 10.6.4.3.2 through 10.6.4.3.9. This TLV, and its use, are defined in IEEE Std 1588-2008 (see 16.2 and Table 34 of IEEE Std 1588-2008).

Table 10-9—Path trace TLV

Bits								Octets	Offset from start of TLV
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
tlvType								2	0
lengthField								2	2
pathSequence								8N	4

10.6.3.3.2 tlvType (Enumeration16)

The value of the tlvType field is 0x8.

NOTE—This is the value that indicates the TLV is a path trace TLV, as specified in 16.2.7.1 and Table 34 of IEEE Std 1588-2008. The value is specified there as PATH_TRACE, whose value is 0x8.

10.6.3.3.3 lengthField (UInteger16)

The value of the lengthField is 8N.

10.6.3.3.4 pathSequence (ClockIdentity[N])

The value of pathSequence is a ClockIdentity array. The array elements are the clockIdentities of the successive time-aware systems that receive and send an Announce message. The quantity N is the number of time-aware systems, including the grandmaster, that the Announce information has traversed.

NOTE—N is equal to stepsRemoved+1 (see 10.6.3.2.6). The size of the pathSequence array increases by 1 for each time-aware system that the Announce information traverses.

10.6.4 Signaling message

10.6.4.1 General Signaling message specifications

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

Table 10-10—Signaling message fields

Bits								Octets	Offset
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
header (see 10.6.2)								34	0
targetPortIdentity								10	34
message interval request TLV or gPTP capable TLV								16	44

10.6.4.2 Signaling message field specifications

10.6.4.2.1 targetPortIdentity (PortIdentity)

The value is 0xFF.

10.6.4.2.2 Message interval request TLV or gPTP capable TLV

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.

10.6.4.3 Message interval request TLV definition

10.6.4.3.1 General

The fields of the message interval request TLV shall be as specified in Table 10-11 and in 10.6.4.3.2 through 10.6.4.3.9. This TLV is a standard organization extension TLV for the Signaling message, as specified in 14.3 of IEEE Std 1588-2008.

Table 10-11—Message interval request TLV

Bits	Octets	Offset from start of TLV
~~8~~7 ~~7~~6 ~~6~~5 ~~5~~4 ~~4~~3 ~~3~~2 ~~2~~1 ~~1~~0		
tlvType	2	0
lengthField	2	2
organizationId	3	4
organizationSubType	3	7
linkDelayInterval	1	10
timeSyncInterval	1	11
announceInterval	1	12
flags	1	13
reserved	2	14

10.6.4.3.2 tlvType (Enumeration16)

The value of the tlvType field is 0x3.

NOTE—This is the value that indicates the TLV is a vendor and standard organization extension TLV, as specified in 14.3.2.1 of IEEE Std 1588-2008. The value is specified there as ORGANIZATION_EXTENSION, whose value is 0x3.

10.6.4.3.3 lengthField (UInteger16)

The value of the lengthField is 12.

10.6.4.3.4 organizationId (Octet3)

The value of organizationId is 00-80-C2.

10.6.4.3.5 organizationSubType (Enumeration24)

The value of organizationSubType is 2.

10.6.4.3.6 linkDelayInterval (Integer8)

The value is the logarithm to base 2 of the mean time interval, desired by the port that sends this TLV, between successive Pdelay_Req messages sent by the port at the other end of the link. The format and allowed values of linkDelayInterval are the same as the format and allowed values of initialLogPdelayReqInterval, see 11.5.2.2.

The values 127, 126, and –128 are interpreted as defined in Table 10-12.

Table 10-12—Interpretation of special values of linkDelayInterval

Value	Instruction to time-aware system that receives this TLV
127	Instructs the port that receives this TLV to stop sending link delay measurement messages.
126	Instructs the port that receives this TLV to set currentLogPdelayReqInterval to the value of initialLogPdelayReqInterval, see 11.5.2.2.
–128	Instructs the port that receives this TLV not to change the mean time interval between successive Pdelay_Req messages.

10.6.4.3.7 timeSyncInterval (Integer8)

The value is the logarithm to base 2 of the mean time interval, desired by the port that sends this TLV, between successive time-synchronization event messages sent by the port at the other end of the link. The format and allowed values of timeSyncInterval are the same as the format and allowed values of initialLogSyncInterval, see 10.7.2.3, 11.5.2.3, 12.7, and 13.9.2.

The values 127, 126, and –128 are interpreted as defined in Table 10-13.

Table 10-13—Interpretation of special values of timeSyncInterval

Value	Instruction to time-aware system that receives this TLV
127	Instructs the port that receives this TLV to stop sending time-synchronization event messages.
126	Instructs the port that receives this TLV to set currentLogSyncInterval to the value of initialLogSyncInterval, see 10.7.2.3, 11.5.2.3, 12.7, and 13.9.2.
–128	Instructs the port that receives this TLV not to change the mean time interval between successive time-synchronization event messages.

When a signaling message that contains this TLV is sent by a port, the value of syncReceiptTimeoutTimeInterval for that port (see 10.2.4.3) shall be set equal to syncReceiptTimeout (see 10.7.3.1) multiplied by the value of the interval, in seconds, reflected by timeSyncInterval.

10.6.4.3.8 announceInterval (Integer8)

The value is the logarithm to base 2 of the mean time interval, desired by the port that sends this TLV, between successive Announce messages sent by the port at the other end of the link. The format and allowed values of announceInterval are the same as the format and allowed values of initialLogAnnounceInterval, see 10.7.2.2.

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

Table 10-14—Interpretation of special values of announceInterval

Value	Instruction to time-aware system that receives this TLV
127	Instructs the port that receives this TLV to stop sending Announce messages.
126	Instructs the port that receives this TLV to set currentLogAnnounceInterval to the value of initialLogAnnounceInterval, see 10.7.2.2.
–128	Instructs the port that receives this TLV not to change the mean time interval between successive Announce messages.

When a signaling message that contains this TLV is sent by a port, the value of announceReceiptTimeoutTimeInterval for that port (see 10.3.9.1) shall be set equal to announceReceiptTimeout (see 10.7.3.2) multiplied by the value of the interval, in seconds, reflected by announceInterval.

10.6.4.3.9 flags (Octet)

Bits ~~1~~0 through 2~~and 2~~ of the octet are defined in Table 10-15 and take on the values TRUE and FALSE.

Table 10-15—Definitions of bits of flags field of message interval request TLV

Bit	Name
~~1~~0	computeNeighborRateRatio
~~2~~1	computeNeighborPropDelay
2	oneStepReceiveCapable

<<Editor's note: The bit numbering has been changed from 1 - 3 to 0 - 2, consistent with 6.4.4.2.>>

Bits not defined in Table 10-15 are set to FALSE and ignored on receipt.

NOTE—For full-duplex, point-to-point links (see Clause 11), it is expected that the port sending this TLV will set ~~one or both of these bits~~ bits 1 and/or 2 to FALSE if this port will not provide valid timing information in its subsequent responses (Pdelay_Resp and Pdelay_Resp_Follow_Up) to Pdelay_Req messages. Similarly, it is expected that the port sending this TLV will set bit 3 to TRUE if it is capable of receiving and correctly processing one-step Sync messages.

10.6.4.4 gPTP capable TLV

10.6.4.4.1 General

The fields of the gPTP capable TLV shall be as specified in Table 10-16 and in 10.6.4.4.2 through 10.6.4.4.5. This TLV is a standard organization extension TLV for the Signaling message, as specified in 14.3 of IEEE Std 1588-2008.

Table 10-16—gPTP capable TLV

Bits								Octets	Offset from start of TLV
7	6	5	4	3	2	1	0		
tlvType								2	0
lengthField								2	2
organizationId								3	4
organizationSubType								3	7
logGptpCapableMessageInterval								1	10
flags								1	11
reserved								4	12

10.6.4.4.2 tlvType (Enumeration16)

The value of the tlvType field is 0x3.

NOTE—This is the value that indicates the TLV is a vendor and standard organization extension TLV, as specified in 14.3.2.1 of IEEE Std 1588-2008. The value is specified there as ORGANIZATION_EXTENSION, whose value is 0x3.

10.6.4.4.3 lengthField (UInteger16)

The value of the lengthField is 12.

10.6.4.4.4 organizationId (Octet3)

The value of organizationId is 00-80-C2.

10.6.4.4.5 organizationSubType (Enumeration24)

The value of organizationSubType is 4.

<<Editor’s note: The organizationId is the same as that of other TLVs defined in IEEE 802.1AS-2011. The organizationSubType value is the next value available.>

10.6.4.4.6 logGptpCapableMessageInterval

The value of logGptpCapableMessageInterval is the logarithm to base 2 of the mean gPTP capable TLV interval in seconds (see 10.7.2.1 and 10.7.2.5)

10.6.4.4.7 Flags

The flag bits shall be transmitted as FALSE and ignored on receipt.

<<Editor’s note: The Flags and reserved field are included in case they are needed in the future. They are not needed now. If the TSN TG believes there will never be a need for these fields, they could be omitted; however, it seems that including them allows for possible future additional uses for this TLV. The reserved field length is chosen as 4 octets only so that this TLV will have the same lenght as the message interval request TLV; however, the length could be different if desired. Comments are requested on this.>>

10.7 Protocol timing characterization

10.7.1 General

This subclause specifies timing and timeout attributes for the media-independent sublayer state machines.

10.7.2 Message transmission intervals

10.7.2.1 General interval specification

The mean time interval between the sending of successive Announce messages, referred to as the *announce interval,* shall be as specified in 10.7.2.2.

The mean time interval between the sending of successive time-synchronization event messages for full-duplex point-to-point, IEEE 802.11, and CSN links, and successive general messages containing time-synchronization information for IEEE 802.3 EPON links, is referred to as the sync interval. The sync interval shall be as specified in 10.7.2.3.

The mean time interval between the sending of successive Signaling messages that contain the gPTP capable TLV, referred to as the *gPTP capable TLV interval*, shall be as specified in 10.7.2.5.

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, t~~T~~he initialLogAnnounceInterval specifies the announce interval when the port is initialized and the value the announce interval is set to when a message interval request TLV is received with the announceInterval field set to 126 (see the AnnounceIntervalSetting state machine, 10.3.16). The currentLogAnnounceInterval specifies the current value of the announce interval. The default value of initialLogAnnounceInterval is 0. Every port supports the value 127; the port does not send Announce messages when currentLogAnnounceInterval has this value (see 10.3.16). A port may support other values, except for the reserved values –128 through –125, inclusive, and 124 through 126, inclusive. A port ignores requests (see 10.3.16) for unsupported values. The initialLogAnnounceIntervaland currentLogAnnounceInterval are per-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 s, between message transmissions is within ±30% of $2^{\text{currentLogAnnounceInterval}}$. In addition, a gPTP 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 $2^{\text{portDS.logAnnounceInterval}}$, in order to prevent causing an announceReceiptTimeout event. The PortAnnounceTransmit state machine (see 10.3.15) 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 gPTP 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 [B20].

NOTE ~~1~~2—If useMgtSettableLogAnnounceInterval is FALSE, t~~T~~he value of initialLogAnnounceInterval is the value of the mean time interval between successive Announce messages when the port is initialized. The value of the mean time interval between successive Announce messages ~~may~~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 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 announceInterval is 126, see 10.6.4.3.8.

NOTE ~~2~~3—A port that requests (using a Signaling message that contains a message interval request TLV, see 10.6.4 and 10.3.16) that the 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.

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, t~~T~~he initialLogSyncInterval specifies the sync interval when the port is initialized and the value the sync interval is set to when a message interval request TLV is received with the timeSyncInterval field set to 126 (see the ~~LinkDelay~~SyncIntervalSetting state machine, 11.2.19). The default value is media-dependent; the value is specified in the respective media-dependent clauses. The initialLogSyncInterval is a per-port attribute.

The currentLogSyncInterval specifies the current value of the sync interval, and is a per-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 s, between message transmissions is within ±30% of $2^{\text{currentLogSyncInterval}}$. In addition, a gPTP 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 $2^{\text{portDS.logSyncInterval}}$, in order to prevent causing a syncReceiptTimeout event. The PortSyncSyncSend state machine (see 10.2.11) 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 gPTP 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 [B20]

NOTE 2—If useMgtSettableLogSyncInterval is FALSE t~~T~~he value of initialLogSyncInterval is the value of the sync interval when the port is initialized. The value of the sync interval ~~may~~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 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 timeSyncInterval is 126, see 10.6.4.3.7.

10.7.2.4 Interval for providing synchronization information by ClockMaster entity

The clockMasterLogSyncInterval specifies the mean time interval between successive instants at which the ClockMaster entity provides time-synchronization information to the SiteSync entity. The value is less than or equal to the smallest currentLogSyncInterval (see 10.7.2.3) value for all the ports of the time-aware system. The clockMasterLogSyncInterval is an internal, per-time-aware-system variable.

10.7.2.5 Interval for sending the gPTP capable TLV Signaling message

The logarithm to the base 2 of the gPTP capable interval (in seconds) is carried in the logGptpCapableMessageInterval field of the gPTP capable TLV. The default value shall be TBD. The range shall be TBD.

<<Editor's note: Contributions are requested on the default value and range.>>

10.7.3 Timeouts

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 BMC algorithm needs to be run, if appropriate. The condition of the slave port not receiving synchronization information for syncReceiptTimeout sync intervals is referred to as *sync receipt timeout.*

The default value shall be 3. The syncReceiptTimeout is a per-port attribute.

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 BMC algorithm needs to be run, if appropriate. The condition of the slave port not receiving an Announce message for announceReceiptTimeout announce intervals is referred to as *announce receipt timeout.*

The default value shall be 3. The announceReceiptTimeout is a per-port attribute.

10.7.3.3 gPtpCapableReceiptTimeout

The value of this attribute tells a port the number of gPTP capable TLV 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 the gPTP protocol.

NOTE—A determination that its neighbor is no longer invoking the gPTP protocol will cause the port to set asCapable to FALSE.

The default value shall be TBD. The range shall be TBD.

<<Editor’s note: Contributions are requested on the default value and range.>>

<<Editor’s note: The value 9 has been discussed, but more thought and discussion on this is needed. More thought is also needed on the periodicity of refresh and the garbage collection time.>>

10.8 Media-independent performance requirements

The time-aware system clock performance complies with the requirements specified in B.1. The time-aware system performance should follow the recommendations of B.2.2 and B.2.3. The time-aware system performance complies with the requirements specified in B.2.4.

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

11.1 Overview

11.1.1 General

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

11.1.2 Propagation delay measurement

The measurement of propagation delay on a full-duplex, point-to-point link using the peer delay mechanism is illustrated in Figure 11-1. The mechanism is the same as the peer delay mechanism described in IEEE Std 1588-2008, specialized to a two-step clock[16] and sending the requestReceiptTimestamp and the responseOriginTimestamp separately [see 11.4.3 of IEEE Std 1588-2008, item (c)(8)]. The measurement is made by each port at the end of every full-duplex, point-to-point link. Thus, both ports sharing a link will independently make the measurement, and both ports will know the propagation delay as a result. This allows the time-synchronization information described in 11.1.3 to be transported irrespective of the direction taken by a Sync message. The propagation delay measurement is made on ports otherwise blocked by non-PTP algorithms (e.g., Rapid Spanning Tree Protocol) used to eliminate cyclic topologies. This enables either no loss of synchronization or faster resynchronization, after a reconfiguration, because propagation delays are already known and do not have to be initially measured when the reconfiguration occurs.

Since, as will be explained shortly, the propagation delay measurement is made using timestamps relative to the LocalClock entities at each port at the ends of the link and the resulting mean delay is expressed in the responder timebase (see 11.2.17.3.4), there is no need to measure the mean delay for the link in each domain, because the mean delay is the same in each domain. In addition, the quantity neighborRateRatio (see 10.2.4.7) is the ratio of the responder to requestor LocalClock frequency, and is also the same in all domains. Therefore, the propagation delay and neighborRateRatio measurements are domain-independent. Single instances of the respective state machines that cause these measurements to be made are invoked, rather than one instance per domain, and the results are available to all domains. The PTP messages used for the measurements (i.e., Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up, see 11.4.5 through 11.4.7) carry 0 in the domainNumber field, but this value is not used.

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

[16]See 3.1.47 of IEEE Std 1588-2008 for the definition of a *two-step clock*.

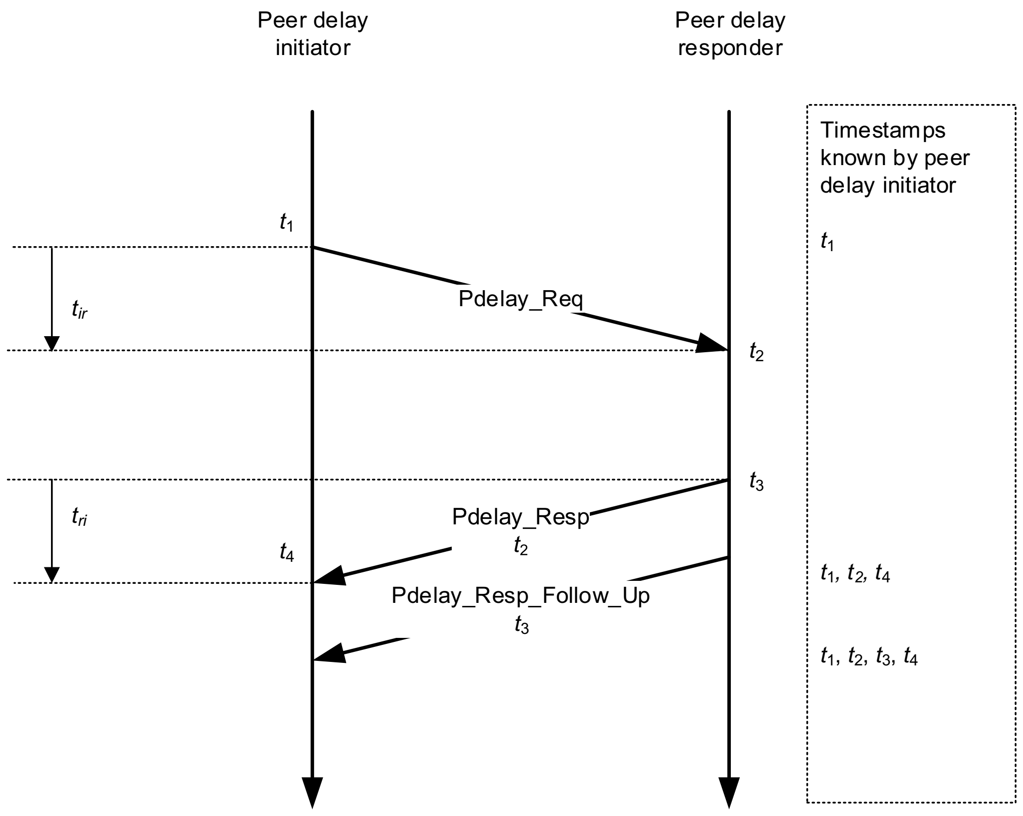


Figure 11-1—Propagation delay measurement using peer delay mechanism

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

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

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

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

The accuracy of the mean propagation delay measurement depends on how accurately the times t_1, t_2, t_3, and t_4 are measured. In addition, Equation (11-1) assumes that the initiator and responder timestamps are taken relative to clocks that have the same frequency. In practice, t_1 and t_4 are measured relative to the LocalClock entity of the initiator time-aware system, and t_2 and t_3 are measured relative to the LocalClock entity of the responder time-aware system. If the propagation delay measurement is desired relative to the responder time base, the term $(t_4 – t_1)$ in Equation (11-1) must be multiplied by the rate ratio of the responder relative to the initiator, otherwise there will be an error equal to $0.5y(t_4 - t_1)$, where y is the frequency offset of the responder relative to the initiator. Likewise, if the propagation delay measurement is desired relative to the initiator time base, the term $(t_3 – t_2)$ in Equation (11-1) must be multiplied by the rate ratio of the initiator relative to the responder, otherwise there will be an error equal to $0.5y(t_3 - t_2)$, where y is the frequency offset of the initiator relative to the responder. Finally, if the propagation delay measurement is desired relative to the grandmaster time base, each term must be multiplied by the rate ratio of the grandmaster relative to the time base that term is expressed in.

There can also be an error in measured propagation delay due to time measurement granularity (see B.1.2). For example, if the time measurement granularity is 40 ns (as specified in B.1.2), the timestamps t_1, t_2, t_3, and/or *t*4 can undergo 40 ns step changes. When this occurs, the measured propagation delay, *D*, will change by 20 ns (or by a multiple of 20 ns if more than one of the timestamps has undergone a 40 ns step change). The actual propagation delay has not changed by 20 ns; the effect is due to time measurement granularity. The effect can be reduced, and the accuracy improved, by averaging successive measured propagation delay values. For example, an exponential averaging filter can be used, i.e., as shown in Equation (11-2):

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

where D_k is the k^{th} propagation delay measurement, $D_{avg,k}$ is the k^{th} computed average propagation delay, and k is an index for the propagation delay measurements (i.e., peer delay message exchange). The quantity a is the exponential weighting factor; it can be set so that the weight of a past propagation delay measurement is $1/e$ after M measurements, i.e., as shown in Equation (11-3):

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

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

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

The rate ratio of the responder relative to the initiator is the quantity neighborRateRatio (see 10.2.4.7). It is computed by the function computePdelayRateRatio() (see 11.2.17.3.3) of the MDPdelayReq state machine (see 11.2.15) using successive values of t_3 and t_4. As indicated in the description of computePdelayRateRatio(), any scheme that uses this information is acceptable as long as the performance requirements of B.2.4 are met. One example scheme is given in NOTE 1 of 11.2.17.3.3.

11.1.3 Transport of time-synchronization information

The transport of time-synchronization information by a time-aware system, using Sync and Follow_Up (or just Sync) messages, is illustrated in Figure 11-2. The mechanism is mathematically equivalent to the mechanism described in IEEE Std 1588-2008 for a one-step or two-step,[16] peer-to-peer transparent clock that is syntonized (see 11.4.5.1, 11.5.1, and 11.5.2.2 of IEEE Std 1588-2008). However, as will be seen shortly, the processes of transporting synchronization by a ~~two-step,~~ peer-to-peer transparent clock that is syntonized and by a ~~two-step~~ boundary clock are mathematically and functionally equivalent.[17] The main functional difference between the two types of clocks is that the boundary clock participates in best master selection and invokes the BMCA, while the peer-to-peer transparent clock does not participate in best master selection and does not invoke the BMCA (and implementations of the two types of clocks can be different).

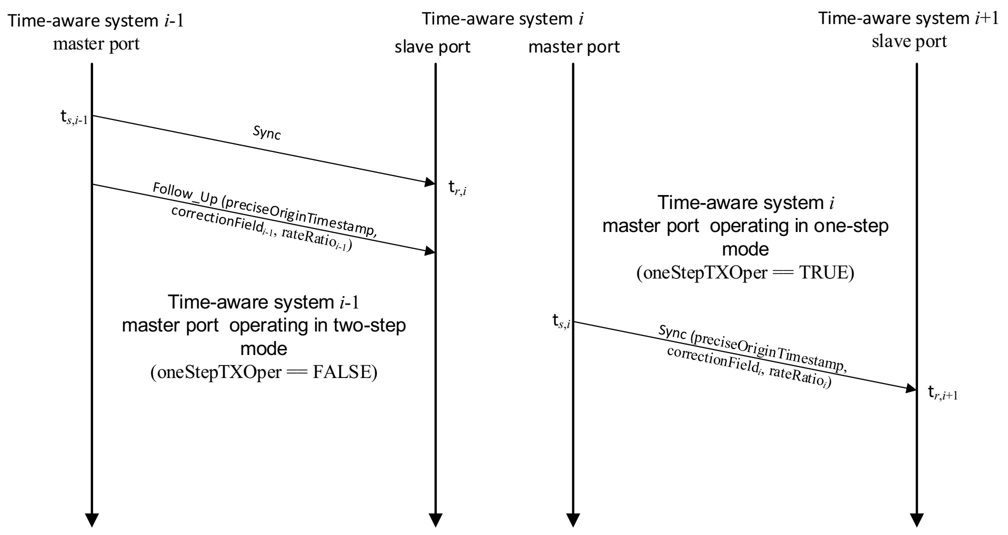


Figure 11-2—Transport of time-synchronization information

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

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

[17]The same mathematical and functional equivalence exists for one-step boundary and syntonized peer-to-peer transparent clocks. One-step clocks are not discussed here because time-aware systems described in this standard are two-step devices from the standpoint of IEEE Std 1588-2008.

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

a) The propagation delay on the link between time-aware systems i–1 and i, expressed in the grandmaster time base, and
b) The difference between $t_{s,i}$ and $t_{r,i}$ (i.e., the residence time), expressed in the grandmaster time base.

The mean propagation delay on the link between time-aware systems i–1 and i, relative to the LocalClock entity of time-aware system i–1, is equal to neighborPropDelay (see 10.2.4.8). This must be multiplied by rateRatio$_{i-1}$ to express it in the grandmaster time base. The total propagation delay is equal to the mean propagation delay plus the quantity delayAsymmetry (see 8.3 and 10.2.4.9); delayAsymmetry is already expressed in the grandmaster time base. The residence time, $t_{s,i}$–$t_{r,i}$, must be multiplied by rateRatio$_i$ to express it in the grandmaster time base.

The preceding computation is organized slightly differently in the state machines of 11.2.13 and 11.2.14. Rather than explicitly expressing the link propagation delay in the grandmaster time base, the local time at time-aware system i that corresponds to $t_{s,i-1}$ is computed; this is the upstreamTxTime member of the MDSyncReceive structure (see 10.2.2.2.7; recall that $t_{s,i-1}$ is relative to the LocalClock entity of time-aware system i–1). upstreamTxTime is equal to the quantity $t_{r,i}$ minus the link propagation delay expressed relative to the LocalClock entity of time-aware system i. The link propagation delay expressed relative to the LocalClock entity of time-aware system i is equal to the sum of the following:

c) The quantity neighborPropDelay (see 10.2.4.8) divided by neighborRateRatio (see 10.2.4.7), and
d) The quantity delayAsymmetry (see 10.2.4.9) divided by rateRatio$_i$.

The division of delayAsymmetry by rateRatio$_i$ is performed after rateRatio$_i$ has been updated, as described shortly. The computation of upstreamTxTime is done by the MDSyncReceiveSM state machine in the function setMDSyncReceive() (see 11.2.13.2.1). When time-aware system i sends a Sync message to time-aware system i+1, it computes the sum of the link propagation delay and residence time, expressed in the grandmaster time base, as:

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

As in item d) above, this computation is performed after rateRatio$_i$ has been updated, as described shortly. The quantity of item e) is added to correctionField$_{i-1}$ to obtain correctionField$_i$. The computation of item e) and correctionField$_i$ is done by the MDSyncSendSM state machine in the function setFollowUp() (see 11.2.14.2.3). The quantity correctionField$_i$ is inserted in the Follow_Up message sent by time-aware system i.

~~Note that t~~The difference between mean propagation delay relative to the grandmaster time base and relative to the time bases of the time-aware system at the other end of the attached link or of the current time-aware system is usually negligible. T~~o see this, note that t~~he former can be obtained from the latter by multiplying the latter by the ratio of the grandmaster frequency to the frequency of the LocalClock entity of the time-aware system at the other end of the 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 link, relative to the grandmaster, of 200 ppm, and a measured propagation time of 100 ns, the difference in D relative to the two time bases is 20 ps. The corresponding difference for link delay asymmetry in this example is also negligible because the magnitude of the link delay asymmetry is of the same order of magnitude as the mean propagation time, or less. However, the difference is usually not negligible for residence time, because residence time can be much larger (see B.2.2).

It was previously indicated that the processes of transporting synchronization by a ~~two-step,~~ peer-to-peer transparent clock that is syntonized and by a ~~two-step~~ boundary clock are mathematically and functionally equivalent. This is because the computations described above compute the synchronized time when the Sync message is sent by the time-aware system. The same computations are done if time-aware system *i* sends a Sync message without having received a new Sync message, i.e., if Sync receipt timeout occurs (see 10.7.3.1). In this case, time-aware system *i* uses the most recently received time-synchronization information from time-aware system *i*–1, which would be prior to time-aware system *i* having sent its most recent Sync message. The synchronized time corresponding to the sending of a Sync message is equal to the sum of the preciseOriginTimestamp and correctionField. Normally a boundary clock places this entire value, except for any sub-nanosecond portion, in the preciseOriginTimestamp, while a transparent clock retains the preciseOriginTimestamp and updates the ~~correction field~~correctionField. However, the sum of the two fields is equal to the synchronized time when the Sync message is sent in both cases.

The ratio of the grandmaster frequency to the frequency of the LocalClock entity at time-aware system *i*, rateRatio$_i$, is equal to the same quantity at time-aware system *i*–1, rateRatio$_{i-1}$, multiplied by the ratio of the frequency of the LocalClock entity at time-aware system *i*–1 to the frequency of the LocalClock entity at time-aware system *i*, neighborRateRatio (see 10.2.4.7). If neighborRateRatio is sufficiently small, this is approximately equal to the sum of rateRatio$_{i-1}$ and the quantity neighborRateRatio–1, which is the frequency offset of time-aware system *i*–1 relative to time-aware system *i*. This computation is done by the PortSyncSyncReceive state machine (see 10.2.7).

NOTE—the sending of time-synchronization information by the master ports of a time-aware system might or might not be tightly synchronized with the receipt of time-synchronization information by the slave port. If the master port has the same logMessageInteval as the slave port, it will transmit timing event messages as soon as possible after the slave port has received the corresponding timing event messages, and the master port is operating in “syncLocked” mode (see 10.2.4.15). If the master port and slave port have different logMessageInterval values, then the master port can send timing event messages without any synchronization with the slave port.

11.1.4 Model of operation

A time-aware system contains one MD entity per port. This entity contains functions generic to all media, which are described in Clause 10, and functions specific to the respective medium for the link. Functions specific to full-duplex, point-to-point links are described in the current clause.

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

The model for a time-aware system with full-duplex, point-to-point links is shown in Figure 11-3. It assumes the presence of one full-duplex, point-to-point MD entity per port. The media-independent entities shown in Figure 11-3 are described in 10.1.1.

A general, media-independent description of the generation of timestamps is given in 8.4.3. A more specific description for PTP event messages is given in 11.3.2.1. A PTP event message is timestamped relative to the LocalClock entity when the message timestamp point (see 3.12) crosses the timestamp measurement plane (see 3.27). The timestamp is corrected for any ingressLatency or egressLatency (see 8.4.3) to produce a timestamp relative to the reference plane (see 3.18). The corrected timestamp value is provided to the MD entity.

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

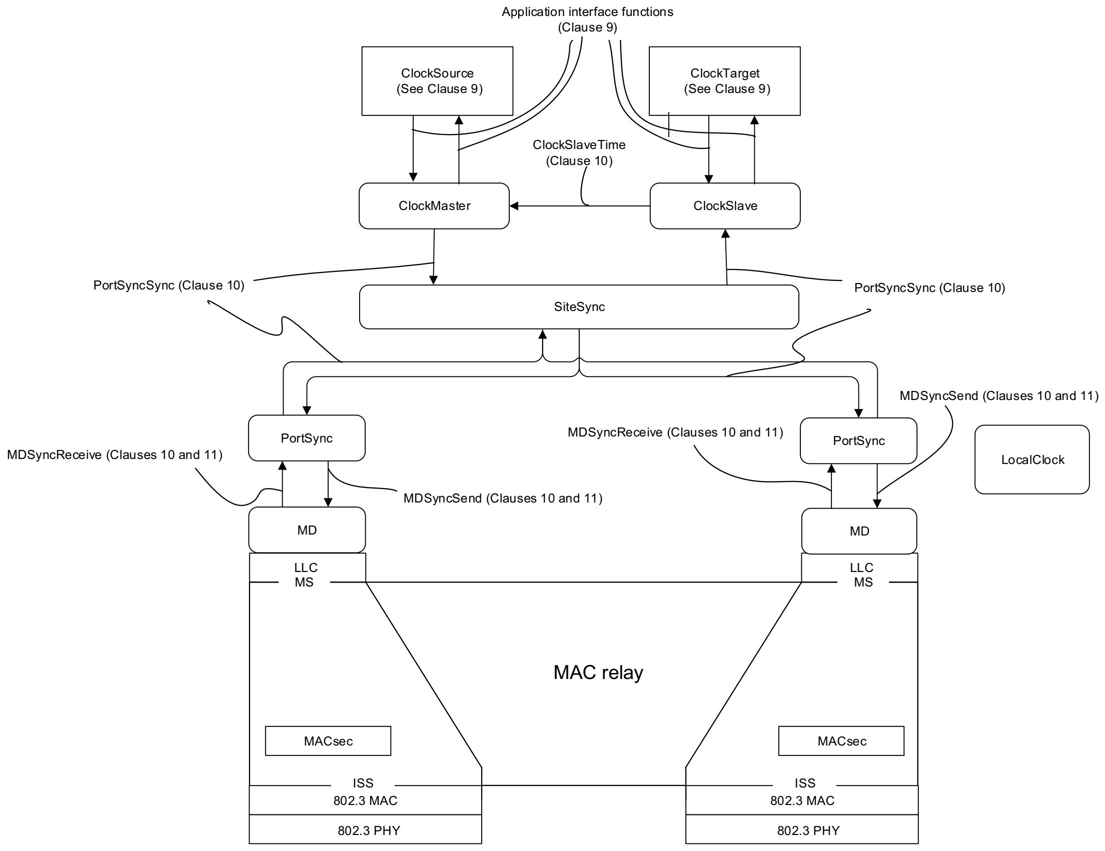


Figure 11-3—Model for time-aware system with full-duplex, point-to-point links

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

11.2.1 General

This subclause describes the media-dependent state machines for an MD entity, for the case of full-duplex, point-to-point links. The state machines are all per port, because an instance of each is associated with an MD entity. However, the MDSyncReceiveSM, MDSyncSendSM, and SyncIntervalSetting state machines are also per domain, while there is a single instance of the MDPdelayReq, MDPdelayResp, and LinkDelayIntervalSetting state machines for all the domains (but still per port). The state machines are as follows:

a) MDSyncReceiveSM (shown in Figure 10-2, and in more detail in Figure 11-4): receives Sync and, if the received information is two-step, Follow_Up messages, and sends the time-synchronization information carried in these messages to the PortSync entity of the same port. There is one instance of this state machine per port, per domain.
b) MDSyncSendSM (shown in Figure 10-2, and in more detail in Figure 11-4): receives an MDSyncSend structure from the PortSync entity of the same port, transmits a Sync message, uses the ~~<~~syncEventEgressTimestamp~~>~~, corrected for egressLatency, and information contained in the MDSyncSend structure to compute information needed for the Sync message if the port is currently operating as a one-step port and for the corresponding Follow_Up message if the port is currently

operating as a two-step port, and transmits the Follow_Up message if the port is two-step. There is one instance of this state machine per port, per domain.

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

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

e) ~~LinkDelay~~SyncIntervalSetting state machine (not shown): receives a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3), and sets the global variables that give the duration of the mean intervals between successive Sync~~ and successive Pdelay_Req~~ messages.

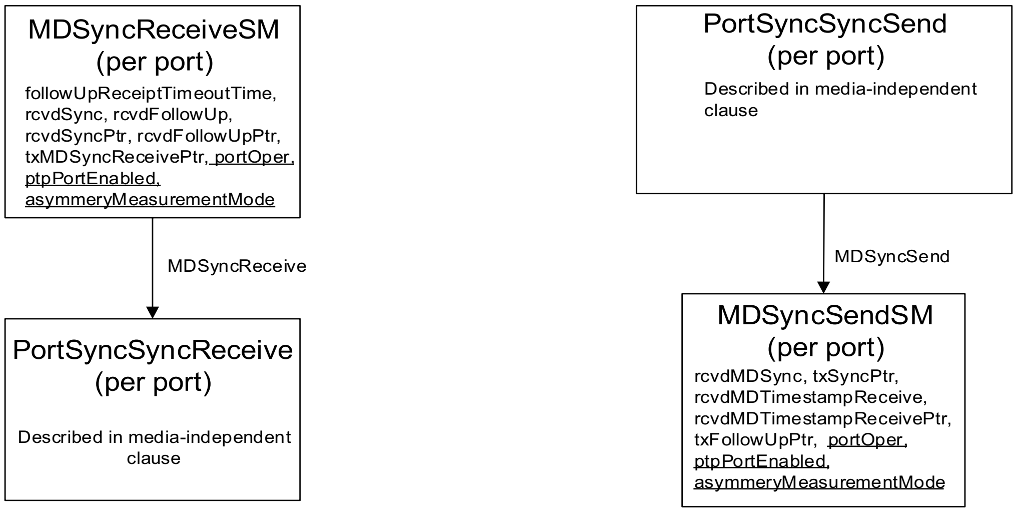

MDSyncReceiveSM (per port)
followUpReceiptTimeoutTime, rcvdSync, rcvdFollowUp, rcvdSyncPtr, rcvdFollowUpPtr, txMDSyncReceivePtr, portOper, ptpPortEnabled, asymmeryMeasurementMode
MDSyncReceive
PortSyncSyncReceive (per port)
Described in media-independent clause
PortSyncSyncSend (per port)
Described in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

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

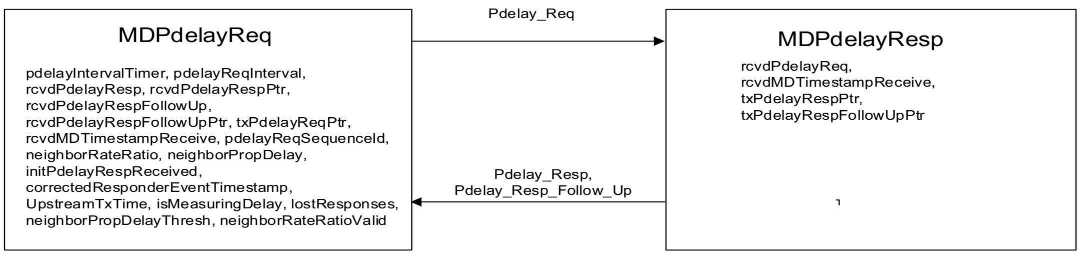

MDPdelayReq
pdelayIntervalTimer, pdelayReqInterval, rcvdPdelayResp, rcvdPdelayRespPtr, rcvdPdelayRespFollowUp, rcvdPdelayRespFollowUpPtr, txPdelayReqPtr, rcvdMDTimestampReceive, pdelayReqSequenceId, neighborRateRatio, neighborPropDelay, initPdelayRespReceived, correctedResponderEventTimestamp, UpstreamTxTime, isMeasuringDelay, lostResponses, neighborPropDelayThresh, neighborRateRatioValid
Pdelay_Req
Pdelay_Resp, Pdelay_Resp_Follow_Up
MDPdelayResp
rcvdPdelayReq, rcvdMDTimestampReceive, txPdelayRespPtr, txPdelayRespFollowUpPtr

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

There is one instance of this state machine per port, per domain.

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

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

The state machines described 11.2 and its subclauses use some of the global per-time-aware system variables defined in 10.2.3, the global per-port variables defined in 10.2.4, and the functions defined in 10.2.5.

11.2.2 Determination of asCapable and asCapableAcrossDomains

There is one instance of the global variable asCapable (see 10.2.4.1) per port, per domain.There is one instance of the global variable asCapableAcrossDomains (see 11.2.12.12, per port, that is common across and accessible by all the domains.

The per-port global variable asCapable (see 10.2.4.1) indicates whether or not the IEEE 802.1AS protocol is operating~~,~~ in this domain, on the link attached to this port, and can provide the time-synchronization performance described in B.3. asCapable is used by the PortSync entity, which is media-independent; however, the determination of asCapable is media-dependent.

The per-port global variable asCapableAcrossDomains is set by the MDPdelayReq state machine (see 11.2.15 and Figure 11-8). For a port attached to a full-duplex, point-to-point link, asCapableAcrossDomains shall be set to TRUE if and only if it is determined, via the peer delay mechanism, that the following conditions hold for the port:

a) The port is exchanging peer delay messages with its neighbor,
b) The measured delay does not exceed neighborPropDelayThresh,
c) The port does not receive multiple Pdelay_Resp or Pdelay_Resp_Follow_Up messages in response to a single Pdelay_Req message, and
d) The port does not receive a response from itself or another port of the same time-aware system.

~~asCapable is set in the MDPdelayReq state machine (see 11.2.15 and Figure 11-8).~~

NOTE—If a time-aware system implements only domain 0 and the MDPdelayReq and MDPdelayResp state machines are invoked on domain 0 (see 11.2.15), asCapableAcrossDomains is still set by the MDPdelayReq state machine.

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

Table 11-1—Value of neighborPropDelayThresh for various links

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

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

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

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

11.2.3 Use of MAC Control PAUSE operation

A time-aware system shall not use the MAC Control PAUSE operation.

11.2.4 Use of priority-based flow control

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

11.2.5 Use of link aggregation

~~The use of link aggregation is not specified. If link aggregation is used, Sync and Pdelay_Resp messages must be part of the same conversation. In addition, if link aggregation is used and the Sync and Pdelay_Resp messages use a different physical link from that used by Pdelay_Req messages in the opposite direction, there can be errors in measured propagation time, i.e., in neighborPropDelay, and measured time offset between the two time-aware systems. The absolute value of the error in neighborPropDelay and measured time offset is equal to the absolute value of one-half the difference between the actual propagation times in the two directions, i.e., the absolute value of the quantity delayAsymmetry (see 8.3 and 10.2.4.9), for the physical links actually used in the two directions.~~gPTP PDUs, when used with physical ports that are attached to an IEEE 802.1AX Aggregated Link, shall be transmitted and received over the physical ports, not the Aggregated Port. Using the Parser/Multiplexer functions described in Clauses 6.1.3 and 6.2.7 of IEEE 802.1AX-2014, received gPTP PDUs are recognized as control frames, separated from the incoming data stream, and directed to the gPTP layers. gPTP PDUs output from the gPTP layers are integrated into the port's data stream by the Parser/Multiplexer.

Assuming that ptpPortEnabled is true for all the physical links on the Aggregated Link, and that all the physical links are connected to the same two systems, gPTP will measure the delay on each physical link, and the protocol will choose one of the physical links for transmitting time from the master clock.

11.2.6 Service interface primitives and data structures communicated between state machines

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

11.2.7 DL-UNITDATA.request

This service primitive is used by an MD entity to request to the associated LLC the transmission of a Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, or Pdelay_Resp_Follow_Up message. The primitive is described in 2.2.1.1.1 of ISO/IEC 8802-2 [B9].

11.2.8 DL-UNITDATA.indication

This service primitive is used by the LLC to indicate to the associated MD entity the receipt of a Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, or Pdelay_Resp_Follow_Up message. The primitive is described in 2.2.1.1.1 of ISO/IEC 8802-2 [B9].

11.2.9 MDTimestampReceive

11.2.9.1 General

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

The structure is:

```
MDTimestampReceive{
        timestamp
}
```

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

11.2.9.2 timestamp (UScaledNs)

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

11.2.10 MDSyncReceive

This structure is specified in 10.2.2.2.

11.2.11 MDSyncSend

This structure is specified in 10.2.2.1.

11.2.12 MD entity global variables

11.2.12.1 currentLogPdelayReqInterval: the current value of the logarithm to base 2 of the mean time interval, in seconds, between the sending of successive Pdelay_Req messages (see 11.5.2.2). This value is set in the ~~LinkDelaySyncInterval~~LinkDelayIntervalSetting state machine (see 11.2.19). The data type for currentLogPdelayReqInterval is Integer8. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

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

11.2.12.3 pdelayReqInterval: a variable containing the mean Pdelay_Req message transmission interval for the port corresponding to this MD entity. The value is set in the ~~LinkDelaySyncInterval~~LinkDelayIntervalSetting state machine (see 11.2.19). The data type for pdelayReqInterval is UScaledNs. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.12.4 allowedLostResponses: the number of Pdelay_Req messages for which a valid response is not received, above which a port is considered to not be exchanging peer delay messages with its neighbor. The data type for allowedLostResponses is UInteger16. The ~~required~~default value and range of allowedLostResponses is given in 11.5.3. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.12.5 allowedFaults: the number of instances where neighborPropDelay (see 10.2.4.8) exceeds neighborPropDelayThresh (see 11.2.12.7) and/or the computation of neighborRateRatio is invalid (see 11.2.17.2.11), above which asCapableAcrossDomains is set to FALSE, i.e., the port will be considered to not be capable of interoperating with its neighbor via the 802.1AS protocol (see 10.2.4.1). The default value and range of allowedFaults is given in 11.5.4. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

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

11.2.12.7 neighborPropDelayThresh: the propagation time threshold, above which a port is not considered capable of participating in the IEEE 802.1AS protocol. If neighborPropDelay (see 10.2.4.8) exceeds neighborPropDelayThresh, then asCapableAcrossDomains (see 11.2.12.4) is set to FALSE. The data type for neighborPropDelayThresh is UScaledNs. There is one instance of this variable for all the domains (per port). The variable is accessible by all the domains.

11.2.12.8 syncSequenceId: the sequenceId ~~-~~(see 11.4.2.7) for the next Sync message to be sent by this MD entity. The data type for syncSequenceId is UInteger16.

11.2.12.9 oneStepReceive: a Boolean variable that is TRUE if the port is capable of receiving one-step Sync messages.

11.2.12.10 oneStepTransmit: a Boolean variable that is TRUE if the port is capable of transmitting one-step Sync messages.

11.2.12.11 oneStepTxOper: a Boolean variable that is TRUE if the port will be transmitting one-step Sync messages (see 11.1.3).

11.2.12.12 asCapableAcrossDomains: a Boolean that isTRUE if and only if conditions (a)-(d) of 11.2.2 are satisfied. This Boolean is set by the MDPdelayReq state machine, and is used in determining asCapable for a port (see 11.2.2).

11.2.13 MDSyncReceiveSM state machine

11.2.13.1 State machine variables

The following variables are used in the state diagram of 11.2.13.3:

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

11.2.13.1.2 rcvdSync: a Boolean variable that notifies the current state machine when a Sync message is received. This variable is reset by the current state machine.

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

11.2.13.1.4 rcvdSyncPtr: a pointer to a structure whose members contain the values of the fields of the Sync message whose receipt is indicated by rcvdSync (see 11.2.13.1.2).

11.2.13.1.5 rcvdFollowUpPtr: a pointer to a structure whose members contain the values of the fields of the Follow_Up message whose receipt is indicated by rcvdFollowUp (see 11.2.13.1.3).

11.2.13.1.6 txMDSyncReceivePtr: a pointer to a structure whose members contain the values of the parameters of an MDSyncReceive structure to be transmitted.

11.2.13.1.7 upstreamSyncInterval: the sync interval (see 10.7.2.1) for the upstream port that sent the received Sync message.

11.2.13.2 State machine functions

The following functions are used in the state diagram of 11.2.13.3:

11.2.13.2.1 setMDSyncReceive(): creates an MDSyncReceive structure, and returns a pointer to this structure. The members of this structure are set as follows:

a) If twoStepFlag of the most recently received Sync message is TRUE, followUpCorrectionField is set equal to the sum of the correctionField (see 11.4.2.511.4.2.5) of the most recently received Sync message, plus the correctionField of the most recently received FollowUp message; else followUpCorrectionField is set equal to the correctionField of the most recently received Sync message,

b) sourcePortIdentity is set equal to the sourcePortIdentity (see 11.4.2.6) of the most recently received Sync message (see 11.4.3),

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

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

<<Editor's note: Even though this member carries an originTimestamp in the one-step case (and a preciseOriginTimestamp only in the two-step case, for now we retain the name "preciseOriginTimestap.>>

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

f) upstreamTxTime is set equal to the ~~<~~syncEventIngressTimestamp~~>~~ for the most recently received Sync message (see 11.4.3), minus the mean propagation time on the link attached to this port (neighborPropDelay, see 10.2.4.8) divided by neighborRateRatio (see 10.2.4.7), minus delayAsymmetry (see 10.2.4.9) for this port divided by rateRatio [see e) above]. The ~~<~~syncEventIngressTimestamp~~>~~ is equal to the timestamp value measured relative to the timestamp measurement plane, minus any ingressLatency (see 8.4.3),

NOTE 1—The mean propagation time is divided by neighborRateRatio to convert it from the time base of the time-aware system at the other end of the attached link to the time base of the current time-aware system. The delayAsymmetry is divided by rateRatio to convert it from the time base of the grandmaster to the time base of the current time-aware system. The two quotients are then subtracted from ~~<~~syncEventIngressTimestamp~~>~~, which is measured relative to the time base of the current time-aware system.

NOTE 2—The difference between the mean propagation time in the grandmaster time base, the time base of the time-aware system at the other end of the link, and the time base of the current time-aware system is usually negligible. The same is true of any delayAsymmetry. See NOTE 2 of 11.2.17.3.4.

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

11.2.13.2.2 txMDSyncReceive (txMDSyncReceivePtr): transmits an MDSyncReceive structure to the PortSyncSyncReceive state machine of the PortSync entity of this port.

11.2.13.3 State diagram

The MDSyncReceiveSM state machine shall implement the function specified by the state diagram in Figure 11-6, the local variables specified in 11.2.13.1, the functions specified in 11.2.13.2, the structure specified in 11.2.10 and 10.2.2.1, the messages specified in 11.4, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. The state machine receives Sync and, if the received information is two-step, Follow_Up messages, places the time-synchronization information in an

MDSyncReceive structure, and sends the structure to the PortSyncSyncReceive state machine of the PortSync entity of this port.

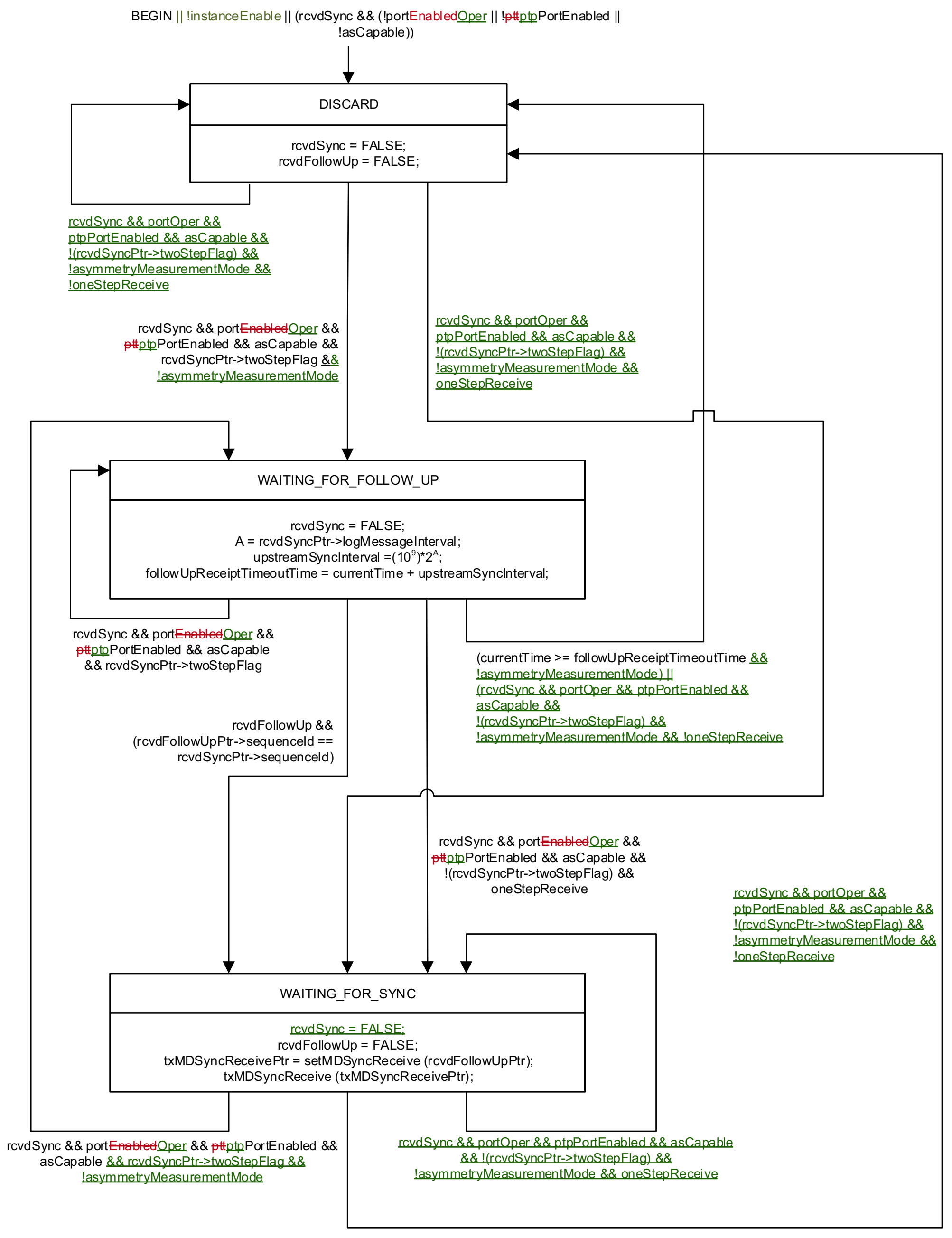

BEGIN || !instanceEnable || (rcvdSync && (!port~~Enabled~~Oper || !~~pt~~ptpPortEnabled || !asCapable))
DISCARD
rcvdSync = FALSE;
rcvdFollowUp = FALSE;
rcvdSync && portOper && ptpPortEnabled && asCapable && !(rcvdSyncPtr->twoStepFlag) && !asymmetryMeasurementMode && !oneStepReceive
rcvdSync && port~~Enabled~~Oper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

Figure 11-6—MDSyncReceiveSM state machine

<<Editor's note : The changes to Figure 11-6 are (a) a transition from the WAITING_FOR_FOLLOW_UP state back to itself, with the appropriate condition, has been added to address the problem of a single

FOLLOW_UP message causing Sync Receipt Timeout, (b) the variable A has been introduced in the WAITING_FOR_FOLLOW_UP state and set equal to the exponent of 2, and then 2 is raised to this power, to make the figure more readable, (c) the condition !asymmetryMeasurementMode has been added to the transitions from DISCARD to WATIING_FOR_FOLLOW_UP and WAITING_FOR_SYNC to WATIING_FOR_FOLLOW_UP, and the condition asymmetryMeasurementMode has been added (via an OR) to the transition from WAITING_FOR_FOLLOW_UP to DISCARD, (d) portEnabled is changed to portOper and pttPortEnabled is changed to ptpPortEnabled, and (e) one-step on receive has been added (addition of WAITING_FOR_SYNC state and transitions to and from this state, and checking whether the twoStepFlag of the pointer to the received Sync message is TRUE (two-step message received) or FALSE (one-step message received)(see http://www.ieee802.org/1/files/public/docs2014/as-garner-one-step-on-receive-1113-v01.pdf).>>

<<Editor's note : Item (a) in the previous editor's note addresses the problem of Sync Receipt Timeout being caused by loss of a single Follow_Up message, discussed in the July, 2013 TSN TG meeting (see http://www.ieee802.org/1/files/public/docs2013/as-garner-sync-receipt-timeout-issue-0713-v01.pdf). In the revised state diagram in that presentation (slide 11) a separate fix introduced in 802.AS-Cor-1 was neglected. This fix was to add the scale factor 10^9 to the right-hand side of the second line of code in the WAITING_FOR_FOLLOW_UP block. This fix is included here.>>

11.2.14 MDSyncSendSM state machine

11.2.14.1 State machine variables

The following variables are used in the state diagram of 11.2.14.3:

11.2.14.1.1 rcvdMDSync: a Boolean variable that notifies the current state machine when an MDSyncSend structure is received. This variable is reset by the current state machine.

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

11.2.14.1.3 rcvdMDTimestampReceive: a Boolean variable that notifies the current state machine when the ~~<~~syncEventEgressTimestamp~~>~~ (see 11.3.2.1) for a transmitted Sync message is received. This variable is reset by the current state machine.

11.2.14.1.4 rcvdMDTimestampReceivePtr: a pointer to the received MDTimestampReceive structure (see 11.2.9).

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

11.2.14.2 State machine functions

The following functions are used in the state diagram of 11.2.14.3:

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

a) correctionField is set equal to 0,
b) sourcePortIdentity is set equal to the sourcePortIdentity member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
c) sequenceId is set equal to syncSequenceId (see 11.2.12.8),
d) logMessageInterval is set equal to the logMessageInterval member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11), and
e) remaining fields are set as specified in 11.4.2 ~~-~~and 11.4.3, for the case where twoStepFlag is TRUE.

<<Editor's note: The name of the function setSync has been changed to setSyncTwoStep (in D1.0), to emphasize that the function is used if the time-aware system is two-step (an new function, setSyncOneStep is added (see 11.2.14.2.5) for the case where the time-aware system is one-step.>>

11.2.14.2.2 txSync (txSyncPtr): transmits a Sync message from this MD entity, whose fields contain the parameters in the structure pointed to by txSyncPtr (see 11.2.14.1.2).

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

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

      rateRatio × (~~<~~syncEventEgressTimestamp~~>~~ – upstreamTxTime),

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

NOTE—If the time-aware system that contains this PortSync entity is a ~~Bridge~~time-aware relay, the quantity

~~<~~syncEventEgressTimestamp~~>~~ – upstreamTxTime

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

rateRatio × (~~<~~syncEventEgressTimestamp~~>~~ – upstreamTxTime)

is the sum of the residence time and link propagation delay on the upstream link, relative to the grandmaster (see 11.2.13.2.1 for details on the setting of upstreamTxTime).

b) sourcePortIdentity is set equal to the sourcePortIdentity member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
c) sequenceId is set equal to syncSequenceId (see 11.2.12.8),
d) logMessageInterval is set equal to the logMessageInterval member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
e) preciseOriginTimestamp is set equal to the preciseOriginTimestamp member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
f) rateRatio is set equal to the rateRatio member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
g) gmTimeBaseIndicator is set equal to the gmTimeBaseIndicator member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
h) lastGmPhaseChange is set equal to the lastGmPhaseChange member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
i) lastGmFreqChange is set equal to the scaledLastGmFreqChange member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11), multiplied by 2^{41},
j) domainNumber is set equal to the domainNumber of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11), and
k) remaining fields are set as specified in 11.4.2 and 11.4.4.

11.2.14.2.4 txFollowUp (txFollowUpPtr): transmits a Follow_Up message from this MD entity, whose fields contain the parameters in the structure pointed to by txFollowUpPtr (see 11.2.14.1.5).

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

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

<<Editor's note: The correctionField portion of the incoming time-synchronization information can be placed in the correctionField of the outgoing Sync message when the outgoing Sync message is prepared; it is only the portion that depends on the <syncEventEgressTimestamp>> (i.e., the portion that includes residence time) that must be added as the Sync message is transmitted. This is done by the function modifySync() (see 11.2.14.2.6).>>

b) sourcePortIdentity is set equal to the sourcePortIdentity member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
c) sequenceId is set equal to syncSequenceId (see 11.2.12.8),
d) logMessageInterval is set equal to the logMessageInterval member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
e) originTimestamp is set equal to the preciseOriginTimestamp member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),

<<Editor's note: Even though this member carries an originTimestamp in the one-step case (and a preciseOriginTimestamp only in the two-step case, for now we retain the name "preciseOriginTimestap.>>

f) rateRatio is set equal to the rateRatio member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
g) gmTimeBaseIndicator is set equal to the gmTimeBaseIndicator member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
h) lastGmPhaseChange is set equal to the lastGmPhaseChange member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11),
i) lastGmFreqChange is set equal to the scaledLastGmFreqChange member of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11), multiplied by 2^{41},
j) domainNumber is set equal to the domainNumber of the most recently received MDSyncSend structure (see 10.2.2.1 and 11.2.11), and
k) remaining fields are set as specified in 11.4.2 and 11.4.3, for the case where twoStepFlag is FALSE.

11.2.14.2.6 **modifySync():** adds to the correctionField of the transmitted Sync message, after the syncEventEgressTimestamp is taken and known, the quantity:

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

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

NOTE—If the time-aware system that contains this PortSync entity is a time-aware relay, the quantity

$$\text{syncEventEgressTimestamp} - \text{upstreamTxTime}$$

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

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

is the sum of the residence time and link propagation delay on the upstream link, relative to the grandmaster (see

11.2.13.2.1 for details on the setting of upstreamTxTime).

11.2.14.3 State diagram

The MDSyncSendSM state machine shall implement the function specified by the state diagram in Figure 11-7; the local variables specified in 11.2.14.1; the functions specified in 11.2.14.2; the structures specified in 11.2.9, 11.2.11 and 10.2.2.1; the messages specified in 11.4; the MD entity global variables specified in 11.2.12; and the relevant global variables and functions specified in 10.2.3 through 10.2.5. The state machine receives an MDSyncSend structure from the PortSyncSyncSend state machine of the PortSync entity of this port and transmits a Sync and corresponding Follow_Up message.

<<Editor's note : In Figure 11-7 (a) the condition !asymmetryMeasurementMode has been added to the transitions from INITIALIZING to SEND_SYNC and from SEND_FOLLOW_UP to SEND_SYNC, portEnabled has been changed to portOper, and pttPortEnabled has been changed to ptpPortEnabled; (b) the state machine has been extended to cover one-step operation in addition to two-step operation (the states SEND_SYNC_ONE_STEP and SET_CORRECTION_FIELD have been added, with appropriate new state transitions, and the name of the existing SEND_SYNC state has been changed to SEND_SYNC_TWO_STEP; the name of the existing function setSync has been changed to setSyncTwoStep).>>

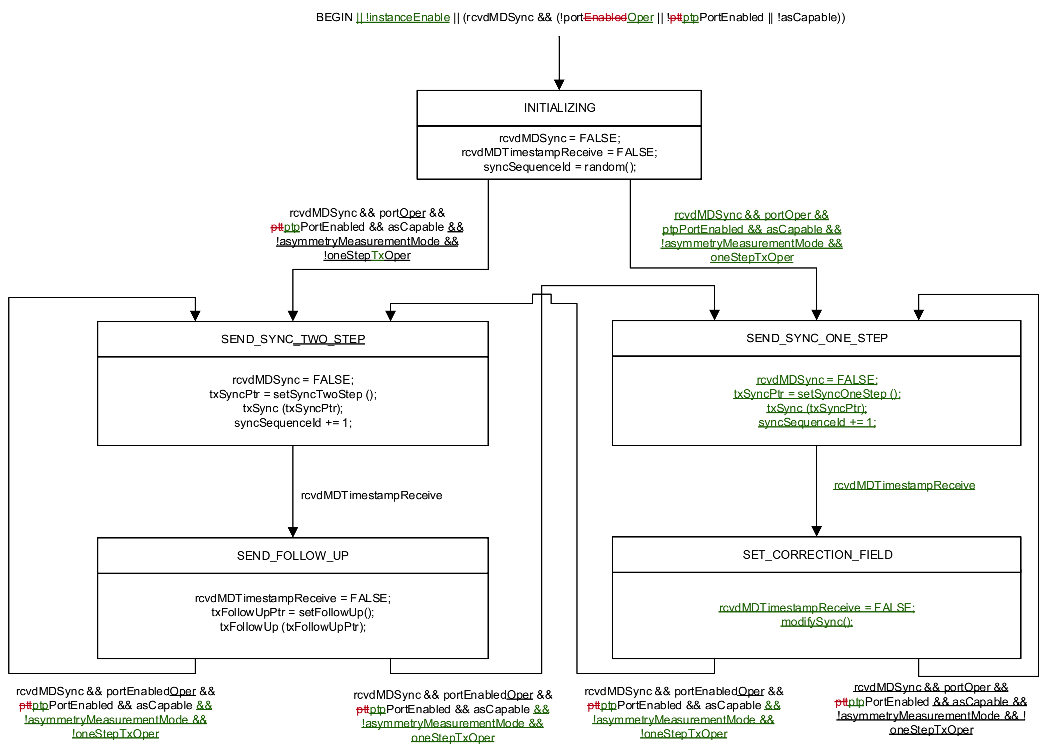

BEGIN || !instanceEnable || (rcvdMDSync && (!port~~Enabled~~Oper || !~~ptt~~ptpPortEnabled || !asCapable))
INITIALIZING
rcvdMDSync = FALSE;
rcvdMDTimestampReceive = FALSE;
syncSequenceId = random();
rcvdMDSync && portOper && ~~ptt~~ptpPortEnabled && asCapable && !asymmetryMeasurementMode && !oneStepTxOper
rcvdMDSync && portOper && ptpPortEnabled && asCapable && !asymmetryMeasurementMode && oneStepTxOper
SEND_SYNC_TWO_STEP
rcvdMDSync = FALSE;
txSyncPtr = setSyncTwoStep ();
txSync (txSyncPtr);
syncSequenceId += 1;
SEND_SYNC_ONE_STEP
rcvdMDSync = FALSE;
txSyncPtr = setSyncOneStep ();
txSync (txSyncPtr);
syncSequenceId += 1;
rcvdMDTimestampReceive
rcvdMDTimestampReceive
SEND_FOLLOW_UP
rcvdMDTimestampReceive = FALSE;
txFollowUpPtr = setFollowUp();
txFollowUp (txFollowUpPtr);
SET_CORRECTION_FIELD
rcvdMDTimestampReceive = FALSE;
modifySync();
rcvdMDSync && portEnabledOper && ~~ptt~~ptpPortEnabled && asCapable && !asymmetryMeasurementMode && !oneStepTxOper
rcvdMDSync && portEnabledOper && ~~ptt~~ptpPortEnabled && asCapable && !asymmetryMeasurementMode && oneStepTxOper
rcvdMDSync && portEnabledOper && ~~ptt~~ptpPortEnabled && asCapable && !asymmetryMeasurementMode && !oneStepTxOper
rcvdMDSync && portOper && ~~ptt~~ptpPortEnabled && asCapable && !asymmetryMeasurementMode && ! oneStepTxOper

Figure 11-7—MDSyncSendSM state machine

11.2.15 Common Mean Link Delay Service

Each port of a time-aware system invokes a single instance of the MDPdelayReq state machine (see 11.2.16) and the MDPdelayResp state machine (see 11.2.18). 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 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 link attached to this port, to the frequency of the LocalClock entity of this time-aware system). The CMLDS makes the mean propagation delay and neighbor rate ratio available to all active domains. If the time-aware system implements one domain (the domainNumber of this domain is 0, see 8.1), these two state machines may provide the CMLDS; however, if they do not provide the CMLDS (i.e., if only the PTP-instance-specific peer delay mechansim is provided), they shall be invoked on domain 0. This means that if the domain number is not 0, portDS.delayMechanism (see 14.8.5 and Table 14-8) must not be P2P.

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

In accordance with IEEE Std 1588-20xx, the term Link Port is used to refer to a port of the CMLDS. A PTP port for which portDS.delayMechanism is COMMON_P2P uses the CMLDS provided by the Link Port that

The value of majorSdoId for the CMLDS shall be 0x2. The value of minorSdoId for the Common Mean Link Delay Service shall be 0x00. This means that 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 gPTP port that invokes the CMLDS receives a Pdelay_Req messsage with majorSdoId value of 0x1, minorSdoId value of 0x00, and domainNumber value of 0, the 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 delay mechanism. These instance-specific messages have majorSdoId value of 0x1, minorSdoId value of 0x00, and domainNumber value of 0.

NOTE 3—The above requirement ensures (a) backward compatibility with time-aware systems that comply with the 2011 version of this standard, and (b) compatibility with time-aware systems that implement only one domain and invoke the MDPdelayReq and MDPdelayResp state machines on domain 0.

NOTE 4—In general, a port can receive (i) peer delay messages of the CMLDS, (ii) gPTP instance-specific peer delay messages of domain 0 (with sdoId of 0x100), and, (iii) 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 (i) if it invokes CMLDS, the messages of (ii) if it invokes gPTP domain 0, and the messages of (iii) if it invokes the respective other PTP profiles.

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

11.2.16 Common Mean Link Delay Service global variables

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

<<Editor's note: Other global variables of clause 11 that are common across all domains must be moved to this subclause. Actually, two subclauses are needed for the variables that are common across all domains, one for those that are per port, and one for those that are per time-aware system See item 18 of the frontmatter, and comment #12 against D4.5.>>

11.2.17 MDPdelayReq state machine

11.2.17.1 General

This state machine is invoked as part of the Common Mean Link Delay Service. There is one instance of this state machine for all the domains (per port). This means that there also is one instance of each of the state machine variables of 11.2.17.2, state machine functions of 11.2.17.3, and relevant global variables of 10.2.4 and 11.2.12 for all the domains (per port). None of the variables used or functions invoked in this state machine are specific to a single domain. However, the single instances of all of these objects or entities are accessible to all the domains.

11.2.17.2 State machine variables

The following variables are used in the state diagram of 11.2.17.4:

11.2.17.2.1 pdelayIntervalTimer: a variable used to save the time at which the Pdelay_Req interval timer is started, see Figure 11-8. A Pdelay_Req message is sent when this timer expires. The data type for pdelayIntervalTimer is UScaledNs.

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

11.2.17.2.3 rcvdPdelayRespPtr: a pointer to a structure whose members contain the values of the fields of the Pdelay_Resp message whose receipt is indicated by rcvdPdelayResp (see 11.2.17.2.2).

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

11.2.17.2.5 rcvdPdelayRespFollowUpPtr: a pointer to a structure whose members contain the values of the fields of the Pdelay_Resp_Follow_Up message whose receipt is indicated by rcvdPdelayRespFollowUp (see 11.2.17.2.4).

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

11.2.17.2.7 rcvdMDTimestampReceive: a Boolean variable that notifies the current state machine when the ~~<~~pdelayReqEventEgressTimestamp~~>~~ (see 11.3.2.1) for a transmitted Pdelay_Req message is received. This variable is reset by the current state machine.

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

11.2.17.2.9 initPdelayRespReceived: a Boolean variable that indicates whether or not initial Pdelay_Resp and Pdelay_Resp_Follow_Up messages have been received when initializing the neighborRateRatio measurement. This variable is initialized to FALSE when the port initializes or re-initializes, and after a Pdelay_Resp and/or Pdelay_Resp_Follow_Up message are not received in response to a sent Pdelay_Req message.

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

11.2.17.2.11 neighborRateRatioValid: a Boolean variable that indicates whether or not the function computePdelayRateRatio() (see 11.2.17.3.3) successfully computed neighborRateRatio (see 10.2.4.7).

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

11.2.17.2.13 portEnabled0: a Boolean variable whose value is equal to ptpPortEnabled (see 10.2.4.13) if this state machine is invoked by the instance-specific peer delay mechanism, and is equal to cmldsLinkPortEnabled (see 11.2.16.1) if this state machine is invoked by the CMLDS.

11.2.17.3 State machine functions

The following functions are used in the state diagram of 11.2.17.4:

11.2.17.3.1 setPdelayReq(): creates a structure containing the parameters (see 11.4 and its subclauses) of a Pdelay_Req message to be transmitted, and returns a pointer, txPdelayReqPtr (see 11.2.17.2.6), to this structure. The parameters are set as follows:

1) sourcePortIdentity is set equal to the port identity of the port corresponding to this MD entity (see 8.5.2),
2) sequenceId is set equal to pdelayReqSequenceId (see 11.2.17.2.8), and
3) remaining parameters are set as specified in 11.4.2 and 11.4.5.

11.2.17.3.2 txPdelayReq(txPdelayReqPtr): transmits a Pdelay_Req message from the MD entity, containing the parameters in the structure pointed to by txPdelayReqPtr (see 11.2.17.2.6).

11.2.17.3.3 computePdelayRateRatio(): computes neighborRateRatio (see 10.2.4.7) using the following information conveyed by successive Pdelay_Resp and Pdelay_Resp_Follow_Up messages:

a) The ~~<~~pdelayRespEventIngressTimestamp~~>~~ (see 11.3.2.1) values for the respective Pdelay_Resp messages
b) The correctedResponderEventTimestamp values, whose data type is UScaledNs, obtained by adding the following fields of the received Pdelay_Resp_Follow_Up message:
    1) The seconds field of the responseOriginTimestamp field, multiplied by 10^9,
    2) The nanoseconds field of the responseOriginTimestamp parameter, and
    3) The correctionField, divided by 2^{16}.

Any scheme that uses the preceding information, along with any other information conveyed by the successive Pdelay_Resp and Pdelay_Resp_Follow_Up messages, to compute neighborRateRatio is acceptable as long as the performance requirements specified in B.2.4 are met. If neighborRateRatio is successfully computed, the Boolean neighborRateRatioValid (see 11.2.17.2.11) is set to TRUE. If neighborRateRatio is not successfully computed (e.g., if the MD entity has not yet exchanged a sufficient number of peer delay messages with its peer), the Boolean neighborRateRatioValid is set to FALSE.

NOTE 1—As one example, neighborRateRatio can be estimated as the ratio of the elapsed time of the LocalClock entity of the time-aware system at the other end of the link attached to this port, to the elapsed time of the LocalClock entity of this time-aware system. This ratio can be computed for the time interval between a set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages and a second set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages some number of Pdelay_Req message transmission intervals later, i.e.,

$$\frac{\langle \text{correctedResponderEventTimestamp} \rangle_N - \langle \text{correctedResponderEventTimestamp} \rangle_0}{\langle \text{pdelayRespEventIngressTimestamp} \rangle_N - \langle \text{pdelayRespEventIngressTimestamp} \rangle_0}$$

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

NOTE 2—This function must account for non-receipt of Pdelay_Resp and/or Pdelay_Resp_Follow_Up for a Pdelay_Req message, and also for receipt of multiple Pdelay_Resp messages within one Pdelay_Req message transmission interval.

11.2.17.3.4 computePropTime(): computes the mean propagation delay on the link attached to this MD entity, D, and returns this value. D is given by Equation (11-5):

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

where

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

t_1 = ~~<~~pdelayReqEventEgressTimestamp~~>~~ (see 11.3.2.1) for the Pdelay_Req message sent by the P2PPort entity, expressed in ns

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

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

r = current value of neighborRateRatio for this MD entity (see 10.2.4.7)

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 link. It is divided by neighborRateRatio (see 10.2.4.7) to convert it to the time base of the current time-aware system when adding to ~~<~~syncEventIngressTimestamp~~>~~ to compute upstreamTxTime [see 11.2.13.2.1 f)].

NOTE 2—The difference between mean propagation delay relative to the grandmaster time base and relative to the time base of the time-aware system at the other end of the attached 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 frequency to the frequency of the LocalClock entity of the time-aware system at the other end of the 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 link, relative to the grandmaster, 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 1588-2008, the computation of Eq. (11-5) is organinzed differently. Using the defintions of t_2 and t_3 above, Eq. (11-5) can be rewritten:

$$D = [r \cdot (t_4 - t_1) - (\text{responseOriginTimestamp} - \text{requestReceiptTimestamp}$$

$$+ (\text{correctionField of Pdelay\_Resp}) - (\text{correctionField of Pdelay\_Resp\_Follow\_Up})] / 2 \quad (11\text{-}6)$$

where each term is expressed in units of ns as described in the definitions of t_1, t_2, t_3, and t_4 above. In IEEE Std 1588-2008, the fractional ns portion of t_2 is subtracted from the correctionField of Pdelay_Resp, rather than added as in this standard; however, the correctionField of Pdelay_Resp is then subtracted in Eq. (11-6) rather than added, and the two minus signs cancel. The computations of *D* in this standard and IEEE 1588-2008 are mathematically equivalent.

11.2.17.4 State diagram

The MDPdelayReq state machine shall implement the function specified by the state diagram in Figure 11-8, the local variables specified in 11.2.17.2, the functions specified in 11.2.17.3, the messages specified in 11.4, and the relevant global variables and functions specified in 11.2.12 and 10.2.3 through 10.2.5. This state machine is responsible for the following:

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

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

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

<<Editor’s note : The MDPdelayReq state machine in Figure 11-8 is modified in accordance with the agreement at the May, 2014 802.1 TSN TG meeting on resolving the Pdelay_Req storm issue (maintenance request 135). The “UCT” transition from the RESET to SEND_PDELAY_REQ state is changed to “currentTime - pdelayIntervalTimer >= pdelayReqInterval”. In addition, rcvdPdelayResp and rcvdPdelayRespFollowUp are initialized.>>

11.2.18 MDPdelayResp state machine

11.2.18.1 General

This state machine is invoked as part of the Common Mean Link Delay Service. There is one instance of this state machine for all the domains (per port). This means that there also is one instance of each of the state machine variables of 11.2.18.2, state machine functions of 11.2.18.3, and relevant global variables of 10.2.4 and 11.2.12 for all the domains (per port). None of the variables used or functions invoked in this state machine are specific to a single domain. However, the single instances of all of these objects or entities are accessible to all the domains.

11.2.18.2 State machine variables

The following variables are used in the state diagram of 11.2.18.4:

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

11.2.18.2.2 rcvdMDTimestampReceive: a Boolean variable that notifies the current state machine when the ~~<~~pdelayRespEventEgressTimestamp~~>~~ (see 11.3.2.1) for a transmitted Pdelay_Resp message is received. This variable is reset by the current state machine.

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

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

11.2.18.2.5 portEnabled1: a Boolean variable whose value is equal to ptpPortEnabled (see 10.2.4.13) if this state machine is invoked by the instance-specific peer delay mechanism, and is equal to cmldsLinkPortEnabled (see 11.2.16.1) if this state machine is invoked by the CMLDS.

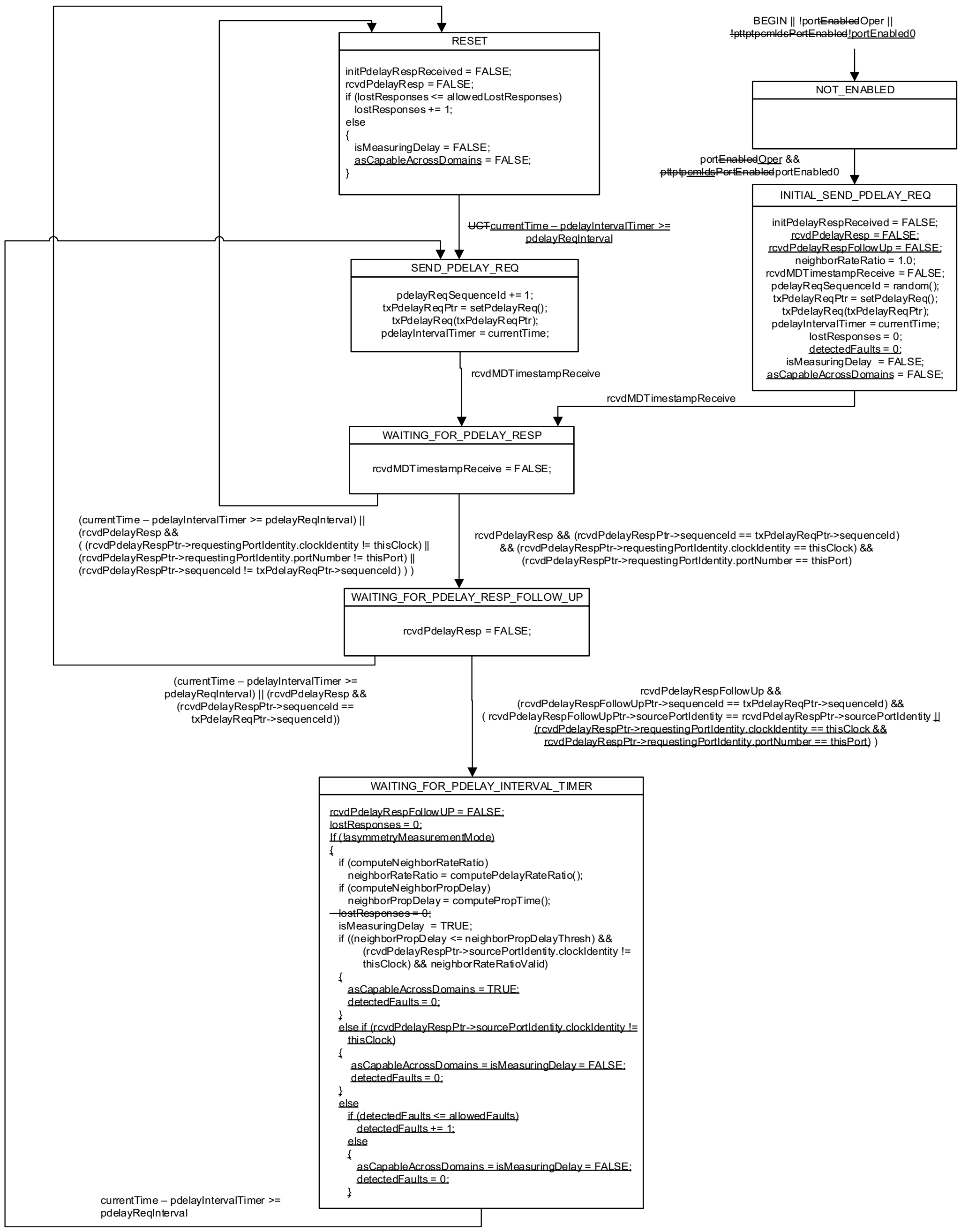

RESET
initPdelayRespReceived = FALSE;
rcvdPdelayResp = FALSE;
if (lostResponses <= allowedLostResponses)
lostResponses += 1;
else
{
isMeasuringDelay = FALSE;
asCapableAcrossDomains = FALSE;
}
BEGIN || !port~~Enabled~~Oper ||
~~!pttptpcmldsPortEnabled~~!portEnabled0
NOT_ENABLED
port~~Enabled~~Oper &&
~~pttptpcmldsPortEnabled~~portEnabled0
INITIAL_SEND_PDELAY_REQ
initPdelayRespReceived = FALSE;
rcvdPdelayResp = FALSE;
rcvdPdelayRespFollowUp = FALSE;
neighborRateRatio = 1.0;
rcvdMDTimestampReceive = FALSE;
pdelayReqSequenceId = random();
txPdelayReqPtr = setPdelayReq();
txPdelayReq(txPdelayReqPtr);
pdelayIntervalTimer = currentTime;
lostResponses = 0;
detectedFaults = 0;
isMeasuringDelay = FALSE;
asCapableAcrossDomains = FALSE;
~~UCT~~currentTime – pdelayIntervalTimer >= pdelayReqInterval
SEND_PDELAY_REQ
pdelayReqSequenceId += 1;
txPdelayReqPtr = setPdelayReq();
txPdelayReq(txPdelayReqPtr);
pdelayIntervalTimer = currentTime;
rcvdMDTimestampReceive
rcvdMDTimestampReceive
WAITING_FOR_PDELAY_RESP
rcvdMDTimestampReceive = FALSE;
(currentTime – pdelayIntervalTimer >= pdelayReqInterval) ||
(rcvdPdelayResp &&
((rcvdPdelayRespPtr->requestingPortIdentity.clockIdentity != thisClock) ||
(rcvdPdelayRespPtr->requestingPortIdentity.portNumber != thisPort) ||
(rcvdPdelayRespPtr->sequenceId != txPdelayReqPtr->sequenceId)))
rcvdPdelayResp && (rcvdPdelayRespPtr->sequenceId == txPdelayReqPtr->sequenceId)
&& (rcvdPdelayRespPtr->requestingPortIdentity.clockIdentity == thisClock) &&
(rcvdPdelayRespPtr->requestingPortIdentity.portNumber == thisPort)
WAITING_FOR_PDELAY_RESP_FOLLOW_UP
rcvdPdelayResp = FALSE;
(currentTime – pdelayIntervalTimer >= pdelayReqInterval) || (rcvdPdelayResp && (rcvdPdelayRespPtr->sequenceId == txPdelayReqPtr->sequenceId))
rcvdPdelayRespFollowUp &&
(rcvdPdelayRespFollowUpPtr->sequenceId == txPdelayReqPtr->sequenceId) &&
(rcvdPdelayRespFollowUpPtr->sourcePortIdentity == rcvdPdelayRespPtr->sourcePortIdentity ||
(rcvdPdelayRespPtr->requestingPortIdentity.clockIdentity == thisClock &&
rcvdPdelayRespPtr->requestingPortIdentity.portNumber == thisPort))
WAITING_FOR_PDELAY_INTERVAL_TIMER
rcvdPdelayRespFollowUP = FALSE;
lostResponses = 0;
If (!asymmetryMeasurementMode)
{
if (computeNeighborRateRatio)
neighborRateRatio = computePdelayRateRatio();
if (computeNeighborPropDelay)
neighborPropDelay = computePropTime();
~~lostResponses = 0;~~
isMeasuringDelay = TRUE;
if ((neighborPropDelay <= neighborPropDelayThresh) &&
(rcvdPdelayRespPtr->sourcePortIdentity.clockIdentity !=
thisClock) && neighborRateRatioValid)
{
asCapableAcrossDomains = TRUE;
detectedFaults = 0;
}
else if (rcvdPdelayRespPtr->sourcePortIdentity.clockIdentity !=
thisClock)
{
asCapableAcrossDomains = isMeasuringDelay = FALSE;
detectedFaults = 0;
}
else
if (detectedFaults <= allowedFaults)
detectedFaults += 1;
else
{
asCapableAcrossDomains = isMeasuringDelay = FALSE;
detectedFaults = 0;
}
currentTime – pdelayIntervalTimer >= pdelayReqInterval

Figure 11-8—MDPdelayReq state machine

<<Editor's note: In Figure 11-8, the variable asCapableAcrossDomains has replaced asCapable, to reflect the fact that the peer delay mechanism is invoked by the common pdelay service, which runs outside of any domain and whose results are accessible by all domains. In addition, whereas previously asCapable was set to FALSE immediately whenever neighborPropDelay exceeded the threshold or Pdelay_Resp was received from this clock or neighborRateRatio was not valid, now

some hysteresis is added for the cases where the threshold is exceeded or neighborRateRatio is invalid.>>

11.2.18.3 State machine functions

The following functions are used in the state diagram of 11.2.18.4:

11.2.18.3.1 setPdelayResp(): creates a structure containing the parameters (see 11.4 and its subclauses) of a Pdelay_Resp message to be transmitted, and returns a pointer, txPdelayRespPtr (see 11.2.18.2.3), to this structure. The parameters are set as follows:

a) sourcePortIdentity is set equal to the port identity of the port corresponding to this MD entity (see 8.5.2),
b) sequenceId is set equal to the sequenceId field of the corresponding Pdelay_Req message,
c) requestReceiptTimestamp is set equal to the ~~<~~pdelayReqEventIngressTimestamp~~>~~ (see 11.3.2) of the corresponding Pdelay_Req message, with any fractional ns portion truncated,
d) correctionField is set equal to the fractional ns portion of the ~~<~~pdelayReqEventIngressTimestamp~~>~~ of the corresponding Pdelay_Req message,
e) requestingPortIdentity is set equal to the sourcePortIdentity field of the corresponding Pdelay_Req message, and
f) remaining parameters are set as specified in 11.4.2 and 11.4.6.

11.2.18.3.2 txPdelayResp(txPdelayRespPtr): transmits a Pdelay_Resp message from the MD entity, containing the parameters in the structure pointed to by txPdelayRespPtr (see 11.2.18.2.3).

11.2.18.3.3 setPdelayRespFollowUp(): creates a structure containing the parameters (see 11.4 and its subclauses) of a Pdelay_Resp_Follow_Up message to be transmitted, and returns a pointer, txPdelayRespFollowUpPtr (see 11.2.18.2.4), to this structure. The parameters are set as follows:

a) sourcePortIdentity is set equal to the port identity of the port corresponding to this MD entity (see 8.5.2),
b) sequenceId is set equal to the sequenceId field of the corresponding Pdelay_Req message,
c) responseOriginTimestamp is set equal to the ~~<~~pdelayRespEventEgressTimestamp~~>~~ (see 11.3.2) of the corresponding Pdelay_Resp message, with any fractional ns truncated,
d) correctionField is set equal to the fractional ns portion of the ~~<~~pdelayRespEventEgressTimestamp~~>~~ of the corresponding Pdelay_Resp message,
e) requestingPortIdentity is set equal to the sourcePortIdentity field of the corresponding Pdelay_Req message, and
f) remaining parameters are set as specified in 11.4.2 and 11.4.6.

11.2.18.3.4 txFollowUp(txFollowUpPtr): transmits a Pdelay_Resp_Follow_Up message from the P2PPort entity containing the parameters in the structure pointed to by txPdelayRespFollowUpPtr (see 11.2.18.2.4).

11.2.18.4 State diagram

The MDPdelayResp state machine shall implement the function specified by the state diagram in Figure 11-9, the local variables specified in 11.2.18.2, the functions specified in , the messages specified in 11.4, and the relevant global variables and functions specified in 10.2.3 through 10.2.5. This state machine is responsible for responding to Pdelay_Req messages, received from the MD entity at the other end of the attached link, with Pdelay_Resp and Pdelay_Resp_Follow_Up messages.

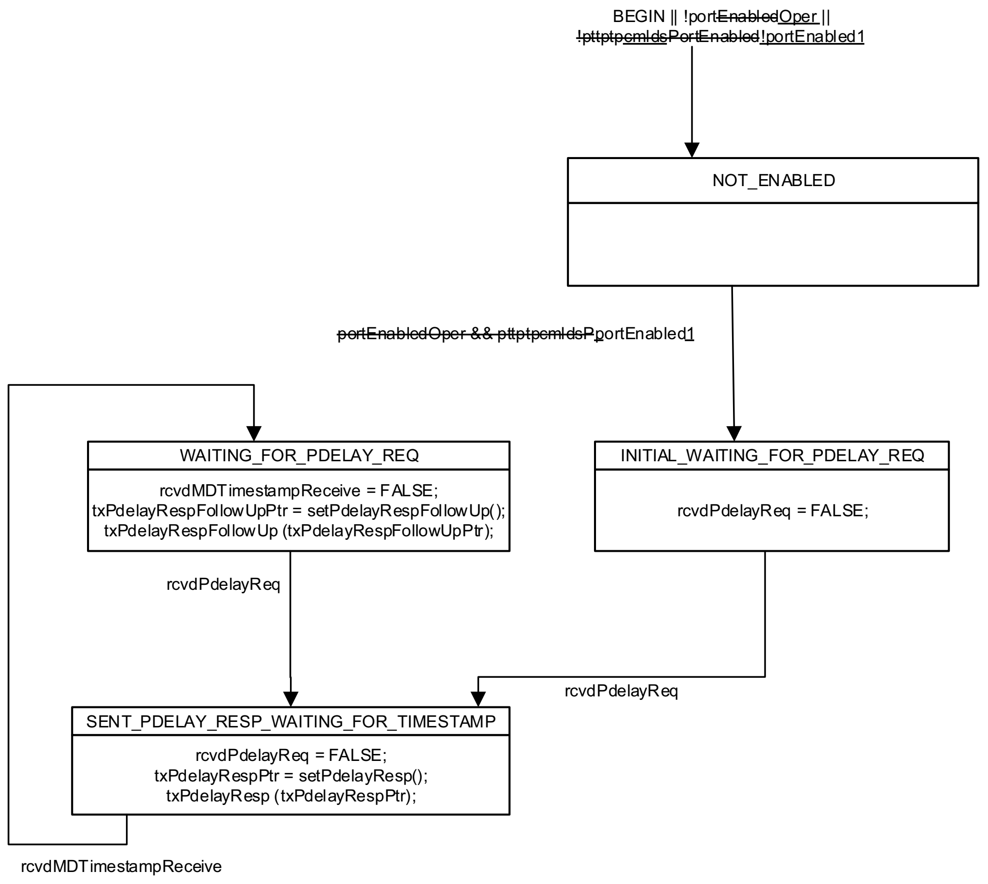

BEGIN || !port~~Enabled~~Oper ||
~~!pttptpcmldsPortEnabled~~!portEnabled1
NOT_ENABLED
~~portEnabledOper && pttptpcmldsP~~portEnabled1
WAITING_FOR_PDELAY_REQ
rcvdMDTimestampReceive = FALSE;
txPdelayRespFollowUpPtr = setPdelayRespFollowUp();
txPdelayRespFollowUp (txPdelayRespFollowUpPtr);
INITIAL_WAITING_FOR_PDELAY_REQ
rcvdPdelayReq = FALSE;
rcvdPdelayReq
rcvdPdelayReq
SENT_PDELAY_RESP_WAITING_FOR_TIMESTAMP
rcvdPdelayReq = FALSE;
txPdelayRespPtr = setPdelayResp();
txPdelayResp (txPdelayRespPtr);
rcvdMDTimestampReceive

Figure 11-9—MDPdelayResp state machine

<<Editor's note : In Figure 11-9, portEnabled is changed to portOper, and pttPortEnabled is changed to ptpPortEnabled.>>

11.2.19 ~~LinkDelay~~SyncIntervalSetting state machine

11.2.19.1 State machine variables

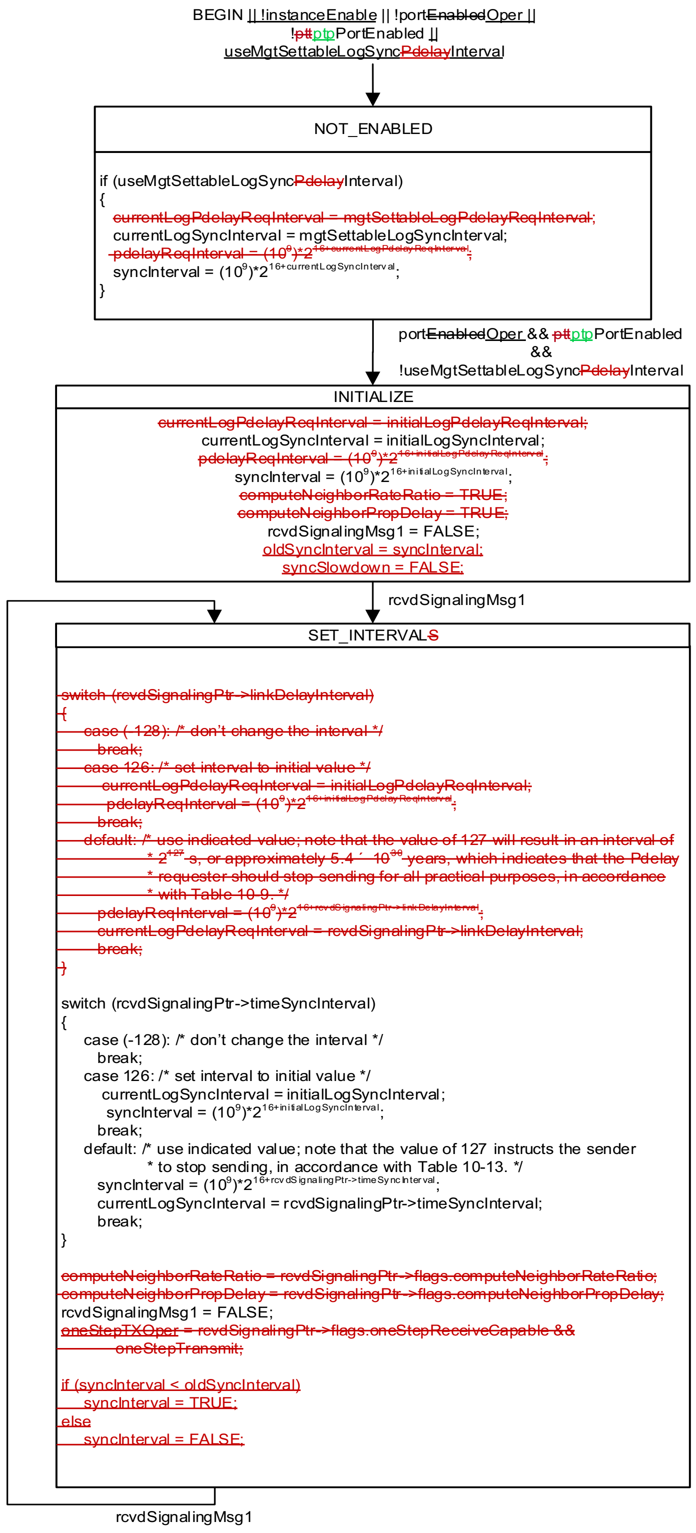
The following variables are used in the state diagram of 11.2.19.2:

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

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

11.2.19.2 State diagram

The ~~LinkDelay~~SyncIntervalSetting state machine shall implement the function specified by the state diagram in Figure 11-10, the local variables specified in 11.2.19.1, the messages specified in 10.6 and 11.4, the relevant global variables specified in 10.2.4 and 11.2.12, the relevant managed objects specified in 14.8, and the relevant timing attributes specified in 10.7 and 11.5. This state machine is responsible for setting the global variables that give the duration of the mean intervals between successive Sync ~~and successive Pdelay_Req~~ messages, both at initialization and in response to the receipt of a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3).


BEGIN || !instanceEnable || !port~~Enabled~~Oper ||
!~~ptt~~ptpPortEnabled ||
useMgtSettableLogSync~~Pdelay~~Interval
NOT_ENABLED
port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled
&&
!useMgtSettableLogSync~~Pdelay~~Interval
INITIALIZE
rcvdSignalingMsg1
SET_INTERVAL~~S~~
rcvdSignalingMsg1

```
NOT_ENABLED

if (useMgtSettableLogSync~~Pdelay~~Interval)
{
  ~~currentLogPdelayReqInterval = mgtSettableLogPdelayReqInterval;~~
  currentLogSyncInterval = mgtSettableLogSyncInterval;
  ~~pdelayReqInterval = $(10^9)*2^{16+currentLogPdelayReqInterval}$;~~
  syncInterval = $(10^9)*2^{16+currentLogSyncInterval}$;
}

INITIALIZE

~~currentLogPdelayReqInterval = initialLogPdelayReqInterval;~~
currentLogSyncInterval = initialLogSyncInterval;
~~pdelayReqInterval = $(10^9)*2^{16+initialLogPdelayReqInterval}$;~~
syncInterval = $(10^9)*2^{16+initialLogSyncInterval}$;
~~computeNeighborRateRatio = TRUE;~~
~~computeNeighborPropDelay = TRUE;~~
rcvdSignalingMsg1 = FALSE;
oldSyncInterval = syncInterval;
syncSlowdown = FALSE;

SET_INTERVAL~~S~~

~~switch (rcvdSignalingPtr->linkDelayInterval)~~
~~{~~
    ~~case (-128): /* don't change the interval */~~
      ~~break;~~
    ~~case 126: /* set interval to initial value */~~
        ~~currentLogPdelayReqInterval = initialLogPdelayReqInterval;~~
        ~~pdelayReqInterval = $(10^9)*2^{16+initialLogPdelayReqInterval}$;~~
      ~~break;~~
    ~~default: /* use indicated value; note that the value of 127 will result in an interval of~~
              ~~* $2^{127}$ s, or approximately 5.4 ´ $10^{30}$ years, which indicates that the Pdelay~~
              ~~* requester should stop sending for all practical purposes, in accordance~~
              ~~* with Table 10-9. */~~
      ~~pdelayReqInterval = $(10^9)*2^{16+rcvdSignalingPtr->linkDelayInterval}$;~~
      ~~currentLogPdelayReqInterval = rcvdSignalingPtr->linkDelayInterval;~~
      ~~break;~~
~~}~~

switch (rcvdSignalingPtr->timeSyncInterval)
{
    case (-128): /* don't change the interval */
      break;
    case 126: /* set interval to initial value */
        currentLogSyncInterval = initialLogSyncInterval;
        syncInterval = $(10^9)*2^{16+initialLogSyncInterval}$;
      break;
    default: /* use indicated value; note that the value of 127 instructs the sender
              * to stop sending, in accordance with Table 10-13. */
      syncInterval = $(10^9)*2^{16+rcvdSignalingPtr->timeSyncInterval}$;
      currentLogSyncInterval = rcvdSignalingPtr->timeSyncInterval;
      break;
}

~~computeNeighborRateRatio = rcvdSignalingPtr->flags.computeNeighborRateRatio;~~
~~computeNeighborPropDelay = rcvdSignalingPtr->flags.computeNeighborPropDelay;~~
rcvdSignalingMsg1 = FALSE;
~~oneStepTXOper = rcvdSignalingPtr->flags.oneStepReceiveCapable &&~~
        ~~oneStepTransmit;~~

if (syncInterval < oldSyncInterval)
    syncInterval = TRUE;
else
    syncInterval = FALSE;
```

Figure 11-10—~~LinkDelay~~SyncIntervalSetting state machine

11.2.20 LinkDelayIntervalSetting state machine

11.2.20.1 General

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

11.2.20.2 State machine variables

The following variables are used in the state diagram of 11.2.19.2:

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

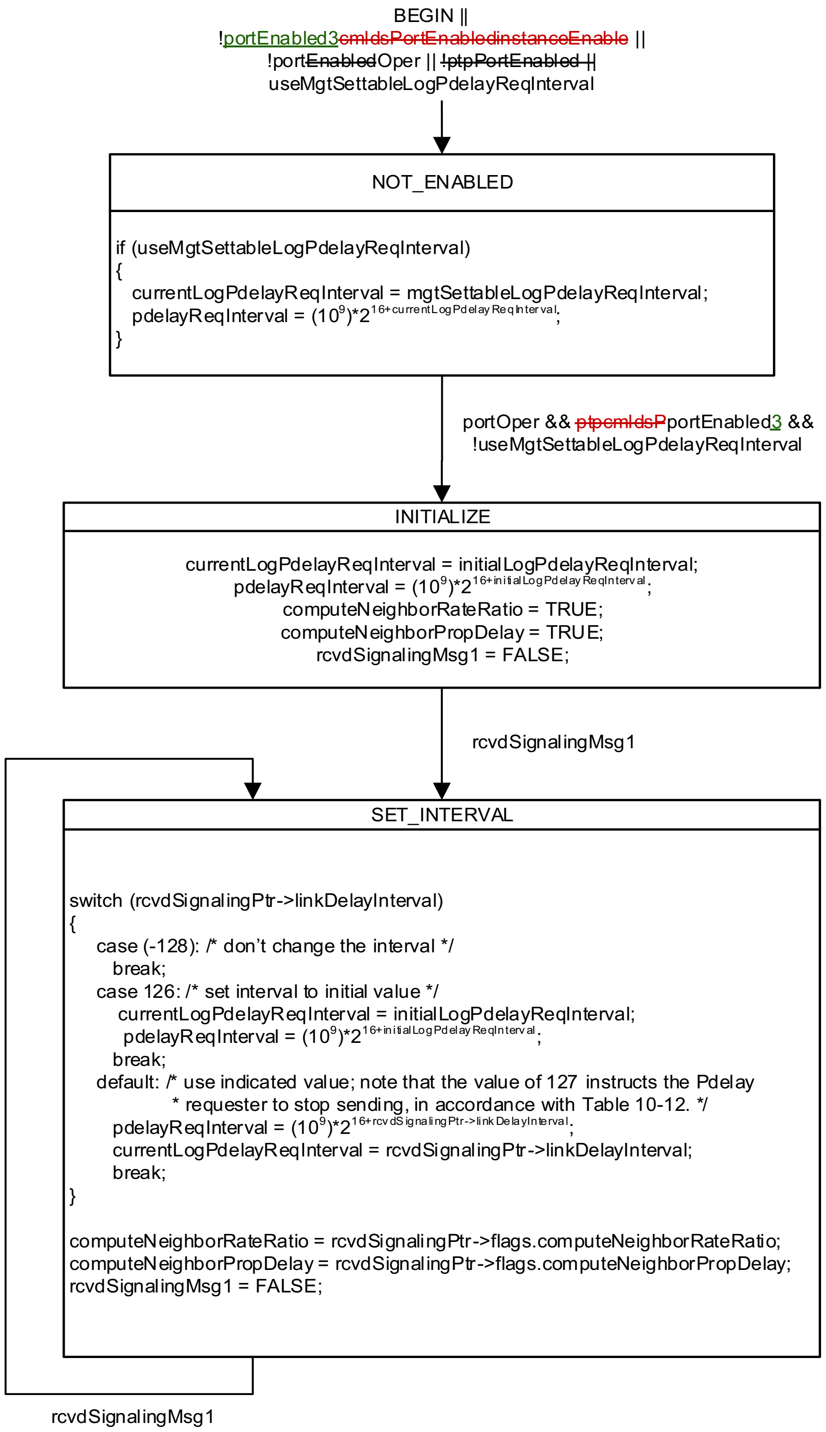
11.2.20.2.2 rcvdSignalingPtr: a pointer to a structure whose members contain the values of the fields of the received Signaling message that contains a Message Interval Request TLV (see 10.6.4.3).

11.2.20.2.3 portEnabled3: a Boolean variable whose value is equal to ptpPortEnabled (see 10.2.4.13) if this state machine is invoked by the instance-specific peer delay mechanism, and is equal to cmldsLinkPortEnabled (see 11.2.16.1) if this state machine is invoked by the CMLDS.

11.2.20.3 State diagram

The LinkDelayIntervalSetting state machine shall implement the function specified by the state diagram in Figure 11-11, the local variables specified in 11.2.20.2, the messages specified in 10.6 and 11.4, the relevant global variables specified in 10.2.4 and 11.2.12, the relevant managed objects specified in 14.8, and the relevant timing attributes specified in 10.7 and 11.5. This state machine is responsible for setting the global variables that give the duration of the mean interval between successive Pdelay_Req messages, both at initialization and in response to the receipt of a Signaling message that contains a Message Interval Request TLV (see 10.6.4.3).

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


BEGIN ||
!portEnabled3~~cmldsPortEnabledinstanceEnable~~ ||
!port~~Enabled~~Oper || ~~!ptpPortEnabled ||~~
useMgtSettableLogPdelayReqInterval
NOT_ENABLED
portOper && ~~ptpcmldsP~~portEnabled3 &&
!useMgtSettableLogPdelayReqInterval
INITIALIZE
rcvdSignalingMsg1
SET_INTERVAL
rcvdSignalingMsg1

NOT_ENABLED:

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

INITIALIZE:

```
currentLogPdelayReqInterval = initialLogPdelayReqInterval;
pdelayReqInterval = $(10^9)*2^{16+initialLogPdelayReqInterval}$;
computeNeighborRateRatio = TRUE;
computeNeighborPropDelay = TRUE;
rcvdSignalingMsg1 = FALSE;
```

SET_INTERVAL:

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

computeNeighborRateRatio = rcvdSignalingPtr->flags.computeNeighborRateRatio;
computeNeighborPropDelay = rcvdSignalingPtr->flags.computeNeighborPropDelay;
rcvdSignalingMsg1 = FALSE;
```

Figure 11-11—LinkDelayIntervalSetting state machine

11.2.21 OneStepTxOperSetting state machine

11.2.21.1 State machine variables

The following variables are used in the state diagram of 11.2.19.2:

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

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

11.2.21.2 State diagram

The OneStepTxOperSetting state machine shall implement the function specified by the state diagram in Figure 11-12, the local variables specified in 11.2.21.1, the messages specified in 10.6 and 11.4, the relevant global variables specified in 10.2.4 and 11.2.12, and the relevant managed objects specified in 14.8. This state machine is responsible for setting the relevant global variables and managed objects pertaining to one-step/two-step operation.

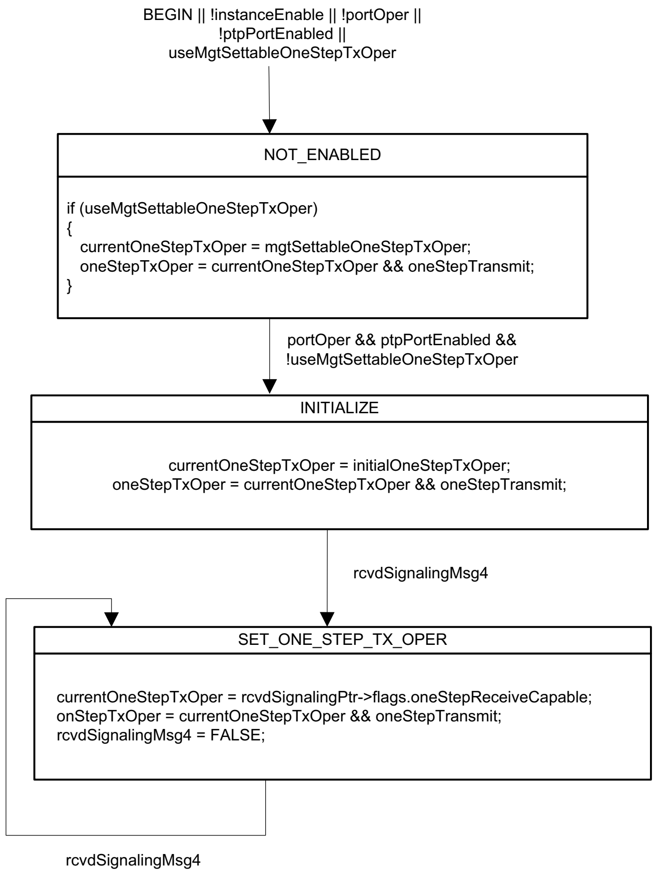

BEGIN || !instanceEnable || !portOper || !ptpPortEnabled || useMgtSettableOneStepTxOper
NOT_ENABLED
if (useMgtSettableOneStepTxOper)
{
currentOneStepTxOper = mgtSettableOneStepTxOper;
oneStepTxOper = currentOneStepTxOper && oneStepTransmit;
}
portOper && ptpPortEnabled && !useMgtSettableOneStepTxOper
INITIALIZE
currentOneStepTxOper = initialOneStepTxOper;
oneStepTxOper = currentOneStepTxOper && oneStepTransmit;
rcvdSignalingMsg4
SET_ONE_STEP_TX_OPER
currentOneStepTxOper = rcvdSignalingPtr->flags.oneStepReceiveCapable;
onStepTxOper = currentOneStepTxOper && oneStepTransmit;
rcvdSignalingMsg4 = FALSE;
rcvdSignalingMsg4

Figure 11-12—OneStepTxOperSetting state machine

11.3 Message attributes

11.3.1 General

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

11.3.2 Message types contained in each message class

11.3.2.1 Event message class

The event message class contains the following message types:

a) Sync: A Sync message contains time-synchronization information that originates at a ClockMaster entity. The appearance of a Sync message at the reference plane of the port corresponding to an MD entity is an event to which the LocalClock assigns a timestamp, the ~~<~~syncEventIngressTimestamp~~>~~ or ~~<~~syncEventEgressTimestamp~~>~~, based on the time of the LocalClock. The ~~<~~syncEventIngressTimestamp~~>~~ and ~~<~~syncEventEgressTimestamp~~>~~ are measured relative to the timestamp measurement plane; the MD entity corrects them for ingress and egress latencies, respectively (see 8.4.3). The Sync message is followed by a Follow_Up message containing synchronization information that is based in part on the sum of the ~~<~~syncEventEgressTimestamp~~>~~ and any egressLatency (see 8.4.3).
b) Pdelay_Req: A Pdelay_Req message is transmitted by an MD entity to another MD entity as part of the peer delay mechanism (see 11.2.15 and) to determine the delay on the link between them. The appearance of a Pdelay_Req message at the reference plane of the port of an MD entity is an event to which the LocalClock assigns a timestamp, the ~~<~~pdelayReqEventIngressTimestamp~~>~~ or ~~<~~pdelayReqEventEgressTimestamp~~>~~, based on the time of the LocalClock. The ~~<~~pdelayReqEventIngressTimestamp~~>~~ and ~~<~~pdelayReqEventEgressTimestamp~~>~~ are measured relative to the timestamp measurement plane; the MD entity corrects them for ingress and egress latencies, respectively (see 8.4.3).
c) Pdelay_Resp: A Pdelay_Resp message is transmitted by an MD entity to another MD entity in response to the receipt of a Pdelay_Req message. The Pdelay_Resp message contains the ~~<pdelay-req-ingress-timestamp>~~pdelayReqEventIngressTimestamp of the Pdelay_Req message that it is transmitted in response to. The appearance of a Pdelay_Resp message at the reference plane of the port of an MD entity is an event to which the LocalClock assigns a timestamp, the ~~<~~pdelayRespEventIngressTimestamp~~>~~ or ~~<~~pdelayRespEventEgressTimestamp~~>~~, based on the time of the LocalClock. The ~~<~~pdelayRespEventIngressTimestamp~~>~~ and ~~<~~pdelayRespEventEgressTimestamp~~>~~ are measured relative to the timestamp measurement plane; the MD entity corrects them for ingress and egress latencies, respectively (see 8.4.3). The Pdelay_Resp message is followed by a Pdelay_Resp_Follow_Up message containing the sum of the ~~<~~pdelayRespEventEgressTimestamp~~>~~ and any egressLatency (see 8.4.3).

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

11.3.2.2 General message class

The general message class contains the following message types:

a) Follow_Up: A Follow_Up message communicates the value of the ~~<~~syncEventEgressTimestamp~~>~~ for the associated Sync message.
b) Pdelay_Resp_Follow_Up: A Pdelay_Resp_Follow_Up message communicates the value of the ~~<~~PdelayRespEventEgressTimestamp~~>~~ for the associated Pdelay_Resp message.

General messages are not required to be timestamped.

11.3.3 VLAN tag

A frame that carries an IEEE 802.1AS message shall not have a VLAN tag nor a priority tag.

11.3.4 Addresses

The destination address of Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages shall be the reserved multicast address given in Table 11-2.

Table 11-2—Destination address for Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages

Destination address
01-80-C2-00-00-0E
NOTE—This address is taken from Table 8-1,Table 8-2, and Table 8-3, of IEEE Std 802.1Q-201~~1~~4.

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

11.3.5 Ethertype

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

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

Ethertype
$88F7_{16}$

11.3.6 Subtype

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

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

11.3.7 Source port identity

TheSync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages each contain a sourcePortIdentity field (see 11.4.2.6) that identifies the egress port (see 8.5) on which the respective message is sent.

11.3.8 Sequence number

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

Each Sync and Pdelay_Req message contains a sequenceId field, see 11.4.2.7, that carries the message sequence number. The sequenceId of a Sync message shall be one greater than the sequenceId of the previous Sync message sent by the transmitting port, subject to the constraints of the rollover of the UInteger16 data type used for the sequenceId field. The sequenceId of a Pdelay_Req message shall be one greater than the sequenceId of the previous Pdelay_Req message sent by the transmitting port, subject to the constraints of the rollover of the UInteger16 data type used for the sequenceId field.

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

a) Pdelay_Resp
b) Follow_Up
c) Pdelay_Resp_Follow_Up

For these exceptions, the sequenceId value is specified in 11.4.2.7, Table 11-7.

11.3.9 Event message timestamp point

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

11.4 Message formats

11.4.1 General

The PTP messages Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up shall each have a header, body, and if present, a suffix that contains one or more TLVs (see 10.6.2, 11.4.3, , 11.4.5, 11.4.6, 11.4.7, and Clause 14 of IEEE 1588-2008). Reserved fields shall be transmitted with all bits of the field 0 and ignored by the receiver, unless otherwise specified. The data type of the field shall be the type indicated in brackets in the title of each subclause.

~~This subclause~~Subclause 11.4.4.3 defines the Follow_Up information TLV, which is carried by the Follow_Up message if the corresponding Sync message is two-step (i.e., twoStepFlag of the Sync message, see 10.6.2.2.8, is TRUE) and by the Sync message if the message is one-step (i.e., twoStepFlag is FALSE).~~(see 11.4.4.3).~~ The Follow_Up information TLV ~~shall be~~is the first TLV of ~~a~~the Follow_Up message or Sync message. If a time-aware system cannot parse a TLV, it shall ignore it and attempt to parse the next TLV (see 14.1 of IEEE Std 1588-2008).

<<Editor's note: The shalls related to the Follow Up information TLV being the first TLV of the respective message are not needed here, because they are in 11.4.3.2.2 and 11.4.4.2.2.>>

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

11.4.2 Header

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

11.4.2.1 messageType (Enumeration4)

The value indicates the type of the message, as defined in Table 11-4.

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

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

11.4.2.2 messageLength (UInteger16)

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

Table 11-4—Values for messageType field

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

NOTE—See 10.6.2.2.5 for an example.

11.4.2.3 domainNumber (UInteger8)

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

11.4.2.4 flags (Octet2)

The value of the bits of the array are defined in Table 10-6. ~~For message types where the bit is not defined in Table 11-5, the value of the bit is set to FALSE.~~

Table 11-5—~~Values of flag bits~~

~~Octet~~	~~Bit~~	~~Message type~~	~~Name~~	~~Value~~
~~0~~	~~2~~	~~Sync, Pdelay_Resp~~	~~twoStepFlag~~	~~Reserved as TRUE, ignored on reception~~

11.4.2.5 correctionField (Integer64)

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

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

11.4.2.6 sourcePortIdentity (PortIdentity)

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

11.4.2.7 sequenceId (UInteger16)

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

Table 11-6—Value of ~~correction field~~correctionField

Message type	Value
Follow_Up Sync (sent by a one-step port, see 11.1.3 and 11.2.12.9)	Corrections for fractional nanoseconds (see 10.2.8 and Figure 10-5), difference between preciseOriginTimestamp~~-~~field (if sent by a two-step port) or originTimestamp field (if sent by a one-step port) ~~field~~ and current synchronized time (see 11.2.14.2.3 and Figure 11-7), and asymmetry corrections (see 8.3, 11.2.13.2.1, and 11.2.14.2.3; the quantity delayAsymmetry is used in the computation of upstreamTxTime in 11.2.13.2.1, and upstreamTxTime is used in computing an addition to the ~~correction field~~correctionField in 11.2.14.2.3)
~~Pdelay_Req,~~ Pdelay_Resp, Pdelay_Resp_Follow_Up	Corrections for fractional nanoseconds (see Figure 11-8 and Figure 11-9)
Sync (sent by a two-step port), Pdelay_Req, Announce, Signaling	The value is 0 (see 10.6.2.2.9)
NOTE—IEEE Std 1588-2008 describes asymmetry corrections for the Pdelay_Req and Pdelay_Resp messages. However, the peer delay mechanism computes the mean propagation delay. In the case here where the communication path is a full-duplex, point-to-point link, these corrections cancel in the mean propagation delay computation and therefore are not needed.	

Table 11-7—References for sequenceId value exceptions

Message type	Reference
Follow_Up	See 11.2.14 and Figure 11-7
Pdelay_Resp	See and Figure 11-9
Pdelay_Resp_Follow_Up	See and Figure 11-9

11.4.2.8 control (UInteger8)

The value is as specified in Table 11-8.

Table 11-8—Value of control field

Message type	Value
Sync	0x0
Follow_Up	0x2
Announce, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up	0x5

11.4.2.9 logMessageInterval (Integer8)

For Sync and Follow_Up messages, the value is the value of currentLogSyncInterval, see 10.2.4.4 and 10.7.2.3. For Pdelay_Req messages, the value is the value of currentLogPdelayReqInterval. For Pdelay_Resp and Pdelay_Resp_Follow_Up messages, the value is ~~reserved~~transmitted as 0x7F and ignored on reception.

11.4.3 Sync

11.4.3.1 General Sync message specifications

If the twoStep flag of the PTP common header (see Table 10-6) of the Sync message is TRUE, the fields of the Sync message shall be as specified in Table 11-9. 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 11-10 and 11.4.3.2 and its subclauses.

~~The fields of the Sync message shall be as specified in Table 11-9.~~

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

Bits								Octets	Offset
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
header (see 11.4.2)								34	0
reserved								10	34

Table 11-10—Sync message fields if twoStep flag is FALSE

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

11.4.3.2 Sync message field specifications if twoStep flag is FALSE

11.4.3.2.1 originTimestamp (Timestamp)

The value of the originTimestamp field is the sourceTime of the ClockMaster entity of the grandmaster, when the Sync message was sent by that grandmaster, with any fractional nanoseconds truncated (see 10.2.8).

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

11.4.3.2.2 Follow_Up information TLV

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

<<Editor's note: For now, the name "Follow_Up information TLV" is retained, even though with the introduction of one-step operation the TLV is carried by Sync or Follow_Up.>>

11.4.4 Follow_Up

11.4.4.1 General Follow_Up message specifications

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

Table 11-11—Follow_Up message fields

Bits								Octets	Offset
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
header (see 11.4.2)								34	0
preciseOriginTimestamp								10	34
Follow_Up information TLV								32	44

11.4.4.2 Follow_Up message field specifications

11.4.4.2.1 preciseOriginTimestamp (Timestamp)

The value of the preciseOriginTimestamp field is the sourceTime of the ClockMaster entity of the grandmaster, when the associated ~~time synchronization event~~Sync message was sent by that grandmaster, with any fractional nanoseconds truncated (see 10.2.8).

The sum of the ~~correction field~~correctionFields in the Follow_Up and associated ~~time synchronization event~~ Sync messages, added to the preciseOriginTimestamp field of the Follow_Up message, is the value of the synchronized time corresponding to the ~~<~~syncEventEgressTimestamp~~>~~ at the time-aware system that sent the associated ~~time synchronization event~~Sync message, including any fractional nanoseconds.

11.4.4.2.2 Follow_Up information TLV

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

11.4.4.3 Follow_Up information TLV

11.4.4.3.1 General

The fields of the Follow_Up information TLV shall be as specified in Table 11-12 and in 11.4.4.3.2 through 11.4.4.3.9. This TLV is a standard organization extension TLV for the Follow_Up message, as specified in 14.3 of IEEE Std 1588-2008.

Table 11-12—Follow_Up information TLV

Bits								Octets	Offset
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
tlvType								2	0
lengthField								2	2
organizationId								3	4
organizationSubType								3	7
cumulativeScaledRateOffset								4	10
gmTimeBaseIndicator								2	14
lastGmPhaseChange								12	16
scaledLastGmFreqChange								4	28

11.4.4.3.2 tlvType (Enumeration16)

The value of the tlvType field is 0x3.

NOTE—This is the value that indicates the TLV is a vendor and standard organization extension TLV, as specified in 14.3.2.1 of IEEE Std 1588-2008. The value is specified there as ORGANIZATION_EXTENSION, whose value is 0x3.

11.4.4.3.3 lengthField (UInteger16)

The value of the lengthField is 28.

11.4.4.3.4 organizationId (Octet3)

The value of organizationId is 00-80-C2.

11.4.4.3.5 organizationSubType (Enumeration24)

The value of organizationSubType is 1.

11.4.4.3.6 cumulativeScaledRateOffset (Integer32)

The value of cumulativeScaledRateOffset is equal to (rateRatio – 1.0) × (2^{41}), truncated to the next smaller signed integer, where rateRatio is the ratio of the frequency of the grandMaster to the frequency of the LocalClock entity in the time-aware system that sends the message.

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

11.4.4.3.7 gmTimeBaseIndicator (UInteger16)

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

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

11.4.4.3.8 lastGmPhaseChange (ScaledNs)

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

11.4.4.3.9 scaledLastGmFreqChange (Integer32)

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

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

11.4.5 Pdelay_Req message

<<Editor's note : Subclauses 11.4.3 and 11.4.4 are entitled "Sync" and "Follow_Up", respectively. However, subclauses 11.4.5, 11.4.6, and 11.4.7 are entitled "Pdelay_Req message", "Pdelay_Resp message", and "Pdelay_Resp_Follow_Up message", respectively. Should we make the titles consistent with respect to the use of the word "message", i.e., either add it to 11.4.3 and or remove it from 11.4.5, 11.4.6, and 11.4.7?>>

The fields of the Pdelay_Req message shall be as specified in Table 11-13.

Table 11-13—Pdelay_Req message fields

Bits								Octets	Offset
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
header (see 11.4.2)								34	0
reserved								10	34
reserved								10	44

11.4.6 Pdelay_Resp message

11.4.6.1 General Pdelay_Resp message specifications

The fields of the Pdelay_Resp message shall be as specified in Table 11-14 and 11.4.6.2 and its subclauses.

Table 11-14—Pdelay_Resp message fields

Bits								Octets	Offset
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
header (see 11.4.2)								34	0
requestReceiptTimestamp								10	34
requestingPortIdentity								10	44

11.4.6.2 Pdelay_Resp message field specifications

11.4.6.2.1 requestReceiptTimestamp (Timestamp)

The value is the seconds and nanoseconds portion of the ~~<~~pdelayReqEventIngressTimestamp~~>~~ of the associated Pdelay_Req message, see .

11.4.6.2.2 requestingPortIdentity (PortIdentity)

The value is the value of the sourcePortIdentity field of the associated Pdelay_Req message, see .

11.4.7 Pdelay_Resp_Follow_Up message

11.4.7.1 General Pdelay_Resp_Follow_Up message specifications

The fields of the Pdelay_Resp_Follow_Up message shall be as specified in Table 11-15 and 11.4.7.2 and its subclauses.

Table 11-15—Pdelay_Resp_Follow_Up message fields

Bits								Octets	Offset
~~8~~7	~~7~~6	~~6~~5	~~5~~4	~~4~~3	~~3~~2	~~2~~1	~~1~~0		
header (see 11.4.2)								34	0
responseOriginTimestamp								10	34
requestingPortIdentity								10	44

11.4.7.2 Pdelay_Resp_Follow_Up message field specifications

11.4.7.2.1 responseOriginTimestamp (Timestamp)

The value is the seconds and nanoseconds portion of the ~~<~~pdelayRespEventEgressTimestamp~~>~~ of the associated Pdelay_Resp message, see .

11.4.7.2.2 requestingPortIdentity (PortIdentity)

The value is the value of the sourcePortIdentity field of the associated Pdelay_Req message, see .

11.5 Protocol timing characterization

11.5.1 General

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

11.5.2 Message transmission intervals

11.5.2.1 General interval specification

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

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

11.5.2.2 Pdelay_Req message transmission interval

When useMgtSettableLogPdelayReqInterval (see 14.13.12) is FALSE, t~~T~~he initialLogPdelayReqInterval specifies the following:

a) The mean time interval between successive Pdelay_Req messages sent over a link when the port is initialized, and
b) The value the mean time interval between successive Pdelay_Req messages is set to when a message interval request TLV is received with the linkDelayIntervalField set to 126 (see 11.2.19).

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 11.2.19). A port may support other values, except for the reserved values –128 through –125, inclusive, and 124 through 126, inclusive. A port shall ignore requests (see 11.2.19) for unsupported values. The initialLogPdelayReqIntervaland currentLogPdelayReqInterval are per-port attributes.

When useMgtSettableLogPdelayReqInterval is TRUE, currentLogSyncInterval is set equal to mgtSettableLogPdelayReqInterval (see 14.13.13) and initialLogPdelayReqInterval is ignored.

NOTE 1—If useMgtSettableLogPdelayReqInterval is FALSE, t~~T~~he 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 ~~may~~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 linkDelayInterval is 126, see 10.6.4.3.6.

NOTE 2—A port that requests (using a Signaling message that contains a message interval request TLV, see 10.6.4 and 11.2.19) that the port at the other end of the attached link set its currentLogPdelayReqInterval to a specific value can determine if the request was honored by examining the logMessageInterval field of subsequent received Pdelay_Req messages.

NOTE 3—The MDPdelayReq state machine ensures that the times between transmission of successive Pdelay_Req messages, expressed in s, are not smaller than $2^{\text{currentLogPdelayReqInterval}}$. This is consistent with IEEE Std 1588, which requires that the logarithm to the base 2 of the mean value of the interval in seconds between Pdelay_Req message transmissions is no smaller than the interval computed from the value of the portDS.logMinPdelayReqInterval member of the data set of the transmitting PTP Instance. The sending of Pdelay_Req messages is governed by the LocalClock, and not the synchronized time (i.e., the estimate of the grandmaster time). Since the LocalClock frequency can be slightly larger than the grandmaster frequency (e.g., by 100 ppm, which is the specified frequency accuracy of the LocalClock, see Annex 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.

11.5.2.3 Sync message transmission interval default value

The default value of initialLogSyncInterval (see 10.7.2.3) is –3. Every port supports the value 127; the port does not send Sync messages when currentLogSyncInterval has this value (see 11.2.19). A port may support other values, except for the reserved values –128 through –125, inclusive, and 124 through 126, inclusive. A port ignores requests (see 11.2.19) for unsupported values.

NOTE—A port that requests (using a Signaling message that contains a message interval request TLV, see 10.6.4 and 11.2.19) that the port at the other end of the attached link set its currentLogSyncInterval to a specific value can determine if the request was honored by examining the logMessageInterval field of subsequent received Sync messages.

11.5.3 allowedLostResponses

The variable allowedLostResopnse (see 11.2.12.4) is the number of Pdelay_Req messages for which a valid response is not received, above which a port is considered to not be exchanging peer delay messages with its neighbor. The default value of allowedLostResponses shall be 9~~3~~. The range shall be 1 - 255.

11.5.4 allowedFaults

The variable allowedFaults (see 11.2.12.5) is the number of faults, i.e., the number of

a) instances where the computed mean propagation delay, neighborPropDelay (see 10.2.4.8), exceeds the threshold, neighborPropDelayThresh (see 11.2.12.7), and/or
b) instances where the computation of neighborRateRatio is invalid (see 11.2.17.2.11)

above which asCapableAcrossDomains is set to FALSE, i.e., the port will be considered to not be capable of interoperating with its neighbor via the 802.1AS protocol (see 10.2.4.1). The default value of allowedFaults shall be 9. The range shall be 1 - 255.

12. Media-dependent layer specification for IEEE 802.11 links

<<Editor's note: This editor's note pertains to comment #79 against D2.0.

A high-level description of the primitives used for FTM has been added to 12.1 and 12.1.1. One of the main differences between the operation of FTM and TM is that in FTM the initiation of the sending of time-synchronization information occurs at the slave, via the MLME-FINETIMINGM SMTRQ.request; this primitive causes an initial FTM request frame to be sent from slave to master that contains respective FTM parameters. This in turn causes the FTM frames to be sent from master to slave. In contrast, in TM the TM frames are sent from master to slave as a result of MLME-TIMINGMSMT.request primitives that originate at the master. This aspect of FTM differs not only from TM, but also from the fundamental way that the sending of timing information is initiated in IEEE 1588 - 2008 in general, and also in IEEE 802.1AS-211. All other media in IEEE 802.1AS (i.e., other than 802.11) operate in a manner similar to TM. Since the architecture of IEEE 802.1AS is divided into media-independent and media-dependent layers, and since the aspect of FTM described here (i.e., initiation of the sending of timing information by the slave) is specific to FTM, it is desirable to specify this aspect of FTM in the media-dependent layer and not change the media-independent layer. The rest of this editor's note describes, at high-level, an initial approach on how this will be done (this will require modifications to 12.1.2 and subsequent subclauses of Clause 12). This description represents the thinking of the editors at the time D3.0 was completed. It is planned to bring a more detailed presentation to the May 2016 IEEE 802.1 TSN TG meeting, to describe the approach in more detail. It is possible that, as that presentation is prepared, some of the ideas described below may be refined or modified.

At present (i.e., for TM), the 802.11 media-dependent layer has a single master state machine and a single slave state machine. The master state machine receives an MDSyncSend structure from the media-independent layer. This structure contains time-synchronization information received from upstream, which is processed in the MD layer. The structure also causes a TM frame to be sent. The master state machine also receives a confirm primitivfe, which is triggered by the receipt of an ACK from the slave. The master processes the confirm and then waits for the next MDSyncSend structure from the higher layer. The slave state machine receives an indication indicating that a TM frame has been received from from the master; it processes the indication and sends information from the higher layer.

In FTM, the initial FTM frame is sent by the slave. Since this behavior is media-dependent, the controlling of this will be added to the slave state machine (i.e., at the media-dependent layer). The slave state machine will have logic that triggers the sending of the MLME-FINETIMINGM SMTRQ.request, which will contain parameters that control the sending by the master of a burst of FTM frames (these parameters will be discussed more below). The slave will then wait to receive each FTM frame of the burst; on receipt of each frame, it will process the frame in a manner similar to the processing in TM and send the resulting information to the higher layer (just as it does now). After receiving NB such frames (where NB is the number of frames in the burst; this is one of the FTM parameters (see below)), the slave will go to sleep for a user-specified (via management) time interval (this interval could be 0 if desired) and then send a new initial FTM request to start the process again.

The master state machine will be somewhat more complex, because in FTM the master is receiving time-synchronization information that originates upstream, via the MDSyncSend structure, but the sending of FTM frames is controlled by the receipt of the initial FTM request from the slave. Therefore, the current master state machine will be modified so that, in FTM, receipt of an MDSyncSend structure causes the information in this structure to be processed and saved, but no FTM frames are sent. A new, additional master state machine will be constructed that receives the initial FTM request and then sends NB FTM frames (but waiting for the confirm after each one). When an FTM frame is sent, the information contained in the frame is computed using the current information computed from the most recently received MDSyncSend structure.

The parameters sent in in the initial FTM request contain, among other things, information that determines the average rate at which timing information is sent from master to slave. The most important parameters are the burst duration, min delta, and number of frames per burst (we will assume a single burst per session and ASAP=1. The parameters are described in detail in IEEE 802.11-RevMC (the latest draft, as of the preparation of 802.1AS-Rev/D3.0, is D5.3). 802.1AS-Rev will not reproduce material in 802.11-RevMC in detail; rather, 802.1AS will describe the parameter values for fixed parameters, and indicate which paramters are settable by the user for purposes of the slave requesting a desired rate. There will likely be an informative annex that describes how one might

choose the parameter values to request a particular rate. Note that in 802.11 FTM the slave can request a rate (i.e., request certain parameters), but that does not mean the master will actually give that rate.

For now, this editor's note is retained for background information for reviewers of this draft, as not all reviewers will necessarily be familiar with the details of FTM. The TSN TG should decide when this editor's note should be removed.>>

12.1 Overview

Accurate synchronized time is distributed across a domain through time measurements between adjacent time-aware systems in a ~~bridged LAN~~packet network. Time is communicated from the root of the clock spanning tree (i.e., the grandmaster) to the leaves of the tree, by recursively propagating time from a leaf-facing "master" port to some number of root-facing "slave" ports in devices at the next level of the tree through measurements made across the links connecting the devices. While the time semantics are consistent across the time-aware ~~bridged LAN~~packet network, the method for communicating synchronized time from a master port to the immediate downstream link partner varies depending on the type of link interconnecting the two ~~stations/Bridges~~systems.

This clause specifies the interface primitives and state machines that provide accurate synchronized time across wireless IEEE 802.11 links as part of a ~~bridged LAN~~packet network. This clause builds upon time measurement features defined in IEEE Std 802.11, and makes no distinction between stations with an Access Point function and stations without an Access Point function.

12.1.1 IEEE802.11 timing measurement procedure

IEEEStd 802.11 defines a family of wireless measurements, including~~-~~ both "timing measurement~~,~~" and "fine timing measurement," which captures timestamps~~s~~ of the transmit time and receive time of a round-trip message exchange between associated WLAN stations.

In contrast to the protocol defined for full-duplex point-to-point links, this clause does not define any new frames nor the transmission of any frames. Rather, it makes use of a MAC Layer Management Entity (MLME) interface, which causes the IEEE 802.11 layer to not only take timestamps of measurement frames as they are transmitted and received, but to also *generate* and *consume* the measurement frames, all within the IEEE 802.11 MLME layer, and then to provide timestamp information from the MLME to this media-dependent layer through a set of well-defined service primitives. However, as an aid to the reader, the protocol and frames used by the IEEE 802.11 MLME for ~~time~~both timing measurement and fine timing measurement are described briefly as follows and illustrated in Figure 12-1 and Figure 12-2, respectively.

Both ~~Time~~timing measurement~~-~~and fine timing measurement are~~is~~ accomplished through a round-trip frame exchange. For timing measurement, t~~T~~he first frame of the round-trip measurement~~ (the "request" frame)~~ is generated by the master within the IEEE 802.11 MLME when~~-~~the MLME-TIMINGMSMT.request primitive is invoked~~ by the requesting station~~. For fine timing measurement, an initial fine timing measurement request frame is generated by the slave within the IEEE 802.11 MLME when the MLME-FINETIMINGMSMTRQ.request primitive is invoked. After this frame is successfully received by the master, the first frame of the round-trip measurement is generated by the master within the IEEE 802.11 MLME when the MLME-FINETIMINGMSMT.request primitive is invoked. As defined by IEEE Std 802.11, upon receipt of the resulting timing measurement or fine timing measurement frame ~~unicast (action)~~ frame, the slave~~receiving~~ station transmits an IEEE 802.11 ACK control frame to the ~~requesting~~master station. Four timestamps are captured during this two-frame exchange, as follows:

a) *t1* is when (in the master~~requesting~~ station's time base) the request frame is transmitted
b) *t2* is when (in the slave~~responding~~ station's time base) the request frame is received
c) *t3* is when (in the slave~~responding~~ station's time base) the ACK control frame is transmitted

d) *t4* is when (in the master~~requesting~~ station's time base) the ACK control frame is received

When the master~~requester~~ sends ~~initiates~~ either a fine timing measurement or a timing measurement frame~~request~~, it passes the *t1* and *t4* timestamps (and other end-to-end synchronization information) along with FollowUp information from the previous measurement to the slave~~responder~~. A pair of tokens is~~are~~ passed in each timing or fine timing measurement frame~~request~~, one to identify the current measurement and the other to allow the slave~~responder~~ to associate the timestamp information with the previous measurement.

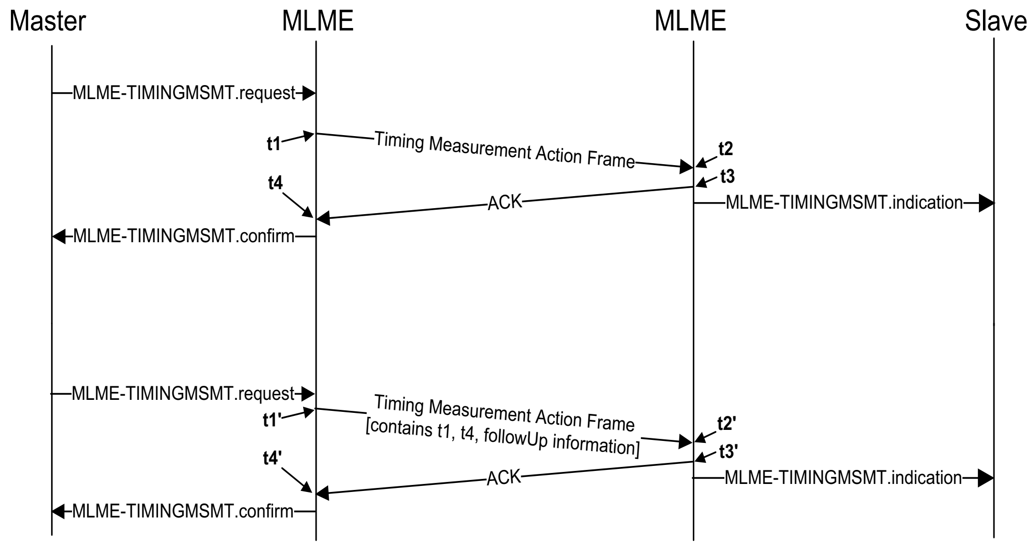

Master
MLME
MLME
Slave
MLME-TIMINGMSMT.request
t1
Timing Measurement Action Frame
t2
t3
t4
ACK
MLME-TIMINGMSMT.indication
MLME-TIMINGMSMT.confirm
MLME-TIMINGMSMT.request
t1'
Timing Measurement Action Frame [contains t1, t4, followUp information]
t2'
t3'
t4'
ACK
MLME-TIMINGMSMT.indication
MLME-TIMINGMSMT.confirm

Figure 12-1—Timing measurement procedure for IEEE 802.11 links

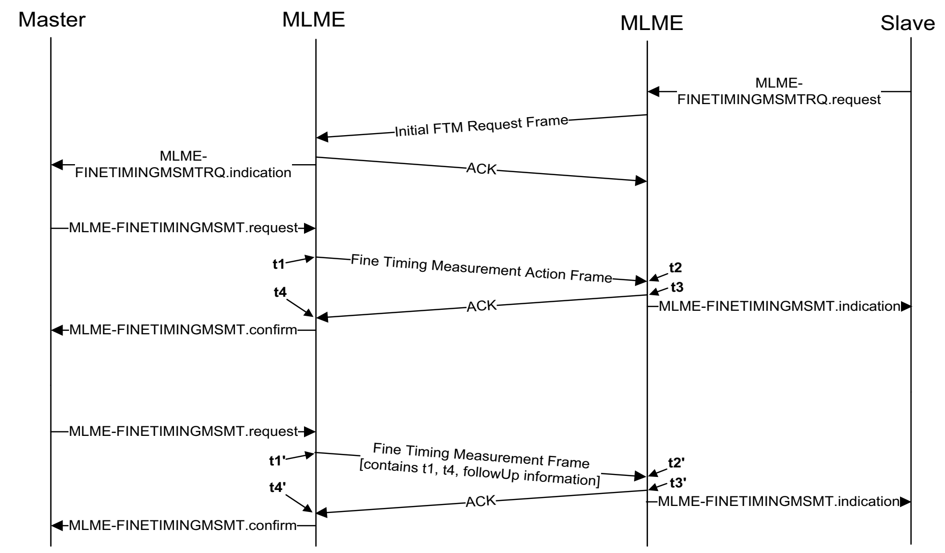

Master
MLME
MLME
Slave
MLME-FINETIMINGMSMTRQ.request
Initial FTM Request Frame
MLME-FINETIMINGMSMTRQ.indication
ACK
MLME-FINETIMINGMSMT.request
t1
Fine Timing Measurement Action Frame
t2
t3
t4
ACK
MLME-FINETIMINGMSMT.indication
MLME-FINETIMINGMSMT.confirm
MLME-FINETIMINGMSMT.request
t1'
Fine Timing Measurement Frame [contains t1, t4, followUp information]
t2'
t3'
t4'
ACK
MLME-FINETIMINGMSMT.indication
MLME-FINETIMINGMSMT.confirm

Figure 12-2—Fine timing measurement procedure for IEEE 802.11 links

Note that, unlike point-to-point full-duplex ports, IEEE 802.11 ports do not compute the link delay measurements in both directions since only a port in the Slave state makes use of that information. As a result, a port that transitions from the Master state to the Slave state (e.g., due to selection of a new grandmaster), ~~may~~might collect a number of link delay measurements and perform averaging or other filtering before achieving the desired accuracy.

12.1.2 Layering for IEEE 802.11 links

The *media-dependent* (MD) entity is tailored to the link technology and is responsible for translating the PortSync entity's media-independent actions to media-dependent PDUs or primitives as necessary for communicating synchronized time from the master port over the link to a single slave port. In the case of an IEEE 802.11 link, this one-to-one relationship between the MD entities of the master and slave implies that if the one physical IEEE 802.11 port is associated with multiple stations, each association requires its own instantiation of the IEEE 802.1AS PortSync entity and MD entity. The MLME-TIMINGMSMT and MLME-FINETIMINGMSMT service primitives defined in IEEE Std 802.11 and Draft P802.11-REVmc D5.3 are used to perform timing measurements and fine timing measurements, respectively, between a master IEEE 802.11 station and associated IEEE 802.11 slave station. Figure 12-3 illustrates how the MD entity interacts with the higher and lower layers.

<<Editor's note: Draft P802.11-REVmc_D6.0 is the latest draft of this document as of D4.0 of the current document. This reference will be updated to an approved, published document prior to sponsor ballot.>>

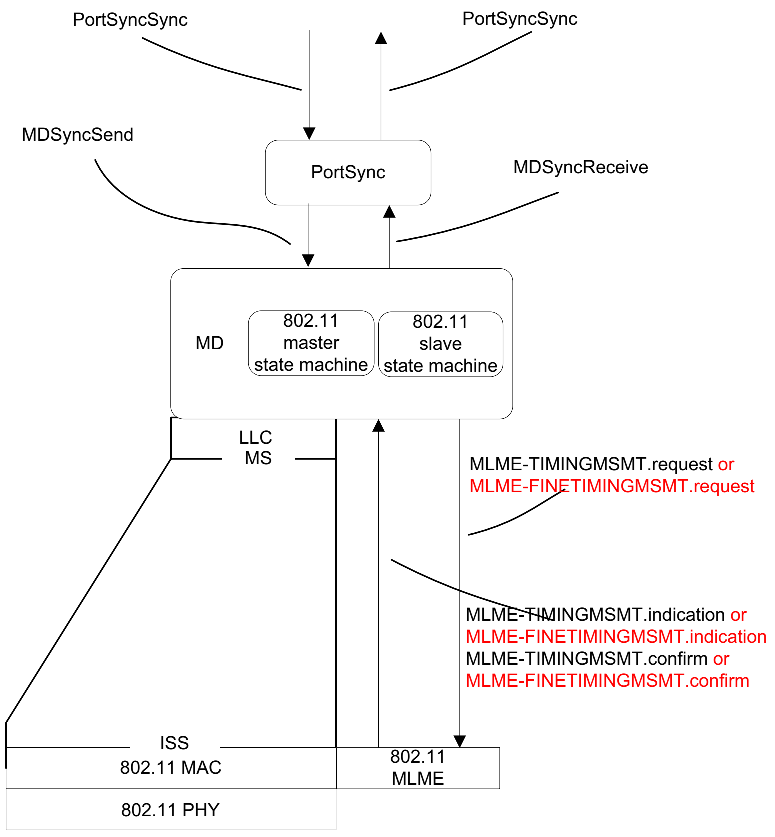

PortSyncSync
PortSyncSync
MDSyncSend
PortSync
MDSyncReceive
MD
802.11 master state machine
802.11 slave state machine
LLC
MS
MLME-TIMINGMSMT.request or
MLME-FINETIMINGMSMT.request
MLME-TIMINGMSMT.indication or
MLME-FINETIMINGMSMT.indication
MLME-TIMINGMSMT.confirm or
MLME-FINETIMINGMSMT.confirm
ISS
802.11 MAC
802.11 MLME
802.11 PHY

Figure 12-3—Media-dependent and lower entities in stations with IEEE 802.11 links

12.2 Messages

All media-dependent frames are generated and consumed by the lower-layer IEEE 802.11 MLME and thus none are defined here. Also, since the IEEE 802.11 event messages are timestamped by the MAC/PHY, the timestamp point is defined in IEEE_Std 802.11 as well. Media-independent messages, i.e., Announce and Signaling messages, are transmitted using the unicast address of the WLAN station instead of the group address defined in 10.5.3.

12.3 Determination of timing measurement and fine timing measurement capability

The bits of the per-port global variable tmFtmSupport (see 12.5.1.5) shall be set as indicated in Table 12-1.

Table 12-1 — Values of bits of tmFtmSupport

Bit	Value
0	TRUE if (a) the port supports timing measurement, and (b) the timing measurement bit in the Extended Capabilities information element defined in Table 8-103 of IEEE Std 802.11-2012 indicates that the peer 802.11 station is capable of participating in the timing measurement protocol FALSE otherwise
1	TRUE if (a) the port supports fine timing measurement, and (b) the fine timing measurement responder and initiator bits in the Extended Capabilities information element defined in Table 9-134 of Draft P802.11REVmc_D5.3 indicate that the peer 802.11 station is capable of participating in the fine timing measurement protocol FALSE otherwise
2 - 7	Reserved as FALSE

12.4 Determination of asCapable

The per-port, per-domain instance of the global variable asCapable ~~(~~(see 10.2.4.1) shall be set to TRUE if the following conditions hold:

a) The value of tmFtmSupport is not zero,
b) neighborGptpCapable is TRUE ~~and the value of domainNumber is not zero~~, and
c) in the case of a master port that is using FTM, the port can support (i.e., grant) the parameters request by the slave in the initial FTM request frame.

If the value of tmFtmSupport is not zero, the value of neighborGptpCapable is FALSE, and the value of domainNumber is zero, asCapable can be set to TRUE. In all other instances, asCapable shall be set to FALSE.

<<Editor's note: Condition (c) is added because, in the case of FTM, the burst of frames is sent by the master only if it can support the parameters requested by the slave. Condition (c) does not apply to the slave because the FTM mechansim does not provide for the master to first send the granted parameter values back to the slave and then send the FTM frames; rather, it simply sends an ACK to the initial FTM request (for which there is no indication primitive at the slave) and then either sends or does not send the burst, depending on whether the parameters requested are or are not supported, respectively. If the parameters are not supported, the slave simply times out and makes a new request.

It certainly could be possible that master and slave both support FTM, but the master is not able to grant the parameters requested by the slave. Right now, the logic of the state machines would set asCapable to FALSE in this instance. It might seem that the ports could try to use TM instead, if they both support TM. However, there seems to be no mechansim for this right now. If there are no comments on this, this editor's note will be deleted.>>

NOTE—The above conditions ensure backward compatibility with the 2011 edition of this standard. A time-aware system that is compliant with the 2011 edition of this standard will not process the gPTP capable TLV, and asCapable will be determined as specified in the 2011 edition. A time-aware system compliant with the current edition of this standard that is attached, via an 802.11 link, to a node compliant with the 2011 edition of this standard will not receive Signaling messages that contain the gPTP capable TLV and will not set neighborGptpCapable to TRUE; however, the first sentence immediately after (b) ensures that asCapable for this port and domain (i.e., domain 0) will still be set in a manner consistent with that of the 2011 edition of this standard.

<<Editor's note: The above is edited in accordance with discussion of slide 18 of the presentation from the November 2015 TSN TG meeting "Aspects of Multiple Domains in 802.1AS-Rev and

Backward Compatibility" (see http://www.ieee802.org/1/files/public/docs2015/as-garner-multiple-domains-aspects-1115-v1.pdf). The TSN TG should review this to ensure that the behavior is as intended. In addition, "may" is changed to "can" in accordance with the discussion and resolution of comment #33 (item pertaining to 12.4 on p.177) against D2.0; however, note that "may" is used for the corresponding requirement in IEEE 802.1AS-2011. The TSN TG should review this point. Finally, note that the above text may change due to resolution of open issues related to backward compatibility and related treatment of domain 0.>>

~~FALSE if the *timing measurement* bit in the Extended Capabilities information element defined in Table 7-35a of IEEE Std 802.11-2012 indicates that the peer IEEE 802.11 station is incapable of participating in the timing measurement protocol. Otherwise, asCapable may be set to TRUE.~~

12.5 State machines

12.5.1 Media-dependent master state machine

12.5.1.1 Overview

The MD entity of an IEEE 802.11 port whose port ~~role~~state is MasterPort (see Table 10-1) shall behave in a manner that is indistinguishable, relative to an observer external to a system, from a strict implementation of the master state ~~diagram~~machines in Figure 12-4 and Figure 12-5 (referred to as master state machine A and master state machine B, respectively), the local variables specified in 12.5.1.3, the functions specified in 12.5.1.4, the shared variables specified in 12.5.1.5, and the primitives defined in 12.5.1.6.

In the case of timing measurement, ~~The~~master state machine A is responsible for initiating a time measurement whenever the PortSync entity requests it do so, as indicated by the rcvdMDSync Boolean. ~~The~~ M~~m~~aster state machine A invokes the IEEE 802.11 MLME-TIMINGMSMT.request primitive and waits for the subsequent MLME-TIMINGMSMT.confirm primitive. It collects local timestamp information from the measurement (*t1* and *t4*, provided by the confirm primitive) and includes the information in the subsequent request. See 8.4.3 for more information on timestamps. Master state machine B is not used for timing measurement.

For fine timing measurement, master state machine A receives and stores information from the PortSync entity. Master state machine B receives the MLME-FINETIMINGMSMTRQ.indication caused by the initial FTM request from the slave. It sets asCapable to TRUE as specified in 12.4, but only if it can grant the parameters received from the slave in the initial FTM request. If asCapable is TRUE, it then generates successive MLME-FINETIMINGMSMT.request primitives to cause information saved by master state machine A to be sent to the slave. It receives MLME-FINETIMINGMSMT.confirm primitives caused by ACKs received from the slave. It collects local timestamp information from the current measurement (*t1* and *t4*, provided by the confirm primitive) and includes the information in the subsequent MLME-FINETIMINGMSMT.request.

12.5.1.2 State diagrams

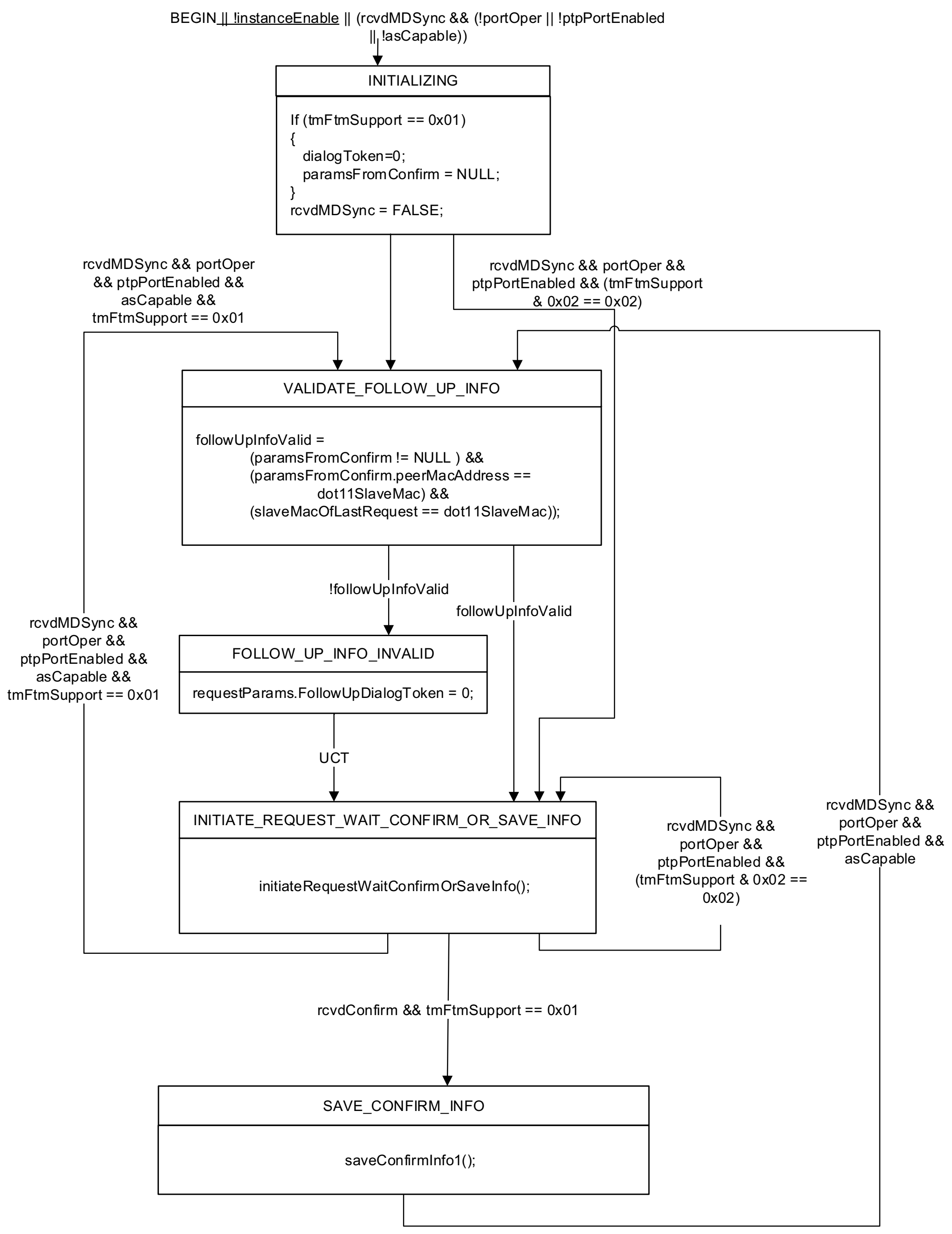

BEGIN || !instanceEnable || (rcvdMDSync && (!portOper || !ptpPortEnabled || !asCapable))
INITIALIZING
If (tmFtmSupport == 0x01)
{
dialogToken=0;
paramsFromConfirm = NULL;
}
rcvdMDSync = FALSE;
rcvdMDSync && portOper && ptpPortEnabled && asCapable && tmFtmSupport == 0x01
rcvdMDSync && portOper && ptpPortEnabled && (tmFtmSupport & 0x02 == 0x02)
VALIDATE_FOLLOW_UP_INFO
followUpInfoValid = (paramsFromConfirm != NULL) && (paramsFromConfirm.peerMacAddress == dot11SlaveMac) && (slaveMacOfLastRequest == dot11SlaveMac));
!followUpInfoValid
followUpInfoValid
rcvdMDSync && portOper && ptpPortEnabled && asCapable && tmFtmSupport == 0x01
FOLLOW_UP_INFO_INVALID
requestParams.FollowUpDialogToken = 0;
UCT
INITIATE_REQUEST_WAIT_CONFIRM_OR_SAVE_INFO
initiateRequestWaitConfirmOrSaveInfo();
rcvdMDSync && portOper && ptpPortEnabled && (tmFtmSupport & 0x02 == 0x02)
rcvdMDSync && portOper && ptpPortEnabled && asCapable
rcvdConfirm && tmFtmSupport == 0x01
SAVE_CONFIRM_INFO
saveConfirmInfo1();

Figure 12-4—Master state machine A. (a) For TM, receives information from the PortSync entity and sends to slave, and (b) for FTM, receives and stores information from the Port-Sync entity.

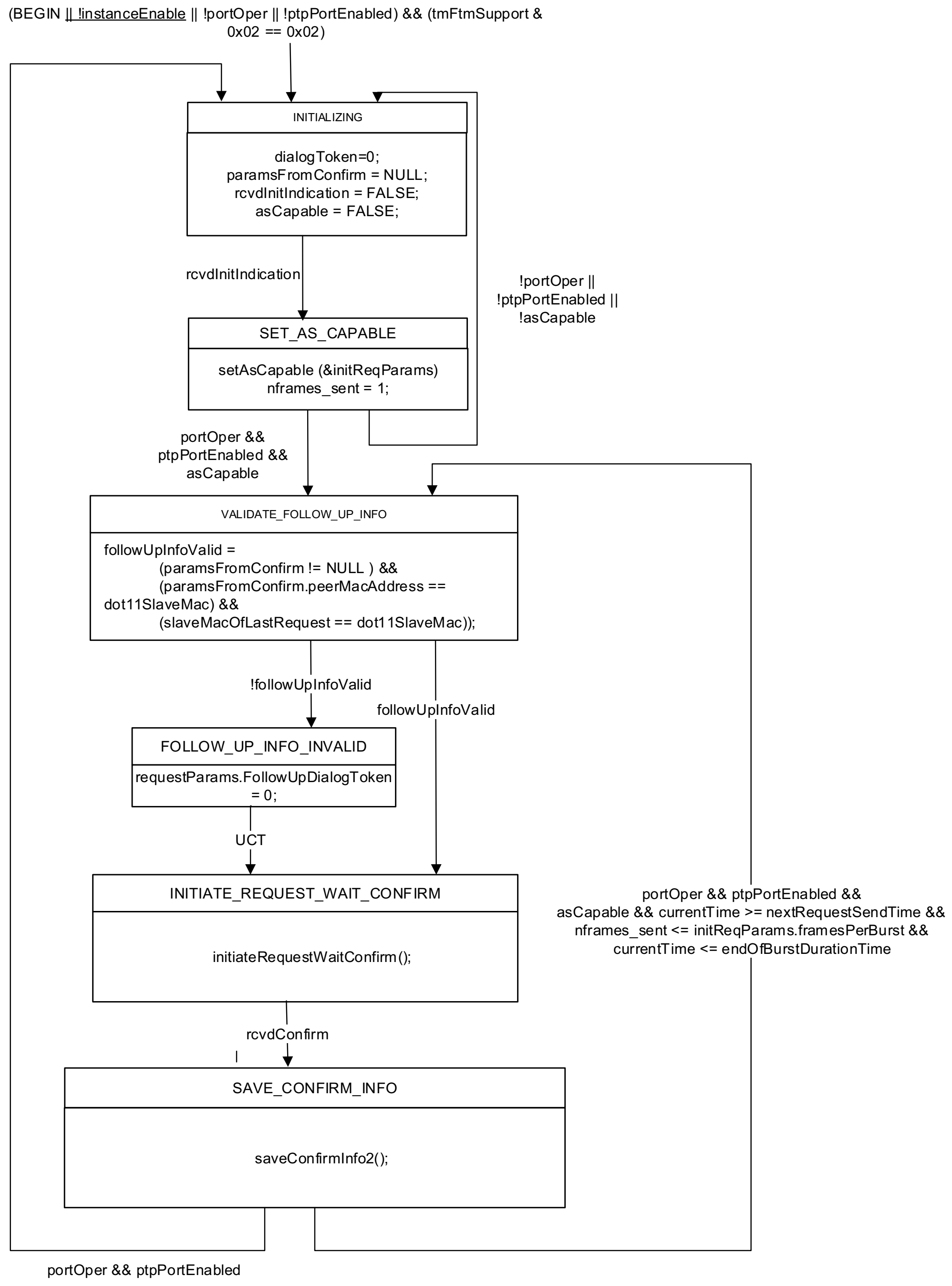

(BEGIN || !instanceEnable || !portOper || !ptpPortEnabled) && (tmFtmSupport & 0x02 == 0x02)
INITIALIZING
dialogToken=0;
paramsFromConfirm = NULL;
rcvdInitIndication = FALSE;
asCapable = FALSE;
rcvdInitIndication
!portOper || !ptpPortEnabled || !asCapable
SET_AS_CAPABLE
setAsCapable (&initReqParams)
nframes_sent = 1;
portOper && ptpPortEnabled && asCapable
VALIDATE_FOLLOW_UP_INFO
followUpInfoValid = (paramsFromConfirm != NULL) && (paramsFromConfirm.peerMacAddress == dot11SlaveMac) && (slaveMacOfLastRequest == dot11SlaveMac));
!followUpInfoValid
followUpInfoValid
FOLLOW_UP_INFO_INVALID
requestParams.FollowUpDialogToken = 0;
UCT
INITIATE_REQUEST_WAIT_CONFIRM
initiateRequestWaitConfirm();
portOper && ptpPortEnabled && asCapable && currentTime >= nextRequestSendTime && nframes_sent <= initReqParams.framesPerBurst && currentTime <= endOfBurstDurationTime
rcvdConfirm
SAVE_CONFIRM_INFO
saveConfirmInfo2();
portOper && ptpPortEnabled

Figure 12-5—Master state machine B. (a) For TM, not invoked, and (b) for FTM, receives initial FTM request from slave and sends information received from upstream to slave in successive FTM frames

12.5.1.3 State machine local variables

a) **dialogToken:** an unsigned 8-bit integer used to identify a measurement from among those preceding and following it.
b) **followUpInfoValid:** a Boolean variable indicating whether the FollowUp information (e.g., timestamps and rateRatio) and the link partner are unchanged since the last timing measurement or fine timing measurement.
c) **requestParams:** a structure whose members contain the values of the fields of either the MLME-TIMINGMSMT.request or MLME-FINETIMINGMSMT.request primitive.
d) **paramsFromConfirm:** a structure whose members contain the values of the fields of either the MLME-TIMINGMSMT.confirm or MLME-FINETIMINGMSMT.confirm primitive.
e) **dot11SlaveMac:** the MAC address of the station associated with the current port.
f) **slaveMacOfLastRequest:** the MAC address of the station of the previous request, used to validate FollowUp information.
g) **residenceTime:** a temporary variable that holds the computation of the time between receipt of the last synchronization information and transmission of synchronization information.
h) **rcvdMDSync:** a Boolean variable that is set to TRUE when an MDSyncSend structure is provided by the PortSync entity.
i) **rcvdconfirm:** a Boolean variable that is set to TRUE when either the MLME-TIMINGMSMT.confirm or MLME-FINETIMINGMSMT.confirm primitive is received.
j) **nframes_sent:** an unsigned 8-bit integer used to count the number of frames sent by the slave (the number of indications received is counted).

<<Editor's note: It must be confirmed that 8 bits is sufficient for nframes_sent.>>

k) **rcvdInitIndication:** a Boolean variable that is set to TRUE when the initial MLME-FINETIMINGMSMTRQ.indication primitive is received.
l) **initReqParams:** a structure whose members contain the values of the fields of an MLME-FINETIMINGMSMTRQ.indication primitive.
m) **endOfBurstDurationTime:** a UScaledNs variable whose value is the time at which the current burst ends.
n) **nextRequestSendTime:** a UScaledNs variable whose value is the expected time that the next MLME-FINETIMINGMSMTRQ.indication primitive, for the next request from the slave, will be received.

12.5.1.4 State machine functions

a) **setRequestParams(&requestParams, MDSyncSend):** assigns values to the parameters of the request primitive of either MLME-TIMINGMSMT or MLME-FINETIMINGMSMT (see 12.5.1.6) as follows:
   1) Members of the FollowUpInformation member of the VendorSpecific information element, as defined in ~~12.5.2.6.2~~12.6, are assigned as defined in 11.4.4 with the exception of the correctionField, which is assigned as shown in the Master state machine in 12.5.1.2.
   2) The other fields of the VendorSpecific information element are assigned as follows:
      i) ElementID is assigned the value 221 as defined in Table 8-54 (Element IDs) of IEEE Std 802.11-2012, indicating that the information element is of type Vendor Specific.
      ii) The Length field is set to 80.

NOTE—This is equal to the length of the Follow_Up payload defined in 11.4.4 (including the common header) plus the length of the OUI or CID field and the Type field~~s~~ (see Figure 12-7).

      iii) The OUI or CID field is set to 00-80-C2.
      iv) The Type field is set to 0.

3) ~~MaxT1Error~~Max t1 Error, ~~MaxT4Error~~Max-t4 Error are set to zero.
4) For Fine Timing Measurement frames, the location configuration information (LCI) Report, Location Civic Report are not present.

<<Editor's note to editor: If multiple FTMs per burst are supported, the Fine Timing Measurement Parameters would be present.>>

5) All other members are left unchanged.

b) **setAsCapable (&initReqParams):** determines the value of asCapable consistent with 12.4 and whether the master is able to grant the parameters requested by the slave. This function is used only in the case of FTM.

<<Editor's note: See the editor's note in 12.4 on whether TM should be used, if the master cannot grant the parameters requested by the slave but both master and slave also support TM. Note that if that editor's note is deleted, this editor's note also will be deleted.>>

c) **initiateRequestWaitConfirmOrSaveInfo():** This function is defined as indicated below. It is used in Master state machine A. It is defined so that the detailed code that it invokes does not need to be fit into the state machine diagram.

```
initiateRequestWaitConfirmOrSaveInfo()
{
        rcvdMDSync = FALSE;

        If (tmFtmSupport == 0x01)
        {
                if ((++dialogToken % 256) == 0) dialogToken++;
                requestParams.DialogToken=dialogToken;
                requestParams.PeerMACAddress = dot11SlaveMac;
                setRequestParams(&requestParams, MDSyncSend);
                MLME-TIMINGMSMT.request(requestParams);
                requestParams.FollowUpDialogToken = 0;
                                        //In case no confirm is received
                slaveMacOfLastRequest = dot11SlaveMac;
        }
        else if ((tmFtmSupport & 0x02 == 0x02))
                saveMDSyncSend (&MDSyncSendSave, MDSyncSend);
}
```

d) **saveConfirmInfo1():** This function is defined as indicated below. It is used in Master state machine A. It is defined so that the detailed code that it invokes does not need to be fit into the state machine diagram.

```
saveConfirmInfo1()
{
        MLME-TIMINGMSMT.confirm(&paramsFromConfirm);

        requestParams.FollowUpDialogToken = paramsFromConfirm.DialogToken;
        requestParams.T1 = paramsFromConfirm.T1;
        requestParams.T4 = paramsFromConfirm.T4;

        // NOTE: In Timing Measurement, T1 is in units of 10 ns.
        // upstreamTxTime is units of 2-16 nanoseconds.

        K = 1;
        // K is 1 for Timing Measurement.
        residenceTime = MDSyncSend.rateRatio *
                (paramsFromConfirm.T1 * $10^K$*($2^{16}$) - MDSyncSend.upstreamTxTime);

        requestParams.VendorSpecific.correctionField =
                residenceTime + MDSyncSend.followUpCorrectionField;
        // NOTE: T1 and T4 are timestamps from a single
        // local clock source.  The roll-over of the 32-bit timestamps returned by
        // MLME-TIMINGMSMT.request and MLME-TIMINGMSMT.indication
        //        must be accounted for.
}
```

e) **initiateRequestWaitConfirm():** This function is defined as indicated below. It is used in Master state machine B. It is defined so that the detailed code that it invokes does not need to be fit into the state machine diagram.

```
initiateRequestWaitConfirm()
{
        If ((++dialogToken % 256) == 0) dialogToken++;
        If (nframes_sent == initReqParams.framesPerBurst)
                dialogToken = 0;

        requestParams.DialogToken=dialogToken;
        requestParams.PeerMACAddress = dot11SlaveMac;
        setRequestParams(&requestParams, MDSyncSend);
        // In the following statement, MinDeltaFTM, which is in units of 100
        //    microseconds, is converted to UScaledNs (i.e., units of $2^{-16}$ ns, see 6.3.3.2)
        nextRequestSendTime = currentTime + initReqParams.MinDeltaFTM *
                (65536 x $10^5$);
        MLME-FINETIMINGMSMT.request(requestParams);
        requestParams.FollowUpDialogToken = 0;    //In case no confirm is received
        slaveMacOfLastRequest = dot11SlaveMac;
        // In the following statement, the burst duration parameter from 802.11RevMC is
        //         converted to UScaledNs (i.e., units of $2^{-16}$ ns, see 6.3.3.2). The
        //         quantity $2^{initReqParams.burstDuration - 2}$ is the burst duration in µs.
        // Also in the following statement, we are assuming that the burst duration
        //    starts when the initial FTM request is received. In actuality, the timer begins
        //    by the partial TSF timer value indicated in the initial FTM frame, which is
        //    slightly after the initial FTM request is received.
        A = initReqParams.burstDuration – 2;
        endOfBurstDurationTime = currentTime + 1000 * ($2^{16}$) * 250 * $2^A$;
}
```

f) **saveConfirmInfo2():** This function is defined as indicated below. It is used in Master state machine B. It is defined so that the detailed code that it invokes does not need to be fit into the state machine diagram.

```
saveConfirmInfo2()
{
        MLME-FINETIMINGMSMT.confirm(&paramsFromConfirm);

        requestParams.FollowUpDialogToken = paramsFromConfirm.DialogToken;
        requestParams.T1 = paramsFromConfirm.T1;
        requestParams.T4 = paramsFromConfirm.T4;

        // NOTE: In Fine Timing Measurement, T1 is in units of 0.1 ns.
        // upstreamTxTime is units of $2^{-16}$ nanoseconds.
```

```
            K = -3;
            // K is 1 for Timing Measurement and -3 for Fine Timing Measurement.
            residenceTime = MDSyncSend.rateRatio *
                  (paramsFromConfirm.T1 * 10^K*(2^16) - MDSyncSend.upstreamTxTime);

            requestParams.VendorSpecific.correctionField =
            residenceTime + MDSyncSend.followUpCorrectionField;
            // NOTE: T1 and T4 are timestamps from a single
            // local clock source. The roll-over of the 48-bit timestamps returned by
            // MLME-FINETIMINGMSMT.request and MLME-FINETIMINGMSMT.indication
            // must be accounted for.

            // A frame is only counted as being sent, for purposes of number of frames in a
            // burst, if a confirm is received. It is up to IEEE Std 802.11 to handle the case
            // where a confirm is not received.
            nframes_sent += 1;
      }
```

12.5.1.5 Shared Variables

<<Editor's note: The variables in this section are global variables, and are defined elsewhere. It is planned to organized clause 12 more similarly to clause 11, i.e., have one single section with per-port global variables.>>

a) **MDSyncSend:** a structure as defined in 10.2.2.1.
b) **~~portEnabled~~portOper:** a Boolean as defined in 10.2.4.12.
c) **~~pttPortEnabled~~ptpPortEnabled:** a Boolean as defined in 10.2.4.13.
d) **asCapable:** a Boolean whose value is specified in 12.4.
e) **tmFtmSupport:** an Octet whose value is specified in 12.3.

12.5.1.6 Master primitives

[Editor's note: A description of the MLME-FINETIMINGMSMTRQ.indication primitive must be added.]

12.5.1.6.1 MLME-TIMINGMSMT.request

The MLME-TIMINGMSMT request primitive is used by a master station to initiate a timing measurement and also communicates timestamps *t1* and *t4* captured by the master during a previous measurement.

~~Its~~The primitive parameters are as follows:

```
MLME-TIMINGMSMT.request(
            PeerMACAddress,
            DialogToken,
            FollowUpDialogToken,
            ~~T1~~t1,
            Max ~~T~~t1 Error,
            ~~T4~~t4,
            Max ~~T~~t4 Error,
            VendorSpecific)
```

Table 12-2—Parameters of MLME-TIMINGMSMT.request

Name	Type	Valid range	Description
PeerMACAddress	MACAddress	N/A	The address of the peer MAC entity with which the Timing Measurement is initiated.
DialogToken	UInteger8	1–255	The dialog token used to identify this Timing Measurement transaction.
FollowUpDialog-Token	UInteger8	0–255	The dialog token of a previous Timing Measurement from which the FollowUpInformation is derived.
~~T1~~t1	UInteger32	0– (2^{32}–1)	Transmit (time of departure) timestamp *t1* from the time measurement indicated by the FollowUpDialogToken,~~.~~ ~~In~~in units of 10 ns.
Max ~~T~~t1 Error	Integer	0–255	Timestamp error, in units of 10 ns.
~~T4~~t4	UInteger32	0– (2^{32}–1)	Receive (time of arrival) timestamp *t4* from the time measurement indicated by the FollowUpDialogToken~~. In~~, in units of 10 ns.
Max ~~T~~t4 Error	Integer	0–255	Timestamp error, in units of 10 ns.
VendorSpecific	N/A	N/A	A Vendor Specific information element containing various time-synchronization parameters, including an *entire* Follow_Up message, as defined in ~~12.5.2.6.2~~12.6.

Refer to IEEEStd 802.11 for more information on the MLME primitives.

12.5.1.6.2 MLME-TIMINGMSMT.confirm

The MLME-TIMINGMSMT.confirm primitive indicates that a timing measurement request has completed.

~~Its~~ The primitive parameters are as follows:

MLME-TIMINGMSMT.confirm(
PeerMACAddress,
DialogToken,
~~T1~~t1,
Max ~~T~~t1 Error,
~~T4~~t4,
Max ~~T~~t4 Error)

Table 12-3—Parameters of MLME-TIMINGMSMST.confirm

Name	Type	Valid range	Description
PeerMACAddress	MACAddress	N/A	The address of the peer MAC entity, which acknowledges the receipt of the Timing Measurement action frame.
DialogToken	Integer	1–255	The dialog token to identify the Timing Measurement transaction.
~~T1~~t1	UInteger32	0– (2^{32}–1)	The transmit timestamp *t1* in units of 10 ns.

Max ~~T~~t1 Error	UInteger8	0–255	Maximum error of timestamp *t1* in units of 10 ns.
~~T4~~t4	UInteger32	0– (2^{32}–1)	The receive timestamp *t4* in units of 10 ns.
Max ~~T~~t4 Error	UInteger8	0–255	Maximum error of timestamp *t4* in units of 10 ns.

Refer to IEEE Std 802.11 for more information on this MLME primitive.

12.5.1.6.3 MLME-FINETIMINGMSMT.request

The MLME-FINETIMINGMSMT request primitive is used by a master station to initiate a fine timing measurement, and also communicates timestamps *t1* and *t4* captured by the master during a previous measurement.

The primitive parameters are as follows:

```
MLME-FINETIMINGMSMT.request    (
                  PeerMACAddress,
                  DialogToken,
                  Follow Up Dialog Token,
                  t1,
                  Max t1 Error,
                  t4,
                  Max t4 Error,
                  LCI Report,
                  Location Civic Report,
                  Fine Timing Measurement Parameter,
                  VendorSpecific)
```

Table 12-4—Parameters of MLME-FINETIMINGMSMT.request

Name	Type	Valid range	Description
PeerMACAddress	MACAddress	Any valid individual addressed MAC Address	The address of the peer MAC entity to which the Fine Timing Measurement frame is sent.
DialogToken	UInteger8	1–255	The dialog token to identify the Fine Timing Measurement transaction. A value of 0 indicates the end of the transaction.
FollowUpDialogToken	UInteger8	0–255	The dialog token of a Fine Timing Measurement frame which the current frame follows.
t1	UInteger48	0– (2^{48}–1)	Set to the value of t1 expressed in 0.1 ns units.
Max t1 Error Exponent	UInteger5	0–31	The maximum error in the t1 value, in ps, is is given by Eq. (12-1), evaluated at Max t1 Error Exponent.
t4	UInteger48	0– (2^{48}–1)	Set to the value of t4 expressed in 0.1 ns units.

Max t4 Error	UInteger5	0–31	The maximum error in the t4 value, in ps, is is given by Eq. (12-1), evaluated at Max t4 Error Exponent.
LCI Report	N/A	N/A	Optional element to report LCI information of sender
Location Civic Report	N/A	N/A	Optional element to report Location Civic information of sender
Fine Timing Measurement Parameter	N/A	N/A	Optional element containing the proposed fine timing measurement configuration
VendorSpecific	N/A	N/A	A Vendor Specific information element containing various time-synchronization parameters (including a full Follow_Up message), as defined in 12.6.

primitive.

The maximum error in a timestamp, t1, t2, t3, or t4, is given by

$$E_{max}(0) = unknown$$

$$E_{max}(F) = 2^{F-1}, 1 \le F \le 30 \qquad (12\text{-}1)$$

$$E_{max}(31) \ge 2^{30},$$

where the function $E_{max}(F)$ is the maximum error in a timestamp, in ps, and is evaluated at Max t1 Error Exponent, Max t2 Error Exponent, Max t3 Error Exponent, or Max t4 Error Exponent, respectively.

12.5.1.6.4 MLME-FINETIMINGMSMT.confirm

The MLME-FINETIMINGMSMT request primitive indicates that a fine timing measurement request has completed.

The primitive parameters are as follows:

```
MLME-FINETIMINGMSMT.confirm  (
                  PeerMACAddress,
                  DialogToken,
                  t1,
                  Max t1 Error,
                  t4,
                  Max t4 Error)
```

Table 12-5—Parameters of MLME-FINETIMINGMSMT.confirm

Name	Type	Valid range	Description
PeerMACAddress	MACAddress	Any valid individual addressed MAC Address	The address of the peer MAC entity to which the Fine Timing Measurement frame is sent.

DialogToken	UInteger8	1–255	The dialog token to identify the Fine Timing Measurement transaction. A value of 0 indicates the end of the transaction.
t1	UInteger48	0– (2^{48}–1)	Set to the value of t1 expressed in 0.1 ns units.
Max t1 Error	UInteger5	0–31	The maximum error in the t1 value, in ps, is is given by Eq. (12-1), evaluated at Max t1 Error Exponent.
t4	UInteger48	0– (2^{48}–1)	Set to the value of t4 expressed in 0.1 ns units.
Max t4 Error	UInteger5	0–31	The maximum error in the t4 value, in ps, is is given by Eq. (12-1), evaluated at Max t4 Error Exponent.

12.5.2 Media-dependent slave state machine

12.5.2.1 Overview

The MD entity of an IEEE 802.11 port whose port ~~role~~state is SlavePort or PassivePort (see 10.3.6) shall behave in a manner that is indistinguishable, relative to an observer external to a system, from a strict implementation of the slave state ~~diagram~~machine in 12.5.2.2, the local variables specified in 12.5.2.3, the functions specified in 12.5.2.4, the shared variables specified in 12.5.2.5, and the primitives defined in 12.5.2.6.

The slave state machine is responsible for collecting information from the Timing measurement or Fine Timing measurement indication, constructing an MDSyncReceive structure with the relevant information, and passing the structure to the PortSync entity for further processing. In order to do this, the state machine saves locally captured timestamps (i.e., *t2* and *t3*) received in the indication, associating them with the timestamps sent from the master port in a future indication (i.e., *t1* and *t4*). In addition, in the case of Fine Timing measurement, the slave state machine is responsible for generating the MLME-FINETIMINGMSMT.request primitive, which causes the initial FTM request frame to be sent to the master.

12.5.2.2 State diagram

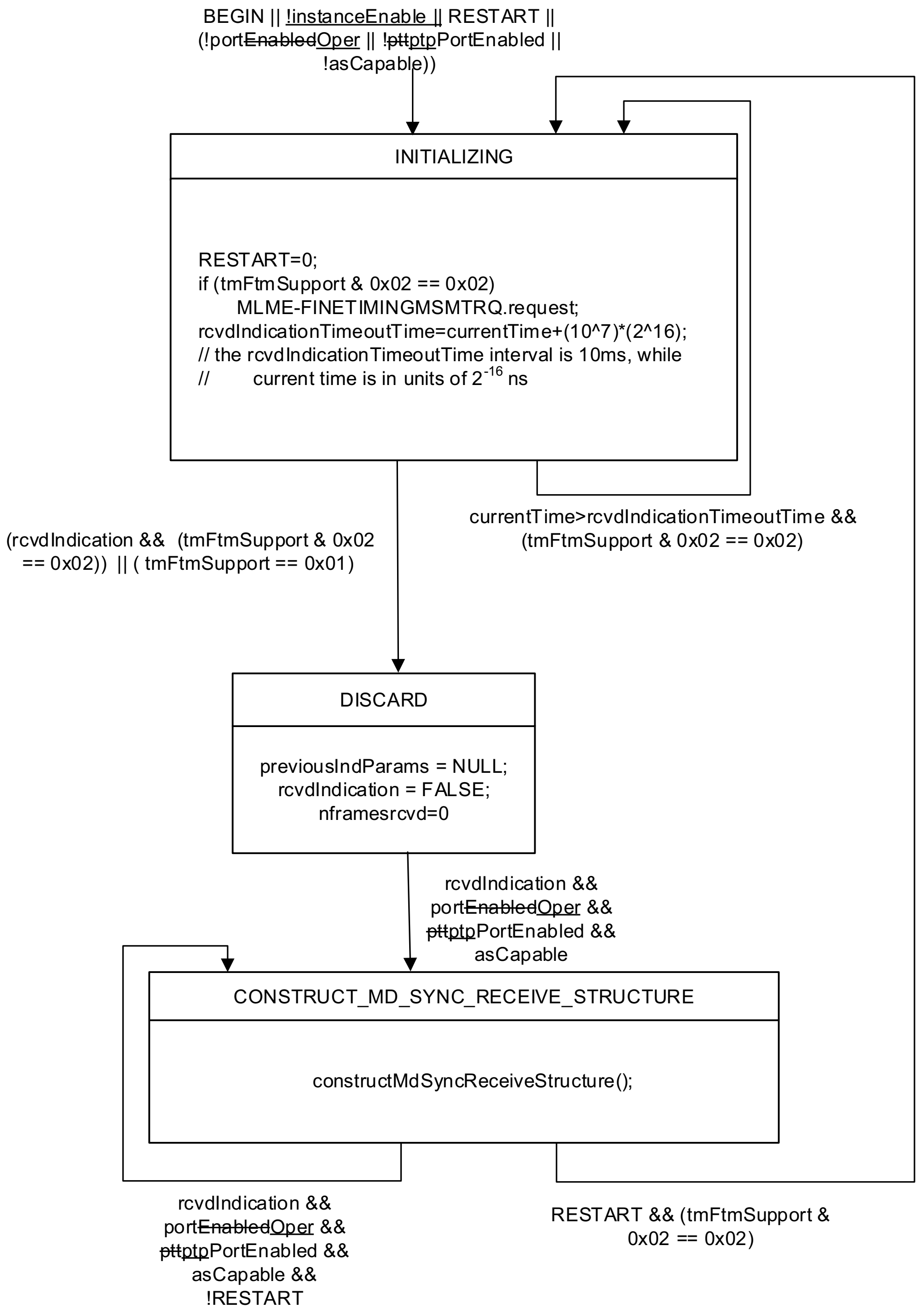

BEGIN || !instanceEnable || RESTART || (!port~~Enabled~~Oper || !~~ptt~~ptpPortEnabled || !asCapable))
INITIALIZING
RESTART=0;
if (tmFtmSupport & 0x02 == 0x02)
MLME-FINETIMINGMSMTRQ.request;
rcvdIndicationTimeoutTime=currentTime+(10^7)*(2^16);
// the rcvdIndicationTimeoutTime interval is 10ms, while
// current time is in units of 2^-16 ns
currentTime>rcvdIndicationTimeoutTime && (tmFtmSupport & 0x02 == 0x02)
(rcvdIndication && (tmFtmSupport & 0x02 == 0x02)) || (tmFtmSupport == 0x01)
DISCARD
previousIndParams = NULL;
rcvdIndication = FALSE;
nframesrcvd=0
rcvdIndication && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && asCapable
CONSTRUCT_MD_SYNC_RECEIVE_STRUCTURE
constructMdSyncReceiveStructure();
rcvdIndication && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && asCapable && !RESTART
RESTART && (tmFtmSupport & 0x02 == 0x02)

Figure 12-6—Slave state machine

While quantities are shown to be computed from information in consecutive indications, an implementation ~~may~~can choose to compute over longer intervals as long as the clock performance requirements of Annex B are met.

12.5.2.3 State machine local variables

a) **indParams:** a structure whose members contain the values of the fields of the MLME-TIMINGMSMT.indication primitive, as defined in 12.5.2.6.
b) **previousIndParams:** a structure with members identical to those of indParams, used to save parameters from the previous indication.
c) **neighborRateRatio:** the measured ratio of the frequency of the LocalClock entity of the remote system to the frequency of the LocalClock entity of this system. The data type is Double.
d) **rcvdIndication:** a Boolean ~~which~~that is set to TRUE when either the MLME-TIMINGMSMT.indication~~-~~ or MLME-FINETIMINGMSMT.indication primitive is received.
e) **RESTART:** a Boolean that indicates, in the case FTM is being used, that a new burst should be initiated.
f) **rcvdIndicationTimeoutTime:** a UScaledNs variable whose value is the time after which the state machine will not wait any longer for the next MLME-FINETIMINGMSMT.indication, and a new burst is initiated.
g) **nframesrcvd:** an unsigned 8-bit integer used to count the number of frames received by the master in the burst (the number of indications received from the master are counted).

[Editor's note: It must be confirmed that 8 bits is sufficient for nframesrcvd.]

h) **initReqParams:** a structure whose members contain the values of the fields of an MLME-FINETIMINGMSMTRQ.indication primitive.

12.5.2.4 State machine functions

a) **setMDSyncReceive(indParams):** creates an MDSyncReceive structure and returns the structure. All fields are assigned from FollowUpInformation (contained in the VendorSpecific information element) of indParams as in 11.2.13.2.1.
b) **passMDSyncReceiveToPortSync():** passes an MDSyncReceive structure to the PortSync entity of this port.
~~c)~~ ~~**setAsCapable(&initReqParams):** determines the value of asCapable consistent with 12.4 and whether the master is able to grant the parameters requested by the slave. This function is used only in the case of FTM.~~

~~**<<Editor's note: See the editor's note in 12.4 on whether TM should be used, if the master cannot grant the parameters requested by the slave but both master and slave also support TM.>>**~~

<<Editor's note: It was indicated in the Septebmer 2016 TSN TG meeting that the editor should check whether the above function setAsCapable is needed for the slave state machine. On review of the FTM mechanism and the editor's note in 12.4 (just after the paragraph that is immediately after the list with entries a), b), and c)), it appears that this function is not needed. If there are comments on this, the above function and editor's note, and the current editor's note, will be deleted.>>

d) **constructMdSyncReceiveStructure():** This function is defined as indicated below. It is used in the Slave state machine. It is defined so that the detailed code that it invokes does not need to be fit into the state machine diagram.

```
constructMdSyncReceiveStructure()
{
        if (tmFtmSupport == 0x01)
                MLME-TIMINGMSMT.indication(&indParams);
        else if (tmFtmSupport & 0x02 == 0x02)
        {
            MLME-FINETIMINGMSMT.indication(&indParams);
            nframesrcvd++;
            if (nframesrcvd == initReqParams.framesPerBurst) ||
                (currentTime > endofBurstDurationTime)
                    RESTART=1;
        }

        if ((previousIndParams != NULL) &&
            (previousIndParams.PeerMacAddress == dot11SlaveMac) &&
            (indParams.FollowUpDialogToken != 0))
        {

            neighborRateRatio =
                (indParams.T1-previousIndParams.T1) /
                (indParams.T2-previousIndParams.T2);
            //NOTE: Other methods of computing neighborRateRatio
                    ~~may~~ can be used.

            if (tmFtmSupport == 0x01)
                K = 1;
            else if (tmFtmSupport & 0x02 == 0x02)
                K = -3;
            //K = 1 for Timing Measurement and K = -3 for Fine Timing Measurement
            neighborPropDelay =
                (((indParams.T4 - indParams.T1) -
                neighborRateRatio * (indParams.T3 - indParams.T2)) /
                            (2.0)) * $(10^K)$;

            //NOTE: Other methods of computing neighborPropDelay
                    ~~may~~ can be used.

            MDSyncReceive = setMDSyncReceive(indParams);
            MDSyncReceive.VendorSpecific.rateRatio += (neighborRateRatio – 1);
            MDSyncReceive.VendorSpecific.upstreamTxTime =
                    indParams.T2* $(2^{16})$ *$(10^K)$ -
                    neighborPropDelay*$(2^{16})$/neighborRateRatio;
            //NOTE: Actions performed with the timestampError
                parameters of indParams are implementation independent.

            passMDSyncReceiveToPortSync(&MDSyncReceive);
            ~~previousIndParams = indParams;~~
        }
        previousIndParams = indParams;
        rcvdIndication = FALSE;
    }
```

12.5.2.5 State machine shared variables

<<Editor's note: The variables in this section are global variables, and are defined elsewhere. It is planned to organized clause 12 more similarly to clause 11, i.e., have one single section with per-port global variables.>>

a) **MDSyncReceive:** a structure used for passing information between MD and PortSync, as defined in 10.2.2.2.
b) ~~**portEnabled**~~portOper: a Boolean as defined in 10.2.4.12.
c) ~~**pttPortEnabled**~~ptpPortEnabled: a Boolean as defined in 10.2.4.
d) **asCapable:** a Boolean as defined in 12.4.
e) **neighborPropDelay:** the delay over the link to the associated WLAN station as defined in 10.2.4.8.
f) **tmFtmSupport:** an Octet whose value is specified in 12.3.

12.5.2.6 Slave primitives

[Editor's note: A description of the MLME-FINETIMINGMSMTRQ.request primitive must be added.]

12.5.2.6.1 MLME-TIMINGMSMT.indication

The MLME-TIMINGMSMT.indication primitive is received by a slave station as the natural result of the peer master station issuing the corresponding request primitive, and carries the same parameters plus local timestamp information.

Its parameters are as follows:

```
MLME-TIMINGMSMT.indication (
                PeerMACAddress,
                DialogToken,
                FollowUpDialogToken,
                ~~T1~~t1,
                Max_~~T~~t1_Error,
                ~~T4~~t4,
                Max_~~T~~t4_Error,
                ~~T2~~t2,
                Max_~~T~~t2_Error,
                ~~T3~~t3,
                Max_~~T~~t3_Error,
                VendorSpecific)
```

Table 12-6—Parameters of MLME-TIMINGMSMT.indication

Name	Type	Valid range	Description
PeerMACAddress	MACAddress	N/A	The address of the peer MAC entity from which the Timing Measurement was initiated.
DialogToken	UInteger8	1–255	The dialog token used to identify this Timing Measurement.
FollowUpDialog-Token	UInteger8	0–255	The dialog token of a Timing Measurement to which the indication is a follow on.
~~T1~~t1	UInteger32	0– 2^{32}–1	Transmit timestamp *t1* from the time measurement indicated by the FollowUpDialog Token, in units of 10 ns.
Max ~~T~~t1 Error	UInteger8	0–255	Error of the timestamp in units of 10 ns.
~~T4~~t4	UInteger32	0– 2^{32}–1	Receive timestamp *t4* from the time measurement indicated by the FollowUpDialog Token, in units of 10 ns.
Max ~~T~~t4 Error	UInteger8	0–255	Error of the timestamp in units of 10 ns.
~~T2~~t2	UInteger32	0– 2^{32}–1	Receive timestamp *t2* captured from the current time measurement, in units of 10 ns.
Max ~~T~~t2 Error	UInteger8	0–255	Error of the timestamp in units of 10 ns.
~~T3~~t3	UInteger32	0– 2^{32}–1	Transmit timestamp *t3* captured from the current time measurement, in units of 10 ns.
Max ~~T~~t3 Error	UInteger8	0–255	Error of the timestamp in units of 10 ns.
VendorSpecific	N/A	N/A	A Vendor Specific information element containing various time-synchronization parameters (including a full Follow_Up message), as defined in 12.6~~12.5.2.6.2~~.

12.5.2.6.2 MLME-FINETIMINGMSMT.indication

The MLME-FINETIMINGMSMT.indication primitive is received by a slave station as the natural result of the peer master station issuing the corresponding request primitive, and carries the same parameters plus local timestamp information.

The primitive parameters are as follows:

```
MLME-FINETIMINGMSMT.indication(
                PeerMACAddress,
                DialogToken,
                FollowUpDialogToken,
                t1,
                Max t1 Error,
                t4,
                Max t4 Error,
```

t2,
Max t2 Error,
t3,
Max t3 Error,
VendorSpecific)

Table 12-7—Parameters of MLME-FINETIMINGMSMT.indication

Name	Type	Valid range	Description
PeerMACAddress	MACAddress	Any valid individual addressed MAC Address	The address of the peer MAC entity from which the Timing Measurement was sent.
DialogToken	UInteger8	1–255	The dialog token to identify the Fine Timing Measurement transaction. A value of 0 indicates the end of the transaction.
FollowUpDialog-Token	UInteger8	0–255	The dialog token of a Fine Timing Measurement frame that the current frame follows.
t1	UInteger48	0– 2^{48}–1	Set to the value of t1 expressed in 0.1 ns units.
Max t1 Error	UInteger5	0–31	The maximum error in the t1 value, in ps, is is given by Eq. (12-1), evaluated at Max t1 Error Exponent.
t4	UInteger48	0– 2^{48}–1	Set to the value of t4 expressed in 0.1 ns units.
Max t4 Error	UInteger5	0–31	The maximum error in the t4 value, in ps, is is given by Eq. (12-1), evaluated at Max t4 Error Exponent.
t2	UInteger48	0– 2^{48}–1	Set to the value of t2 expressed in 0.1 ns units.
Max t2 Error	UInteger5	0–31	The maximum error in the t2 value, in ps, is is given by Eq. (12-1), evaluated at Max t2 Error Exponent.
t3	UInteger48	0– 2^{48}–1	Set to the value of t3 expressed in 0.1 ns units.
Max t3 Error	UInteger5	0–31	The maximum error in the t3 value, in ps, is is given by Eq. (12-1), evaluated at Max t3 Error Exponent.
VendorSpecific	N/A	N/A	A Vendor Specific information element containing various time-synchronization parameters (including a full Follow_Up message), as defined in 12.6.

12.6 Format of VendorSpecific information element

The IEEE 802.11 MLME request and indication primitives for timing measurement and fine timing measurement support an ability to carry data transparently between stations using the VendorSpecific information element. The Type field within the VendorSpecific Content identifies the type of information that follows the Type field. See~~Figure~~ Figure 12-7 and Table 12-8.

Table 12-8—Values of the Type field in the VendorSpecific information element

Value	Description
0	The Type field is followed by FollowUp-Information
1–255	Reserved

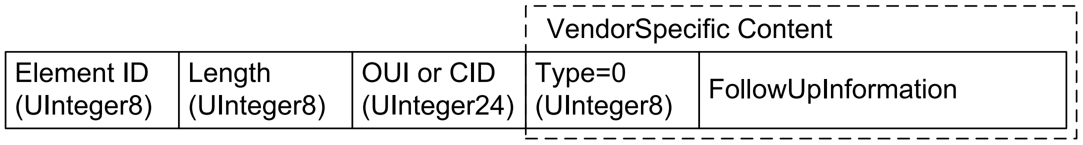


Figure 12-7—Format of VendorSpecific information element when Type = 0

This mechanism ~~is~~shall be used to carry end-to-end link-independent timing information from the master port to the associated slave port, including preciseOriginTimestamp, rateRatio, correctionField, and other fields of the Follow-Up message, as described in 12.5.1.4. For consistency, all of these fields are packed into the FollowUpInformation field using exactly the same format as used for full-duplex point-to-point links. In other words, the master state machine communicates an entire Follow_Up message (i.e., including all the fields of the common header, see 11.4.2 and 10.6.2, the preciseOriginTimestamp, and all the fields of the Follow_Up information TLV, see 11.4.4) using this mechanism. The Type field, illustrated in Figure 12-7, identifies this use of the OUI or CID within the VendorSpecific information element. Table 12-8 lists values for the Type field.

12.7 Synchronization message interval

12.7.1 General synchronization message interval specification

The mean time interval between successive synchronization messages shall be as specified in 10.7.2.1, 10.7.2.3, and 12.7.2.

12.7.2 Synchronization message interval default value

The default value of initialLogSyncInterval (see 10.7.2.3) is –3. Every port supports the value 127; the port does not send Sync messages when currentLogSyncInterval has this value (see). A port may support other values, except for the reserved values –128 through –125, inclusive, and 124 through 126, inclusive. A port ignores requests (see) for unsupported values.

Processing of the message interval request TLV carried in a Signaling message (see 10.6.4) shall be supported, as specified by the ~~LinkDelay~~SyncIntervalSetting state machine of (see ~~Figure 11-10~~11.2.19), except that: the linkDelayInterval (which is not relevant to IEEE 802.11 ports) is set to –128 by the sender of the Signaling message, the linkDelayInterval is ignored by the receiver, and unsupported values of timeSyncInterval are ignored by the receiver.

NOTE 1—A port that requests (using a Signaling message that contains a message interval request TLV, see 10.6.4 and 11.2.19) that the port at the other end of the attached link set its currentLogSyncInterval to a specific value can determine if the request was honored by examining the logMessageInterval field of a FollowUpInformation contained in the VendorSpecific information element of a subsequent MLME indication primitive.

NOTE 2—The time interval between every pair of adjacent timing or fine timing measurements is not guaranteed to be precisely the same. Some variation is expected, due to factors such as the MAC protocol (e.g., delay in accessing the medium and/or packet retries) and the power state of the associated station. However, timestamp ~~T1~~t1 is not captured when either TIMINGMSMT.request or FINETIMINGMSMT.request is invoked, but only after the action frame resulting from the request is actually transmitted.

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

13.1 Overview

13.1.1 General

This clause specifies the service interface primitives, state machines, and message formats that provide accurate synchronized time across IEEE 802.3 Ethernet passive optical network (EPON) links, through the use of the timing process and measurements specified in 64.2.1.1, ~~and~~ 64.3.2.4, 77.2.1.1, and 76.1.2 of IEEE Std 802.3-20~~08~~12~~, and 77.2.1.1 and 76.1.2 of IEEE Std 802.3av-2009~~. For purposes of this clause, an EPON link is an EPON that contains one optical line terminal (OLT) and associated optical network units (ONUs).

A time-aware system contains at most one ONU, but may contain more than one OLT (i.e., a time-aware system is a clock slave to at most one EPON link, but may be a clock master to more than one EPON link).

13.1.2 Description of the EPON timing process

The timing process in EPON relies on the 32-bit counters (see 64.2.2.2 and 77.2.2.2 of IEEE Std 802.3-20~~08~~12 ~~and 77.2.2.2 of IEEE Std 802.3av-2009~~) at both the OLT and the ONU. The 32-bit counter used by EPON is the LocalClock entity of the time-aware system. These counters increment every time_quantum, which is equal to 16 ns (see 64.2.2.1 and 77.2.2.1 of IEEE Std 802.3-20~~08~~12 ~~and 77.2.2.1 of IEEE Std 802.3av-2009~~). IEEE Std 802.3-20~~08~~12 ~~and IEEE Std 802.3av-2009~~ defines multipoint control protocol (MPCP), which is one of the protocols that enables MAC clients to communicate over a point-to-multipoint optical network. When either the clock master (OLT) or the clock slave (ONU) transmits an MPCP data unit (MPCPDU), its counter value is mapped into the timestamp field. Clauses 64 and 77 of IEEE Std 802.3-20~~08~~12 ~~and Clause 77 of IEEE Std 802.3av-2009~~ specify the EPON timing mechanism.

13.1.3 Best master selection

13.1.3.1 General

An EPON link contains one OLT and the associated ONUs. The OLT is the clock master and the associated ONUs are clock slaves. The OLT initiates the time synchronization as a requester. The ONUs are the responders of the time synchronization. This means that the invocation of BMCA results in the OLT having the port ~~role~~state MasterPort and the ONU having the port ~~role~~state SlavePort (see 10.3.1 and Table 10-1), regardless of the attributes of time-aware systems downstream from the ONU. This behavior is achieved using the acceptable master table feature defined in 17.6 of IEEE Std 1588-2008.

A time-aware system that contains an ONU port shall maintain a configured table, the acceptableMasterTable, and a per-port Boolean variable acceptableMasterTableEnabled. The data type of acceptableMasterTable is AcceptableMasterTable (see 13.1.3.2).

13.1.3.2 AcceptableMasterTable

The AcceptableMasterTable type represents a table of AcceptableMaster entries.

```
struct AcceptableMasterTable{
        UInteger16 maxTableSize;
        UInteger16 actualTableSize;
        AcceptableMaster[actualTableSize] acceptableMaster;
}
```

The maxTableSize member is the maximum size of the AcceptableMasterTable. The actualTableSizeMember is the actual size of the AcceptableMasterTable. The AcceptableMaster array contains a list of AcceptableMaster ports. The value of maxTableSize is implementation specific. actualTableSize shall be less than or equal to maxTableSize.

An AcceptableMasterTable is configurable and may contain a number of AcceptableMaster entries up to maxTableSize.

13.1.3.3 AcceptableMaster

The AcceptableMaster type represents a port that can be considered, in the execution of the BMCA, as a candidate for master.

```
struct AcceptableMaster{
        PortIdentity acceptablePortIdentity;
        UInteger8 alternatePriority1;
}
```

The acceptablePortIdentity member is the PortIdentity of an acceptable master port. The alternatePriority1 member contains an alternate value for the priority1 attribute of the acceptable master port (see 13.1.3.4).

13.1.3.4 Acceptable master table feature

The acceptable master table feature shall modify the operation of the BMCA (see 10.3) as follows:

a) If acceptableMasterTableEnabled for a port is FALSE, the BMCA operates as described in 10.3 and its subclauses.
b) If acceptableMasterTableEnabled for a port is TRUE, then:
   1) The function qualifyAnnounce() of the PortAnnounceReceive state machine (see 10.3.10.2.1) is replaced by the following:

      qualifyAnnounce (rcvdAnnouncePtr): qualifies the received Announce message pointed to by rcvdAnnouncePtr as follows:
      i) if the Announce message was sent by the current time-aware system, i.e., if sourcePortIdentity.clockIdentity (see 10.6.2.2.11 and 8.5.2) is equal to thisClock (see 10.2.3.22), the Announce message is not qualified and FALSE is returned;
      ii) if the stepsRemoved field is greater than or equal to 255, the Announce message is not qualified and FALSE is returned;
      iii) if the sourcePortIdentity of the Announce message is not equal to the sourcePortIdentity of one of the entries of the acceptableMasterTable, FALSE is returned;
      iv) if a path trace TLV is present and one of the elements of the pathSequence array field of the path trace TLV is equal to thisClock (i.e., the clockIdentity of the current time-aware system, see 10.2.3.22), the Announce message is not qualified and FALSE is returned; otherwise, the Announce message is qualified and TRUE is returned. If a path trace TLV is present and the port~~Role~~State of the port is SlavePort, the pathSequence array field of the TLV is copied to the global array pathTrace, and thisClock is appended to pathTrace (i.e., is added to the end of the array).
   2) If the alternatePriority1 member of the AcceptableMaster array element that corresponds to the sourcePortIdentity of a received Announce message is 0, the alternatePriority1 member has no effect on the operation of BMCA.
   3) If the alternatePriority1 member of the AcceptableMaster array element that corresponds to the sourcePortIdentity of a received Announce message is greater than 0, the value of the grandmasterPriority1 field of the Announce message is replaced by the value of alternatePriority1 of this AcceptableMaster array element for use in the invocation of BMCA.

13.1.3.5 Default configuration of acceptable master table feature

The default configuration of the acceptable master table feature for a time-aware system that is attached to an IEEE 802.3 EPON link shall be as follows:

a) If the time-aware system does not contain an ONU port, the default acceptableMasterTable is empty, i.e., the member actualTableSize is 0 and there are no AcceptableMaster array entries. The variable acceptableMasterTableEnabled for each port is set to FALSE.
b) If the time-aware system contains an ONU port, the default acceptableMasterTable contains one element in the AcceptableMaster array. The member actualTableSize is 1. The acceptablePortIdentity of that element is set equal to the portIdentity of the OLT port that the ONU port is attached to, and alternatePriority1 set equal to 244. The variable acceptableMasterTableEnabled for each port is set to TRUE.

NOTE—These default settings ensure that, with the default priority1 values of 8.6.2.1, Table 8-3, used for all time-aware systems, the time-aware system that contains the ONU port will consider Announce messages only from the OLT that the ONU port is attached to when invoking the BMCA. The alternatePriority1 value of 244 ensures that the OLT will be considered better than the ONU in the sense of the BMCA, which will cause the OLT port ~~role~~state to be set to MasterPort and the ONU port ~~role~~state to be set to SlavePort. All other ports of this time-aware system that are not disabled and for which asCapable is TRUE will have port ~~role~~states of either MasterPort or PassivePort. If all time-aware systems downstream from the ONU have priority1 greater than 244, then the port at the other end of each link attached to each non-ONU port that is not disabled and for which asCapable is TRUE will have port ~~role~~states of either SlavePort or Passive port; in this case, the downstream network portions will get their timing through the EPON. However, if a downstream time-aware system has priority1 less than 244, or priority1 equal to 244 and is better than the grandmaster information contained in the Announce message received by the ONU based on other attributes, then the portion of the network that is downstream of the ONU and includes that better time-aware system will get its timing from that better downstream time-aware system. In this case, the endpoints of the link of that network portion attached to the time-aware system that contains the ONU will both have port ~~role~~states of MasterPort, and the ports at each end of the link will send Announce messages. However, the Announce messages sent by the downstream time-aware system will be ignored by the time-aware system that contains the ONU because the sourcePortIdentity of those Announce messages will not be contained in the acceptableMasterTable. The Announce messages sent by the time-aware system that contains the ONU will be used in the invocation of the BMCA at the downstream ~~node~~time-aware system; however, those Announce messages will not reflect the best master because one of the downstream time-aware systems is better.

13.1.4 Time synchronization in EPON

Transmission in the EPON downstream direction (from OLT to ONUs) utilizes time division multiplexing (TDM). In the upstream direction (from ONUs to OLT), time division multiple access (TDMA) is employed. Due to the frame queuing in TDMA, the downstream delay is different from the upstream delay. Asymmetric delay also occurs in the EPON physical layer due to upstream and downstream transmission using different wavelengths. The index of refraction is frequency dependent, which results in the upstream and downstream delays being asymmetric. The accurate time synchronization across the EPON links is operated as follows. It is assumed that the clock master (the OLT) has an accurate synchronized time. The clock master informs the clock slave (the ONU) what the accurate synchronized time will be when the counter of the clock slave reaches a certain value. The information transfer can be accomplished using the organization-specific slow protocol (OSSP) message (see Clause 57 of IEEE Std 802.3-20~~08~~12).

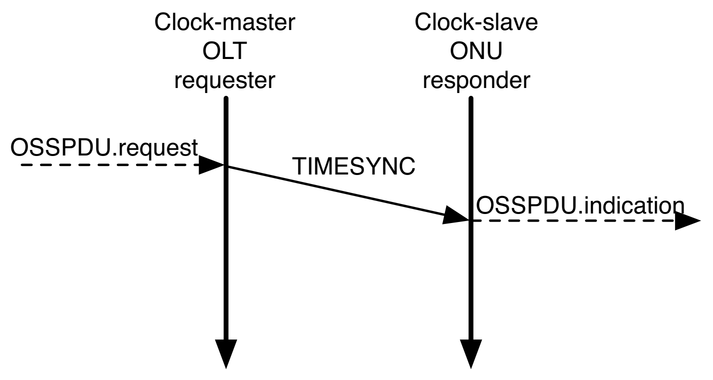


Figure 13-1—IEEE 802.3 EPON time-synchronization interfaces

The following reference process, illustrated schematically in Figure 13-1, will result in the clock slave of an ONU being synchronized to the clock master of the OLT:

a) The clock master selects a value *X* of the local MPCP counter that is used as the timing reference. Any value ~~may~~can be chosen, provided it is relative to the current epoch of the MPCP counter.
b) The clock master calculates the $ToD_{X,i}$ based on $ToD_{X,o}$ using Equation (13-1).

$$ToD_{X,\,i} = ToD_{X,\,o} + RTT_i \cdot \frac{\mathrm{ndown}}{(\mathrm{nup} + \mathrm{ndown})} \cdot \mathrm{rateRatio} \quad (13\text{-}1)$$

where $ToD_{X,i}$ is the synchronized time when the MPCP counter at the clock slave *i* reaches a value equal to the timestamp X minus the *onuLatencyFactor*; $ToD_{X,o}$ is the synchronized time when the MPCP counter at the clock master reaches a value equal to the timestamp X plus the *oltLatencyFactor*; RTT_i is the round-trip time measured by the clock master for clock slave *i*, i.e., ONU *i*; nup is the effective refraction index of the light propagating in the upstream channel; ndown is the effective refraction index of the light propagating in the downstream channel; and rateRatio is the rateRatio member of the most recently received MDSyncSend structure. The *onuLatencyFactor* and *oltLatencyFactor* are given in Equation (13-2) and Equation (13-3), respectively. The impact of the worst-case variation in the transmission wavelength for the clock master and clock slave transmitters is examined in VII of ITU-T G.984.3, Amendment 2.

$$\begin{aligned}&onuLatencyFactor = onuIngressLatency - \\&(onuIngressLatency + onuEgressLatency) \cdot \frac{\mathrm{ndown}}{(\mathrm{nup} + \mathrm{ndown})} \cdot \mathrm{rateRatio}\end{aligned} \quad (13\text{-}2)$$

$$\begin{aligned}&oltLatencyFactor = oltEgressLatency - \\&(oltIngressLatency + oltEgressLatency) \cdot \frac{\mathrm{ndown}}{(\mathrm{nup} + \mathrm{ndown})} \cdot \mathrm{rateRatio}\end{aligned} \quad (13\text{-}3)$$

c) The clock master sends the pair of values (X, $ToD_{X,i}$) to clock slave i via the downstream TIMESYNC message.

NOTE—After the clock slave receives the downstream TIMESYNC message, it can compute the synchronized time, *ToD*, when the value of the local MPCP counter is equal to *S*; *ToD* is given by the following equation:

$$ToD = ToD_{X,\,i} + [(S - X) \bmod (2^{32})](16\text{ ns}) \cdot \text{rateRatio}$$

where (A) mod (B) is A modulo B.

The OSSP message is a general message (see 3.8), analogous to Follow_Up. Note that the preceding synchronized time values correspond to timestamps that are referenced to the MAC control sublayer. Both the clock master and clock slave are responsible for compensating their processing delays (e.g., the ingressLatency and egressLatency, as described in 8.4.3). RTT_i is measured using MPCPDU timestamps, inserted into the frame structure as specified by 64.2.1.1 and 77.2.1.1 of IEEE Std 802.3-20~~08~~12~~ and 77.2.1.1 of IEEE Std 802.3av-2009~~.

13.2 Message attributes

13.2.1 Message class

The TIMESYNC message is a general message (see 3.8 and 8.4.2.2). It is transmitted in the downstream direction, from OLT to ONU.

13.3 Message format

13.3.1 TIMESYNC message

13.3.1.1 General TIMESYNC message specifications

The fields of the body of the TIMESYNC message shall be as specified in Table 13-1 and 13.3.1.2 and its subclauses.

13.3.1.2 TIMESYNC message field specifications

13.3.1.2.1 Destination address (Octet6)

The destination address field is equal to 01-80-C2-00-00-02~~0x0180C2000002~~ (see 57A.3 of IEEE Std 802.3-20~~08~~12).

13.3.1.2.2 Source address (Octet6)

The source address field is the individual MAC address associated with the port through which the TIMESYNC message is transmitted (see 57B.1.1 of IEEE Std 802.3-20~~08~~12).

13.3.1.2.3 Length/Type (Octet2)

The value of this field is equal to 0x8809 (see 57A.4 of IEEE Std 802.3-20~~08~~12).

13.3.1.2.4 Subtype (Octet)

The value of this field is equal to 0x0A (see 57A.4 of IEEE Std 802.3-20~~08~~12).

Table 13-1—TIMESYNC message fields

Bits: ~~8~~7 ~~7~~6 ~~6~~5 ~~5~~4 ~~4~~3 ~~3~~2 ~~2~~1 ~~1~~0	Octets	Offset
Destination Address	6	0
Source Address	6	6
Length/Type	2	12
Subtype	1	14
~~Organizationally Unique Identifier~~OUI or CID	3	15
Message Identifier	2	18
X	4	20
$ToD_{X,i}$	10	24
sourcePortIdentity	10	34
logMessageInterval	1	44
rateRatio	8	45
gmTimeBaseIndicator	2	53
lastGmPhaseChange	12	55
scaledLastGmFreqChange	4	67
domainNumber	1	71
FCS	4	72~~1~~

13.3.1.2.5 ~~Organizationally Unique Identifier~~OUI or CID (Octet3)

This field contains the ~~Organizationally Unique Identifier (OUI) to~~OUI or CID that identif~~y~~ies the Organization-Specific Data. The value is 00-80-C2, i.e., the OUI assigned to IEEE 802.1~~ is 0x0080C2~~.

13.3.1.2.6 Message identifier (Octet2)

This field is the TIMESYNC message identifier. The value of this field is 1.

13.3.1.2.7 *X* (UInteger32)

The *X* field is the selected timestamp that will be used as the timing reference as specified in 13.1.4.

13.3.1.2.8 $ToD_{X,i}$ (Timestamp)

$ToD_{X,i}$ is the synchronized time when the MPCP counter at the clock slave *i* reaches a value equal to *X* minus the *onuLatencyFactor* (see 13.1.4). *X* is carried in the respective TIMESYNC message. Synchronization of the MPCP clock is described in detail in 64.2.1.1 and 77.2.1.1 in IEEE Std 802.3-20~~08~~12, for 1G-EPON ~~and in 77.2.1.1 in IEEE Std 802.3av-2009 for~~ 10G-EPON, respectively.

NOTE—Any subnanosecond portion of synchronized time (in this case, time of day), normally transported in a correction field (see 10.2.2.1.2, 10.2.2.2.2, and 10.2.2.3.5), is not transported over EPON.

13.3.1.2.9 sourcePortIdentity (PortIdentity)

This field is specified as the sourcePortIdentity member of the MDSyncSend structure most recently received from the PortSync entity of the OLT (see 10.2.2.1.4).

13.3.1.2.10 logMessageInterval (Integer8)

This field is specified as the logMessageInterval member of the MDSyncSend structure most recently received from the PortSync entity of the OLT (see 10.2.2.1.5). It is the value of the currentLogSyncInterval for this port (see 10.7.2.3).

13.3.1.2.11 rateRatio (Double)

This field is specified as the rateRatio member of the MDSyncSend structure most recently received from the PortSync entity of the OLT (see 10.2.2.1.8).

13.3.1.2.12 gmTimeBaseIndicator (UInteger16)

This field is specified as the gmTimeBaseIndicator member of the MDSyncSend structure most recently received from the PortSync entity of the OLT (see 10.2.2.1.9).

13.3.1.2.13 lastGmPhaseChange (ScaledNs)

This field is specified as the lastGmPhaseChange member of the MDSyncSend structure most recently received from the PortSync entity of the OLT (see 10.2.2.1.10).

13.3.1.2.14 scaledLastGmFreqChange (Integer32)

The value of scaledLastGmFreqChange is the fractional frequency offset of the current grandmaster relative to the previous grandmaster, at the time that the current grandmaster became grandmaster, or relative to itself prior to the last change in gmTimeBaseIndicator, multiplied by 2^{41} and truncated to the next smaller signed integer. The value is obtained by multiplying the lastGmFreqChange member of MDSyncSend (see 10.2.2.1) whose receipt causes the MD entity to send the TIMESYNC message by 2^{41}, and truncating to the next smaller signed integer.

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

13.3.1.2.15 domainNumber

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

13.3.1.2.16 FCS (Octet4)

This field is the Frame Check Sequence (see 57B.1.1 of IEEE 802.3-20~~08~~12).

13.4 Determination of asCapable

The default value of the per-port, per-domain global variable asCapable shall be~~set to~~ TRUE.

The per-port, per-domain global variable asCapable shall be set to TRUE if the value of neighborGptpCapable for this port is TRUE.

NOTE—The above conditions ensure backward compatibility with the 2011 edition of this standard. A time-aware system that is compliant with the 2011 edition of this standard will not process the gPTP capable TLV, and asCapable will be determined as specified in the 2011 edition. A time-aware system compliant with the current edition of this standard that is attached, via an EPON link, to a node compliant with the 2011 edition of this standard will not receive Signaling messages that contain the gPTP capable TLV and will not set neighborGptpCapable to TRUE. However, the above ensures that asCapable for this port and domain (i.e., domain 0) will still be set in a manner consistent with that of the 2011 edition of this standard, because the default value of asCapable is TRUE in that edition.

<<Editor's note: The TSN TG should review the above to determine whether this is the desired behavior in determining asCapable for an EPON link. Comments are requested.>>

13.5 Layering for IEEE 802.3 EPON links

The MD entity is media-dependent and is responsible for translating the media-independent layer to media-dependent PDUs or primitives as necessary for communicating synchronized time over the EPON link from the OLT to a single ONU. This implies that if one OLT port is associated with multiple ONUs, it will require one IEEE 802.1AS PortSync entity and one MD entity per associated ONU. The OSSPDU primitives are used to communicate synchronized time information. Figure 13-2 illustrates how the MD entity interacts with the OSSP sublayer.

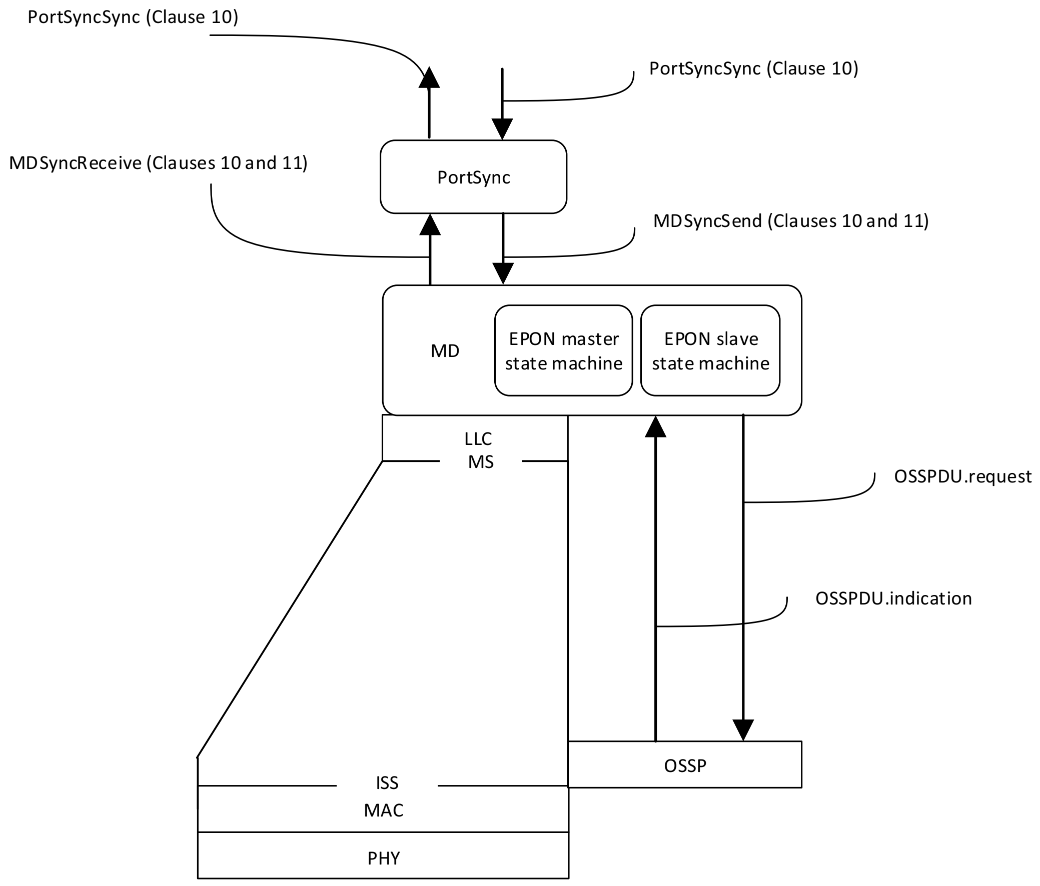


Figure 13-2—IEEE 802.3 EPON interface model

13.6 Service interface definitions

13.6.1 OSSPDU.request

13.6.1.1 General

This service interface primitive is generated periodically by the MD entity of the clock master every sync interval (see 10.7.2.1). It triggers transmission of a TIMESYNC message from the clock master to the clock slave. The values of the parameters of the primitive are sent to the clock slave via the TIMESYNC message.

13.6.1.2 OSSPDU.request parameters

OSSPDU.request {
X
$ToD_{X,i}$
sourcePortIdentity
logMessageInterval
rateRatio
gmTimeBaseIndicator

```
        lastGmPhaseChange
        scaledLastGmFreqChange
        domainNumber
}
```

The parameter definitions are as follows:

13.6.1.3 *X* (Integer32)

The *X* field is the selected timestamp that will be used as the timing reference as specified in 13.1.4.

13.6.1.4 $ToD_{X,i}$ (Timestamp)

$ToD_{X,i}$ is the synchronized time when the MPCP counter at the clock slave *i* reaches a value equal to *X* minus the *onuLatencyFactor* (see 13.1.4). *X* is carried in the respective TIMESYNC message. Synchronization of the MPCP clock is described in detail in 64.2.1.1 and 77.2.1.1 in IEEE Std 802.3-20~~08~~12, for 1G-EPON and ~~in 77.2.1.1 in IEEE Std 802.3av-2009 for~~ 10G-EPON, respectively.

13.6.1.5 sourcePortIdentity (PortIdentity)

This parameter identifies the sourcePortIdentity value for this port (see 13.3.1.2.9).

13.6.1.6 logMessageInterval (Integer8)

This parameter identifies the currentLogSyncInterval value for this port (see 13.3.1.2.10).

13.6.1.7 rateRatio (Double)

This parameter identifies the rateRatio value for this port (see 13.3.1.2.11).

13.6.1.8 gmTimeBaseIndicator (UInteger16)

This parameter identifies the gmTimeBaseIndicator value for this port (see 13.3.1.2.12).

13.6.1.9 lastGmPhaseChange (ScaledNs)

This parameter identifies the lastGmPhaseChange value for this port (see 13.3.1.2.13).

13.6.1.10 scaledLastGmFreqChange (Integer32)

This parameter identifies the scaledLastGmFreqChange value for this port (see 13.3.1.2.14).

13.6.1.11 domainNumber

This parameter identifies the domainNumber for this instance of gPTP (see 13.3.1.2.15).

13.6.1.12 When generated

This primitive is generated by the clock master every $2^{\text{currentLogSyncInterval}}$ seconds when it is in the MASTER state, as the first phase of synchronized time information transfer.

13.6.1.13 Effect of receipt

Upon receipt of this primitive, a TIMESYNC message is enqueued for transmission.

NOTE—Arrival of the TIMESYNC message at the ONU after the selected time *X* does not impede proper operation of the synchronization mechanism defined in this clause.

13.6.2 OSSPDU.indication

13.6.2.1 General

This service interface primitive is generated on receipt of a TIMESYNC message by the responder, and provides the values contained in the corresponding OSSPDU.request primitive to the clock slave.

13.6.2.2 OSSPDU.indication parameters

```
OSSPDU.indication {
        X
        ToD_{X,i}
        sourcePortIdentity
        logMessageInterval
        rateRatio
        gmTimeBaseIndicator
        lastGmPhaseChange
        scaledLastGmFreqChange
}
```

The parameters of the OSSPDU.indication are set equal to the corresponding fields of the most recently received TIMESYNC message. Their definitions are given in 13.3.1.2.7 through 13.3.1.2.15, respectively.

13.6.2.3 When generated

This primitive is generated by the receipt of a TIMESYNC message during the phase of synchronized time information transfer.

13.6.2.4 Effect of receipt

Upon receipt, the OSSPDU.indication parameters are used by the MD entity to compute the parameters of the MDSyncReceive structure that will be transmitted to the PortSync entity of this port.

13.7 MD entity global variables

13.7.1 ***RTT_i*****:** is used only by the OLT MD entity. RTT_i is the RTT between the clock master and clock slave. The data type for RTT_i is UInteger32.

NOTE—RTT is measured and updated by the MPCP using the mechanism specified in IEEE Std 802.3-20~~08~~12~~ and IEEE Std 802.3av-2009~~, and stored in RTT_i when measured and updated. RTT_i is not used by the ONU, and is set to zero in an ONU MD entity.

13.8 State machines

13.8.1 Requester state machine

13.8.1.1 Function

This state machine generates and consumes primitives, at the requester, used to provide accurate synchronized time across EPON links to the responder.

13.8.1.2 State machine variables

The following variables are used in the state diagram of 13.8.1.4:

13.8.1.2.1 ndown: the effective index of the light propagating in the downstream channel. The data type for ndown is Double.

13.8.1.2.2 nup: the effective index of the light propagating in the upstream channel. The data type for ndown is Double.

13.8.1.2.3 rcvdMDSync:a Boolean variable that notifies the current state machine when an MDSyncSend structure is received. This variable is reset by the current state machine.

13.8.1.2.4 rcvdMDSyncPtr: a pointer to the received MDSyncSend structure.

13.8.1.2.5 registered: a Boolean variable that indicates an ONU has registered to EPON.

13.8.1.2.6 $\boldsymbol{ToD_{X,i}}$: the synchronized time when the MPCP counter at the clock slave *i* reaches a value equal to *X* (see 13.8.1.2.7) minus the *onuLatencyFactor* (see 13.1.4). The data type for $ToD_{X,i}$ is Timestamp.

13.8.1.2.7 $\boldsymbol{ToD_{X,o}}$: the synchronized time when the MPCP counter at the clock master reaches a value equal to *X* (see 13.8.1.2.7) plus the *oltLatencyFactor* (see 13.1.4). The data type for $ToD_{X,o}$ is Timestamp.

13.8.1.2.8 *X*: the value of the timestamp [see 13.1.4a)] that is selected as the reference time. The data type for *X* is UInteger32.

13.8.1.3 State machine functions

The following function is used in the state diagram of 13.8.1.4:

13.8.1.3.1 setToDXo(): computes the state machine variable $ToD_{X,o}$ (see 13.8.1.2.7) as the sum of:

a) The preciseOriginTimestamp member of the most recently received MDSyncSend structure,
b) The followUpCorrectionField of the most recently received MDSyncSend structure, and
c) The quantity

$$\mathrm{rateRatio} \times (X \times (16\ \mathrm{ns}) - \mathrm{upstreamTxTime}) \qquad (13\text{-}4)$$

where rateRatio and upstreamTxTime are the rateRatio and upstreamTxTime members, respectively, of the most recently received MDSyncSend structure, and *X* is defined in 13.8.1.2.8 (see 13.8.1.2.7).

13.8.1.4 State diagram

The requester state machine shall implement the function specified by the state diagram in Figure 13-3, the local variables specified in 13.8.1.2, the service interface primitives specified in 13.6, the structure specified in 10.2.2.1, the message specified in 13.3, and the relevant global variables specified in 10.2.4 and 13.7. The state machine receives an MDSyncSend structure from the PortSyncSyncSend state machine of the PortSync entity of this port and transmits an OSSPDU.request primitive to cause a TIMESYNC message to be sent to the responder (ONU).

<<Editor's note: In Figure 13-3, portEnabled is changed to portOper, and pttPortEnabled is changed to ptpPortEnabled.>>

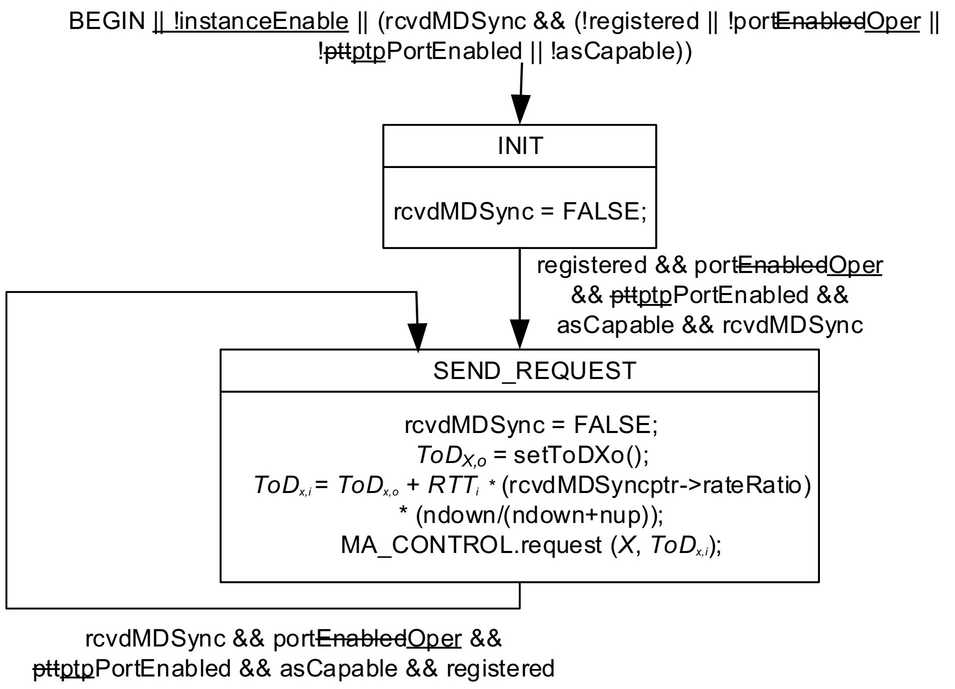

BEGIN || !instanceEnable || (rcvdMDSync && (!registered || !port~~Enabled~~Oper || !~~ptt~~ptpPortEnabled || !asCapable))
INIT
rcvdMDSync = FALSE;
registered && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && asCapable && rcvdMDSync
SEND_REQUEST
rcvdMDSync = FALSE;
$ToD_{X,o}$ = setToDXo();
$ToD_{x,i} = ToD_{x,o} + RTT_i$ * (rcvdMDSyncptr->rateRatio) * (ndown/(ndown+nup));
MA_CONTROL.request (X, $ToD_{x,i}$);
rcvdMDSync && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && asCapable && registered

Figure 13-3—State machine for IEEE 802.3 EPON requester

13.8.2 Responder state machine

13.8.2.1 Function

This state machine responds to EPON-specific primitives generated by receipt of a TIMESYNC message from the requester.

13.8.2.2 State machine variables

The following variables are used in the state diagram of 13.8.1.4:

13.8.2.2.1 rcvdOSSPDUind: a Boolean variable that notifies the responder state machine when a TIMESYNC message is received and the OSSPDU.indication primitive is generated.

13.8.2.2.2 txMDSyncReceivePtr: a pointer to a structure whose members contain the values of the parameters of an MDSyncReceive structure to be transmitted.

13.8.2.2.3 rcvdOSSPDUptr: a pointer to a structure whose members contain the values of the parameters of the OSSPDU.indication primitive whose receipt is indicated by rcvdOSSPDUind (see 13.8.2.2.1).

13.8.2.3 State machine functions

The following functions are used in the state diagram of 13.8.2.4:

13.8.2.3.1 setMDSyncReceive(): creates an MDSyncReceive structure (see 10.2.2.2) using members of the structure pointed to by rcvdOSSPDUptr (see 13.8.2.2.3), and returns a pointer to this structure. The members of this structure are set as follows:

a) followUpCorrectionField is set equal to 0,
b) sourcePortIdentity is set equal to an 8-byte clockIdentity plus a 2-byte portNumber. The 8-byte clockIdentity is generated by mapping the 6 byte Source Address (see 13.3.1.2.2) of the most recently received TIMESYNC message, which is an EUI-48, to an EUI-64 format (see 8.5.2.2.1). The 2-byte portNumber is set equal to 1,
c) logMessageInterval is set equal to the logMessageInterval of the most recently received TIMESYNC message (see 13.3.1.2.10),
d) preciseOriginTimestamp is set equal to the $ToD_{X,i}$ field of the most recently received TIMESYNC message (see 13.3.1.2.8),
e) rateRatio is set to the rateRatio of the most recently received TIMESYNC message (see 13.3.1.2.11),
f) upstreamTxTime is set equal to *X* multiplied by 16 ns, where *X* is the value of the *X* field of the most recently received TIMESYNC message (see 13.3.1.2.7),
g) gmTimeBaseIndicator is set equal to the gmTimeBaseIndicator of the most recently received TIMESYNC message (see 13.3.1.2.12),
h) lastGmPhaseChange is set equal to the lastGmPhaseChange of the most recently received TIMESYNC message (see 13.3.1.2.13),~~ and~~
i) lastGmFreqChange is set equal to the scaledLastGmFreqChange of the most recently received TIMESYNC message (see 13.3.1.2.14), divided by 2^{41}, and
j) domainNumber is set equal to the domainNumber of the most recently received TIMESYNC message (see 13.3.1.2.15).~~.~~

13.8.2.3.2 txMDSyncReceive (txMDSyncReceivePtr): transmits an MDSyncReceive structure to the PortSyncSyncReceive state machine of the PortSync entity of this port.

13.8.2.4 State diagram

The responder state machine shall implement the function specified by the state diagram in Figure 13-4, the local variables specified in 13.8.2.2, the functions specified in 13.8.2.3, the service interface primitives specified in 13.6, the structure specified in 10.2.2.2, the message specified in 13.3, and the relevant global variables specified in 10.2.4 and 13.7. The state machine receives an OSSPDU.indication primitive in response to its having received a TIMESYNC message from the requester (OLT), and transmits an MDSyncReceive structure to the PortSync entity of this port.

<<Editor's note: In Figure 13-4, portEnabled is replaced by portOper, and pttPortEnabled is replaced by ptpPortEnabled.>>

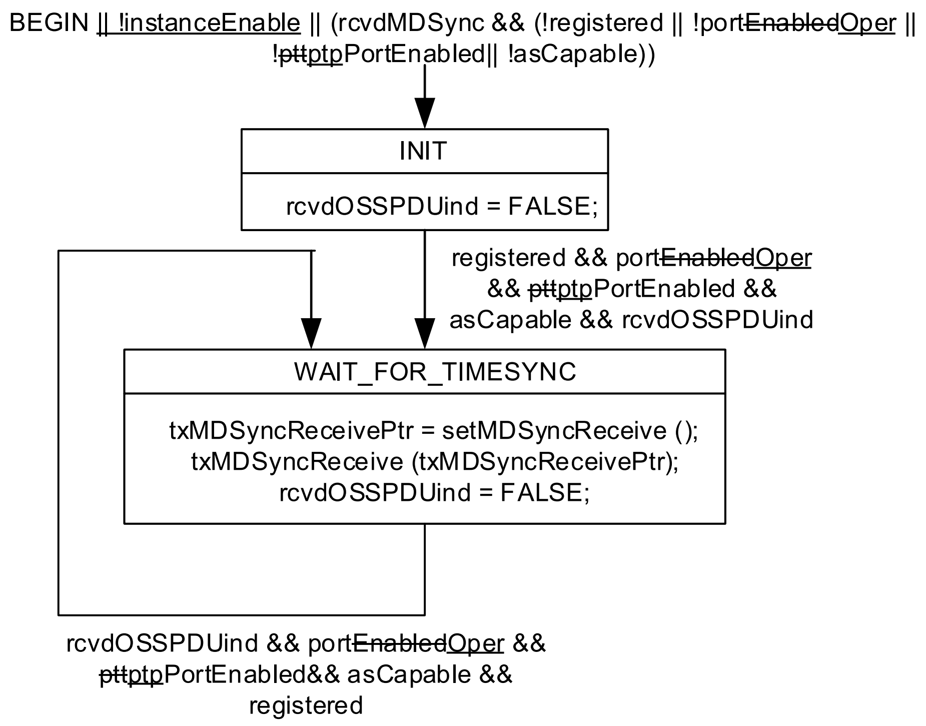

BEGIN || !instanceEnable || (rcvdMDSync && (!registered || !port~~Enabled~~Oper || !~~ptt~~ptpPortEnabled|| !asCapable))
INIT
rcvdOSSPDUind = FALSE;
registered && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled && asCapable && rcvdOSSPDUind
WAIT_FOR_TIMESYNC
txMDSyncReceivePtr = setMDSyncReceive ();
txMDSyncReceive (txMDSyncReceivePtr);
rcvdOSSPDUind = FALSE;
rcvdOSSPDUind && port~~Enabled~~Oper && ~~ptt~~ptpPortEnabled&& asCapable && registered

Figure 13-4—State machine for IEEE 802.3 EPON responder

13.9 Message transmission intervals

13.9.1 General interval specification.

The mean time interval between successive TIMESYNC messages shall be as specified in 10.7.2.1, 10.7.2.3, and 13.9.2.

13.9.2 TIMESYNC message transmission interval default value

The default value of initialLogSyncInterval (see 10.7.2.4) is –3. Every port supports the value 127; the port does not send TIMESYNC messages when currentLogSyncInterval has this value. A port may support other values, except for the reserved values –128 through –125, inclusive, and 124 through 126, inclusive.

Processing of the message interval request TLV carried in a Signaling message (see 10.6.4) shall be supported, as specified by the ~~LinkDelay~~SyncIntervalSetting state machine of (see 11.2.19 Figure 11-10), except that: the LinkDelayInterval (which is not relevant to IEEE 802.3 EPON ports) is set to 128 by the sender of the Signaling message, the LinkDelayInterval is ignored by the receiver, and unsupported values of timeSyncInterval are ignored by the receiver.

14. Timing and synchronization management

[Editor's note: In accordance with comment # 32 against D4.5, the order of datasets under portList[] and under cmldsLinkPortList[] was changed so that corresponding datasets in each place would be first, and would be in the same order. Deletions are not shown via track changes, but insertions are shown via track changes. The following changes were made:

1) The descriptionPortDS was moved so it would come before the portStatisticsDS

2) The cmldsAsymmetryMeasurementModeDS was moved so it would come after the cmldsPortStatisticsDS.

These changes were made in the hierarchy and lettered list in 14.1, and in the order of the subclauses where these datasets are described.]

14.1 General

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

<<Editor's note: The text below is based on 8.1.4.1 of the 1588-Rev WG ballot draft, in accordance with the resolution of comment #19 against D4.2.>>

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.

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

<<Editor's note: The order of the data sets in the hierarchy below is taken from 8.1.4.1 of the 1588-Rev WG ballot draft. The subclauses of this clause (clause 14) have been re-ordered to correspond to the ordering in the hierarchy.>>

<<Editor's note: The data set names in 802.1AS, when written out fully, contain the word 'Parameter.' This word is generally not used in 1588 when writing data set names out fully (e.g., 1588 refers to the 'default data set' and also does not capitalize the words). However, 1588 also generally refers to 'defaultDS', i.e., the full name is not used very often. For now, we retain the convention of 802.1AS. However, for consistency, 'Port Parameter Statistics' is changed to 'Port Parameter Statistics Data Set.'

Comments are requested on whether we should follow 1588 instead (any comments should be very specific on what is desired).>>

<<Editor's note: In accordance with discussion in the June 12, 2017 802.1AS-Rev call: (a)relevbant members of the cmldsLinkPortDS and cmldsLinkPortStatisticsDS are also included in the portDS and portStatisticsDS, respectively, for the case where instance-specific peer delay is used; and (b) the cmldsDescriptionLinkPortDS is eliminated, as its only member is profileIdentifier, which is not relevant for CMLDS. However, it was realized that members of the cmldsAsymmetryMeasurementModeDS and cmldsEponDS need to be included in instance-specific datasets, for the case where CMLDS is not used. These members are included in the portDS, as that is where they were previously (i.e., in 802.1AS-2011 and/or earlier drafts of 802.1AS-Rev). Should these instance-specific members be in the portDS, or in their own datasets as is the case for CMLDS?>>

<<Editor's note: Table 14-9 and Table 14-15 are identical; they describe the pdelayTruncatedTimestampsArray. The reason there are 2 such tables is that this array is a member of both the portDS and cmldsLinkPortDS. However, only one such table is needed.

Normally, the first occurrence would be used; however, in this case it is the second occurrence that is most useful. Should the first ocurrence be referenced, or the second (and the occurrence that is not referenced eliminated)?>>

A) instanceList[]
  a) defaultDS
  b) currentDS
  c) parentDS
  d) timePropertiesDS
  e) pathTraceDS
  f) acceptableMasterTableDS
  g) portList[]
    1) portDS
    2) descriptionPortDS
    3) portStatisticsDS
    4) acceptableMasterPortDS
B) commonServices
  a) commonMeanLinkDelayService
    1) cmldsDefaultDS
    2) cmldsLinkPortList[]
      i) cmldsLinkPortDS
      ii) cmldsLinkPortStatisticsDS
      iii) cmldsAsymmetryMeasurementModeDS
      iv) cmldsEponDS
  b) 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 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.

Follolwing 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 only one common service, namely the Common Mean Link Delay Service, and the corresponding sub-structure is the commonMeanLinkDelayService structure. The item “future common services can follow” is a placeholder for any common serices that might be defined in the future. The commonMeanLinkDelayService structure contains the data sets and lists that are needed by the Common Mean Link Delay Service.

<<Editor’s note: Based on a recent discussion with P1588 members of the Editing Subcommittee, the the hierarchy above has reverted to the scheme of D4.4, which had an overall commonServices structure that contained a set of substructures, one for each common service. The first such substructure was the commonMeanLinkDelayService substructure, and the “future common services can follow” placeholder followed this.>>

The commonMeanLinkDelayService structure contains the cmldsLinkPortList, which is a list of CMLDS logical ports, i,e, Link Ports (see 11.2.15), of the time-aware system that will run the common service. The CMLDS must must be implemented (i.e., a CMLDS executable must be present) on every physical port for which there is 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 logical 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 logical port that invokes the CMLDS (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 cmldsLinkPortList index and cmldsLinkPortDS.portIdentity.portNumber values for a logical port of the Common Mean Link Delay Service will not necessarily be the same. CMLDS logical 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 port of the time-aware system.

It must be possible for a PTP Instance to determine which Link Port it must use when it obtains information provieded by the Common Mean Link Delay Service. The means for doing this is outside the scope of this standard.

NOTE—For example, one possible way of providing the capability for a logical port of a PTP Instance to determine the corresponding Link Port of the Common Mean Link Delay Service is to use a set of indices for this service and all the PTP Instances that use the service. Specifically, the CMLDS would be invoked on every Link port of this service that corresponded to a logical port of at least one PTP Instance. An index for the cmldsLinkPortList would be created, and this index would have a unique value for each Link Port of the service. The portList for each PTP Instance would have an index whose value for a logical port of that instance would would be the same as the value of the index for the corresponding CMLDS Link Port. It is expected that corresponding logical ports of a PTP Instance and the CMLDS would reside on the same physical port, because the delay, neighbor rate ratio, and other properties of the physical link attached to the physical port are the same for all the logical ports that reside on that physical port. As an example, suppose that a time-aware relay has 7 physical ports and two PTP Instances. Let the CMLDS be invoked on all the ports. Let the cmldsLinkPortList have an index whose range is a, b, c, d, e, f, g, with corresponding portNumbers 1, 2, 3, 4, 5, 6, 7, respectively. Let the first PTP Instance, i.e., PTP Instance 1, use the physical ports corresponding to a, b, and c, with portNumbers 1, 2, and 3, respectively. Let the second PTP Instance, i.e., PTP Instance 2, use the physical ports corresponding to c, d, and e, with portNumbers 1, 2, and 3, respectively. Then, if the portList index for PTP Instance 1 takes on values in the range a, b, c, and the portList index for PTP Instance 2 takes on values in the range c, d, e, the indices can be used by each PTP Instance to determine the Common Mean Link Delay Service portNumber that corresponds to the portNumber of any of its logical ports. For example, if PTP Instance 2 wanted to determine the CMLDS Link Port portNumber that corresponds to its portNumber 2, it would note that portNumber 2 has the portList index d, and that this index of the cmldsPortList has portNumber 4.

NOTE—This hierarchy is intended to support a wide variety of time-aware system implementations. Some examples include a) a time-aware system containing four time-aware relays, each of which use the same physical ports, but different domainNumber values, and b) 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.

The objects that comprise this management resource are as follows:

a) The Default Parameter Data Set (defaultDS in the above hierarchy, see Table 14-1), which represents the native capabilities of a time-aware system, i.e., a ~~Bridge~~time-aware relay or a~~n~~ time-aware end station;
b) The Current Parameter Data Set (currentDS in the above hierarchy, see Table 14-2), which represents the position of a local system and other information, relative to the grandmaster;
c) The Parent Parameter Data Set (parentDS in the above hierarchy, see Table 14-3), which represents capabilities of the up-stream system, toward the grandmaster, as measured at a local system;
d) The Time Properties Parameter Data Set (timePropertiesDS in the above hierarchy, see Table 14-4), which represents capabilities of the grandmaster, as measured at a local system;
e) The Path Trace Parameter Data Set (pathTraceDS in the above hierarchy, see Table 14-5), which represents the current path trace information (see 10.3.8.23) available at the time-aware system;

f) The Acceptable Master Table Parameter Data Set (acceptableMasterTableDS in the above hierarchy, see Table 14-6), which represents the acceptable master table used when an EPON port is present in a time-aware system;
g) The Port Parameter Data Set (portDS in the above hierarchy, see Table 14-10), which represents time-aware capabilities at a given ~~Bridge~~time-aware relay or time-aware end station port;
h) The Description Port Parameter Data Set (descriptionPortDS in the above hierarchy, see Table 14-11), which contains the profileIdentifier for this PTP profile as specified in F.1;
i) The Port Parameter Statistics Data Set (portStatisticsDS in the above hierarchy, see Table 14-6), which represent statistics and counters associated with time-aware capabilities at a given ~~Bridge~~time-aware relay or time-aware end station port;~~ and~~
j) ~~The Acceptable Master Table Parameter Data Set (Table 14-6), which represents the acceptable master table used when an EPON port is present in a time-aware system.~~
k) The Acceptable Master Port Parameter Data Set (acceptableMasterPortDS in the above hierarchy, see Table 14-13), which represents the capability to enable/disable the acceptable master table feature on a port;
l) The Common Mean Link Delay Service Default Parameter Data Set (cmldsDefaultDS in the above hierarchy, see Table 14-14), which describes the per-time-aware-system attributes of the Common Mean Link Delay Service;
m) The Common Mean Link Delay Service Link Port Parameter Data Set (cmldsLinkPortDS, see Table 14-16), which represents time-aware Link Port capabilities for the Common Mean Link Delay Service of a time-aware system;
n) The Common Mean Link Delay Service Description Link Port Data Set (cmldsDescriptionPortDS, see Table 14-17), which contains the profileIdentifier for this PTP profile as specified in F.1;
o) The Common Mean Link Delay Service Link Port Parameter Statistics Data Set (cmldsPortStatisticsDS, see Table 14-18), which represents statistics and counters associated with Link Port capabilities at a given time-aware system;
p) The Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set (cmldsAsymmetryMeasurementDS, see Table 14-19), which represents the capability to enable/disable the Asymmetry Compensation Measurement Procedure on a port (see Annex G); and
q) The Common Mean Link Delay Service EPON Parameter Data Set (cmldsEponDS, see Table 14-20), which describes the per-port attributes of the Common Mean Link Delay Service that are specific to IEEE 802.3 EPON links.

NOTE—The Port Parameter Data Set, ~~and the Port Parameter Statistics~~Port Parameter Statistics Data Set, Description Port Data Set, and acceptableMasterPortDS correspond to a logical PTP port of a PTP instance; a ~~Bridge~~time-aware relay or time-aware end station physical port ~~may~~can contain one or more logical ports (see 8.5.1). For example, a ~~bridge~~time-aware relay physical port can be connected to a full-duplex, point-to-point link that contains one logical port. As another example, a ~~bridge~~time-aware relay physical port can be connected to a CSN link that contains more than one logical port.

14.2 Default Parameter Data Set

The Default Parameter Data Set represents the native capabilities of a time-aware system, i.e., a ~~Bridge~~time-aware relay~~,~~ or a~~n~~ time-aware end station.

14.2.1 clockIdentity

The value is the clockIdentity, see 8.5.2.2, of the local clock.

14.2.2 numberPorts

The value is the number of ports of the time-aware system, see 8.6.2.8. For an end station the value is 1.

14.2.3 clockQuality

This managed object is a structure whose data type is ClockQuality (see 6.4.3.8).

14.2.3.1 clockQuality.clockClass

The value is the clockClass of the time-aware system, which implements the clockClass specifications of 8.6.2.2.

14.2.3.2 clockQuality.clockAccuracy

The value is the clockAccuracy of the time-aware system, which implements the clockAccuracy specifications of 8.6.2.3.

14.2.3.3 clockQuality.offsetScaledLogVariance

The value is the offsetScaledLogVariance of the time-aware system, which implements the offsetScaledLogVariance specifications of 8.6.2.4.

14.2.4 priority1

The value is the priority1 attribute of the time-aware system, see 8.6.2.1.

14.2.5 priority2

The value is the priority2 attribute of the time-aware system, see 8.6.2.5.

14.2.6 gmCapable

The value is TRUE if the time-aware system is capable of being a grandmaster, and FALSE if the time-aware system is not capable of being a grandmaster.

14.2.7 currentUtcOffset

The value is the offset between TAI and UTC, relative to the ClockMaster entity of this time-aware system. It is equal to the global variable sysCurrentUtcOffset (see 10.3.8.18). The value is in units of seconds.

The default value is selected as follows:

a) The value is the value obtained from a primary reference if the value is known at the time of initialization, else
b) The value is the current number of leap seconds, see 8.2.3, when the time-aware system is designed.

14.2.8 currentUtcOffsetValid

The value is TRUE if the currentUtcOffset, relative to the ClockMaster entity of this time-aware system, is known to be correct. It is equal to the global variable sysCurrentUtcOffsetValid (see 10.3.8.14).

The default value is TRUE if the value of currentUtcOffset is known to be correct, otherwise it is set to FALSE.

14.2.9 leap59

A TRUE value indicates that the last minute of the current UTC day, relative to the ClockMaster entity of this time-aware system, will contain 59 s. It is equal to the global variable sysLeap59 (see 10.3.8.13).

The value is selected as follows:

a) The value is obtained from a primary reference if known at the time of initialization, else
b) The value is set to FALSE.

14.2.10 leap61

A TRUE value indicates that the last minute of the current UTC day, relative to the ClockMaster entity of this time-aware system, will contain 61 s. It is equal to the global variable sysLeap59 (see 10.3.8.12).

The value is selected as follows:

a) The value is obtained from a primary reference if known at the time of initialization, else
b) The value is set to FALSE.

14.2.11 timeTraceable

The value is set to TRUE if the timescale and the value of currentUtcOffset, relative to the ClockMaster entity of this time-aware system, are traceable to a primary reference standard; otherwise the value is set to FALSE. It is equal to the global variable sysTimeTraceable (see 10.3.8.16).

The value is selected as follows:

a) If the time and the value of currentUtcOffset are traceable to a primary reference standard at the time of initialization, the value is set to TRUE, else
b) The value is set to FALSE.

14.2.12 frequencyTraceable

The value is set to TRUE if the frequency determining the timescale of the ClockMaster Entity of this time-aware system is traceable to a primary standard; otherwise the value is set to FALSE. It is equal to the global variable sysFrequencyTraceable (see 10.3.8.17).

The value is selected as follows:

a) If the frequency is traceable to a primary reference standard at the time of initialization the value is set to TRUE, else
b) The value is set to FALSE.

14.2.13 ptpTimescale

The value set to TRUE if the clock timescale of the ClockMaster Entity of this time-aware system is PTP (see 8.2) and FALSE otherwise.

14.2.14 timeSource

The value is the source of time used by the grandmaster clock (see 8.6.2.7).

14.2.15 domainNumber

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

14.2.16 sdoId

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

NOTE—The attribute sdoId is specified as a 12-bit unsigned integer in 8.1. The data type for the managed object sdoId is UInteger16 in Table 14-1, for compatibility with IEEE Std 1588. The range of the managed object is limited to 12 bits; in addition, only the single value 0x100 is specified in this standard for the gPTP domain of a PTP Instance.

14.2.17 externalPortConfiguration

The value is the externalPortConfiguration attribute of the time-aware system (see 10.3.8.24).

14.2.18 instanceEnable

The value is the instanceEnable attribute of the time-aware system (see 10.2.3.24).

14.2.19 Default Parameter Data Set Table

There is one Default Parameter Table per~~-~~PTP Instance of a time-aware system, as detailed in Table 14-1.

Table 14-1—Default Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
clockIdentity	ClockIdentity	R	T	14.2.1
numberPorts	UInteger~~8~~16	R	T	14.2.2
clockQualtiy.clockClass	~~Enumeration8~~UInteger8	R	T	14.2.3.1 IEEE Std 1588-2008, 7.6.2.4
clockQuality.clockAc-curacy	Enumeration8	R	T	14.2.3.2 IEEE Std 1588-2008, 7.6.2.5
clockQuality.off-setScaledLogVariance	UInteger16	R	T	14.2.3.3
priority1	UInteger8	RW	T	14.2.4
priority2	UInteger8	RW	T	14.2.5
gmCapable	Boolean	R	T	14.2.6
currentUtcOffset	Integer16	RW	T	14.2.7
currentUtcOffsetValid	Boolean	RW	T	14.2.8

Table 14-1—Default Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
leap59	Boolean	RW	T	14.2.9
leap61	Boolean	RW	T	14.2.10
timeTraceable	Boolean	R	T	14.2.11
frequencyTraceable	Boolean	R	T	14.2.12
ptpTimescale	Boolean	R	T	14.2.13
timeSource	Enumeration8	R	T	14.2.14 and Table 8-4
domainNumber	UInteger8	RW	T	14.2.15
sdoId	UInteger16	R	T	14.2.16
externalPortConfiguration	Enumeration8	RW	O	14.2.17
instanceEnable	Boolean	RW	T	14.2.18

[a]R = Read only access; RW = Read/write access.
[b]T= Required for time-aware port;
O = Optional for time-aware port.

14.3 Current Parameter Data Set

The Current Parameter Data Set represents the position of a local system and other information, relative to the grandmaster.

<<Editor's note: A time-aware system is required to support domain 0, and may support a domain with domain number in the range 1 - 127. The domainNumber in Table 14-1 is indicated as supporting RW. However, if there is only one domain, the domain number must be 0; if there is more than one domain, one of the domains must have domain number of 0. Since the parameters and datasets in this clause are specified separately for each domain, it is not clear how to describe these constraints (i.e., RW is supported, but there are constraints on what can be written).>>

14.3.1 stepsRemoved

The value is the number of gPTP communication paths traversed between the local clock and the grandmaster clock, as specified in 10.3.3.

The default value is 0.

NOTE—For example, stepsRemoved for a slave clock on the same PTP communication path as the grandmaster clock will have a value of 1, indicating that a single path was traversed.

14.3.2 offsetFromMaster

The value is an implementation-specific representation of the current value of the time difference between a slave and the grandmaster, as computed by the slave, and as specified in 10.2.9. It is recommended that the data type be scaledNs. The default value is implementation specific.

14.3.3 lastGmPhaseChange

The value (see 10.2.3.16) is the phase change that occurred on the most recent change in either grandmaster or gmTimeBaseIndicator (see 9.2.2.3).

14.3.4 lastGmFreqChange

The value (see 10.2.3.17) is the frequency change that occurred on the most recent change in either grandmaster or gmTimeBaseIndicator (see 9.2.2.3).

14.3.5 gmTimebaseIndicator

The value is the value of timeBaseIndicator of the current grandmaster (see 9.2.2.3 and 9.6.2.3).

14.3.6 gmChangeCount

This statistics counter tracks the number of times the grandmaster has changed in a gPTP domain. This counter increments when the PortAnnounceInformation state machine enters the SUPERIOR_MASTER_PORT state or the INFERIOR_MASTER_OR_OTHER_PORT state (see 10.3.11 and Figure 10-13).

14.3.7 timeOfLastGmChangeEvent

This timestamp denotes the system time when the most recent grandmaster change occurred in a gPTP domain. This timestamp is updated when the PortAnnounceInformation state machine enters the SUPERIOR_MASTER_PORT state or the INFERIOR_MASTER_OR_OTHER_PORT state (see 10.3.11 and Figure 10-13).

<<Editor’s note: This subclause, and also 14.3.8 and 14.3.9, use the term “system time.” These (and corresponding places in the MIB) are the only places where “system time” is used. It appears that “system time” simply means “grandmaster time,” or “synchronized time,” or gPTP time. It is suggested that this be clarified, as the editor received a question on this. The editor suggests “synchronized time,” as this term is defined in 3.21 and is used throughout the document.

It is also possible to use the term “Local PTP Clock” to refer to the local estimate of the grandmaster time. This term is used in P1588-Rev/D1.1 and preliminary versions of D1.2.>>

14.3.8 timeOfLastGmPhaseChangeEvent

This timestamp denotes the system time when the most recent change in grandmaster phase occurred, due to a change of either the grandmaster or the grandmaster time base. This timestamp is updated when one of the following occurs:

a) The PortAnnounceInformation state machine enters the SUPERIOR_MASTER_PORT state or the INFERIOR_MASTER_OR_OTHER_PORT state (see 10.3.11 and Figure 10-13), or
b) The gmTimebaseIndicator managed object (see 14.3.5) changes and the lastGmPhaseChange field of the most recently received Follow_Up information TLV is nonzero.

14.3.9 timeOfLastGmFreqChangeEvent

This timestamp denotes the system time when the most recent change in grandmaster frequency occurred, due to a change of either the grandmaster or the grandmaster time base. This timestamp is updated when one of the following occurs:

a) The PortAnnounceInformation state machine enters the SUPERIOR_MASTER_PORT state or the INFERIOR_MASTER_OR_OTHER_PORT state (see 10.3.11 and Figure 10-13), or
b) The gmTimebaseIndicator managed object (see 14.3.5) changes and the lastGmFreqChange field of the most recently received Follow_Up information TLV is nonzero.

14.3.10 Current Parameter Data Set Table

There is one Current Parameter Data Set Table per PTP Instance of a time-aware system, as detailed in Table 14-2.

Table 14-2—Current Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
stepsRemoved	UInteger16	R	T	14.3.1
offsetFromMaster	~~ScaledNs~~TimeInterval ~~(recommended)~~	R	T	14.3.2
lastGmPhaseChange	ScaledNs	R	T	14.3.3
lastGmFreqChange	Double	R	T	14.3.4
gmTimebaseIndicator	UInteger16	R	T	14.3.5
gmChangeCount	UInteger32	R	T	14.3.6
timeOfLastGmChangeEvent	UInteger32 (0.01 s 32-bit system time)	R	T	14.3.7
timeOfLastGmPhaseChangeEvent	UInteger32 (0.01 s 32-bit system time)	R	T	14.3.8
timeOfLastGmFreqChangeEvent	UInteger32 (0.01 s 32-bit system time)	R	T	14.3.9

[a]R = Read only access; RW = Read/write access.
[b]T= Required for time-aware port.

14.4 Parent Parameter Data Set

The Parent Parameter Data Set represents capabilities of the upstream system, toward the grandmaster, as measured at a local system.

14.4.1 parentPortIdentity

If this time-aware system is the grandmaster, the value is a portIdentity whose clockIdentity is the clockIdentity of this time-aware system, and whose portNumber is 0.

If this time-aware system is not the grandmaster, the value is the portIdentity of the MasterPort (see Table 10-4) of the gPTP communication path attached to the single slave port of this time-aware system.

The default value is a portIdentity for which:

a) The clockIdentity member is the value of the clockIdentity member of the default data set, and
b) The portNumber member is 0.

14.4.2 cumulativeRateRatio

The value is an estimate of the ratio of the frequency of the grandmaster to the frequency of the LocalClock entity of this time-aware system. cumulativeRateRatio is expressed as the fractional frequency offset multiplied by 2^{41}, i.e., the quantity (rateRatio – 1.0)(2^{41}), where rateRatio is computed by the PortSyncSyncReceive state machine (see 10.2.7.1.4).

14.4.3 grandmasterIdentity

The value is the clockIdentity attribute, see 8.5.2.2, of the grandmaster clock.

The default value is the clockIdentity member of the Default Parameter Data Set (14.2.1).

14.4.4 grandmasterClockQuality

This managed object is a structure whos data type is ClockQuality (see 6.4.3.8).

14.4.5 grandmasterClockQuality.clockClass

The value is the clockClass, see 8.6.2.2, of the grandmaster clock.

The default value is the clockClass member of the default data set.

14.4.6 grandmasterClockQuality.clockAccuracy

The value is the clockAccuracy, see 8.6.2.3, of the grandmaster clock.

The default value is the clock accuracy member of the default data set.

14.4.7 grandmasterClockQuality.~~O~~offsetScaledLogVariance

The value is the offsetScaledLogVariance, see 8.6.2.4, of the grandmaster clock.

The default value is the offsetScaledLogVariance member of the default data set.

14.4.8 grandmasterPriority1

The value is the priority1 attribute, see 8.6.2.1, of the grandmaster clock.

The default value is the priority1 value of the default data set.

14.4.9 grandmasterPriority2

The value is the priority2 attribute, see 8.6.2.5, of the grandmaster clock.

The default value is the priority2 value of the default data set.

14.4.10 ParentParameter Data Set Table

There is one Parent Parameter Data Set Table per PTP Instance of a time-aware system, as detailed in Table 14-3.

14.5 Time Properties Parameter Data Set

The Time Properties Parameter Data Set represents capabilities of the grandmaster, as measured at a local system.

Table 14-3—Parent Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
parentPortIdentity	PortIdentity (see 6.4.3.7)	R	T	14.4.1
cumulativeRateRatio	Integer32	R	T	14.4.2
grandMasterIdentity	ClockIdentity	R	T	14.4.3
grandmasterClockQuality.clockClass	UInteger8	R	T	14.4.5 IEEE Std 1588-2008, 7.6.2.4
grandmasterClockQuality.clockAccuracy	Enumeration8	R	T	14.4.6 IEEE Std 1588-2008, 7.6.2.5
grandmasterClockQuality.o~~O~~ffsetScaled-LogVariance	UInteger16	R	T	14.4.7
grandmasterPriority1	UInteger8	R	T	14.4.8
grandmasterPriority2	UInteger8	R	T	14.4.9

[a]R = Read only access; RW = Read/write access.
[b]T= Required for time-aware port.

14.5.1 currentUtcOffset

The value is currentUtcOffset for the current grandmaster (see 14.2.7). It is equal to the value of the global variable currentUtcOffset (see 10.3.8.10). The value is in units of seconds.

14.5.2 currentUtcOffsetValid

The value is currentUtcOffsetValid for the current grandmaster (see 14.2.8). It is equal to the global variable currentUtcOffsetValid (see 10.3.8.6).

14.5.3 leap59

The value is leap59 for the current grandmaster (see 14.2.9). It is equal to the global variable leap59 (see 10.3.8.5).

14.5.4 leap61

The value is leap59 for the current grandmaster (see 14.2.10). It is equal to the global variable leap61 (see 10.3.8.4).

14.5.5 timeTraceable

The value is timeTraceable for the current grandmaster (see 14.2.11). It is equal to the global variable timeTraceable (see 10.3.8.8).

14.5.6 frequencyTraceable

The value is frequencyTraceable for the current grandmaster (see 14.2.12). It is equal to the global variable frequencyTraceable (see 10.3.8.9).

14.5.7 ptpTimescale

The value is ptpTimescale for the current grandmaster (see 14.2.13).

14.5.8 timeSource

The value is timeSource for the current grandmaster (see 14.2.14). It is equal to the global variable timeSource (see 10.3.8.11).

14.5.9 Time Properties Parameter Data Set Table

There is one Time Properties Parameter Data Set Table per PTP Instance of a time-aware system, as detailed in Table 14-4.

Table 14-4—Time Properties Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
currentUtcOffset	Integer16	R	T	14.5.1
currentUtcOffsetValid	Boolean	R	T	14.5.2
leap59	Boolean	R	T	14.5.3
leap61	Boolean	R	T	14.5.4
timeTraceable	Boolean	R	T	14.5.5
frequencyTraceable	Boolean	R	T	14.5.6
ptpTimescale	Boolean	R	T	14.5.7
timeSource	Enumeration8	R	T	14.5.8 and Table 8-4

[a]R = Read only access; RW = Read/write access.
[b]T= Required for time-aware port.

14.6 Path Trace Parameter Data Set

The Path Trace Parameter Data set represents the current path trace information available at the time-aware system.

14.6.1 list

The value is the array of ClockIdentity values contained in the pathTrace array (see 10.3.8.23), which represents the current path trace information, and which is carried in the path trace TLV (see 10.6.3.3).

The initialization value shall be the empty list (i.e., an array of length 0).

The pathTraceDS.list shall be initialized to the empty list whenever the PTP Instance updates data sets based on decision code M1 or M2; see 9.3.5.

14.6.2 enable

The value is TRUE.

NOTE—This member is included for compatibility with IEEE Std 1588. In IEEE Std 1588, the path trace mechanism is optional, and the pathTraceDS.enable member is configurable (its value in IEEE Std 1588 is TRUE or FALSE, depending on whether the path trace mechanism is operational or not operational, respectively.. However, the pathTrace mechanism is manditory in this standard, and the value of enable is always TRUE.

14.6.3 Path Trace Parameter Data Set Table

There is one Path Trace Parameter Table per time-aware system, as detailed in Table 14-5.

Table 14-5—Path Trace Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
list	ClockIdentity[N], where N is defined in 10.3.8.23	R	T	14.6.1
enable	Boolean	R	T	14.6.2

[a]R = Read only access; RW = Read/write access.
[b]T= Required for time-aware port.

14.7 Acceptable Master Table Parameter Data Set

<<Editor's note: This subclause has been moved, so that the order of subclauses agrees with the order in the data set hierarchy at the beginning of Clause 14. However, in moving this material, none of the material was changed. Therefore, track changes was disabled for this operation (so that there would not be an indication of this material being deleted). Change bars were not turned off, and are visible in the left margin.>>

The Acceptable Master Table Parameter Data Set represents the acceptable master table used when an EPON port is present in a time-aware system.

14.7.1 maxTableSize

The value is the maximum size of the AcceptableMasterTable. It is equal to the maxTableSize member of the AcceptableMasterTable structure (see 13.1.3.2).

14.7.2 actualTableSize

The value is the actual size of the AcceptableMasterTable. It is equal to the actualTableSize member of the AcceptableMasterTable structure (see 13.1.3.2 and 13.1.3.5), i.e., the current number of elements in the acceptable master array. The actual table size is less than or equal to the max table size.

14.7.3 acceptableMasterArray

Each element of this array is an AcceptableMaster structure, see 13.1.3.3 and 13.1.3.5.

14.7.4 Acceptable Master Table Parameter Data Set Table

There is one Acceptable Master Table Parameter Data Set Table per ~~gPTP instance~~PTP Instance of a time-aware system, as detailed in Table 14-6.

Table 14-6—Acceptable Master Table Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
maxTableSize	UInteger16	R	Tdot3ONU	14.7.1
actualTableSize	UInteger16	RW	Tdot3ONU	14.7.2
acceptableMasterArray	AcceptableMaster[actualTableSize] (see 13.1.3.3)	RW	Tdot3ONU	14.7.3

[a]R = Read only access; RW = Read/write access.
[b]Tdot3ONU = Required for time-aware IEEE 802.3 EPON ONU ports.

14.8 Port Parameter Data Set

The Port Parameter Data Set represents time-aware port capabilities ~~at a given Bridge or end station~~for a PTP Instance of a time-aware system.

14.8.1 General

For the single port of a PTP Instance of a time-aware end station and for each port of a PTP Instance of a time-aware ~~Bridge~~relay, the following Port Parameter Data Set is maintained as the basis for protocol decisions and providing values for message fields. The number of such data sets is the same as the numberPorts value of the Default Parameter Data Set.

14.8.2 portIdentity

The value is the portIdentity attribute of the local port, see 8.5.2.

14.8.3 port~~Role~~State

The value is the value of the port ~~role~~state of this port (see Table 10-1) and is taken from the enumeration in Table 14-7.

Table 14-7—port~~Role~~State enumeration

State	Value
DisabledPort	3
MasterPort	6
PassivePort	7
SlavePort	9
	All other values reserved

NOTE—The enumeration values are consistent with IEEE Std 1588-2008, Table 8.

[Editor’s note: The above note is a table note, and has been moved to the bottom of the table in accordance with the IEEE Style guide. The correct reference for IEEE 1588-Rev will be filled in prior to sponsor ballot.]

~~NOTE—The enumeration values are consistent with IEEE Std 1588-2008, Table 8.~~

The default value is 3 (DisabledPort).

14.8.4 ~~pttPortEnabled~~ptpPortEnabled

The value is equal to the value of the Boolean ~~pttPortEnabled~~ptpPortEnabled (see 10.2.4.13).

14.8.5 delayMechanism

The value indicates the mechanism for measuring mean propagation delay and neighbor rate ratio on the link attached to this port, and is taken from the enumeration in Table 14-8. If the domain number is not 0,

Table 14-8—delayMechanism enumeration

Delay Mechanism	Value	Specification
P2P	02	The gPTP port uses the peer delay mechanism
COMMON_P2P	03	The gPTP port uses the CMLDS
SPECIAL	04	The gPTP port uses a transport that has a native time transfer mechanism and, therefore, does not use the peer delay mechanism (e.g., IEEE Std 802.11, IEEE Std 802.3 EPON)
	All other values reserved	

NOTE—The enumeration values are consistent with P1588/D1.2v14, Table 18.

[Editor's note: the correct reference for IEEE 1588-Rev will be filled in prior to sponsor ballot.]

portDS.delay mechanism must not be P2P (see 11.2.15).

14.8.6 isMeasuringDelay

The value is equal to the value of the Boolean isMeasuringDelay (see 11.2.12.6 and 16.4.3.2).

14.8.7 asCapable

The value is equal to the value of the Boolean asCapable (see 10.2.4.1).

14.8.8 neighborPropDelay

The value is equal to the value of the per-port global variable neighborPropDelay (see 10.2.4.8). It is an estimate of the current one-way propagation time on the link attached to this port, measured as specified for the respective medium (see 11.2.15, 12.5.2, and 16.4). The value is zero for ports attached to IEEE 802.3 EPON links and for the master port of an IEEE 802.11 link, because one-way propagation delay is not measured on the latter and not directly measured on the former. It is recommended that the data type be scaledNs. The default value is zero.

14.8.9 neighborPropDelayThresh

The value is equal to the value of the per-port global variable neighborPropDelayThresh (see 11.2.12.7). It is the propagation time threshold, above which a port is not considered capable of participating in the IEEE 802.1AS protocol.

14.8.10 delayAsymmetry

The value is the asymmetry in the propagation delay on the link attached to this port relative to the grandmaster time base, as defined in 8.3. If propagation delay asymmetry is not modeled, then delayAsymmetry is 0.

14.8.11 neighborRateRatio

The value is an estimate of the ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this port, to the frequency of the LocalClock entity of this time-aware system (see 10.2.4.7). neighborRateRatio is expressed as the fractional frequency offset multiplied by 2^{41}, i.e., the quantity (neighborRateRatio – 1.0)(2^{41}).

14.8.12 initialLogAnnounceInterval

If useMgtSettableLogAnnounceInterval is FALSE, t~~T~~he value is the logarithm to base 2 of the announce interval used when (a) the port is initialized, or (b) a message interval request TLV is received with the announceInterval field set to 126 (see 10.7.2.2 and the AnnounceIntervalSetting state machine, 10.3.16).

14.8.13 currentLogAnnounceInterval

The value is the logarithm to the base 2 of the of the current announce interval, see 10.7.2.2.

14.8.14 useMgtSettableLogAnnounceInterval

The managed object is a boolean that determines the source of the announce interval. If the value is TRUE, the value of currentLogAnnounceInterval is set equal to the value of mgtSettableLogAnnounceInterval (see 14.8.15). If the value is FALSE, the value of currentLogAnnounceInterval is determined by the AnnounceIntervalSetting state machine (see 10.3.16). The default value of useMgtSettableLogAnnounceInterval is FALSE for domain 0 and TRUE for domains other than domain 0.

14.8.15 mgtSettableLogAnnounceInterval

The value is the logarithm to base 2 of the announce interval used if useMgtSettableLogAnnounceInterval is TRUE. The value is not used if useMgtSettableLogAnnounceInterval is FALSE.

14.8.16 announceReceiptTimeout

The value is the number of Announce message transmission intervals that a slave port waits without receiving an Announce message, before assuming that the master is no longer transmitting Announce messages and the BMCA needs to be run, if appropriate (see 10.7.3.2).

14.8.17 initialLogSyncInterval

If useMgtSettableLogSyncInterval is TRUE, t~~T~~he value is the logarithm to base 2 of the sync interval used when (a) the port is initialized, or (b) a message interval request TLV is received with the timeSyncInterval field set to 126 (see 10.7.2.3, 11.5.2.3, 12.7.2, 13.9.2, and the ~~LinkDelay~~SyncIntervalSetting state machine, 11.2.19).

14.8.18 currentLogSyncInterval

The value is the logarithm to the base 2 of the current time-synchronization transmission interval, see 10.7.2.3.

14.8.19 useMgtSettableLogSyncInterval

The managed object is a boolean that determines the source of the sync interval. If the value is TRUE, the value of currentLogSyncInterval is set equal to the value of mgtSettableLogSyncInterval (see 14.8.20). If the value of the managed object is FALSE, the value of currentLogSyncInterval is determined by the SyncIntervalSetting state machine (see 11.2.19). The default value of useMgtSettableLogSyncInterval is FALSE for domain 0 and TRUE for domains other than domain 0.

14.8.20 mgtSettableLogSyncInterval

The value is the logarithm to base 2 of the sync interval if useMgtSettableLogSyncInterval is TRUE. The value is not used if useMgtSettableLogSyncInterval is FALSE.

14.8.21 syncReceiptTimeout

The value is the number of time-synchronization transmission intervals that a slave port waits 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 (see 10.7.3.1).

14.8.22 syncReceiptTimeoutTimeInterval

The value is equal to the value of the per-port global variable syncReceiptTimeoutTimeInterval (see 10.2.4.3). It is the time interval after which sync receipt timeout occurs if time-synchronization information has not been received during the interval.

14.8.23 initialLogPdelayReqInterval

For full-duplex, IEEE 802.3 media and CSN media that use the peer delay mechanism to measure path delay (see 16.4.3.1), the value is the logarithm to base 2 of the Pdelay_Req message transmission interval used when (a) the port is initialized, or (b) a message interval request TLV is received with the linkDelayInterval field set to 126 (see 11.5.2.2 and the LinkDelaySyncIntervalSetting state machine,).

For all other media, the value is 127.

14.8.24 currentLogPdelayReqInterval

For full-duplex, IEEE 802.3 media and CSN media that use the peer delay mechanism to measure path delay (see E.4.3.1), the value is the logarithm to the base 2 of the current Pdelay_Req message transmission interval, see 11.5.2.2.

For all other media, the value is 127.

14.8.25 useMgtSettableLogPdelayReqInterval

The managed object is a boolean that determines the source of the sync interval and mean time interval between successive Pdelay_Req messages. If the value is TRUE, the value of currentLogPdelayReqInterval is set equal to the value of mgtSettableLogPdelayReqInterval (see 14.13.13). If the value of the managed object is FALSE, the value of currentLogPdelayReqInterval is determined by the LinkDelayIntervalSetting state machine (see 11.2.20). The default value of useMgtSettableLogPdelayReqInterval is FALSE.

<<Editor's note: The resolution of comment #17 against D4.3 indicated that the default is to use the management-settable variables for domains other than 0 and to use Signaling messages with management for initialization for domain 0. For the peer delay process, we use the common mean link delay service, and there are no domains. However, in the case that only one domain is present, that domain is domain 0, and the situation is as if the pdelay messages are sent on domain 0. Therefore, this is initialized to FALSE. This also preserves backward compatibility.

Also, this managed object, along with others pertaining to pdelay, may be moved to a common mean link delay service data set when data sets for this service are defined.>>

14.8.26 mgtSettableLogPdelayReqInterval

The value is the logarithm to base 2 of the mean time interval between successive Pdelay_Req messages if useMgtSettableLogPdelayReqInterval is TRUE. The value is not used if useMgtSettableLogPdelayReqInterval is FALSE.

14.8.27 allowedLostResponses

The value is equal to the value of the per-port global variable allowedLostResponses (see 11.5.3 and 11.2.12.4). It is the number of Pdelay_Req messages for which a valid response is not received, above which a port is considered to not be exchanging peer delay messages with its neighbor.

14.8.28 allowedFaults

The value is equal to the value of the per-Link-Port global variable allowedFaults (see 11.5.4 and 11.2.12.5). It is the number of faults (see 11.5.4), above which asCapableAcrossDomains is set to FALSE, i.e., a Link Port is considered to not be be capable of interoperating with its neighbor via the 802.1AS protocol (see 10.2.4.1).

14.8.29 logGptpCapableMessageInterval

The value is the logarithm to the base 2 of the transmission interval between successive Signaling messages that contain the gPTP capable TLV (see 10.7.2.1 and 10.7.2.5).

14.8.30 gPtpCapableReceiptTimeout

The value is the number of transmission intervals that a port waits without receiving the gPTP capable TLV, before assuming that the neighbor port is no longer invoking the gPTP protocol (see 10.7.3.3).

14.8.31 versionNumber

This value is set to versionPTP as specified in 10.6.2.2.4.

14.8.32 nup

For an OLT port of an IEEE 802.3 EPON link, the value is the effective index of refraction for the EPON upstream wavelength light of the optical path (see 13.1.4 and 13.8.1.2.2). The default value is 1.46770 for 1 Gb/s upstream links, and 1.46773 for 10 Gb/s upstream links.

For all other ports, the value is 0.

14.8.33 ndown

For an OLT port of an IEEE 802.3 EPON link, the value is the effective index of refraction for the EPON downstream wavelength light of the optical path (see 13.1.4 and 13.8.1.2.1). The default value is 1.46805 for 1 Gb/s downstream links, and 1.46851 for 10 Gb/s downstream links.

For all other ports, the value is 0.

14.8.34 ~~acceptableMasterTableEnabled~~

~~The value is equal to the value of the Boolean acceptableMasterTableEnabled (see 13.1.3.1 and 13.1.3.5).~~

14.8.35 oneStepTxOper

The value is equal to the value of the per-port global variable oneStepTxOper (see 11.2.12.11). Its value is TRUE if the port is sending one-step Sync messages, and FALSE if the port is sending two-step Sync and Follow-Up messages.

<<Editor's note: It was noticed that in some places this was written oneStepTXOper, i.e., with X upper case. This has been fixed; it is now oneStepTxOper uniformly.>>

14.8.36 oneStepReceive

The value is equal to the value of the per-port global variable oneStepReceive (see 11.2.12.9). Its value is TRUE if the port is capable of receiving and processing one-step Sync messages.

14.8.37 OneStepTransmit

The value is equal to the value of the per-port global variable oneStepTransmit (see 11.2.12.10). Its value is TRUE if the port is capable of transmitting one-step Sync messages.

14.8.38 initialOneStepTxOper

If useMgtSettableOneStepTxOper is FALSE, the value is used to initialize currentOneStepTxOper when the port is initialized. If useMgtSettableOneStepTxOper is TRUE, the value of initialOneStepTxOper is not used.

14.8.39 currentOneStepTxOper

The value is TRUE if it is desired, either via management or via a received Signaling message, that the port transmit one-step Sync messages. The value is FALSE if it is not desired, either via management or via a received Signaling message, that the port transmit one-step Sync messages.

NOTE—The port will send one-step Sync messages only if currentOneStepTxOper and oneStepTransmit (see 14.8.37) are both TRUE (see 11.2.21 and Figure 11-12).

14.8.40 useMgtSettableOneStepTxOper

The managed object is a boolean that determines the source of currentOneStepTxOper. If the value is TRUE, the value of currentOneStepTxOper is set equal to the value of mgtSettableOneStepTxOper (see 14.8.41). If the value is FALSE, the value of currentOneStepTxOper is determined by the OneStepTxOperSetting state machine (see 11.2.21). The default value of useMgtSettableOneStepTxOper is TRUE.

14.8.41 mgtSettableOneStepTxOper

If useMgtSettableOneStepTxOper is TRUE, currentOneStepTxOper is set equal to the value of mgtSettableOneStepTxOper. The value of mgtSettableOneStepTxOper is not used if useMgtSettableLogAnnounceInterval is FALSE. The default value of mgtSettableOneStepTxOper is FALSE for domains other than domain 0.

<<Editor's note: The resolution of comment #17 of D4.2 indicates that for domain 0, the default is to look only at management for one-step/two-step operation, and for domains other than domain 0, the default is two-step. This is achieved by defaulting useMgtSettableOneStepTxOper to TRUE in all cases, and defaulting mgtSettableOneStepTxOper to TRUE.>>

14.8.42 syncLocked

The value is equal to the value of the per-port global variable syncLocked (see 10.2.4.15). Its value is TRUE if the port will transmit a Sync as soon as possible after the slave port receives a Sync.

14.8.43 pdelayTruncatedTimestampsArray

For full-duplex IEEE 802.3 media, and CSN media that use the peer delay mechanism to measure path delay (see 16.4.3.1), the values of the four elements of this array are as described in Table 14-15. For all other media, the values are zero. Array elements 0, 1, 2, and 3 correspond to the timestamps t1, t2, t3, and t4, modulo 2^{32}, respectively, in Figure 7-7, and expressed in units of 2^{-16} ns (i.e., the value of each array element is equal to the remainder obtained upon dividing the respective timestamp, expressed in units of 2^{-16} ns, by 2^{48}). At any given time, the timestamp values stored in the array are for the same, and most recently completed, peer delay message exchange.

<<Editor's note: The above managed objects contain values associated with the timestamps t1, t2, t3, and t4 (in array elements 0, 1, 2, and 3, respectively) of the peer delay exchange. We use timestamps associated with the peer delay exchange rather than with Sync and Follow_Up, because the peer delay mechanism is used here. The timestamps associated with the Sync message, i.e., the syncEventEgressTimestamp and syncEventIngressTimestamp cannot be used because they are not both available at the same Link Port, and because they are associated with a message sent only in one direction.>>

14.8.44 asymmetryMeasurementMode

The value is equal to the value of the Boolean asymmetryMeasurementMode(see G.3). For full-duplex IEEE 802.3 media, the value is TRUE if an asymmetry measurement is being performed for the link attached to this Link Port, and FALSE otherwise. For all other media, the value shall be FALSE (see 10.2.4.2).

NOTE—If an asymmetry measurement is being performed for a link, asymmetryMeasurementMode must be TRUE for the Link Ports at each end of the link.

14.8.45 minorVersionNumber

This value is set to minorVersionPTP as specified in 10.6.2.2.3.

14.8.46 Port Parameter Data Set Table

There is one Port Parameter Data Set Table per port, per PTP Instance of a time-aware system. Each Port Parameter Data Set Table contains a set of parameters for each port that supports the time-synchronization capability, as detailed in Table 14-10. Each table can be created or removed dynamically in implementations that support dynamic configuration of ports and components.

Table 14-9—Description of pdelayTruncatedTimestampsArray

Array element	Timestamp description	Corresponding timestamp of Figure 7-7	Units
0	pdelayReqEventEgressTimestamp for Pdelay_Req message, of most recently-completed peer delay message exchange, transmitted by this time-aware system (NOTE 1)	t1	2^{-16} ns
1	pdelayReqEventIngressTimestamp for Pdelay_Req message received at peer delay responder that this Link Port sends Pdelay_Req to, of most recently-completed peer delay message exchange; it is equal to the sum of the following: a) the ns field of the requestReceiptTimestamp (see Table 11-14), multiplied by 2^{16}, and b) the correctionField (see Table 11-6) (NOTE 2)	t2	2^{-16} ns
2	pdelayRespEventEgressTimestamp for Pdelay_Resp message, of most recently-completed peer delay message exchange, transmitted by peer delay responder that this Link Port sends Pdelay_Req to; it is equal to the sum of the following: a) the ns field of the responseOriginTimestamp (see Table 11-15), multiplied by 2^{16}, b) the correctionField (see Table 11-6). (NOTE 3)	t3	2^{-16} ns
3	pdelayRespEventIngressTimestamp for Pdelay_Resp message, of most recently-completed peer delay message exchange, received by this time-aware system (NOTE 4)	t4	2^{-16} ns

Table 14-9—Description of pdelayTruncatedTimestampsArray

Array element	Timestamp description	Corresponding timestamp of Figure 7-7	Units
NOTE 1—This quantity is not simply the nanoseconds plus fractional nanoseconds portion of the pdelayReqEventEgressTimestamp. Rather, it is equal to [pdelayReqEventEgressTimestamp.seconds × $(10^9)(2^{16})$ + pdelayReqEventEgressTimestamp.fractionalNanoseconds] mod 2^{48}, where pdelayReqEventEgressTimestamp is expressed as an ExtendedTimestamp (see 6.4.3.5). NOTE 2—This quantity is not simply the nanoseconds plus fractional nanoseconds portion of the pdelayReqEventIngressTimestamp. Rather, it is equal to [pdelayReqEventIngressTimestamp>.seconds × $(10^9)(2^{16})$ + pdelayReqEventIngressTimestamp.fractionalNanoseconds] mod 2^{48}, where pdelayReqEventIngressTimestamp is expressed as an ExtendedTimestamp (see 6.4.3.5). NOTE 3—This quantity is not simply the nanoseconds plus fractional nanoseconds portion of the pdelayRespEventIngressTimestamp. Rather, it is equal to [pdelayRespEventIngressTimestamp.seconds × $(10^9)(2^{16})$ + pdelayRespEventIngressTimestamp.fractionalNanoseconds] mod 2^{48}, where pdelayRespEventIngressTimestamp is expressed as an ExtendedTimestamp (see 6.4.3.5). NOTE 4—This quantity is not simply the nanoseconds plus fractional nanoseconds portion of the pdelayRespEventIngressTimestamp. Rather, it is equal to [pdelayRespEventIngressTimestamp.seconds × $(10^9)(2^{16})$ + pdelayRespEventIngressTimestamp.fractionalNanoseconds] mod 2^{48}, where pdelayRespEventIngressTimestamp is expressed as an ExtendedTimestamp (see 6.4.3.5). Its units are 2^{-16} ns.			

Table 14-10—Port Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
portIdentity	PortIdentity (see 6.4.3.7)	R	T	14.8.2
port~~Role~~State	Enumeration8	RW	T	14.8.3, Table 14-7
~~pttPortEnabled~~ptpPortEnabled	Boolean	RW	T	14.8.4
delayMechanism	Enumeration8	RW	T	14.8.5
isMeasuringDelay	Boolean	R	T	14.8.6
asCapable	Boolean	R	T	14.8.7
neighborPropDelay	UScaledNs (recommended)	R	T	14.8.8
neighborPropDelayThresh	UScaledNs (recommended)	RW	T	14.8.9
delayAsymmetry	UScaledNs (recommended)	RW	Tdot3FD	14.8.10
neighborRateRatio	Integer32	R	T	14.8.11
initialLogAnnounceInterval	Integer8	RW	T	14.8.12
currentLogAnnounceInterval	Integer8	R	T	14.8.13
useMgtSettableLogAnnounceInterval	Boolean	RW	T	14.8.14
mgtSettableLogAnnouncerInterval	Integer8	RW	T	14.8.15
announceReceiptTimeout	UInteger8	RW	T	14.8.16
initialLogSyncInterval	Integer8	RW	T	14.8.17

Table 14-10—Port Parameter Data Set Table *(continued)*

Name	Data type	Operations supported[a]	Conformance[b]	References
currentLogSyncInterval	Integer8	R	T	14.8.18
useMgtSettableLogSyncInterval	Boolean	RW	T	14.8.19
mgtSettableLogSyncInterval	Integer8	RW	T	14.8.20
syncReceiptTimeout	UInteger8	RW	T	14.8.21
syncReceiptTimeoutTimeInterval	UScaledNs	R	T	14.8.22
initialLogPdelayReqInterval	Integer8	RW	T	14.8.23
currentLogPdelayReqInterval	Integer8	R	T	14.8.24
useMgtSettableLogPdelayReqInterval	Boolean	RW	T	14.8.25
MgtSettableLogPdelayReqInterval	Integer8	RW	T	14.8.26
allowedLostResponses	UInteger16	RW	T	14.8.27
allowedFaults	UInteger16	RW	T	14.8.28
logGptpCapableMessageInterval	Integer8	RW	T	14.8.29
gPtpCapableReceiptTimeout	UInteger8	RW	T	14.8.30
versionNumber	UInteger4	R	T	14.8.31
nup	Double	RW	Tdot3OLT	14.8.32
ndown	Double	RW	Tdot3OLT	14.8.33
oneStepTxOper	Boolean	R	O	14.8.35
oneStepReceive	Boolean	R	O	14.8.36
oneStepTransmit	Boolean	R	O	14.8.37
initialOneStepTxOper	Boolean	RW	O	14.8.38
currentOneStepTxOper	Boolean	R	O	14.8.39
useMgtSettableOneStepTxOper	Boolean	RW	O	14.8.40
mgtSettableOneStepTxOper	Boolean	RW	O	14.8.41
syncLocked	Boolean	R	O	14.8.42
pdelayTruncatedTimestampsArray	UInteger48[4]	R	O	14.8.43
asymmetryMeasurementMode		RW	O	14.8.44
minorVersionNumber	UInteger4	R	T	14.8.45

[a]R = Read only access; RW = Read/write access.
[b]T= Required for time-aware port;
O = Optional for time-aware port;
Tdot3FD = Required for time-aware IEEE 802.3 full-duplex port;
Tdot3OLT = Required for time-aware IEEE 802.3 EPON OLT ports;
Tdot3ONU = Required for time-aware IEEE 802.3 EPON ONU ports.

14.9 Description Port Parameter Data Set

The Description Port Parameter Data Set contains the profileIdentifier for this PTP profile, as specified in F.1.

<<Editor's note: In accordance with comment 33 against D4.2, only the profileIdentifier member is included, and not the protocolAddress (i.e., MAC address in this case). Given that protocolAddress is

not included, this data set technically does not need to be per port; it is assumed it is per port for consistencey with 1588-Rev. Should protocolAddress, i.e., MAC address, be included?>>

14.9.1 profileIdentifier

The value is the profileIdentifier for this PTP profile (see Annex F, F.1).

14.9.2 Description Port Parameter Data Set Table

There is one Description Port Parameter Data Set Table per port of a time-aware system, as detailed in . Table 14-11.

Table 14-11—Description Port Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
profileIdentifier	Octet6	R	O	14.10.2

[a]R= Read only access.
[b]O= Optional.

<<Editor's note: Comment 33 of D4.2 did not indicate whether this data set, and/or the profileIdentifier member, should be required or optional. For now, the member is listed as optional, as this data set and member were not included in 802.1AS-2011. Comments are requested on this.>>

14.10 Port Parameter Statistics Data Set

The ~~Port Parameter Statistics~~Port Parameter Statistics Data Set provides counters associated with port capabilities at a given ~~Bridge or end station~~time-aware system.

14.10.1 General

For the single port of a PTP Instance of a time-aware end station and for each port of a PTP Instance of a time-aware ~~Bridge~~relay, the following ~~Port Parameter Statistics~~Port Parameter Statistics Data Set provides counters. The number of such statistics sets is the same as the numberPorts value of the Default Parameter Data Set.

14.10.2 rxSyncCount

A counter that increments every time synchronization~~-~~information is received, denoted by a transition to TRUE from FALSE of the rcvdSync variable of the MDSyncReceiveSM state machine (see 11.2.13.1.2 and Figure 11-6), when in the DISCARD, ~~or~~ WAITING_FOR_SYNC, or WAITING_FOR_FOLLOW_UP states; or rcvdIndication transitions to TRUE (see Figure 12-6).

14.10.3 rxOneStepSyncCount

A counter that increments every time a one-step Sync message is received, denoted by a transition to TRUE from FALSE of the rcvdSync variable of the MDSyncReceiveSM state machine (see 11.2.13.1.2 and Figure 11-6) with the variable rcvdSyncPtr->twoStepFlag FALSE, when in the DISCARD, WAITING_FOR_SYNC, or WAITING_FOR_FOLLOW_UP states.

<<Editor's note: time-synchronization event messages received on 802.11 links are considered to be two-step. time-synchronization event messages received on EPON links, or on CSN links for the case

where a CSN network reference clock is present, are considered to be internal to the timing mechanism of the medium and are not counted.>>

14.10.4 rxFollowUpCount

A counter that increments every time a Follow_Up message is received, denoted by a transition to TRUE from FALSE of the rcvdFollowUp variable of the MDSyncReceiveSM state machine (see 11.2.13.1.3 and Figure 11-6) when in the WAITING_FOR_FOLLOW_UP state.

14.10.5 rxPdelayRequestCount

A counter that increments every time a Pdelay_Req message is received, denoted by a transition to TRUE from FALSE of the rcvdPdelayReq variable of the MDPdelayResp state machine (see 11.2.18.2.1 and Figure 11-9) when in the WAITING_FOR_PDELAY_REQ or INITIAL_WAITING_FOR_PDELAY_REQ states.

14.10.6 rxPdelayResponseCount

A counter that increments every time a Pdelay_Resp message is received, denoted by a transition to TRUE from FALSE of the rcvdPdelayResp variable of the MDPdelayReq state machine (see 11.2.17.2.2 and Figure 11-8) when in the WAITING_FOR_PDELAY_RESP state.

14.10.7 rxPdelayResponseFollowUpCount

A counter that increments every time a Pdelay_Resp_Follow_Up message is received, denoted by a transition to TRUE from FALSE of the rcvdPdelayRespFollowUp variable of the MDPdelayReq state machine (see 11.2.17.2.4 and Figure 11-8) when in the WAITING_FOR_PDELAY_RESP_FOLLOW_UP state.

14.10.8 rxAnnounceCount

A counter that increments every time an Announce message is received, denoted by a transition to TRUE from FALSE of the rcvdAnnounce variable of the PortAnnounceReceive state machine (see 10.3.10 and Figure 10-12) when in the DISCARD or RECEIVE states.

14.10.9 rxPTPPacketDiscardCount

A counter that increments every time a PTP message of the respective PTP Instance is discarded, caused by the occurrence of any of the following conditions:

a) A received Announce message is not qualified, denoted by the function qualifyAnnounce (see 10.3.10.2.1 and 13.1.3.4) of the PortAnnounceReceive state machine (see 10.3.10 and Figure 10-12) returning FALSE;
b) A Follow_Up message corresponding to a received Sync message is not received, denoted by a transition of the condition (currentTime >= followUpReceiptTimeoutTime) to TRUE from FALSE when in the WAITING_FOR_FOLLOW_UP state of the MDSyncReceiveSM state machine (see 11.2.13 and Figure 11-6);
c) A Pdelay_Resp message corresponding to a transmitted Pdelay_Req message is not received, denoted by a transition from the WAITING_FOR_PDELAY_RESP state to the RESET state of the MDPdelayReq state machine (see 11.2.15 and Figure 11-8);
d) A Pdelay_Resp_Follow_Up message corresponding to a transmitted Pdelay_Req message is not received, denoted by a transition from the WAITING_FOR_PDELAY_RESP_FOLLOW_UP state to the RESET state of the MDPdelayReq state machine (see 11.2.15 and Figure 11-8).

14.10.10 syncReceiptTimeoutCount

A counter that increments every time sync receipt timeout occurs, denoted by entering the AGED state of the PortAnnounceInformation state machine (see 10.3.11 and Figure 10-13), with the condition (currentTime>= syncReceiptTimeoutTime && gmPresent) evaluating to TRUE.

14.10.11 announceReceiptTimeoutCount

A counter that increments every time announce receipt timeout occurs, denoted by entering the AGED state of the PortAnnounceInformation state machinefrom the CURRENT state of the PortAnnounceInformation state machine (see 10.3.11 and Figure 10-13), with the condition (currentTime >= announceReceiptTimeoutTime) evaluating to TRUE.

14.10.12 pdelayAllowedLostResponsesExceededCount

A counter that increments every time the value of the variable lostResponses (see, 11.2.17.2.10) exceeds the value of the variable allowedLostResponses (see 11.2.12.4), in the RESET state of the MDPdelayReq state machine (see 11.2.15 and Figure 11-8).

14.10.13 txSyncCount

A counter that increments every time~~-~~synchronization information is transmitted, denoted by a transition to TRUE from FALSE of the rcvdMDSync variable of the MDSyncSendSM state machine (see 11.2.14.1.1 and Figure 11-7), when in the INITIALIZING, ~~or~~ SEND_FOLLOW_UP, or SET_CORRECTION_FIELD states; or the INITIATE_REQUEST_WAIT_CONFIRM state is entered in Figure 12-4.

14.10.14 txOneStepSyncCount

A counter that increments every time a one-step Sync message is transmitted, denoted by a transition to TRUE from FALSE of the rcvdMDSync variable of the MDSyncSendSM state machine (see 11.2.14.1.1 and Figure 11-7) with the variable oneStepTxOper TRUE, when in the INITIALIZING, SEND_FOLLOW_UP, or SET_CORRECTION_FIELD states.

<<Editor's note: time-synchronization event messages transmitted on 802.11 links are considered to be two-step. time-synchronization event messages transmitted on EPON links, or on CSN links for the case where a CSN network reference clock is present, are considered to be internal to the timing mechanism of the medium and are not counted.>>

14.10.15 txFollowUpCount

A counter that increments every time a Follow_Up message is transmitted, denoted by a transition to TRUE from FALSE of the rcvdMDTimestampReceive variable of the MDSyncSendSM state machine (see 11.2.14.1.3 and Figure 11-7), when in the SEND_SYNC state.

14.10.16 txPdelayRequestCount

A counter that increments every time a Pdelay_Req message is transmitted, denoted by entering the INITIAL_SEND_PDELAY_REQ or SEND_PDELAY_REQ states of the MDPdelayReq state machine (see 11.2.15 and Figure 11-8).

14.10.17 txPdelayResponseCount

A counter that increments every time a Pdelay_Resp message is transmitted, denoted by a transition to TRUE from FALSE of the rcvdPdelayReq variable of the MDPdelayResp state machine (see 11.2.18.2.1 and

Figure 11-9) when in the WAITING_FOR_PDELAY_REQ or INITIAL_WAITING_FOR_PDELAY_REQ states, and resulting entry to the SENT_PDELAY_RESP_WAITING_FOR_TIMESTAMP state.

14.10.18 txPdelayResponseFollowUpCount

A counter that increments every time a Pdelay_Resp_Follow_Up message is transmitted, denoted by a transition to TRUE from FALSE of the rcvdMDTimestampReceive variable of the MDPdelayResp state machine (see 11.2.18.2.2 and Figure 11-9) when in the SENT_PDELAY_RESP_WAITING_FOR_TIMESTAMP state, and resulting entry to the WAITING_FOR_PDELAY_REQ state.

14.10.19 txAnnounceCount

A counter that increments every time an Announce message is transmitted, denoted by entering the TRANSMIT_ANNOUNCE state of the PortAnnounceReceive state machine (see 10.3.15 and Figure 10-17).

14.10.20 ~~Port Parameter Statistics~~Port Parameter Statistics Data Set Table

There is one Port Parameter Statistics Data Set Table per port~~,~~ per PTP Instance of a time-aware system. Each Port Parameter Statistics Data Set Table contains a set of counters for each port that supports the ~~time synchronization~~time-synchronization capability, as detailed in Table 14-12. Each table can be created or removed dynamically in implementations that support dynamic configuration of ports and components.

Table 14-12—Port Parameter Statistics Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
rxSyncCount	UInteger32	R	O	14.10.2
rxOneStepSyncCount	UInteger32	R	O	14.10.3
rxFollowUpCount	UInteger32	R	O	14.10.4
rxPdelayRequestCount	UInteger32	R	O	14.10.5
rxPdelayResponseCount	UInteger32	R	O	14.10.6
rxPdelayResponseFollowUp-Count	UInteger32	R	O	14.10.7
rxAnnounceCount	UInteger32	R	O	14.10.8
rxPTPPacketDiscardCount	UInteger32	R	O	14.10.9
syncReceiptTimeoutCount	UInteger32	R	O	14.10.10
announceReceiptTimeout-Count	UInteger32	R	O	14.10.11
pdelayAllowedLostResponses-ExceededCount	UInteger32	R	O	14.10.12
txSyncCount	UInteger32	R	O	14.10.13
txOneStepSyncCount	UInteger32	R	O	14.10.14
txFollowUpCount	UInteger32	R	O	14.10.15
txPdelayRequestCount	UInteger32	R	O	14.10.16
txPdelayResponseCount	UInteger32	R	O	14.10.17
txPdelayResponseFollowUp-Count	UInteger32	R	O	14.10.18
txAnnounceCount	UInteger32	R	O	14.10.19

[a]R= Read only access.
[b]O= Optional.

14.11 Acceptable Master Port Parameter Data Set

The Acceptable Master Port Parameter Data Set represents the capability to enable/disable the acceptable master table feature on a port.

14.11.1 General

For the single port of a time-aware end station and for each port of a time-aware relay, the Acceptable Master Port Parameter Data Set contains the single member acceptableMasterTableEnalbed, which is used to enable/disable the Acceptable Master Table Feature. The number of such data sets is the same as the numberPorts value of the Default Parameter Data Set.

14.11.2 acceptableMasterTableEnabled

The value is equal to the value of the Boolean acceptableMasterTableEnabled (see 13.1.3.1 and 13.1.3.5).

14.11.3 Acceptable Master Port Parameter Data Set Table

There is one Acceptable Master Port Parameter Data Set Table per port, per PTP Instance of a time-aware system as detailed in Table 14-13.

Table 14-13—Acceptable Master Port Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
acceptableMasterT-ableEnabled	Boolean	RW	Tdot3ONU	14.11.2

[a]R = Read only access; RW = Read/write access.
[b]Tdot3ONU = Required for time-aware IEEE 802.3 EPON ONU ports.

14.12 Common Mean Link Delay Service Default Parameter Data Set

The Common Mean Link Delay Service Default Parameter Data Set describes the per-time-aware-system attributes of the Common Mean Link Delay Service.

14.12.1 clockIdentity

The value is the clockIdentity, see 8.5.2.2, that will be used to identify the Common Mean Link Delay Service.

14.12.2 numberLinkPorts

The value is the number of Link Ports of the time-aware system on which the Common Mean Link Delay Service is implemented. For an end station the value is 1.

14.12.3 sdoId

The value is 0x200. This is the sdoId for the Common Mean Link Delay Service (see 11.2.15).

NOTE—The attribute sdoId is specified as a 12-bit unsigned integer in 8.1. The data type for the managed object sdoId is UInteger16 in Table 14-1, for compatibility with IEEE Std 1588. The range of the managed object is limited to 12 bits; in addition, only the single value 0x100 is specified in this standard for the gPTP domain of a PTP Instance.

14.12.4 Common Mean Link Delay Service Default Parameter Data Set Table

There is one Common Mean Link Delay Service Default Parameter Data Set Table per time-aware system, as detailed in Table 14-14.

Table 14-14—Common Mean Link Delay Service Default Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
clockIdentity	ClockIdentity	R	T	14.12.1
numberLinkPorts	UInteger16	R	T	14.12.2
sdoId	UInteger16	R	T	14.12.3

[a]R = Read only access; RW = Read/write access.
[b]T = Required for time-aware port.

14.13 Common Mean Link Delay Service Link Port Parameter Data Set

The Common Mean Link Delay Service Link Port Parameter Data Set represents time-aware Link Port capabilities for the Common Mean Link Delay Service of a Link Port of a time-aware system.

14.13.1 General

For the single Link Port of the Common Mean Link Delay Service of a time-aware end station and for each Link Port of the Common Mean Link Delay Service of a time-aware relay, the following Link Port Parameter Data Set is maintained as the basis for protocol decisions and providing values for message fields. The number of such data sets is the same as the numberLinkPorts value of the Common Mean Link Delay Service Default Parameter Data Set.

14.13.2 portIdentity

The value is the portIdentity attribute of the local port, see 8.5.2.

14.13.3 cmldsLinkPortEnabled

The value is equal to the value of the Boolean cmldsLinkPortEnabled (see 11.2.16.1).

14.13.4 isMeasuringDelay

The value is equal to the value of the Boolean isMeasuringDelay (see 11.2.12.6 and 16.4.3.2).

14.13.5 asCapableAcrossDomains

The value is equal to the value of the Boolean asCapableAcrossDomains (see 11.2.2 and 11.2.12.12).

14.13.6 neighborPropDelay

The value is equal to the value of the per-port global variable neighborPropDelay (see 10.2.4.8). It is an estimate of the current one-way propagation time on the link attached to this Link Port, measured as specified for the respective medium (see 11.2.15, 12.5.2, and 16.4). The value is zero for Link Ports attached to IEEE 802.3 EPON links and for the master port of an IEEE 802.11 link, because one-way propagation delay is not measured on the latter and not directly measured on the former. It is recommended that the data type be scaledNs. The default value is zero.

14.13.7 neighborPropDelayThresh

The value is equal to the value of the per-Link-port global variable neighborPropDelayThresh (see 11.2.12.7). It is the propagation time threshold, above which a Link Port (and therefore any PTP ports that use the CMLDS on this Link Port) is not considered capable of participating in the IEEE 802.1AS protocol.

14.13.8 delayAsymmetry

The value is the asymmetry in the propagation delay on the link attached to this Link Port relative to the grandmaster time base, as defined in 10.2.4.9 and 8.3. If propagation delay asymmetry is not modeled, then delayAsymmetry is 0.

14.13.9 neighborRateRatio

The value is an estimate of the ratio of the frequency of the LocalClock entity of the time-aware system at the other end of the link attached to this Link Port, to the frequency of the LocalClock entity of this time-aware system (see 10.2.4.7). neighborRateRatio is expressed as the fractional frequency offset multiplied by 2^{41}, i.e., the quantity (neighborRateRatio – 1.0)(2^{41}).

14.13.10 initialLogPdelayReqInterval

If useMgtSettableLogPdelayReqInterval is TRUE then, for full-duplex, IEEE 802.3 media and CSN media that use the peer delay mechanism to measure path delay (see 16.4.3.1), the value is the logarithm to base 2 of the Pdelay_Req message transmission interval used when (a) the Link Port is initialized, or (b) a message interval request TLV is received with the linkDelayInterval field set to 126 (see 11.5.2.2 and the LinkDelayIntervalSetting state machine, 11.2.19).

For all other media, the value is 127.

14.13.11 currentLogPdelayReqInterval

For full-duplex, IEEE 802.3 media and CSN media that use the peer delay mechanism to measure path delay (see 16.4.3.1), the value is the logarithm to the base 2 of the current Pdelay_Req message transmission interval, see 11.5.2.2.

For all other media, the value is 127.

14.13.12 useMgtSettableLogPdelayReqInterval

The managed object is a boolean that determines the source of the sync interval and mean time interval between successive Pdelay_Req messages. If the value is TRUE, the value of currentLogPdelayReqInterval is set equal to the value of mgtSettableLogPdelayReqInterval (see 14.13.13). If the value of the managed object is FALSE, the value of currentLogPdelayReqInterval is determined by the LinkDelayIntervalSetting state machine (see 11.2.20). The default value of useMgtSettableLogPdelayReqInterval is FALSE.

<<Editor's note: The resolution of comment #17 against D4.3 indicated that the default is to use the management-settable variables for domains other than 0 and to use Signaling messages with management for initialization for domain 0. For the peer delay process, we use the common mean link delay service, and there are no domains. However, in the case that only one domain is present, that

domain is domain 0, and the situation is as if the pdelay messages are sent on domain 0. Therefore, this is initialized to FALSE. This also preserves backward compatibility.

Also, this managed object, along with others pertaining to pdelay, may be moved to a common mean link delay service data set when data sets for this service are defined.>>

14.13.13 mgtSettableLogPdelayReqInterval

The value is the logarithm to base 2 of the mean time interval between successive Pdelay_Req messages if useMgtSettableLogPdelayReqInterval is TRUE. The value is not used if useMgtSettableLogPdelayReqInterval is FALSE.

14.13.14 allowedLostResponses

The value is equal to the value of the per-Link-Port global variable allowedLostResponses (see 11.5.3 and 11.2.12.4). It is the number of Pdelay_Req messages for which a valid response is not received, above which a Link Port is considered to not be exchanging peer delay messages with its neighbor.

14.13.15 allowedFaults

The value is equal to the value of the per-Link-Port global variable allowedFaults (see 11.5.4 and 11.2.12.5). It is the number of faults (see 11.5.4), above which asCapableAcrossDomains is set to FALSE, i.e., a Link Port is considered to not be be capable of interoperating with its neighbor via the 802.1AS protocol (see 10.2.4.1).

14.13.16 versionNumber

This value is set to versionPTP as specified in 10.6.2.2.4.

14.13.17 pdelayTruncatedTimestampsArray

For full-duplex IEEE 802.3 media, and CSN media that use the peer delay mechanism to measure path delay (see 16.4.3.1), the values of the four elements of this array are as described in Table 14-15. For all other media, the values are zero. Array elements 0, 1, 2, and 3 correspond to the timestamps t1, t2, t3, and t4, modulo 2^{32}, respectively, in Figure 7-7, and expressed in units of 2^{-16} ns (i.e., the value of each array element is equal to the remainder obtained upon dividing the respective timestamp, expressed in units of 2^{-16} ns, by 2^{48}). At any given time, the timestamp values stored in the array are for the same, and most recently completed, peer delay message exchange.

<<Editor's note: The above managed objects contain values associated with the timestamps t1, t2, t3, and t4 (in array elements 0, 1, 2, and 3, respectively) of the peer delay exchange. We use timestamps associated with the peer delay exchange rather than with Sync and Follow_Up, because the peer delay mechanism is used here. The timestamps associated with the Sync message, i.e., the syncEventEgressTimestamp and syncEventIngressTimestamp cannot be used because they are not both available at the same Link Port, and because they are associated with a message sent only in one direction.>>

<<Editor's note: This managed object was added for use with the asymmetry measurement compensation procedure, based on line-swapping. However, it is included in the cmldsLink PortDS, rather than the cmldsAsymmetryMeasurementModeDS because the Pdelay timestamps can be made available regardless of whether or not the Link Port is in asymmetry measurement mode. Comments are requested (if there are no comments, it will remain where it is).>>

14.13.18 minorVersionNumber

This value is set to minorVersionPTP as specified in 10.6.2.2.3.

Table 14-15—Description of pdelayTruncatedTimestampsArray

Array element	Timestamp description	Corresponding timestamp of Figure 7-7	Units
0	pdelayReqEventEgressTimestamp for Pdelay_Req message, of most recently-completed peer delay message exchange, transmitted by this time-aware system (NOTE 1)	t1	2^{-16} ns
1	pdelayReqEventIngressTimestamp for Pdelay_Req message received at peer delay responder that this Link Port sends Pdelay_Req to, of most recently-completed peer delay message exchange; it is equal to the sum of the following: a) the ns field of the requestReceiptTimestamp (see Table 11-14), multiplied by 2^{16}, and b) the correctionField (see Table 11-6) (NOTE 2)	t2	2^{-16} ns
2	pdelayRespEventEgressTimestamp for Pdelay_Resp message, of most recently-completed peer delay message exchange, transmitted by peer delay responder that this Link Port sends Pdelay_Req to; it is equal to the sum of the following: a) the ns field of the responseOriginTimestamp (see Table 11-15), multiplied by 2^{16}, b) the correctionField (see Table 11-6). (NOTE 3)	t3	2^{-16} ns
3	pdelayRespEventIngressTimestamp for Pdelay_Resp message, of most recently-completed peer delay message exchange, received by this time-aware system (NOTE 4)	t4	2^{-16} ns

Table 14-15—Description of pdelayTruncatedTimestampsArray

Array element	Timestamp description	Corresponding timestamp of Figure 7-7	Units
NOTE 1—This quantity is not simply the nanoseconds plus fractional nanoseconds portion of the pdelayReqEventEgressTimestamp. Rather, it is equal to [pdelayReqEventEgressTimestamp.seconds × $(10^9)(2^{16})$ + pdelayReqEventEgressTimestamp.fractionalNanoseconds] mod 2^{48}, where pdelayReqEventEgressTimestamp is expressed as an ExtendedTimestamp (see 6.4.3.5). NOTE 2—This quantity is not simply the nanoseconds plus fractional nanoseconds portion of the pdelayReqEventIngressTimestamp. Rather, it is equal to [pdelayReqEventIngressTimestamp>.seconds × $(10^9)(2^{16})$ + pdelayReqEventIngressTimestamp.fractionalNanoseconds] mod 2^{48}, where pdelayReqEventIngressTimestamp is expressed as an ExtendedTimestamp (see 6.4.3.5). NOTE 3—This quantity is not simply the nanoseconds plus fractional nanoseconds portion of the pdelayRespEventIngressTimestamp. Rather, it is equal to [pdelayRespEventIngressTimestamp.seconds × $(10^9)(2^{16})$ + pdelayRespEventIngressTimestamp.fractionalNanoseconds] mod 2^{48}, where pdelayRespEventIngressTimestamp is expressed as an ExtendedTimestamp (see 6.4.3.5). NOTE 4—This quantity is not simply the nanoseconds plus fractional nanoseconds portion of the pdelayRespEventIngressTimestamp. Rather, it is equal to [pdelayRespEventIngressTimestamp.seconds × $(10^9)(2^{16})$ + pdelayRespEventIngressTimestamp.fractionalNanoseconds] mod 2^{48}, where pdelayRespEventIngressTimestamp is expressed as an ExtendedTimestamp (see 6.4.3.5). Its units are 2^{-16} ns.			

14.13.19 Common Mean Link Delay Service Link Port Parameter Data Set Table

There is one Common Mean Link Delay Service Link Port Parameter Data Set Table per Link Port, for the PTP Instance of a time-aware system. Each Common Mean Link Delay Service Link Port Parameter Data Set Table contains a set of parameters for each Link Port that supports the time-synchronization capability, as detailed in Table 14-16. Each table can be created or removed dynamically in implementations that support dynamic configuration of Link Ports and components.

14.14 Common Mean Link Delay Service Description Link Port Data Set

The Common Mean Link Delay Service Description Link Port Parameter Data Set contains the profileIdentifier for this PTP profile, as specified in F.1.

<<Editor's note: In accordance with comment 33 against D4.2, only the profileIdentifier member is included, and not the protocolAddress (i.e., MAC address in this case). Given that protocolAddress is not included, this data set technically does not need to be per Link Port; it is assumed to be per Link Port for consistency with 1588-Rev. Should protocolAddress, i.e., MAC address, be included?>>

14.14.1 profileIdentifier

The value is the profileIdentifier for this PTP profile (see F.1).

14.14.2 Common Mean Link Delay Service Description Link Port Parameter Data Set Table

There is one Common Mean Link Delay Service Description Link Port Parameter Data Set Table per Link Port of a time-aware system, as detailed in Table 14-17.

Table 14-16—Common Mean Link Delay Service Link Port Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
portIdentity	PortIdentity (see 6.4.3.7)	R	T	14.13.2
cmldsLinkPortEnabled	Boolean	R	T	14.13.3
isMeasuringDelay	Boolean	R	T	14.13.4
asCapableAcrossDomains	Boolean	R	T	14.13.5
neighborPropDelay	UScaledNs (recommended)	R	T	14.13.6
neighborPropDelayThresh	UScaledNs (recommended)	RW	T	14.13.7
delayAsymmetry	scaledNs (recommended)	RW	Tdot3FD	14.13.8
neighborRateRatio	Integer32	R	T	14.13.9
initialLogPdelayReqInterval	Integer8	RW	T	14.13.10
currentLogPdelayReqInterval	Integer8	R	T	14.13.11
useMgtSettableLogPdelayReq-Interval	Boolean	RW	T	14.13.12
mgtSettableLogPdelayReqIn-terval	Integer8	RW	T	14.13.13
allowedLostResponses	UInteger16	RW	T	14.13.14
allowedFaults	UInteger16	RW	T	14.13.15
versionNumber	UInteger4	R	T	14.13.16
pdelayTruncatedTimestamp-sArray	UInteger48[4]	R	O	14.13.17
minorVersionNumber	UInteger4	R	T	14.13.18

[a]R = Read only access; RW = Read/write access.
[b]T= Required for time-aware Link Port;
O = Optional for time-aware Link Port;
Tdot3FD = Required for time-aware Link Port associated with IEEE 802.3 full-duplex port.

Table 14-17—Common Mean Link Delay Service Description Link Port Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
profileIdentifier	Octet6	R	O	14.10.2

[a]R= Read only access.
[b]O= Optional.

<<Editor's note: Comment 33 of D4.2 did not indicate whether this data set, and/or the profileIdentifier member, should be required or optional. For now, the member is listed as optional, as this data set and member were not included in 802.1AS-2011. Comments are requested on this.>>

14.15 Common Mean Link Delay Service Link Port Parameter Statistics Data Set

The Common Mean Link Delay Service Link Port Parameter Statistics Data Set provides counters associated with Link Port capabilities at a given time-aware system.

14.15.1 General

For the single Link Port of a time-aware end station and for each Link Port of a time-aware relay, the following Common Mean Link Delay Service Link Port Parameter Statistics Data Set provides counters. The number of such statistics sets is the same as the numberLinkPorts value of the Default Parameter Data Set.

14.15.2 rxPdelayRequestCount

A counter that increments every time a Pdelay_Req message is received, denoted by a transition to TRUE from FALSE of the rcvdPdelayReq variable of the MDPdelayResp state machine (see 11.2.18.2.1 and Figure 11-9) when in the WAITING_FOR_PDELAY_REQ or INITIAL_WAITING_FOR_PDELAY_REQ states.

14.15.3 rxPdelayResponseCount

A counter that increments every time a Pdelay_Resp message is received, denoted by a transition to TRUE from FALSE of the rcvdPdelayResp variable of the MDPdelayReq state machine (see 11.2.17.2.2 and Figure 11-8) when in the WAITING_FOR_PDELAY_RESP state.

14.15.4 rxPdelayResponseFollowUpCount

A counter that increments every time a Pdelay_Resp_Follow_Up message is received, denoted by a transition to TRUE from FALSE of the rcvdPdelayRespFollowUp variable of the MDPdelayReq state machine (see 11.2.17.2.4 and Figure 11-8) when in the WAITING_FOR_PDELAY_RESP_FOLLOW_UP state.

14.15.5 rxPTPPacketDiscardCount

A counter that increments every time a PTP message of the Common Mean Link Delay Service is discarded, caused by the occurrence of any of the following conditions:

A Pdelay_Resp message corresponding to a transmitted Pdelay_Req message is not received, denoted by a transition from the WAITING_FOR_PDELAY_RESP state to the RESET state of the MDPdelayReq state machine (see 11.2.15 and Figure 11-8);

A Pdelay_Resp_Follow_Up message corresponding to a transmitted Pdelay_Req message is not received, denoted by a transition from the WAITING_FOR_PDELAY_RESP_FOLLOW_UP state to the RESET state of the MDPdelayReq state machine (see 11.2.15 and Figure 11-8).

14.15.6 pdelayAllowedLostResponsesExceededCount

A counter that increments every time the value of the variable lostResponses (see, 11.2.17.2.10) exceeds the value of the variable allowedLostResponses (see 11.2.12.4), in the RESET state of the MDPdelayReq state machine (see 11.2.15 and Figure 11-8).

14.15.7 txPdelayRequestCount

A counter that increments every time a Pdelay_Req message is transmitted, denoted by entering the INITIAL_SEND_PDELAY_REQ or SEND_PDELAY_REQ states of the MDPdelayReq state machine (see 11.2.15 and Figure 11-8).

14.15.8 txPdelayResponseCount

A counter that increments every time a Pdelay_Resp message is transmitted, denoted by a transition to TRUE from FALSE of the rcvdPdelayReq variable of the MDPdelayResp state machine (see 11.2.18.2.1 and Figure 11-9) when in the WAITING_FOR_PDELAY_REQ or INITIAL_WAITING_FOR_PDELAY_REQ states, and resulting entry to the SENT_PDELAY_RESP_WAITING_FOR_TIMESTAMP state.

14.15.9 txPdelayResponseFollowUpCount

A counter that increments every time a Pdelay_Resp_Follow_Up message is transmitted, denoted by a transition to TRUE from FALSE of the rcvdMDTimestampReceive variable of the MDPdelayResp state machine (see 11.2.18.2.2 and Figure 11-9) when in the SENT_PDELAY_RESP_WAITING_FOR_TIMESTAMP state, and resulting entry to the WAITING_FOR_PDELAY_REQ state.

14.15.10 Common Mean Link Delay Service Link Port Parameter Statistics Data Set Table

There is one Common Mean Link Delay Service Link Port Parameter Statistics Data Set Table per Link Port of a time-aware system. The Common Mean Link Delay Service Link Port Parameter Statistics Data Set Table contains a set of counters for each Link Port that supports the time-synchronization capability, as detailed in Table 14-18. Each table can be created or removed dynamically in implementations that support dynamic configuration of Link Ports and components.

Table 14-18—Common Mean Link Delay Service Link Port Parameter Statistics Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
rxPdelayRequestCount	UInteger32	R	O	14.10.5
rxPdelayResponseCount	UInteger32	R	O	14.10.6
rxPdelayResponseFollowUp-Count	UInteger32	R	O	14.10.7
rxPTPPacketDiscardCount	UInteger32	R	O	14.10.9
pdelayAllowedLostResponses-ExceededCount	UInteger32	R	O	14.10.12
txPdelayRequestCount	UInteger32	R	O	14.10.16
txPdelayResponseCount	UInteger32	R	O	14.10.17
txPdelayResponseFollowUp-Count	UInteger32	R	O	14.10.18

[a]R= Read only access.
[b]O= Optional.

14.16 Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set

The Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set represents the capability to enable/disable the Asymmetry Compensation Measurement Procedure on a Link Port (see Annex G).

14.16.1 General

For the single Link Port of a time-aware end station and for each Link Port of a time-aware relay, the Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set contains the single member asymmetryMeasurementMode, which is used to enable/disable the Asymmetry Compensation Measurement Procedure. The number of such data sets is the same as the numberLinkPorts value of the Common Mean Link Delay Service Default Parameter Data Set.

14.16.2 asymmetryMeasurementMode

The value is equal to the value of the Boolean asymmetryMeasurementMode(see G.3). For full-duplex IEEE 802.3 media, the value is TRUE if an asymmetry measurement is being performed for the link attached to this Link Port, and FALSE otherwise. For all other media, the value shall be FALSE (see 10.2.4.2).

NOTE—If an asymmetry measurement is being performed for a link, asymmetryMeasurementMode must be TRUE for the Link Ports at each end of the link.

14.16.3 Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set Table

There is one Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set Table for all PTP Instances, per Link Port, as detailed in Table 14-19.

Table 14-19—Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
asymmetryMeasure-mentMode	Boolean	RW	O	14.16.2

[a]R = Read only access; RW = Read/write access.
[b]Tdot3ONU = Required for time-aware IEEE 802.3 EPON ONU Link Ports.

14.17 Common Mean Link Delay Service EPON Parameter Data Set

The Common Mean Link Delay Service EPON Parameter Data Set describes the per-Link-Port attributes of the Common Mean Link Delay Service that are specific to IEEE 802.3 EPON links (see Clause 13). This data set is present only for IEEE 802.3 EPON links.

14.17.1 nup

For an OLT Link Port of an IEEE 802.3 EPON link, the value is the effective index of refraction for the EPON upstream wavelength light of the optical path (see 13.1.4 and 13.8.1.2.2). The default value is 1.46770 for 1 Gb/s upstream links, and 1.46773 for 10 Gb/s upstream links.

For all other Link Ports, the value is 0.

14.17.2 ndown

For an OLT Link Port of an IEEE 802.3 EPON link, the value is the effective index of refraction for the EPON downstream wavelength light of the optical path (see 13.1.4 and 13.8.1.2.1). The default value is 1.46805 for 1 Gb/s downstream links, and 1.46851 for 10 Gb/s downstream links.

For all other Link Ports, the value is 0.

14.17.3 Common Mean Link Delay Service EPON Parameter Data Set Table

There is one Common Mean Link Delay Service EPON Parameter Data Set Table per Link Port of a time-aware system, for which the transport on the attached link is IEEE 802.3 EPON, as detailed in Table 14-17.

Table 14-20—Common Mean Link Delay Service EPON Parameter Data Set Table

Name	Data type	Operations supported[a]	Conformance[b]	References
nup	Double	RW	Tdot3OLT	14.17.1
ndown	Double	RW	Tdot3OLT	14.17.2

[a]RW = Read/write access.
[b]Tdot3OLT = Required for time-aware IEEE 802.3 EPON OLT Link Ports.

15. Managed object definitions

<<Editor's note: The MIB must be changed to allow for multiple domains. There is one set of managed objects and datasets for each domain (i.e., all the MIB objects are replicated for each domain).

However, a number of managed objects (mainly in 14.8) correspond to variables that are common to all domains. It must be decided how this will be handled in the MIB. Comments are requested on this.>>

<<Editor's note: The domainNumber object of the default data set must be added to the MIB.>>

<<Editor's note: The object pdelayTruncatedTimestampsArray, whose members contain the peer delay timestamps, and the object asymmetryMeasurementMode, associated with the feature for management support for compensation of link asymmetry, must be added to the MIB (see 14.8.35). The timestamps contained in pdelayTruncatedTimestampsArray must be from the same, and most recently completed, peer delay message exchange.>>

<<Editor's note: The per-port objects oneStepReceive (see 14.8.36), oneStepTransmit (see 14.8.37), and oneStepTxOper (see 14.8.35) must be added to the MIB.

<<Editor's note: The MIB must be changed to reflect the renaming of portEnabled; the name is changed to portOper. The MIB must be changed to reflect the renaming of pttPortEnabled; the name is changed to ptpPortEnabled.>>

<<Editor's note: Cross references in the MIB file (cl15mib.fm) must be checked; a number of them seem to have been deleted when going from D0.8 to D0.9, in Framemaker. (D0.9 was the first draft where the title and filename reflected 802.1AS-Rev (i.e., not 802.1ASbt), though the document has been in revision format since D0.6. This appears to be a Framemaker cross-reference problem. This issue should be resolved before any editing of the MIB (i.e., of the file cl15mib.fm) is done. >>

<<Editor's note: Cross references to Annex E must be changed to Clause 16, since the Annex E content has been moved to Clause 16.>>

<<Editor's note: The MIB must reflect the resolution of the handling of linkAgg, in accordance with the resolution of comment #58 against D1.0. That comment reads:

Comment Type: MIB shoud hot be tied to Bridge Ports, but to the Interface Stack

SuggestedRemedy: Change to git rid of all dependency on the Bridge MIBs. Tie to the Interface Stack, instead.

Response: ACCEPT IN PRINCIPLE. Group comments 19, 46, 56, 57, 58, 88. The MIB table with the per-port variables is indexed by both the bridge port number and the ifIndex. Therefore, no fundamental change to the MIB table is required. The descriptions of the indices of this table do not make it clear how to apply the table to a non-bridge time relay or to an aggregated link. Norm Finn will propose an alternative description.

Mike Thornburg points out that we will need new MIB tables indexed by domain. These tables will be indexed by domain and ifIndex>>

<<Editor's note: The descriptions of timeSource, on pages 228 and 234, must be updated in accordance with the update made to the first paragraph of 8.6.2.7.>>

<<Editor's note: The per-port object allowedFaults (see 14.13.15) must be added to the MIB.

<<Editor's note: The per-time-aware-system object sdoId (see 14.2.16) must be added to the MIB.>>

<<Editor's note: The per-port object delayMechanism (see 14.8.5) must be added to the MIB.>>

<<Editor's note: The per-time-aware-system object numberPorts is indicated as having data type Unsigned32; however, the range seems to be 0 to 255. The range should be 1 to the maximum allowed by the Unsigned32 data type. Actually, in Clause 14 the data type for numberPorts is UInteger16; it should be the same here.>>

The clause contains a complete SMIv2 Management Information Base (MIB) set for all features of this standard.

15.1 Internet Standard Management Framework

For a detailed overview of the documents that describe the current Internet Standard Management Framework, please refer to section 7 of IETF RFC 3410 (2002).

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This clause specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, IETF RFC 2578, IETF RFC 2579, and IETF RFC 2580.

15.2 Structure of the MIB

The IEEE 802.1AS MIB provides objects to configure and managed the IEEE 802.1AS Timing and Synchronization for Time-Sensitive Applications~~ in Bridged LAN~~.

The MIB contains a set of textual conventions and is additionally subdivided into seven subtrees. Each subtree is organized as a set of related objects. They are as follows:

1) The default data set represents the native capabilities of a time-aware system, i.e., a ~~bridge~~time-aware relay~~,~~ or a~~n~~ time-aware end station.
2) The current data set represents topological position of a local system relative to the grandmaster;
3) The parent data set represents capabilities of the upstream system, toward the grandmaster as measured at a local system.
4) The time properties data set represents capabilities of the grandmaster, as measured at a local system.
5) The port data set represents time-aware capabilities at a given ~~bridge~~time-aware relay or time-aware end station port, as a set of augmentation to the interface table entry (ifEntry).
6) The port statistics represent statistics and counters associated with time-aware capabilities at a given ~~Bridge~~time-aware relay or time-aware end station port.
7) The acceptable master table data set represents the acceptable master table used when media-dependent port type of EPON is present in a time-aware system.

Table 15-1 show the structure of the MIB and the relationship of the MIB objects to the default data set, current data set, parent data set, time properties data set, port data set, port statistics, and acceptable master table data set.

15.3 Security considerations

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in these MIB module.

It is recommended that implementers consider the security features as provided by the SNMPv3 framework (see IETF RFC 3410, section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is not recommended. Instead, it is recommended to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of these MIB modules is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

There are a number of management objects defined in the IEEE8021-AS MIB module that have a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than notaccessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

The following objects in the IEEE8021-AS MIB can be manipulated to interfere with the operation of timing synchronization. This could, for example, be used to force a reinitialization of state machines, thus causing timing synchronization and network instability. Another possibility would be for an attacker to override grandmaster status, thus giving a user (or an attacker) unauthorized control over the network time.

Improper manipulation of the following writable objects could result in an unintended grandmaster to be elected, when a system is grandmaster capable in a gPTP domain. It could also be used maliciously to cause frequent grandmaster changes, thereby affecting network stability.

Ieee8021ASDefaultDSPriority1
Ieee8021ASDefaultDSPriority2

Improper manipulation of the following writable objects could result in a segmented time-aware network, could compromise the expected accuracy, and could interrupt paths of the PTP domain.

Ieee8021ASPortDSPtp~~t~~PortEnabled
Ieee8021ASPortDSDelayAsymmetry

Unintended access to any of the readable tables or variables in the IEEE8021-AS MIB alerts the reader that timing synchronization in gPTP domain is configured, and on which values timing parameters are configured, and which system is current grandmaster. This information can suggest to an attacker what applications are being run, and thus suggest application-specific attacks, or to enable the attacker to detect whether their attacks are being successful or not. It is thus important to control even GET access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

Table 15-1—IEEE8021-AS MIB structure and object cross reference

MIB table	MIB object	Reference
ieee802AsDefaultDataSet		Default Parameter Data Set Table 14-1
	ieee802AsDefaultDSClockIdentity	14.2.1
	ieee802AsDefaultDSNumberPorts	14.2.2
	ieee802AsDefaultDSClockClass	14.2.3.1
	ieee802AsDefaultDSClockAccuracy	14.2.3.2
	ieee802AsDefaultDSOffsetScaledLogVariance	14.2.3.3
	ieee802AsDefaultDSPriority1	14.2.4
	ieee802AsDefaultDSPriority2	14.2.5
	ieee802AsDefaultDSGmCapable	14.2.6
	ieee802AsDefaultDSCurrentUtcOffset	14.2.7
	ieee802AsDefaultDSCurrentUtcOffsetValid	14.2.8
	ieee802AsDefaultDSLeap59	14.2.9
	ieee802AsDefaultDSLeap61	14.2.10
	ieee802AsDefaultDSTimeTraceable	14.2.11
	ieee802AsDefaultDSFrequencyTraceable	14.2.12
	ieee802AsDefaultDSTimeSource	14.2.14
ieee802AsCurrentDataSet		Current Parameter Data Set Table 14-2
	ieee802AsCurrentDSStepsRemoved	14.3.1
	ieee802AsCurrentDSOffsetFromMaster	14.3.2
	ieee802AsCurrentDSLastGmPhaseChange	14.3.3
	ieee802AsCurrentDSLastGmFreqChange	14.3.4
	ieee802AsCurrentDSGmTimebaseIndicator	14.3.5
	ieee802AsCurrentDSGmChangeCount	14.3.6
	ieee802AsCurrentDSTimeOfLastGmChangeEvent	14.3.7
	ieee802AsCurrentDSTimeOfLastGmPhaseChangeEvent	14.3.8
	ieee802AsCurrentDSTimeOfLastGmFreqChangeEvent	14.3.9
ieee802AsParentDataSet		Parent Parameter Data Set Table 14-3
	ieee802AsParentDSParentClockIdentity	14.4.1
	ieee802AsParentDSParentPortNumber	14.4.1
	ieee802AsParentDSCumulativeRateRatio	14.4.2
	ieee802AsParentDSGrandmasterIdentity	14.4.3
	ieee802AsParentDSGrandmasterClockClass	14.4.5
	ieee802AsParentDSGrandmasterClockAccuracy	14.4.6
	ieee802AsParentDSGrandmasterOffsetScaledLogVariance	14.4.7
	ieee802AsParentDSGrandmasterPriority1	14.4.8
	ieee802AsParentDSGrandmasterPriority2	14.4.9
ieee802AsTimePropertiesDataSet		Time Properties Parameter Data Set Table 14-4
	ieee802AsTimePropertiesDSCurrentUTCOffset	14.5.1
	ieee802AsTimePropertiesDSCurrentUTCOffsetValid	14.5.2
	ieee802AsTimePropertiesDSLeap59	14.5.3
	ieee802AsTimePropertiesDSLeap61	14.5.4
	ieee802AsTimePropertiesDSTimeTraceable	14.5.5
	ieee802AsTimePropertiesDSFrequencyTraceable	14.5.6

Table 15-1—IEEE8021-AS MIB structure and object cross reference *(continued)*

MIB table	MIB object	Reference
	ieee802AsTimePropertiesDSTimeSource	14.5.8
ieee802AsPortDataSet		Port Parameter Data Set Table 14-10
	ieee802AsPortDSClockIdentity	14.8.2
	ieee802AsPortDSPortNumber	14.8.2
	ieee802AsPortDSPort~~Role~~State	14.8.3
	ieee802AsPortDSPtp~~t~~PortEnabled	14.8.4
	ieee802AsPortDSIsMeasuringDdelay	14.8.6
	ieee802AsPortDSAsCapable	14.8.7
	ieee802AsPortDSNeighborPropDelay	14.8.8
	ieee802AsPortDSNeighborPropDelayThresh	14.8.9
	ieee802AsPortDSDelayAsymmetry	14.8.10
	ieee802AsPortDSNeighborRateRatio	14.8.11
	ieee802AsPortDSInitialLogAnnounceInterval	14.8.12
	ieee802AsPortDSCurrentLogAnnounceInterval	14.8.13
	ieee802AsPortDSAnnounceReceiptTimeout	14.8.16
	ieee802AsPortDSInitialLogSyncInterval	14.8.17
	ieee802AsPortDSCurrentLogSyncInterval	14.8.18
	ieee802AsPortDSSyncReceiptTimeout	14.8.21
	ieee802AsPortDSSyncReceiptTimeoutTimeInterval	14.8.22
	ieee802AsPortDSInitialLogPdelayReqInterval	14.8.23
	ieee802AsPortDSCurrentLogPdelayReqInterval	14.8.24
	ieee802AsPortDSAllowedLostResponses	14.8.27
	ieee802AsPortDSVersionNumber	14.8.31
	ieee802AsPortDSNup	14.8.32
	ieee802AsPortDSNdown	14.8.33
	ieee802AsPortDSAcceptableMasterTableEnabled	14.8.34
ieee802AsPortStatistics		Port Statistics Data Set Table 14-12
	ieee802AsRxSyncCount	14.10.2
	ieee802AsRxFollowUpCount	14.10.4
	ieee802AsRxPdelayRequestCount	14.10.5
	ieee802AsRxPdelayResponseCount	14.10.6
	ieee802AsRxPdelayResponseFollowUpCount	14.10.7
	ieee802AsRxAnnounceCount	14.10.8
	ieee802AsRxPTPPacketDiscardCount	14.10.9
	ieee802AsSyncReceiptTimeoutCount	14.10.10
	ieee802AsAnnounceReceiptTimeoutCount	14.10.11
	ieee802AsPdelayAllowedLostResponsesExceededCount	14.10.12
	ieee802AsTxSyncCount	14.10.13
	ieee802AsTxFollowUpCount	14.10.15
	ieee802AsTxPdelayRequestCount	14.10.16
	ieee802AsTxPdelayResponseCount	14.10.17
	ieee802AsTxPdelayResponseFollowUpCount	14.10.18
	ieee802AsTxAnnounceCount	14.10.19

Table 15-1—IEEE8021-AS MIB structure and object cross reference *(continued)*

MIB table	MIB object	Reference
ieee802AsAcceptableMasterTableDataSet		Acceptable Master Parameter Data Set Table 14-6
	ieee802AscceptableMasterTableDSMaxTableSize	14.7.1
	ieee802AscceptableMasterTableDSActualTableSize	14.7.2
	ieee802AscceptableMasterTableDSAcceptableMaster-Array	14.7.3

15.4 Textual conventions defined in this MIB

The following textual conventions are defined in this MIB:

a) ClockIdentity. IEEE 802 MAC address represented in “canonical” order defined by IEEE Std 802® [B4], EUI-64 [B2].
b) IEEE8021ASClockClassValue. Clock class value (see 8.6.2.2),
c) IEEE8021ASClockAccuracyValue. Clock accuracy value (see 8.6.2.3),
d) IEEE8021ASTimeSourceValue. Source of time used by grandmaster (see 8.6.2.7).

15.5 IEEE 802.1AS MIB module

In the following MIB module, should any discrepancy between the DESCRIPTION text and the corresponding definition in Clause 14 occur, the definition in Clause 14 shall take precedence.

<<Editor’s note: All the occurrences in the MIB of “bridge”, “bridging”, or other forms of the word must be examined and edited. For example, “time-aware bridge” should be replaced by “time-aware relay”.

Comment 142 agains draft D1.0 asked that this editor’s note be addressed. This has not been done as of draft 2.0, but will be done when the MIB edits are made by the MIB editor.>>

```
IEEE8021-AS-MIB DEFINITIONS ::= BEGIN
-- ================================================================
-- MIB for support of 802.1AS Timing and Synchronization in
-- IEEE 802.1Q Bridged Local Area Networks
-- ================================================================

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Unsigned32, Integer32, Counter32
        FROM SNMPv2-SMI              -- [RFC2578]
    TEXTUAL-CONVENTION, TruthValue, RowStatus, TimeStamp
        FROM SNMPv2-TC               -- [RFC2579]
    MODULE-COMPLIANCE, OBJECT-GROUP  -- [RFC2580]
        FROM SNMPv2-CONF
    ifGeneralInformationGroup, InterfaceIndexOrZero
        FROM IF-MIB                  -- [RFC2863]
    IEEE8021BridgePortNumber
        FROM IEEE8021-TC-MIB
    ;

ieee8021AsTimeSyncMib MODULE-IDENTITY
    LAST-UPDATED "201212120000Z" -- December 12, 2012
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
            "WG-URL: http://www.ieee802.org/1/index.html
            WG-EMail: STDS-802-1@IEEE.ORG

            Contact: Geoffrey M. Garner
            Postal:  196 Ambassador Drive
                     Red Bank, NJ 07701
                     USA
            E-mail:  gmgarner@alum.mit.edu"

    DESCRIPTION
            "The Management Information Base module for
            IEEE 802.1AS ~~time synchronization~~time-synchronization
protocol."

    REVISION "201212120000Z" -- December 12, 2012
    DESCRIPTION "Published as part of IEEE Std 802.1AS Cor-1
          ieee8021AsPortDSAsIfIndex SYNTAX corrected to
              InterfaceIndexOrZero, from Integer32.
          ieee8021AsPortDSAnnounceReceiptTimeout DESCRIPTION and
              default value corrected.
          ieee8021AsPortDSSyncReceiptTimeout DESCRIPTION corrected,
          ieee8021AsDefaultDSOffsetScaledLogVariance and
              ieee8021AsParentDSGrandmasterOffsetScaledLogVariance
              SYNTAX corrected to Unsigned32 (from Integer32),
              and corresponding range to reflect unsigned integer 16.
          ieee8021AsComplianceCor1 added.
          Other editorial and corrections made."

    REVISION "201011110000Z" -- November 11, 2010
    DESCRIPTION
            "Published as part of IEEE Std 802.1AS
```

```

              Copyright (C) IEEE (2012)."

    ::= { iso(1) org(3) ieee(111)
            standards-association-numbers-series-standards (2)
            lan-man-stds (802) ieee802dot1 (1) ieee802dot1mibs (1) 20 }

ieee8021AsMIBObjects    OBJECT IDENTIFIER ::= {ieee8021AsTimeSyncMib 1}
ieee8021AsConformance   OBJECT IDENTIFIER ::= {ieee8021AsTimeSyncMib 2}

-- ================================================================
-- Textual Conventions
-- ================================================================

ClockIdentity ::= TEXTUAL-CONVENTION
    DISPLAY-HINT
        "1x:"
    STATUS current
    DESCRIPTION
        "Represents an IEEE 802 MAC address represented in the
         `canonical' order defined by IEEE 802.1a, EUI-64.  EUI-48
         converts to EUI-64 as specified by IEEE.  The conversion
         assigns values 255 and 254 to octets 3 and 4 respectively,
         where octet 0 is the most significant and octet 7 the least.
         For example, EUI-48 of AC:DE:48:23:45:67 would extend to
         AC:DE:48:FF:FE:23:45:67."
    REFERENCE    "6.4.3.6 and 8.5.2.2.1"
    SYNTAX OCTET STRING (SIZE (8))


IEEE8021ASClockClassValue ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "Clock Class Value from IEEE Std 1588-2008 7.6.2.4,
         with the following interpretation placed on the value:

         6: A clock that is synchronized to a primary reference
            time source,
         7: A clock that has previously been designated as clockClass
            6  but that has lost the ability to synchronize to a
            primary reference time source and is in holdover mode and
            within holdover specifications,
        13: A clock that is synchronized to an application-specific
            source of time,
        14: A clock that has previously been designated as clockClass
            13 but that has lost the ability to synchronize to an
            application-specific source of time and is in holdover mode
            and within holdover specifications,
        52: Degradation alternative A for a clock of clockClass 7 that
            is not within holdover specification,
        58: Degradation alternative A for a clock of clockClass 14 that
            is not within holdover specification,
        68..122: For use by alternate PTP profiles (68..122),
       133..170: For use by alternate PTP profiles (133..170),
            187: Degradation alternative B for a clock of clockClass 7 that
                 is not within holdover specification,
            193: Degradation alternative B for a clock of clockClass 14 that
                 is not within holdover specification,
            216..232: For use by alternate PTP profiles,
            248:  Default none of the other clockClass definitions apply,
            255:  A slave-only clock(255)."
        REFERENCE    "14.2.3 and IEEE Std 1588-2008 7.6.2.4"
        SYNTAX     INTEGER {
            primarySync(6),
            primarySyncLost(7),
            applicationSpecificSync(13),
            applicationSpecficSyncLost(14),
            primarySyncAlternativeA(52),
            applicationSpecificAlternativeA(58),
            primarySyncAlternativeB(187),
            applicationSpecficAlternativeB(193),
            defaultClock(248),
            slaveOnlyClock(255)
            }

    IEEE8021ASClockAccuracyValue ::= TEXTUAL-CONVENTION
        STATUS       current
        DESCRIPTION
            "Clock Accuracy Value from 8.6.2.3, with the following
             interpretation placed on the value:

             32: The time is accurate to within 25 ns,
             33: The time is accurate to within 100 ns,
             34: The time is accurate to within 250 ns,
             35: The time is accurate to within 1 us,
             36: The time is accurate to within 2.5 us,
             37: The time is accurate to within 10 us,
             38: The time is accurate to within 25 us,
             39: The time is accurate to within 100 us,
             40: The time is accurate to within 250 us,
             41: The time is accurate to within 1 ms,
             42: The time is accurate to within 2.5 ms,
             43: The time is accurate to within 10 ms,
             44: The time is accurate to within 25 ms,
             45: The time is accurate to within 100 ms,
             46: The time is accurate to within 250 ms,
             47: The time is accurate to within 1 s,
             48: The time is accurate to within 10 s,
             49: The time is accurate to within > 10 s,
            254: Default indicating unknown"
        REFERENCE    "8.6.2.3"
        SYNTAX     INTEGER {
            timeAccurateTo25ns(32),
            timeAccurateTo100ns(33),
            timeAccurateTo250ns(34),
            timeAccurateTo1us(35),
            timeAccurateTo2dot5us(36),
            timeAccurateTo10us(37),
            timeAccurateTo25us(38),
            timeAccurateTo100us(39),
            timeAccurateTo250us(40),
            timeAccurateTo1ms(41),
            timeAccurateTo2dot5ms(42),
            timeAccurateTo10ms(43),
            timeAccurateTo25ms(44),
            timeAccurateTo100ms(45),
            timeAccurateTo250ms(46),
            timeAccurateTo1s(47),
            timeAccurateTo10s(48),
            timeAccurateToGT10s(49),
            timeAccurateToUnknown(254)
            }

IEEE8021ASTimeSourceValue ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
        "The timeSource is an information only attribute indicating
         the type of source of time used by a ClockMaster,
         representing categories.  For example, the GPS entry would
         include not only the GPS system of the U.S. Department of
         Defense but the European Galileo system and other present and
         future satellite-based timing systems.

         In the absence of a default value set by a user of this
         standard, the default value of timeSource shall be OTHER. See
        7.6.2.6 of IEEE Std 1588 - 2008 for more detailed description of
         timeSourceIndicates the source of time used by the grandmaster
         clock.

         The following interpretation placed on the value:
            16:  Atomic Clock,
            32:  GPS,
            48:  Terrestrial Radio,
            64:  PTP,
            80:  NTP,
            96:  Hand Set,
           144:  Other,
           160:  Internal Oscillator "
    REFERENCE  "8.6.2.7 and Table 8-4"
    SYNTAX      INTEGER {
            atomicClock(16),
            gps(32),
            terrestrialRadio(48),
            ptp(64),
            ntp(80),
            handSet(96),
            other(144),
            internalOscillator(160)
           }


-- =================================================================
-- subtrees in the IEEE8021-AS-MIB
-- System Time-Aware Parameters/Capability
-- ===================================================================


-- ===================================================================
-- The Default data set represent native time capability of a time-
-- aware system and is consistent with respective IEEE 1588 data set.
-- ===================================================================
ieee8021AsDefaultDS
    OBJECT IDENTIFIER ::= { ieee8021AsMIBObjects 1 }

ieee8021AsDefaultDSClockIdentity OBJECT-TYPE
    SYNTAX      ClockIdentity
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
              "Globally unique manufacturer-assigned clock identifier
               for the local clock. The identifier is based on an
               EUI-64."
    REFERENCE   "14.2.1"
    ::= { ieee8021AsDefaultDS 1 }


ieee8021AsDefaultDSNumberPorts OBJECT-TYPE
    SYNTAX      Unsigned32(0..255)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The number of PTP ports on the device.
                For an end station the value is 1."
    REFERENCE   "14.2.2"
    ::= { ieee8021AsDefaultDS 2 }


ieee8021AsDefaultDSClockClass OBJECT-TYPE
    SYNTAX      IEEE8021ASClockClassValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "Denotes the traceability of the time or frequency of the
                local clock.  The value shall be selected as follows:
                a) If the Default Parameter Data Set member gmCapable is
                   TRUE, then clockClass is set to the value that
                   reflects the combination of the LocalClock and
                   ClockSource entities; else if the value that reflects
                   the LocalClock and ClockSource entities is not
                   specified or not known,clockClass is set to 248;
               b) If the Default Parameter Data Set member gmCapable is
                   FALSE (see 8.6.2.1), clockClass is set to 255.
               "
    REFERENCE   "14.2.3.1"
    DEFVAL { defaultClock }
    ::= { ieee8021AsDefaultDS 3 }
```

```
ieee8021AsDefaultDSClockAccuracy OBJECT-TYPE
    SYNTAX      IEEE8021ASClockAccuracyValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "Characterizes local clock for the purpose of the best
                 master clock algorithm.  The value shall be selected as
                 follows:
                 a) clockAccuracy is set to the value that reflects the
                    combination of the LocalClock and ClockSource
                    entities if specified or known;
                 b) if the value that reflects the LocalClock and
                     ClockSource entities is not specified or unknown,
                     clockAccuracy is set to 254.
                 "
    REFERENCE   "14.2.3.2"
    ::= { ieee8021AsDefaultDS 4 }


ieee8021AsDefaultDSOffsetScaledLogVariance OBJECT-TYPE
    SYNTAX      Unsigned32(0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
             "The offsetScaledLogVariance is scaled, offset
              representation of an estimate of the PTP variance. The
              PTP variance characterizes the precision and frequency
              stability of the ClockMaster. The PTP variance is the
              square of PTPDEV (see B.1.3.2).  The value shall be
              selected as follows:
              a) offsetScaledLogVariance is set to the value that
                 reflects the combination of the LocalClock and
                 ClockSource entities; else
              b) if the value that reflects these entities is not
                 specified or not known, offsetScaledLogVariance is
                 set to 16640 (0x4100). This value corresponds to the
                 value of PTPDEV for observation interval equal to the
                 default Sync message transmission interval (i.e.,
                 observation interval of 0.125 s, see 11.5.2.3 and
                 B.1.3.2).


               A value of 65535 (0xFFFF) indicates value is too
               large to be represented or has not been computed.
               "
    REFERENCE   "14.2.3.3"
    ::= { ieee8021AsDefaultDS 5 }

ieee8021AsDefaultDSPriority1 OBJECT-TYPE
    SYNTAX      Unsigned32(0..255)
    MAX-ACCESS  read-write
    STATUS      current
```

```
        DESCRIPTION
                  "Most-significant priority declaration in the execution of
                   the best master clock algorithm. Lower values take
                   precedence. The value of priority1 shall be 255 for a
                   time-aware system that is not grandmaster-capable.
                   The value of priority1 shall be less than 255 for a
                   time-aware system that is grandmaster-capable. The value
                  0 shall be reserved for future management use, i.e., the
                   valueof priority1 shall be set to 0 only via management
                   action, and shall not be specified as a default value by
                   a user of this standard.  In the absence of a default
                   value set by a user of this standard, the default value
                   shall be set as below:
                    a) system type of network infrastructure time-aware
                       system to value 246;
                    b) portable time-aware system, 250;
                    c) other time-aware systems, 248."
        REFERENCE   "14.2.4"
        ::= { ieee8021AsDefaultDS 6 }


    ieee8021AsDefaultDSPriority2 OBJECT-TYPE
        SYNTAX      Unsigned32(0..255)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                    "Least-significant priority declaration in the execution
                    of the best master clock algorithm. Lower values take
                    precedence.  The default value is 248"
        REFERENCE   "14.2.5"
        DEFVAL { 248 }
        ::= { ieee8021AsDefaultDS 7 }


    ieee8021AsDefaultDSGmCapable OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "True (1) if master clock capable; false (2)
                    otherwise."
        REFERENCE   "14.2.6"
        ::= { ieee8021AsDefaultDS 8 }


    ieee8021AsDefaultDSCurrentUTCOffset OBJECT-TYPE
        SYNTAX      Integer32(-32768..32767)
        UNITS       "seconds"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "The value is the offset between TAI and UTC, relative to
                    the ClockMaster entity of this time-aware system. It is
                    equal to the global variable sysCurrentUtcOffset (see
                    10.3.8.18). The value is in units of seconds.
                    The initialization default value is selected as
                    follows:
                    a) the value is the value obtained from a primary
                       reference if the value is known at the at the time of
                       initialization,
                    b) else the value is the current number ofleap seconds,
                        see 8.2.3, when the time-aware system is designed."
        REFERENCE   "14.2.7"
        ::= { ieee8021AsDefaultDS 9 }


    ieee8021AsDefaultDSCurrentUTCOffsetValid OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "True (1) if ieee8021AsDefaultDSCurrentUTCOffset is known
                   to be correct; false (2) otherwise."
        REFERENCE   "14.2.8"
        ::= { ieee8021AsDefaultDS 10 }


    ieee8021AsDefaultDSLeap59 OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "A true (1) value indicates that the last minute of the
                     current UTC day, relative to the ClockMaster entity of
                     this time-aware system, will contain 59 seconds. It is
                     equal to the global variable sysLeap59 (see 10.3.8.13).

                     The initialization value is selected as follows:
                     a) Set to true (1) if the value is obtained from a
                        primary reference if known at the at the time of
                        initialization, else
                      b) The value is set to false (2)."
        REFERENCE   "14.2.9"
        ::= { ieee8021AsDefaultDS 11 }


    ieee8021AsDefaultDSLeap61 OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "A true (1) value indicates that the last minute of the
                     current UTC day, relative to the ClockMaster entity of
                     this time-aware system, will contain 59 seconds. It is
                     equal to the global variable sysLeap61 (see 10.3.8.12).

                     The initialization value is selected as follows:
```

```
                a) Set to true (1) if the value is obtained from a
                   primary reference if known at the at the time of
                   initialization, else
                 b) The value is set to false (2)."
    REFERENCE   "14.2.10"
    ::= { ieee8021AsDefaultDS 12 }


ieee8021AsDefaultDSTimeTraceable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "The value is set to true (1) if the timescale and the
                value of Ieee8021AsCurrentUtcOffset, relative to the
                ClockMaster entity of this time-aware system, are
                traceable to a primary reference standard; otherwise
                the value is set to false (2). It is equal to the global
                variable sysTimeTraceable (see 10.3.8.16).

                The initialization value is selected as follows:
                a) If the time and the value of currentUtcOffset are
                   traceable to a primary reference standard at the
                   time of initialization, the value is set to true
                   (1), else
                 b) The value is set to false (2)."
    REFERENCE   "14.2.11"
    ::= { ieee8021AsDefaultDS 13 }


ieee8021AsDefaultDSFrequencyTraceable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
              "The value is set to true (1) if the frequency determining
              the timescale of the ClockMaster Entity of this time-
              aware system is traceable to a primary reference
              standard; otherwise the value is set to false (2). It is
              equal to the global variable sysFrequencyTraceable (see
              10.3.8.17).

              The initialization value is selected as follows:
              a) If the frequency is traceable to a primary reference
                 standard at the time of initialization, the value is
                 set to true (1), else
               b) The value is set to false (2).."
    REFERENCE   "14.2.12"
    ::= { ieee8021AsDefaultDS 14 }


ieee8021AsDefaultDSTimeSource OBJECT-TYPE
    SYNTAX      IEEE8021ASTimeSourceValue
    MAX-ACCESS  read-only
```

```
        STATUS      current
        DESCRIPTION
                    "The timeSource is an information-only attribute
                     indicating the type of source of time used by a
                     ClockMaster. The value is not used in the selection of
                     the grandmaster. The values shall be as specified in
                     Table 8-4. These represent categories. For example, the
                     GPS entry would include not only the GPS system of the
                     U.S. Department of Defense but the European Galileo
                     system and other present and future satellite-based
                     timing systems.  All unused values in Table 8-4 are
                     reserved.

                     The initialization value is selected as follows:
                     a) If the timeSource (8.6.2.7 and Table 8-4), is known
                        at the time of initialization, the value is derived
                        from the table, else
                     b) The value is set to INTERNAL_OSCILLATOR (160).
    "
        REFERENCE   "14.2.14"
        ::= { ieee8021AsDefaultDS 15 }


    -- ===================================================================
    -- The Current data set represent this system's topological location
    -- relative to the known grandmaster system.
    -- This data set is consistent with respective IEEE 1588 data set.
    -- ===================================================================
    ieee8021AsCurrentDS
        OBJECT IDENTIFIER ::= { ieee8021AsMIBObjects 2 }

    ieee8021AsCurrentDSStepsRemoved OBJECT-TYPE
        SYNTAX      Integer32(-32768..32767)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                  "The number of communication paths traversed between the
                   local clock and the grandmaster clock (see Table 10.3.3).

                   For example, stepsRemoved for a slave clock on the same
                   PTP communication path as the grandmaster clock will have
                  a value of 1, indicating that a single path was traversed.
    "
        REFERENCE   "14.3.1"
        DEFVAL { 0 }
        ::= { ieee8021AsCurrentDS 1 }


    ieee8021AsCurrentDSOffsetFromMasterHs OBJECT-TYPE
        SYNTAX      Integer32
        UNITS       "2**-16 ns * 2**64"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
```

```
                "The most significant 32 bits of the offset, signed
                 96 bit number in 2**-16 ns, an implementation-specific
                 computation of the current value of the time difference
                 between a master and a slave as computed by the slave.

                 This object MUST be read at the same time as
                 ieee8021AsCurrentDSOffsetFromMasterMs, and
                 ieee8021AsCurrentDSOffsetFromMasterLs, which
                 represents middle and least significant 32 bits of
                 values, respectively, in order for the read operation
                 to succeed.
                 "
        REFERENCE   "14.3.2"
        ::= { ieee8021AsCurrentDS 2 }

    ieee8021AsCurrentDSOffsetFromMasterMs OBJECT-TYPE
        SYNTAX      Integer32
        UNITS       "2**-16 ns * 2**32"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                "The middle significant 32 bits of the offset, signed
                 96 bit number in 2**-16 ns, an implementation-specific
                 computation of the current value of the time difference
                 between a master and a slave as computed by the slave.

                 This object MUST be read at the same time as
                 ieee8021AsCurrentDSOffsetFromMasterHs, and
                 ieee8021AsCurrentDSOffsetFromMasterLs, which
                 represents most (highest) and least significant 32 bits
                 of values, respectively, in order for the read operation
                 to succeed.
                 "
        REFERENCE   "14.3.2"
        ::= { ieee8021AsCurrentDS 3 }

    ieee8021AsCurrentDSOffsetFromMasterLs OBJECT-TYPE
        SYNTAX      Integer32
        UNITS       "2**-16 ns"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                "The least significant 32 bits of the offset, signed
                 96 bit number in 2**-16 ns, an implementation-specific
                 computation of the current value of the time difference
                 between a master and a slave as computed by the slave.

                 This object MUST be read at the same time as
                 ieee8021AsCurrentDSOffsetFromMasterHs, and
                 ieee8021AsCurrentDSOffsetFromMasterMs, which
                 represents most (highest) and middle significant 32 bits
                 of values, respectively, in order for the read operation
                 to succeed.
                 "
    REFERENCE   "14.3.2"
    ::= { ieee8021AsCurrentDS 4 }

ieee8021AsCurrentDSLastGmPhaseChangeHs OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "The value (see 10.2.3.16) is the phase change that
                occurred on the most recent change in either
                grandmaster or gmTimeBaseIndicator (see 9.2.2.2).

                This object MUST be read at the same time as
               ieee8021AsCurrentDSLastGmPhaseChangeMs, and
                ieee8021AsCurrentDSLastGmPhaseChangeLs, which
                represents middle and least significant 32 bits of
                values, respectively, in order for the read operation
                to succeed.

                "
    REFERENCE   "14.3.3"
    ::= { ieee8021AsCurrentDS 5}

ieee8021AsCurrentDSLastGmPhaseChangeMs OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "The value (see 10.2.3.16) is the phase change that
                occurred on the most recent change in either
                grandmaster or gmTimeBaseIndicator (see 9.2.2.2).

                This object MUST be read at the same time as
               ieee8021AsCurrentDSLastGmPhaseChangeHs, and
                ieee8021AsCurrentDSLastGmPhaseChangeLs, which
                represents most and least significant 32 bits of
                values, respectively, in order for the read operation
                to succeed.
                "
    REFERENCE   "14.3.3"
    ::= { ieee8021AsCurrentDS 6}

ieee8021AsCurrentDSLastGmPhaseChangeLs OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "The value (see 10.2.3.16) is the phase change that
                occurred on the most recent change in either
                grandmaster or gmTimeBaseIndicator (see 9.2.2.2).

                This object MUST be read at the same time as
               ieee8021AsCurrentDSLastGmPhaseChangeMs, and
                ieee8021AsCurrentDSLastGmPhaseChangeLs, which
```

```
                    represents middle and least significant 32 bits of
                    values, respectively, in order for the read operation
                    to succeed.
                    "
        REFERENCE   "14.3.3"
        ::= { ieee8021AsCurrentDS 7}

    ieee8021AsCurrentDSLastGmFreqChangeMs OBJECT-TYPE
        SYNTAX      Integer32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "The value (see 10.2.3.17) is the frequency change that
                    occurred on the most recent change in either grandmaster
                    or gmTimeBaseIndicator (see 9.2.2.2).

                    This object MUST be read at the same time as
                    ieee8021AsCurrentDSLastGmFreqChangeLs, which
                    represents least significant 32 bits of the value
                    in order for the read operation to succeed.
                    "
        REFERENCE   "14.3.4"
        ::= { ieee8021AsCurrentDS 8 }

    ieee8021AsCurrentDSLastGmFreqChangeLs OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "The value (see 10.2.3.17) is the frequency change that
                    occurred on the most recent change in either grandmaster
                    or gmTimeBaseIndicator (see 9.2.2.2).

                    This object MUST be read at the same time as
                    ieee8021AsCurrentDSLastGmFreqChangeMs, which
                    represents most significant 32 bits of the value
                    in order for the read operation to succeed.
                    "
        REFERENCE   "14.3.4"
        ::= { ieee8021AsCurrentDS 9 }

    ieee8021AsCurrentDSGmTimebaseIndicator OBJECT-TYPE
        SYNTAX      Unsigned32(0..65535)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "This reports the grandmaster’s time base change value
                    conveyed in the Sync message.  The value is the value of
                   timeBaseIndicator of the current grandmaster (see 9.2.2.2
                   and 9.6.2.2)
                    "
        REFERENCE   "14.3.5"
        ::= { ieee8021AsCurrentDS 10 }
```

```
ieee8021AsCurrentDSGmChangeCount OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "This statistics counter tracks the number of times the
                 grandmaster has changed in a gPTP domain. This counter
                  increments when the PortAnnounceInformation state
                 machine enters the SUPERIOR_MASTER_PORT state or the
                 INFERIOR_MASTER_OR_OTHER_PORT state (see 10.3.11 and
                 Figure 10-13).
                "
    REFERENCE   "14.3.6"
    ::= { ieee8021AsCurrentDS 11 }

ieee8021AsCurrentDSTimeOfLastGmChangeEvent OBJECT-TYPE
    SYNTAX      TimeStamp
    UNITS       "0.01 seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "This timestamp denotes the system time when the most
                recent grandmaster change occurred in a gPTP domain.
                This timestamp is updated when the
                PortAnnounceInformation state machine enters the
                SUPERIOR_MASTER_PORT state or the
                INFERIOR_MASTER_OR_OTHER_PORT state (see 10.3.11 and
                Figure 10-13).
                "
    REFERENCE   "14.3.7"
    ::= { ieee8021AsCurrentDS 12 }

ieee8021AsCurrentDSTimeOfLastGmFreqChangeEvent OBJECT-TYPE
    SYNTAX      TimeStamp
    UNITS       "0.01 seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "This timestamp denotes the system time when the most
                 recent change in grandmaster phase occured, due to a
                 change of either the grandmaster or the grandmaster
                 time base. This timestamp is updated when the
                 PortAnnounceInformation state machine enters the
                 SUPERIOR_MASTER_PORT state or the
                 INFERIOR_MASTER_OR_OTHER_PORT state (see 10.3.11 and
                 Figure 10-13), and when the
                 ieee802AsCurrentDSGmTimebaseIndicator managed object
                 (see 14.3.5) changes.
                "
    REFERENCE   "14.3.8"
    ::= { ieee8021AsCurrentDS 13 }

ieee8021AsCurrentDSTimeOfLastGmPhaseChangeEvent OBJECT-TYPE
    SYNTAX      TimeStamp
```

```
        UNITS       "0.01 seconds"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "This timestamp denotes the system time when the most
                     recent change in grandmaster frequency occured, due to
                     a change of either the grandmaster or the grandmaster
                     time base. This timestamp is updated when the
                     PortAnnounceInformation state machine enters the
                     SUPERIOR_MASTER_PORT state or the
                     INFERIOR_MASTER_OR_OTHER_PORT state (see 10.3.11 and
                     Figure 10-13), and when the
                     ieee802AsCurrentDSGmTimebaseIndicator managed object
                     (see 14.3.5) changes.
                    "
        REFERENCE   "14.3.9"
        ::= { ieee8021AsCurrentDS 14 }


    -- ============================================================
    -- The Parent data set represent timing upstream (toward grandmaster)
    -- system’s parameters as measured at this system.
    -- This data set is consistent with respective IEEE 1588 data set.
    -- ============================================================
    ieee8021AsParentDS
        OBJECT IDENTIFIER ::= { ieee8021AsMIBObjects 3 }

    ieee8021AsParentDSParentClockIdentity OBJECT-TYPE
        SYNTAX      ClockIdentity
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "Clock identifier (clockIdentity) of the local clock’s
                     parent clock.  The default value is set to
                     ieee8021AsDefaultDSClockIdentity.

                     If this time-aware system is the grandmaster, the value
                     is the clockIdentity of this time-aware system.
                     If this time-aware system is not the grandmaster, the
                     value is the clockIdentity of the MasterPort (see
                     Table 10-1) of the gPTP communication path attached to
                     the single slave port of this time-aware system.
                     "
        REFERENCE   "14.4.1"
        ::= { ieee8021AsParentDS 1 }

    ieee8021AsParentDSParentPortNumber OBJECT-TYPE
        SYNTAX      Unsigned32(0..65535)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                  "Port number (portNumber) of the local clock’s parent gPTP
                   port number.

```

```
                If this time-aware system is the grandmaster, the value
                is the gPTP portNumber of this time-aware system.
                If this time-aware system is not the grandmaster, the
                value is the portNumber of the MasterPort (see
                Table 10-1) of the gPTP communication path attached to
                the single gPTP slave port of this time-aware system.
                "
    REFERENCE   "14.4.1"
    DEFVAL { 0 }
    ::= { ieee8021AsParentDS 2 }

ieee8021AsParentDSCumlativeRateRatio OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The value is an estimate of the ratio of the frequency
                 of the grandmaster to the frequency of the LocalClock
                 entity of this time-aware system.

                 Cumulative rate ratio is expressed as the fractional
                 frequency offset multiplied by 2**41, i.e., the
                 quantity (rateRatio - 1.0)(2**41), where rateRatio is
                 computed by the PortSyncSyncReceive state machine (see
                 10.2.7.1.4).
"
    REFERENCE   "14.4.2"
   ::= { ieee8021AsParentDS 3 }

ieee8021AsParentDSGrandmasterIdentity OBJECT-TYPE
    SYNTAX      ClockIdentity
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "Clock identifier (clockIdentity) of the grandmaster.
                 The default value is set to
                 ieee8021AsDefaultDSClockIdentity."
    REFERENCE   "14.4.3"
    ::= { ieee8021AsParentDS 4 }

ieee8021AsParentDSGrandmasterClockClass OBJECT-TYPE
    SYNTAX      IEEE8021ASClockClassValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "Denotes the traceability of the time or frequency of the
                 grandmaster.  The default value is set to
                 ieee8021AsDefaultDSClockClass."
    REFERENCE   "14.4.5"
    ::= { ieee8021AsParentDS 5 }

ieee8021AsParentDSGrandmasterClockAccuracy OBJECT-TYPE
    SYNTAX      IEEE8021ASClockAccuracyValue
    MAX-ACCESS  read-only
```

```
        STATUS      current
        DESCRIPTION
                    "Characterizes the grandmaster clock for the purpose of
                     the best master clock algorithm.  The default value is
                     set to ieee8021AsDefaultDSClockAccuracy."
        REFERENCE   "14.4.6"
        ::= { ieee8021AsParentDS 6 }

    ieee8021AsParentDSGrandmasterOffsetScaledLogVariance OBJECT-TYPE
        SYNTAX      Unsigned32(0..65535)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "Clock Allan variance of the local clock expressed as a
                     base-2 logarithm multiplied by a scale factor of 256.
                     Hysteresis is applied requiring the underlying computed
                     variance to move by at least 128 before a change is
                      reported. A value of 65535 (0xFFFF) indicates value
                   is too large to be represented or has not been computed.
                     The default value is set to
                     ieee8021AsDefaultDSOffsetScaledLogVariance."
        REFERENCE   "14.4.7"
        ::= { ieee8021AsParentDS 7 }

    ieee8021AsParentDSGrandmasterPriority1 OBJECT-TYPE
        SYNTAX      Unsigned32(0..255)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                    "Grandmaster’s most-significant priority declaration in
                     the execution of the best master clock algorithm.
                     Lower values take precedence.

                     The default value is set                  to
    ieee8021AsDefaultDSPriority1."
        REFERENCE   "14.4.8"
        ::= { ieee8021AsParentDS 8 }

    ieee8021AsParentDSGrandmasterPriority2 OBJECT-TYPE
        SYNTAX      Unsigned32(0..255)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                   "Grandmaster’s least-significant priority declaration in
                   the execution of the best master clock algorithm.
                    Lower values take precedence.

                   The default value is set to
    ieee8021AsDefaultDSDSPriority2."
        REFERENCE   "14.4.9"
        ::= { ieee8021AsParentDS 9 }

    -- =============================================================
    -- TimePropertiesDS represents the grandmaster’s parameters, as
```

```
-- measured at this system and are derived from 802.1AS protocol.
-- This data set is consistent with respective IEEE 1588 data set.
-- ================================================================
ieee8021AsTimePropertiesDS
    OBJECT IDENTIFIER ::= { ieee8021AsMIBObjects 4 }

ieee8021AsTimePropertiesDSCurrentUtcOffset OBJECT-TYPE
    SYNTAX      Integer32(-32768..32767)
    UNITS       "seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
              "The value is currentUtcOffset for the current grandmaster
               (see Table 14.2.7). It is equal to the value of the global
                 variable currentUtcOffset (see 10.3.8.10). The value is
in
               units of seconds. The default value is set to
               ieee8021AsDefaultDSCurrentUTCOffset."
    REFERENCE   "14.5.1"
    ::= { ieee8021AsTimePropertiesDS 1 }

ieee8021AsTimePropertiesDSCurrentUtcOffsetValid OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "True (1) if ieee8021AsTimePropertiesDSCurrentUTCOffset
               is known to be correct; false (2) otherwise.  The default
               value is set to ieee8021AsDefaultDSCurrentUTCOffsetValid.

"
    REFERENCE   "14.5.2"
    ::= { ieee8021AsTimePropertiesDS 2 }

ieee8021AsTimePropertiesDSLeap59 OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The value is leap59 for the current grandmaster (see
                 14.2.9). It is equal to the global variable leap59
                 (see 10.3.8.5).

                 A true (1) value indicates that the last minute of the
                 current UTC day, relative to the ClockMaster entity of
                 this time-aware system, will contain 59 seconds.

                The default value is set to
ieee8021AsDefaultDSLeap59."
    REFERENCE   "14.5.3"
    ::= { ieee8021AsTimePropertiesDS 3 }

ieee8021AsTimePropertiesDSLeap61 OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The value is leap61 for the current grandmaster (see
                 14.2.10). It is equal to the global variable leap59
                 (see 10.3.8.4).

                 A true (1) value indicates that the last minute of the
                 current UTC day, relative to the ClockMaster entity of
                 this time-aware system, will contain 61 seconds.The
default value is set to
                ieee8021AsDefaultDSLeap61."
    REFERENCE   "14.5.4"
    ::= { ieee8021AsTimePropertiesDS 4 }

ieee8021AsTimePropertiesDSTimeTraceable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The value is timeTraceable for the current grandmaster
                 (see 14.2.11). It is equal to the global variable
                 timeTraceable (see 10.3.8.8).

                 True (1) if the timescale and the value of
                 timePropertiesDSCurrentUTCOffset are traceable to a
                 primary reference; false (2) otherwise.  The default
                 value is set to ieee8021AsDefaultDSTimeTraceable."

    REFERENCE   "14.5.5"
    ::= { ieee8021AsTimePropertiesDS 5 }

ieee8021AsTimePropertiesDSFrequencyTraceable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The value is frequencyTraceable for the current
                 grandmaster (see 14.2.12). It is equal to the global
                 variable frequencyTraceable (see 10.3.8.9).

                 True (1) if the frequency determining the timescale is
                traceable to a primary reference; false (2) otherwise.
                 The default value is set to
                 ieee8021AsDefaultDSFrequencyTraceable."

    REFERENCE   "14.5.6"
    ::= { ieee8021AsTimePropertiesDS 6 }

ieee8021AsTimePropertiesDSTimeSource OBJECT-TYPE
    SYNTAX      IEEE8021ASTimeSourceValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
```

```
                "The value is timeSource for the current grandmaster
                 (see 14.2.14). It is equal to the global variable
                 timeTraceable (see 10.3.8.11).

                 Indicates the source of time used by the grandmaster
                 clock.
                 The default value is set to
                 ieee8021AsDefaultDSTimeSource."

    REFERENCE   "14.5.8"
    ::= { ieee8021AsTimePropertiesDS 7 }

-- ===============================================================
-- The Time-Sync parameters for each .1AS capable (gPTP) port.
-- One Table per Bridge Component or a end station.
-- One entry per gPTP port.
-- ===============================================================

ieee8021AsPortDSIfTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021AsPortDSIfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A table of gPTP port related variables in a time-aware
        Bridge or for a time-aware end station.  A value of 1 is used in
        a bridge or an end station that does not have multiple
        components.

        For a given media port of a Bridge or an end station, there ~~may~~
        can be one or more gPTP port, and depends whether a media port
        supports point to point link (e.g. IEEE 802.3 Ethernet) or point
        to multi-point (e.g. CSN, IEEE 802.3 EPON, etc) links on the
        media port.
       "
    REFERENCE
       "IEEE 802.1AS clause 14.8"
    ::= { ieee8021AsMIBObjects 5 }

ieee8021AsPortDSIfEntry   OBJECT-TYPE
    SYNTAX      Ieee8021AsPortDSIfEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A list of objects pertaining to a gPTP port of a time-aware
        bridge component or a time-aware end station.
       "
    INDEX { ieee8021AsBridgeBasePort,
            ieee8021AsPortDSAsIfIndex }
    ::= { ieee8021AsPortDSIfTable 1 }

Ieee8021AsPortDSIfEntry ::=
     SEQUENCE {
         ieee8021AsBridgeBasePort          IEEE8021BridgePortNumber,
         ieee8021AsPortDSAsIfIndex         InterfaceIndexOrZero,
         ieee8021AsPortDSClockIdentity              ClockIdentity,
         ieee8021AsPortDSPortNumber                 Unsigned32,
         ieee8021AsPortDSPort~~Role~~State                       INTEGER,
         ieee8021AsPortDSPtp~~t~~PortEnabled                 TruthValue,
         ieee8021AsPortDSIsMeasuringDelay           TruthValue,
         ieee8021AsPortDSAsCapable                  TruthValue,
         ieee8021AsPortDSNeighborPropDelayHs        Unsigned32,
         ieee8021AsPortDSNeighborPropDelayMs        Unsigned32,
         ieee8021AsPortDSNeighborPropDelayLs        Unsigned32,
         ieee8021AsPortDSNeighborPropDelayThreshHs  Unsigned32,
         ieee8021AsPortDSNeighborPropDelayThreshMs  Unsigned32,
         ieee8021AsPortDSNeighborPropDelayThreshLs  Unsigned32,
         ieee8021AsPortDSDelayAsymmetryHs           Integer32,
         ieee8021AsPortDSDelayAsymmetryMs           Unsigned32,
         ieee8021AsPortDSDelayAsymmetryLs           Unsigned32,
         ieee8021AsPortDSNeighborRateRatio          Integer32,
         ieee8021AsPortDSInitialLogAnnounceInterval Integer32,
         ieee8021AsPortDSCurrentLogAnnounceInterval Integer32,
         ieee8021AsPortDSAnnounceReceiptTimeout     Unsigned32,
         ieee8021AsPortDSInitialLogSyncInterval     Integer32,
         ieee8021AsPortDSCurrentLogSyncInterval     Integer32,
         ieee8021AsPortDSSyncReceiptTimeout         Unsigned32,
         ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalHs Unsigned32,
         ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalMs Unsigned32,
         ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalLs Unsigned32,
         ieee8021AsPortDSInitialLogPdelayReqInterval Integer32,
         ieee8021AsPortDSCurrentLogPdelayReqInterval Integer32,
         ieee8021AsPortDSAllowedLostResponses       Unsigned32,
         ieee8021AsPortDSVersionNumber              Unsigned32,
         ieee8021AsPortDSNupMs                      Unsigned32,
         ieee8021AsPortDSNupLs                      Unsigned32,
         ieee8021AsPortDSNdownMs                    Unsigned32,
         ieee8021AsPortDSNdownLs                    Unsigned32,
         ieee8021AsPortDSAcceptableMasterTableEnabled TruthValue
      }

ieee8021AsBridgeBasePort OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
               "This object identifies the bridge port number of
                the port for which this entry contains bridge management
                information.  For end stations, this port number shall
                be (1)."
    REFERENCE   "IEEE Std 802.1AS PortDS Group gPTP Port Index"
    ::= { ieee8021AsPortDSIfEntry 1 }

ieee8021AsPortDSAsIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndexOrZero
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
               "This object identifies the gPTP interface group within
```

```
                the system for which this entry contains information. It
                is the value of the instance of the IfIndex object,
                defined in the IF-MIB, for the gPTP interface group
                corresponding to this port, or the value 0 if the port
                has not been bound to an underlying frame source and
                sink.

                For a given media port of a Bridge or an end station,
              there ~~may~~can be one or more gPTP port, and depends whether
                a media port supports point to point link (e.g. IEEE
                802.3 Ethernet) or point to multi-point (e.g. CSN, IEEE
                802.3 EPON, etc) links on the media port."
    REFERENCE   "IEEE Std 802.1AS PortDS Group gPTP Port Index"
    ::= { ieee8021AsPortDSIfEntry 2 }

ieee8021AsPortDSClockIdentity OBJECT-TYPE
    SYNTAX      ClockIdentity
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The clockIdentity is an 8 octet array formed by
                mapping an IEEE EUI-48 assigned to the time-aware system
                 to IEEE EUI-64 format (i.e., to an array of 8 octets).
                The EUI-48 shall be an Ethernet MAC address owned by the
                organization creating the instance of a clockIdentity
                under the terms of this subclause. The organization
                owning the MAC address shall ensure that the
                MAC address is used in generating only a single instance
                of a clockIdentity, for example by requiring that the
                MAC address be a MAC address embedded in the device
                identified by the clockIdentity. The mapping rules for
                constructing the EUI-64 from the EUI-48 shall be those
                specified by the IEEE [B2]. The 8 octets of the created
                IEEE EUI-64 shall be assigned in order to the 8 octet
                array clockIdentity with most significant octet of the
                IEEE EUI-64 assigned to the clockIdentity octet array
                member with index 0.(see 8.5.2.2)."
    REFERENCE   "14.8.2"
    ::= { ieee8021AsPortDSIfEntry 3 }


ieee8021AsPortDSPortNumber OBJECT-TYPE
    SYNTAX      Unsigned32(0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The portNumber value for a port on a time-aware end
                 station (i.e., a time-aware system supporting a single
                 gPTP port) shall be 1. The portNumber values for the
                 gPTP ports on a time-aware ~~bridge~~Bridge supporting N
                ports shall be 1, 2, …, N, respectively (see 8.5.2.3)  ."
    REFERENCE   "14.8.2"
    ::= { ieee8021AsPortDSIfEntry 4 }
```

```
ieee8021AsPortDSPort~~Role~~State OBJECT-TYPE
    SYNTAX      INTEGER {
            disabledPort(3),
            masterPort(6),
            passivePort(7),
            slavePort(9)
           }
    MAX-ACCESS  read-write~~only~~
    STATUS      current
    DESCRIPTION
              "The value is the value of the port ~~role~~state of this port
                (see Table 10-1), and is taken from the enumeration in
                Table 14-7. All other values reserved.  The enumeration
                values are consistent with IEEE Std 1588TM-2008,
                Table 8. The default value is 3 (DisabledPort)."
    REFERENCE   "14.8.3"
    DEFVAL { 3 }
    ::= { ieee8021AsPortDSIfEntry 5 }


ieee8021AsPortDSPtp~~t~~PortEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
                "802.1AS function enable for a given port.  True (1) if
                 the time-synchronization and best master selection
                 functions of the port are enabled;
                 False (2) otherwise (see 10.2.4.13).

                 The contents of this table SHALL be maintained across a
                 restart         of the system.
       "
    REFERENCE   "14.8.4"
    DEFVAL { 1 }
    ::= { ieee8021AsPortDSIfEntry 6 }


ieee8021AsPortDSIsMeasuringDelay OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
              "True (1) if the port is measuring link propagation delay;
               The value is equal to the value of the Boolean
               isMeasuringPdDelay (see 11.2.12.6 and 16.4.3.2)
               False (2) otherwise."
    REFERENCE   "14.8.6"
    DEFVAL { 2 }
    ::= { ieee8021AsPortDSIfEntry 7 }


ieee8021AsPortDSAsCapable OBJECT-TYPE
    SYNTAX      TruthValue
```

```
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "True (1) if and only if it is determined that this
                     time-aware system and the time-aware system at the
                     other ends of the link attached to this port can
                     interoperate with each other via the IEEE 802.1AS
                     protocol; False (2) otherwise."
        REFERENCE   "14.8.7"
        ::= { ieee8021AsPortDSIfEntry 8 }

    ieee8021AsPortDSNeighborPropDelayHs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "2**-16 ns * 2**64"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "The most (highest) significant 32 bits, unsigned
                    96 bit number in 2**-16 ns, the value is equal to the
                    value of the per-port global variable
                    neighborPropDelay (see 10.2.4.7).  It is an
                    estimate of the current one-way propagation time on the
                    link attached to this port, measured as specified for
                   the respective medium (see 11.2.15, 12.5.2.6.2, and 16.4).
                    The value is zero for ports attached to IEEE 802.3 EPON
                    links and for the master port of an IEEE 802.11 link,
                    because one-way propagation delay is not measured on
                    the latter and not directly measured on the former. It
                    is recommended that the data type be scaled in ns. The
                    initialization value is zero.

                    This object MUST be read at the same time as
                    ieee8021AsPortDSNeighborPropDelayMs, and
                    ieee8021AsPortDSNeighborPropDelayLs, which
                    represents middle and least significant 32 bits
                    of values, respectively, in order for the read operation
                    to succeed.
                    "
        REFERENCE   "14.8.8"
        DEFVAL { 0 }
        ::= { ieee8021AsPortDSIfEntry 9 }

    ieee8021AsPortDSNeighborPropDelayMs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "2**-16 ns * 2**32"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "The second most (middle) significant 32 bits, unsigned
                     96 bit number in 2**-16 ns, the value is equal to the
                     value of the per-port global variable
                     neighborPropDelay (see 10.2.4.7).  It is an
                     estimate of the current one-way propagation time on the
                     link attached to this port, measured as specified for
```

```
                  the respective medium (see 11.2.15, 12.5.2.6.2, and
16.4).
                  The value is zero for ports attached to IEEE 802.3 EPON
                  links and for the master port of an IEEE 802.11 link,
                  because one-way propagation delay is not measured on
                  the latter and not directly measured on the former. It
                  is recommended that the data type be scaled in ns. The
                  initialization value is zero.

                 This object MUST be read at the same time as
                 ieee8021AsPortDSNeighborPropDelayHs, and
                 ieee8021AsPortDSNeighborPropDelayLs, which
                 represents most (highest) and least significant 32 bits
                 of values, respectively, in order for the read operation
                 to succeed.
                 "
    REFERENCE   "14.8.8"
    DEFVAL { 0 }
    ::= { ieee8021AsPortDSIfEntry 10 }

ieee8021AsPortDSNeighborPropDelayLs OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "2**-16 ns"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                 "The least significant 32 bits, unsigned
                  96 bit number in 2**-16 ns, the value is equal to the
                  value of the per-port global variable
                  neighborPropDelay (see 10.2.4.7).  It is an
                  estimate of the current one-way propagation time on the
                  link attached to this port, measured as specified for
                  the respective medium (see 11.2.15, 12.5.2.6.2, and
16.4).
                  The value is zero for ports attached to IEEE 802.3 EPON
                  links and for the master port of an IEEE 802.11 link,
                  because one-way propagation delay is not measured on
                  the latter and not directly measured on the former. It
                  is recommended that the data type be scaled in ns. The
                  initialization value is zero.

                 This object MUST be read at the same time as
                 ieee8021AsPortDSNeighborPropDelayHs, and
                 ieee8021AsPortDSNeighborPropDelayMs, which
                 represents most (highest) and middle significant 32 bits
                 of values, respectively, in order for the read operation
                 to succeed.
                 "
    REFERENCE   "14.8.8"
    DEFVAL { 0 }
    ::= { ieee8021AsPortDSIfEntry 11 }


ieee8021AsPortDSNeighborPropDelayThreshHs OBJECT-TYPE
```

```
    SYNTAX      Unsigned32
    UNITS       "2**-16 ns * 2 ** 64"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
               "The most (highest) significant 32 bits, unsigned
                96 bit number in 2**-16 ns, the value is equal to the
                value of the per-port global variable
                neighborPropDelayThresh (see 11.2.12.6). It
                is the propagation time threshold, above which a port
                is not considered capable of participating in the
                802.1AS protocol

                This object MUST be read or written at the same time as
                ieee8021AsPortDSNeighborPropDelayThreshMs, and
                ieee8021AsPortDSNeighborPropDelayThreshLs, which
                represents middle and least significant 32 bits
                of values, respectively, in order for the read or write
                operation to succeed.

                The contents of this variable SHALL be maintained
                across a restart of the system.
       "
    REFERENCE   "14.8.9"
    ::= { ieee8021AsPortDSIfEntry 12 }

ieee8021AsPortDSNeighborPropDelayThreshMs OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "2**-16 ns * 2 ** 32"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
               "The middle significant 32 bits, unsigned
                96 bit number in 2**-16 ns, the value is equal to the
                value of the per-port global variable
                neighborPropDelayThresh (see 11.2.12.6). It
                is the propagation time threshold, above which a port
                is not considered capable of participating in the
                802.1AS protocol

                This object MUST be read or written at the same time as
                ieee8021AsPortDSNeighborPropDelayThreshHs, and
                ieee8021AsPortDSNeighborPropDelayThreshLs, which
                represents most (highest) and least significant 32 bits
                of values, respectively, in order for the read or write
                operation to succeed.

                The contents of this variable SHALL be maintained
                across a restart of the system.
     "
     REFERENCE   "14.8.9"
    ::= { ieee8021AsPortDSIfEntry 13 }

ieee8021AsPortDSNeighborPropDelayThreshLs OBJECT-TYPE
```

```
        SYNTAX      Unsigned32
        UNITS       "2**-16 ns"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                   "The least significant 32 bits, unsigned
                    96 bit number in 2**-16 ns, the value is equal to the
                    value of the per-port global variable
                    neighborPropDelayThresh (see 11.2.12.6). It
                    is the propagation time threshold, above which a port
                    is not considered capable of participating in the
                    802.1AS protocol

                    This object MUST be read at the same time as
                    ieee8021AsPortDSNeighborPropDelayThreshHs, and
                    ieee8021AsPortDSNeighborPropDelayThreshMs, which
                    represents most (highest) and middle significant 32 bits
                    of values, respectively, in order for the read or write
                    operation to succeed.

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
          "
         REFERENCE   "14.8.9"
        ::= { ieee8021AsPortDSIfEntry 14 }

    ieee8021AsPortDSDelayAsymmetryHs OBJECT-TYPE
        SYNTAX      Integer32
        UNITS       "2**-16 ns * 2**64"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                    "The most (highest) significant 32 bits, signed
                     96 bit number in 2**-16 ns.
                     The value is the asymmetry in the propagation delay on
                     the link attached to this port relative to the
                     grandmaster time base, as defined in 8.3. If
                     propagation delay asymmetry is not modeled, then
                       delayAsymmetry is 0.

                    This object MUST be read or written at the same time as
                    ieee8021AsPortDSDelayAsymmetryMs, and
                    ieee8021AsPortDSDelayAsymmetryLs, which
                    represents middle and least significant 32 bits
                    of values, respectively, in order for the read or write
                    operation to succeed.

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
           "
        REFERENCE   "14.8.10 and 8.3"
        ::= { ieee8021AsPortDSIfEntry 15 }

    ieee8021AsPortDSDelayAsymmetryMs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "2**-16 ns * 2**32"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                   "The middle significant 32 bits, signed
                     96 bit number in 2**-16 ns.
                     The value is the asymmetry in the propagation delay on
                     the link attached to this port relative to the
                     grandmaster time base, as defined in 8.3. If
                     propagation delay asymmetry is not modeled, then
                       delayAsymmetry is 0.

                    This object MUST be read or written at the same time as
                    ieee8021AsPortDSDelayAsymmetryHs, and
                    ieee8021AsPortDSDelayAsymmetryLs, which
                    represents middle and least significant 32 bits
                    of values, respectively, in order for the read or write
                    operation to succeed.

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
           "
        REFERENCE   "14.8.10 and 8.3"
        ::= { ieee8021AsPortDSIfEntry 16 }

    ieee8021AsPortDSDelayAsymmetryLs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "2**-16 ns"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                    "The least significant 32 bits, signed
                     96 bit number in 2**-16 ns.
                     The value is the asymmetry in the propagation delay on
                     the link attached to this port relative to the
                     grandmaster time base, as defined in 8.3. If
                     propagation delay asymmetry is not modeled, then
                       delayAsymmetry is 0.

                    This object MUST be read or written at the same time as
                    ieee8021AsPortDSDelayAsymmetryHs, and
                    ieee8021AsPortDSDelayAsymmetryLs, which
                    represents most (highest) and least significant 32 bits
                    of values, respectively, in order for the read or write
                    operation to succeed.

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
           "
        REFERENCE   "14.8.10 and 8.3"
        ::= { ieee8021AsPortDSIfEntry 17 }
```

```
ieee8021AsPortDSNeighborRateRatio OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
              "The value is an estimate of the ratio of the frequency of
                the LocalClock entity of the time-aware system at the
                other end of the link attached to this port, to the
                frequency of the LocalClock entity of this time-aware
                system (see 10.2.4.7). Neighbor rate ratio is expressed
                as the fractional frequency offset multiplied by 2**41,
                i.e., the quantity (neighborRateRatio - 1.0)(2**41)."
    REFERENCE   "14.8.11"
    ::= { ieee8021AsPortDSIfEntry 18 }


ieee8021AsPortDSInitialLogAnnounceInterval OBJECT-TYPE
    SYNTAX      Integer32(-128..127)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
                "The value is the logarithm to the base 2 of the of the
                 announce  interval used when
                (a) the port is initialized, or
                (b) a message interval request TLV is received with
                    announceInterval field set to 126 (see 10.7.2.2 and
                    and the AnnounceIntervalSetting state machine
                    10.3.16)
                 The default value is 0.

                The contents of this variable SHALL be maintained
                across a restart of the system.
       "
    REFERENCE   "14.8.12"
    DEFVAL { 0 }
    ::= { ieee8021AsPortDSIfEntry 19 }

ieee8021AsPortDSCurrentLogAnnounceInterval OBJECT-TYPE
    SYNTAX      Integer32(-128..127)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The value is the logarithm to the base 2 of the of the
                 current announce transmission interval.

                 The currentLogAnnounceInterval specifies the current
                 value of the announce interval.
                 Every port supports the value 127; the port does not
                 send Announce messages when currentLogAnnounceInterval
                 has this value (see 10.3.16). A port may support other
                 values, except for the reserved values -128 through -
                 125, inclusive, and 124 through 126, inclusive. A port
                 ignores requests (see 10.3.16) for unsupported values.
       "
```

```
        REFERENCE   "14.8.13"
        ::= { ieee8021AsPortDSIfEntry 20 }


    ieee8021AsPortDSAnnounceReceiptTimeout OBJECT-TYPE
        SYNTAX      Unsigned32(0..255)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                   "The value is the number of Announce message
                    transmission intervals that a slave port waits without
                    receiving an Announce message, before assuming that
                    the master is no longer transmitting Announce
                    messages, and that the BMC algorithm needs to be run,
                    if appropriate.  The condition of the slave port not
                    receiving an Announce message for
                    announceReceiptTimeout announce intervals is referred
                    to as ’announce receipt timeout’.

                     The default value is 3 (see 10.7.3.2).

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
           "
        REFERENCE   "14.8.16"
        DEFVAL { 2 }
        ::= { ieee8021AsPortDSIfEntry 21 }


    ieee8021AsPortDSInitialLogSyncInterval OBJECT-TYPE
        SYNTAX      Integer32(-128..127)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                    "The value is the logarithm to the base 2 of the
                     sync interval used when,
                     (a) the port is initialized, or
                     (b) a message interval request TLV is received with the
                        timeSyncInterval field set to 126 (see 10.7.2.3,
                        11.5.2.3, 12.7.2, 13.9.2, and the
                        LinkDelaySyncIntervalSetting state machine, see
                     11.2.19. The initialization value is -3 (see 10.7.2.3).

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
           "
        REFERENCE   "14.8.17"
        DEFVAL { -3 }
        ::= { ieee8021AsPortDSIfEntry 22 }

    ieee8021AsPortDSCurrentLogSyncInterval OBJECT-TYPE
        SYNTAX      Integer32(-128..127)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "The value is the logarithm to the base 2 of the
                     current time-synchronization transmission interval, see
                     10.6.2.3.

                    The initialization value is -3.
           "
        REFERENCE   "14.8.18"
        DEFVAL { -3 }
        ::= { ieee8021AsPortDSIfEntry 23 }


    ieee8021AsPortDSSyncReceiptTimeout OBJECT-TYPE
        SYNTAX      Unsigned32(0..255)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                    "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 BMC algorithm needs to be
                     run, if appropriate.  The condition of the slave port
                     not receiving synchronization information for
                     syncReceiptTimeout sync intervals is referred to as
                     'sync receipt timeout'.

                    The initialization value is 3 (see 10.7.3.1).

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
           "
        REFERENCE   "14.8.21"
        DEFVAL { 3 }
        ::= { ieee8021AsPortDSIfEntry 24 }


    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalHs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "2**-16 ns"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "The most (highest) significant 32 bits, of unsigned
                    96 bit number in 2**-16 ns.
                    The value is equal to the value of the per port global
                    variable syncReceiptTimeoutTimeInterval (see 10.2.4.3).
                    It is the time interval after which sync receipt
                    timeout occurs if time-synchronization information has
                    not been received during the interval.

                    This object MUST be read  at the same time as
                    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalMs, and
                    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalLs, which
                    represents middle and least significant 32 bits
                    of values, respectively, in order for the read
                    operation to succeed.

                    Default value is calculated per 10.2.4.3, or
                    '0000 0000 0000 165A 0BC0 0000'h.

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
                   "
        REFERENCE   "14.8.22"
        DEFVAL { ’00000000’h }
        ::= { ieee8021AsPortDSIfEntry 25 }

    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalMs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "2**-16 ns * 2**32"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "The middle significant 32 bits, unsigned
                    96 bit number in 2**-16 ns.
                    The value is equal to the value of the per port global
                    variable syncReceiptTimeoutTimeInterval (see 10.2.4.3).
                    It is the time interval after which sync receipt
                    timeout occurs if time-synchronization information has
                    not been received during the interval.

                    This object MUST be read  at the same time as
                    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalHs, and
                    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalLs, which
                    represents most (highest) and least significant 32 bits
                    of values, respectively, in order for the read
                    operation to succeed.

                    Default value is calculated per 10.2.4.3, or
                    '0000 0000 0000 165A 0BC0 0000'h.

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
                   "
        REFERENCE   "14.8.22"
        DEFVAL { ’0000165A’h }
        ::= { ieee8021AsPortDSIfEntry 26 }

    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalLs OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "2**-16 ns"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "The least significant 32 bits, unsigned
                    96 bit number in 2**-16 ns.
                    The value is equal to the value of the per port global
```

```
                    variable syncReceiptTimeoutTimeInterval (see 10.2.4.3).
                    It is the time interval after which sync receipt
                    timeout occurs if time-synchronization information has
                    not been received during the interval.

                    This object MUST be read  at the same time as
                    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalHs, and
                    ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalMs, which
                    represents most (highest) and middle significant 32 bits
                    of values, respectively, in order for the read
                    operation to succeed.

                    Default value is calculated per 10.2.4.3, or
                    '0000 0000 0000 165A 0BC0 0000'h.

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
                   "
        REFERENCE   "14.8.22"
        DEFVAL { ’0BC00000’h }
        ::= { ieee8021AsPortDSIfEntry 27 }


    ieee8021AsPortDSInitialLogPdelayReqInterval OBJECT-TYPE
        SYNTAX      Integer32(-128..127)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                   "For full-duplex, IEEE 802.3 media and CSN media that use
                    the peer delay mechanism to measure path delay (see
                    E.4.3.1), the value is the logarithm to the base 2 of the
                    Pdelay_Req message transmission interval used when,
                    (a) the port is initialized, or
                    (b) a message interval request TLV is received with the
                        linkDelayInterval field set to 126 (see 11.5.2.2 and
                        the LinkDelaySyncIntervalSetting state machine, see
                        11.2.19).

                    For these media, the initialization value is 0.
                    For all other media, the value is 127.

                    The contents of this variable SHALL be maintained
                    across a restart of the system.
           "
        REFERENCE   "14.8.23"
        ::= { ieee8021AsPortDSIfEntry 28 }

    ieee8021AsPortDSCurrentLogPdelayReqInterval OBJECT-TYPE
        SYNTAX      Integer32(-128..127)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "For full-duplex, IEEE 802.3 media and CSN media that use
                    the peer delay mechanism to measure path delay (see
                E.4.3.1), the value is the logarithm to the base 2 of the
                current Pdelay_Req message transmission interval,
                see 11.5.2.2.

                For all other media, the value is 127.

                The contents of this variable SHALL be maintained
                across a restart of the system.
           "
        REFERENCE   "14.8.24"
        ::= { ieee8021AsPortDSIfEntry 29 }


    ieee8021AsPortDSAllowedLostResponses OBJECT-TYPE
        SYNTAX      Unsigned32(0..65535)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
                "The value is equal to the value of the per-port global
                 variable allowedLostResponses (see 11.2.12.4). It is
                 the number of Pdelay_Req messages for which a valid
                 response is not received, above which a port is
                 considered to not be exchanging peer delay messages
                 with its neighbor.

           "
        REFERENCE   "14.8.27 and 11.5.3"
        DEFVAL { 3 }
        ::= { ieee8021AsPortDSIfEntry 30 }


    ieee8021AsPortDSVersionNumber OBJECT-TYPE
        SYNTAX      Unsigned32(0..63)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                "Indicates the PTP version in use on the port.
                 The version number for this standard is set to
                 the value 2 (see 10.6.2.2.4).

                The contents of this variable SHALL be maintained
                across a restart of the system.
           "
        REFERENCE   "14.8.31"
        DEFVAL { 2 }
        ::= { ieee8021AsPortDSIfEntry 31 }


    ieee8021AsPortDSNupMs OBJECT-TYPE
        SYNTAX      Unsigned32
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
               "The most significant 32 bits, of unsigned
```

```
                  64 bit fixed point number between 0 and less than 2.
                  For an OLT port of an IEEE 802.3 EPON link, the value is
                  the effective index of refraction for the EPON upstream
                  wavelength light of the optical path (see 13.1.4 and
                  13.8.1.2). The default value is
                  1.46770 for 1 Gb/s upstream links, and
                  1.46773 for 10 Gb/s upstream links.
                  For all other ports, the value is 0.

                  This object MUST be read or written at the same time as
                  ieee8021AsPortDSNupLs, which represents least
                  significant 32 bits of the value in order for the read
                  or write operation to succeed.

                  The contents of this variable SHALL be maintained
                  across a restart of the system.
                  "
    REFERENCE   "14.8.32"
    ::= { ieee8021AsPortDSIfEntry 32 }

ieee8021AsPortDSNupLs OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
                  "The least significant 32 bits, of unsigned
                  64 bit fixed point number between 0 and less than 2.
                  For an OLT port of an IEEE 802.3 EPON link, the value is
                  the effective index of refraction for the EPON upstream
                  wavelength light of the optical path (see 13.1.4 and
                  13.8.1.2). The default value is
                  1.46770 for 1 Gb/s upstream links, and
                  1.46773 for 10 Gb/s upstream links.

                  For all other ports, the value is 0.

                  This object MUST be read or written at the same time as
                  ieee8021AsPortDSNupMs, which represents the most
                  significant 32 bits of the value in order for the read
                  or write operation to succeed.

                  The contents of this variable SHALL be maintained
                  across a restart of the system.
                  "
    REFERENCE   "14.8.32"
    ::= { ieee8021AsPortDSIfEntry 33 }

ieee8021AsPortDSNdownMs OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
                  "The least significant 32 bits, of unsigned
                  64 bit fixed point number between 0 and less than 2.
```

```
                 For an OLT port of an IEEE 802.3 EPON link, the value is
                the effective index of refraction for the EPON
                downstream wavelength light of the optical path (see
                13.1.4 and 13.8.1.2.2). The default value is
                1.46805 for 1 Gb/s downstream links, and
                1.46851 for 10 Gb/s downstream links.

                For all other ports, the value is 0.

                This object MUST be read or written at the same time as
                ieee8021AsPortDSNdownLs, which represents the least
                significant 32 bits of the value in order for the read
                or write operation to succeed.

                The contents of this variable SHALL be maintained
                across a restart of the system.
                "
    REFERENCE   "14.8.33"
    ::= { ieee8021AsPortDSIfEntry 34 }

ieee8021AsPortDSNdownLs OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
               "The least significant 32 bits, of unsigned
                64 bit fixed point number between 0 and less than 2.
                 For an OLT port of an IEEE 802.3 EPON link, the value is
                the effective index of refraction for the EPON
                downstream wavelength light of the optical path (see
                13.1.4 and 13.8.1.2.1). The default value is
                1.46805 for 1 Gb/s downstream links, and
                1.46851 for 10 Gb/s downstream links.

                For all other ports, the value is 0.

                This object MUST be read or written at the same time as
                ieee8021AsPortDSNdownMs, which represents the most
                significant 32 bits of the value in order for the read
                or write operation to succeed.

                The contents of this variable SHALL be maintained
                across a restart of the system.
                "
    REFERENCE   "14.8.33"
    ::= { ieee8021AsPortDSIfEntry 35 }

ieee8021AsPortDSAcceptableMasterTableEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
               "True (1) if acceptableMasterTableEnabled (see 13.1.3.1)
                  and 13.1.3.5) is true and an ONU port attached to an
```

```
                    IEEE 802.3 EPON link in a time-aware system.


                   False (2), otherwise.
                 The default value               is FALSE.

                   The contents of this variable SHALL be maintained
                   across a restart of the system.
         "
      REFERENCE   "14.8.34"
      DEFVAL { false }
      ::= { ieee8021AsPortDSIfEntry 36 }

  -- ===============================================================
  -- The Statistics for each .1AS capable port.
  -- One Table per Bridge Component or a end station.
  -- One entry per port.
  -- ===============================================================

  ieee8021AsPortStatIfTable   OBJECT-TYPE
      SYNTAX      SEQUENCE OF Ieee8021AsPortStatIfEntry
      MAX-ACCESS  not-accessible
      STATUS      current
      DESCRIPTION
         "A table of time-aware port related counters in a gPTP domain.
          A value of 1 is used in a bridge or an end station that does not
          have multiple components.
         "
      REFERENCE
         "IEEE 802.1AS clause 14.10"
      ::= { ieee8021AsMIBObjects 6 }

  ieee8021AsPortStatIfEntry   OBJECT-TYPE
      SYNTAX      Ieee8021AsPortStatIfEntry
      MAX-ACCESS  not-accessible
      STATUS      current
      DESCRIPTION
         "A list of statistics pertaining to a port of a gPTP domain.
          This statistics table uses ieee8021AsPortDSAsIfIndex, and
          corresponds to ieee8021ASPortDSTable entries.
         "
      INDEX { ieee8021AsBridgeBasePort,
              ieee8021AsPortDSAsIfIndex }
      ::= { ieee8021AsPortStatIfTable 1 }

  Ieee8021AsPortStatIfEntry ::=
       SEQUENCE {
           ieee8021AsPortStatRxSyncCount                Counter32,
           ieee8021AsPortStatRxFollowUpCount            Counter32,
           ieee8021AsPortStatRxPdelayRequest            Counter32,
           ieee8021AsPortStatRxPdelayResponse           Counter32,
           ieee8021AsPortStatRxPdelayResponseFollowUp  Counter32,
           ieee8021AsPortStatRxAnnounce                 Counter32,
           ieee8021AsPortStatRxPTPPacketDiscard        Counter32,
           ieee8021AsPortStatRxSyncReceiptTimeouts     Counter32,
           ieee8021AsPortStatAnnounceReceiptTimeouts   Counter32,
           ieee8021AsPortStatPdelayAllowedLostResponsesExceeded
                                                       Counter32,
           ieee8021AsPortStatTxSyncCount               Counter32,
           ieee8021AsPortStatTxFollowUpCount           Counter32,
           ieee8021AsPortStatTxPdelayRequest           Counter32,
           ieee8021AsPortStatTxPdelayResponse          Counter32,
           ieee8021AsPortStatTxPdelayResponseFollowUp  Counter32,
           ieee8021AsPortStatTxAnnounce                Counter32
        }

  ieee8021AsPortStatRxSyncCount OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
                  "A counter that increments every time synchronization
                   information is received, denoted by a transition to
                   TRUE from FALSE of the rcvdSync variable of the
                   MDSyncReceiveSM state machine (see 11.2.13.1.2 and
                   Figure 11-6), when in the DISCARD or WAITING_FOR_SYNC
                   states; or rcvdIndication transitions to TRUE (see
                   Figure 12-6).
                  "
      REFERENCE   "14.10.2"
      ::= { ieee8021AsPortStatIfEntry 1 }

  ieee8021AsPortStatRxFollowUpCount OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
                 "A counter that increments every time a Follow_Up message
                  is received, denoted by a transition to TRUE from FALSE
                  of the rcvdFollowUp variable of the MDSyncReceiveSM
                   state machine (see 11.2.13.1.3 and Figure 11-6) when in
                  the WAITING_FOR_FOLLOW_UP state.
                  "
      REFERENCE   "14.10.4"
      ::= { ieee8021AsPortStatIfEntry 2 }

  ieee8021AsPortStatRxPdelayRequest OBJECT-TYPE
      SYNTAX      Counter32
      MAX-ACCESS  read-only
      STATUS      current
      DESCRIPTION
                "A counter that increments every time a Pdelay_Req message
                 is received, denoted by a transition to TRUE from FALSE
                 of the rcvdPdelayReq variable of the MDPdelayResp state
                 machine (see 11.2.18.2.1 and Figure 11-9) when in the
                  WAITING_FOR_PDELAY_REQ or INITIAL_WAITING_FOR_PDELAY_REQ
                 states.
                  "
```

```
        REFERENCE   "14.10.5"
        ::= { ieee8021AsPortStatIfEntry 3 }

    ieee8021AsPortStatRxPdelayResponse OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "A counter that increments every time a Pdelay_Resp
                     message is received, denoted by a transition to TRUE
                     from FALSE of the rcvdPdelayResp variable of the
                     MDPdelayReq state machine (see 11.2.17.2.2 and Figure
                     11-8) when in the WAITING_FOR_PDELAY_RESP state.
                    "
        REFERENCE   "14.10.6"
        ::= { ieee8021AsPortStatIfEntry 4 }

    ieee8021AsPortStatRxPdelayResponseFollowUp OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "A counter that increments every time a
                    Pdelay_Resp_Follow_Up message is received, denoted by a
                     transition to TRUE from FALSE of the
                     rcvdPdelayRespFollowUp variable of the MDPdelayReq
                    state machine (see 11.2.17.2.4 and Figure 11-8) when in
                     the WAITING_FOR_PDELAY_RESP_FOLLOW_UP state.
                    "
        REFERENCE   "14.10.7"
        ::= { ieee8021AsPortStatIfEntry 5 }

    ieee8021AsPortStatRxAnnounce OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                   "A counter that increments every time an Announce message
                    is received, denoted by a transition to TRUE from FALSE
                    of the rcvdAnnounce variable of the PortAnnounceReceive
                    state machine (see 10.3.10 and Figure 10-12) when in the
                    DISCARD or RECEIVE states.
                    "
        REFERENCE   "14.10.8"
        ::= { ieee8021AsPortStatIfEntry 6 }

    ieee8021AsPortStatRxPTPPacketDiscard OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
                    "A counter that increments every time a PTP message is
                     discarded, caused by the occurrence of any of the
                     following conditions:
```

```
                    a) A received Announce message is not qualified,
                       denoted by the function qualifyAnnounce (see
                       10.3.10.2.1 and 13.1.3.4) of the PortAnnounceReceive
                       state machine (see 10.3.10 and Figure 10-12)
                       returning FALSE;
                    b) A Follow_Up message corresponding to a received Sync
                       message is not received, denoted by a transition of
                       the condition (currentTime greater or equal to
                       followUpReceiptTimeoutTime) to TRUE from FALSE when
                       in the WAITING_FOR_FOLLOW_UP state of the
                       MDSyncReceiveSM state machine (see 11.2.13 and
                       Figure 11-6);
                     c) A Pdelay_Resp message corresponding to a
                        transmitted Pdelay_Req message is not received,
                        denoted by a transition from the
                        WAITING_FOR_PDELAY_RESP state to the RESET state of
                        the MDPdelayReq state machine (see 11.2.15 and
                        Figure 11-8);
                     d) A Pdelay_Resp_Follow_Up message corresponding to a
                        transmitted Pdelay_Req message is not received,
                        denoted by a transition from the
                        WAITING_FOR_PDELAY_RESP_FOLLOW_UP state to the
                        RESET state of the MDPdelayReq state machine (see
                        11.2.15 and Figure 11-8).
                   "
    REFERENCE   "14.10.9"
    ::= { ieee8021AsPortStatIfEntry 7 }

ieee8021AsPortStatRxSyncReceiptTimeouts OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "A counter that increments every time sync receipt timeout
                occurs, denoted by entering the AGED state of the
                PortAnnounceInformation state machine (see 10.3.11 and
                Figure 10-13), with the condition (currentTime greater or
                equual to announceReceiptTimeoutTime) TRUE
                "
    REFERENCE   "14.10.10"
    ::= { ieee8021AsPortStatIfEntry 8 }

ieee8021AsPortStatAnnounceReceiptTimeouts OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "A counter that increments every time announce receipt
                 timeout occurs, denoted by entering the AGED state of
                  the PortAnnounceInformation state machine (see 10.3.11
                 and Figure 10-13), with the condition ((currentTime
                 greater than or equal to syncReceiptTimeoutTime) AND
                 gmPresent)) TRUE.
                "
    REFERENCE   "14.10.11"
    ::= { ieee8021AsPortStatIfEntry 9 }

ieee8021AsPortStatPdelayAllowedLostResponsesExceeded OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "A counter that increments everytime the value of the
                 variable lostResponses (see, 11.2.17.2.10) exceeds the
                 value of the variable allowedLostResponses (see
                11.2.12.4), in the RESET state of the MDPdelayReq state
                 machine (see 11.2.15 and Figure 11-8)
                "
    REFERENCE   "14.10.12"
    ::= { ieee8021AsPortStatIfEntry 10 }

ieee8021AsPortStatTxSyncCount OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "A counter that increments every time synchronization
                 information is transmitted, denoted by a transition to
                 TRUE from FALSE of the rcvdMDSync variable of the
                 MDSyncSendSM state machine (see 11.2.14.1.1 and
                 Figure 11-7), when in the INITIALIZING or
                 SEND_FOLLOW_UP states; or the
                 INITIATE_REQUEST_WAIT_CONFIRM state is entered
                 in ~~Figure 12-3~~Figure 12-4.
                "
    REFERENCE   "14.10.13"
    ::= { ieee8021AsPortStatIfEntry 11 }

ieee8021AsPortStatTxFollowUpCount OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "A counter that increments every time a Follow_Up message
                is transmitted, denoted by a transition to TRUE from
                FALSE of the rcvdMDTimestampReceive variable of the
                MDSyncSendSM state machine (see 11.2.14.1.3 and Figure
               11-7), when in the SEND_SYNC state increments every time
                a Follow_Up packet is transmitted.
                "
    REFERENCE   "14.10.15"
    ::= { ieee8021AsPortStatIfEntry 12 }

ieee8021AsPortStatTxPdelayRequest OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "A counter that increments every time a Pdelay_Req message
                is transmitted, denoted by entering the
                INITIAL_SEND_PDELAY_REQ or SEND_PDELAY_REQ states of the
                MDPdelayReq state machine (see 11.2.15 and Figure 11-8).
                "
    REFERENCE   "14.10.16"
    ::= { ieee8021AsPortStatIfEntry 13 }

ieee8021AsPortStatTxPdelayResponse OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "A counter that increments every time a Pdelay_Resp
                 message is transmitted, denoted by a transition to TRUE
                 from FALSE of the rcvdPdelayReq variable of the
                 MDPdelayResp state machine (see 11.2.18.2.1 and
                 Figure 11-9) when in the WAITING_FOR_PDELAY_REQ or
                 INITIAL_WAITING_FOR_PDELAY_REQ states, and resulting
                 entry to the SENT_PDELAY_RESP_WAITING_FOR_TIMESTAMP
                 state.
                "
    REFERENCE   "14.10.17"
    ::= { ieee8021AsPortStatIfEntry 14 }

ieee8021AsPortStatTxPdelayResponseFollowUp OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "A counter that increments every time a
                 Pdelay_Resp_Follow_Up message is transmitted, denoted
                 by a transition to TRUE from FALSE of the
                 rcvdMDTimestampReceive variable of the MDPdelayResp
                 state machine (see 11.2.18.2.2 and Figure 11-9) when in
                 the SENT_PDELAY_RESP_WAITING_FOR_TIMESTAMP state, and
                 resulting entry to the WAITING_FOR_PDELAY_REQ state.
                "
    REFERENCE   "14.10.18"
    ::= { ieee8021AsPortStatIfEntry 15 }

ieee8021AsPortStatTxAnnounce OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
               "A counter that increments every time an Announce message
                is transmitted, denoted by entering the
                TRANSMIT_ANNOUNCE state of the PortAnnounceReceive state
                machine (see 10.3.15 and Figure 10-17).
                "
    REFERENCE   "14.10.19"
    ::= { ieee8021AsPortStatIfEntry 16 }
```

```
-- *****************************************************************
-- Acceptable Master Table derived from IEEE 1588-2008.
-- One Table per time-aware system, and used when any of the system
-- is of type IEEE 802.3 EPON, i.e. if any of port in a corresponding
-- system has ieee8021AsPortDSAcceptableMasterTableEnabled set to true.
-- Not used otherwise (Table exists without an entry).
-- *****************************************************************


ieee8021AsAcceptableMasterTableDS
    OBJECT IDENTIFIER ::= { ieee8021AsMIBObjects 7 }

ieee8021AsAcceptableMasterTableDSBase
    OBJECT IDENTIFIER ::= { ieee8021AsAcceptableMasterTableDS 1 }

ieee8021AsAcceptableMasterTableDSMaster
    OBJECT IDENTIFIER ::= { ieee8021AsAcceptableMasterTableDS 2 }

ieee8021AsAcceptableMasterTableDSMaxTableSize OBJECT-TYPE
    SYNTAX      Unsigned32(0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
                "The value is the maximum size of the
                 AcceptableMasterTable. It is equal to the maxTableSize
                 member of the AcceptableMasterTable structure (see
                 13.1.3.2)"
    REFERENCE   "14.7.1 and 13.1.3.2"
    ::= { ieee8021AsAcceptableMasterTableDSBase 1 }

ieee8021AsAcceptableMasterTableDSActualTableSize OBJECT-TYPE
    SYNTAX      Unsigned32(0..65535)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
                "The value is the actual size of the
                 AcceptableMasterTable. It is equal to the
                 actualTableSize member of the AcceptableMasterTable
                   structure (see 13.1.3.2 and 13.1.3.5),
                 i.e., the current number of
                 elements in the acceptable master array. The actual
                 table size is less than or equal to the max table size.

                 This value SHOULD  be reflect the number of entries in
                 the ieee8021AsAcceptableMasterTableDSMasterTable.

                 For a time-aware system that contains an ONU attached
                 to an IEEE 802.3 EPON link, the initialization value
                 is 1. For a time-aware system that does not contain an
                 ONU attached to an IEEE 802.3 EPON link, the
                  initialization value is 0."
    REFERENCE   "14.7.2 and 13.1.3.2"
    ::= { ieee8021AsAcceptableMasterTableDSBase 2 }


ieee8021AsAcceptableMasterTableDSMasterTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021AsAcceptableMasterTableDSMasterEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A table of time-aware port related variables in a time-aware
        bridge or for a time-aware end station.  A value of 1 is used in
       a bridge or an end station that does not have multiple
       components.

        The contents of this table SHALL be maintained across a restart
        of the system.
       "
    REFERENCE
       "IEEE 802.1AS clause 14.11"
    ::= { ieee8021AsAcceptableMasterTableDSMaster 1 }


ieee8021AsAcceptableMasterTableDSMasterEntry   OBJECT-TYPE
    SYNTAX      Ieee8021AsAcceptableMasterTableDSMasterEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "A list of objects pertaining to a port of a time-aware bridge
        component or a time-aware end station.
       "
    INDEX { ieee8021AsAcceptableMasterTableDSMasterId }
    ::= { ieee8021AsAcceptableMasterTableDSMasterTable 1 }


Ieee8021AsAcceptableMasterTableDSMasterEntry ::=
    SEQUENCE {
      ieee8021AsAcceptableMasterTableDSMasterId         Unsigned32,
      ieee8021AsAcceptableMasterClockIdentity           ClockIdentity,
      ieee8021AsAcceptableMasterPortNumber              Unsigned32,
      ieee8021AsAcceptableMasterAlternatePriority1      Unsigned32,
      ieee8021AsAcceptableMasterRowStatus               RowStatus
      }

ieee8021AsAcceptableMasterTableDSMasterId   OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "Acceptable Master row entry index in this
        ieee8021AsAcceptableMasterTabdDSMaster Entry applies.
        If the does not contain Media type of EPON, this variable
        (index) MUST be equal to 0."
    ::= { ieee8021AsAcceptableMasterTableDSMasterEntry 1 }

ieee8021AsAcceptableMasterClockIdentity OBJECT-TYPE
```

```
    SYNTAX      ClockIdentity
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
                "Globally unique manufacturer-assigned clock identifier
                for the local clock port. The identifier is based on an
                EUI-64."
    REFERENCE   "14.7.3"
    ::= { ieee8021AsAcceptableMasterTableDSMasterEntry 2 }

ieee8021AsAcceptableMasterPortNumber OBJECT-TYPE
    SYNTAX      Unsigned32(0..65535)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
                "This object represents a Port or aggregated port
                 on a bridge component or ~~end-station~~end station.
                 This object and ieee8021AsAcceptableMasterClockIdentity
                 together forms AcceptableMasterDS Port Identity."
    REFERENCE   "14.7.3"
    ::= { ieee8021AsAcceptableMasterTableDSMasterEntry 3 }

ieee8021AsAcceptableMasterAlternatePriority1 OBJECT-TYPE
    SYNTAX      Unsigned32(0..255)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
               "If the alternatePriority1 member of the AcceptableMaster
                array element that corresponds to the sourcePortIdentity
                of a received Announce message is greater than 0, the
                value of the grandmasterPriority1 field of the Announce
                message is replaced by the value of alternatePriority1
                of this AcceptableMaster array element for use in the
                invocation of BMCA"
    REFERENCE   "14.7.3 and 13.1.3.4"
    DEFVAL { 244 }
    ::= { ieee8021AsAcceptableMasterTableDSMasterEntry 4 }

ieee8021AsAcceptableMasterRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "This object indicates the status of an entry, and is used
        to create/delete entries.
       "
    REFERENCE  "14.7.3"
    ::= { ieee8021AsAcceptableMasterTableDSMasterEntry 5 }



-- *****************************************************************
-- IEEE 802.1AS MIB Module - Conformance Information
-- *****************************************************************
```

```
ieee8021AsCompliances OBJECT IDENTIFIER ::= { ieee8021AsConformance 1 }
ieee8021AsGroups      OBJECT IDENTIFIER ::= { ieee8021AsConformance 2 }
ieee8021AsCompliancesCor1  OBJECT IDENTIFIER ::= { ieee8021AsConformance
3 }

-- ****************************************************************
-- Units of conformance
-- ****************************************************************

ieee8021ASSystemDefaultReqdGroup OBJECT-GROUP
    OBJECTS {
       ieee8021AsDefaultDSClockIdentity,
       ieee8021AsDefaultDSNumberPorts,
       ieee8021AsDefaultDSClockClass,
       ieee8021AsDefaultDSClockAccuracy,
       ieee8021AsDefaultDSOffsetScaledLogVariance,
       ieee8021AsDefaultDSPriority1,
       ieee8021AsDefaultDSPriority2,
       ieee8021AsDefaultDSGmCapable,
       ieee8021AsDefaultDSCurrentUTCOffset,
       ieee8021AsDefaultDSCurrentUTCOffsetValid,
       ieee8021AsDefaultDSLeap59,
       ieee8021AsDefaultDSLeap61,
       ieee8021AsDefaultDSTimeTraceable,
       ieee8021AsDefaultDSFrequencyTraceable,
       ieee8021AsDefaultDSTimeSource
    }
    STATUS      current
    DESCRIPTION
       "Objects in the System Default required global group."
    ::= { ieee8021AsGroups 1 }

ieee8021ASSystemCurrentGroup OBJECT-GROUP
    OBJECTS {
          ieee8021AsCurrentDSStepsRemoved,
          ieee8021AsCurrentDSOffsetFromMasterHs,
          ieee8021AsCurrentDSOffsetFromMasterMs,
          ieee8021AsCurrentDSOffsetFromMasterLs,
          ieee8021AsCurrentDSLastGmPhaseChangeHs,
          ieee8021AsCurrentDSLastGmPhaseChangeMs,
          ieee8021AsCurrentDSLastGmPhaseChangeLs,
          ieee8021AsCurrentDSLastGmFreqChangeMs,
          ieee8021AsCurrentDSLastGmFreqChangeLs,
          ieee8021AsCurrentDSGmTimebaseIndicator,
          ieee8021AsCurrentDSGmChangeCount,
          ieee8021AsCurrentDSTimeOfLastGmChangeEvent,
          ieee8021AsCurrentDSTimeOfLastGmPhaseChangeEvent,
          ieee8021AsCurrentDSTimeOfLastGmFreqChangeEvent
    }
    STATUS      current
    DESCRIPTION
       "Objects in the System Current global group."
    ::= { ieee8021AsGroups 2 }
```

```
ieee8021AsSystemClockParentGroup OBJECT-GROUP
    OBJECTS {
        ieee8021AsParentDSParentClockIdentity,
        ieee8021AsParentDSParentPortNumber,
        ieee8021AsParentDSCumlativeRateRatio,
        ieee8021AsParentDSGrandmasterIdentity,
        ieee8021AsParentDSGrandmasterClockClass,
        ieee8021AsParentDSGrandmasterClockAccuracy,
        ieee8021AsParentDSGrandmasterOffsetScaledLogVariance,
        ieee8021AsParentDSGrandmasterPriority1,
        ieee8021AsParentDSGrandmasterPriority2
    }
    STATUS      current
    DESCRIPTION
       "Objects in the Clock Parent global group."
    ::= { ieee8021AsGroups 3 }

ieee8021AsSystemTimePropertiesGroup OBJECT-GROUP
    OBJECTS {
         ieee8021AsTimePropertiesDSCurrentUtcOffset,
         ieee8021AsTimePropertiesDSCurrentUtcOffsetValid,
         ieee8021AsTimePropertiesDSLeap59,
         ieee8021AsTimePropertiesDSLeap61,
         ieee8021AsTimePropertiesDSTimeTraceable,
         ieee8021AsTimePropertiesDSFrequencyTraceable,
         ieee8021AsTimePropertiesDSTimeSource
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Time Properties Global group."
    ::= { ieee8021AsGroups 4 }

ieee8021AsPortDataSetGlobalGroup OBJECT-GROUP
    OBJECTS {
           ieee8021AsPortDSClockIdentity,
           ieee8021AsPortDSPortNumber,
           ieee8021AsPortDSPort~~Role~~State,
           ieee8021AsPortDSPtp~~t~~PortEnabled,
           ieee8021AsPortDSIsMeasuringDelay,
           ieee8021AsPortDSAsCapable,
           ieee8021AsPortDSNeighborPropDelayHs,
           ieee8021AsPortDSNeighborPropDelayMs,
           ieee8021AsPortDSNeighborPropDelayLs,
           ieee8021AsPortDSNeighborPropDelayThreshHs,
           ieee8021AsPortDSNeighborPropDelayThreshMs,
           ieee8021AsPortDSNeighborPropDelayThreshLs,
           ieee8021AsPortDSDelayAsymmetryHs,
           ieee8021AsPortDSDelayAsymmetryMs,
           ieee8021AsPortDSDelayAsymmetryLs,
           ieee8021AsPortDSNeighborRateRatio,
           ieee8021AsPortDSInitialLogAnnounceInterval,
           ieee8021AsPortDSCurrentLogAnnounceInterval,
           ieee8021AsPortDSAnnounceReceiptTimeout,
           ieee8021AsPortDSInitialLogSyncInterval,
           ieee8021AsPortDSCurrentLogSyncInterval,
           ieee8021AsPortDSSyncReceiptTimeout,
           ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalHs,
           ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalMs,
           ieee8021AsPortDSSyncReceiptTimeoutTimeIntervalLs,
           ieee8021AsPortDSInitialLogPdelayReqInterval,
           ieee8021AsPortDSCurrentLogPdelayReqInterval,
           ieee8021AsPortDSAllowedLostResponses,
           ieee8021AsPortDSVersionNumber,
           ieee8021AsPortDSNupMs,
           ieee8021AsPortDSNupLs,
           ieee8021AsPortDSNdownMs,
           ieee8021AsPortDSNdownLs,
           ieee8021AsPortDSAcceptableMasterTableEnabled
    }
    STATUS      current
    DESCRIPTION
       "Objects for the port dataset  global group."
    ::= { ieee8021AsGroups 5 }
ieee8021ASPortStatisticsGlobalGroup OBJECT-GROUP
    OBJECTS {
         ieee8021AsPortStatRxSyncCount,
         ieee8021AsPortStatRxFollowUpCount,
         ieee8021AsPortStatRxPdelayRequest,
         ieee8021AsPortStatRxPdelayResponse,
         ieee8021AsPortStatRxPdelayResponseFollowUp,
         ieee8021AsPortStatRxAnnounce,
         ieee8021AsPortStatRxPTPPacketDiscard,
         ieee8021AsPortStatRxSyncReceiptTimeouts,
         ieee8021AsPortStatAnnounceReceiptTimeouts,
         ieee8021AsPortStatPdelayAllowedLostResponsesExceeded,
         ieee8021AsPortStatTxSyncCount,
         ieee8021AsPortStatTxFollowUpCount,
         ieee8021AsPortStatTxPdelayRequest,
         ieee8021AsPortStatTxPdelayResponse,
         ieee8021AsPortStatTxPdelayResponseFollowUp,
         ieee8021AsPortStatTxAnnounce
    }
    STATUS      current
    DESCRIPTION
       "Objects in the Port statistics global group."
    ::= { ieee8021AsGroups 6 }


ieee8021AsAcceptableMasterBaseGroup OBJECT-GROUP
    OBJECTS {
      ieee8021AsAcceptableMasterTableDSMaxTableSize,
      ieee8021AsAcceptableMasterTableDSActualTableSize
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Acceptable Master group."
    ::= { ieee8021AsGroups 7 }
```

```
ieee8021AsAcceptableMasterTableGroup OBJECT-GROUP
    OBJECTS {
      ieee8021AsAcceptableMasterClockIdentity,
      ieee8021AsAcceptableMasterPortNumber,
      ieee8021AsAcceptableMasterAlternatePriority1,
      ieee8021AsAcceptableMasterRowStatus
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Acceptable Master group."
    ::= { ieee8021AsGroups 8 }


-- ******************************************************************
-- MIB Module Compliance statements
-- ******************************************************************

ieee8021AsCompliance MODULE-COMPLIANCE
    STATUS      deprecated
    DESCRIPTION
       "The compliance statement for support of
        the IEEE8021-AS-MIB module."

    MODULE SNMPv2-MIB -- The SNMPv2-MIB, RFC 3418
        MANDATORY-GROUPS {
            systemGroup
        }

    MODULE IF-MIB -- The interfaces MIB, RFC 2863
        MANDATORY-GROUPS {
            ifGeneralInformationGroup
        }

    MODULE
        MANDATORY-GROUPS {
            ieee8021ASSystemDefaultReqdGroup,
            ieee8021ASSystemCurrentGroup,
            ieee8021AsSystemClockParentGroup,
            ieee8021AsSystemTimePropertiesGroup,
            ieee8021AsPortDataSetGlobalGroup,
            ieee8021AsAcceptableMasterBaseGroup,
            ieee8021AsAcceptableMasterTableGroup
         }

    OBJECT ieee8021AsAcceptableMasterRowStatus
      SYNTAX       RowStatus { active(1), notInService(2) }
      WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                               destroy(6) }
      DESCRIPTION "Support for createAndWait is not required."

    GROUP ieee8021ASPortStatisticsGlobalGroup
        DESCRIPTION
          "This group is optional and provides time-aware Bridges and
```

```
                end stations that choose to implement gPTP port statistics."

        ::= { ieee8021AsCompliances 1 }


    ieee8021AsComplianceCor1 MODULE-COMPLIANCE
        STATUS      current
        DESCRIPTION
           "The compliance statement for support of
            the IEEE8021-AS-MIB module."

        MODULE SNMPv2-MIB -- The SNMPv2-MIB, RFC 3418
            MANDATORY-GROUPS {
                systemGroup
            }

        MODULE IF-MIB -- The interfaces MIB, RFC 2863
            MANDATORY-GROUPS {
                ifGeneralInformationGroup
            }

        MODULE
            MANDATORY-GROUPS {
                ieee8021ASSystemDefaultReqdGroup,
                ieee8021ASSystemCurrentGroup,
                ieee8021AsSystemClockParentGroup,
                ieee8021AsSystemTimePropertiesGroup,
                ieee8021AsPortDataSetGlobalGroup,
                ieee8021AsAcceptableMasterBaseGroup,
                ieee8021AsAcceptableMasterTableGroup
             }

        OBJECT ieee8021AsParentDSGrandmasterPriority1
          MIN-ACCESS read-only
          DESCRIPTION "Support for write is not required and deprecated."

        OBJECT ieee8021AsParentDSGrandmasterPriority2
          MIN-ACCESS read-only
          DESCRIPTION "Support for write is not required and deprecated."

        OBJECT ieee8021AsAcceptableMasterRowStatus
          SYNTAX       RowStatus { active(1), notInService(2) }
          WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                                   destroy(6) }
          DESCRIPTION "Support for createAndWait is not required."

        GROUP ieee8021ASPortStatisticsGlobalGroup
            DESCRIPTION
               "This group is optional and provides time-aware Bridges and
                end stations that choose to implement gPTP port statistics."

        ::= { ieee8021AsCompliances 2 }
```

```
END
```

16. Media-dependent layer specification for CSN Network

~~(normative)~~

~~Media-dependent layer specification for CSN Network~~

<<Editor's note: References to the MIB in Clause 16 are on the following pages of IEEE 802.1AS/Cor 1-2013: pages 71, 72, 73 (2 places), 81, and 82.>>

<<Editor's note: On p.vi of IEEE Std 802.1AS-2011, Philippe Klein is listed as the clause editor of Annex F. This should be ~~Annex E, and should be~~ changed to Clause 16~~after the references to Annex E are updated~~.>>

16.1 Overview

Accurate synchronized time is distributed throughout a gPTP domain through time measurements between adjacent time-aware ~~node~~relays or end stations in a ~~bridged LAN~~packet network. Time is communicated from the root of the clock spanning tree (i.e., the grandmaster) toward the leaves of the tree (i.e., from leaf-facing “master” ports to root-facing “slave” ports) through measurements made across the links connecting the time-aware systems. While the semantics of time transfer are consistent across the time-aware ~~bridged LAN~~packet network, the method for communicating synchronized time from a master station to its immediate downstream link partner varies depending on the type of link interconnecting the two time-aware systems.

This appendix specifies the protocol that provides accurate synchronized time across links of a *coordinated shared network* (CSN) as part of a ~~bridged LAN~~packet network.

16.2 Coordinated Shared Network characteristics

A CSN is a contention-free, time-division, multiplexed-access network of devices sharing a common medium and supporting reserved bandwidth based on priority or flow (QoS). One of the nodes of the CSN acts as the network coordinator, granting transmission opportunities to the other~~-~~CSN nodes of the network. A CSN network physically is a shared medium, in that a CSN node has a single physical port connected to the half-duplex medium, but is logically a fully connected one-hop mesh network, in that every~~-~~CSN node can transmit frames to every other CSN node over the shared medium.

A CSN supports two types of transmission: unicast transmission for point-to-point (CSN node-to-node) transmission and multicast/broadcast transmission for point-to-multipoint (CSN node-to-other/all-nodes) transmission. Figure 16-1 illustrates a CSN network acting as a backbone for time-aware systems.

NOTE—In this clause~~annex~~, the term *node* is used to refer to a CSN node (i.e., it does not refer to a time-aware ~~Bridge~~relay or end station). A CSN node is a 2-port ~~Bridge~~time-aware relay that forwards data packets between a segment external to the CSN (which ~~may~~can connect to an upstream or downstream time-aware system~~ or Bridge~~) and the CSN network, all at the data link layer. Nonetheless, to avoid confusion the term *node* is usually preceded by *CSN*, except in cases where it is obvious that CSN nodes are being referred to.

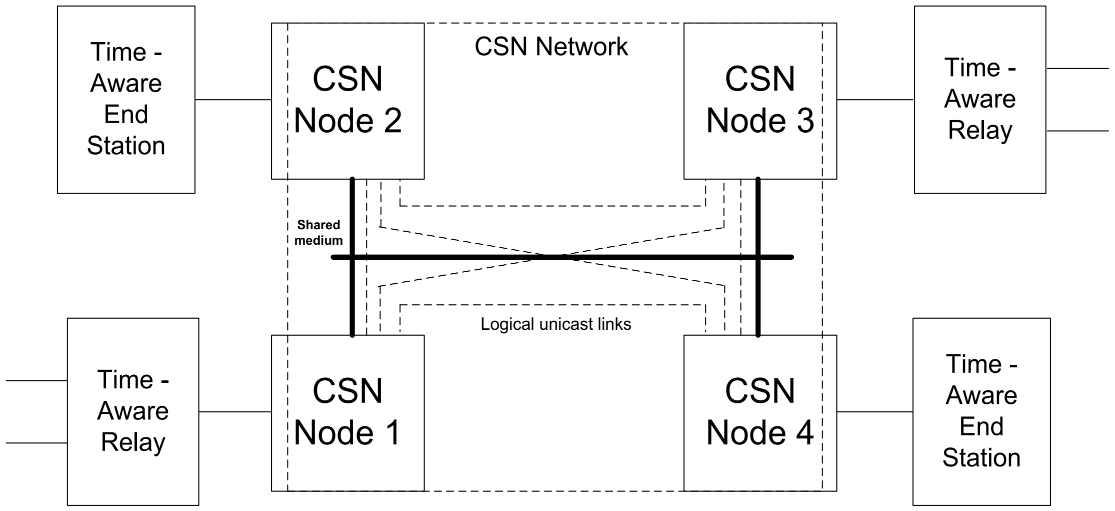


Figure 16-1—Example of CSN backbone in an AVB LAN

16.3 Layering for CSN links

One PortSync entity and one MD entity are together associated with each CSN logical port (CSN node-to-node link) as illustrated in Figure 16-2. The PortSync entities is described in 10.1.1. The MD entity translates media-independent primitives to MD primitives as necessary for communicating synchronized time over the CSN links. The CSN MD entity shall implement the MDSyncSendSM and MDSyncReceiveSM states machines of 11.2.13 and 11.2.14.

The CSN MD entity either implements the MDPdelayReq and MDPdelayResp state machines of 11.2.15 and to measure the propagation delay on a CSN link, or measures it through a CSN-native method and populates the variables described in 16.4.3.2.

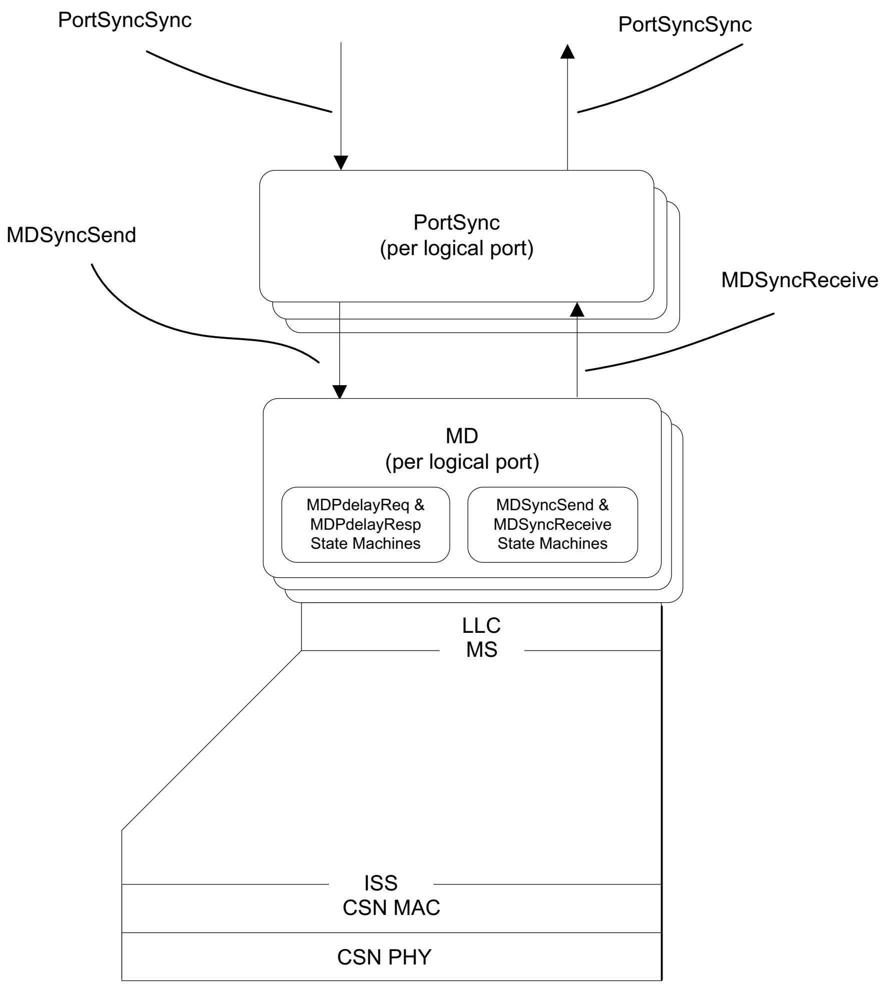


Figure 16-2—Media-dependent and lower entities in CSN nodes

16.4 Path delay measurement over a CSN backbone

16.4.1 General

The Path Delay over a CSN backbone is calculated for the following path types: (1) between the upstream time-aware ~~Bridge~~relay and the ingress CSN node, (2) between the ingress and egress CSN nodes, and (3) between the egress CSN node and the downstream time-aware system (~~Bridge~~time-aware relay or time-aware end station).

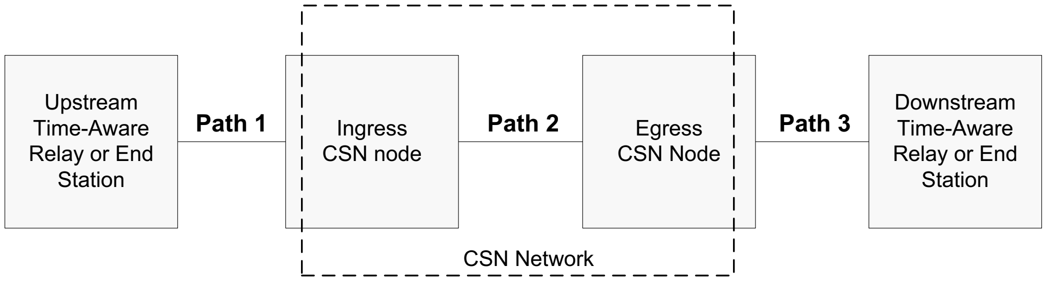

Upstream Time-Aware Relay or End Station
Path 1
Ingress CSN node
Path 2
Egress CSN Node
Path 3
Downstream Time-Aware Relay or End Station
CSN Network

Figure 16-3—Path types over CSN as IEEE 802.1AS backbone

To maintain the synchronization, residence time on each ~~node~~time-aware system and the propagation delay between ~~nodes~~time-aware systems is measured, requiring precise timestamping on both CSN node ingress and egress ports as illustrated in Figure 16-4 ("Path i" in the figure refers to the paths enumerated in Figure 16-3). In Figure 16-4, ti_1 is the ~~<~~syncEventEgressTimestamp~~>~~ for the Sync message at the upstream time-aware ~~Bridge~~relay, ti_2 is the ~~<~~syncEventIngressTimestamp~~>~~ for the Sync message at the ingress CSN time-aware ~~Bridge~~node, te_1 is the ~~<~~syncEventEgressTimestamp~~>~~ for the Sync message at the egress CSN time-aware ~~Bridge~~node, and te_2 is the ~~<~~syncEventIngressTimestamp~~>~~ for the Sync message at the downstream time-aware ~~Bridge~~relay or ~~end-station~~end station.

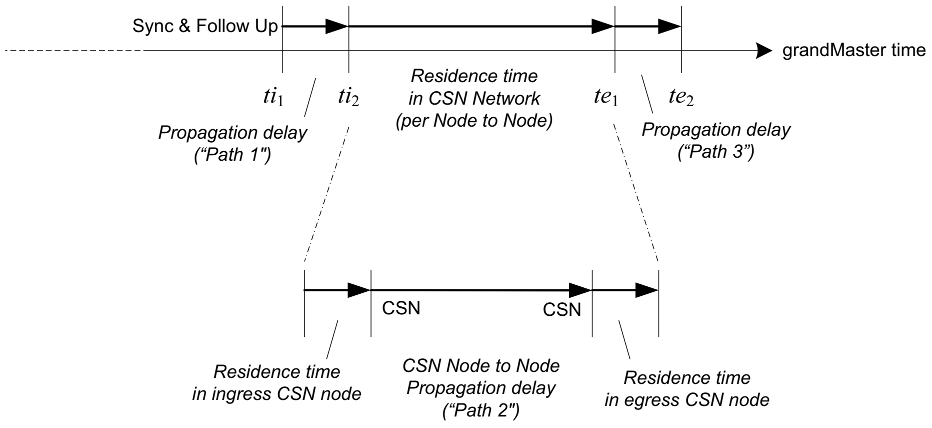

Sync & Follow Up
grandMaster time
ti1
ti2
Residence time in CSN Network (per Node to Node)
te1
te2
Propagation delay ("Path 1")
Propagation delay ("Path 3")
CSN
CSN
Residence time in ingress CSN node
CSN Node to Node Propagation delay ("Path 2")
Residence time in egress CSN node

Figure 16-4—Propagation delay and residence time over a CSN Backbone

16.4.2 Path delay measurement between CSN node and neighbor time-aware system

The path delay measurement between a CSN node and a neighbor time-aware system is made as specified for the respective medium. This path delay measurement is made for the link between the CSN node and the neighbor time-aware system.

16.4.3 Path delay measurement between CSN nodes

The path delay between the two nodes of a CSN is the propagation delay for the logical link that connects those two nodes. The method of measuring the path delay between two CSN nodes has two variations which are described in 16.4.3.1 and 16.4.3.2, respectively. The specific method to be used for a specific link technology is specified in 16.6.

16.4.3.1 Path delay measurement without network clock reference-

Each CSN node has a free-running local clock. The path delay measurement uses the protocol, messages, and state machines described in Clause 11 for full-duplex, point-to-point links, as illustrated by Figure 16-5.

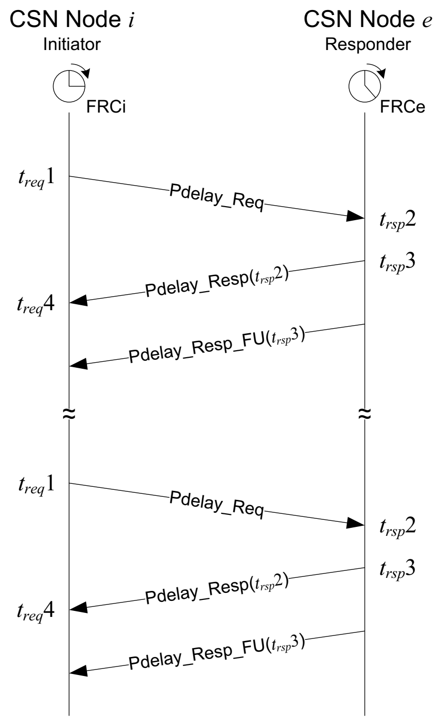


Figure 16-5—CSN node-to-node path delay measurement

The computation of the neighborRateRatio and neighborPropDelay between two CSN nodes is done using the timestamps at the initiator and information conveyed in the successive Pdelay_Resp and Pdelay_Resp_Follow_Up messages. Any scheme that uses this information is acceptable, as long as the performance requirements of B.2.4 are met. As one example, the neighborRateRatio is computed as the ratio between a time interval measured by the local clock of the responder and its associated time interval measured by the local clock of the initiator, using a set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages and a second set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages some number of Pdelay_Req message transmission intervals later, i.e., as show in Equation (16.1):

$$\frac{(t_{rsp}3)_N - (t_{rsp}3)_0}{(t_{req}4)_N - (t_{req}4)_0} \tag{16.1}$$

where $(t_{rsp}3)_k$ is the time relative to the local clock of the responder that the k^{th} Pdelay_Resp message is sent, $(t_{req}4)_k$ is the time relative to the local clock of the initiator that the k^{th} Pdelay_Resp message is received, N is the number of Pdelay_Req message transmission intervals separating the first set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages and the second set, and the successive sets of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages are indexed from 0 to N with the first set indexed 0. The neighborPropDelay between the time-aware system and the CSN node is computed as shown in Equation (16.2):

$$\frac{(t_{req}4 – t_{req}1)r - (t_{rsp}3 - t_{rsp}2)}{2} \tag{16.2}$$

where r is equal to neighborRateRatio, $t_{req}1$ is the time relative to the local clock of the initiator that the Pdelay_Req message for this message exchange is sent, $t_{rsp}2$ is the time relative to the local clock of the responder that the Pdelay_Req message for this message exchange is received, $t_{rsp}3$ is the time relative to the local clock of the responder that the Pdelay_Resp message for this message exchange is sent, and $t_{req}4$ is the time relative to the local clock of the initiator that the Pdelay_Resp message for this message exchange is received.

NOTE—The difference between mean propagation delay relative to the grandmaster time base and relative to the time base of the CSN node at the other end of the attached link (i.e., the responder CSN node) 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 frequency to the frequency of the LocalClock entity of the CSN node at the other end of the link. This ratio differs from 1 by 200 ppm or less. For example, for a worst-case frequency offset of the LocalClock entity of the CSN node at the other end of the link, relative to the grandmaster, of 200 ppm, and a measured propagation time of 100 ns, the difference in mean propagation delay relative to the two time bases is 20 ps.

Although the propagation delay between two CSN nodes is constant, a Pdelay_Req message is still sent periodically by each CSN node to each other active CSN node of the network to measure the neighborRateRatio between the node and each other node. Each CSN node shall implement the state machines described in 11.2.15 and .

16.4.3.2 Native 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 ~~may~~can be provided using the native measurement method rather than using the Pdelay protocol defined in 11.2.15 and 11.2.18. Such a situation is described in more detail as follows. The CSN MD entity populates the following per-port and MD-entity global variables (described respectively in 10.2.4 and 11.2.12) as indicated:

— asCapable (10.2.4.1) is set to TRUE,
— neighborRateRatio (10.2.4.7) is set to the value provided by the native CSN measurement,
— neighborPropDelay (10.2.4.8) is set to the value provided by the native CSN measurement,
— computeNeighborRateRatio (10.2.4.10) is set to FALSE,
— computeNeighborPropDelay (10.2.4.11) is set to FALSE, and
— isMeasuringDelay (11.2.12.6) is set to TRUE to indicate that the CSN MD entity is measuring path delay (in this case, using its internal mechanism).
— domainNumer (8.1) is set to the domain number of this gPTP domain

16.4.3.3 Intrinsic CSN path delay measurement

If the CSN network features a native mechanism that causes each CSN node's local clock to be fully synchronized to the local clocks of other nodes of the CSN such that the synchronized CSN time complies with the requirements specified in B.1, the CSN nodes need not implement the path delay mechanism but rather treat the path delay as part of the residence time of the distributed system. The propagation of the Sync messages in this case is described in 16.5.2.

16.5 Synchronization messages

The CSN network shall propagate synchronized time over the CSN to CSN end stations and to downstream non-CSN links, using Sync (and, if the message is two-step, the associated Follow_Up) messages, as illustrated in Figure 16-6.

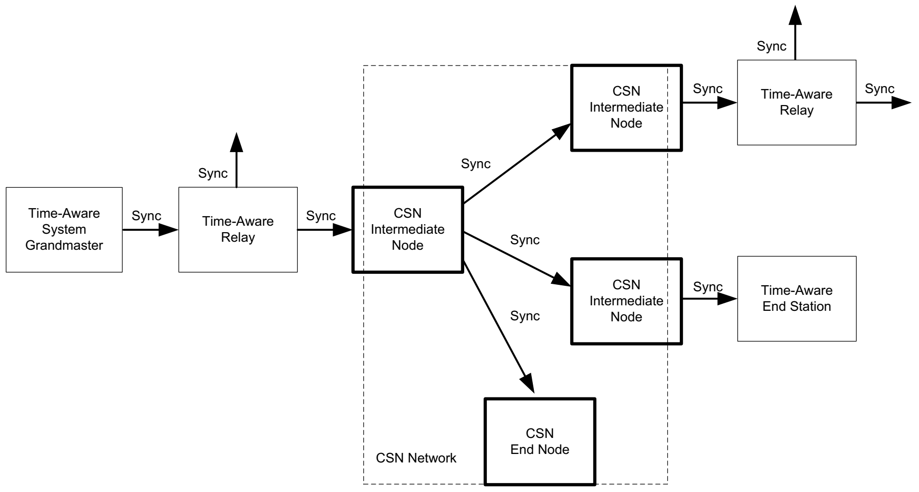


Figure 16-6—IEEE 802.1AS Sync Message Propagation over the CSN backbone

Once the path delays have been measured (a) between the upstream time-aware ~~Bridge~~relay or end station and the ingress CSN node, (b) between the CSN nodes, and (c) between the egress CSN node and the downstream time-aware ~~Bridge~~relay or end station, the CSN backbone can propagate the synchronization information received at its boundary nodes.

As with path delay measurements, various CSN technologies choose various methods for propagating time. These methods are described as follows.

16.5.1 Synchronization message propagation on CSN without network reference clock

If the CSN network does not feature a native mechanism that synchronizes the CSN node local clocks to each other or to a reference, such that the CSN synchronized time complies with the requirements specified in B.1, the CSN local clock at CSN ingress and CSN egress nodes are considered independent free running clocks.

In this case, synchronization over the CSN links uses the Sync and Follow_Up protocol (or only Sync if optional one-step processing is used), messages, and state machines specified for full-duplex point-to-point links in 10.2.7, 10.2.11, 11.2.13, and 11.2.14. Individual CSN link technologies may specify media-specific encapsulation of gPTP event messages. See Table 16-2 for selection of options per link technology.

One PortSync and one MD entity is instantiated per logical port (i.e., per CSN link). A CSN node behaves equivalently to a time-aware system. Sync and, in the two-step case, Follow_Up messages are either transmitted using unicast on each link or broadcasted. However if Sync and Follow_Up messages are broadcasted:

— the Sync and Follow_Up messages are broadcast with the same port number used to broadcast Announce messages,

— all PortSync/MD entity pairs except one set their logSyncInterval attribute (see 10.7.2.3) to 127, causing them to not generate any Sync messages, and
— a dynamic selection of the MD entity that broadcasts the Sync message is needed (a CSN node can dynamically leave the CSN network). The dynamic selection algorithm is implementation specific and out of the scope of this standard.

16.5.2 Synchronization message propagation on a CSN with network reference clock

If the CSN network features a native mechanism that allows the CSN node's local clocks to be fully synchronized to each other in a way that complies with the requirements specified in B.1, it is possible to simplify the path delay mechanism, as described below. This method is an alternative to 16.5.1. Individual CSN link technologies may specify media-specific encapsulation of gPTP event messages. See Table 16-2 for selection of options per link technology.

Sync messages are timestamped (1) when received at the ingress CSN node's time-aware system port (~~<~~syncEventIngressTimestamp~~>~~) and (2) when transmitted at the egress CSN node's time-aware system port (~~<~~syncEventEgressTimestamp~~>~~). The elapsed time between the egress and ingress timestamps is computed as the CSN residence time. In this scheme, the Sync message handling is split between the MDSyncSendSM state machine in the CSN ingress node and the MDSyncReceiveSM state machine in the CSN egress node as described in 16.5.2.1 and 16.5.2.2.

The reference plane for a CSN port and the path delay measurement method are specific to the type of CSN technology, and are defined in Table 16-2.

16.5.2.1 CSN ingress node

The CSN ingress node timestamps Sync messages received from the upstream time-aware system and compute the upstreamTxTime as described in 11.2.13.2.1, item f).

In addition, the setFollowUp function of the MDSyncSendSM state machine [see 11.2.14.2.3, item a)] is~~be~~ modified as follows:

a) The quantity *rateRatio* × (~~<~~*syncEventEgressTimestamp* – *upstreamTxTime*~~>~~) is not added to the followUpCorrectionField of the Follow_Up message (or the Sync message in the one-step case) to be transmitted by the ingress to the egress CSN node
b) The CSN TLV (see 16.5.2.1.1) is appended to the Follow_Up message (or to the Sync message in the one-step case) transmitted by the ingress to the egress CSN node.

16.5.2.1.1 CSN TLV

16.5.2.1.1.1 General

The fields of the CSN TLV are specified in Table 16-1 and 16.5.2.1.1.3 through 16.5.2.1.1.7. This TLV is a standard organization extension TLV for the Follow_Up message (or the Sync message in the one-step case), as specified in 14.3 of IEEE Std 1588-2008. This TLV is not allowed to occur before the Follow_Up information TLV (see 11.4.4.3).

Table 16-1—CSN TLV

Bits	Octets	Offset From Start of TLV
~~8~~7 ~~7~~6 ~~6~~5 ~~5~~4 43 ~~3~~2 ~~2~~1 ~~1~~0		
tlvType	2	0
lengthField	2	2
organizationId	3	4
organizationSubType	3	7
rxTime	12	10
neighborRateRatio	4	14
neighborPropDelay	12	26
delayAsymmetry	12	38
domainNumber	1	50

16.5.2.1.1.2 tlvType (Enumeration16)

The value of the tlvType field is 0x3.

NOTE—This is the value that indicates the TLV is a vendor and standard organization extension TLV, as specified in 14.3.2.1 of IEEE Std 1588-2008. The value is specified there as ORGANIZATION_EXTENSION, whose value is 0x3.

16.5.2.1.1.3 lengthField (UInteger16)

The value of the length is 46.

16.5.2.1.1.4 organizationId (Octet3)

The value of organizationId is 00-80-C2.

16.5.2.1.1.5 organizationSubType (Enumeration24)

The value of organizationSubType is 3.

16.5.2.1.1.6 upstreamTxTime (UScaledNs)

The computed upstreamTxTime value as described in 11.2.13.2.1 item f).

16.5.2.1.1.7 neighborRateRatio (Integer32)

The neighborRateRatio value described in 10.2.4.7.

16.5.2.1.1.8 neighborPropDelay (UScaledNs)

The neighborPropDelay value described in 10.2.4.8.

16.5.2.1.1.9 delayAsymmetry (~~U~~ScaledNs)

The delayAsymmetry value described in 10.2.4.9.

16.5.2.1.1.10 domainNumber(UInteger8)

The domain number of this gPTP domain.

16.5.2.2 CSN egress node

The CSN egress port sets neighborRateRatio to 1 and neighborPropDelay to 0 for its CSN port.

The CSN egress port modifies the function setMDSyncReceive of the MDSyncReceiveSM state machine [11.2.13.2.1 item f)] for the port attached to the CSN link entity to extract upstreamTxTime, neighborPropDelay, and neighborRateRatio from the respective fields of the CSN TLV in the Follow_Up message (or Sync message in the one-step case) received from the CSN ingress node.

The CSN egress port also modifies the ClockSlaveSync state machine (see 10.2.12) to get the upstreamTxTime, neighborPropDelay, neighborRateRatio, and delayAsymmetry values from the respective fields of the CSN TLV in the Follow_Up message (or Sync message in the one-step case) received from the CSN ingress node.

The CSN egress node removes the CSN TLV from the Follow_Up message (or Sync message in the one-step case) ~~it creates~~received from the CSN ingress node and transmits the messageto the downstream time-aware system.

16.6 Specific CSN requirements

The reference plane for a CSN port is specific to the type of CSN technology and is defined in Table 16-2.

Table 16-2—Definitions and option selections per link technology

CSN link technology	Reference plane	Path Delay measurement	gPTP event message encapsulation
Multimedia over Coax Alliance (MoCA) v2.0	The first bit of an Event message crossing to and from the MoCA CTC clock domain.	MoCA Ranging Protocol (method of either 16.4.3.2 or 16.4.3.3)	Encapsulated in control frames as described in the MoCA MAC/PHY Specification v2.0.
ITU-T G.hn (SG15)	The first bit of an Event message crossing the A-interface. See 16.6.2.	Pdelay mechanism as defined in 16.4.3.1.	None

16.6.1 MoCA-specific behavior

The MoCA network is a CSN. The non-MoCA port of a ~~Bridge~~CSN time-aware node behaves as a time-aware system port, which might or might not use the MoCA CTC clock for timestamping event messages. If the non-MoCA port of the ~~Bridge~~CSN time-aware node uses a different clock than the MoCA CTC clock, then the ~~Bridge~~CSN time-aware node reconciles the non-CTC timestamp with the CTC time.

Sync messages shall be timestamped using the CTC clock (1) when the Sync message crosses the MoCA~~'s~~ ingress node's timestamp reference plane (~~<~~syncEventIngressTimestamp~~>~~) and (2) when it crosses the egress CSN node's reference plane (~~<~~syncEventEgressTimestamp~~>~~).

The elapsed time between the egress and ingress timestamps, ~~<~~syncEventIngressTimestamp~~>~~ – ~~<~~syncEventIngressTimestamp~~>~~, is computed as the MoCA residence time.

The MoCA port whose port ~~role~~state is MasterPort propagates the Sync and Follow_Up messages (or only the Sync message in the one-step case) as described in 16.5.2. The CSN TLV values of the Follow_Up message sent over the MoCA network ~~is~~are computed using the LocalClock, i.e., the MoCA CTC clock.

IEEE 802.1 AS Frames shall be transmitted over the MoCA network as MoCA control frames as described in the MoCA MAC/PHY Specification v2.0.

NOTE—The Channel-Time Clock (CTC) is specified in the MoCA MAC/PHY Specification v2.0.

16.6.2 ITU-T G.hn-specific behavior

A port of a time-aware system that includes one or more ITU-T G.hn ports shall behave as defined in 16.3, 16.4, and 16.5, but for aspects where more than one behavior or option is described, the system behaves as defined in Table 16-2.

ITU-T G.hn defines a 32-bit timestamp (which is placed the TSMP field of a G.hn encapsulation header). This timestamp, as described in Table 16-2, is captured each time an Event message is transmitted or received by an ITU-T G.hn port and is used for all gPTP event messages, including SYNC and PDELAY messages.

16.7 Grandmaster capability

Each CSN Node may be grandmaster capable, allowing a CSN node to act as the grandmaster clock either for a homogeneous CSN network or for a heterogeneous network.

The Announce messages are either sent via unicast on each link or broadcasted. However if Announce messages are broadcasted, the Announce message shall use the same port number as used by the Sync and Follow_Up messages (or only the Sync message in the one-step case) by a single PortSync/MD entity pair on the port. This is accomplished by setting the logAnnounceInterval attribute (see 10.7.2.2) to 127 for all but one PortSync/MD entity pair, causing them to not generate any Announce messages.

16.8 CSN clock and node performance requirements

The CSN clock performance ~~shall comply~~complies with the requirements specified in B.1. The CSN node performance ~~shall comply~~should follow the recommendations of B.2.2 and B.2.3. The CSN node performance complies with the requirements specified in ~~B.2.2, B.2.3, and~~ B.2.4.

Annex A

(normative)

Protocol Implementation Conformance Statement (PICS) proforma[18]

A.1 Introduction

The supplier of a protocol implementation that is claimed to conform to this standard shall complete the following PICS proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. The PICS can have a number of uses, including the following:

a) By the protocol implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight;
b) By the supplier and acquirer—or potential acquirer—of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
c) By the user—or potential user—of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICS);
d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A.2 Abbreviations and special symbols

A.2.1 Status symbols

M	mandatory
O	optional
O.n	optional, but support of at least one of the group of options labeled by the same numeral n is required
X	prohibited
pred:	conditional-item symbol, including predicate identification (see A.3.4)
¬	logical negation, applied to a conditional item’s predicate

A.2.2 General abbreviations

N/A	not applicable
PICS	Protocol Implementation Conformance Statement

[18]*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.3 Instructions for completing the PICS proforma

A.3.1 General structure of the PICS proforma

The first part of the PICS proforma, implementation identification and protocol summary, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire, divided into several subclauses, each containing a number of individual items. Answers to the questionnaire items are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (usually Yes or No), or by entering a value or a set or range of values. (Note that there are some items where two or more choices from a set of possible answers can apply; all relevant choices are to be marked.)

Each item is identified by an item reference in the first column. The second column contains the question to be answered; the third column records the status of the item—whether support is mandatory, optional, or conditional: see also A.3.4. The fourth column contains the reference or references to the material that specifies the item in the main body of this standard, and the fifth column provides the space for the answers.

A supplier may also provide (or be required to provide) further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labeled *Ai* or *Xi,* respectively, for cross-referencing purposes, where *i* is any unambiguous identification for the item (e.g., simply a numeral). There are no other restrictions on its format and presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the Protocol Implementation Conformation Statement for the implementation in question.

NOTE—Where an implementation is capable of being configured in more than one way, a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation's configuration capabilities, in case that makes for easier and clearer presentation of the information.

A.3.2 Additional Information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and a PICS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations, or information about aspects of the implementation that are outside the scope of this standard but that have a bearing upon the answers to some items.

References to items of Additional Information may be entered next to any answer in the questionnaire and may be included in items of Exception Information.

A.3.3 Exception Information

It may occasionally happen that a supplier will wish to answer an item with mandatory status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this; instead, the supplier shall write the missing answer into the Support column, together with an *Xi* reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception item itself.

An implementation for which an Exception item is required in this way does not conform to this standard.

NOTE—A possible reason for the situation described above is that a defect in this standard has been reported, a correction for which is expected to change the requirement not met by the implementation.

A.3.4 Conditional status

A.3.4.1 Conditional items

The PICS proforma contains a number of conditional items. These are items for which both the applicability of the item itself, and its status if it does apply—mandatory or optional—are dependent upon whether or not certain other items are supported.

Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the Not Applicable (N/A) answer is selected. Otherwise, individual conditional items are indicated by a conditional symbol in the Status column.

A conditional symbol is of the form "**pred:** S" where **pred** is a predicate as described in A.3.4.2, and S is a status symbol, M or 0.

If the value of the predicate is true (see A.3.4.2), the conditional item is applicable, and its status is indicated by the status symbol following the predicate; the answer column is to be marked in the usual way. If the value of the predicate is false, the N/A answer is to be marked.

A.3.4.2 Predicates

A predicate is one of the following:

a) An item-reference for an item in the PICS proforma; the value of the predicate is true if the item is marked as supported, and is false otherwise;
b) A predicate-name, for a predicate defined as a Boolean expression constructed by combining item-references using the Boolean operator OR; the value of the predicate is true if one or more of the items is marked as supported;
c) The logical negation symbol "¬" prefixed to an item-reference or predicate-name; the value of the predicate is true if the value of the predicate formed by omitting the "¬" symbol is false, and vice versa.

Each item whose reference is used in a predicate or predicate definition, or in a preliminary question for grouped conditional items, is indicated by an asterisk in the Item column.

A.4 PICS proforma for IEEE Std 802.1AS-2011

NOTE 1—Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification.

NOTE 2—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).

A.4.1 Implementation identification

Supplier	
Contact point for queries about the PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating system names	

A.4.2 Protocol summary, IEEE Std 802.1AS

Identification of protocol specification	IEEE Std 802.1AS-2011, IEEE Standard for Local and metropolitan area networks—Timing and Synchronization for Time-Sensitive Applications ~~in Bridged Local Area Networks~~
Identification of amendments and corrigenda to the PICS proforma which have been completed as part of the PICS	Amd. : Corr. : Amd. : Corr. :
Have any Exception items been required? (See A.3.3: the answer Yes means that the implementation does not conform to IEEE Std 802.1AS-2011)	No [ ] Yes [ ]

Date of Statement	

A.5 Major capabilities

Item	Feature	Status	References	Support
MIB	Is the IEEE 8021-AS-MIB module fully supported (per its MODULE-COMPLIANCE)?	O	Clause 15	Yes [ ] N/A [ ]
DOM0	Does the device support domain 0, in accordance with the requirements of 8.1?	M	8.1	Yes [ ]
DOMADD	Does the device support one or more domans whose domain numbers are in the range 1-127?	O	8.1	Yes [ ] No [ ]
MINTA	Does the device support media-independent slave functionality on at least one port?	M	10.2.12	Yes [ ]
BMC	Does the device implement the best master clock algorithm?	M	10.3	Yes [ ]
SIG	Does the device transmit signaling messages?	O	10.6.4	Yes [ ] No [ ]
GMCAP	Is the device capable of acting as a grandmaster?	O	8.6.2.1, 10.1.2	Yes [ ] No [ ]
BRDG	Does the device act as a time-aware ~~Bridge~~relay on two or more ports?	O	10.2.6	Yes [ ] No [ ]
MIMSTR	Does the device support media-independent master functionality on at least one port?	GMCAP or BRDG:M	10.2.11	Yes [ ] N/A [ ]
MDFDPP	Does the device support media-dependent full-duplex, point-to-point functionality on one or more ports?	O.1	11.1	Yes [ ] No [ ]
MDDOT11	Does the device support media-dependent IEEE 802.11 link functionality on one or more ports?	O.1	12.1	Yes [ ] No [ ]
MDEPON	Does the device support IEEE 802.3 Passive Optical Networking (EPON)	O.1	13.1	Yes [ ] No [ ]
MDGHN	Does the device support media-dependent G.hn functionality on one or more ports?	O.1	Clause 16	Yes [ ] No [ ]
MDMOCA	Does the device support media-dependent MoCA functionality on one or more ports?	O.1	Clause 16	Yes [ ] No [ ]
MDCSN	Does the device support media-dependent CSN functionality on one or more ports?	MDGHN or MDMOCA:M	Clause 16	Yes [ ] No [ ]
MGT	Is management of the tim~~ing~~e synchronization in ~~Bridges~~time-aware relays supported?	O	Clause 15	Yes [ ] No [ ]

A.6 Media access control methods

Item	Feature	Status	References	Support
MAC-IEEE-802.3 MAC-IEEE-802.11	Which MAC methods are implemented in conformance with the relevant MAC standards	O:2 O:2	11.1 12.1	Yes [ ] No [ ] Yes [ ] No [ ]
MAC-1	Has a PICS been completed for each of the MAC methods implemented as required by the relevant MAC Standards?	M		Yes [ ]
MAC-2	Do all the MAC methods implemented support the MAC Timing aware Service as specified?	M	Clause 11 Clause 12 Clause 13	Yes [ ]

A.7 Minimal time-aware system

Item	Feature	Status	References	Support
MINTA-1	Does the device implement the functionality specified by the SiteSyncSync state machine in Figure 10-3 in compliance with the requirements of 10.2.6?	M	10.2.6	Yes [ ]
MINTA-2	Does the device implement the functionality specified by the PortSyncSyncReceive state machine in Figure 10-4 on each port in compliance with the requirements of 10.2.7?	M	10.2.7	Yes [ ]
MINTA-3	Does the device implement the functionality specified by the ClockSlaveSync state machine in Figure 10-9 in compliance with the requirements of 10.2.12?	M	10.2.12	Yes [ ]
MINTA-4	Does the device port sending a Signaling message that contains a message interval request TLV adjust its syncReceiptTimeoutTimeInterval in compliance with the requirements of 10.6.4.3.7 and Table 10-13?	SIG:M	10.6.4.3.7	Yes [ ] N/A [ ]
MINTA-5	Is the clockIdentity constructed in compliance with the requirements of 8.5.2.2, its subclauses, and Table 8-2?	M	8.5.2.2	Yes [ ]
MINTA-6	Is the domain number for all transmitted messages set to 0 in compliance with the requirements of 8.1?	M	8.1	Yes [ ]
MINTA-7	Is the IEEE 802.1AS time measured relative to the PTP epoch in compliance with the requirements of 8.2.2?	M	8.2.2	Yes [ ]
MINTA-8	If path delay asymmetry is modeled by this device does it comply with the requirements of 8.3?	O	8.3	Yes [ ] No [ ]
MINTA-9	Do all derived data types that are transmitted in IEEE 802.1AS messages and headers comply with 6.4.4?	M	6.4.4	Yes [ ]

Item	Feature	Status	References	Support
MINTA-10	Is the granularity of the local clock 40 ns or better in compliance with the requirements of B.1.2?	M	B.1.2	Yes [ ]
MINTA-11	Is the frequency of the local clock relative to TAI ±100 ppm in compliance with the requirements of B.1.1?	M	B.1.1	Yes [ ]
MINTA-12	Does the time-aware system ignore TLVs, of Announce and Signaling messages, that it cannot parse and attempt to parse the next TLV, in compliance with the requirements of 10.6.1?	M	10.6.1	Yes [ ]

A.8 Signal~~l~~ing

Item	Feature	Status	References	Support
SIG-1	Do the sequence numbers of Signaling messages comply with the requirements of 10.5.7?	SIG:M	10.5.7	Yes [ ]
SIG-2	Does the Signaling message body comply with the requirements of 10.6.4.1 and Table 10-10?	SIG:M	10.6.4.1	Yes [ ]
SIG-3	Does the Signaling message header comply with the requirements of Table 10-4 and 10.6.2.1, including all of its subclauses (10.6.2.2.1–10.6.2.2.14)?	SIG:M	10.6.2.1	Yes [ ]
SIG-4	Are all Signaling message reserved fields equal to 0 in compliance with the requirements of 10.6.1?	SIG:M	10.6.1	Yes [ ]
SIG-5	Is the destination MAC address for all Signaling messages equal to 01:80:C2:00:00:0E in compliance with the requirements of 10.5.3?	SIG:M	10.5.3	Yes [ ]
SIG-6	Is the ethertype for all Signaling messages equal to 0x88F7 in compliance with the requirements of 10.5.4?	SIG:M	10.5.4	Yes [ ]
SIG-7	Does the message interval request TLV for signaling messages comply with the requirements in 10.6.4.3.2 through 10.6.4.3.9 and Table 10-11?	SIG:M	10.6.4.3.2	Yes [ ]

A.9 Best master clock

Item	Feature	Status	References	Support
BMC-1	Does the device implement the functionality specified by the PortAnnounceReceive state machine in Figure 10-12 on each port in compliance with the requirements of 10.3.10?	M	10.3.10	Yes [ ]
BMC-2	Does the device implement the functionality specified by the PortAnnounceInformation state machine in Figure 10-13 on each port in compliance with the requirements of ?	M	10.3.11	Yes [ ]
BMC-3	Does the device implement the functionality specified by the Port~~Role~~StateSelection state machine in Figure 10-14 on each port in compliance with the requirements of 10.3.12? NOTE—There is one instance of the PortStateSelection state machine for the time-aware system, for each gPTP domain. Some of the PortStateSelection state machine computations are performed for each port, and some of the computations are performed for the time-aware system as a whole (and all the computations are performed for each gPTP domain).	M	10.3.12	Yes [ ]
BMC-4	If the value of clockA's SystemIdentity is less than that of clockB, is clockA selected as Grandmaster in compliance with the requirements of 10.3.2?	M	10.3.2	Yes [ ]
BMC-5	Does the value of priority1 comply with the requirements of 8.6.2.1?	M	8.6.2.1	Yes [ ]
BMC-6	Does the value of clockClass comply with the requirements of 8.6.2.2?	M	8.6.2.2	Yes [ ]
BMC-7	Does the value of priority2 comply with the requirements of 8.6.2.5?	M	8.6.2.5	Yes [ ]
BMC-8	Does the value of clockAccuracy comply with requirements of 8.6.2.3?	M	8.6.2.3	Yes [ ]
BMC-9	Does the value of offsetScaledVariance comply with the requirements of 8.6.2.4?	M	8.6.2.4	Yes [ ]
BMC-10	Does the value of timeSource comply with requirements of 8.6.2.7 and Table 8-4?	M	8.6.2.7	Yes [ ]
BMC-11	Is the port number equal to 1 in compliance with the requirements of 8.5.2.3?	~BRDG:M	8.5.2.3	Yes [ ] N/A [ ]
BMC-12	Are the ports numbered 1 through N for each of N ports in compliance with the requirements of 8.5.2.3?	M	8.5.2.3	Yes [ ]
BMC-13	Does the clockIdentity field comply with the requirements of 8.5.2.2?	M	8.5.2.2	Yes [ ]
BMC-14	When no grandmaster capable device is available does the behavior of the device comply with the requirements of 10.2.12.2, i.e., the clockSlaveTime should be provided by the local clock?	M	10.2.12.2	Yes [ ]

Item	Feature	Status	References	Support
BMC-15	Does the value of announceReceiptTimeout comply with the requirements of 10.7.3.2?	M	10.7.3.2	Yes [ ]
BMC-16	Does the SlavePort remove the port from the BMC selection after announceReceiptTimeout expires in compliance with the requirements of 10.7.3.2?	M	10.7.3.2	Yes [ ]
BMC-17	Does the value of syncReceiptTimeout comply with the requirements of 10.7.3.1?	M	10.7.3.1	Yes [ ]
BMC-18	Does the SlavePort remove the port from the BMC selection after syncReceiptTimeout expires in compliance with 10.7.3.1?	M	10.7.3.1	Yes [ ]
BMC-19	Does the device port sending a message interval request signaling message adjust its announceReceiptTimeoutTimeInterval in compliance with the requirements of 10.6.4.3.8 and Table 10-14?	SIG:M	10.6.4.3.8	Yes [ ]
BMC-20	If the device implements the ClockSourceTime interface, does the value of lastGmPhaseChange comply with the requirements of 9.2.2 and 6.4.3.3?	O	9.2.2	Yes [ ]
BMC-21	Does the transmitted timing information comply with the requirements of 10.3.1?	GMCAP:M	10.3.1	Yes [ ]

A.10 Grandmaster-capable system

Item	Feature	Status	References	Support
GMCAP-1	Does the device implement the the functionality specified by the ClockMasterSyncSend state machine in compliance with the requirements of 10.2.8 and Figure 10-5?	GMCAP:M	10.2.8	Yes [ ]
GMCAP-2	Does the device implement the the functionality specified by the ClockMasterSyncOffset state machine in compliance with the requirements of 10.2.9 and Figure 10-6?	GMCAP:M	10.2.9	Yes [ ]
GMCAP-3	Does the device implement the the functionality specified by the ClockMasterSyncReceive state machine in compliance with the requirements of 10.2.10 and Figure 10-7?	GMCAP:M	10.2.10	Yes [ ]

A.11 Media-independent master

Item	Feature	Status	References	Support
MIMSTR-1	Does the device implement the functionality of the AnnounceIntervalSetting state machine in compliance with the requirements of 10.3.16 and Figure 10-18 on each port?	MIMSTR:M	10.3.16	Yes [ ]
MIMSTR-2	Does the device implement the functionality of the PortSyncSyncSend state machine in compliance with the requirements of 10.2.11 and Figure 10-8 on each port?	MIMSTR:M	10.2.11	Yes [ ]
MIMSTR-3	Does the device implement the functionality of the PortAnnounceTransmit state machine in compliance with the requirements of 10.3.15 and Figure 10-17 on each port?	MIMSTR:M	10.3.15	Yes [ ]
MIMSTR-4	Does the destination MAC address of all Announce messages equal 01:80:C2:00:00:0E?	MIMSTR:M	10.5.3	Yes [ ]
MIMSTR-5	Does the ethertype of all Announce messages equal 0x88F7?	MIMSTR:M	10.5.4	Yes [ ]
MIMSTR-6	Do the sequence numbers of Announce messages comply with the requirements of 10.5.7?	MIMSTR:M	10.5.7	Yes [ ]
MIMSTR-7	Does the Announce message header comply with Table 10-4 and 10.6.2.2, including all of its subclauses (10.6.2.2.1–10.6.2.2.14)?	MIMSTR:M	10.6.2.1	Yes [ ]
MIMSTR-8	Does the Announce message body comply with the requirements in 10.6.3.1 and Table 10-8?	MIMSTR:M	10.6.3.1	Yes [ ]
MIMSTR-9	Are all Announce message reserved fields equal to 0?	MIMSTR:M	10.6.1	Yes [ ]
MIMSTR-10	If it is not otherwise specified is the logAnnounceInterval equal to zero or within the allowed range?	MIMSTR:M	10.7.2.1	Yes [ ]
MIMSTR-11	Does the value of currentUtcOffset comply with the requirements of 8.2.3?	MIMSTR:M	8.2.3	Yes [ ]
MIMSTR-12	Do the values of the leap59, leap61, and currentUtcOffsetValid flags comply with the requirements of 10.3.8?	MIMSTR:M	10.3.8	Yes [ ]
MIMSTR-13	Does this device ensure that messages that traverse it or originate from it are not transmitted with VLAN tags in complaince with the requirements of 11.3.3?	MIMSTR:M	11.3.3	Yes [ ]
MIMSTR-14	Is the computation of cumulative rateRatio in accordance with 10.2.7.3?	MIMSTR:M	10.2.7.3	Yes [ ] N/A [ ]
MIMSTR-15	Does this port support the optional syncLocked mode, as specified in 10.2.11?	MIMSTR:O	10.2.11	Yes [ ] No [ ]

A.12 Media-independent performance requirements

Item	Feature	Status	References	Support
MIPERF-1	Does the device comply with the performance requirements of B.1?	MIPERF:M	B.1	Yes [ ]
MIPERF-2	Does the device comply with the performance requirements of B.2.4?	MIPERF:M	B.2.4	Yes [ ]

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

Item	Feature	Status	References	Support
MDFDPP-1	Does this port implement the functionality of the MDSyncReceiveSM state machine in compliance with the requirements of 11.2.13 and Figure 11-6?	MDFDPP:M	11.2.13	Yes [ ]
MDFDPP-2	Does this port implement the functionality of the MDSyncSendSM state machine in compliance with the requirements of 11.2.14 and Figure 11-7?	MIMSTR and MDFDPP:M	11.2.14	Yes [ ]
MDFDPP-3	Does this port implement the functionality of the MDPdelayRequest state machine in compliance with the requirements of 11.2.15 and Figure 11-8?	MDFDPP:M	11.2.17	Yes [ ]
MDFDPP-4	Does this port implement the functionality of the MDPdelayResponse state machine in compliance with the requirements of 11.2.18 and Figure 11-9?	MDFDPP:M	11.2.18	Yes [ ]
MDFDPP-5	Does this port implement the functionality of the ~~LinkDelay~~SyncIntervalSetting state machine in compliance with the requirements of 11.2.19 and Figure 11-10?	MDFDPP:M	11.2.19	Yes [ ]
MDFDPP-6~~5~~	Does this port implement the functionality of the LinkDelay~~Sync~~IntervalSetting state machine in compliance with the requirements of 11.2.20 and Figure 11-11~~Figure 11-11~~?	MDFDPP:M	11.2.20	Yes [ ]
MDFDPP-7~~5~~	Does this port implement the functionality of the ~~LinkDelaySyncInterval~~OneStepTxOperSetting state machine in compliance with the requirements of 11.2.21 and Figure 11-12~~Figure 11-10~~? **<<Editor's note: Should this be mandatory? Note that one-step support is optional. Also, should this be moved to follow the one step capability (MDFDPP-34?>>**	MDFDPP:M	11.2.21	Yes [ ]
MDFDPP-~~6~~8	Does this port timestamp Sync messages on ingress with respect to the LocalClock in compliance with 11.3.2.1 and 11.3.9?	MDFDPP:M	11.3.2.1	Yes [ ]
MDFDPP-~~7~~9	Does this port timestamp Sync messages on egress with respect to the LocalClock in compliance with the requirements of 11.3.2.1 and 11.3.9?	MIMSTR and MDFDPP:M	11.3.2.1	Yes [ ]
MDFDPP-~~8~~10	Does this port timestamp Pdelay_Req messages on ingress and egress with respect to the LocalClock in compliance with the requirements of 11.3.2.1 and 11.3.9?	MDFDPP:M	11.3.2.1	Yes [ ]
MDFDPP-~~9~~11	Does this port timestamp Pdelay_Resp messages on ingress and egress with respect to the LocalClock in compliance with the requirements of 11.3.2.1 and 11.3.9?	MDFDPP:M	11.3.2.1	Yes [ ]

Item	Feature	Status	References	Support
MDFDPP-1~~0~~2	Are all IEEE 802.1AS messages on this port sent without a Q-tag in compliance with the requirements of 11.3.3?	MDFDPP:M	11.3.3	Yes [ ]
MDFDPP-13~~1~~	Do all media-dependent messages transmitted on this port use a destination MAC address taken from Table 11-2 in compliance with the requirements of 11.3.4 [01-80-C2-00-00-0E]?	MDFDPP:M	11.3.4	Yes [ ]
MDFDPP-14~~2~~	Do all media-dependent messages transmitted on this port use a source MAC address that is assigned to that port in compliance with the requirements of 11.3.4?	MDFDPP:M	11.3.4	Yes [ ]
MDFDPP-15~~3~~	Do all media-dependent message tranmitted on this port us an ethertype specified in Table 11-3 [0x88F7]?	MDFDPP:M	11.3.5	Yes [ ]
MDFDPP-16~~4~~	Does the header of all the media-dependent messages on this port comply with the requirements of the subclauses of 11.4.2 and Table 10-4?	MDFDPP:M	11.4.2	Yes [ ] N/A [ ]
MDFDPP-17~~5~~	Does the body of Sync messages sent on this port comply with the requirements of 11.4.3,~~ and~~ Table 11-9, and Table 11-10?	MDFDPP:M	11.4.3	Yes [ ]
MDFDPP-18~~6~~	Does the body of Follow_Up messages sent on this port comply with the requirements of 11.4.4,~~ and~~ 6.4.3.3 (lastGmPhaseChange), and Table 11-11?	MDFDPP:M	11.4.4, 6.4.3.3	Yes [ ]
MDFDPP-19~~7~~	Does the body of Pdelay_Req messages sent on this port comply with the requirements of 11.4.5 and Table 11-13?	MDFDPP:M	11.4.5	Yes [ ]
MDFDPP-20~~18~~	Does the body of Pdelay_Resp messages sent on this port comply with the requirements of 11.4.6 and Table 11-14?	MDFDPP:M	11.4.6	Yes [ ]
MDFDPP-21~~19~~	Does the body of Pdelay_Resp_Follow_Up messages sent on this port comply with the requirements of 11.4.7 and Table 11-15?	MDFDPP:M	11.4.7	Yes [ ]
MDFDPP-22~~0~~	Are all reserved fields in media-dependent messages sent on this port set to 0 in compliance with the requirements of 11.4.1?	MDFDPP:M	11.4.1	Yes [ ]
MDFDPP-23~~1~~	Do the Sync message sequence numbers comply with the requirements of 11.3.8?	MIMSTR and MDFDPP:M	11.3.8	Yes [ ] N/A [ ]
MDFDPP-24~~2~~	Do the Pdelay_Req message sequence numbers comply with the requirements of 11.3.8?	MDFDPP:M	11.3.8	Yes [ ]
MDFDPP-25~~3~~	Does the Pdelay mean request transmission interval comply with the requirements of 11.5.2.2?	MDFDPP:M	11.5.2.2	Yes [ ]
MDFDPP-26~~4~~	Does the Sync mean transmission interval comply with the requirements of 11.5.2.3?	MDFDPP:M	11.5.2.3	Yes [ ]
MDFDPP-27~~5~~	Does the full-duplex, point-to-point media-dependent layer set the asCapable global variable in the media-independent PortSync entity in compliance with the requirements of 11.2.2?	MDFDPP:M	11.2.2	Yes [ ]

Item	Feature	Status	References	Support
MDFDPP-28~~6~~	Does the device's use of flow control comply with the requirements of 11.2.3 and 11.2.4?	MDFDPP:M	11.2.3, 11.2.4	Yes [ ]
MDFDPP-29~~7~~	Does the device consider the port to not be exchanging Pdelay messages when a valid response is not received in compliance with the requirements of 11.5.3?	MDFDPP:M	11.5.3	Yes [ ]
MDFDPP-30~~28~~	Does the time-aware system ignore TLVs, of PTP messages, that it cannot parse and attempt to parse the next TLV, in compliance with the requirements of 11.4.1?	MDFDPP:M	11.4.1	Yes [ ]
MDFDPP-31~~29~~	Does the time-aware system initialize neighborPropDelayThresh as specified in 11.2.2?	MDFDPP:M	11.2.2	Yes [ ]
MDFDPP-32	Does this port of the time-aware system support asymmetry measurement mode, as specified in 14.8.35, 14.8.35, 10.2.7, 10.3.11, 10.3.12, 10.3.15, 11.2.13, 11.2.14, and 11.2.15?	MDFDPP:O	14.6, 10.2, 10.3, 11.2	Yes [ ] No [ ]
MDFDPP-33	Does this port support the optional one-step capability on receive, as specified in 11.2.13?	MDFDPP:O	11.2.13	Yes [ ] No [ ]
MDFDPP-34	Does this port support the optional one-step capability on transmit, as specified in 11.2.14?	MDFDPP:O	11.2.14	Yes [ ] No [ ]
MDFDPP-35	Does this port support the optional propagation delay averaging, as specified in 11.2.17.3.4?	MDFDPP:O	11.2.17.3.4	Yes [ ] No [ ]

A.14 Media-dependent IEEE 802.11 link

Item	Feature	Status	References	Support
MDDOT11-1	Does the IEEE 802.11 MAC implement the master port functionality in compliance with the requirements of 12.5.1?	MDDOT11 and MIMSTR:M	12.5.1	Yes [ ]
MDDOT11-2	Does the IEEE 802.11 MAC implement the ~~master~~slave port functionality in compliance with the requirements of 12.5.2?	MDDOT11:M	12.5.2	Yes [ ]
MDDOT11-3	Does the IEEE 802.11 MAC determine the value of asCapable in compliance with the requirements of 12.4?	MDDOT11:M	12.4	Yes [ ]
MDDOT11-4	Does the IEEE 802.11 MAC determine the value of mean time interval between synchronization messages in compliance with the requirements of 12.7?	MDDOT11 and MIMSTR:M	12.7	Yes [ ]
MDDOT11-5	Does the 802.11 MAC support the use of the VendorSpecific information element of 12.5 to carry end-to-end link-independent timing information?	MDDOT11:M	12.5	Yes [ ]
MDDOT11-6	Does the 802.11 MAC implement Fine Timing Measurement as a master port	MDDOT11-1:O	12.5.1	Yes [ ] No [ ]
MDDOT11-7	Does the 802.11 MAC implement Fine Timing Measurement as a slave port	MDDOT11-2:O	12.5.2	Yes [ ] No [ ]

A.15 Media-dependent IEEE 802.3 EPON link

Item	Feature	Status	References	Support
MDEPON-1	Does the TIMESYNC message format comply with the requirements of 13.3 and Table 13-1?	MDEPON:M	13.3	Yes [ ]
MDEPON-2	Does the device implement the functionality specified by the requester state machine in compliance with the requirements of 13.8.1 and Figure 13-3?	MDEPON and MIMSTR:M	13.8.1.4	Yes [ ]
MDEPON-3	Does the device implement the functionality specified by the responder state machine in compliance with the requirements of 13.8.2 and Figure 13-4?	MDEPON:M	13.8.2.4	Yes [ ]
MDEPON-4	Does the TIMESYNC message transmission interval comply with the requirements of 13.9.1 and 13.9.2?	MDEPON:M	13.9.1, 13.9.2	Yes [ ]
MDEPON-5	Does the implementation of best master selection comply with the requirements of 13.1.3?	MDEPON:M	13.1.3	Yes [ ]
MDEPON-6	Does the determination of the value of asCapable comply with the requirements of 13.4?	MDEPON:M	13.4	Yes [ ]

A.16 Media-dependent CSN link

Item	Feature	Status	References	Support
MDCSN-1	Does the device implement the functionality of the MDSyncSendSM state machine in compliance with 11.2.14?	MDCSN and MIMSTR:M	11.2.14	Yes [ ]
MDCSN-2	Does the device implement the functionality of the MDSyncReceiveSM state machine in compliance with 11.2.13?	MDCSN:M	11.2.13	Yes [ ]
MDCSN-3	Does the device calculate path delay in compliance with the requirement of 16.4 and its subclauses?	MDCSN:M	16.4.1, 16.4.2, 16.4.3	Yes [ ]
MDCSN-4	Does the device propogate ~~synchronized time~~synchronized time in compliance with the requirements of 16.5 and its subclauses?	MDCSN:M	16.5.1, 16.5.2	Yes [ ]
MDCSN-5	Does the device act as grandmaster in compliance with the requirements of 16.7 and its subclauses?	GMCAP and MDCSN:M	16.7	Yes [ ]
MDCSN-6	Does the device comply with the performance requirements of ~~E.8~~16.8?	GMCAP and MDCSN:M	16.8	Yes [ ]

A.17 Media-dependent MoCA link

Item	Feature	Status	References	Support
MDMOCA-1	Does the MoCA MD entity propogate Sync messages in compliance with the requirements of 16.6.1?	MDMOCA:M	~~E.6.1~~16.6.1	Yes [ ]

A.18 Media-dependent ITU-T G.hn link

Item	Feature	Status	References	Support
MDGHN-1	Does the GHN MD entity propogate Sync messages in compliance with the requirements of 16.6.2?	MDGHN:M	~~E.6.2~~16.6.2	Yes [ ]

Annex B

(normative)

Performance requirements

<<Editor's note: The requirements of this annex apply separately to each gPTP domain. Since there is a general statement of this in 8.1, no additional text on this is needed here.>>

B.1 LocalClock requirements

B.1.1 Frequency accuracy

The fractional frequency offset of the LocalClock relative to the TAI frequency (see Annex C) shall be within ±100 ppm.

B.1.2 Time measurement granularity

The granularity with which the LocalClock measures time shall be less than or equal to 40 ns.

B.1.3 Noise generation

B.1.3.1 Jitter generation

The jitter generation of the free-running LocalClock shall not exceed 2 ns peak-to-peak, when measured over a 60 s measurement interval using a band-pass filter that consists of the following low-pass and high-pass filters:

a) High-pass filter: first-order characteristic (i.e., 0 dB gain peaking), 20 dB/decade roll-off, and 3 dB bandwidth (i.e., corner frequency) of 10 Hz
b) Low-pass filter: maximally-flat (i.e., Butterworth) characteristic, 60 dB/decade roll-off, and 3 dB bandwidth equal to the Nyquist rate of the LocalClock entity (i.e., one-half the nominal frequency of the LocalClock entity)

B.1.3.2 Wander generation

Wander generation is specified using the Time Deviation (TDEV) parameter. The corresponding values of the Allan Deviation (ADEV) and PTP Deviation (PTPDEV) are given for information; the former is also useful in describing the wander generation of clocks and oscillators, and the latter is related to the offsetScaledLogVariance attribute (see 8.6.2.4). Information on ADEV and TDEV are contained in ITU-T G.810 [B13] and IEEE Std 1139™-1999 [B6]. Information on ADEV and PTPDEV are contained in 7.6.3 of IEEE Std 1588-2008.

TDEV, denoted $\sigma_x(\tau)$, is estimated from a set of measurements, as shown in Equation (B.1):

$$\sigma_x(\tau) = \sqrt{\frac{1}{6n^2(N-3n+1)} \sum_{j=1}^{N-3n+1} \left[\sum_{i=j}^{n+j-1} (x_{i+2n} - 2x_{i+n} + x_i) \right]^2}, n = 1, 2, \ldots \left\lfloor \frac{N}{3} \right\rfloor \quad \text{(B.1)}$$

where

$\tau = n\tau_0$ = observation interval
τ_0 = sampling interval
N = total number of samples [$(N-1)\tau_0$ = measurement interval]
$\lfloor y \rfloor$ denotes the floor function, i.e., the greatest integer less than or equal to y
x_i = measured phase (time) error at the i^{th} sampling time [the units of x_i and $\sigma_x(\tau)$ are the same]

ADEV, denoted $\sigma_y(\tau)$, is estimated from a set of measurements, as shown in Equation (B.2):

$$\sigma_y(\tau) = \sqrt{\frac{1}{2n^2\tau_0^2(N-2n)}\sum_{i=1}^{N-2n}(x_{i+2n}-2x_{i+n}+x_i)^2}, n = 1, 2, \ldots\left\lfloor\frac{N-1}{2}\right\rfloor \tag{B.2}$$

where the notation is the same as defined above for TDEV.

PTPDEV, denoted $\sigma_{PTP}(\tau)$, is estimated from a set of measurements, as shown in Equation (B.3):

$$\sigma_{PTP}(\tau) = \sqrt{\frac{1}{6(N-2n)}\sum_{i=1}^{N-2n}(x_{i+2n}-2x_{i+n}+x_i)^2}, n = 1, 2, \ldots\left\lfloor\frac{N-1}{2}\right\rfloor \tag{B.3}$$

where the notation is the same as defined above for TDEV.

TDEV, ADEV, and PTPDEV are second-order statistics on the phase error. All three statistics are functions of second differences of the phase error. This means that these statistics are not affected by a constant frequency offset. This behavior is desired, because these statistics are used here to constrain noise generation.

TDEV for the LocalClock entity shall not exceed the mask of Table B.1 and Figure B.1, when measured using

a) A measurement interval that is at least 120 s (i.e., at least 12 times the longest observation interval),
b) A low-pass filter with 3 dB bandwidth of 10 Hz, first-order characteristic, and 20 dB/decade roll-off, and
c) A sampling interval τ_0 that does not exceed 1/30 s.

Table B.1—Wander generation TDEV requirement for LocalClock entity

TDEV limit	Observation interval τ
No requirement	$\tau < 0.05$ s
5.0τ ns	$0.05 \le \tau \le 10$ s
No requirement	$\tau > 10$ s

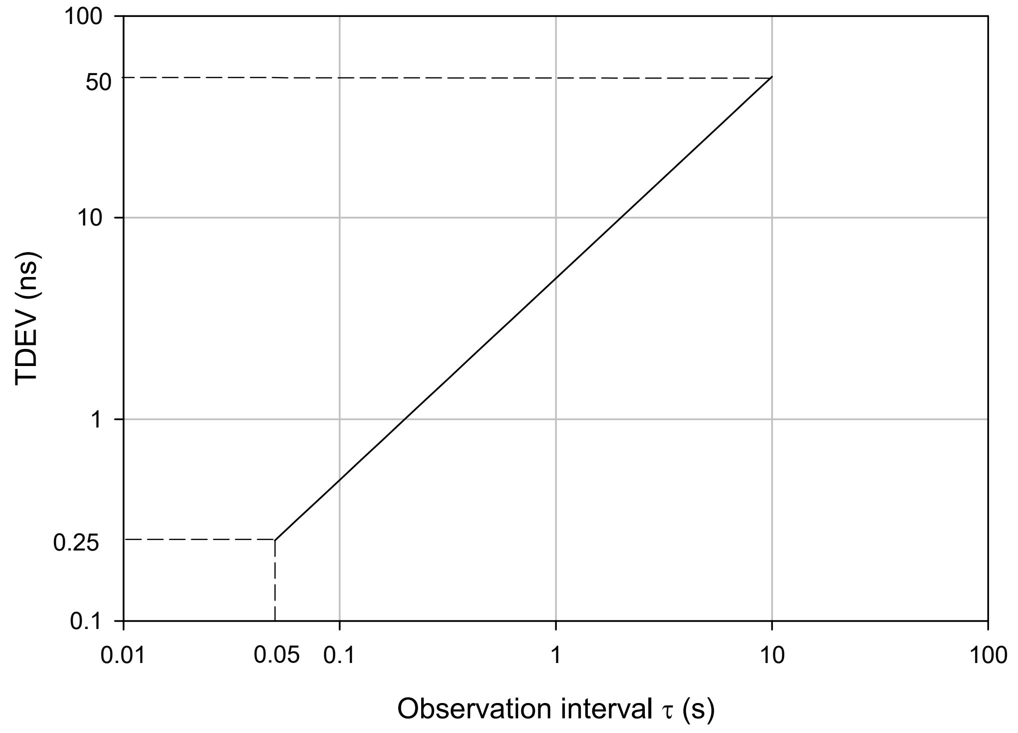


Figure B.1—Wander generation (TDEV) requirement for LocalClock entity

The ADEV limit that corresponds to the TDEV requirement of Table B.1 and Figure B.1 is shown in Table B.2 and Figure B-2, respectively.

Table B.2—ADEV limit corresponding to wander generation requirement of Table B.1

ADEV limit	Observation interval τ
No requirement	$\tau < 0.05$ s
1.054×10^{-8}	$0.05 \le \tau \le 10$ s
No requirement	$\tau > 10$ s

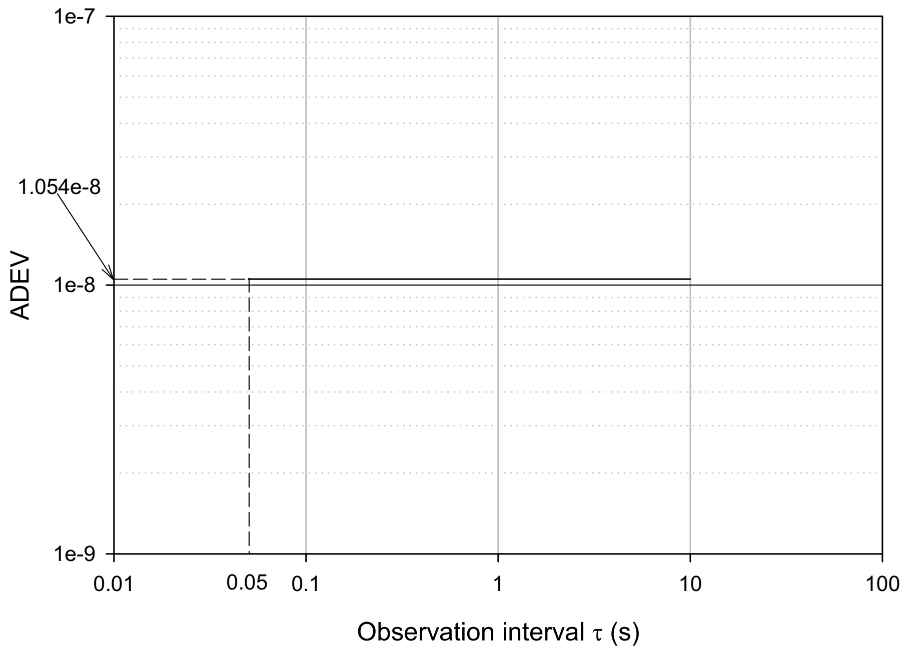


Figure B-2—ADEV limit corresponding to wander generation requirement of Figure B.1

The PTPDEV limit that corresponds to the TDEV requirement of Table B.1 and Figure B.1 is shown in Table B.3 and Figure B.3, respect, respectively.

Table B.3—PTPDEV limit corresponding to wander generation requirement of Table B.1

PTPDEV limit	Observation interval τ
No requirement	$\tau < 0.05$ s
6.08τ ns	$0.05 \le \tau \le 10$ s
No requirement	$\tau > 10$ s

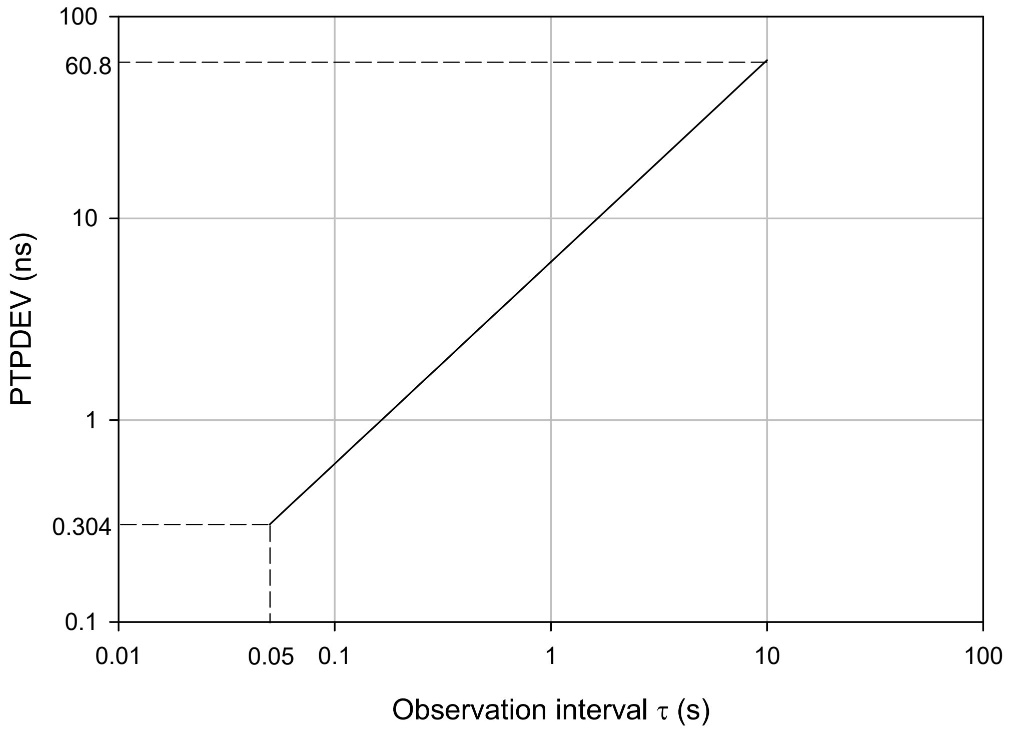


Figure B.3—PTPDEV limit corresponding to wander generation requirement of Figure B.1

B.2 Time-aware system requirements

B.2.1 General

In order to achieve the accuracy goals, certain constraints are placed on the responsiveness and accuracy of time-aware systems.

B.2.2 Residence time

The residence time (see 3.19) of a time-aware system, measured relative to the TAI frequency (see 8.2), should be less than or equal to 10 ms.

Any error in the measured frequency offset relative to the grandmaster (i.e., error in accumulated rateRatio) results in an error in the transported ~~synchronized time~~synchronized time that is equal to the frequency offset error multiplied by the residence time (see 7.3.3 and 11.1.3).

B.2.3 Pdelay turnaround time

The pdelay turnaround time is the duration of the interval between the receipt of a Pdelay_Req message by a port of a time-aware system, and the sending of the corresponding Pdelay_Resp message.

The pdelay turnaround time of a time-aware system, measured relative to the TAI frequency (see 8.2), should be less than or equal to 10 ms.

A non-zero pdelay turnaround time and any error in the measured frequency offset between the peer delay initiator and peer delay respondor (i.e., error in neighborRateRatio) results in an error in the measured mean propagation delay. This in turn results in an error in the transported ~~synchronized time~~synchronized time. See 11.1.2 for more details.

NOTE—While a larger value of pdelay turnaround time can result in worse ~~time synchronization~~time-synchronization performance, the peer delay protocol will still operate as long as the peer delay initiator receives Pdelay_Resp and Pdelay_Resp_Follow_Up within a time interval since sending Pdelay_Req that is less than the current Pdelay_Req message transmission interval (see 11.2.17.4 and 11.5.2.2).

B.2.4 Measurement of rate ratio

This standard requires the measurement of rate ratio or, equivalently, frequency offset, in several subclauses (see 10.2.9, 10.2.10, 11.2.15, 12.5.2, 16.4.2, and 16.4.3.1). The error inherent in any scheme used to measure rate ratio shall not exceed ±0.1 ppm.

NOTE—This requirement is consistent with a rate ratio measurement made by measuring the default Pdelay_Req message transmission interval (the nominal interval duration is 1 s, see 11.5.2.2) relative to the clocks whose rate ratio is desired, assuming the clocks meet the time measurement granularity requirement of B.1.2 (i.e., no worse than 40 ns).

B.3 End-to-end time-synchronization performance

The requirements of this standard and of standards referenced for each medium ensure that any two time-aware systems separated by six or fewer time-aware systems (i.e., seven or fewer hops) will be synchronized to within 1 µs peak-to-peak of each other during steady-state operation (i.e., each time-aware system receives time-synchronization information every sync interval).

B.4 End-to-end jitter and wander performance

The requirements of this standard and standards referenced by this standard ensure that the ~~synchronized time~~synchronized time at a time-aware system that is separated from the grandmaster by six or fewer time-aware systems (i.e., seven or fewer hops) will, when filtered by a reference endpoint filter~~s~~ with rolloff of 20 dB/decade, gain peaking that does not exceed 0.1 dB, and bandwidth that does not exceed the value given in each entry of Table B.4, have maximum time interval error (MTIE) that does not exceed the maximum time interval error (MTIE) for that entry of Table B.4, and jitter that does not exceed the peak-to-peak jitter of Table B.4 when measured through the corresponding high-pass jitter measurement filter given in Table B.4.

NOTE—For example, the endpoint filter can be of the following form:

$$y_k = a_1 y_{k-1} + a_2 y_{k-2} + \dots + a_n y_{k-n} + b_0 x_k + b_1 x_{k-1} + \dots + b_n x_{k-n}$$

where the x_k are the unfiltered ~~synchronized time~~synchronized time values, the y_k are the filtered ~~synchronized time~~synchronized time values, and the a_k and b_k are filter coefficients. The a_k and b_k are chosen such that the filter has desired bandwidth and gain peaking that does not exceed 0.1 dB. The preceding equation is a general infinite impulse response (IIR) digital filter. Simplified forms, e.g., a second order IIR filter obtained by setting $n = 2$, or a finite impulse response (FIR) filter obtained by setting the a_k to zero are possible.

Table B.4—Maximum endpoint filter bandwidths needed to meet respective MTIE masks and peak-to-peak jitter limits

Endpoint filter maximum bandwidth (Hz)	Corresponding MTIE mask of Figure B.4 that is not exceeded	Corresponding jitter high-pass measurement filter (Hz)	Corresponding peak-to-peak jitter that~~n~~ is not exceeded (ns)
10	Mask 2 (Figure B.4,Table B.6)	8000	10.2
1	Mask 1 (Figure B.4,Table B.5)	200	11.1
0.01	Mask 3 (Figure B.4,Table B.7)	—	—

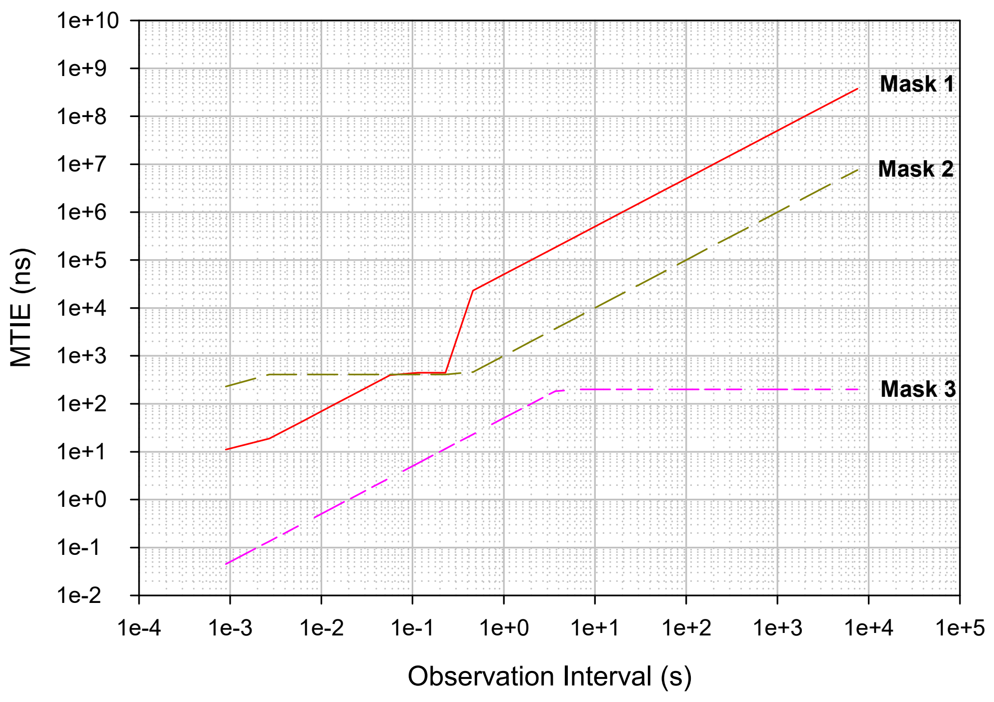


Figure B.4—MTIE masks met for maximum endpoint filter bandwidths of Table B.4

Table B.5—Breakpoints for Mask 1

Observation interval S (s)	MTIE (ns)
$0.05 \leq S < 0.0637$	6954.8S
$0.0637 \leq S < 0.3183$	443
$0.3183 \leq S \leq 10000$	50000S

Table B.6—Breakpoints for Mask 2

Observation interval S (s)	MTIE (ns)
$0.05 \le S < 0.4069$	407
$0.4069 \le S < 10000$	1000S

Table B.7—Breakpoints for Mask 3

Observation interval S (s)	MTIE (ns)
$6.67 \times 10^{-4} \le S < 4.0$	50S
$4.0 \le S < 10000$	200

Annex C

(informative)

Timescales and epochs

C.1 Overview

A more detailed discussion of many of the topics in this annex can be found in Allan, Ashby, and Hodge [B1].

For historical reasons, time is specified in a variety of ways as listed in Table C.1. GPS, PTP, and TAI times are based on values yielded by atomic clocks and advance on each second. NTP and UTC times are similar, but are occasionally adjusted by one leap second, to account for differences between the atomic clocks and the rotation time of the earth.

Table C.1—Timescale parameters

	Timescale				
Parameter	**GPS**	**PTP**	**TAI**	**NTP**	**UTC**
approximate epoch†	1980-01-06 1999-08-22	1970-01-01	no epoch defined	1900-01-01	no epoch defined
representation	weeks.seconds	seconds	YYYY-MM-DD hh:mm:ss	seconds	YYYY-MM-DD hh:mm:ss
rollover (years)	19.7	≈8 900 000	10 000	≈136	10 000
leapSeconds	no			yes	
NOTE 1—After 1972-01-01 00:00:00 TAI, TAI and UTC differ by only integer seconds. NOTE 2—The following represent the same instant in time: 1958-01-01 00:00:00 TAI and 1958-01-01 00:00:00 UTC					

Legend:

†	Each approximate epoch occurs at 00:00:00 on the respective date
GPS	global positioning satellite
NTP	Network Time Protocol
PTP	Precision Time Protocol
TAI	International Atomic Time (from the French term *Temps Atomique International*)
UTC	Coordinated Universal Time (the acronym is a compromise between the English term *coordinated universal time* and the French term *temps universel coordonné* English speakers wanted the initials of their language: CUT for *coordinated universal time* French speakers wanted the initials of their language: TUC for *temps universel coordonné*

C.2 TAI and UTC

TAI and UTC are international standards for time based on the SI second as realized on the rotating geoid; (see ITU-R TF.460-6 [B12] and *The International System of Units (SI)* [B18] for the definitions of TAI and UTC, *The International System of Units (SI)* [B18] for the SI second, and Jekeli [B14] for more information on the rotating geoid). TAI is implemented by a suite of atomic clocks and forms the timekeeping basis for other timescales in common use. The rate at which UTC time advances is normally identical to the rate of TAI. An exception is an occasion when UTC is modified by adding or subtracting exactly one whole leap second.

The TAI and UTC timescales were introduced as of 1958-01-01, and the times 1958-01-01 00:00:00 TAI and 1958-01-01 00:00:00 UTC represent the same instant in time. However, this instant in time is not an epoch for TAI and UTC. An epoch, in the sense of the origin of a timescale (see 8.2.2), is not defined for either TAI or UTC. TAI and UTC are expressed in the form YYYY-MM-DD hh:mm:ss, rather than as elapsed time since an epoch; here, YYYY-MM-DD denotes the date and hh:mm:ss denotes the time in each day.

Prior to 1972-01-01, corrections to the offset between UTC and TAI were made by applying fractional-second corrections to UTC and corrections to the rate at which UTC advanced relative to the rate at which TAI advanced. After 1972-01-01, leap-second corrections are applied to UTC by inserting or deleting second(s) at the end of the last minute of preferably the last day of June or December. Also after 1972-01-01, UTC and TAI advance at the same rate. As of 2006-01-01, TAI and UTC times differed by +33 s (i.e., TAI time minus UTC time equals +33 s for 2006-01-01).

In POSIX-based computer systems, the common time conversion algorithms can produce the correct ISO 8601:2004 [B8] printed representation format "YYYY-MM-DD hh:mm:ss" for both TAI and UTC. ISO 8601:2004 specifies that the seconds of each minute are numbered from 00 to 59. A leap second added to the end of a minute is numbered 60. A second is deleted from the end of a minute by deleting the second labeled 59.

The PTP epoch is set such that a direct application of the POSIX algorithm to a PTP timescale timestamp yields the ISO 8601:2004 printed representation of TAI. Subtracting the current *leapSeconds* value from a timestamp prior to applying the POSIX algorithm yields the ISO 8601:2004 printed representation of UTC. Conversely, applying the inverse POSIX algorithm and adding *leapSeconds* converts from the ISO 8601:2004 printed form of UTC to the form convenient for generating a timestamp.

Example: The POSIX algorithm applied to a timestamp value of 8 seconds yields 1970-01-01 00:00:08 (8 s after midnight on 1970-01-01 TAI). At this time the value of *leapSeconds* was approximately 8 s. Subtracting this 8 s from this time yields 1970-01-01 00:00:00 UTC.

Example: The POSIX algorithm applied to a timestamp value of 0 s yields 1970-01-01 00:00:00 TAI. At this time the value of *leapSeconds* was approximately 8 s. Subtracting this 8 s from this time yields 1969-12-31 23:59:52 UTC.

C.3 NTP and GPS

Two standard time sources of particular interest in implementing PTP systems: NTP and GPS. Both NTP and GPS systems are expected to provide time references for calibration of the grand-master supplied PTP time.

NTP represents seconds as a 32-bit unsigned integer that rolls-over every 2^{32} s $\approx$ 136 year, with the first such rollover occurring in the year 2036. The precision of NTP systems is usually in the millisecond range.

NTP is a widely used protocol for synchronizing computer systems. NTP is based on sets of servers, to which NTP clients synchronize. These servers themselves are synchronized to time servers that are traceable to international standards.

NTP provides the current time in NTP version 4, the current *leapSeconds* value, and warning flags indicating that a *leapSecond* will be inserted at the end of the current UTC day. The NTP clock effectively stops for one second when the leap second is inserted.

GPS time comes from a global positioning satellite system, GPS, maintained by the U.S. Department of Defense. The precision of GPS system is usually in the 10 ns to 100 ns range. GPS system transmissions represent the time as {*weeks*, *secondsInWeek*}, the number of weeks since the GPS epoch and the number of seconds since the beginning of the current week.

GPS also provides the current *leapSeconds* value, and warning flags marking the introduction of a leap second correction. UTC and TAI times can be computed solely based the information contained in the GPS transmissions.

GPS timing receivers generally manage the epoch transitions (1024-week rollovers), providing the correct time (YYYY-MM-DD hh:mm:ss) in TAI and/or UTC timescales, and often also local time; in addition to providing the raw GPS week, second of week, and leap-second information.

C.4 Timescale conversions

Previously discussed representations of time can be readily converted to/from PTP *time* based on a constant offset and the distributed *leapSeconds* value, as specified in Table C.2. Within Table C.2, all variables represent integers; "/" and "%" represent a integer divide and remainder operation, respectively.

Table C.2—Timescale conversions

ta		PTP value tb
Name	**Format**	
GPS	weeks:seconds	tb = ta.seconds + 315 964 819 + (gpsRollovers * 1024 + ta.weeks) * (7 * DAYSECS);

		ta.weeks = (tb – 315 964 819) / (7 * DAYSECS) – gpsRollovers*1024;
		ta.seconds = (tb – 315 964 819) % (7 * DAYSECS);

TAI	date{YYYY,MM,DD}:time{hh,mm,ss}	tb = DateToDays("1970-01-01", ta.date) * DAYSECS + ((ta.time.hh * 24) + ta.time.mm) *60) + ta.time.ss;

		secs = tb % DAYSECS;
		ta.date = DaysToDate("1970-01-01", tb / DAYSECS);
		ta.time.hh = secs / 3600;
		ta.time.mm = (secs % 3600)/60;
		ta.time.ss = (secs % 60);

NTP	seconds	tb = (ta+leapSeconds)–2208988800;
		ta = (tb–leapSeconds)+2208988800;
UTC	date{YYYY,MM,DD}:time{hh,mm,ss}	tb = DateToDays("1970-01-01", ta.date) * DAYSECS + ((ta.time.hh * 24) + ta.time.mm) *60) + ta.time.ss + leapSeconds;

		tc = tb – leapSeconds;
		secs = tc % DAYSECS;
		ta.date = DaysToDate("1970-01-01", tc/DAYSECS);
		ta.time.hh = secs / 3600;
		ta.time.mm = (secs % 3600)/60;
		ta.time.ss = (secs % 60);

NOTE—*gpsRollovers*	Currently equals 1; changed from 0 to 1 between 1999-08-15 and 1999-08-22.
DAYSECS	The number of seconds within a day: (60*60*24).
leapSeconds	Extra seconds to account for variations in the earth-rotation times: 33 on 2006-01-01.
DateToDays	For arguments DateToDays(*past*, *present*), returns days between *past* and *present* dates.
DaysToDate	For arguments DaysToDate(*past*, *days*), returns the current date, *days* after the *past* date.

C.5 Time zones and GMT

The term Greenwich Mean Time (GMT) once referred to mean solar time at the Royal Observatory in Greenwich, England. GMT now commonly refers to the timescale UTC; or the UK winter time zone (Western European Time, WET). Such GMT references are, strictly speaking, incorrect but nevertheless quite common. The following representations correspond to the same instant of time:

18:07:00 (GMT), commonplace usage
18:07:00 (UTC)
18:07:00 (Western European Time, WET)
06:07 PM (Western European Time, WET)
13:07:00 (Eastern Standard Time, EST)
01:07 PM (Eastern Standard Time, EST)
10:07:00 (Pacific Standard Time, PST)
10:07 AM (Pacific Standard Time, PST)

Annex D

(normative)

State diagram notation

State diagrams are used to represent the operation of the protocol by a number of cooperating state machines each comprising a group of connected, mutually exclusive states. Only one state of each machine can be active at any given time. Each state is represented in the state diagram as a rectangular box, divided into two parts by a horizontal line. The upper part contains the state identifier, written in upper case letters. The lower part, the state block, contains procedures that are executed on entry to the state.

All permissible transitions between states are represented by arrows, the arrowhead denoting the direction of the possible transition. Labels attached to arrows denote the condition(s) that must be met in order for the transition to take place. All conditions are expressions that evaluate to TRUE or FALSE; if a condition evaluates to TRUE, then the condition is met. The label UCT denotes an unconditional transition (i.e., UCT always evaluates to TRUE). A transition that is global in nature (i.e., a transition that occurs from any of the possible states if the condition attached to the arrow is met) is denoted by an open arrow (i.e., no specific state is identified as the origin of the transition). When the condition associated with a global transition is met, it supersedes all other exit conditions including UCT. The special global condition BEGIN supersedes all other global conditions. Once BEGIN is asserted it remains asserted, and completion of each state block is followed by reentry to that state, until all state blocks have executed to the point that variable assignments and other consequences of their execution remain unchanged.

On entry to a state, the procedures defined for the state (if any) are executed exactly once, in the order that they appear on the page. Each action is deemed to be atomic, i.e., execution of a procedure completes before the next sequential procedure starts to execute. The procedures in only one state block execute at a time, even if the conditions for execution of state blocks in different state machines are satisfied, and all procedures in an executing state block complete execution before the transition to and execution of any other state block occurs, i.e., the execution of any state block appears to be atomic with respect to the execution of any other state block and the transition condition to that state from the previous state is TRUE when execution commences. The order of execution of state blocks in different state machines is undefined except as constrained by their transition conditions. A variable that is set to a particular value in a state block retains that value until a subsequent state block modifies the value.

On completion of all of the procedures within a state, all exit conditions for the state (including all conditions associated with global transitions) are evaluated continuously until one of the conditions is met. The label ELSE denotes a transition that occurs if none of the other conditions for transitions from the state are met (i.e., ELSE evaluates to TRUE if all other possible exit conditions from the state evaluate to FALSE). Where two or more exit conditions with the same level of precedence become TRUE simultaneously, the choice as to which exit condition causes the state transition to take place is arbitrary. A UCT has the lowest level of precedence.

Where it is necessary to split a state machine description across more than one diagram, a transition between two states that appear on different diagrams is represented by an exit arrow drawn with dashed lines, plus a reference to the diagram that contains the destination state. Similarly, dashed arrows and a dashed state box are used on the destination diagram to show the transition to the destination state. In a state machine that has been split in this way, any global transitions that can cause entry to states defined in one of the diagrams are deemed to be potential exit conditions for all of the states of the state machine, regardless of which diagram the state boxes appear in.

Should a conflict exist between the interpretation of a state diagram and the textual description associated with the state machine, the state diagram takes precedence. The interpretation of the special symbols and operators used in the state diagrams is as defined in Table D.1; these symbols and operators are derived from the notation of the “C++” programming language, ISO/IEC 14882:2003 [B11]. If a Boolean variable is described in this clause as being set, it has or is assigned the value TRUE; if reset or clear, the value FALSE.

Table D.1—State machine symbols

Symbol	Interpretation
( )	Used to force the precedence of operators in Boolean expressions and to delimit the argument(s) of actions within state boxes.
;	Used as a terminating delimiter for actions within state boxes. Where a state box contains multiple actions, the order of execution follows the normal English language conventions for reading text.
=	Assignment action. The value of the expression to the right of the operator is assigned to the variable to the left of the operator. Where this operator is used to define multiple assignments, e.g., a = b = X the action causes the value of the expression following the right-most assignment operator to be assigned to all of the variables that appear to the left of the right-most assignment operator.
!	Logical NOT operator.
&&	Logical AND operator.
||	Logical OR operator.
if...then...	Conditional action. If the Boolean expression following the **if** evaluates to TRUE, then the action following the **then** is executed.
{statement 1, ... statement N}	Compound statement. Braces are used to group statements that are executed together as if they were a single statement.
!=	Inequality. Evaluates to TRUE if the expression to the left of the operator is not equal in value to the expression to the right.
==	Equality. Evaluates to TRUE if the expression to the left of the operator is equal in value to the expression to the right.
<	Less than. Evaluates to TRUE if the value of the expression to the left of the operator is less than the value of the expression to the right.
>	Greater than. Evaluates to TRUE if the value of the expression to the left of the operator is greater than the value of the expression to the right.
>=	Greater than or equal to. Evaluates to TRUE if the value of the expression to the left of the operator is either greater than or equal to the value of the expression to the right.
+	Arithmetic addition operator.
–	Arithmetic subtraction operator.

Annex E

(normative)

Annex E is now Clause 16

~~Media-dependent layer specification for CSN Network~~

~~<<Editor's note: Annex E will be renumbered as Clause 16, so that it follows the last numbered clause (Clause 15). All references to the current Annex E will be updated, including references in the MIB (i.e., in 15.5). A placeholder for Annex E will be retained, with the following text just below "(informative)":~~

~~Annex E is now Clause 16.~~

~~Note that the title and all current text will be in Clause 16 after this change.~~

~~Note that, after this change, Clause 16 will be normative and Annex E will be informative.>>~~

~~<<Editor's note: References to the MIB in Annex E are on the following pages of IEEE 802.1AS/Cor 1-2013: pages 71, 72, 73 (2 places), 81, and 82.>>~~

~~<<Editor's note: On p.vi of IEEE Std 802.1AS-2011, Philippe Klein is listed as the clause editor of Annex F. This should be Annex E, and should be changed to Clause 16 after the references to Annex E are updated.>>~~

E.1 ~~Overview~~

~~Accurate synchronized time is distributed throughout a gPTP domain through time measurements between adjacent time-aware nodes or end stations in a bridged LAN. Time is communicated from the root of the clock spanning tree (i.e., the grandmaster) toward the leaves of the tree (i.e., from leaf-facing "master" ports to root-facing "slave" ports) through measurements made across the links connecting the time-aware systems. While the semantics of time transfer are consistent across the time-aware bridged LAN, the method for communicating synchronized time from a master station to its immediate downstream link partner varies depending on the type of link interconnecting the two time-aware systems.~~

~~This appendix specifies the protocol that provides accurate synchronized time across links of a *coordinated shared network* (CSN) as part of a bridged LAN.~~

E.2 ~~Coordinated Shared Network characteristics~~

~~A CSN is a contention-free, time-division, multiplexed-access network of devices sharing a common medium and supporting reserved bandwidth based on priority or flow (QoS). One of the nodes of the CSN acts as the network coordinator, granting transmission opportunities to the other nodes of the network. A CSN network physically is a shared medium, in that a CSN node has a single physical port connected to the half-duplex medium, but is logically a fully connected one-hop mesh network, in that every node can transmit frames to every other node over the shared medium.~~

~~A CSN supports two types of transmission: unicast transmission for point-to-point (node-to-node) transmission and multicast/broadcast transmission for point-to-multipoint (node-to-other/all-nodes) transmission. Figure E.1 illustrates a CSN network acting as a backbone for time-aware systems.~~

~~NOTE—In this annex, the term *node* is used to refer to a CSN node (i.e., it does not refer to a time-aware Bridge or end station). A CSN node is a 2-port Bridge that forwards data packets between a segment external to the CSN (which may connect to an upstream or downstream time-aware system or Bridge) and the CSN network, all at the data link layer.~~

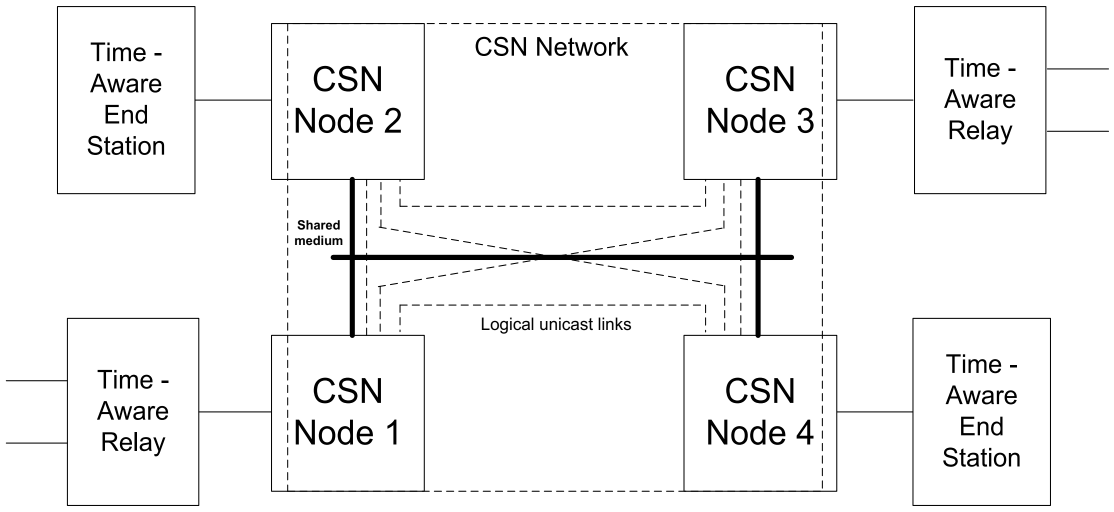


Figure E.1—~~Example of CSN backbone in an AVB LAN~~

E.3 ~~Layering for CSN links~~

~~One PortSync entity and one MD entity are together associated with each CSN logical port (node-to-node link) as illustrated in Figure E.2. The PortSync entities is described in 10.1.1. The MD entity translates media-independent primitives to MD primitives as necessary for communicating synchronized time over the CSN links. The CSN MD entity shall implement the MDSyncSendSM and MDSyncReceiveSM states machines of 11.2.13 and 11.2.14.~~

~~The CSN MD entity either implements the MDPdelayReq and MDPdelayResp state machines of 11.2.15 and to measure the propagation delay on a CSN link, or measures it through a CSN-native method and populates the variables described in E.4.3.2.~~

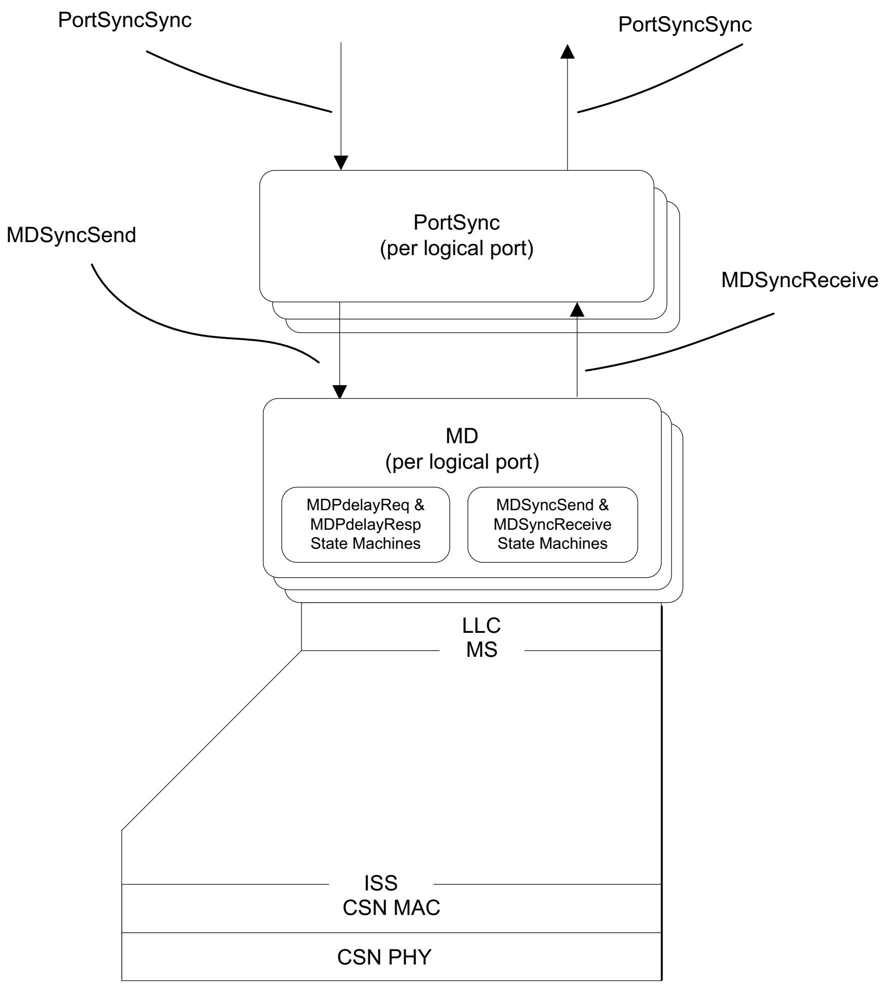


Figure E.2—~~Media-dependent and lower entities in CSN nodes~~

E.4 ~~Path delay measurement over a CSN backbone~~

E.4.1 ~~General~~

~~The Path Delay over a CSN backbone is calculated for the following path types: (1) between the upstream time-aware Bridge and the ingress CSN node, (2) between the ingress and egress CSN nodes, and (3) between the egress CSN node and the downstream time-aware system (Bridge or end station).~~

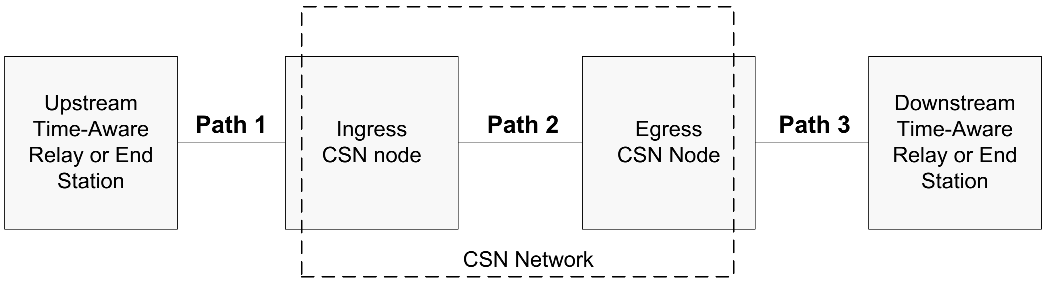

Upstream Time-Aware Relay or End Station
Path 1
Ingress CSN node
Path 2
Egress CSN Node
Path 3
Downstream Time-Aware Relay or End Station
CSN Network

Figure E.3—~~Path types over CSN as IEEE 802.1AS backbone~~

~~To maintain the synchronization, residence time on each node and the propagation delay between nodes is measured, requiring precise timestamping on both CSN node ingress and egress ports as illustrated in Figure E.4 (“Path i” in the figure refers to the paths enumerated in Figure E.3). In Figure E.4,~~ ~~ti_1~~ ~~is the <syncEventEgressTimestamp> for the Sync message at the upstream time-aware Bridge,~~ ~~ti_2~~ ~~is the <syncEventIngressTimestamp> for the Sync message at the ingress CSN time-aware Bridge,~~ ~~te_1~~ ~~is the <syncEventEgressTimestamp> for the Sync message at the egress CSN time-aware Bridge, and~~ ~~te_2~~ ~~is the <syncEventIngressTimestamp> for the Sync message at the downstream time-aware Bridge or end-station.~~

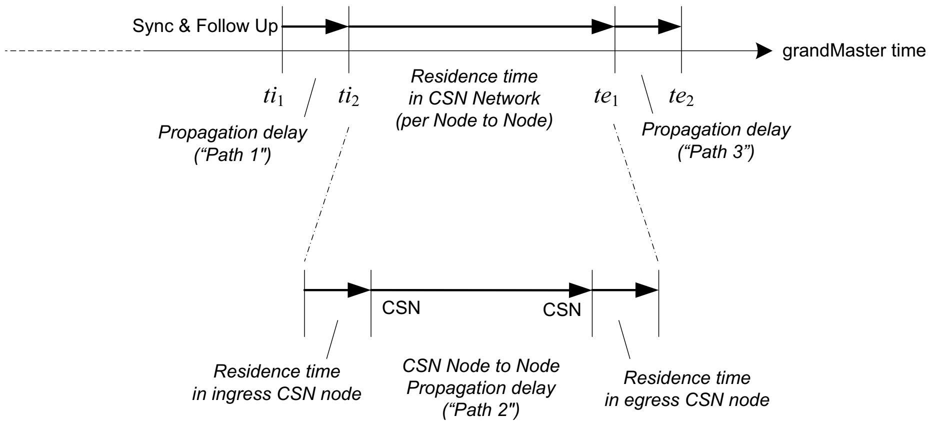

Sync & Follow Up
grandMaster time
ti_1
ti_2
Residence time in CSN Network (per Node to Node)
te_1
te_2
Propagation delay (“Path 1")
Propagation delay (“Path 3”)
CSN
CSN
Residence time in ingress CSN node
CSN Node to Node Propagation delay (“Path 2")
Residence time in egress CSN node

Figure E.4—~~Propagation delay and residence time over a CSN Backbone~~

E.4.2 ~~Path delay measurement between CSN node and neighbor time-aware system~~

~~The path delay measurement between a CSN node and a neighbor time-aware system is made as specified for the respective medium. This path delay measurement is made for the link between the CSN node and the neighbor time-aware system.~~

E.4.3 ~~Path delay measurement between CSN nodes~~

~~The path delay between the two nodes of a CSN is the propagation delay for the logical link that connects those two nodes. The method of measuring the path delay between two CSN nodes has two variations which are described in E.4.3.1 and E.4.3.2, respectively. The specific method to be used for a specific link technology is specified in E.6.~~

E.4.3.1 ~~Path delay measurement without network clock reference~~

~~Each CSN node has a free-running local clock. The path delay measurement uses the protocol, messages, and state machines described in Clause 11 for full-duplex, point-to-point links, as illustrated by Figure E.5.~~

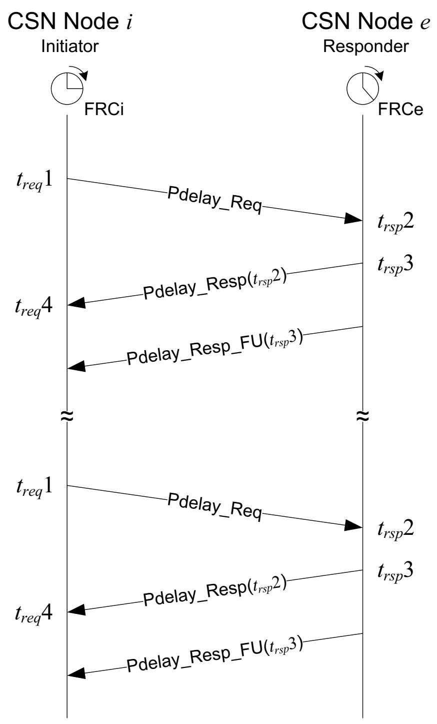


Figure E.5—~~CSN node-to-node path delay measurement~~

~~The computation of the neighborRateRatio and neighborPropDelay between two CSN nodes is done using the timestamps at the initiator and information conveyed in the successive Pdelay_Resp and Pdelay_Resp_Follow_Up messages. Any scheme that uses this information is acceptable, as long as the performance requirements of B.2.4 are met. As one example, the neighborRateRatio is computed as the ratio between a time interval measured by the local clock of the responder and its associated time interval measured by the local clock of the initiator, using a set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages and a second set of received Pdelay_Resp and~~

~~Pdelay_Resp_Follow_Up messages some number of Pdelay_Req message transmission intervals later, i.e., as show in Equation (E.1):~~

$$\cancel{\frac{(t_{rsp}3)_N - (t_{rsp}3)_0}{(t_{req}4)_N - (t_{req}4)_0}} \qquad (E.1)$$

~~where $(t_{rsp}3)_k$ is the time relative to the local clock of the responder that the k^{th} Pdelay_Resp message is sent, $(t_{req}4)_k$ is the time relative to the local clock of the initiator that the k^{th} Pdelay_Resp message is received, N is the number of Pdelay_Req message transmission intervals separating the first set of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages and the second set, and the successive sets of received Pdelay_Resp and Pdelay_Resp_Follow_Up messages are indexed from 0 to N with the first set indexed 0. The neighborPropDelay between the time-aware system and the CSN node is computed as shown in Equation (E.2):~~

$$\cancel{\frac{(t_{req}4 - t_{req}1)r - (t_{rsp}3 - t_{rsp}2)}{2}} \qquad (E.2)$$

~~where r is equal to neighborRateRatio, $t_{req}1$ is the time relative to the local clock of the initiator that the Pdelay_Req message for this message exchange is sent, $t_{rsp}2$ is the time relative to the local clock of the responder that the Pdelay_Req message for this message exchange is received, $t_{rsp}3$ is the time relative to the local clock of the responder that the Pdelay_Resp message for this message exchange is sent, and $t_{req}4$ is the time relative to the local clock of the initiator that the Pdelay_Resp message for this message exchange is received.~~

~~NOTE—The difference between mean propagation delay relative to the grandmaster time base and relative to the time base of the CSN node at the other end of the attached link (i.e., the responder CSN node) 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 frequency to the frequency of the LocalClock entity of the CSN node at the other end of the link. This ratio differs from 1 by 200 ppm or less. For example, for a worst-case frequency offset of the LocalClock entity of the CSN node at the other end of the link, relative to the grandmaster, of 200 ppm, and a measured propagation time of 100 ns, the difference in mean propagation delay relative to the two time bases is 20 ps.~~

~~Although the propagation delay between two CSN nodes is constant, a Pdelay_Req message is still sent periodically by each node to each other active node of the network to measure the neighborRateRatio between the node and each other node. Each node shall implement the state machines described in 11.2.15 and .~~

E.4.3.2 ~~Native 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 may be provided using the native measurement method rather than using the Pdelay protocol defined in 11.2.15 and . Such a situation is described in more detail as follows. The CSN MD entity populates the following per-port and MD entity global variables (described respectively in 10.2.4 and 11.2.12) as indicated:~~

— ~~asCapable (10.2.4.1) is set to TRUE,~~
— ~~neighborRateRatio (10.2.4.7) is set to the value provided by the native CSN measurement,~~
— ~~neighborPropDelay (10.2.4.8) is set to the value provided by the native CSN measurement,~~
— ~~computeNeighborRateRatio (10.2.4.10) is set to FALSE,~~
— ~~computeNeighborPropDelay (10.2.4.11) is set to FALSE, and~~
— ~~isMeasuringDelay (11.2.12.6) is set to TRUE to indicate that the CSN MD entity is measuring path delay (in this case, using its internal mechanism).~~

E.4.3.3 ~~Intrinsic CSN path delay measurement~~

~~If the CSN network features a native mechanism that causes each node's local clock to be fully synchronized to the local clocks of other nodes of the CSN such that the synchronized CSN time complies with the requirements specified in B.1, the CSN nodes need not implement the path delay mechanism but rather treat the path delay as part of the residence time of the distributed system. The propagation of the Sync messages in this case is described in E.5.2.~~

E.5 ~~Synchronization messages~~

~~The CSN network shall propagate synchronized time over the CSN to CSN end stations and to downstream non-CSN links, using Sync (and associated Follow_Up) messages, as illustrated in Figure E.6.~~

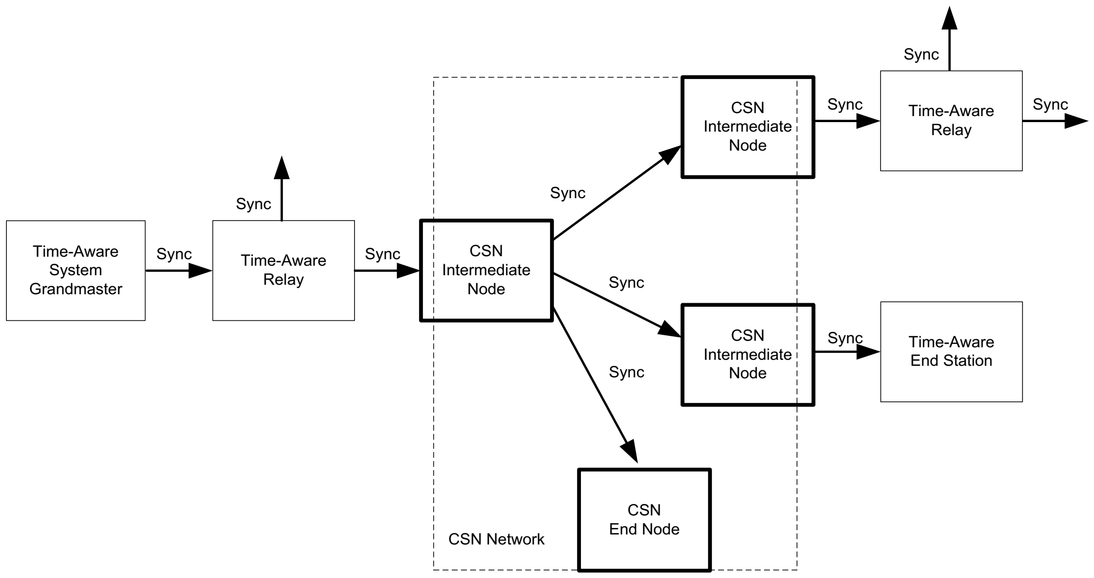


Figure E.6—~~IEEE 802.1AS Sync Message Propagation over the CSN backbone~~

~~Once the path delays have been measured (a) between the upstream time-aware Bridge and the ingress CSN node, (b) between the CSN nodes, and (c) between the egress CSN node and the downstream time-aware Bridge or end station, the CSN backbone can propagate the synchronization information received at its boundary nodes.~~

~~As with path delay measurements, various CSN technologies choose various methods for propagating time. These methods are described as follows.~~

E.5.1 ~~Synchronization message propagation on CSN without network reference clock~~

~~If the CSN network does not feature a native mechanism that synchronizes the CSN node local clocks to each other or to a reference, such that the CSN synchronized time complies with the requirements specified in B.1, the CSN local clock at CSN ingress and CSN egress nodes are considered independent free running clocks.~~

~~In this case, synchronization over the CSN links uses the Sync and Follow_Up protocol, messages, and state machines specified for full-duplex point-to-point links in 10.2.7, 10.2.11, 11.2.13, and 11.2.14. Individual CSN link technologies may specify media-specific encapsulation of gPTP event messages. See Table E.2 for selection of options per link technology.~~

~~One PortSync and one MD entity is instantiated per logical port (i.e., per CSN link). A CSN node behaves equivalently to a time-aware system. Sync and Follow_Up messages are either transmitted using unicast on each link or broadcasted. However if Sync and Follow_Up messages are broadcasted:~~

— ~~the Sync and Follow_Up messages are broadcast with the same port number used to broadcast Announce messages,~~
— ~~all PortSync/MD entity pairs except one set their logSyncInterval attribute (see 10.7.2.3) to 127, causing them to not generate any Sync messages, and~~
— ~~a dynamic selection of the MD entity that broadcasts the Sync message is needed (a CSN node can dynamically leave the CSN network). The dynamic selection algorithm is implementation specific and out of the scope of this standard.~~

E.5.2 ~~Synchronization message propagation on a CSN with network reference clock~~

~~If the CSN network features a native mechanism that allows the CSN node's local clocks to be fully synchronized to each other in a way that complies with the requirements specified in B.1, it is possible to simplify the path delay mechanism, as described below. This method is an alternative to E.5.1. Individual CSN link technologies may specify media-specific encapsulation of gPTP event messages. See Table E.2 for selection of options per link technology.~~

~~Sync messages are timestamped (1) when received at the ingress CSN node's time-aware system port (<syncEventIngressTimestamp>) and (2) when transmitted at the egress CSN node's time-aware system port (<syncEventEgressTimestamp>). The elapsed time between the egress and ingress timestamps is computed as the CSN residence time. In this scheme, the Sync message handling is split between the MDSyncSendSM state machine in the CSN ingress node and the MDSyncReceiveSM state machine in the CSN egress node as described in E.5.2.1 and E.5.2.2.~~

~~The reference plane for a CSN port and the path delay measurement method are specific to the type of CSN technology, and are defined in Table E.2.~~

E.5.2.1 ~~CSN ingress node~~

~~The CSN ingress node timestamps Sync messages received from the upstream time-aware system and compute the upstreamTxTime as described in 11.2.13.2.1, item f).~~

~~In addition, the setFollowUp function of the MDSyncSendSM state machine [see 11.2.14.2.3, item a)] is be modified as follows:~~

a) ~~The quantity *rateRatio* × *(<syncEventEgressTimestamp – upstreamTxTime>)* is not added to the followUpCorrectionField of the Follow_Up message to be transmitted by the ingress to the egress CSN node~~
b) ~~The CSN TLV (see E.5.2.1.1) is appended to the Follow_Up message transmitted by the ingress to the egress CSN node.~~

E.5.2.1.1 ~~CSN TLV~~

E.5.2.1.1.1 ~~General~~

~~The fields of the CSN TLV are specified in Table E.1 and E.5.2.1.1.3 through E.5.2.1.1.7. This TLV is a standard organization extension TLV for the Follow_Up message, as specified in 14.3 of IEEE Std 1588-2008. This TLV is not allowed to occur before the Follow_Up information TLV (see 11.4.4.3).~~

Table E.1—~~CSN TLV~~

Bits ~~8~~7 ~~7~~6 ~~6~~5 ~~5~~4 ~~4~~3 ~~3~~2 ~~2~~1 ~~1~~0	Octets	Offset From Start of TLV
tlvType	2	0
lengthField	2	2
organizationId	3	4
organizationSubType	3	7
rxTime	12	10
neighborRateRatio	4	14
neighborPropDelay	12	26
delayAsymmetry	12	38
domainNumber	1	50

E.5.2.1.1.2 ~~tlvType (Enumeration16)~~

~~The value of the tlvType field is 0x3.~~

~~NOTE—This is the value that indicates the TLV is a vendor and standard organization extension TLV, as specified in 14.3.2.1 of IEEE Std 1588-2008. The value is specified there as ORGANIZATION_EXTENSION, whose value is 0x3.~~

E.5.2.1.1.3 ~~lengthField (UInteger16)~~

~~The value of the length is 46.~~

E.5.2.1.1.4 ~~organizationId (Octet3)~~

~~The value of organizationId is 00-80-C2.~~

E.5.2.1.1.5 ~~organizationSubType (Enumeration24)~~

~~The value of organizationSubType is 3.~~

E.5.2.1.1.6 ~~upstreamTxTime (UScaledNs)~~

~~The computed upstreamTxTime value as described in 11.2.13.2.1 item f).~~

E.5.2.1.1.7 ~~neighborRateRatio (Integer32)~~

~~The neighborRateRatio value described in 10.2.4.7.~~

E.5.2.1.1.8 ~~neighborPropDelay (UScaledNs)~~

~~The neighborPropDelay value described in 10.2.4.8.~~

E.5.2.1.1.9 ~~delayAsymmetry (UScaledNs)~~

~~The delayAsymmetry value described in 10.2.4.9.~~

E.5.2.2 ~~CSN egress node~~

~~The CSN egress port sets neighborRateRatio to 1 and neighborPropDelay to 0 for its CSN port.~~

~~The CSN egress port modifies the function setMDSyncReceive of the MDSyncReceiveSM state machine [11.2.13.2.1 item f)] for the port to CSN link entity to extract upstreamTxTime, neighborPropDelay, and neighborRateRatio from the respective fields of the CSN TLV in the Follow_Up message received from the CSN ingress node.~~

~~The CSN egress port also modifies the ClockSlaveSync state machine (see 10.2.12) to get the upstreamTxTime, neighborPropDelay, neighborRateRatio, and delayAsymmetry values from the respective fields of the CSN TLV in the Follow_Up message received from the CSN ingress node.~~

~~The CSN egress node removes the CSN TLV from the Follow_Up message it creates and transmits to the downstream time-aware system.~~

E.6 ~~Specific CSN requirements~~

~~The reference plane for a CSN port is specific to the type of CSN technology and is defined in Table E.2.~~

Table E.2—~~Definitions and option selections per link technology~~

~~CSN link technology~~	~~Reference plane~~	~~Path Delay measurement~~	~~gPTP event message encapsulation~~
~~Multimedia over Coax Alliance (MoCA) v2.0~~	~~The first bit of an Event message crossing to and from the MoCA CTC clock domain.~~	~~MoCA Ranging Protocol~~	~~Encapsulated in control frames as described in the MoCA MAC/PHY Specification v2.0.~~
~~ITU-T G.hn (SG15)~~	~~The first bit of an Event message crossing the A-interface. See E.6.2.~~	~~Pdelay mechanism as defined in E.4.3.1.~~	~~None~~

E.6.1 ~~MoCA-specific behavior~~

~~The MoCA network is a CSN. The non-MoCA port of a Bridge behaves as a time-aware system port, which might or might not use the MoCA CTC clock for timestamping event messages. If the non-MoCA port of the Bridge uses a different clock than the MoCA CTC clock, then the Bridge reconciles the non-CTC timestamp with the CTC time.~~

~~Sync messages shall be timestamped using the CTC clock (1) when the Sync message crosses the MoCA's ingress node's timestamp reference plane (<syncEventIngressTimestamp>) and (2) when it crosses the egress CSN node's reference plane (<syncEventEgressTimestamp>).~~

~~The elapsed time between the egress and ingress timestamps, <syncEventIngressTimestamp> – <syncEventIngressTimestamp>, is computed as the MoCA residence time.~~

~~The MoCA port whose port role is MasterPort propagates the Sync and Follow_Up messages as described in E.5.2. The CSN TLV values of the Follow_Up message sent over the MoCA network is computed using the LocalClock, i.e., the MoCA CTC clock.~~

~~IEEE 802.1 AS Frames shall be transmitted over the MoCA network as MoCA control frames as described in the MoCA MAC/PHY Specification v2.0.~~

~~NOTE—The Channel Time Clock (CTC) is specified in the MoCA MAC/PHY Specification v2.0.~~

E.6.2 ~~ITU-T G.hn-specific behavior~~

~~A port of a time-aware system that includes one or more ITU-T G.hn ports shall behave as defined in E.3, E.4, and E.5, but for aspects where more than one behavior or option is described, the system behaves as defined in Table E.2.~~

~~ITU-T G.hn defines a 32-bit timestamp (which is placed the TSMP field of a G.hn encapsulation header). This timestamp, as described in Table E.2, is captured each time an Event message is transmitted or received by an ITU-T G.hn port and is used for all gPTP event messages, including SYNC and PDELAY messages.~~

E.7 ~~Grandmaster capability~~

~~Each CSN Node may be grandmaster capable, allowing a CSN node to act as the grandmaster clock either for a homogeneous CSN network or for a heterogeneous network.~~

~~The Announce messages are either sent via unicast on each link or broadcasted. However if Announce messages are broadcasted, the Announce message shall use the same port number as used by the Sync and Follow_Up messages by a single PortSync/MD entity pair on the port. This is accomplished by setting the logAnnounceInterval attribute (see 10.7.2.2) to 127 for all but one PortSync/MD entity pair, causing them to not generate any Announce messages.~~

E.8 ~~CSN clock and node requirements~~

~~The CSN clock performance shall comply with the requirements specified in B.1. The CSN node performance shall comply with the requirements specified in B.2.2, B.2.3, and B.2.4.~~

Annex F

(informative)

PTP profile included in this standard

The specification in this standard of ~~synchronized time~~synchronized time transport over a full-duplex, point-to-point link includes a PTP profile. The information contained in a PTP profile is described in 19.3 of IEEE Std 1588-2008. This annex summarizes the PTP profile for transport of timing over full-duplex, point-to-point links. This PTP profile is also used in the transport of timing over CSN for the case where a CSN network clock reference is not present (see Clause 16). This PTP profile is not used in the transport of timing over IEEE 802.11 links and IEEE 802.3 EPON links; both these transports use native timing mechanisms to assist in the ~~synchronized time~~synchronized time transport.

F.1 Identification

The identification values for this PTP profile (see 19.3.3 of IEEE Std 1588-2008) are as follows:

PTP Profile:
IEEE 802.1AS PTP profile for transport of timing ~~over full-duplex, point-to-point links~~
Profile Name: IEEE 802.1AS PTP profile
profileNumber: 0
primaryVersion 2~~1.0~~
revisionNumber: 0
p~~P~~rofileI~~ i~~dentifier: 00-80-C2-00-02~~1~~-00

This profile is specified by the IEEE 802.1 Working Group of the IEEE 802 LAN/MAN Standards Committee.

A copy may be obtained by ordering IEEE Std 802.1AS-2011 from the IEEE Standards Organization.[19]

This PTP profile is a revision of the PTP profile included in IEEE Std 802.1AS-2011, i.e., major changes have been made to the profile relative to Version 1.0. Therefore, the primaryVersion is changed to 2, with revisionNumber 0.

F.2 PTP attribute values

The ranges and default values for time-aware system attributes covered by this profile are as follows:

a) A domain whose domain number is 0 is present. A domain whose domain number is in the range 1 - 127 may be present (see 8.1).
b) The default logAnnounceInterval (see 10.7.2.2) is 0. The value 127 is supported.
c) The default logSyncInterval (see 11.5.2.3) is –3. The value 127 is supported.
d) The default logPdelayReqInterval (see 11.5.2.2) is 0. The value 127 is supported.
e) The default announceReceiptTimeout (see 10.7.3.2) is 3.
f) The default values of priority1, for different media, are specified in 8.6.2.1, Table 8-3. The value of priority1 for a time-aware system that is not grandmaster-capable is 255.
g) The default value of priority2 is 248 (see 8.6.2.5).

[19]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org).

h) The default observation interval for offsetScaledLogVariance is equal to the default sync interval, i.e., 0.125 s (see 8.6.2.4).

F.3 PTP options

a) The best master clock algorithm is the alternate BMCA specified in 10.3.

~~NOTE—This BMCA is similar, but not identical, to the default BMCA described in IEEE Std 1588-2008. The main differences are (1) Announce information from a potential best master is used immediately, i.e., there is no notion of foreign master qualification, and (2) a port whose port role is determined by the BMCA to be MasterPort has its port role changed to MasterPort immediately, i.e., there is no "PreMasterPortRole."~~

b) The following options of 17.7 of IEEE Std 1588-Rev are invoked:
   1) The FAULTY state is not used.
   2) The UNCALIBRATED state is not used.
   3) The LISTENING state is not used
   4) The PRE_MASTER state, and PRE_MASTER qualification are not used.
   5) The foreign master feature is not used.

c) The management mechanism is the mechanism specified in Clause 14 and Clause 15.

d) The path delay mechanism is the peer delay mechanism (see Clause 11).

e) The transport mechanism is full-duplex, point-to-point, and uses attribute values described in Annex F of IEEE Std 1588-2008 for IEEE 802.3 Ethernet. Specifically, the address, Ethertype, and subtype are specified in 11.3.4, 11.3.5, and 11.3.6.

f) A time-aware system that contains one PortSync and one MD entity is an ordinary clock. A time-aware system that contains more than one PortSync and more than one MD entity is a boundary clock. ~~All time-aware systems are two-step clocks.~~

g) Each port of a time-aware system measures the frequency offset of its neighbor, at the other end of the attached link, relative to itself (see Clause 11). The frequency offset, relative to the grandmaster, is accumulated in a standard organization TLV that is attached to the Follow_Up message if the port is two-step and the Sync message if the port is one-step (see 11.4.4.3). The standard organization TLV also carries information on grandmaster traceability and phase and frequency change due to the most recent grandmaster change. The physical adjustment of the frequency of the LocalClock entity (i.e., physical syntonization) is allowed but not required.

h) The path trace feature of 16.2 of IEEE Std 1588-2008 is used (see 10.6.3.2.8).

i) A standard organization TLV is defined that allows a port of a time-aware system to request that its neighbor slow down or speed up the rate at which it sends Sync/Follow_Up, peer delay, and/or Announce messages (see 10.6.4.3).

j) The acceptable master table feature of IEEE Std 1588-2008 is used with IEEE 802.3 EPON links to ensure that the OLT is master and ONUs are slaves.

NOTE—This feature is used with EPON links, and therefore could be considered to be outside the PTP profile (because EPON links are ~~-~~ not part of the PTP profile). It is included here because it is one of the optional features described in IEEE Std 1588-2008.

k) Except for items g) and j) above, the optional features of Clause 16 and Clause 17 of IEEE Std 1588-2008 are not used.

l) The experimental security protocol of Annex K of IEEE Std 1588-2008 is not used.

m) The cumulative frequency scale factor of Annex L of IEEE Std 1588-2008 is not used, but cumulative frequency offset relative to the grandmaster is accumulated in a TLV, and is encoded in the same way as the cumulative frequency scale factor in Annex L of IEEE Std 1588-2008.

F.4 LocalClock and time-aware system performance requirements

The LocalClock performance requirements are as specified in B.1. The time-aware system performance requirements are as specified in B.2.

Annex G

(informative)

The Asymmetry Compensation Measurement Procedure based on line-swapping

G.1 Introduction

This annex describes the asymmetry compensation measurement procedure based on the line-swapping method. The entire procedure is controlled by the Network Management System (NMS), except that the line-swapping is manually operated.

NOTE—When a port is put into asymmetry measurement mode, it is put into this mode on all domains. The per-port, global variable asymmetryMeasurement mode is common to and accessible by all domains (see 10.2.4.2).

G.2 Pre-conditions for measurement

The following pre-conditions should be met, both to improve the accuracy of the measurement and to make the measurement procedure more convenient:

a) The measurement environment, including the testing nodes (i.e., the time-aware systems at the endpoint of the link whose asymmetry is being compensated) and related nodes (i.e., nodes in the paths between the testing nodes and the grandmaster), should enable gPTP and BMCA, so that the test can be made for each link without changing the configuration.
b) The testing nodes should have redundant paths for synchronization, so that they remain synchronized to the grandmaster during the asymmetry compensation measurement.

G.3 Measurement procedure

The assumed measurement environment is shown in Figure G-1. Before the measurement starts, every node has enabled syntonization (i.e., by measuring neighborRateRatio (see 10.2.4.7) for each port of each time-aware system and accumulating the respective values over the synchronization spanning tree paths to obtain rateRatio (see 10.2.7.1.4) relative to the grandmaster[20]) and time synchronization. The testing link is between Port2 on node ACC1 and Port1 on node ACC2.

[20]This standard neither requires nor prohibits syntonization (see 3.23) at the physical layer, e.g., using Synchronous Ethernet as the physical layer. If frequency is syntonized at the physical layer, the respective neighborRateRatio and rateRatio values are expected to be close to 1.

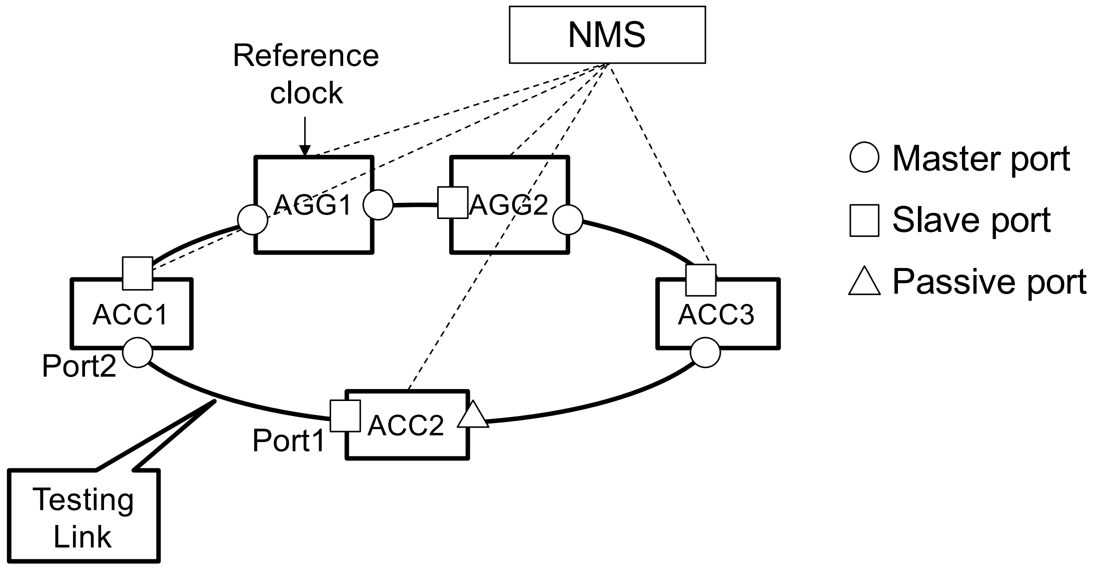


Figure G-1—Asymmetry compensation measurement procedure

The measurement procedure is as follows:

a) The NMS puts Port1 of ACC2 and Port2 of ACC1 into asymmetry measurement mode through the MIB (i.e., by setting the managed object asymmetryMeasurementMode for each port to TRUE). These two ports will not affect the PTP calculations of either node when the ports are in asymmetry measurement mode. If synchronization flowed over the link connecting these ports prior to their being put into asymmetry measurement node, the BMCA will result in a reconfiguration of the synchronization spanning tree so that both ACC1 and ACC2 remain synchronized.
b) Port1 of ACC2 will send Pdelay_Req messages periodically using the peer delay measurement mechanism and receive Pdelay_Resp and Pdelay_Resp_Follow_Up messages. For each set of messages, ACC2 should save t3 (the <pdelayRespEventEgressTimestamp>, see 11.3.2.1, carried in the Pdelay_Resp_Follow_Up message) and t4 (the <pdelayRespEventIngressTimestamp>, taken when the Pdelay_Resp message timestamp point crosses the reference plane at Port1 of ACC2 on reception).
c) The NMS reads and saves multiple sets of (t3, t4) from ACC2 through the MIB. It is necessary that each set of (t3, t4) is from the same measurement, i.e., from the same peer delay message exchange. The number of (t3, t4) sets can be decided as required, and is outside the scope of gPTP.
d) Manually exchange the transmit and receive fibers of Port2 of ACC1 and Port1 of ACC2. Wait until the port status and protocol status become stable again.
e) Port1 of ACC2 will again make periodic peer delay measurements, and save each set of measurement values (t3', t4') (the primes are used to de measurements that have occurred after the transmit and receive fibers have been exchanged).
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). 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).

g) Based on the above result, the NMS sets the asymmetry value for Port1 of ACC2 and Port2 of ACC1 through the MIB.

h) After the NMS sets Port 1 of ACC2 and Port2 of ACC1 into normal mode (i.e., by setting the managed object asymmetryMeasurementMode for each port to FALSE), the delay asymmetry measurement of the testing link is completed. ACC1 and ACC2 will use the computed delay asymmetry as compensation in the PTP calculations.

It is also possible to compute the asymmetry ratio, i.e., the ratio of the delay on the receive fiber at ACC2 (Delay_rx_fiber, the delay of the Pdelay_Resp) to the delay on the transmit fiber at ACC2 (Delay_tx_fiber, the delay of the Pdelay_Req), both after line swapping. In this case, the NMS should collect multiple sets of (t1, t2, t3, t4) and (t1', t2', t3', t4') before and after line-swapping, respectively, where t1 is the <pdelayReqEventEgressTimestamp> (taken when the Pdelay_Req message timestamp point crosses the reference plane at Port1 of ACC2 on transmission), t2 is the <pdelayReqEventIngressTimestamp> (carried in the requestReceiptTimestamp field of the Pdelay_Resp message), and t3 and t4 are as given above. The NMS can compute the delay on the receive fiber as [(t4' - t1)*neighborRateRatio - (t3' - t2)] /2 and the delay on the transmit fiber as [(t4 - t1')*neighborRateRatio - (t3 - t2')]/2. Then the asymmetry ratio is Delay_rx_fiber/Delay_tx_fiber = [(t4' - t1)*neighborRateRatio - (t3' - t2)] / [(t4 - t1')*neighborRateRatio - (t3 - t2')]

Annex H

(informative)

Bibliography

[B1] Allan, David W., Ashby, Neil, and Hodge, Clifford C., "The Science of Timekeeping," Hewlett Packard Application~~Note~~ 1289, 1997.[21]

[B2] IEEE MA-L, MA-M, and MA-S assignments ~~EUI-64, IEEE EUI-48, and IEEE MAC-48 assigned numbers~~ may be obtained from the IEEE Registration Authority. These assignements may be used to form EUI-48s and EUI-64s, as described in the following three tutorials: (a) *Guidelines of for Use Organizationally Unique Identifier (OUI) and Company ID (CID)*, (b) *Guidelines for 64-bit Global Identifier (EUI-64)*, and (c) *Guidelines for 48-Bit Global Identifier (EUI-48)*.[22]

[B3] IEEE Std 802.1Qbb™-2011, IEEE Standard for Local and Metropolitan Area Networks—MediaAccess Control (MAC) Bridges and Virtual Bridged Local Area Networks—Amendment 17: Priority-based Flow Control.

[B4] IEEE Std 802®, IEEE Standard for Local and Metropolitan Area Networks—Overview and Architecture.

[B5] ~~IEEE Std 802.1X™-2010, IEEE Standard for Local and Metropolitan Area Networks—Port-Based Network Access Control.~~[23, 24]

[B6] IEEE Std 1139™-1999, IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology—Random Instabilities.

[B7] IEEE Std 1003.1™-2008, IEEE Standard for Information Technology—Portable Operating System Interface (POSIX®) Base Specifications, Issue 7.

[B8] ISO 8601:2004, Data elements and interchange formats—Information interchange—Representation of dates and times.[25]

[B9] ISO/IEC 8802-2, Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 2: Logical link control.

[B10] ISO/IEC 9945:2003, Information technology. Portable Operating System Interface (POSIX®).

[21]Available at http://www.allanstime.com/Publications/DWA/Science_Timekeeping/TheScienceOfTimekeeping.pdf.

[22]Available at http://standards.ieee.org/regauth/. Tutorials on these assigned numbers can be found on this Web site.

[23]~~IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org.).~~

[24]~~The IEEE standards or products referred to in Annex G are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.~~

[25]ISO publications are available from the ISO Central Secretariat, 1, ch. de la Voie-Creuse, Case Postale 56, CH-1211, Geneva 20, Switzerland (http://www.iso.org/). IEC publications are available from the Central Office of the International Electrotechnical Commission, 3, rue de Varembé, P.O. Box 131, CH-1211, Geneva 20, Switzerland (http://www.iec.ch/). ISO/IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (http://www.ansi.org/).

[B11] ISO/IEC 14882:2003, Programming languages—C++.

[B12] ITU-R Recommendation TF.460-6, Standard-frequency and time-signal emissions, 2002.[26]

[B13] ITU-T Recommendation G.810, Definitions and Terminology for Synchronization Networks, ITU-T, Geneva, August, 1996, Corregendum 1, November, 2001.

[B14] Jekeli, Christopher, "Geometric Reference Systems in Geodesy," Division of Geodesy and Geospatial Science, School of Earth Sciences, Ohio State University, July 2006.[27]

[B15] MoCA MAC/PHY Specification v1.0, MoCA-M/P-SPEC-V1.0-07122009, Multimedia over Coax Alliance (MoCA), July 12, 2009.[28]

[B16] MoCA MAC/PHY Specification Extensions v1.1, MoCA-M/P-SPEC-V1.1-06162009, Multimedia over Coax Alliance (MoCA), June 16, 2009.

[B17] Service de la rotation terrestre, Observatoire de Paris, 61, Av. de l'Observatoire 75014 Paris (France).

[B18] "The International System of Units (SI)," 8th edition, Bureau International des Poids et Mesures, 2006.[29]

[B19] U.S. Naval Observatory.[30]

[B20] Anthanasios Papoulis, "Probability, Random Variables, and Stochastic Processes (Third Edition)," Mc-Graw-Hill, 1991.

[26]ITU publications are available from the International Telecommunications Union, Place des Nations, CH-1211, Geneva 20, Switzerland/Suisse (http://www.itu.int/).

[27]Available at https://kb.osu.edu/dspace/bitstream/1811/24301/1/Geom_Ref_Sys_Geodesy.pdf.

[28]MoCA specifications are available from the Multimedia over Coax Alliance at http://www.mocalliance.org/specs.

[29]Available at http://www.bipm.org/en/si/si_brochure/.

[30]Available at http://maia.usno.navy.mil/ser7/tai-utc.dat.

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 purpos of TG/WG ballot.>>

Z.1 Revision history

Z.1.1 Revision 0.1

This is the initial version, prepared by the editor. It contains the following:

a) Inclusion of placeholders for all clauses and annexes for which there is not yet text.
b) Addition of Annex Z (this annex), including revision history and copying of AVB (TSN) Gen 2 assmptions from latest version (v15) of assumptions document (see Z.3.1).
c) Addressing of the following comments against the final recirculation sponsor ballot of P802.1AS-Cor-1/D3.1, for which it was decided to address the respective comments in 802.1ASbt (note that some of these comments are also in the maintenance database, as indicated; note also that there are still remaining comments and maintenance items to be addressed):
    1) Comment 5 (insertion of the word “trace” between “path” and “TLV” in 10.5.3.3.1).
    2) Comment 6, maintenance item 0113 (removal of Pdelay_Req from row 2 of Table 11-5).
    3) Comment 7, maintenance item 0114 (addition of row to Table 11-5 indicating that correctionField for Sync, Announce, and Pdelay_Req is zero).
    4) Comment 9, maintenance item 0116 (fixing of units for ADEV in Figure B.2 (it should be dimensionless)).
    5) Maintenance item 0118 (Issue related to loss of single Follow_Up causing Sync receipt timeout).

Z.1.2 Revision 0.2

This version contains changes agreed to at the September, 2013 802.1 TSN meeting:

a) A note is added to 10.3.11.2.1 to remind the reader of the what it means for one priority vector to be superior to another. This will address maintenance item 119.
b) A new subclause 10.7 is added, which for general time-aware systems points to the requirements of B.1 and B.2.4, and a corresponding PICS entry is added. Related to this, subclause E.8 is modified to reflect that the former requirements of B.2.2 and B.2.3 are now recommendations.
c) The units of the vertical axis of Figure B-2 are corrected (ADEV is dimensionless).
d) Other changes that are minor or editorial are made.

Z.1.3 Revision 0.3

a) Indicated in 6.3.4.2 that bit numbering is 0 - 7 rather than 1 - 8, and fixed Table 10-6 to reflect this.
b) Included all flags in Table 10-6, and removed Table 11-4 and referred to Table 10-6 instead. (Note that Table 11-4 is incorrectly referenced as Table 11-1 in D0.3; this is fixed in D0.4).
c) Added Sync, Pdelay_Req, Announce, and Signaling message entries to Table 11-5.
d) Added new subclause 10.7, which references respective media-independent performance requirements in Annex B. Added respective PICS entries to Annex A. Fixed Annex E to correctly reference the Annex B performance requirements.

Z.1.4 Revision 0.4

a) Added definition of maxium relative time error (subclause 3.11A)
b) Indicated Annex E will be renumbered as clause 16, and a placeholder Annex E will remain indicating the material was moved.
c) Bit numbering was changed from 1-8 to 0-7 in various tables (showing messages or TLVs).
d) A minor error on the range for the arrays reselect and selected was fixed, in the definiations (10.3.8.1 and 10.3.8.2).
e) An exponent in one of the states of the PortAnnounceInformationStateMachine was made more readable.
f) A note was added to the description of the computation of mean propagation delay in 11.2.15.2.4 indicating the differences in the organization of the computation here versus IEEE Std 1588-2008.
g) An editor's note was added to 10.3.11.3 (PortAnnounceInformation SM) describing the issue where a time-aware system invokes the BMCA and an new GM is selected, but the time-aware system never receives Sync on the respective master port (and may or may not continue receiving Announce).
h) Figure 10-8 (PortSyncSyncSend SM) was fixed to prevent the persistent looping from the SET_SYNC_RECEIPT_TIMEOUT_TIME state back to itself.
i) An editor's note was added to the beginning of clause 10 describing the problem that can occur when syncInterval and clockMasterSyncInterval are almost equal.

Z.1.5 Revision 0.5

a) Changed draft to Revision format, in antipation of Revision PAR being approved and then working on a Revision rather than an Amendment.
b) Updated abstract and scope to reflect latest draft revision PAR text.
c) Indicated in editor's note that the current text that indicates the standard is limited to bridged networks will be generalized, and that it is planned to go through the entire document and remove or generalize the references to "bridges" or "bridging" where appropriate.
d) Revised definition of max|RTE| in Clause 3 as agreed in previous meeting.
e) Added text related to optional, additional domain, to support the multple timescales feature (this was added in various subclauses in the document).
f) Made some changes to descriptions of timescales in 8.2 and its subclauses to make the text consistent with IEEE Std 1588 (both 2008 edition and revisions agreed to in P1588 committee for next edition of 1588).
g) An editor's note eas added to 10.5.4.2.2 to indicate that the processing of the message interval request TLV on reception actually is manditory, as desired (there was discussion in the previous meeting (May 2014) on making the processing of the TLV on reception manditory; it was mistakenly not realized in the discussion that this already is manditory). Note that transmission of the TLV is optional.
h) The MDSyncReceiveSM state machine (Figure 11-6) was updated to address the problem of Sync Receipt Timeout being caused by loss of a single Follow_Up (discussed in the July, 2013 TSN TG meeting).
i) A NOTE was added to 11.2.17.3.4 to describe the differences in the organization of the propagation delay calculation here and in IEEE 1588 - 2008.
j) The MDPdelayReq state machine (Figure 11-8) was modified to address the Pdelay_Req storm issue.
k) All the flags are now defined in Table 10-6, and this table is referenced in 11.4.2.4.
l) Entries for Sync, Pdelay_Req, Announce, and Signaling are added to Table 11-6.
m) Bit numberings in various talbes are corrected (changed to 0 through 7, from 1 through 8).
n) Other, minor edits were made.

Z.1.6 Revision 0.6

a) The list of revisions for D0.5 was added to Annex Z (this was mistakenly forgotten in D0.5).
b) Headers and footers were fixed to reflect P802.1ASbt (with correct revision number, D0.6 here), and copyright date of 2014 with indication that this is an unapproved draft subject to change.
c) A minor edit was made to the Scope.
d) A few minor edits were made on pp. 19 and 27 regarding the description of multiple domains.
e) It was indicated in 8.2.1 that the timescale of domain 0 shall be PTP.
f) The managed objects t1, t2, t3, and t4, corresponding to peer delay mechanism timestamps, are added to clause 14 for the feature of management support for compensation of delay asymmetry.
g) The PortSyncSyncSend state machine, in Figure 10-8, was modified to implement the fix needed for maintenance item 138. This item is also described in Z.3.3.2, second bullet item from bottom.
h) Edited instances of the words "bridge," "bridging," and other forms of the word, and replaced with a more general term. In most cases, "time-aware bridge" was replaced by "time-aware node." However, several points still need to be decided by the TSN TG: (1) should "bridge" be replaced by "node" when not talking about ones that are time-aware, but where more general network elements than bridges could be referred to? (2) Annex E refers to "CSN nodes," which are not bridges. Should a different term be used, or should we simply always precede "node" by "CSN" when we mean "CSN node"?
i) Added multiple-domain network example to clause 7.
j) Added redundancy examples to 7.2.4. As of the preparation of D0.6, this subclause has not been reviewed and must be reviewed.
k) Other, minor edits were made.

Z.1.7 Revision 0.7

a) Further edits were made to remove references to "bridges" and "bridging". "Time-aware node" was replaced by "Time-aware relay" when referring to time-aware systems ("time-aware end station" is still used). But, note that "node" is still used when referring to CSN nodes.
b) References to 802.Q were either removed or made more generic in cases where the reference should be more generic than just bridging.
c) The managed objects t1, t2, t3, and t4 introduced in D0.6 are now UInteger48, and represent the value of the timestamp modulo 2^{32} ns. This means that each of these objects will roll over every (approximately) 4.3 s. Previously, these were 96 bit unsigned integers, and could represent the full PTP time in seconds, nanoseconds, and fractional nanoseconds.
d) An editor's note was added to 10.1.2 pointing out that the statement that "an implementation may optionally provide the ability to configure a time-aware system as grandmaster-capable via a management interface" is inconsistent with the fact that the gmCapable and clockClass managed objects in 14.2 are read only. This needs to be resolved.

Z.1.8 Revision 0.8

a) The document designation was changed from 802.1ASbt to 802.1AS-Rev.
b) A Note was added to 10.1.2 to clarify the relation between priority1 and gmCapable, and the relation between both these parameters to clockClass.
c) The definition of portEnabled in 10.2.4.12 was revised according to discussion at the November, 2014 TSN TG meeting. An editor's note was added indicating that pttPortEnabled will be changed to ptpPortEnabled (i.e., a change in the name of a variable) once the TSN TG confirms that portEnabled will (or will not) be removed (since, if portEnabled is removed, it will be more efficient to do this and change the name of pttPortEnabled at the same time).

d) An Editor’s note was added to E.6 indicating that information on IEEE Std 1901 media is needed and, if this information is not supplied within a few months of the date of this revision, this feature will be dropped.
e) As discussed at the November, 2014 TSN TG meeting, the Port~~Role~~StateSelection state machine (Figure 10-14) was modified so that updt~~Role~~StatesTree() is triggered if a time-aware system attribute (priority1, clockClass, clockAccuracy, offsetScaledLogVariance, or priority2) changes.
f) The per-port managed object asymmetryMeasurementMode is added, used to enable link asymmetry measurement. An editor’s note is added outlining the state machines that must look at this attribute to determine whether Sync, Follow_Up, and Announce should be sent on that port (and changes to the state machines). A new informative annex (Annex G, and the existing Annex G (Bibliography) becomes Annex H) describing the link asymmetry measurement procedure is added.
g) Editor’s notes are added pertaining to work needed for handling links that use link aggregatoin and determining at a port whether an 802.11 or 802.3 EPON link supports a particular gPTP domain.

Z.1.9 Revision 0.9

a) Added to Draft Status a list of contributions and other items that are needed for completion of the standard
a) Replaced portEnabled and pttPortEnabled by portOper and ptpPortEnabled, respectively.
b) Removed editor’s notes on pages 54 - 55.
c) Added editor’s note to 10.3.12.1.1, related to the variable systemIdentityChange.
d) Made changes to state machines identified in editor’s note of 14.26.29, related to management support for asymmetry compensation. Added editor’s note describing possible changes to MDPdelayReq state machine.
e) Combined the managed objects t1, t2, t3, and t4, in subclauses of 14.6, into a single array named pdelayTruncatedTimestamps. Added Notes giving relation between these truncated timestamps and the actual (full) PTP timestamps.

Z.1.10 Revision 1.0

a) Updated list of items up front (in title section) for which contributions/presentations are needed. Items (c) and (d) were updated, and items (g), (h), and (i) were added.
b) Added a NOTE where ptpPortEnabled and portOper are defined, to indicate that these parameters are just name changes (the old names were pttPortEnabled and portEnabled, respectively).
c) Scaling of Figure 10-12 was fixed.
d) It is clarified in 10.3.12.1.1 that s~~w~~ystemIdentity changes when at least one of the attributes changes (it previously implied that exactly one attribute had to change, rather than at least one).
e) An editor’s note below Figure 10-13 is removed, as it was determined that the issue it describes is not actually a problem.
f) The MDPdelayReq state machine, WAITING_FOR_PDELAY_INTERVAL _TIMER state, so that the computation of mean propagation delay and rateRatio starts fresh after exiting asymmetry measurement mode (i.e., values computed prior to entering asymmetry measurement mode are not used).
g) In clause 14, the elements of the pdelayTruncatedTimestamp array are put in a table (rather than being described in separate subclauses). Editor’s notes at the beginning of clause 15 are updated.
h) In Annex A (PICS Proforma), the section number for item BMC-2 is filled in, and a note is added to item BMC-3 to indicate that there is one instance of the Port ~~Role~~State Selection state machine per time-aware system, but that some computations of this state machine are per port and some are for the time-aware system as a whole.
i) Clause 12 (802.11 transport) is updated to include Fine Timing Measurement (FTM). Some errors in arithmetic and other minor errors/typos in the master and slave state machines are fixed.
j) One-step operation (both transmit and receive) is added for full-duplex Ethernet. Received two-step information is processed as it would be in a 1588 BC, i.e., the port waits for the associated

Follow_Up information and then processes the information. This means that, if two-step information is received by a time-aware system and transmitted on a one-step port, the port transmits a Sync message without Follow_Up. Note that a 1588 TC (both P2P and E2E), as specified in the 2008 edition, would process and send Sync, and then forward Follow_Up unchanged. Note also the the P1588 Upkeep subcommittee is considering allowing BC-like processing (as described here) for a TC, for the next edition of 1588.

Z.2 General editor's notes for all clauses and annexes

a) Placeholders are included for all clauses and annexes for which there are not yet any edits/changes. Any clauses or annexes for which there are no edits/changes when the draft is ready for sponsor ballot will be removed.

Z.3 P802.1AS-Rev (formerly P802.1ASbt) assumptions

Z.3.1 General

This subcause documents the TSN assumptions that are relevant to P802.1AS-Rev. The assumptions have been historically maintained in the Audio/Video Bridging (AVB) Gen 2 Assumptions document (Note: the term "Audio/Video Bridging" has been replaced by "Time-Sensitive Networking" (TSN), though the title of the assumptions document has not (yet) been changed. As of the preparation of this 802.1AS-Rev draft, the latest version of this document is v15 (July 16, 2013), at http://www.ieee802.org/1/files/public/docs2013/avb-pannell-gen2-assumptions-0313-v15.pdf). It was decided by the TSN Task Group (TG) around the time of the July, 2013 802.1 meeting that, instead of maintaining all the assumptions in one document, the assumptions relevant to each TSN standard would be maintained in Annex Z of the respective standard (this approach was used for the AVB Gen 1 standards).

The overview of the assumptions document indicates that the document is a collection of possible ideas for possible inclusion in the next versions of SRP (802.1Qat Gen 2) and/or the Gen 2 AVB Shaper (802.1Qbv) and/or Gen 2 gPTP (802.1AS-Rev) or some new standard. It should not be considered as a Work Item list yet until the entries are Green. Each non-Green item needs contributions (i.e., presentations) before it can be agreed to and considered an item to be added to a draft standard. These presentations are needed immediately.

The assumptions document editor used the following color coding scheme to indicate the status of each assumption; this scheme is maintained here:

— Green text = agreed to at a Plenary (was previously blue or red)
— Blue text = newly agreed to (was red at last face to face)
— Black text = not decided
— Changes marked with red from last version

The gPTP assumptions are organized by major section of the assumptions document, then by slide within that section, then by bullet item, and then sub-bullet item. Only the assumptions relevant to gPTP (i.e., to 802.1AS-Rev) are included here (the decision on which assumptions are relevant was made by the editor).

Z.3.2 Requirements

All performance goals are to degrade gracefully over increasing hops.

Z.3.2.1 General

- Gen 2 devices need to co-exist w/Gen 1 devices
  — Don't want to need to modify Gen 1 Talkers & Listeners when connected to a Gen 2 cloud (of bridges) i.e., Backwards compatibility is required
- Need to support islands of Gen 2 bridges connected to Gen 1 or through Gen 1 bridges
- Need to keep IS-IS (Intermediate System to Intermediate System) parameters to a minimum
-
- The next two bullet items must be further discussed during and edited after the joint session:
- Do not want to be able to run one instance of IS-IS for SPB (Shortest Path Bridging – IEEE 802.1aq) and every thing else we need?
- Do not want to be required to run an instance of IS-IS (Shortest Path Bridging – IEEE 802.1aq) that includes all the information needed by 802.1AS and all other TSN features. That is, we would like to be able to have an IS-IS that is 802.1AS-specific, and has only information needed by 802.1AS in its database. This would be useful for situations where only 802.1AS is running but not the other TSN protocols.
- Would like to be able to limit the use of IS-IS to only those features that require it, such that most of the enhanced features of ASbt can be used without needing IS-IS? The specific features that require IS-IS are: TBD.
- Goal is to allow the use of a longer path if the attributes of the nodes and links (e.g., delay asymmetry, time-stamp accuracy and granularity, local clock noise, residence time, pdelay turnaround time, etc.) would result in better time-synchronization performance.
  — e.g., might create other metrics (i.e., in addition to stepsRemoved) that better reflect time-synchronization performance
  — something like this is needed for multi-path in the home, over wireless and powerline.
- asCapability observability
  — e.g., using one of the profile-specific flags for a node to indicate (via Pdelay) that it is asCapable
  — unlike current 802.1AS, would like to allow a slave-only ordinary clock to not be required to send Announce (already is not required to send Sync).

Z.3.2.2 Industrial Needs [7/11]

- An HRM (hypothetical reference model) of 64 hops [1/12]
- At most 512 devices off one controller
- Performance (believed to be currently met by AS, but should be verified from Garner simulations)
  — Any two nodes (separated by at most 64 hops) are synchronized to within +/- 1 us, with:
  • maximum 3ppm/sec rate of change of frequency for the local clock (oscillator in the node) - We may need a lower residence time - Need a frequency change vs time profile based on environmental effects (e.g., temperature, microphonics (vibration), voltage, radiation).
  • 125 MHz gPTP timestamp clock [1/12] (assume that the timestamp instants are synchronous with the clock significant instants at one end of the link)(**this is a fixed, agreed-upon value**)
  — Also need to consider:
  • PHY latency asymmetry (any remaining error after compensating for difference between reference plane and measurement plane)
  • copper (do this first, before fiber); 2 paramaters: (a) minimal mis-match (the spec of matched cables), 3 ns and (b) maximum mismatch, which is the CAT-5E spec itself (50 ns); both (a) and (b) are for the maximum copper cable length, of 100 m. (note: the 3 ns and 45 ns are maximum skew values, i.e., max difference in delay).
  • fiber
  • intra-node error
  — The above are for steady-state. Initially, we will consider steady-state, and after that rearrangements/transient conditions.

— Also will need to know maximum absolute value time error (max|TE|) between any two neighboring nodes

- Meet the Redundancy requirements per given presentations [1/12]
- The presentation “Industrial Requirements for Synchronization in Working Clock Domains” (Feng Chen and Franz-Josef Goetz, Siemens, July 2014) is used in the assumptions for the handling of multiple domains and redundancy (July, 2014). A figure similar to the slide 3 figure, showing 2 domains, was added in D0.6 (agreed in July 2014 that this would be done). Figures illustrating redundancy were prepared for D0.6, but needed to be reviewed as part of the D0.6 review.
- Need to bring the recovery times requirement from the presentation here from Oliver (Franz)
- Allow for heterogeneous networks (Dan)
  — some nodes are not PTP-capable (i.e., the network has partial rather than full timing support)
  — But, if another PTP profile meets this need, could use that
  — Coexistence or interoperability of different profiles
  1) we will have various applications that rely on different profiles that share the same physical network
  2) these would have to co-exist or interoperate
  3) Might have masters that support multiple domains simultaneously
  4) The P1588 Architecture SC plans to discuss this.

Z.3.3 gPTP Generation 2 IEEE 802.1ASbt

Z.3.3.1 gPTP New Work - in PAR (Editors note: as material is added to the draft, indicate for each relevant assumption where that material is).

- Support for Link Agg (IEEE 802.1AX)
- ~~Support for other media:~~
  ~~— IEEE 1901 (if changes are needed)~~
  ~~— WiFi Direct (if changes are needed)~~
  ~~— Others?~~
  —It was agreed to remove IEEE 1901 as of D2.0. Discussions much earlier indicated no changes would be needed for WiFi Direct and, in any case, no material or presentations on this have been submitted.
- Alternate Timescales (e.g., transport time zone information)
  — Agreed to two time domains (domain 0 for universal time; any one of the values 1 - 127 can be used for working clock)
- The next two bullet items and the sub-bullet of the second one were deleted January 21, 2014
- ~~One Step Tolerant on receive for Sync~~
- ~~One Step Tolerant on receive for Pdelay~~
  ~~— Need to define how the information is conveyed by the transmitter~~
- Look at improving performance for long daisy-chained time-aware systems (or long networks) that may be in a large ring
- Look at Faster Grand Master change over
  — Pre select a failover Grand Master so the selection when needed is faster
  — Support both Hot and Cold standbys [11/12]
  — It is assumed that when a GM fails, it either stays silent or the link to it has broken
- Diagnosis of failures other than fail-silent failures.
- How will information from redundant GMs (hot standby) be used
- From a systems perspective, how do the items of synchronization information from the different redundant GMs relate to each other (what are the failure modes of the redundant GMs)?
- Redundancy
  — Short reconfiguration w/redundant paths when one path fails
  — Look for the holes/issues in a redundant/failover system?

— Use non-congruent paths wherever possible (by non-congruent, we mean paths that have no nodes or links in common) <Editor's note: look at definition of "maximally disjoint paths in 3.10 of 802.1Qca/D0.4>

- Automatic measurement of link delay asymmetry
  — [7/2014] see "Measurement of Link Delay Asymmetry" -- Lu Huang and Liuyan Han (China Mobile), presented at July, 2014 TSN TG meeting, for assumptions on this item. The presentation is available at http://www.ieee802.org/1/files/public/docs2014/asbt-huang-measurement-of-link-delay-asymmetry-0714-v01.pdf. It was agreed that the timestamps t1, t2, t3, and t4 can be added to the managed objects, and the actual asymmetry computation can be done outside PTP using these timestamp values from different messages. The values are added to the managed objects, and an editor's note is added indicating they must be added to the MIB.]
- Detect buffered repeaters on other than 802.3 copper links
  — Add in a variable latency in the link delay as an enhanced mechanism? Maybe a MAC address discovery?
  — Need an alternate mechanism for long (fiber) links
- Create an Annex to show Grand Master Re-election time

Z.3.3.2 gPTP Possible New Work

- Reduce BMCA convergence time/work for large (>64 node) networks &/or when a loop exists (i.e., a quicker way to get to a backup GM)
  — Use IS-IS for this [5/12]? But maybe not part of Qca so gPTP can be self contained?
  — Find an alternath that is as disjoint as possible from the primary path
  — See Mick Seaman's work on loop detection – Will not consider unless a contribution is supplied [1/11]
  — Large (64+) node networks force a lot of data examinations
- Provide L2 timing information for revision of 1588-2008
  — Update L2 abstract interface information for 1588 revision
  — Goal is to meet/liaison with P1588 committee [1/12] Michael is it ?? !
- Work with P1588 committee to provide end-to-end quality information
  — Common Mean Link Delay Service interface and information exchange
  — Is there a way to report clock quality and how shall clock quality be defined over the path it took? [1/12]
- Remove the word 'bridge' from 802.1AS title & elsewhere [3/12]
  — It can work through routers too and other devices [3/12]
  — This will require a revision PAR to change the title [9/12; revised 9/13]
- AS Reconfiguration Times [1/12]
  — Define What a reconfiguration is?
    - Death of a Master
    - Loss of a path to a Master
    - Multiple simultaneous Masters?
    - Pre configured paths? (Franz)

  — Grand master change over time is 200 mSec [3/12]
    - Time interval between loss of old Grand Master and locking to the new GrandMaster by the slave?
    - Multiple Grand Masters transmit timing trees at exactly the same time?
- Add full (within reasonable limits) support for TLV rate changes from Slave port to its Master port & modify figure10-8 accordingly [5/12] (Don for a presentation on what reasonable limits are)
- Each bridge to report its worst case Time Stamp accuracy – i.e. what clock rate is it sampling on and what is the worst case uncertainty of PHYs, etc. [5/12]
- Fig 10-3 problem with current GM when its PRI 1's is downgraded lower than another node in the network? Not sure what it does. [5/12]
  — The current GM won't stop being the GM? [11/12]

- No requirement for a higher priority GM joining a network to 1st sync to the existing GM – this causes a jump in time – solve this? [11/12]
- Avoidance of BMCA Thrashing vs. Faster Switchover to a GM [11/12]
  — Need to know what is more important – or do we make this selectable – Default is?
- Improve error checking, for example, in the MDPdelayReq state machine (errored Pdelay_Resp_Follow_Up is not covered). [added 5/2014]
- After realizing that rcvdPdelayResp and rcvdPdelayRespFollowUp needed to be initialize in the MDPdelayReq state machine, it was decided that all the state machines should be checked to make sure all the necessary variables are initialized. [added 5/2014]
- Sync interval issue (maintenance item 138; see also the presentation "802.1ASbt presentation on Short Sync Interval issue for Clause 10 (Media Independent Layer)", Bob Noseworthy, May, 2014 (available at http://www.ieee802.org/1/files/public/docs2014/as-noseworthy-sync-interval-0514-v01.pdf). In the PortSyncSync state machine (see Figure 10-8), a slight mismatch in the upstream and local syncIntervalTimer can cause a number of Sync intervals to be 0.5 times the specified interval (i.e., 62.5 ms if the specified interval is the default 125 ms). A solution is identified in the presentation, in which the lower bound is changed to 0.7*syncInterval and the upper bound to 1.3*syncInterval. This will solve the problem, because now the upper bound is not nominally the same as the Sync interval itself. The change is to change the condition on the transition from SET_SYNC_RECEIPT_TIMEOUT_TIME to SEND_MD_SYNC to

  (((rcvdPSSync && (currentTime – lastSyncSentTime >= ~~0.5~~0.7*syncInterval) &&

      rcvdPSSyncPtr->localPortNumber != thisPort)) ||

      ((currentTime – lastSyncSentTime >= 1.3*syncInterval) &&

      (lastRcvdPortNum != thisPort))) &&

  portEnabled && pttPortEnabled && asCapable && selected~~Role~~State[thisPort] == MasterPort

  [This item was fixed 9/2014.]

- asCapable hair-trigger issue (see the presentation from the May, 2014 802.1 TSN TG meeting "802.1ASbt overview on asCapable hair-trigger issue for Clause 11 (MD layer for FullDup P2P links), Bob Noseworthy, available at http://www.ieee802.org/1/files/public/docs2014/as-noseworthy-ascapable-hairtrigger-0514-v01.pdf). This presentation identified situations where a single instance of not satifying the conditions for asCapable to be TRUE would lead to it being set to FALSE, which would subsequently lead to an SRPDomainBoundary to be detected and Talker Fails to be propagated. The presentation recommended adding some hysteresis, i.e., if asCapable has been TRUE, don't set it to FALSE based on a single instance of not satisfying the respective conditions. It was decided at the May meeting to add this issue to Annex Z for ASbt.

Z.3.3.3 gPTP - Won't Work On

- Security (need the requirements and level of needed security)
- Mapping between NTP & AS (applicable to 1588) ? Will not do [1/11]
- No One Step support on transmit
- How to assess the synchronization performance of a node
  — For certification ? Will not do (Jan 2011)

Z.2.4 Time-Aware Shaper (TAS) IEEE 802.1Qbv

Z.2.4.1 TAS Ideas [1 & 5/12]

- How does a Time~~-~~-Aware Network come up?
  — What if the GM changes & you get a step in time? [11/12]

Z.2.5 SRP Generation 2 Ideas

Z.2.5.1 SRP - Other Ideas

- 802.1AE (MACsec) environments?
  — Does the MACsec block cipher's variable delays affect gPTP timing accuracy?
    - This depends upon where MACSec vs. Time Stamping is performed (1/13)
    - Do we need to support all options via reporting accuracy & jitter? (1/13)