P802.1ASed/D2.0
December 13, 2024
(Amendment to IEEE Std 802.1AS™-202x, as modified by IEEE Draft P802.1ASds)

Draft Standard for Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications

Amendment: Fault-Tolerant Timing with Time Integrity

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Time-Sensitive Networking (TSN) Task Group of IEEE 802.1

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

As part of our IEEE 802® process, the text of the PAR (Project Authorization Request) and CSD (Criteria for Standards Development) is reviewed regularly to ensure their continued validity. A vote of "Approve" on this draft is also an affirmation that the PAR is still valid. It is included in these cover pages.

The text proper of this draft begins with the title page (1). The cover pages (a), (b), (c) etc. are for 802.1 WG information, and will be removed prior to Sponsor Ballot.

Editors' Foreword

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

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

This draft is prepared for initial Working Group ballot.

This draft is based on contributions provided by the 802.1 working group as listed on the task group web site:

https://1.ieee802.org/tsn/802-1ased/

Participation in 802.1 standards development

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

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

http://ieee802.org/1/

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

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

Abdul Jabbar
Editor, P802.1ASed
Email:jabbar@ge.com

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

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

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

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

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

The complete amendment PAR, as approved by IEEE NesCom 26th September 2024, can be found at https://www.ieee802.org/1/files/public/docs2024/ed-draft-PAR-0524-v01.pdf

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

Scope of the Proposed changes:

This amendment specifies protocols, processes, procedures, functions, mechanisms, and managed objects to enable fault-tolerant timing by increasing the availability of the time and adding time integrity. This is achieved using two or more generalized Precision Time Protocol (gPTP) domains, multiple time distribution paths, the local oscillator clock, and a time selection function with individual processes for times that have interdependencies and times that do not have interdependencies. Fault-tolerant timing includes fault-tolerant time generation and distribution.

Need for the Project:

Fault-tolerant timing with time integrity is needed in some applications that use time synchronization (e.g., aerospace onboard networks) to provide reliable time to vital time-sensitive applications.

Criteria for Standards Development:

The complete Criteria for Standards Development (CSD) can be found at:

https://mentor.ieee.org/802-ec/dcn/21/ec-21-0308-00-ACSD-p802-1asds.pdf

P802.1ASed/D2.0

December 13, 2024

(Amendment to IEEE Std 802.1AS™-202x, as modified by IEEE Draft P802.1ASds)

Draft Standard for Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications

Amendment: Fault-Tolerant Timing with Time Integrity

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

Sponsor
LAN/MAN Standards Committee
of the
IEEE Computer Society

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

Abstract: This amendment to IEEE Std 802.1AS™-2020 specifies protocols, processes, procedures, functions, mechanisms, and managed objects to enable fault-tolerant timing by increasing the availability of the time and adding time integrity. This is achieved using two or more generalized Precision Time Protocol (gPTP) domains, multiple time distribution paths, the local oscillator clock, and a time selection function with individual processes for times that have interdependencies and times that do not have interdependencies. Fault-tolerant timing includes fault-tolerant time generation and distribution.

Keywords: fault-tolerant timing, time selection function, dependent gPTP times, independent gPTP times

Important Notices and Disclaimers Concerning IEEE Standards Documents

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

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

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

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

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

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

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

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

Translations

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

Official statements

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

Comments on standards

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

Comments on standards should be submitted to the following address:

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

Laws and regulations

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

Copyrights

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

Photocopies

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

Updating of IEEE Standards documents

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

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

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

Errata

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

Patents

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

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

Participants

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

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

Glenn Parsons, *Chair*
Jessy Rouyer, *Vice Chair*
János Farkas, *TSN Task Group Chair*
Silvana Rodrigues, *Editor IEEE Std 802.1AS*
Abdul Jabbar, *Editor P802.1ASed*

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

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

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

Introduction

This introduction is not part of IEEE Std 802.1ASed™-20xx, IEEE Standard for Local and metropolitan area networks—Timing and Synchronization for Time-Sensitive Applications—Amendment: Fault-Tolerant Timing with Time Integrity

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

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

The third edition, IEEE Std 802.1AS-202x is a roll-up of IEEE Std 802.1AS-2020 with the corrigendum IEEE Std 802.1AS-2020/Cor1, and its amendments: IEEE Std 802.1ASdr, IEEE Std 802.1ASdn, and IEEE Std 802.1ASdm.

This amendment to IEEE Std 802.1AS-202x specifies protocols, processes, procedures, functions, mechanisms, and managed objects to enable fault-tolerant timing by increasing the availability of the time and adding time integrity. This is achieved using two or more generalized Precision Time Protocol (gPTP) domains, multiple time distribution paths, the local oscillator clock, and a time selection function with individual processes for times that have interdependencies and times that do not have interdependencies. Fault-tolerant timing includes fault-tolerant time generation and distribution.

Contents

List of figures

List of tables

Draft IEEE Standard for Local and metropolitan area networks—

Timing and Synchronization for Time-Sensitive Applications

Amendment: Fault-Tolerant Timing with Time Integrity

[This amendment is based on IEEE Std 802.1AS™-20xx (IEEE Std 802.1AS™-2020 Revision).

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

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

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

4. Acronyms and abbreviations

Insert the following acronyms into the existing list of acronyms:

DTSF	Dependent Time Selection Function
FTTM	Fault-Tolerant Timing Module
ITSF	Independent Time Selection Function
MVTISA	Mid-Value Time-Index Selection Algorithm
NQ	Not Qualified
TSF	Time Selection Function

5. Conformance

5.3 Time-aware system requirements

Insert the following at the end of 5.3.

An implementation of a time-aware system may support the Fault-Tolerant Timing Module, as specified in 20.3.

If YANG is supported with a remote management protocol and if the Fault-Tolerant Timing Module (see 20.3) is supported, the YANG data model ieee802-dot1as-fttm in 17.6.4 shall be supported.

6. Conventions

6.4 Data types and on-the-wire formats

6.4.3 Derived data type specifications

Insert the following subclauses after 6.4.3.8.

6.4.3.9 ffttmInputTrustStatus

The fttmTrustStatus type is an Enumerated value that holds a trust status of an an input ClockTarget Interface. The data type is as follows:

typedef Enumeration1 fttmInputTrustStatus;

— 0: TRUSTED
— 1: NOT_TRUSTED

6.4.3.10 fttmInterfaceValueArray

The fttmInterfaceValueArray type provides a UInteger32 value that corresponds to input ClockTarget interfaces x and y for a dependent time selection function (DTSF) or the independent time selection function (ITSF) of the fault-tolerant timing module (FTTM). The data type is as follows:

typedef UInteger32 fttmInterfaceValueArray [fttmNumActiveTimeIndexes] [fttmNumActiveTimeIndexes];

6.4.3.11 fttmOutputTrustState

The fttmOutputTrustState type is an Enumerated value that holds the output trust state of a time selection function (TSF) or of the FTTM. The data type is as follows:

typedef Enumeration3 fttmOutputTrustState;

— 000: NOT_TRUSTED (see 20.3.3.3)
— 001: TIME_TRUSTED (see 20.3.3.3)
— 010: FREQ_TRUSTED (see 20.3.3.3)
— 011: NOT_VALID (see 20.3.3.3)
— 100: TRUST_STATE_4 (for use by an alternate (i.e., non-default) time-index selection algorithm
— 101: TRUST_STATE_5 (for use by an alternate (i.e., non-default) time-index selection algorithm
— 110: TRUST_STATE_6 (for use by an alternate (i.e., non-default) time-index selection algorithm
— 111: TRUST_STATE_7 (for use by an alternate (i.e., non-default) time-index selection algorithm

6.4.3.12 fttmSelectedIndex (UInteger16)

The fttmSelectedIndex type gives the selected ClockTarget interface index for a DTSF or for the ITSF, with the following meanings.

— 0 to 255: The index number of the selected input ClockTarget interface
— 256 to 510: Reserved
— 511: The Not Qualified (NQ) index
— > 511: Reserved

6.4.3.13 fttmUint16NumActiveDtsfs

The fttmUint16NumActiveDtsfs type is a vector of UInteger16 values with NumActiveDtsfs members. The data type is as follows:

typedef UInteger16 fttmUint16NumActiveDtsfs [fttmNumActiveDtsfs];

The UInteger16 values have the following representations:

- 0 to 255: Represents the index number of a selected input ClockTarget interface
- 256 to 510: Reserved
- 511: Represents the NQ index
- > 511: Reserved

6.4.3.14 fttmUint16NumActiveTsfs

The fttmUint16NumActiveTsfs type is a vector of UInteger16 values with NumActiveDtsfs + 1 members. The data type is as follows:

typedef UInteger16 fttmUint16NumActiveTsfs [fttmNumActiveDtsfs + 1];

The UInteger16 values have the following representations:

- 1 to 127: Represents the index number of a selected input ClockTarget interface
- > 128: Reserved

6.4.3.15 fttmExtsfttmNumActiveTimeIndexes

The fttmExtsfttmNumActiveTimeIndexes type is a vector of ExtendedTimestamp values with fttmNumActiveTimeIndexes members. The data type is as follows:

typedef ExtendedTimestamp fttmExtsfttmNumActiveTimeIndexes [fttmNumActiveTimeIndexes];

6.4.3.16 fttmExtsfttmNumActiveTsfs

The fttmExtsfttmNumActiveTsfs type is a vector of ExtendedTimestamp values with fttmNumActiveTsfs members. The data type is as follows:

typedef ExtendedTimestamp fttmExtsfttmNumActiveTsfs [fttmNumActiveTsfs];

6.4.3.17 fttmOctet128NumActiveTsfs

The fttmOctet128NumActiveTsfs type is a vector of Octet128 values with fttmNumActiveDtsfs + 1 members. The data type is as follows:

typedef Octet128 fttmOctet128NumActiveTsfs [fttmNumActiveDtsfs + 1];

6.4.3.18 fttmUint8Uint8fttmNumActiveTimeIndexes

The fttmUint8Uint8fttmNumActiveTimeIndexes type is a vector of a pair of UInteger8 values with fttmNumActiveTimeIndexes members. The data type is as follows:

typedef UInteger8 UInteger8 fttmUint8Uint8fttmNumActiveTimeIndexes [fttmNumActiveTimeIndexes];

6.4.3.19 fttmUint32fttmNumActiveTimeIndexes

The fttmUint32fttmNumActiveTimeIndexes type is a vector of Uint32 values with fttmNumActiveTimeIndexes members. The data type is as follows:

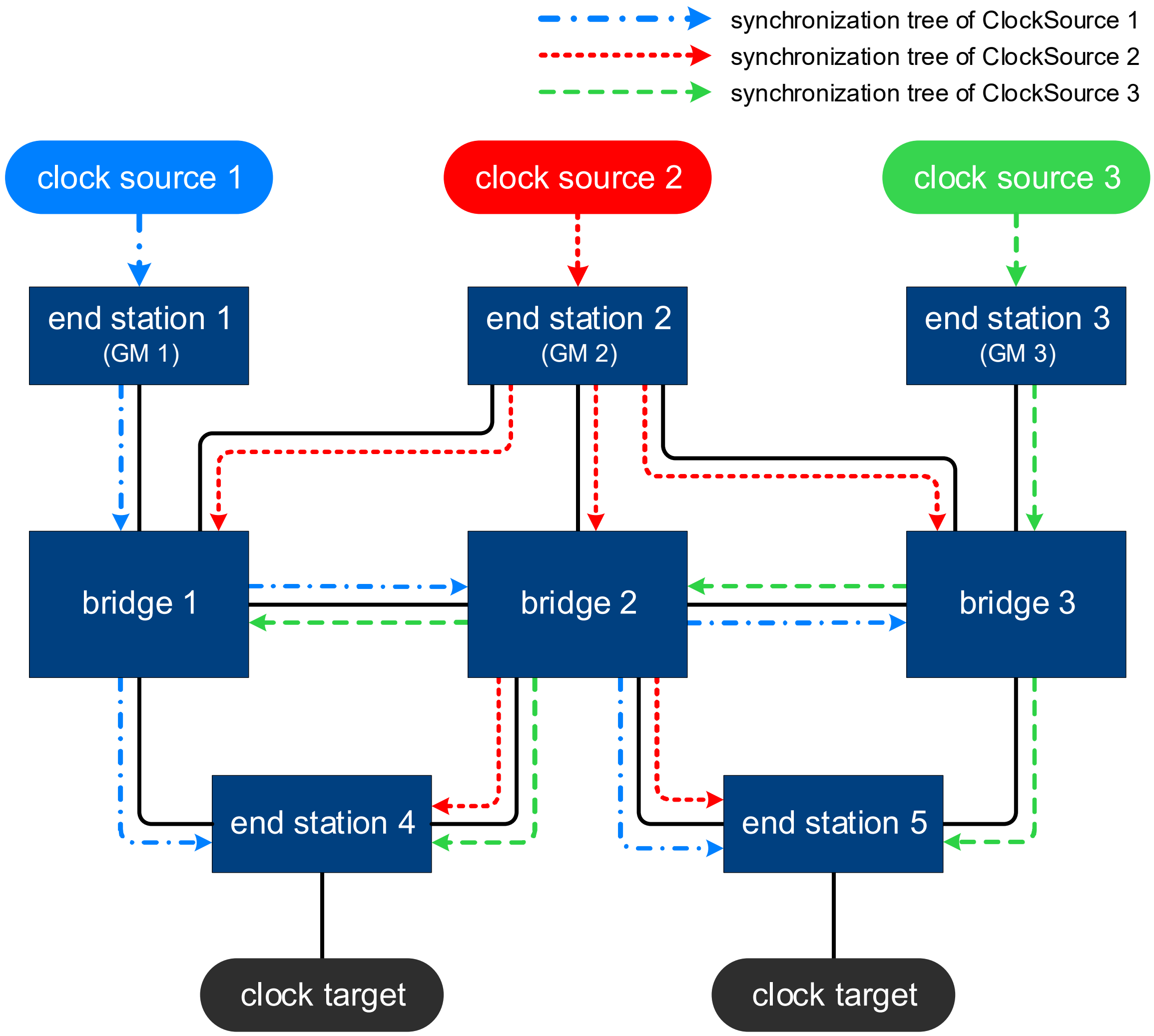
```
typedef UInteger32 fttmUint32fttmNumActiveTimeIndexes [fttmNumActiveTimeIndexes];
```

7. Time-synchronization model for a packet network

Add the following new subclause after 7.2.5, and renumber figures as necessary.

7.2.6 Time-aware network with fault-tolerant timing and time integrity

Figure 7-1 shows an example of a time-aware network with fault-tolerant timing and time integrity (see Clause 20). The network has GM redundancy and path redundancy to all the bridges and to end stations 4 and 5 to enhance availability and integrity of time in the network.



NOTE 1–All the “bridges” in this figure are examples of time-aware systems that contain PTP Relay Instances, PTP End Instances, and a fault-tolerant timing module

NOTE 2–All the end stations in this figure are examples of time-aware systems that contain PTP End Instances and fault-tolerant timing module

NOTE 3–The number of GMs and domains used in a implementation may vary based on the fault tolerance requirements

Figure 7-1—Time-aware network example for fault-tolerant timing with time integrity

In this example, three independent clock sources (1, 2 and 3) provide time to three GMs (1, 2 and 3 respectively). These clock sources are synchronized to each other using a time agreement generation and preservation function (see 20.2.3 and I.3). The synchronization spanning tree of each GM is illustrated by the arrows of each GM synchronization tree. The synchronization trees provide three times (from three

GMs) to all bridges and to non-GM end stations 4 and 5. Fault-tolerant timing modules (FTTMs, see 20.3) in the bridges and in end stations 4 and 5 select a fault-tolerant time from the 3 time inputs. The time selected by each FTTM has integrity. The fault-tolerant time is provided to the local clock targets for use by time-aware services, e.g., enhancements for scheduled traffic. For example, at end station 4, the time received from GM2 and GM3 share a dependency (see 20.2.5) because they both pass through bridge 2 in their synchronization trees. Conversely, nothing in the synchronization trees from GM2 and GM3 are shared with the synchronization tree from GM1 to end station 4 so the time from GM1 is independent of the other two times (see 20.2.6). Because of these relationships, the FTTM of end station 4 first selects between the times from GM2 and GM3 using a dependent time selection function (DTSF, see 20.3.2.2) and then selects between the DTSF's selected time and the time from GM1 using an independent time selection function (ITSF, see 20.3.2.2) to produce a fault-tolerant output time with time integrity. Similarly, the FTTM in bridge 2 sees three independent times and, thus, only uses its ITSF to select between them to produce a fault-tolerant output time with time integrity.

The network can tolerate failure of any one link or device as well as detect erroneous time on any one synchronization tree. For example, if the link between any two bridges (bridge 1 - 2 or bridge 2 - 3) fails, all non-GM end stations and bridges receive time over at least two synchronization trees and select a time using the ITSF. If bridge 1 experiences a fault and sends erroneous time over the GM 1 synchronization tree, all the non-GM end stations and bridges detect the erroneous time in comparison to the non-faulty time received over the other two synchronization trees. For the example shown in Figure 7-1, if any one bridge is faulty and introduces errors in the relayed synchronization tree(s), all the other non-faulty bridges and non-GM end stations are able to detect the erroneous times via the FTTM because the topology allows for at least one non-faulty synchronization tree in the network.

14. Timing and synchronization management

Insert the following subclauses after 14.22.

14.23 Fault-Tolerant Timing Module System Parameter Data Set (fttmSystemDS)

14.23.1 General

The fttmSystemDS describes the attributes of the respective instance of the Fault-Tolerant Timing Module Service.

14.23.2 dtsfMaxNumTimeIndexes (UInteger8)

The dtsfMaxNumTimeIndexes object gives the maximum number of input ClockTarget Interfaces available on each of the DTSFs in the FTTM.

The dtsfMaxNumTimeIndexes object is read-only, with a value from 1 to 127. The default value is implementation specific.

The dtsfMaxNumTimeIndexes object shall be provided if the FTTM service is used.

14.23.3 fttmCollectedTod (fttmExtsfttmNumActiveTimeIndexes)

The fttmCollectedTod object is a vector of collected extended timestamps that correspond to the latest ClockTarget invoke event. The vector member fttmCollectedTod[x] holds the latest timeReceiverTimeCallback result for input ClockTarget interface x to the FTTM.

The value of fttmCollectedToD object is valid when fttmInvokeStatusAvail is TRUE.

The fttmCollectedTod object is read-only, with a default value of 0 for every member of the vector.

The fttmCollectedToD object shall optionally be provided if the FTTM service is used.

14.23.4 fttmMapDtsfToItsf (fttmUint16NumActiveDtsfs)

The fttmMapDtsfToItsf object provides the mapping for all the FTTM's DTSF output ClockTarget Interfaces to the ITSF's input indexes. See J.7.

Each fttmMapDtsfToItsf[x] object, where x is a value from 1 to fttmNumActiveDtsfs (see 14.23.13), consists of the item:

— The index number of the ITSF's input ClockTarget Interface that the DTSF's output ClockTarget Interface is connected to.

Where the ITSF instance's input ClockTarget Interface's index values range from 1 to fttmNumActiveTimeIndexes (see 14.23.14), and a value of 0 means the DTSF's output ClockTarget Interface is not connected.

The fttmMapDtsfToItsf object is read/write, with a default value of 0 for all vector members.

The fttmMapDtsfToItsf object shall be provided if the FTTM service is used and if fttmMaxNumDtsfs is not equal to 0.

An example of a FTTM that has 2 DTSFs (fttmNumActiveDTSFs = 2) that has the following connections is given below:

— "The output of DTSF instance 1 connects to the ITSF input index 3.
— "The output of DTSF instance 2 connects to the ITSF input index 2.

The corresponding example fttmMapDtsfToItsf object would be as follows:

— "fttmMapDtsfToItsf[1] = 3 // DTSF #1's output to ITSF input index #3
— "fttmMapDtsfToItsf[2] = 2 // DTSF #2's output to ITSF input index #2

14.23.5 fttmMapIndexToTsf (fttmUint8Uint8fttmNumActiveTimeIndexes)

The fttmMapIndexToTsf object provides the mapping for all of the FTTM's input ClockTarget Interface index numbers to a Time Selection Function (TSF) instance number and its input ClockTarget Interface index number. See J.6.

Each fttmMapIndexToTsf[x] object, where x is a value from 1 to fttmNumActiveTimeIndexes (see 14.23.14) and is equal to the index number of the FTTM input ClockTarget Interface, consists of two items:

— The TSF instance number that the FTTM's input ClockTarget Interface is connected to.

Where the ITSF instance number is 0, and the DTSF instance numbers range from 1 to fttmMaxNumDtsfs (see 14.23.11).

— The index number of the DTSF's/ITSF's input ClockTarget Interface that the FTTM's input ClockTarget Interface is connected to.

Where: the DTSF instance's input ClockTarget Interface's index values range from 1 to dtsfMaxNumTimeIndexes (see 14.23.2), the ITSF instance's input ClockTarget Interface's index values range from 1 to fttmNumActiveTimeIndexes (see 14.23.14), and a value of 0 means that the DTSF's output ClockTarget Interface is not active.

The fttmMapIndexToTsf object is read/write, with a default value set of {0,0} for all vector members.

The fttmMapIndexToTsf object shall be provided if the FTTM service is used.

An example of a FTTM that uses 5 ClockTarget Interfaces (fttmNumActiveTimeIndexes = 5) and 2 DTSFs (fttmNumActiveDTSFs = 2) that has the following connections is given below:

— The first and second FTTM input ClockTarget Interfaces, with indexes 1 and 2, connect to DTSF instance 1's input indexes 1 and 2, respectively.
— The third and fourth ClockTarget Interfaces, with indexes 3 and 4, connect to DTSF instance 2's input indexes 1 and 2, respectively.
— The fifth ClockTarget Interface, with index 5, connects to the ITSF index 1.

The corresponding example fttmMapIndexToTsf object would be as follows:

— fttmMapIndexToTsf[1] = {1,1} //FTTM input index #1 to DTSF instance #1's input index #1
— fttmMapIndexToTsf[2] = {1,2} //FTTM input index #2 to DTSF instance #1's input index #2
— fttmMapIndexToTsf[3] = {2,1} //FTTM input index #3 to DTSF instance #2's input index #1
— fttmMapIndexToTsf[4] = {2,2} //FTTM input index #4 to DTSF instance #2's input index #2
— fttmMapIndexToTsf[5] = {0,1} //FTTM input index #5 to ITSF input index #1

The connections of the two DTSF's output ClockTarget interfaces to the ITSF's input ClockTarget interfaces would be defined by fttmMapDtsfToItsf (see 14.23.4).

14.23.6 fttmMapPtpInstanceToIndex (fttmUint32fttmNumActiveTimeIndexes)

The fttmMapPtpInstanceToIndex object provides the mapping of the instance index number of the PTP Instance to the FTTM input index number. See J.5.

Each fttmMapPtpInstanceToIndex[x] member, where x is a value from 1 to fttmNumActiveTimeIndexes (see 14.23.14) and represents the index number of the FTTM input ClockTarget interface, is a UInteger32 value that represents the instance index number of the PTP Instance that is connected to the FTTM input port with the input index number x.

The fttmMapPtpInstanceToIndex object is read/write and shall be provided if the FTTM service is used.

14.23.7 fttmHyst (fttmInterfaceValueArray)

This fttmHyst object is a two-dimensional array of UInteger32 values, each in units of 2^{-16} nanoseconds. The array has a size of fttmNumActiveTimeIndexes in each dimension. The object fttmHyst[x][y] holds the hysteresis to be added to fttmMaxAs[x][y] (see 14.23.10) for the times of the two FTTM input ClockTarget interfaces with index numbers x and y.

The hysteresis enables the use of one time skew level to assert the trust status and another time skew level to deassert the trust status. This can prevent superfluous changes in the trust state between the two input ClockTarget interfaces, x and y, when the times of those ClockTarget interfaces are close to the fttmMaxAs[x][y] threshold. The value of this hysteresis is to be determined by the user of the FTTM.

The fttmHyst object is read/write and has a default value of 0 for all array members.

The fttmHyst object shall be provided if the FTTM service is used.

14.23.8 fttmInvokeStatusAvail (Boolean)

The fttmInvokeStatusAvail object indicates that the FTTM has updated the values of the following read-only status objects, after an invoke event from the ClockTarget:

— fttmCollectedTod

When fttmInvokeStatusAvail is TRUE, the above read-only status objects have been updated.

When fttmInvokeStatusAvail is FALSE, the above read-only status objects have not been updated.

fttmInvokeStatusAvail is read-only and has a default value of FALSE.

fttmInvokeStatusAvail is cleared to FALSE by assertion of fttmInvokeStatusAvailClr to TRUE.

To detect each update, fttmInvokeStatusAvailClr must be asserted to TRUE to clear fttmInvokeStatusAvail before each update occurs.

The fttmInvokeStatusAvail object shall optionally be provided if the FTTM service is used.

14.23.9 fttmInvokeStatusAvailClr (Boolean)

The fttmInvokeStatusAvailClr object is used to clear the fttmInvokeStatusAvail object to FALSE. When fttmInvokeStatusAvailClr is changed from FALSE to TRUE, fttmInvokeStatusAvail is cleared to FALSE.

The fttmInvokeStatusAvailClr object must be written to FALSE before it can be used again.

The fttmInvokeStatusAvailClr object is read/write, with a default value of FALSE.

The fttmInvokeStatusAvailClr object shall optionally be provided if the FTTM service is used.

14.23.10 fttmMaxAs (fttmInterfaceValueArray)

The fttmMaxAs object is a two-dimensional array of UInteger32 values, each in units of 2^{-16} nanoseconds. The array has a size of fttmNumActiveTimeIndexes in each dimension. The fttmMaxAs[x][y] object gives the maximum magnitude of expected skew between times provided by the FTTM input ClockTarget interfaces of index x and index y when those times are not faulty. This value is used as the criteria to determine the trustworthiness of the times being compared. See $maxAS_{xy}$ in L.5.

The fttmMaxAs object is read/write and has a default value of 0 for all array members.

The fttmMaxAs object shall be provided if the FTTM service is used.

14.23.11 fttmMaxNumDtsfs (UInteger8)

The fttmMaxNumDtsfs objects gives the maximum number of DTSF instances available in the FTTM. The default value is implementation specific.

The fttmMaxNumDtsfs object is read-only, with a value from 0 to 126.

The fttmMaxNumDtsfs object shall be provided if the FTTM service is used.

14.23.12 fttmMaxNumTimeIndexes (UInteger8)

The fttmMaxNumTimeIndexes object gives the maximum number of input ClockTarget Interfaces available on the FTTM. The default value is implementation specific.

The fttmMaxNumTimeIndexes object is read-only, with a value from 1 to 255.

The fttmMaxNumTimeIndexes object shall be provided if the FTTM service is used.

14.23.13 fttmNumActiveDtsfs (UInteger8)

The fttmNumActiveDtsfs object gives the number of active DTSF instances currently used in the FTTM.

The fttmNumActiveDtsfs object is read-only, with a value from 0 to fttmMaxNumDtsfs.

The fttmNumActiveDtsfs object shall optionally be provided if the FTTM service is used.

14.23.14 fttmNumActiveTimeIndexes (UInteger8)

The fttmNumActiveTimeIndexes object gives the number of input ClockTarget Interfaces currently active on the FTTM, where an active input ClockTarget Interface is one that has been mapped to either a DTSF instance or to the ITSF.

The fttmNumActiveTimeIndexes object is read-only, with a value from 1 to fttmMaxNumTimeIndexes.

The fttmNumActiveTimeIndexes object shall optionally be provided if the FTTM service is used.

14.23.15 fttmSelChangeThresh (fttmExtsNumActiveTsfs)

The fttmSelChangeThresh object gives the time difference change threshold that is used by each TSF instance to determine whether it continues to use the previously selected time index or to change to the current time index that best satisfies the selection criteria (e.g., the time index that currently satisfies the mid-value selection criteria of the MVTISA, see 20.3.5).

If the following conditions are true, then the TSF retains the previously selected time index. Otherwise, it changes to the newly selected time index.

— The previously selected time index is still trusted.
— The magnitude of the difference in time (between the previously selected time index and the current time index that best satisfies the selection criteria) is less than or equal to the value of the fttmSelChangeThresh object for the TSF.

Each fttmSelChangeThresh[x] member, where x is a value from 0 to fttmNumActiveDtsfs (see 14.23.13) and represents the TSF instance number, is given in the format of the ExtendedTimestamp data type and has a default value of 0.

The TSF instance number is assigned as follows.

— The ITSF instance number is 0.
— The DTSF instance numbers range from 1 to fttmNumActiveDtsfs (see 14.23.13).

The fttmSelChangeThresh object is read/write.

The fttmSelChangeThresh object shall optionally be provided if the FTTM service is used.

14.23.16 fttmSelInstanceIndex (UInteger32)

The fttmSelInstanceIndex object gives the instanceIndex value of the PTP Instance that is the source of the ClockTargetEventCapture interface that was selected by the FTTM as its trusted time. The value is only valid if the FTTM output ClockTargetEventCapture has isSynced and gmPresent both equal to TRUE.

The fttmSelInstanceIndex object is read-only.

The fttmSelInstanceIndex object shall optionally be provided if the FTTM service is used.

14.23.17 fttmSelTimeIndexChangeCnt (Uinteger16)

The fttmItsfSelTimeIndexChangeCnt object gives the number of times the ITSF has changed its time index selection.

The fttmItsfSelTimeIndexChangeCnt object is read-only, with a value from 0 to 65535. The count rolls over to 0 if the count is incremented when its current value is 65535. The default value is 0.

The fttmItsfSelTImeIndexChangeCnt object shall optionally be provided if the FTTM service is used.

The fttmItsfSelTImeIndexChangeCnt object is used with fttmInvokeStatusAvail (see 14.23.8) and fttmInvokeStatusAvailClr (see 14.23.9).

14.23.18 fttmSelTimeRateRatioOffset (Uint32)

The fttmSelTimeRateRatioOffset object gives the maximum rate-ratio offset magnitude, between the FTTM's selected time clock rate and the FTTM's local clock (OSC_CLK) rate, that is deemed to be acceptable to go to or remain in the FREQ_TRUSTED state (see 20.3.3.3).

The rateRatio offset magnitude is expressed as a fractional frequency offset multiplied by 2^{41}.

The fttmSelTimeRateRatioOffset object is read/write and has a default rate-ratio value of 0 for all object members.

The fttmSelTimeRateRatioOffset object shall optionally be provided if the FTTM service is used.

14.23.19 fttmSelTimeStdDevRateRatioOffset (Uint32)

The fttmSelTimeStdDevRateRatioOffset object gives the maximum standard deviation of the rate-ratio offset, between the FTTM's selected time clock rate and the FTTM's local clock (OSC_CLK) rate, that is deemed to be acceptable to go to or remain in the FREQ_TRUSTED state (see 20.3.3.3).

The standard deviation of the rateRatio offset is expressed as a fractional frequency offset multiplied by 2^{41}.

The fttmSelTimeStdDevRateRatioOffset object is read/write and has a default rate-ratio value of 0 for all object members.

The fttmSelTimeStdDevRateRatioOffset object shall optionally be provided if the FTTM service is used.

14.23.20 fttmTrustState (fttmOutputTrustState)

The fttmTrustState object holds the output trust state of the FTTM. Valid values are NOT_TRUSTED, TIME_TRUSTED, FREQ_TRUSTED, and NOT_VALID.

The fttmTrustState object is read-only.

The fttmTrustState object shall optionally be provided if the FTTM service is used.

14.23.21 fttmTsfAlgoName (fttmOctet128NumActiveTsfs)

The fttmTsfAlgoName object is a vector of strings, where each vector member provides the name of the algorithm used by each active TSF instance number (the ITSF and the active DTSFs).

Each fttmTsfAlgoName[x] member, where x is a value from 0 to fttmNumActiveDtsfs (see 14.23.13) and represents the TSF instance number, is a 128 octet that contains a string with the name of the algorithm for the TSF instance number.

The TSF instance number is assigned as follows.

— The ITSF instance number is 0.
— The DTSF instance numbers range from 1 to fttmNumActiveDtsfs (see 14.23.13).

The fttmTsfAlgoName object is read-only.

The default TSF algorithm and its name are discussed in 20.3.5.

The fttmTsfAlgoName object shall optionally be provided if the FTTM service is used.

14.23.22 fttmTsfSelTimeIndex (fttmUint16NumActiveTsfs)

The fttmTsfSelTimeIndex object gives the input time index that is selected by each TSF instance.

Each fttmTsfSelTimeIndex[x] member, where x is a value from 0 to fttmNumActiveDtsfs (see 14.23.13) and represents the TSF instance number, is read-only and has a default value of 511 (i.e., the NQ index).

The TSF instance number is assigned as follows.

— The ITSF instance number is 0.
— The DTSF instance numbers range from 1 to fttmNumActiveDtsfs (see 14.23.13).

The fttmTsfSelTimeIndex object is read-only.

The fttmTsfSelTimeIndex object shall optionally be provided if the FTTM service is used.

14.23.23 fttmUseOscClk (Boolean)

The fttmUseOscClk object defines whether the OSC_CLK frequency is used as a reference for time integrity.

If fttmUseOscClk is TRUE, then the OSC_CLK frequency is used as a reference for checking time integrity (e.g., for entering the FREQ_TRUSTED state in the FTTM state machine, per 20.3.3.3).

If fttmUseOscClk is FALSE, then the OSC_CLK frequency is not used as a reference for checking time integrity.

The fttmUseOscClk object is read-only and has an implementation-specific default value.

The fttmUseOscClk object shall optionally be provided if the FTTM service is used.

14.24 Fault-Tolerant Timing Module System Description Parameter Data Set (fttmSystemDescriptionDS)

14.24.1 General

The fttmSystemDescriptionDS contains descriptive information for the respective instance of the Fault-Tolerant Timing Module Service.

14.24.2 userDescription

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

14.24.3 fttmSystemDescriptionDS table

There is one fttmSystemDescriptionDS table per fttmSystem instance, as detailed in Table 14-1.

Table 14-1—fttmSystemDescriptionDS table

Name	Data type	Operations supported[1]	References
userDescription	Octet128	RW	14.24.2

[1]RW = Read/Write access

17. YANG framework

17.1 YANG framework

17.1.1 Relationship to the IEEE Std 1588 data model

Change the first paragraph in 17.1.1 as follows:

The YANG data models specified in this standard are based on, and augment, those specified in IEEE Std 1588. In particular the ieee802-dot1as-gptp.yang module imports the ieee1588-ptp-tt module as a whole, augmenting that module as necessary to meet the requirements of this standard. ~~In addition, t~~The ieee802-dot1as-hs.yang module imports the ieee1588-ptp-tt and ieee802-dot1as-gptp modules as a whole, augmenting those modules as necessary to meet the requirements of this standard. ~~Also, t~~he ieee802-dot1as-hd.yang module imports the ieee1588-ptp-tt, the ieee802-dot1as-gptp, and the ieee802-dot1as-hs modules as a whole, augmenting those modules as necessary to meet the requirements of this standard. The ieee802dot1as-fttm.yang modules imports the ieee1588-tt and ieee802-dot1as-gptp modules as a whole, augmenting those modules as necessary to meet the requirements of this standard.

Change the fourth paragraph in 17.1.1 as follows:

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

17.2 IEEE 802.1AS YANG model

Change the following paragraph as shown:

Figure 17-4 provides detail for the common services, including each data set member. The Common Mean Link Delay Service (cmlds) has a data set~~s~~ for the service itself (e.g., default-ds), and data sets for each PTP Link Port. The Hot Standby Service has data sets for each HotStandbySystem. The Fault-Tolerant Timing Module Service has data sets for each fttmSystem.

1 ***Replace Figure 17-1 with the following:***

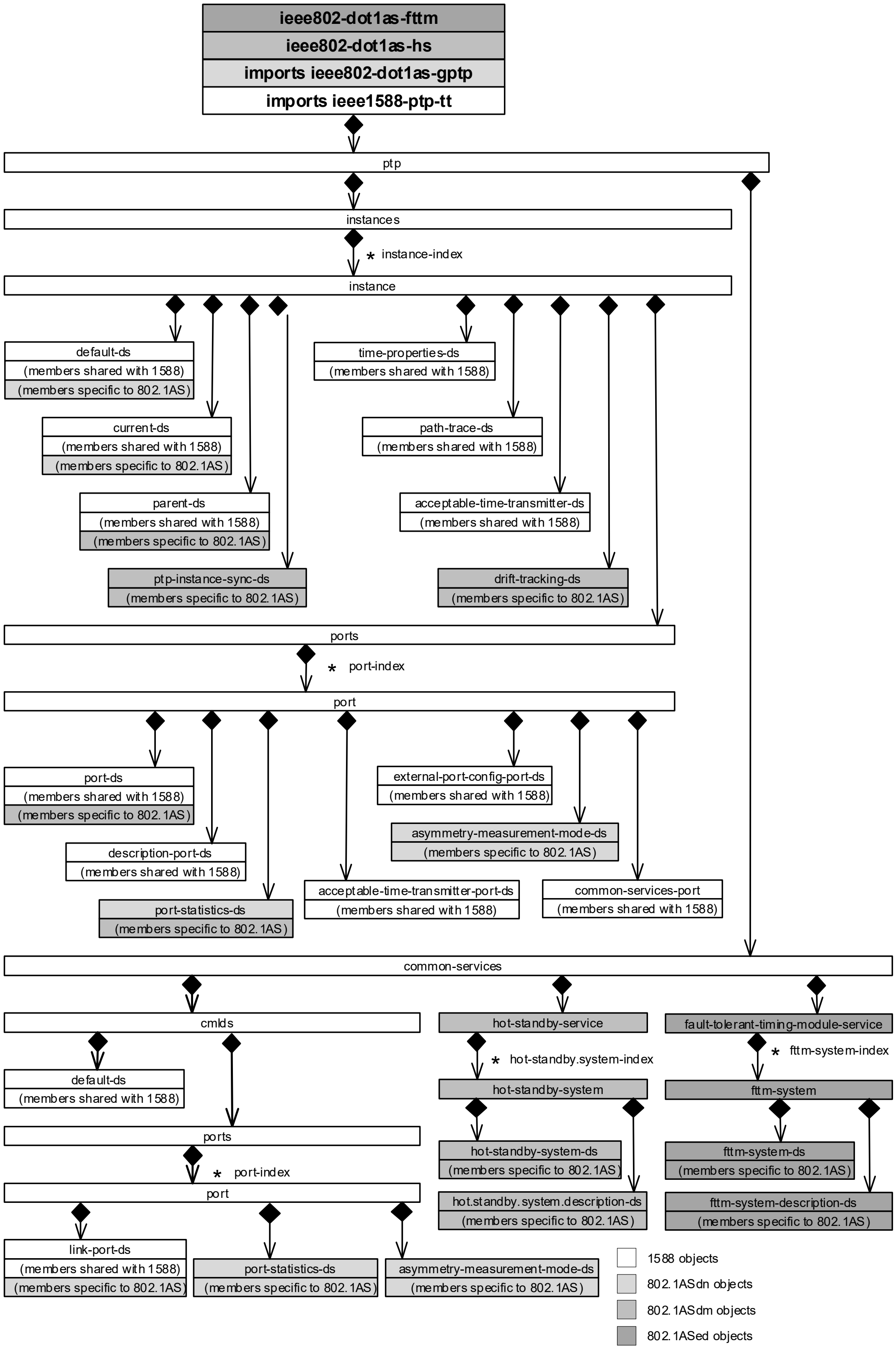


Figure 17-1—Overview of YANG tree

Replace Figure 17-4 with the following:

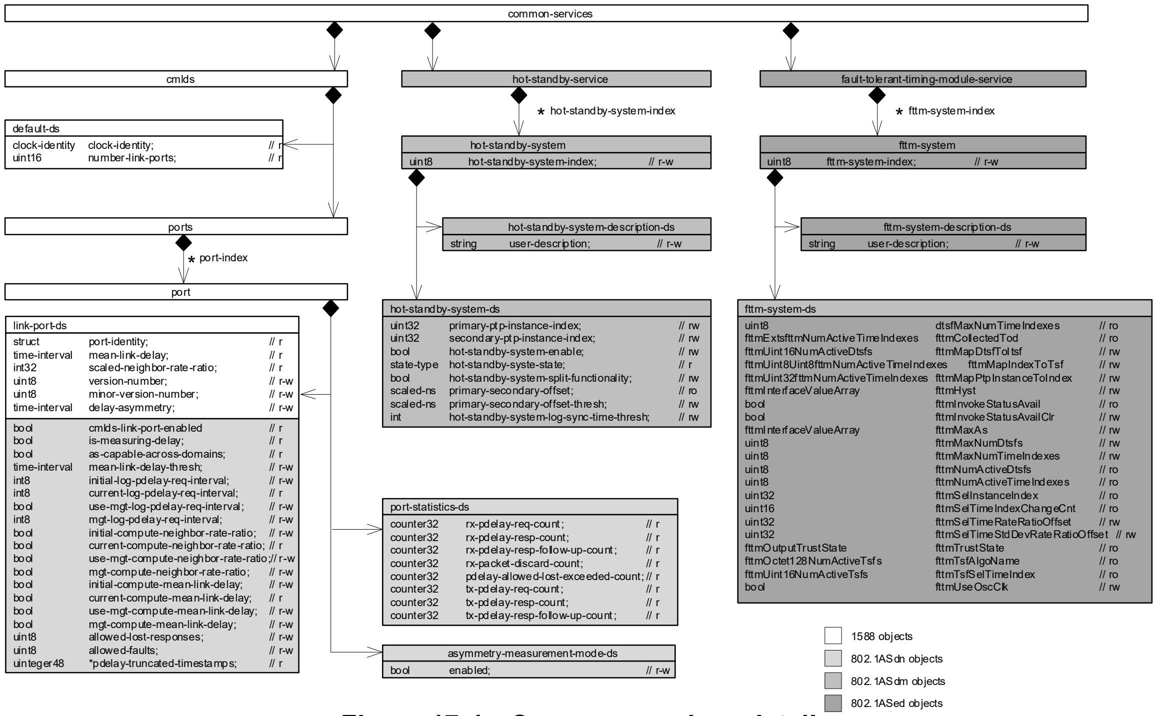


Figure 17-4—Common services detail

17.3 Structure of YANG data model

Change Table 17-1 as follows:

Table 17-1—Summary of the YANG modules

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

Table 17-1—Summary of the YANG modules

Module	Managed functionality	YANG specification notes
ieee802-dot1as-hd	Clause 14	IEEE Std 802.1ASds - YANG Data Model. The YANG module of this clause uses YANG augments to add members from Clause 14 that unique to IEEE Std 802.1ASds
ieee802-dot1as-fttm	Clause 14	IEEE Std 802.1ASed - YANG Data Model. The YANG module of this clause uses YANG augments to add members from Clause 14 that are unique to IEEE Std 802.1ASed.

1 17.5 YANG schema tree definitions

2 ***Insert the following subclause at the end of 17.5:***

3 17.5.4 Tree diagram for ieee802-dot1as-fttm.yang

4
```
module: ieee802-dot1as-fttm

  augment /ptp-tt:ptp/ptp-tt:common-services:
    +--rw fault-tolerant-timing-module-service {fttm}?
       +--rw fttm-system* [fttm-system-index]
          +--rw fttm-system-index             uint8
          +--rw fttm-system-ds
          |  +--ro dtsf-max-num-time-indexes?                 uint8
          |  +--rw fttm-collected-tod-list* [fttm-input-index-number]
          |  |  +--rw fttm-input-index-number    uint8
          |  |  +--ro extended-timestamp-list* [seconds fractional-nanoseconds]
          |  |     +--ro seconds                uint48
          |  |     +--ro fractional-nanoseconds    uint48
          |  +--rw fttm-map-dtsf-to-itsf-list* [tsf-instance-number]
          |  |  +--rw tsf-instance-number        uint8
          |  |  +--rw itsf-input-index-number?   uint8
          |  +--rw fttm-map-index-to-tsf-list* [fttm-input-index-number]
          |  |  +--rw fttm-input-index-number    uint8
          |  |  +--rw tsf-instance-number?       uint8
          |  |  +--rw tsf-input-index-number?    uint8
          |  +--rw fttm-map-ptp-instance-to-index-list* [fttm-input-index-number]
          |  |  +--rw instance-index?            uint32
          |  |  +--rw fttm-input-index-number    uint8
          |  +--rw fttm-hyst-lists* [fttm-input-index-number]
          |  |  +--rw fttm-input-index-number    uint8
          |  |  +--rw fttm-hyst-list* [fttm-input-index-number]
          |  |     +--rw fttm-input-index-number    uint8
          |  |     +--rw fttm-hyst?                 uint32
          |  +--ro fttm-invoke-status-avail?                  boolean
          |  +--rw fttm-invoke-status-avail-clr?              boolean
          |  +--rw fttm-max-as-lists* [fttm-input-index-number]
          |  |  +--rw fttm-input-index-number    uint8
          |  |  +--rw fttm-max-as-list* [fttm-input-index-number]
          |  |     +--rw fttm-input-index-number    uint8
          |  |     +--rw fttm-max-as?               uint32
          |  +--ro fttm-max-num-dtsfs?                        uint8
          |  +--ro fttm-max-num-time-indexes?                 uint8
          |  +--ro fttm-num-active-dtsfs?                     uint8
          |  +--ro fttm-num-active-time-indexes?              uint8
          |  +--rw fttm-sel-change-thresh-list* [tsf-instance-number]
          |  |  +--rw tsf-instance-number        uint8
          |  |  +--rw extended-timestamp-list* [seconds fractional-nanoseconds]
          |  |     +--rw seconds                uint48
          |  |     +--rw fractional-nanoseconds    uint48
          |  +--ro fttm-sel-instance-index?                   uint32
          |  +--ro fttm-sel-time-index-change-cnt?            uint16
          |  +--rw fttm-sel-time-rate-ratio-offset?           uint32
          |  +--rw fttm-sel-time-std-dev-rate-ratio-offset?   uint32
          |  +--ro fttm-trust-state?                          fttm-output-trust-state
          |  +--rw fttm-tsf-algo-name-list* [tsf-instance-number]
          |  |  +--rw tsf-instance-number    uint8
```

```
      |  |  +--ro fttm-tsf-algo-name?    string
      |  +--rw fttm-tsf-sel-time-index-list* [tsf-instance-number]
      |  |  +--rw tsf-instance-number        uint8
      |  |  +--ro fttm-tsf-sel-time-index?   uint16
      |  +--ro fttm-use-osc-clk?                          boolean
      +--rw fttm-system-description-ds
         +--rw user-description?   string
```

17.6 YANG modules

Insert the following subclause:

17.6.4 Module ieee802-dot1as-fttm.yang

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

  import ieee1588-ptp-tt {
    prefix ptp-tt;
  }

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

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

    E-mail: stds-802-1-chairs@ieee.org";
  description
    "This module provides for management of IEEE Std 802.1ASed components
    that support a fault-tolerant timing module.

    Copyright (C) IEEE (2024).
    This version of this YANG module is part of IEEE Std 802.1AS;
    see the standard itself for full legal notices.";

  revision 2024-09-02 {
    description
      "Published as part of IEEE Std 802.1ASed-2024.

      The following reference statement identifies each referenced IEEE
      Standard as updated by applicable amendments.";
    reference
      "IEEE Std 802.1AS-2020 as modified by
       IEEE Std 802.1AS-2020/Cor-1-2021, and amended by
       IEEE Std 802.1ASdr, IEEE Std 802.1ASdn,
       IEEE Std 802.1ASdm,IEEE Std 802.1ASds, and IEEE Std 802.1ASed.";
  }


  feature fttm {
    description
      "This feature indicates that the device supports the Fault-tolerant
       timing module (FTTM) functionality.";
  }

  typedef fttm-output-trust-state {
    type enumeration {
      enum NOT-TRUSTED {
        value 0;
        description
          "Not trusted";
      }
      enum TIME-TRUSTED {
        value 1;
        description
```

```
            "Time trusted";
        }
        enum FREQ-TRUSTED {
          value 2;
          description
            "Frequency trusted";
        }
        enum NOT_VALID {
          value 3;
          description
            "The trust state is not valid";
        }
        enum TRUST-STATE-4 {
          value 4;
          description
            "Trust state 4, reserved";
        }
        enum TRUST-STATE-5 {
          value 5;
          description
            "Trust state 5, reserved";
        }
        enum TRUST-STATE-6 {
          value 6;
          description
            "Trust state 6, reserved";
        }
        enum TRUST-STATE-7 {
          value 7;
          description
            "Trust state 7, reserved";
        }
      }
      description
        "The fttmOutputTrustState type is an enumerated value that holds
        the output trust state of a TSF or of the FTTM.";
      reference
        "6.4.3.11 of IEEE Std 802.1ASed";
    }

    typedef uint48 {
      type uint64 {
        range "0..281474976710655";
      }
      description
        "Unsigned 48-bit integer.";
    }

    grouping fault-tolerant-timing-module-group {
      description
        "Management of a single FTTM.";
      reference
        "14.23 of IEEE Std 802.1ASed";
      leaf dtsf-max-num-time-indexes {
        type uint8;
        config false;
        description
          "Implementation-specific. Gives the maximum number of input
          ClockTarget Interfaces available on each of the DTSFs in the
          FTTM.";
        reference
          "IEEE Std 802.1ASed 14.23.2";
      }
      uses fttm-collected-tod-group;
      uses fttm-map-dtsf-to-itsf-group;
      uses fttm-map-index-to-tsf-group;
      uses fttm-map-ptp-instance-to-index-group;
      uses fttm-hyst-group;

      leaf fttm-invoke-status-avail {
        type boolean;
        default "false";
        config false;
        description
          "The fttmInvokeStatusAvail object indicates that the FTTM has
          updated the values of the fttmCollectedTod status, after an
          invoke event from the ClockTarget";
        reference
          "14.23.8 of IEEE Std 802.1ASed";
      }
      leaf fttm-invoke-status-avail-clr {
```

```
      type boolean;
      default "false";
      description
        "The fttmInvokeStatusAvailClr object is used to clear the
        fttmInvokeStatusAvail object";
      reference
        "IEEE 802.1ASed 14.23.9";
    }
    uses max-as-group;
    leaf fttm-max-num-dtsfs {
      type uint8 {
        range "0..126";
      }
      config false;
      description
        "Implementation-specific. Maximum number of DTSF instances
        available in the FTTM.
        The value is restricted to the range of 0 to 126.";
      reference
        "IEEE Std 802.1ASed 14.23.11";
    }
    leaf fttm-max-num-time-indexes {
      type uint8 {
        range "0..255";
      }
      config false;
      description
        "Implementation-specific. Maximum number of input ClockTarget
        Interfaces available on the FTTM.";
      reference
        "IEEE Std 802.1ASed 14.23.12";
    }
    leaf fttm-num-active-dtsfs {
      type uint8;
      config false;
      description
        "Number of active DTSF instances currently used in the FTTM.";
      reference
        "IEEE Std 802.1ASed 14.23.13";
    }
    leaf fttm-num-active-time-indexes {
      type uint8;
      config false;
      description
        "The number of input ClockTarget Interfaces currently active on
        the FTTM.";
      reference
        "IEEE Std 802.1ASed 14.23.14";
    }
    leaf fttm-sel-instance-index {
      type uint32;
      config false;
      description
        "Gives the instanceIndex value of the PTP Instance that is the
        source of the ClockTarget interface that was selected by the FTTM
        as its trusted time.";
      reference
        "IEEE Std 802.1ASed 14.23.15";
    }
    leaf fttm-sel-time-index-change-cnt {
      type uint16;
      config false;
      description
        "The fttmItsfSelTimeIndexChangeCnt object gives the number of times
        the ITSF has changed its time index selection.";
      reference
        "IEEE 802.1ASed 14.23.16";
    }
    leaf fttm-sel-time-rate-ratio-offset {
      type uint32;
      default "0";
      description
        "The fttmSelTimeRateRatioOffset object gives the maximum rate-ratio
        offset magnitude, between the FTTM's selected time clock rate and
        the FTTM's local clock (OSC_CLK) rate, that is deemed to be acceptable
        to go to or remain in the FREQ_TRUSTED state (see 20.3.3.3).";
      reference
        "IEEE Std 802.1ASed 14.23.17";
    }
    leaf fttm-sel-time-std-dev-rate-ratio-offset {
      type uint32;
```

```
        default "0";
        description
          "The fttmSelTimeStdDevRateRatioOffset object gives the maximum standard
          deviation of the rate-ratio offset, between the FTTM’s selected time
          clock rate and the FTTM’s local clock (OSC_CLK) rate, that is deemed to be
          acceptable to go to or remain in the FREQ_TRUSTED state (see 20.3.3.3).";
        reference
          "IEEE Std 802.1ASed 14.23.19";
      }
      leaf fttm-trust-state {
        type fttm-output-trust-state;
        default "NOT-TRUSTED";
        config false;
        description
          "The fttmTrustState object holds the output trust state of the FTTM.
           Valid values are NOT_TRUSTED, TIME_TRUSTED, FREQ_TRUSTED, and NOT_VALID.";
        reference
          "IEEE Std 802.1ASed 14.23.20";
      }
      uses fttm-sel-change-thresh-group;
      uses fttm-tsf-algo-name-group;
      uses fttm-tsf-sel-time-index-group;
      leaf fttm-use-osc-clk {
        type boolean;
        config false;
        description
          "implementation-specific. Defines whether the OSC_CLK frequency is
          used as a reference for time integrity.";
        reference
          "IEEE Std 802.1ASed 14.23.23";
      }
    }

    grouping instance-index-group {
      description
        "An index number that identifies a ptp-instance input to the FTTM
        from the 1588e YANG module.";
      reference
        "IEEE Std IEEE Std 1588e-2024 15.3.3.2 Structure";
      leaf instance-index {
        type uint32;
        description
          "An index number that identifies a ptp-instance input to the FTTM
          from the 1588e YANG module.";
        reference
          "IEEE Std IEEE Std 1588e-2024 15.3.3.2 Structure";
      }
    }

    grouping fttm-collected-tod-group {
      description
        "The fttmCollectedTod object is a vector of collected extended timestamps
        that correspond to the latest ClockTarget invoke event. The vector member
        fttmCollectedTod[x] holds the latest timeReceiverTimeCallback result for
        input ClockTarget interface x to the FTTM.";
      reference
        "IEEE Std 802.1ASed 14.23.3";
      list fttm-collected-tod-list {
        key "fttm-input-index-number";
        description
          "The fttmCollectedTod object is a vector of collected extended
          timestamps that correspond to the latest ClockTarget invoke event.
          The vector member fttmCollectedTod[x] holds the latest
          timeReceiverTimeCallback result for input ClockTarget
          interface x to the FTTM.";
        reference
          "IEEE Std 802.1ASed 14.23.3.";
        leaf fttm-input-index-number {
          type uint8;
          description
            "The FTTM input index number.";
          reference
            "IEEE Std 802.1ASed 14.23.3";
        }
        list extended-timestamp-list {
          key "seconds fractional-nanoseconds";
          config false;
          description
            "The ExtendedTimestamp type represents a positive time with
            respect to the epoch.";
          reference
```

```
          "IEEE 802.1AS-2020 6.4.3.5";
        leaf seconds {
          type uint48;
          description
            "The integer portion of the timestamp in units of seconds.";
        }
        leaf fractional-nanoseconds {
          type uint48;
          description
            "The fractional portion of the timestamp in units of 2^-16 ns.";
        }
      }
    }
  }

  grouping fttm-map-dtsf-to-itsf-group {
    description
      "The fttmMapDtsfToItsf object provides the mapping for all the FTTM's
      DTSF output ClockTarget Interfaces to the ITSF and its input indexes.
      See J.7.";
    reference
      "IEEE Std 802.1ASed 14.23.4";
    list fttm-map-dtsf-to-itsf-list {
      key "tsf-instance-number";
      description
        "This grouping allows associations of all the DTSF output ClockTarget
        interfaces to ITSF input index numbers.";
      reference
        "IEEE Std 802.1ASed 14.23.4";
      leaf tsf-instance-number {
        type uint8;
        description
          "The DTSF instance number.";
        reference
          "IEEE Std 802.1ASed 14.23.4";
      }
      leaf itsf-input-index-number {
        type uint8;
        default "0";
        description
          "The ITSF's input ClockTarget index number. ";
        reference
          "IEEE Std 802.1ASed 14.23.4";
      }
    }
  }

  grouping fttm-map-index-to-tsf-group {
    description
      "The fttmMapIndexToTsf object provides the mapping for all of the
      FTTM's input ClockTarget Interface index numbers to a Time Selection
      Function (TSF) instance number and its input ClockTarget Interface
      index number. See J.6.";
    reference
      "IEEE Std 802.1ASed 14.23.5";
    list fttm-map-index-to-tsf-list {
      key "fttm-input-index-number";
      description
        "This grouping allows associations of the FTTM input index numbers
        to DTSF input index numbers and ITSF input index numbers.";
      reference
        "IEEE Std 802.1ASed 14.23.5";
      leaf fttm-input-index-number {
        type uint8;
        description
          "The FTTM input index number.";
        reference
          "IEEE Std 802.1ASed";
      }
      leaf tsf-instance-number {
        type uint8;
        description
          "The TSF instance number that the FTTM input index number is
          connected to. A value of 0 represents the ITSF. Other values
          represent the DTSF instance numbers.";
        reference
          "IEEE Std 802.1ASed 14.23.5";
      }
      leaf tsf-input-index-number {
        type uint8;
        description
```

```
          "The input index number of the TSF (DTSF or ITSF)";
        reference
          "IEEE Std 802.1ASed 14.23.5 ";
      }
    }
  }

  grouping fttm-map-ptp-instance-to-index-group {
    description
      "The fttmMapPtpInstanceToIndex object provides the mapping of the
      instance index number of the PTP Instance to the FTTM input index
      number.";
    reference
      "IEEE Std 802.1ASed 14.23.6";
    list fttm-map-ptp-instance-to-index-list {
      key "fttm-input-index-number";
      description
        "This grouping allows associations of the index numbers of PTP
        Instances to FTTM input index numbers.";
      reference
        "IEEE Std 802.1ASed 14.23.6";
      uses instance-index-group;
      leaf fttm-input-index-number {
        type uint8;
        description
          "The FTTM input index number for the FTTM input ClockTarget
           Interface associated with the PTP Instance with the
           corresponding index number.";
        reference
          "IEEE Std 802.1ASed 14.23.6";
      }
    }
  }

  grouping fttm-hyst-group {
    description
      "The fttmHyst object holds the hysteresis to be added to fttmMaxAs[x][y]
      (see 14.23.10) for the times of the two input FTTM input ClockTarget
      interfaces with index numbers x and y.";
    reference
      "IEEE Std 802.1ASed 14.23.7";
    list fttm-hyst-lists {
      key "fttm-input-index-number";
      description
        "The x index.";
      reference
        "IEEE Std 802.1ASed 14.23.7";
      leaf fttm-input-index-number {
        type uint8;
        description
          "The first FTTM input index number of the two-dimensional array .";
        reference
          "IEEE Std 802.1ASed 14.23.7";
      }
      list fttm-hyst-list {
        key "fttm-input-index-number";
        description
          "The y index.";
        reference
          "IEEE Std 802.1ASed 14.23.7";
        leaf fttm-input-index-number {
          type uint8;
          description
            "The second FTTM input index number of the two-dimensional array .";
          reference
            "IEEE Std 802.1ASed 14.23.7";
        }
        leaf fttm-hyst {
          type uint32;
          units "2^-16 nanoseconds";
          default "0";
          description
            "The object fttmHyst[x][y] holds the hysteresis to be added to
            fttmMaxAs[x][y] (see 14.23.13) for the times of the two input
            FTTM input ClockTarget interfaces with index numbers x and y.";
          reference
            "IEEE Std 802.1ASed 14.23.7 ";
        }
      }
    }
  }
```

1

```
  grouping max-as-group {
    description
      "The fttmMaxAs[x][y] object gives the maximum magnitude of expected
      skew between times provided by the FTTM input ClockTarget interfaces
      of index x and index y when those times are not faulty. This value
      is used as the criteria to determine the trustworthiness of the times
      being compared. See maxASxy in M.5";
    reference
      "IEEE Std 802.1ASed 14.23.10";
    list fttm-max-as-lists {
      key "fttm-input-index-number";
      description
        "The index x.";
      reference
        "IEEE Std 802.1ASed 14.23.10";
      leaf fttm-input-index-number {
        type uint8;
        description
          "The first FTTM input index number of the two-dimensional array.";
        reference
          "IEEE Std 802.1ASed 14.23.10";
      }
      list fttm-max-as-list {
        key "fttm-input-index-number";
        description
          "The index y.";
        reference
          "IEEE Std 802.1ASed 14.23.10";
        leaf fttm-input-index-number {
          type uint8;
          description
            "The second FTTM input index number of the two-dimensional array.";
          reference
            "IEEE Std 802.1ASed 14.23.10";
        }
        leaf fttm-max-as {
          type uint32;
          units "2^-16 nanoseconds";
          default "0";
          description
            "The maximum magnitude of expected skew between times provided
            by the input ClockTarget interfaces of index x and index y when
            those times are not faulty.";
          reference
            "IEEE Std 802.1ASed 14.23.10";
        }
      }
    }
  }

  grouping fttm-sel-change-thresh-group {
    description
      "The fttmSelChangeThresh object gives the time difference change
      threshold that is used by each TSF instance to determine whether
      it continues to use the previously selected time index or to change
      to the current time index that best satisfies the selection criteria
      (e.g., the time index that currently satisfies the mid-value selection
      criteria of the MVTISA, see 20.3.5).";
    reference
      "IEEE Std 802.1ASed 14.23.15";
    list fttm-sel-change-thresh-list {
      key "tsf-instance-number";
      description
        "The fttmSelChangeThresh object gives the time difference change
        threshold that is used by each TSF instance to determine whether
        it continues to use the previously selected time index or to change
        to the current time index that best satisfies the selection criteria.";
      reference
        "IEEE Std 802.1ASed 14.23.15";
      leaf tsf-instance-number {
        type uint8;
        description
          "The TSF instance number that the fttmSelChangeThresh object
          is associated with. A value of 0 represents the ITSF. Other
          values represent the DTSF instance numbers.";
        reference
          "IEEE Std 802.1ASed 14.23.15";
      }
      leaf seconds {
        type uint48;
```

```
        default "0";
        description
          "The integer portion of the timestamp in units of seconds.";
      }
      leaf fractional-nanoseconds {
        type uint48;
        default "0";
        description
          "The fractional portion of the timestamp in units of 2^-16 ns.";
      }
    }
  }

  grouping fttm-tsf-algo-name-group {
    description
      "The fttmTsfAlgoName object is a vector of strings, where each vector
      member provides the name of the algorithm used by each active TSF
      instance number (the ITSF and the active DTSFs).";
    reference
      "IEEE Std 802.1ASed 14.23.21";
    list fttm-tsf-algo-name-list {
      key "tsf-instance-number";
      description
        "The fttmTsfAlgoName object is a vector of strings, where each vector
        member provides the name of the algorithm used by each active TSF
        instance number (the ITSF and the active DTSFs)";
      reference
        "IEEE Std 802.1ASed 14.23.21";
      leaf tsf-instance-number {
        type uint8;
        description
          "The TSF instance number for which the algorithm is to be queried.";
        reference
          "IEEE Std 802.1ASed 20.3.2.2";
      }
      leaf fttm-tsf-algo-name {
        type string;
        config false;
        description
          "Each fttmTsfAlgoName[x] member, where x is a value from 0 to
          fttmNumActiveDtsfs (see 14.23.13) and represents the TSF instance
          number, is a 128 octet that contains a string with the name of
          the algorithm for the TSF instance number.";
        reference
          "IEEE Std 802.1ASed 14.23.21";
      }
    }
  }

  grouping fttm-tsf-sel-time-index-group {
    description
      "The fttmTsfSelTimeIndex object gives the input time index that is
         selected by each TSF instance.";
    reference
      "IEEE Std 802.1ASed 14.23.22";
    list fttm-tsf-sel-time-index-list {
      key "tsf-instance-number";
      description
        "The fttmTsfSelTimeIndex object gives the input time index that is
        selected by each TSF instance.";
      reference
        "IEEE Std 802.1ASed 14.23.22";
      leaf tsf-instance-number {
        type uint8;
        description
          "The TSF instance number for which the selected time-index is
          to be queried.";
        reference
          "IEEE Std 802.1ASed 14.23.8";
      }
      leaf fttm-tsf-sel-time-index {
        type uint16;
        default "511";
        config false;
        description
          "Gives the input time index that is selected to be the output of
          the TSF instance.";
        reference
          "IEEE Std 802.1ASed 14.23.22";
      }
    }
```

```
  }

  augment "/ptp-tt:ptp/ptp-tt:common-services" {
    description
      "Augment IEEE Std 1588 commonServices with fault-tolerant-timing-module service.";
    container fault-tolerant-timing-module-service {
      if-feature "fttm";
      description
        "The Fault-Tolerant Timing Module Service structure contains the
         fttmSystemList, which is a list of instances of the Fault-Tolerant
         Timing Module Service.";
      reference
        "14.23 of IEEE Std 802.1AS";
      list fttm-system {
        key "fttm-system-index";
        description
          "Indexed list of FTTM systems in the FTTM Service";
        leaf fttm-system-index {
          type uint8;
          description
            "Index for the FTTM system.";
        }
        container fttm-system-ds {
          description
            "The fttmSystemDS describes the attributes of the
             respective instance of the Fault-Tolerant Timing Module Service.";
          reference
            "14.23 of IEEE Std 802.1AS";
          uses fault-tolerant-timing-module-group;
        }
        container fttm-system-description-ds {
          description
            "The fttmSystemDescriptionDS contains descriptive information for
             the respective instance of the Fault-Tolerant Timing Module Service.";
          reference
            "14.24 Fault-Tolerant Timing Module System Description Parameter
             Data Set (fttmSystemDescriptionDS) of IEEE Std 802.1ASed";
          leaf user-description {
            type string {
              length "0..128";
            }
            description
              "Configuration description of the Fault-Tolerant Timing Module system.";
            reference
              "14.24.3 of IEEE Std 802.1ASed";
          }
        }
      }
    }
  }
}
```

Insert the following new Clause 20:

20. Fault-tolerant timing with time integrity

20.1 General

It is important for some time-sensitive applications (e.g., aerospace networks, per IEEE P802.1DP) to consider fault tolerance, including availability and integrity of the synchronizing function, to enable reliable and trustworthy system behavior. Features of gPTP that can be used to support fault-tolerant time synchronization include its provisions for multiple time domains, multiple GMs, multiple time distribution paths, and multiple PTP Instances per port in Bridges and end stations, and the external port configuration mode (10.3.1.3) that allows static time distribution paths to be established. The use of these features must be carefully considered by a system designer to ensure that the application's requirements for assured systems are met. For example, an aerospace network is typically expected to tolerate one or more arbitrary faults in Bridges, end stations, links, and GMs to maintain availability and integrity of clock synchronization.

To achieve fault-tolerant timing with time integrity, this standard defines a Fault-Tolerant Timing Module (FTTM) for use with the gPTP features listed above. The concepts used by the FTTM are described in 20.2. Details on the FTTM and its default operations are given in 20.3. General information about fault-tolerant timing with time integrity can be found in Annexes J, L, and M. Genral information on using the FTTM can be found in Annexes K and N.

20.2 Fault-tolerant time synchronization concepts

20.2.1 Availability of time

The continuous availability of time is enhanced by redundancy. For gPTP, this redundancy can be implemented by using multiple time domains, multiple time distribution paths, and multiple PTP Instances in Bridges and end stations.

20.2.2 Trust and integrity of time

For this standard, a trusted time is one that passes a specified criterion that identifies it as being within a safe bound of a non-faulty time and is, thus, safe to use. This establishment of trust gives integrity to the time.

For gPTP, trust, and hence integrity, can be established through the comparison of the times coming from independent time sources and the observation that they match within the specified criterion. See L.5 for an example of such a criterion.

20.2.3 Time agreement generation and preservation

Time agreement generation and preservation is the process by which multiple time source nodes (GMs) come to an agreement on the time and maintain that agreement in the presence of both faults and oscillator drift. This process preserves both the collective accuracy and relative precision of the set of GMs.

Time agreement generation and preservation should be done in a manner that is resilient to faults, including Byzantine faults. See [B1], [B2], [B3], [B4], and [B5].

20.2.4 Time distribution

Time distribution is the process of distributing the time established by time agreement generation from time source nodes (GMs) to time receivers (PTP instances) in Bridges and end stations. Time distribution is performed using gPTP, per the models described in Clause 7 and the mechanisms specified in clauses 8 to 16 of this standard.

20.2.5 Dependent gPTP times

Dependent gPTP times share one or more common time-influencing components. This could be a common GM, continuously synchronized GMs, GMs that share a common (continuously connected) ClockSource, or a common PTP Relay Instance.

Because dependent gPTP times share one or more common influencers, they do not, on their own, enable end-to-end integrity checking of the time synchronization function. However, they can be used to improve the availability of a given time source and can provide partial integrity checks. For example, an application that receives timing from a single GM through more than one redundant synchronization trees has increased availability of that GM's time and can check the integrity of the synchronization trees by comparing the time received from them. However, because the time originates from a single GM, the integrity of that GM's time cannot be confirmed and, thus, end-to-end integrity of the time synchronization function is not achieved.

When a set of dependent gPTP times is used in combination with other gPTP times, that are independent (see 20.2.6), the set of dependent gPTP times can be reduced to a single independent gPTP time and used to enhance the ability to achieve end-to-end integrity of the time synchronization function. This operation is performed by the Fault-Tolerant Timing Module (see 20.3).

Dependent gPTP times can be identified by one of the following methods:

— They have the same gPTP domainNumber. This indicates that gPTP messages from the same PTP GM are received by two PTP End Instances, on two distinct physical ports, that are serviced by the FTTM.
— They have different gPTP domainNumbers but the same gmtimeBaseIndicator. This indicates that the gPTP messages come from different PTP GMs that share the same ClockSource.
— They have gPTP domainNumbers that are defined by a management entity, which is out of scope of this standard, to be dependent.

NOTE—Faults that cause the masquerading of any of the above gPTP fields can be mitigated by the Fault-Tolerant Timing Module (see 20.3).

In a network that supports fault-tolerant timing with time integrity, the gmTimeBaseIndicator shall be made unique across all ClockSources in the network.

NOTE—A network management entity (outside the scope of this standard) could be used to ensure that gmtimeBaseIndicator is unique across all clock sources present.

20.2.6 Independent gPTP times

Independent gPTP times do not share any common time-influencing components with each other and, therefore, deliver independent time values. Independent gPTP times are, by definition, from different domains.

Because independent gPTP times do not share any common influencers, they can enable end-to-end integrity checking of the time synchronization function, provided their times track sufficiently closely. Independent domains need to be aligned to each other in a manner that is resilient to faults (i.e., achieve time agreement and preservation, see 20.2.3) For example, time agreement mechanisms can be used to align the clocks of two independent GMs.

Because the independent domains are synchronized to each other, they provide a redundant source of time to the end application and, thus, also improve the availability of the time synchronization function to the end application.

20.3 Fault-Tolerant Timing Module

To enable fault-tolerant timing with time integrity, a Fault-Tolerant Timing Module (FTTM) operating at the application layer, per Clause 9, may be implemented in a time-aware system that supports multiple timing domains. The FTTM manages the selection of a time source from amongst two or more gPTP times (and PTP Instances) to support increased availability (see 20.2.1) and integrity (see 20.2.2). The FTTM also supports single domain solutions but, in this scenario, it does not provide any enhancements for increased availability or integrity.

The availability and integrity scenarios supported by the FTTM are listed below.

- Enhanced availability and integrity scenario:
  This scenario operates with multiple times and multiple domains.
  In this mode, the FTTM supports increased availability and integrity.
- Enhanced availability and limited integrity scenario:
  This scenario operates with multiple times and a single domain.
  In this mode, the FTTM supports increased availability and potential time distribution integrity, but not GM integrity.
- Regular availability and no integrity scenario:
  This scenario operates with a single time and, hence, a single domain.
  In this mode, the FTTM does not support increased availability or integrity.

The application is expected to be aware of the availability and integrity scenario in which the FTTM is operating under and its effects on the integrity of the FTTM's output.

Figure 20-1 illustrates the FTTM operating with three PTP Instances. The FTTM can also use the local oscillator's clock (OSC_CLK) as an input to its selection algorithm.

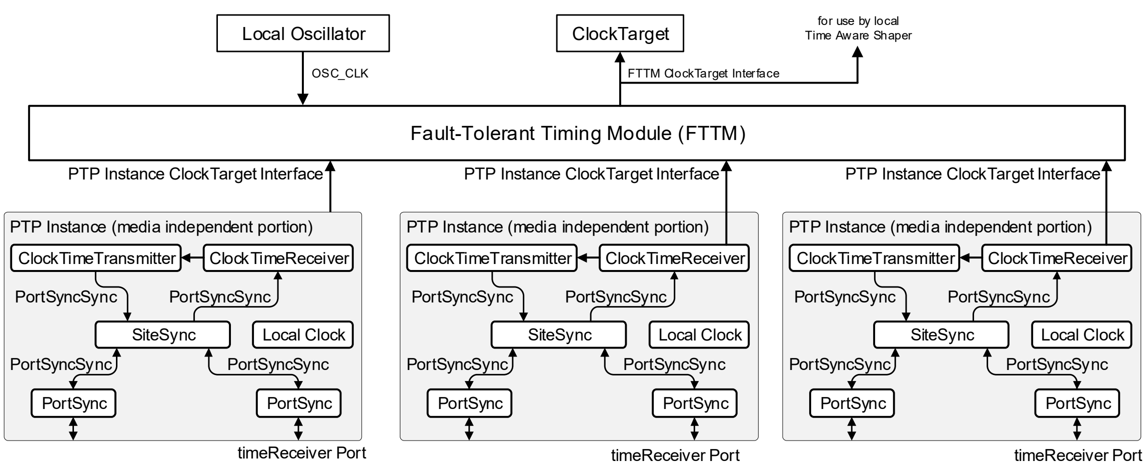


Figure 20-1 — Fault-Tolerant Timing Module in operation

Because the FTTM resides between PTP Instances and a ClockTarget, its effect is localized. This allows custom algorithms that can better service specific timing characteristics of the network to be used by the

FTTM. A default algorithm that can be used by the Time Selection Function (TSF) instances (see 20.3.2) in the FTTM is defined in 20.3.5.

20.3.1 Scope and assumptions

The following list provides the detailed assumptions and goals for the FTTM:

— A fault-tolerant network (with time integrity) and its configuration are static during normal operation.
— All gPTP ports are configured using the external port configuration provision (i.e., the BTCA is not used).
— There is no administrative reconfiguration during run-time in the event of faults.
— While operation with one gPTP time received over one domain is supported by the FTTM, operation with more than one gPTP time received over more than one domain is required to enable fault-tolerant timing with time integrity. To support interoperability, a minimum number of times and domains needs to be specified for an application.
— gPTP times are recognized as being dependent or independent as defined in clause 20.2.5 and 20.2.6, respectively.

20.3.2 Functional description

The FTTM shall consist of the following functions.

— A local oscillator clock (OSC_CLK). If fttmUseOscClk is TRUE, OSC_CLK shall be a free-running clock that is independent of the times being received by the PTP Instances that are connected to the FTTM. The health and trust of OSC_CLK is outside the scope of this standard.
— Input ClockTargetEventCapture application interfaces (see Clause 9.3) providing time information to the FTTM, where PTP End Instances serve as the ClockTimeReceiver entities and the Time Selection Functions (TSFs) within the FTTM serves as the ClockTarget entity. Time is passed to the TSFs via each ClockTarget application interface's timeReceiverTimeCallback parameter. The instanceIndex number associated with each gPTP End Instance is also passed to the TSFs.
— Zero or more instances of TSFs for servicing dependent times (see 20.2.5), each of which is called a Dependent Time Selection Functions, (DTSF).
— One instance of a TSF for servicing independent times (see 20.2.6), called the Independent Time Selection Function (ITSF).
— Zero or more instances of ClockTargetEventCapture application interface(s) (see clause 9.3) providing time information from DTSF(s) to the ITSF, where each DTSF serves as a ClockTimeReceiver entity and the ITSF serves as a ClockTarget entity. Time is passed to the ITSF via each ClockTargetEventCapture application interface's timeReceiverTimeCallback parameter. The instanceIndex number of the PTP End Instance associated with the DTSF output ClockTargetEventCapture interface is also passed to the ITSF.
— Output ClockTargetEventCapture application interface (see clause 9.3) providing time information from the FTTM, where the FTTM's output (FTTM_OUTPUT) serves as the ClockTimeReceiver entity to the application's ClockTarget entity. Time is passed to the application's ClockTarget entity via the ClockTarget application interface's timeReceiverTimeCallback parameter. The instanceIndex number of the PTP End Instance associated with the Output ClockTargetEventCapture interface is also provided by the FTTM

A functional block diagram of the FTTM is shown in Figure 20-2.

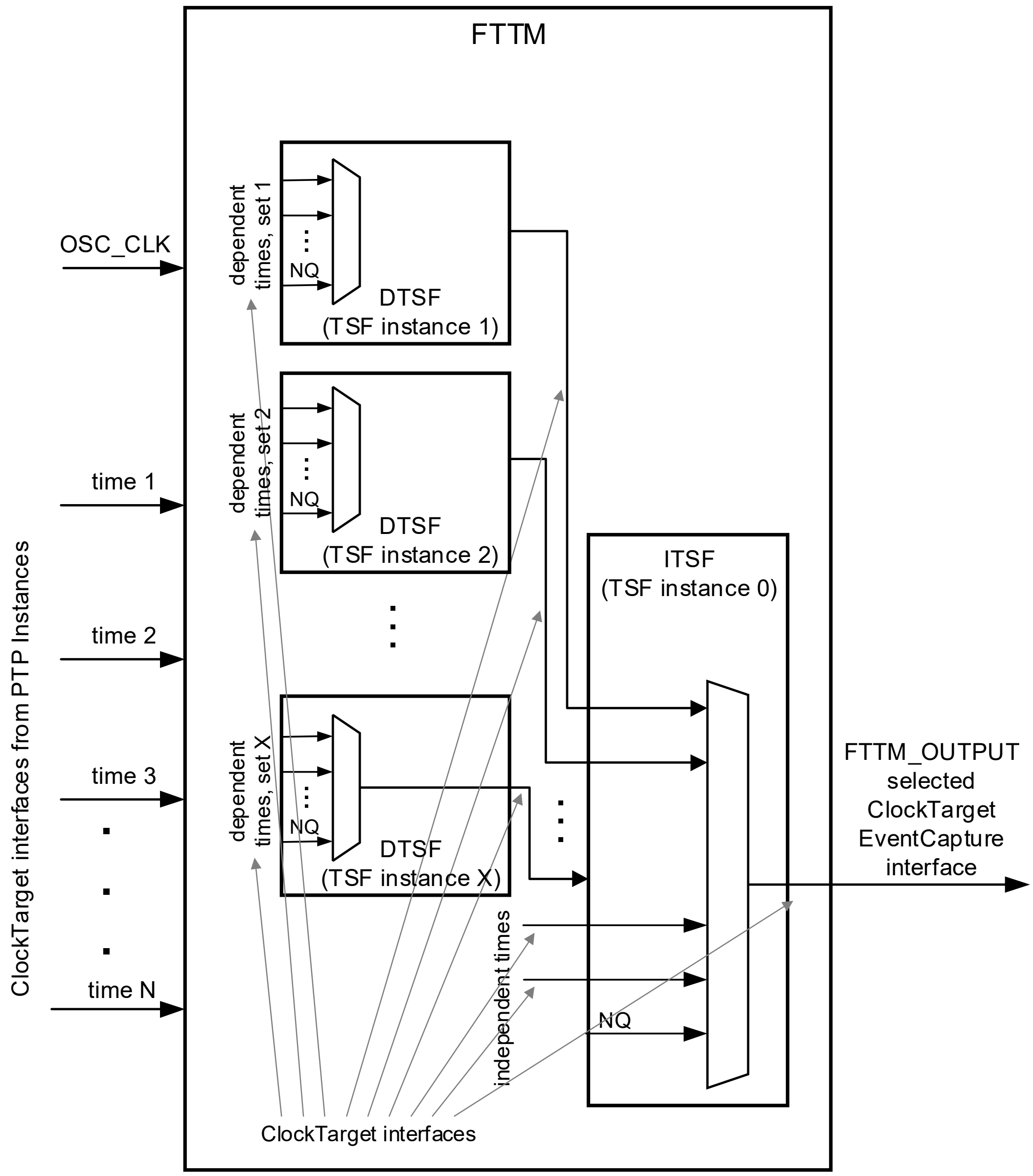


Figure 20-2 — FTTM functional block diagram

When operating in the enhanced availability and integrity scenario (see 20.3):

— Each set of input times to the FTTM that share a common dependency shall be processed as a group by one DTSF instance. This grouping allows each DTSF instance to produce an output that is independent from the outputs of the other DTSFs and from the other input times to the FTTM.
— All the input times to the FTTM that share no dependency with any other time shall be processed by the ITSF.
— The output times of every DTSF instance shall be processed by the ITSF.
— The ITSF shall select one of its input times, or the NQ time if none of its input times can be determined to be trusted, as its result.

When operating in the enhanced availability and limited integrity scenario (see 20.3), it is expected that all the input times to the FTTM are connected to the ITSF and treated as if they are independent from each other (i.e., DTSFs are not used in this scenario). The ITSF shall select one of its input times, or the NQ time if none of its input times can be determined to be trusted, as its result.

When operating in the regular availability and no integrity scenario, (see 20.3), the FTTM shall always select its sole input time as its result.

The FTTM's local oscillator clock, OSC_CLK, may be used by the FTTM as a frequency reference to infer additional information about the qualities of the input times.

20.3.2.1 Input ClockTarget interfaces

The input ClockTarget interfaces to the FTTM are the output ClockTarget interface of the PTP Instances connected to the FTTM. They pass time information from the PTP Instances to the TSFs of the FTTM. The FTTM accompanies each input ClockTarget interface with the instanceIndex number of the corresponding PTP Instance.

The FTTM uses the ClockTargetEventCapture interface type (see clause 9.3). The use of other ClockTarget interface types (e.g., see clauses 9.4 and 9.5) with the FTTM is discussed in Annex M.

The FTTM's input ClockTarget interfaces can be for times that have a dependency with one or more times of other input ClockTarget interfaces or can be for times that have no dependency with the times of other input ClockTarget interfaces.

20.3.2.2 Time Selection Function (TSF)

Each Time Selection Function (TSF) instance has an embedded time-index selection algorithm that analyzes the TSF's set of input ClockTarget interfaces, determines which corresponding input time(s) can be trusted (see 20.2.2), and selects a trusted time and its corresponding PTP Instance, if found, to be its output. If no trusted time is found, the algorithm selects the Not Qualified (NQ) time (see 20.3.2.3) for the TSF's output.

The state machine for the TSF is defined in 20.3.4.

One TSF instance in the FTTM is used for selecting a time from a set of independent times (see 20.2.6). This TSF is called the ITSF and is assigned the TSF instance number of zero.

If any of the input times to the FTTM are dependent (see 20.2.5), then one TSF instance is used to select a time for each set of dependent times. These TSFs are called DTSFs and are assigned TSF instance numbers from 1 to fttmNumActiveDtsfs (see 14.23.13).

The default time-index selection algorithm for the TSF is the mid-value time-index selection algorithm (MVTISA, see 20.3.5). Other algorithms, which are out of scope of this standard, can be used by any TSF instance. These other algorithms are not required to produce the same result as the default algorithm, MVTISA, or as each other to achieve time convergence.

Being trusted does not mean the time is non-faulty. However, as long as a time remains trusted (i.e., as long as its time continues to pass the specified criteria), it can be safely used by a DTSF or by the ITSF.

Any time that has an isSynced status (see 18.4.1.1) equal to FALSE or a gmPresent status (see 10.2.4.13) equal to FALSE is, without further consideration, declared to be untrusted.

20.3.2.3 Not Qualified (NQ) time

The Not Qualified (NQ) time is used to represent the condition where none of the input times to a TSF instance can be determined to be trusted. The NQ time contains the ClockTarget interface from any one of the input times (arbitrarily selected or implementation specific) being processed by the time-index selection algorithm but shall have the isSynced status (see 18.4.1.1) and the gmPresent status forced to FALSE, to indicate the untrusted condition.

NOTE—Because the NQ time is, by definition, not trusted, the values of its other parameters (aside from isSynced and gmPresent) have no functional impact.

All the possible parameters in the NQ time are listed below.

— domainNumber = the domainNumber from the arbitrarily selected time from the set of input times being processed by the algorithm
— timeReceiverTimeCallback = the timeReceiverTimeCallback value from the arbitrarily selected time
— isSynced = FALSE
— gmPresent = FALSE
— errorCondition = the errorCondition value from the arbitrarily selected time
— clockPeriod = the clockPeriod value from the arbitrarily selected time
— timeReceiverCallbackPhase = the timeReceiverCallbackPhase value invoked by the ClockTarget entity of the ClockTarget interface
— grandmasterIdentity = the grandmasterIdentity value from the arbitrarily selected time
— gmTimeBaseIndicator = the gmTimeBaseIndicator value from the arbitrarily selected time
— lastGmPhaseChange = the lastGmPhaseChange value from the arbitrarily selected time
— lastGmFreqChange = the lastGmFreqChange value from the arbitrarily selected time

20.3.2.4 Output ClockTarget Interfaces

The output ClockTarget interface from the FTTM is a reference to the ClockTarget interface of the PTP Instance selected by the TSFs in the FTTM.

The FTTM accompanies the output ClockTarget interface with the instanceIndex number of the selected PTP Instance.

The FTTM uses the ClockTargetEventCapture interface type (see 9.3). The use of other ClockTarget interface types (e.g., see 9.4 and 9.5) with the FTTM is discussed in Annex M.

20.3.3 FTTM state machine

20.3.3.1 General

The FTTM state machine described in this subclause interacts with all the TSF state machines (instantiated in the FTTM's DTSF instances and in the FTTM's ITSF) to find and select a trusted input ClockTarget interface, if one exists, to present as the FTTM's output.

The TSF state machine is defined in 20.3.4.

The default time-index selection algorithm used in the TSF state machine is the MVTISA, which is defined in 20.3.5.

The FTTM uses the ClockTargetEventCapture interface type (see clause 9.3). The use of other ClockTarget interface types with the FTTM is discussed in Annex M.

20.3.3.2 FTTM State machine variables

20.3.3.2.1 fttmTimeInterfaceRateRatioOff

The variable fttmTimeInterfaceRateRatioOff[x] is equal to the configured magnitude of the offset ratio between the rate of the time arriving on input ClockTarget interface x to the rate of the FTTM's local clock, OSC_CLK that is allowed to remain in the FREQ_TRUSTED state.

The value is expressed as the fractional frequency offset multiplied by 2^{41}, i.e.,

$$fttmTimeInterfaceRateRatioOff[x] = (|\text{rate of incoming clock} / \text{rate of OSC\_CLK} - 1.0|) \times 2^{41}$$

The variable fttmTimeInterfaceRateRatioOff is a vector of size numTimeIndexes of Uint32 values and the data type is as follows:

typedef Uint32 fttmTimeInterfaceRateRatioOff [numTimeIndexes];

The value for each fttmTimeInterfaceRateRatioOff vector member is selected from the fttmTimeInterfaceRatioRatioOff management object vector (see 14.23.18) based on the mapping of FTTM input ClockTarget interfaces to the default ITSF, as determined by the fttmMapIndexToTsf management object (see 14.23.5).

20.3.3.2.2 itsfInstanceIndex

The itsfInstanceIndex variable contains the UInteger32 instanceIndex value of the PTP Instance that originally generated the ClockTargetEventCapture.result for the selected input Clock Target of the ITSF.

20.3.3.2.3 itsfSelTimeIndex

The itsfSelTimeIndex variable is the index number of the selected input ClockTarget interface of the ITSF. This ITSF's selected input ClockTarget interface is presented as the output ClockTarget interface of the FTTM.

The range of values ranges from 1 to numTimeIndexes and the value of 511 represents the NQ time (see 20.3.2.3).

20.3.3.2.4 itsfTrustState (fttmOutputTrustState)

The variable itsfTrustState gives the determined trust state from the ITSF state machine (see 20.3.4). Valid values are NOT_TRUSTED and TIME_TRUSTED.

20.3.3.2.5 fttmTrustState (fttmOutputTrustState)

The variable fttmTrustState gives the determined trust state of the FTTM. Valid values are NOT_TRUSTED, TIME_TRUSTED, and FREQ_TRUSTED.

20.3.3.2.6 prevFttmTrustState (fttmOutputTrustState)

The variable prevFttmTrustState gives the previously determined trust state of the FTTM. Valid values are NOT_TRUSTED, TIME_TRUSTED, and FREQ_TRUSTED.

20.3.3.2.7 rRatioOff (Uinteger32)

The variable rRatioOff is the offset of the ratio between the rate of the time provided on the output of the ITSF to the rate of the FTTM's local clock, OSC_CLK. The value is expressed as a fractional frequency offset multiplied by 2^{41}, i.e.,

$$rRatioOff = (|\text{rate of incoming clock} / \text{rate of OSC\_CLK} - 1.0|) \times 2^{41}$$

The magnitude of this offset is one of the considerations used by FTTM state machine to determine whether it enters the FREQ_TRUSTED state or the NOT_TRUSTED state. See 20.3.3.3.

20.3.3.2.8 rRatioSdOff (Uinteger32)

The variable rRatioSdOff is the standard deviation of the offset of the ratio between the rate of the time provided on the output of the ITSF to the rate of the FTTM's local clock, OSC_CLK. It is one of the considerations used by the FTTM state machine to determine whether it enters the FREQ_TRUSTED state or the NOT_TRUSTED state. See 20.3.3.3.

20.3.3.3 FTTM State diagram

The FTTM state machine is shown in Figure 20-3.

The INITIALIZE state of the FTTM state machine resets the dtsfEval and itsfEval variables to FALSE. This prepares them for future initiation of a default DTSF and default ITSF selection evaluation, respecdtively.

The WAIT_INVOKE state of the FTTM state machine waits for a ClockTargetEventCapture.invoke event from the ClockTarget. When the event is received, the state machine transfers to the SEND_INVOKE state.

The SEND_INVOKE state of the FTTM state machine simultaneously sends a ClockTargetEventCapture.invoke event to all the FTTM's input ClockTarget interfaces to request new timing information from the PTP Instances that are connected to the FTTM. Once all the input ClockTarget interfaces respond to the invoke event with ClockTargetEventCapture.result, this state transitions to one of the ONE_INDEX, DTSF_SELECT_TIME, or ITSF_SELECT_TIME states depending on the value of fttmNumActiveTimeIndexes and fttmNumActiveDTSFs.

If fttmNumActiveTimeIndexes is equal to 1, the state machine moves to the ONE_INDEX state. This state simply transfers the ClockTargetEventCapture.result from the input port of the FTTM to the output port of the FTTM. This state then transitions unconditionally back to the WAIT_INVOKE state.

If fttmNumActiveTimeIndexes is greater than 1 and fttmNumActiveDTSFs is not equal to 0, the state machine transitions to the DTSF_SELECT_TIME state of the FTTM state machine. This state sends a ClockTargetEventCapture.invoke to all the active DTSF instances in the FTTM. This causes the TSF state machine (see 20.3.4) in each of the active DTSF instances to perform another round of searching for and selecting, if available, a valid dependent time. Once every active DTSF instance responds to the invoke event with ClockTargetEventCapture.result and outputInstanceIndex, this state transitions to the ITSF_SELECT_TIME state.

If fttmNumActiveTimeIndexes is greater than 1 and fttmNumActiveDTSFs is equal to 0, the state machine transitions to the ITSF_SELECT_TIME state of the FTTM state machine. This state sends a ClockTargetEventCapture.invoke to the ITSF. This causes the TSF state machine (see 20.3.4) in the ITSF to perform another round of searching for and selecting, if available, a valid independent time. Once the ITSF responds to the invoke event with ClockTargetEventCapture.result, outputInstanceIndex, outputSelTimeIndex, outputTrustState, and outputTimeIndexChangeCnt, this state transitions to the UPDATE_STATE state.

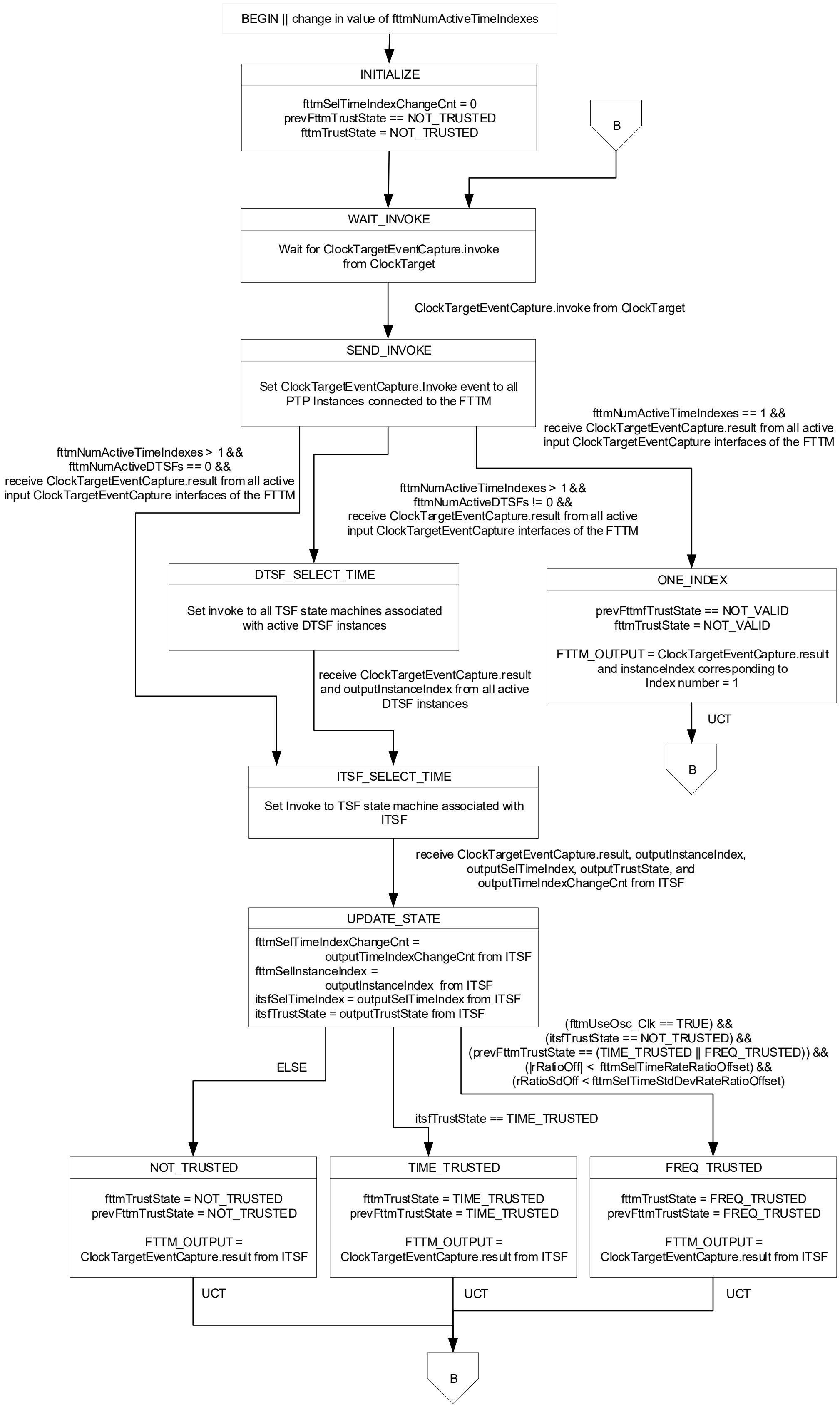


Figure 20-3 — FTTM state machine

The UPDATE_STATE state of the FTTM state machine updates is local variables with the outputs from the ITSF. Based on the outputTrustState from the ITSF, this state will transition to either the NOT_TRUSTED, the TIME_TRUSTED, or the FREQ_TRUSTED states.

If the ITSF produced a trust state of TIME_TRUSTED, the state machine transitions to the TIME_TRUSTED state. In this state, the FTTM state machine sets its trust state variables and management objects to TIME_TRUSTED and passes the ITSF's ClockTargetEventCapture.result to its output. The state machine then transitions back to the WAIT_INVOKE state.

If the ITSF produced a trust state of NOT_TRUSTED and the FTTM's previous trust state was either TIME_TRUSTED or FREQ_TRUSTED and the currently selected time has a rateRatio (relative to the FTTM's OSC_CLK) and a rateRatio standard deviation that is within configured thresholds and the fttmUseOscClk managed object is TRUE, then the state machine transitions to the FREQ_TRUSTED state. In this state, the FTTM state machine sets its trust state variables and management objects to FREQ_TRUSTED and passes the ITSF's ClockTargetEventCapture.result to its output. The state machine then transitions back to the WAIT_INVOKE state.

If neither of the conditions for the FTTM state machine to go to the TIME_TRUSTED or FREQ_TRUSTED states were met, then the state machine transitions to the NOT_TRUSTED state. In this state, the FTTM state machine sets its trust state variables and management objects to NOT_TRUSTED and passes the ITSF's ClockTargetEventCapture.result to its output. In this state, the ClockTargetEventCapture.result will have its isSynced and gmPresent parameters set to FALSE. The state machine then transitions back to the WAIT_INVOKE state.

20.3.4 Time Selection Function (TSF) state machine

20.3.4.1 General

The TSF state machine is individually instantiated in each DTSF instance and in the ITSF to perform the time index selection for the corresponding function.

The TSF state machine calls a time-index selection algorithm (see the SELECT_TIME_INDEX state in Figure 20-4). The default time-index selection algorithm for TSFs is the mid-value time-index selection algorithm (MVTISA), which is described in 20.3.5.

20.3.4.2 TSF State machine variables

The TSF variables described in this subclause are unique to each TSF instance in the FTTM and its corresponding time-index selection algorithm (e.g., MVTISA in 20.3.5). Some are local to the TSF instance and some, as indicated in the variables' descriptions, are passed down to the TSF from the FTTM state machine (see 20.3.3), derived from management objects (see 14.23), derived from ClockTargetEventCapture.result, and/or derived from ClockTargetEventCapture.result parameters (see 9.3.3) associated with the input ClockTargetEventCapture interfaces connected to the TSF.

20.3.4.2.1 gmIdentityStatus

The variable gmIdentityStatus is a vector of ClockIdentity. The vector member gmIdentityStatus[x] holds the grandmasterIdentity status from the ClockTargetEventCapture.result for Clock Target interface x of the TSF instance, where x ranges from 1 to numTimeIndexes.

For an instance of the TSF, the vector is of size numTimeIndexes and the data type is as follows:

typedef ClockIdentity gmIdentityStatus [numTimeIndexes];

20.3.4.2.2 hystVal

The variable hystVal[x][y] holds the hysteresis added to maxAsVal[x][y] (see 20.3.4.2.3) for the times of the two input ClockTargetEventCapture interfaces, with index numbers x and y, on the TSF instance. The hysteresis enables the use of one time skew level to set the trust status and another time skew level to clear the trust status.

The variable hystVal[x][y] is a two-dimensional array of UInteger32 values, each in units of 2^{-16} nanoseconds with a size of numTimeIndexes in each dimension. The range of x and of y is from 1 to numTimeIndexes. The data type of hystVal is as follows:

typedef UInteger32 hystVal [numTimeIndexes][numTimeIndexes];

The value for each hystVal array member is selected from an array member from the fttmHyst management object array (see 14.23.7) based on the mapping of FTTM input Clock Target interfaces to the instance of the TSF, as determined by the fttmMapIndexToTsf management object (see 14.23.5) and (for the ITSF only) the fttmMapDtsfToItsf management object (see 14.23.4).

20.3.4.2.3 maxAsVal

The value maxAsVal[x][y] is used as the criteria to determine the trustworthiness of the times from ClockTarget interfaces x and y when they are compared to each other. See $maxAs_{xy}$ in L.5.

The variable maxAsVal[x][y] is a two-dimensional array of UInteger32 values, each in units of 2^{-16} nanoseconds with a size of numTimeIndexes in each dimension. The data type is as follows:

typedef UInteger32 maxAsVal [numTimeIndexes][numTimeIndexes];

The value for each maxAsVal array member is selected from an array member from the fttmMaxAs management object array (see 14.23.10) based on the mapping of FTTM input Clock Target interfaces to the TSF instance, as determined by the fttmMapIndexToTsf management object (see 14.23.5) and (for the ITSF only) the fttmMapDtsfToItsf management object (see 14.23.4).

20.3.4.2.4 numTimeIndexes (UInteger8)

The variable numTimeIndexes contains the number of input ClockTarget interfaces assigned to the TSF instance, where the assignment is done via the fttmMapIndexToTsf management object (see 14.23.5) and (for the ITSF only) the fttmMapDtsfToItsf management object (see 14.23.4).

20.3.4.2.5 outputInstanceIndex (Uinteger32)

The variable outputInstanceIndex gives the instanceIndex number of the PTP Instance of the ClockTarget interface selected by the TSF.

20.3.4.2.6 outputSelTimeIndex (fttmSelectedIndex)

The outputSelTimeIndex variable is the index number of the selected input ClockTarget interface of the TSF instance. This selected input ClockTarget interface is presented as the output ClockTarget interface of the instance of the TSF.

For any instance of the TSF, the range of values for the input ClockTarget interfaces ranges from 1 to numTimeIndexes and the value of 511 represents the NQ time (see 20.3.2.3).

20.3.4.2.7 outputTimeIndexChangeCnt (UInteger16)

The variable outputTimeIndexChangeCnt gives the number of times the TSF has changed its time index selection. The value increments every time the TSF changes its time index selection and rolls over to 0 if the value is incremented when its current value is 65535.

20.3.4.2.8 outputTrustState (fttmOutputTrustState)

The variable outputTrustState gives the determined trust state of the TSF instance's state machine. Valid values are NOT_TRUSTED and TIME_TRUSTED.

20.3.4.2.9 prevSelTimeIndex (fttmSelectedIndex)

This prevSelTimeIndex variable is the index number of the previously selected input ClockTarget interface of the TSF instance. This previously selected input ClockTarget interface can be used as a condition in the time-index selection algorithm.

For any instance of the TSF, the range of values for the selected input ClockTarget interfaces ranges from 1 to numTimeIndexes and the value of 511 represents the NQ time (see 20.3.2.3).

20.3.4.2.10 prevTimeIndexStatus

The variable prevTimeIndexStatus is a vector that holds the previous trust status of a ClockTarget interface of the TSF instance. The variable prevTimeIndexStatus[x] holds the previous trust status of ClockTarget interface x. Valid values are NOT_TRUSTED and TRUSTED. The data type is as follows:

typedef fttmInputTrustStatus prevTimeIndexStatus [numTimeIndexes];

20.3.4.2.11 prevTimeIndexPairStatus

The variable prevTimeIndexPairStatus is a 2-dimensional array that holds the previous trust status between two ClockTarget interfaces of the instance of the TSF. The variable prevTimeIndexPairStatus[x][y] holds the previous trust status between ClockTarget interface x and ClockTarget interface y. Valid values are NOT_TRUSTED and TRUSTED. The data type is as follows:

typedef fttmInputTrustStatus prevTimeIndexPairStatus [numTimeIndexes][numTimeIndexes];

20.3.4.2.12 prevTrustState (fttmOutputTrustState)

The variable prevTrustState gives the previously determined trust state of the TSF instance's state machine. Valid values are NOT_TRUSTED and TIME_TRUSTED for the default time-index selection algorithm, MVTISA (see 20.3.5).

20.3.4.2.13 savedDomainNumber

The variable savedDomainNumber is a vector of UInteger8. The vector member savedDomainNumber[x] holds the domainNumber status from the ClockTargetEventCapture.result for Clock Target interface x of the TSF instance, where x ranges from 1 to numTimeIndexes.

For an instance of the TSF, the vector is of size numTimeIndexes and the data type is as follows:

typedef UInteger8 savedDomainNumber [numTimeIndexes];

20.3.4.2.14 savedGmIdentity

The variable savedGmIdentity is a vector of ClockIdentity. The vector member savedGmIdentity[x] holds the clockIdentity of the Grandmaster PTP Instance from the ClockTargetEventCapture.result for Clock Target interface x of the TSF instance, where x ranges from 1 to numTimeIndexes.

For an instance of the TSF, the vector is of size numTimeIndexes and the data type is as follows:

typedef ClockIdentity savedGmIdentity [numTimeIndexes];

20.3.4.2.15 savedGmPresent

The variable savedGmPresent is a vector of Boolean. The vector member savedGmPresent[x] holds the gmPresent status from the ClockTargetEventCapture.result for Clock Target interface x of the TSF instance, where x ranges from 1 to numTimeIndexes.

For an instance of the TSF, the vector is of size numTimeIndexes and the data type is as follows:

typedef Boolean savedGmPresent [numTimeIndexes];

20.3.4.2.16 savedInstanceIndex

The variable savedInstanceIndex is a vector of UInteger32. The vector member savedInstanceIndex[x] holds the instanceIndex value of the PTP Instance that originally generated the ClockTargetEventCapture.result for Clock Target interface x of the TSF instance, where x ranges from 1 to numTimeIndexes.

For an instance of the TSF, the vector is of size numTimeIndexes and the data type is as follows:

typedef UInteger32 savedInstanceIndex [numTimeIndexes];

20.3.4.2.17 savedIsSynced

The variable savedIsSynced is a vector of Boolean. The vector member savedIsSynced[x] holds the isSynced status from the ClockTargetEventCapture.result for Clock Target interface x of the TSF instance. The range of x is from 1 to numTimeIndexes.

For each instance of the TSF, the vector is of size numTimeIndexes and the data type is as follows:

typedef Boolean savedIsSynced [numTimeIndexes];

20.3.4.2.18 savedTimeCallback

The variable savedTimeCallback is a vector of extended timestamps. The vector member savedTimeCallback[x] holds the latest timeReceiverTimeCallback result for input ClockTarget interface x.

The vector is of size numTimeIndexes and the data type is as follows:

typedef ExtendedTimestamp savedTimeCallback [numTimeIndexes];

20.3.4.2.19 selTimeIndex (fttmSelectedIndex)

The selTimeIndex variable is the index number of the selected input ClockTarget interface of the TSF instance. This selected input ClockTarget interface is presented as the output ClockTarget interface of the instance of the TSF.

For any instance of the TSF, the range of values for the input ClockTarget interfaces ranges from 1 to numTimeIndexes and the value of 511 represents the NQ time (see 20.3.2.3).

20.3.4.2.20 selTimeIndexChangeCnt (UInteger16)

The variable selTimeIndexChangeCnt gives the number of times the TSF has changed its time index selection. The value increments every time the TSF changes its time index selection and rolls over to 0 if the value is incremented when its current value is 65535.

20.3.4.2.21 TimeIndexStatus

The variable TimeIndexStatus is a vector that holds the trust status of a ClockTarget interface of the TSF instance. The variable TimeIndexStatus[x] holds the trust status of ClockTarget interface x. Valid values are NOT_TRUSTED and TRUSTED. The data type is as follows:

typedef fttmInputTrustStatus TimeIndexStatus [numTimeIndexes];

20.3.4.2.22 TimeIndexPairStatus

The variable TimeIndexPairStatus is a 2-dimensional array that holds the trust status between two ClockTarget interfaces of the instance of the TSF. The variable TimeIndexPairStatus[x][y] holds the trust status between ClockTarget interface x and ClockTarget interface y. Valid values are NOT_TRUSTED and TRUSTED. The data type is as follows:

typedef fttmInputTrustStatus TimeIndexPairStatus [numTimeIndexes][numTimeIndexes];

20.3.4.2.23 trustState (fttmOutputTrustState)

The variable trustState gives the determined trust state of the TSF instance's state machine. Valid values are NOT_TRUSTED and TIME_TRUSTED for the default time-index selection algorithm, MVTISA (see 20.3.5).

20.3.4.2.24 x (UInteger8)

The variable x is a local variable used for looping functions in the TSF instance.

20.3.4.2.25 y (UInteger8)

The variable y is a local variable used for looping functions in the TSF instance.

20.3.4.3 TSF State diagram

The TSF state machine is shown in Figure 20-4. The default algorithm used by a TSF for its time-index selection algorithm is the MVTISA of 20.3.5, which finds all the trusted input times and selects the index that corresponds to the ClockTarget interface that gives the median trusted input time. The ClockTarget interface of the selected index is presented as the output of the TSF.

The INITIALIZE state of the TSF state machine sets up the starting values of variables. Once the starting values are configured, this state machine unconditionally transfers to the WAIT_INVOKE state.

The WAIT_INVOKE state of the TSF state machine waits for an invoke event from the FTTM state machine. When this happens, the state machine transitions to the ASSIGN_INTF_RESULT_VARIABLES state.

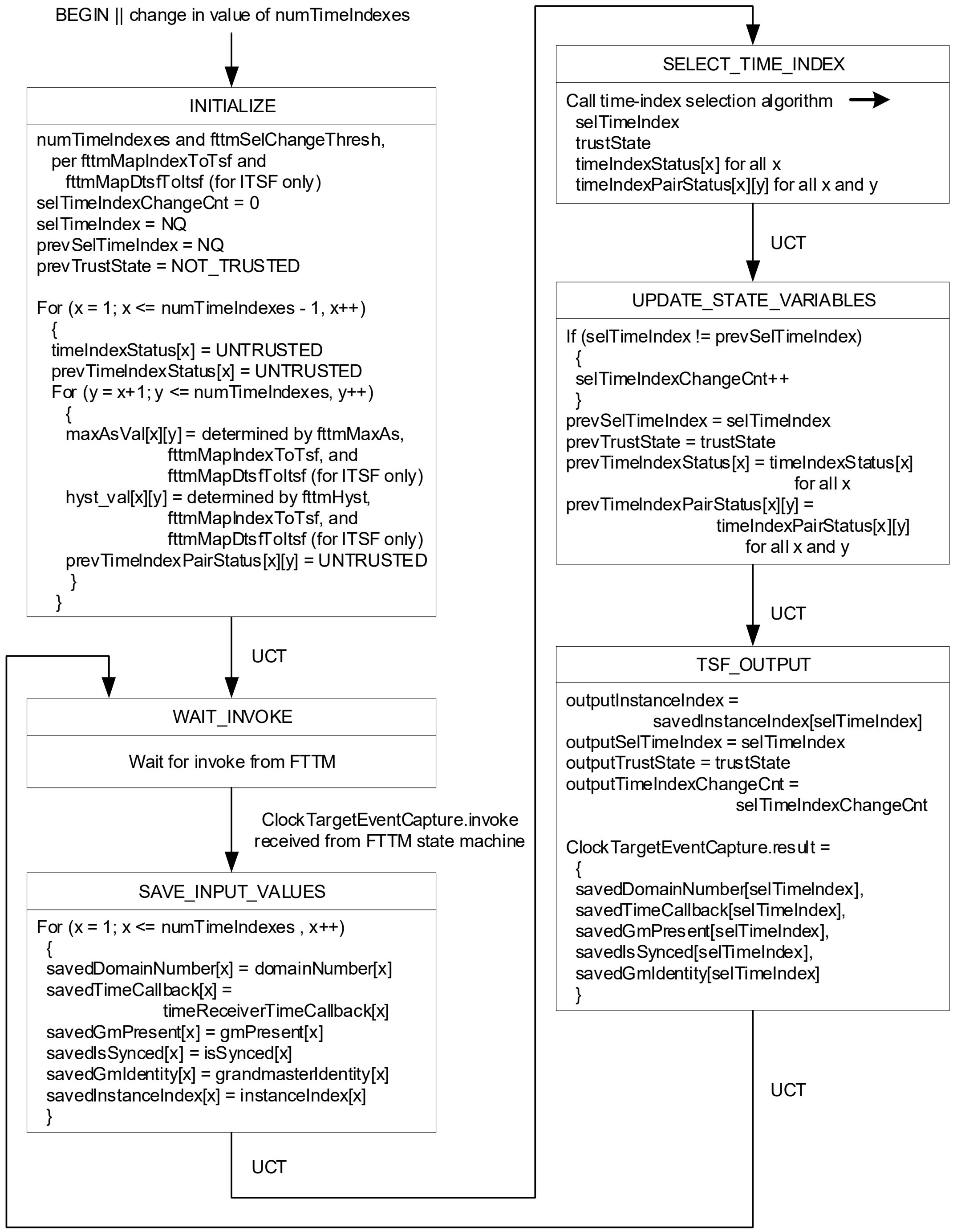


Figure 20-4 — TSF state machine

The SAVE_INPUT_VALUES state of the TSF state machine assigns the timing result parameters (domainNumber, timeReceiverTimeCallback, gmPresent, isSynced, and grandmasterIdentity, see clause 9.3.3) from each of the input ClockTargetEventCapture interfaces to TSF variables. Once this assignment is finished, this state unconditionally transfers to the SELECT_TIME_INDEX state.

The SELECT_TIME_INDEX state of the TSF state machine calls a time-index selection algorithm (with the default algorithm being the MVTISA from 20.3.3.4) to find the trusted time, if any, from the input

ClockTarget interfaces. If no trusted time is found, then the index number for the NQ time is identified. This state machine then unconditionally transfers to the UPDATE_STATE_VARIABLES state.

The UPDATE_STATE_VARIABLES state of the TSF state machine increments the selTimeIndexChangeCnt value, when appropriate, and updates the prevSelTimeIndex, prevTimeIndexStatus, prevTimeIndexPairStatus, and prevTrustState variables.

The TSF_OUTPUT state of the TSF state machine generates the ClockTargetEventCapture.result to the output of the TSF instance.

20.3.5 Mid-value time-index selection algorithm (MVTISA)

The mid-value trusted time-index selection algorithm (MVTISA) is the default time-index selection algorithm used for a TSF. Its name (see 14.23.21) is "MVTISA".

The MVTISA determines which time indexes have trusted times and then finds, amongst these time indexes with trusted times, the time index with the median time. The MVTISA is the default algorithm for the TSF function.

The MVTISA looks at all possible combinations of input time index pairs to determine which pairs satisfy their specified maximum accepted skew magnitude threshold, $maxAs_{xy}$ (see L.5). All time indexes from any time index pair that satisfies its corresponding $maxAs_{xy}$ threshold is deemed to be trusted. The time index that has the median time amongst all the trusted time indexes is selected as the output of the MVTISA. If the number of trusted time indexes is even, the selected time index is the one with the smaller index value.

20.3.5.1 MVTISA variables

The variables local to the MVTISA are described in this subclause. Other variables used by the MVTISA are passed down to the MVTISA from the TSF that the MVTISA is embedded in or passed up from the MVTISA to the TSF that the MVTISA is embedded in. These variables are described in 20.3.4.2.

20.3.5.1.1 excludeTimeIndex

This variable excludeTimeIndex[x] is a vector, of size numTimeIndexes, of Boolean values that temporarily holds the exclusion status of the trusted input ClockTarget interfaces as they are sorted into ascending order. The data type is as follows:

typedef Boolean excludeTimeIndex [numTimeIndexes];

When excludeTimeIndex[x] is TRUE, the input ClockTarget interface to the process with index x is excluded from the process.

When excludeTimeIndex[x] is FALSE, the input ClockTarget interface to the process with index x is not excluded from the process.

20.3.5.1.2 minValue (ExtendedTimestamp)

The variable minValue is a temporary value used by the MVTISA for sorting the times from the trusted ClockTarget interfaces, in time ascending order.

20.3.5.1.3 numSorted (UInteger8)

The variable numSorted holds a temporary count of the number of trusted time indexes that have been sorted in time ascending order.

20.3.5.1.4 orderedTimeIndex

The variable orderedTimeIndex is a vector of UInteger8 values, each of which correspond to an index number of a trusted input ClockTarget interface to the MVTISA. The vector member orderedTimeIndex[x] contains the Xth entry (starting from 1) of the trusted ClockTarget interface index numbers, ordered from lowest to highest ToD value. The data type of is as follows:

typedef UInteger8 orderedTimeIndex [numTimeIndexes];

20.3.5.1.5 TodDiff

The variable TodDiff is a 2-dimensional array, where the array member TodDiff[x][y] gives the magnitude of the time skew between the timeReceiverTimeCallback values of two ClockTarget interfaces, x and y, given as:

$$\begin{aligned} TodDiff[x][y] = |&\text{timeReceiverTimeCallback of ClockTarget Interface x} \\ &- \text{timeReceiverTimeCallback of ClockTarget Interface y}| \end{aligned}$$

For this process, each dimension of the array has a size of numTimeIndexes and the data type is as follows:

typedef ExtendedTimestamp TodDiff [numTimeIndexes][numTimeIndexes];

20.3.5.1.6 x (UInteger8)

The variable x is a local variable used for looping functions.

20.3.5.1.7 y (UInteger8)

The variable y is a local variable used for looping functions.

20.3.5.2 MVTISA pseudo-code

Pseudo-code that represents the MVTISA is given below.

```
// ########################################################################
// Gather the current skews between the ToDs of all the time indexes.
// ########################################################################
For (x = 1; x <= numTimeIndexes - 1, x++) {
    For (y = x + 1, y <= numTimeIndexes, y++) {
        TodDiff[x][y] = |savedTimeCallback[x] - savedTimeCallback[y]|
        }
    }

// ########################################################################
// Clear status before starting a new round of time index comparisons.
// ########################################################################
trustState = NOT_TRUSTED
timeIndexPairStatus[x][y] = UNTRUSTED for all x and all y
timeIndexStatus[x] = UNTRUSTED for all x
numSorted = 1
excludeTimeIndex[x] = FALSE for all x

// ########################################################################
// Find all trusted time indexes, considering hysteresis.
// ########################################################################
For (x = 1, x <= numTimeIndexes - 1, x++) {
    For (y = x + 1, y <= numTimeIndexes, y++) {
```

```
        if ((TodDiff[x][y] <= maxAsVal[x][y] &&
             prevTimeIndexPairStatus[x][y] == UNTRUSTED) ||
             (TodDiff[x][y] <= (maxAsVal[x][y] + hystVal[x][y]) &&
              prevTimeIndexPairStatus[x][y] == TRUSTED)) &&
            (savedIsSynced[x] && savedGmPresent[x]) &&
            (savedIsSynced[y] && savedGmPresent[y]))
        {
             // trust found for the pair
             trustState = TIME_TRUSTED
             timeIndexPairStatus[x][y] = TRUSTED
             timeIndexStatus[x] = TRUSTED
             timeIndexStatus[y] = TRUSTED
        }
        else
        {
             // trust not found for the pair
        }
    }
}

// ##########################################################################
// If trustState = TIME_TRUSTED, find time index with the mid-value ToD.
// ##########################################################################
// Trusted times detected.
If {trustState == TIME_TRUSTED)
{
  // Sort all trusted time indexes in order of their ToD, from smallest
  // to largest, using two loops.
  // Outer loop iterates over all time indexes.
  For (x = 1, x <= numTimeIndexes, x++) {
    minValue = 2^48 seconds  // Start with ToD value that is larger
                             // than any possible gPTP ToD value.
    // Inner loop finds and records the time index with the minimum ToD
    // value and excludes it from further iterations of the outer loop.
    For (y = 1, y <= numTimeIndexes, y++) {
       if (timeIndexStatus[y] == TRUSTED &&
           excludeTimeIndex[y] == FALSE &&
           savedTimeCallback[y] <= minValue)
       {
         minValue = savedTimeCallback[y] // record latest min ToD value found
                                         // in the inner loop
         orderedTimeIndex[numSorted] = y  // record latest time index
                                          // with min ToD value
       }
    }
    // Exclude latest time index with the min ToD value and add to sort index.
    excludeTimeIndex[orderedTimeIndex[numSorted]] = TRUE
    numSorted = numSorted + 1
  }

  // Get median trusted time index.
  //
  // Check if the following are true:
  //  -The magnitude of the time difference between the previously
  //   selected median trusted time index and the current median
  //   trusted time index is less than or equal to the value of
  //   fttmSelChangeThresh for this TSF.
  //  -The previously selected median trusted time index is still trusted.
  // If both are true, then continue using the previously selected median
  // Trusted time index.
  // Otherwise, change the selection to the current median trusted time index.
  //
  // When changing the selection, the lower time index is selected if the
  // number of trusted time indexes is even.
```

```
    If ((timeIndexStatus[prevSelTimeIndex] == TRUSTED) &&
        (TodDiff[prevSelTimeIndex][orderedTimeIndex[numSorted)/2] <=
                                                    fttmSelChangeThresh))
    {
      selTimeIndex = prevSelTimeIndex
    }
    else
    {
      selTimeIndex = orderedTimeIndex[INT((numSorted)/2)]
    }
}

// No time trust or frequency trust so set output time indexes to NQ
// and clear previous trust status to NOT_TRUSTED.
else
{
  selTimeIndex = 511  // NQ
  trustState = NOT_TRUSTED
}
```

Annex A (normative)

Protocol Implementation Conformance Statement (PICS) proforma[2]

A.5 Major capabilities

Add a row at the end of Table A.5 as follows:

Item	Feature	Status	References	Support
FTTM1	Does the time-aware system implement the fault-tolerant timing module as specified in 20.3.1 through 20.3.4?	O	5.3, 9.3, 14.23, 17.6.4, 20.3	Yes [ ] No [ ]
FTTM2	Does the time-aware system implement the fault-tolerant timing module using an alternative to the default time-index selection angorithm specified in 20.3.1 through 20.3.4?	O	State what algorithms supported	Yes [ ] No [ ] [ ]

A.19 Remote management

Add a row at the end of Table A.19 as follows:

Item	Feature	Status	References	Support
RMGT-6	If a remote management protocol that supports YANG is listed in RMGT-2, and if the Fault-Tolerant Timing Module is supported, is the YANG data model ieee802-dot1as-fttm of Clause 17 supported?	RMGT: O	5.3, Clause 17, 20.3	Yes [ ] No [ ] N/A[ ]

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

Annex H (informative)

Bibliography

Insert the following items alphabetically into the bibliography and renumber as appropriate.

[B1] L Lamport, PM Melliar-Smith, Synchronizing clocks in the presence of Faults, 1982.

[B2] D Dolev, J Halpern, On the Possibility and Impossibility of Achieving Clock Synchronization, 1984.

[B3] P Miner, A Geser, L Pike, Jeffery Maddalon, A Unified Fault-Tolerance Protocol, 2004.

[B4] P Ramanathan, KG Shin, RW Butler, Fault-Tolerant Clock Synchronization in Distributed Systems, 1990.

[B5] M Pease, R Shostak, L Lamport, Reaching Agreement in the Presence of Faults, 1980.

[B6] IEEE Std 1588aTM-2023, IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Amendment 3: Precision Time Protocol (PTP) Enhancements for Best Master Clock Algorithm (BMCA) Mechanisms.

[B7] ITU-T G.8232/Y.1344, Ethernet ring protection switching

Insert the following annex.

Annex I (informative)

Time synchronization with Fault Tolerance and Integrity

I.1 Introduction

This Annex provides examples for time synchronization with fault tolerance and time integrity.

Time-sensitive applications that require fault tolerance are expected to tolerate multiple (typically 2) simultaneous arbitrary faults in end stations, Bridges, links, and GMs while maintaining availability and integrity of time synchronization.

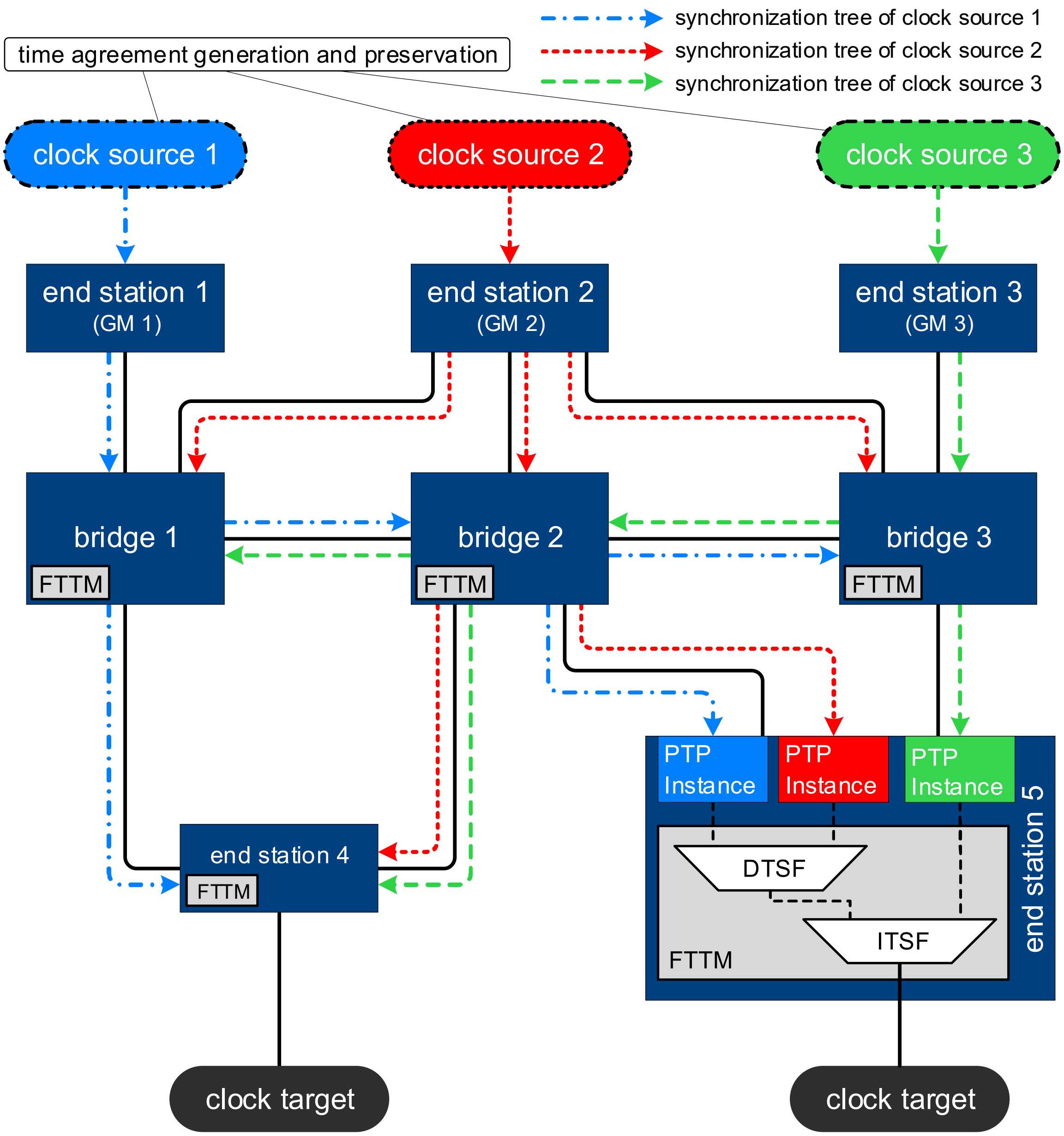
Fault tolerance, or availability, and integrity, which covers the availability and the trustworthiness of time, addresses the need for reliable and accurate distribution of time in the presence of arbitrary faults in the network (link, Bridge, end station, and GM). Thus, under fault conditions, a correctly operating end station is expected to maintain a target maximum time error relative to the correctly operating GM. If unable to remain within the maximum time error, the correctly operating end station will detect an erroneous time synchronization state. To support this, it is expected that multiple clock domains are configured and managed in the network.

I.2 Clock domain management

This standard supports the use of multiple domains, redundant synchronization trees, and potential reorganization of the synchronization trees, via the BTCA, to overcome faults in the time synchronization function. The BTCA is not compatible with some applications that have stringent fault-tolerance requirements. First, a reorganization of the synchronization tree to overcome a fault takes time to complete. Depending on the size of the network and where the fault occurred, there might be a high degree of variability in the fault recovery time of the BTCA. Second, many of these applications use engineered networks with static configuration and deterministic behavior and, thus, cannot use the BTCA.

The FTTM mechanism defined in this standard uses multiple domains with multiple (potentially redundant) synchronization trees to provide configurable and deterministic fault recovery behavior. The FTTM also enables the support of time integrity functions.

To illustrate how the FTTM works in an example application, Figure I-1 shows a simple network using three clock sources distributing time through multiple time synchronization trees to bridges and end stations that implement FTTMs to select a time source from the multiple domains. The existence of multiple overlapping timing domains presents a problem for the forwarding of scheduled traffic because each port can only forward traffic using one clock reference. The three clock sources in the example are therefore assumed to share a common time through implementation of a time agreement and preservation function, 20.2.3, to allow the forwarding of time-aware traffic across domains. The time agreement and preservation function is implementation-specific and is not specified in this standard. The clock sources in the example figure synchronize three GMs in end stations 1, 2 and 3, that distribute time through the network on separate, but overlapping, synchronization trees. Bridges 1, 2 and 3 distribute time from the three GMs through the network to all bridges and end stations where FTTMs select the best available clock to generate a local clock target that is used to support the enhancements for scheduled traffic described in IEEE Std 802.1Q, including the transmission and forwarding of scheduled traffic.



NOTE 1 — The methods to synchronize the ClockSources of all GMs are not specified in this standard
NOTE 2 — All the "bridges" in this figure are examples of time-aware systems that contain PTP Instances and a fault-tolerant timing module
NOTE 3 — All the end stations in this figure are examples of time-aware systems that contain PTP End Instances and fault-tolerant timing module

Figure I-1— Multiple domains with FTTMs

End station 5 shows 3 PTP Instances receiving time through the network and providing clock target interfaces to an FTTM consisting of one DTSF and one ITSF as instantiations of time selection functions (TSFs), 20.3.2.2. Clock sources 1 and 2 arrive at end station 5 through a common bridge device, bridge 2, and are therefore considered to be dependent and require a DTSF to select the best clock source to provide this as an independent input to the ITSF along with the output of the PTP Instance that uses clock source 3. The time from clock source 3 arrives at end station 5 through a path that is independent from the path taken by clock sources 1 and 2, and can therefore be considered to be independent from these. The output of the ITSF is then used to generate the local clock target for end station 5.

As shown in the figure, each of the bridges in the network and end station 4 also include FTTMs to generate local clock targets. FTTMs can be configured differently in each device in the network depending upon the relationships of the available clock sources to one another, and select a clock source to generate the local clock target that is used to support generation and forwarding of time-aware traffic. To simplify the figure, end stations 1, 2 and 3 are not shown as recipients of the synchronization trees from the other two clock sources and do not show FTTMs; however, in a real network application they might be expected to receive the other time signals and to contain FTTMs to support fault-tolerant traffic generation. Once the three clock sources have established time agreement and each of the FTTMs have selected the best clock at each of network nodes, then all of the clock targets will be synchronized and time-aware traffic from any end point in the network can be forwarded synchronously to any other end point through the available bridges.

Under normal operating conditions, each of the nodes in the example network will receive time from each of the three domains and generate a local clock target by selecting the best clock source according to the configured FTTM attributes. In the case of a fault in the network it can be expected that either one or two of the time signals will be unavailable at any particular bridge or end station and the FTTM will select the best available clock from those remaining, By example, if the link between bridges 2 and 3 were to fail, the synchronization tree from clock source 3 would not propagate to bridge 2 and to end station 4, and the synchronization trees from clock sources 1 and 2 would not propagate to bridge 5 and to end station 5. In this case, end station 5would only be able to select the time from clock source 3, and end station 4 would select the best clock between the two independent domains of clock source s 1 and 2. Redundant links from end station 2 to bridge 1 and to bridge 3, carrying redundant time signals, could in this case be added to provide improved time availability.

This simple example does not address time integrity. See I.4 for a discussion of integrity and availability.

I.3 Time agreement generation and preservation examples

Examples of time agreement generation and preservation implementations are shown in this annex. These implementations need to be tolerant to an occurrence or some occurrences of the following fault types, which include Byzantine faults. All implementations have to reduce the possibility and/or probability of occurrence for all of these fault types.

— Symmetric omissive
— Symmetric transmissive
— Asymmetric omissive
— Asymemtric transmissive

Symmetric faults are faults that are the same for all the intended users of the information. Asymmetric faults are faults that are different for some of or all the intended users of the information. Omissive faults are faults where the information is not given to the intended user. Transmissive faults are faults where incorrect information is given to the intended user. Byzantine faults are asymmetric.

It is commonly accepted (see [B1]) that 3N + 1 Clock Sources are required to achieve tolerance for N Byzantine faults. This would be sufficient to achieve tolerance for N faults of any of the types listed above. The examples shown in this annex have four Clock Sources and, thus, can tolerate one Byzantine fault.

It is not necessary for a FTTM to have four input times to achieve time integrity for a Clock Target because, unlike the time agreement and preservation function, the time distribution function does not need to be tolerant to Byzantine faults.

I.3.1 PPS-based implementation

An example PPS-based time agreement generation and preservation implementation is shown in Figure I-2.

In this implementation, PPS events are broadcast from each Clock Source to all the other Clock Sources. These PPS events and their corresponding times are used to syntonize the frequencies and synchronize the times of all the Clock Sources.

The Tuning Timer creates and broadcasts a periodic signal (e.g., a pulse) to each of the Clock Sources to coordinate the time that they perform their comparison and tuning operations. For this periodic signal, it is important that it occurs at approximately the expected period. This period does not have to be exact. It only needs to be no less than the time needed for all the Clock Sources to complete their comparison and tuning operations and no longer than the time in which any Clock Source could drift away from its previously tuned value with a significance that would make it appear to have lost synchronization. Any Clock Source that does not detect this periodic signal within an expected time period would declare a fault condition. Adding redundancy to this tuning timer mechanisms could improve the fault tolerance of the system.

For synchronization, each Clock Source could detect whether the incoming PPS events arrive within an expected time period of each other, exclude the Clock Sources (including itself) that do not meet this expectation, and then tune its local clock frequency to match the average of the clock frequencies of all the

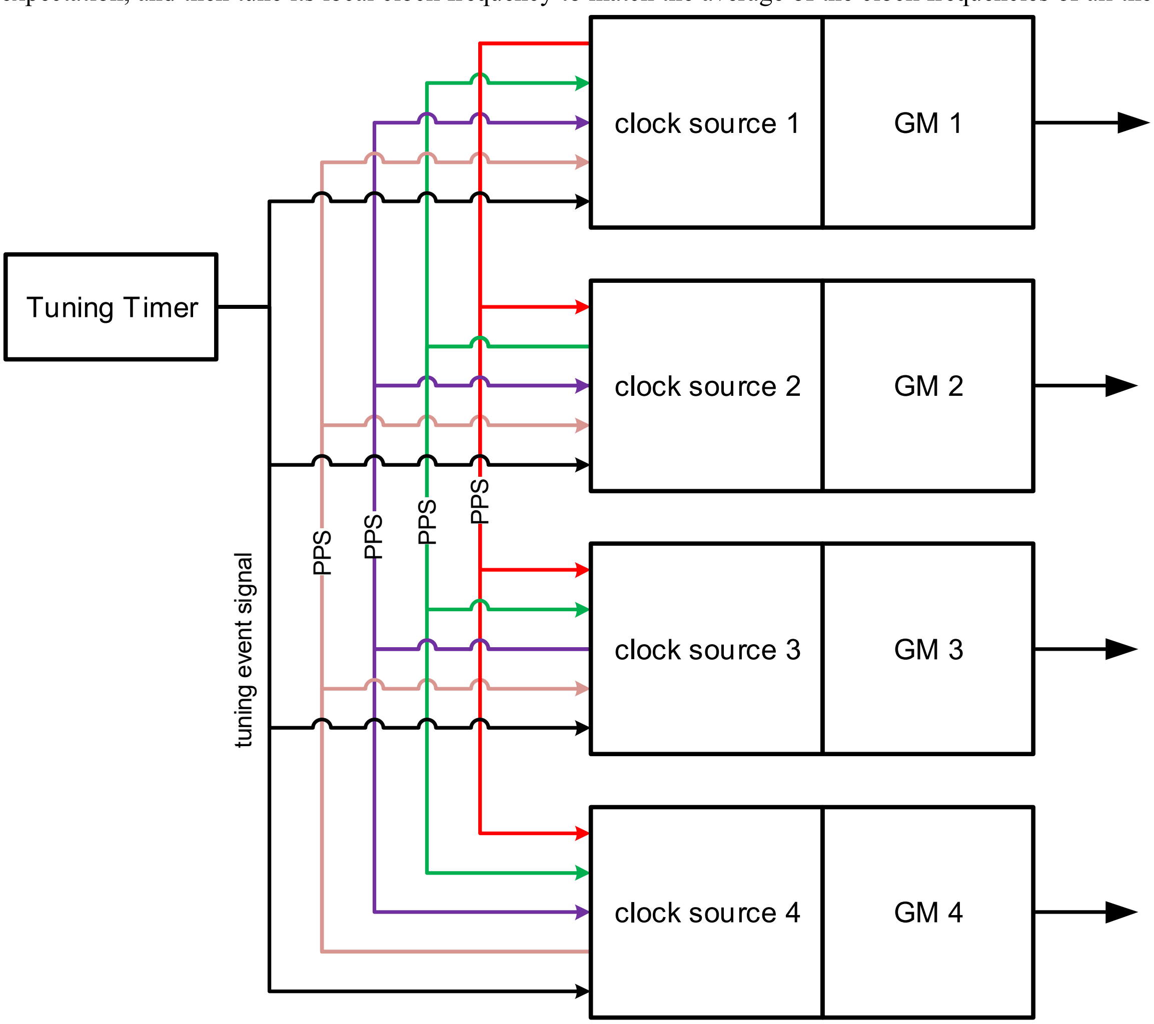


Figure I-2— Example PPS-based time agreement generation and preservation implementation

non-excluded Clock Sources. Once it is syntonized to all the non-excluded Clock Sources, it can perform a similar compare, exclude, average, and alignment process for its time-of-day. Once this initial syntonization and time alignment is achieved, each Clock Source could continuously perform a process that checks the times of the incoming PPS events, excludes those that differ by a pre-determined threshold, and tune itself to the average time of the non-excluded Clock Sources. It can restart the initial syntonization and time alignment processes if the status of non-excluded Clock Sources drops below a certain threshold.

UARTs are commonly used to convey the time-of-day associated with PPS events. This mechanism must also be implemented to be tolerant to Byzantine faults.

I.3.2 PTP-based implementations.

An example PTP-based time agreement generation and preservation implementation is shown in Figure I-3.

In this implementation, PTP Sync messages are broadcast from a PTP port, which is configured to be in the PASSIVE state (see [B1]), of each Clock Source through a high integrity PTP message distributor with PTP Transparent Clock (see [B1]) capabilities to a PTP Port, which is also configured to be in the PASSIVE state, on all the other Clock Sources. These PTP Sync messages and their corresponding departure times and arrival times are used to syntonize the frequencies and synchronize the times of all the Clock Sources.

.

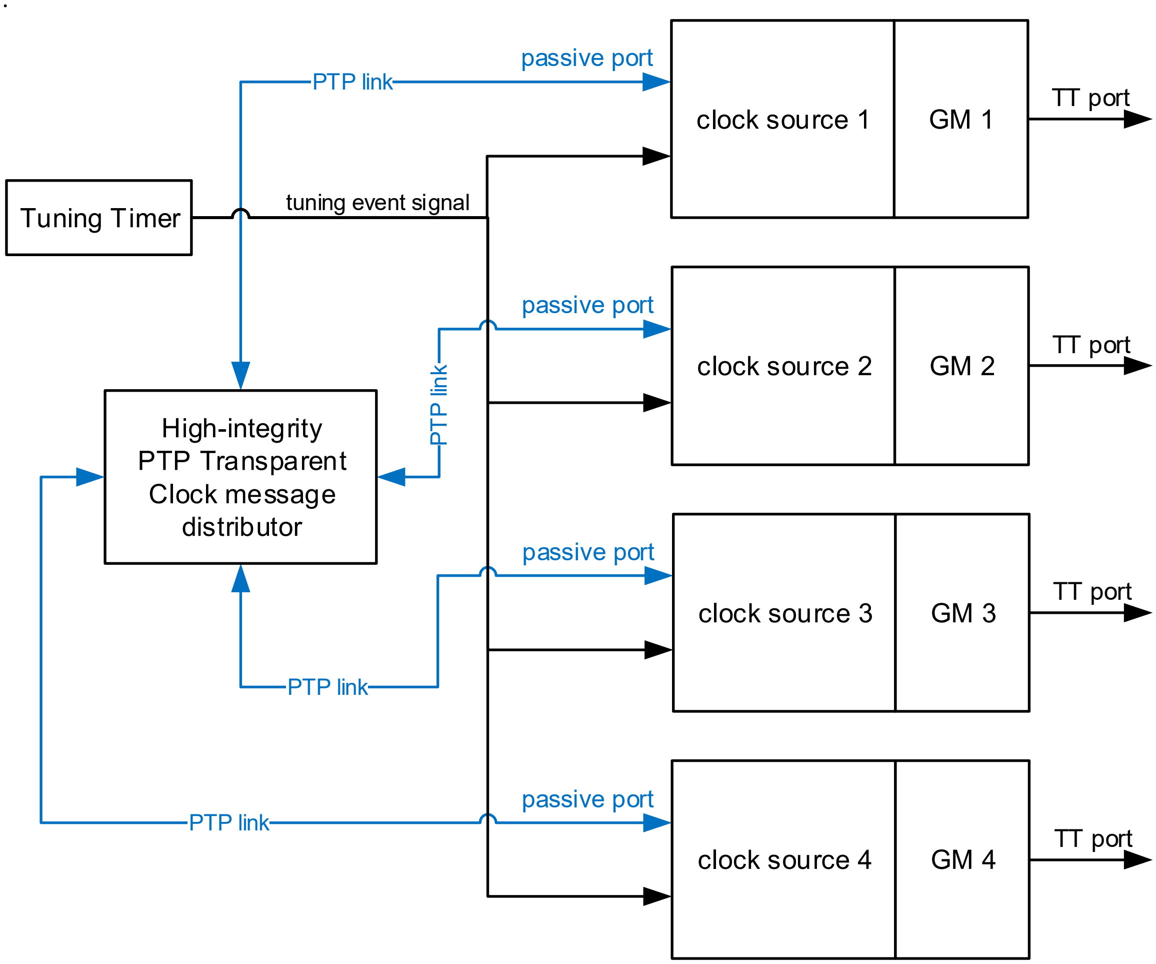


Figure I-3— Example PTP-based time agreement generation and preservation implementation

The integrity of the PTP message distributor is critical in this implementation because it is a common potential source of failures in the conveyance of the PTP information between the Clock Sources. This PTP message distributor needs to reduce the probability of it creating false information while distributing the PTP Sync messages. This could be done by adding redundancy and error checking to the PTP message distributor's timestamping, correctionField updating, and broadcasting functions.

The Tuning Timer creates and broadcasts a periodic signal (e.g., a pulse) to each of the Clock Sources to coordinate the time that they perform their comparison and tuning operations. For this periodic signal, it is important that it occurs at approximately the expected period. This period does not have to be exact. It only needs to be no less than the time needed for all the Clock Sources to complete their comparison and tuning operations and no longer than the time in which any Clock Source could drift away from its previously tuned value with a significance that would make it appear to have lost synchronization. Any Clock Source that does not detect this periodic signal within an expected time period would declare a fault condition. Adding redundancy to this tuning timer mechanisms could improve the fault tolerance of the system.

For synchronization, each Clock Source could measure the frequency offsets of the incoming PTP Sync messages and see which match within an expected range, exclude the Clock Sources (including itself) that do not meet this expectation, and then tune its local clock frequency to match the average of the clock frequencies of all the non-excluded Clock Sources. Once it is syntonized to all the non-excluded Clock Sources, it can perform a similar compare, exclude, average, and alignment process for its time-of-day. Once this initial syntonization and time alignment is achieved, each Clock Source could continuously perform a process that checks the times of the incoming Sync messages, excludes the Clock Sources that differ by a pre-determined threshold, and tune itself to the average time of the non-excluded Clock Sources. It can restart the initial syntonization and time alignment processes if the status of non-excluded Clock Sources drops below a certain threshold.

I.4 Balancing availability and integrity

The following compromises to fault-tolerant timing and time integrity are common in many implementations:

— Use of a common clock source for all GMs.
— Use of only two domains instead of three or more.
— An insufficient number of independent paths from the GMs to the FTTM.

For the first listed compromise, some applications only need the time-aware components of the local system to be synchronized to the arbitrary timescale of the local system and the integrity of this timescale does not need to be protected. In this scenario, a single clock source running off a local oscillator might be sufficient to satisfy the time agreement and preservation requirements of the system and be used as the source for all the GMs in the system. An example network topology with this compromise is discussed in A.1.4.

For the second listed compromise, some applications have a fail-stop requirement instead of a fail-operational requirement. A fail-operational requirement means the application has to continue operating correctly even if a fault (or up to N faults) has been detected. A fail-stop requirement means the application stops operation once a fault is detected. Only two independent times are needed to satisfy a fail-stop requirement, and this can be achieved using two domains. Example network topologies with this type of compromise are discussed in I.5. Example network topologies with this compromise are discussed in A.1.3, A.1.4, and A.1.5.

For the third listed compromise, it can be difficult and perhaps even impossible to create independent paths from every GMs to every FTTM in a network. For these networks, the enhanced availability and integrity scenario (see 20.3) can still be achieved for fail-stop applications if there are at least two independent paths from two GMs to the FTTM. Otherwise, if there are no independent paths from any of the GMs, then the

FTTM is only able to operate in the enhanced availability and limited integrity scenario (see 20.3). Example network topologies with this compromise are discussed in A.1.5 and A.1.6.

I.5 FTTM operation in example network topologies

The use of FTTMs in examples of commonly used network topologies is shown in this annex, with details on the potential enhancements to time availability and time integrity.

I.5.1 N x point-to-point networks

An example N × point-to-point network topology is shown in Figure I-4. This network topology provides N independent times from N GMs to a ClockTarget entity, through its PTP Instances and its FTTM.

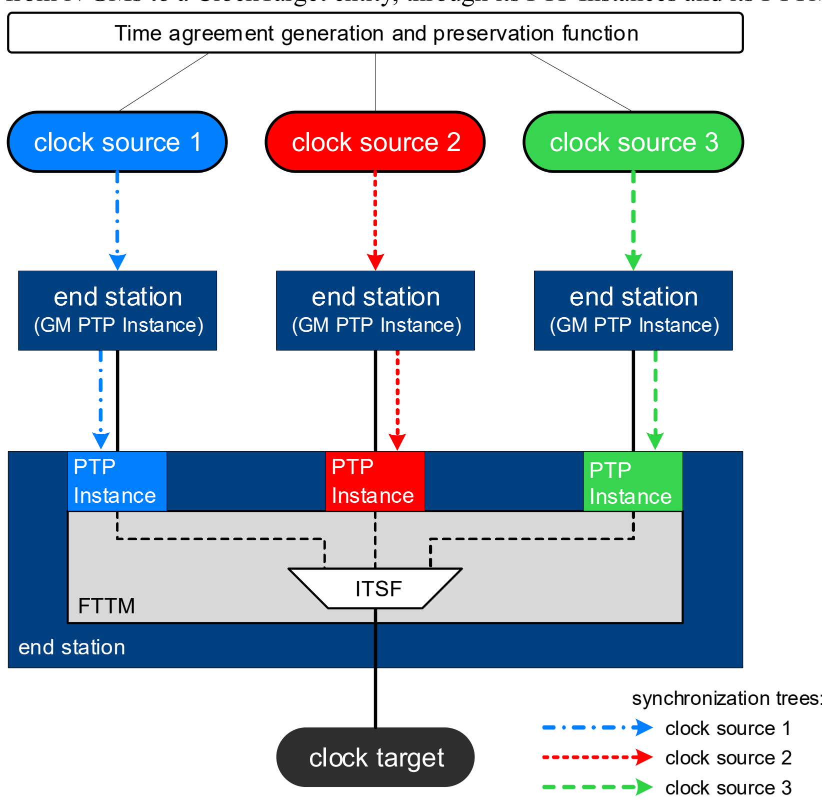


Figure I-4— FTTM in example 3 × point-to-point network

In a scenario with N = 2, time availability with time integrity is achievable only when there are no faults. Thus, a 2 × point-to-point network topology would be suitable for fail-stop applications, which terminate operation when any fault is detected in the time integrity function.

In the scenario of Figure I-4, with N = 3, time availability with time integrity is achievable when there is up to one fault. Thus, a 3 × point-to-point network topology would be suitable for fail-operational applications that can continue running when there is up to one fault in the time integrity function.

I.5.2 Dual-homed network

An example dual-homed network is shown in Figure I-5. It provides two independent times from just one GM to a ClockTarget entity, through its associated PTP Instances and FTTM.

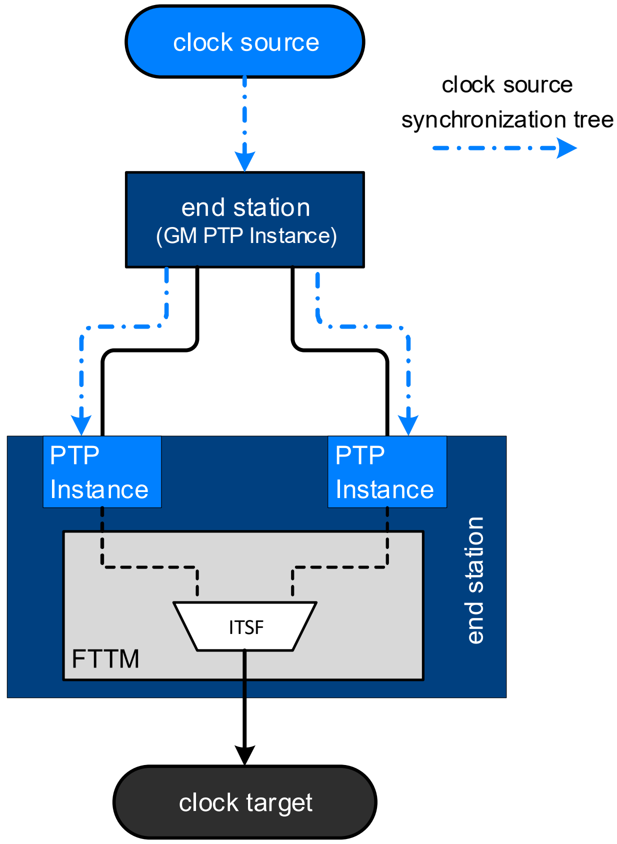


Figure I-5— FTTM in example dual-homed network

Because there is just one GM in this network topology, the FTTM operates in the enhanced availability and limited integrity scenario (see 20.3). It supports increased time availability and time distribution integrity, but not GM integrity. Also, because there are just two time distribution paths, the limited integrity is lost if one of these two time distribution paths becomes faulty. Thus, the dual-homed network topology would be suitable for fail-stop applications in which the integrity of time distribution is important, but the integrity of the GM itself is not important.

I.5.3 Dual-star network

In dual redundant star network topologies, every end station is connected independently, in a star fashion, to a bridge and to a redundant bridge. A failure of any connection or of any bridge is overcome by using the second connection or the second bridge.

The example dual-star network shown in Figure I-6 provides two independent times, one from each GM to every ClockTarget entity, through their associated PTP Instances and FTTMs. This dual-star network also provides two dependent times (one associated with each of the independent times), but with only two GMs, these dependent times do not enhance the FTTM's ability to determine time integrity. Thus, this dual-star network topology would be suitable for fail-stop applications in which the integrity of time is important.

The example dual-star network shown in Figure I-7provides three independent times from each of the three GMs to the two bridges, two independent times from each of two GMs to any End Station, and two dependent times from a third GM to any End Station, all passing through the corresponding bridge's or End Stations' FTTM.

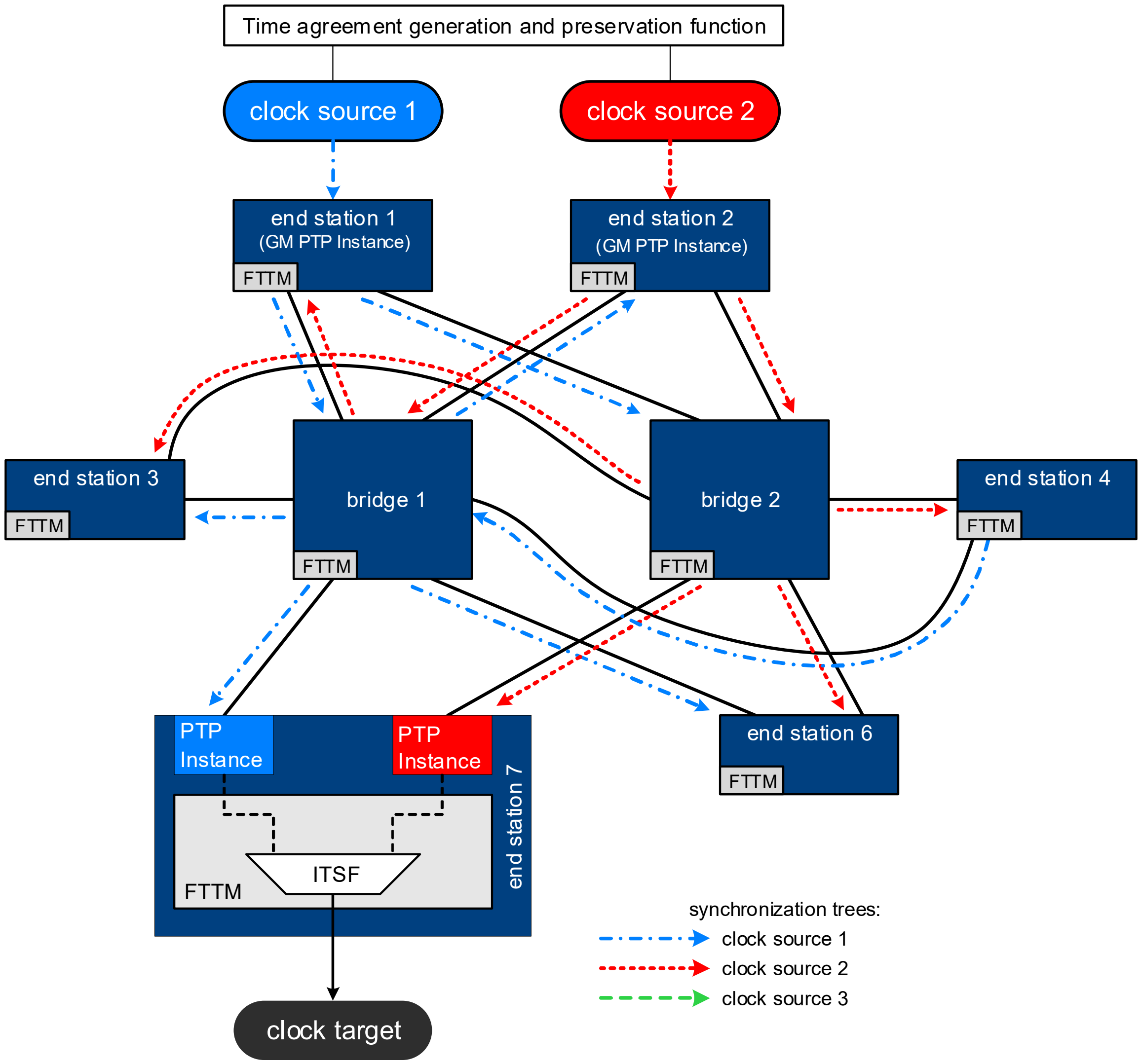


Figure I-6— FTTM in example dual-star network with fail-stop time integrity

Even though the two dependent times from clock source 3's GM to any End Station have more than one dependency (i.e., the common clock source 3 and either bridge 1 or bridge 2), the FTTM's DTSF only uses the dependency on clock source 3 to determine the trustworthiness of these times for the following reasons.

- Using clock source 3 as the dependent element results in the use of one DTSF, with two input times, and generates three input times to the ITSF.
  - If the DTSF determines that the two arriving times from clock source 3 match and, thus, can be trusted, then there is no need to pass clock source 3 through a DTSF with either clock source 1 or with clock source 2 as this is done in the ITSF.
  - If the DTSF determines that the two arriving times from clock source 3 do not match and, thus, cannot be trusted, then clock source 3 does not provide an applicable input to the ITSF and is removed from contention as a trusthworthy source of time to the ClockTarget entity. With this result, there is again no need to pass clock source 3 through a DTSF with either clock source 1 or with clock source 2.
  - The result is three input times to the ITSF, which enables trust determination when there is up to one fault.

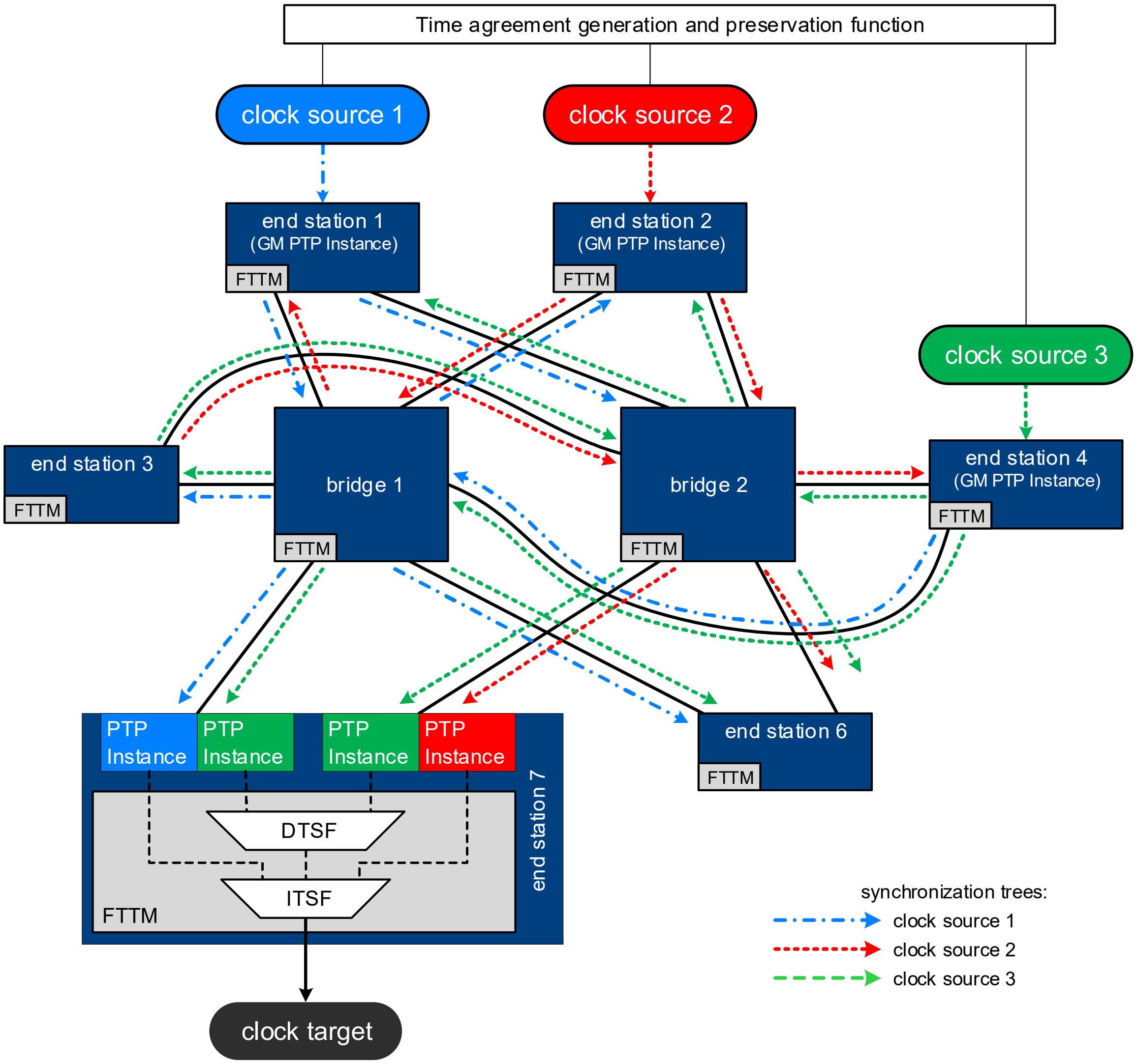


Figure I-7— FTTM in example dual-star network with fail-operational time integrity

— Using bridge 1 and bridge 2 as the dependent element results in the use of two DTSFs, each with two input times. In turn, this results in just two input times to the ITSF. Not finding trust in either DTSF results in an inability to determine trust at the ITSF. This configuration can detect a fault but cannot find trust when there is one fault in the system.

This dual-star network topology is suitable for fail-operational applications that can continue running when there is up to one fault in the time integrity function.

I.5.4 Ring network

A basic ring network topology provides a low fan-out network with availability recovery for a single fault. However, the following fundamental characteristics of the ring network topology impede the FTTM's goals of maintaining time availability and integrity during fault conditions:

— The low fan-out characteristic of a ring can prevent an FTTM from receiving at least two independent times and, thus, impedes its ability to achieve enhanced availability and integrity (see 20.3).

— The forwarding path of traffic, which includes PTP messages, is rearranged based on the location of a fault in the ring. This could lead to delay or loss of PTP messages, which in turn could affect a PTP timeReceiver's ability to retain a good PTP time.

The Ethernet ring protection mechanisms defined in ITU-T G.8232 [B7] use a mechanism called the ring protection link (RPL) to introduce a break in the ring by blocking frames. This break prevents the formation of loops in the ring, and it determines which direction around the ring a frame takes to get to its destination. When a failure introduces another break in the ring, the nodes on both sides of this break are reconfigured to block frames and the RPL is reconfigured so it no longer blocks frames, which opens a new path for frames to get around the ring.

The ring network shown in Figure I-8 provides two independent times, one coming from two dependent times (because they share at least one common bridge on the ring), to each bridge in the ring because the break in the ring (due to the RPL) does not require any bridge to receive all three domains from the same ring direction. For simplicity, the end stations that connect to these bridges are not shown.

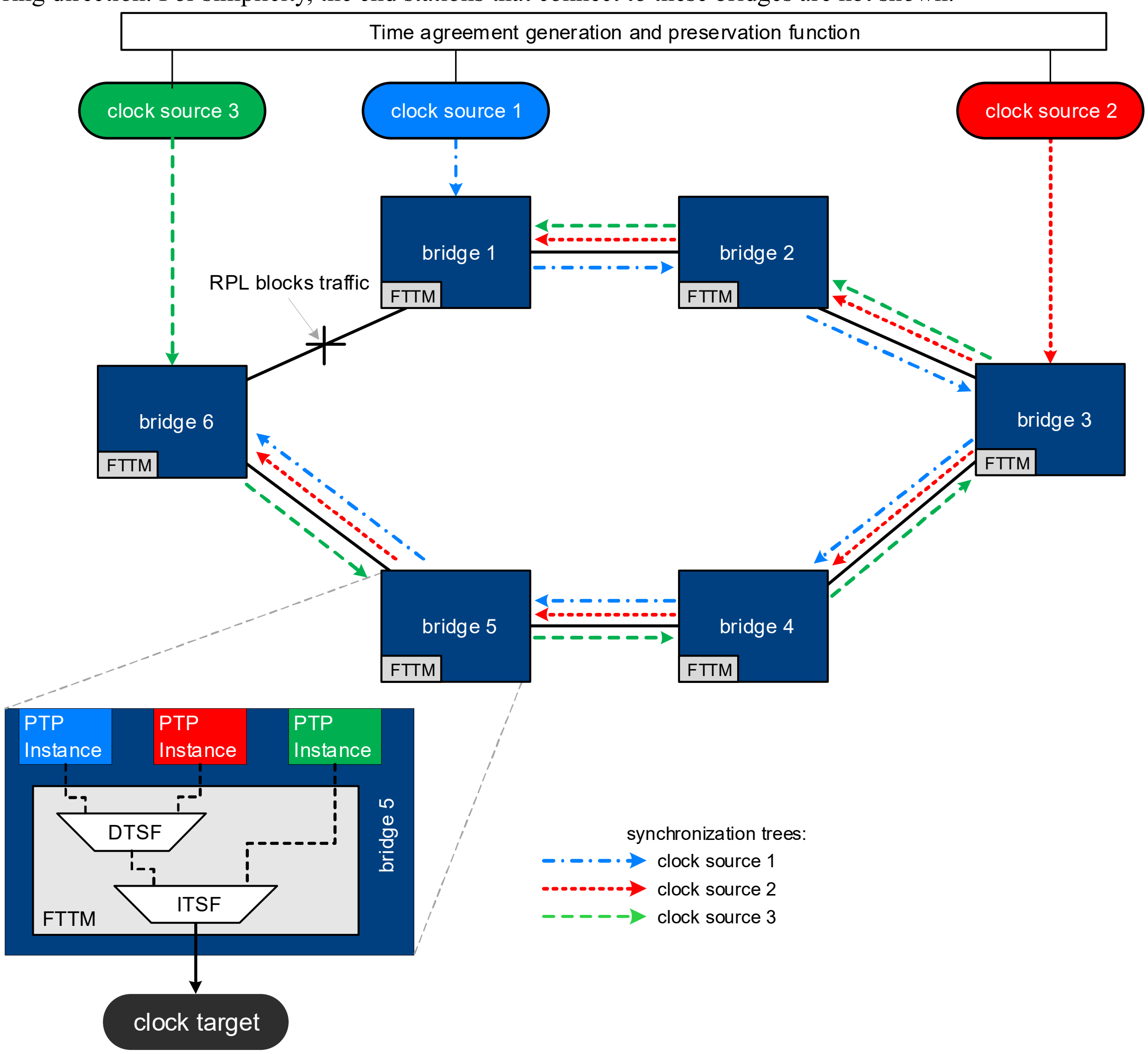


Figure I-8— FTTM in example a ring network

The same ring network is shown in Figure I-9 with a repositioned break in the ring, resulting from a fault. With this repositioned break, this ring network can provide two independent times, one coming from two dependent times (because they share at least one common bridge on the ring), to only four of the six bridges. The other two bridges are provided with three dependent times (they all share at least one common bridge on the ring). As a result, these two bridges can only operate in the enhanced availability and limited integrity scenario and not in the enhanced availability and integrity scenario (see 20.3).

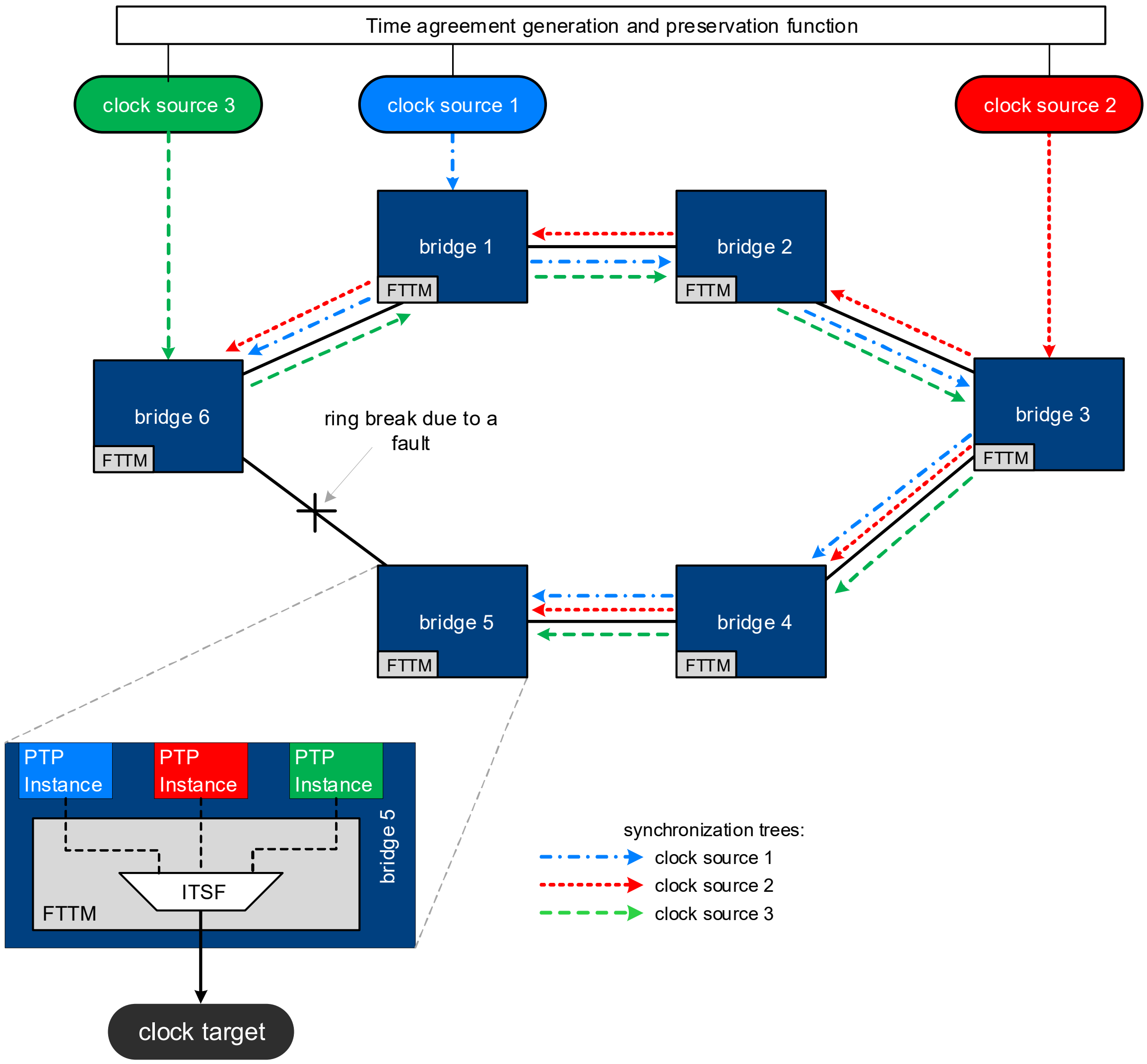


Figure I-9— FTTM in example a ring network with repositioned break

I.5.5 Mesh networks

A mesh network topology connects every network element directly to every other network element. The failure of one element does not adversely affect the ability of another working element to communicate with any other working element in the mesh.

Figure I-10 shows a mesh network of bridges with three GMs. For simplicity, the end stations that connect to these bridges are not shown.

Because every bridge has a direct connection to each GM, it can receive an independent time from each of the three GMs (this is shown for only one bridge in Figure I-10). Thus, this network topology would be suitable for fail-operational applications that can continue running when there is up to one fault in the time integrity function.

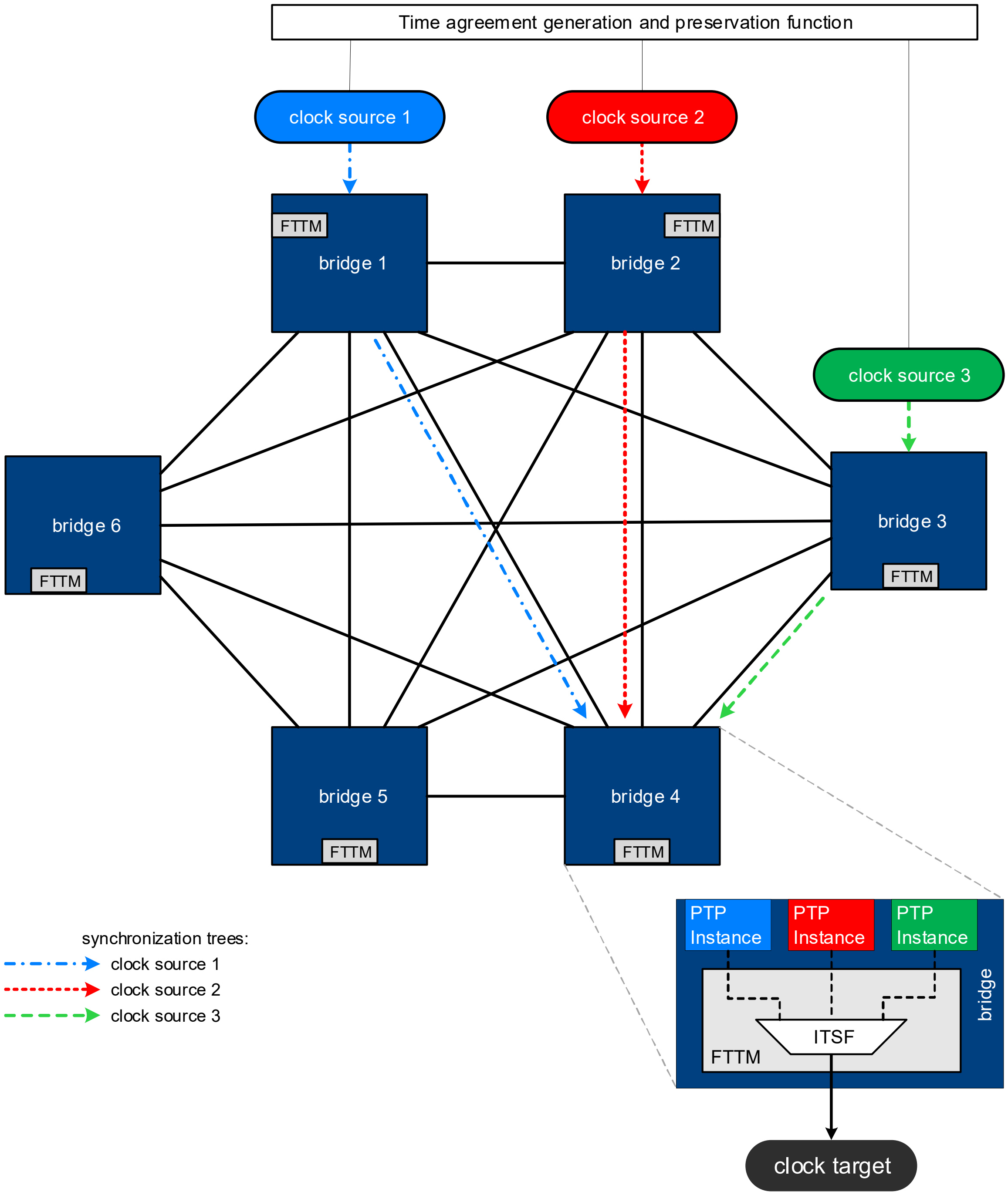


Figure I-10— FTTM in example mesh network

Insert the following annex.

Annex J (informative)

FTTM Configuration Examples

This annex offers a detailed guide through a specific configuration example that utilizes simple YANG instance data that has been validated in accordance with the ieee802-dot1as-fttm module. The explanatory text provided between segments of XML instance data clarifies its alignment and connection with the FTTM service described in Clause 20.

J.1 Initial Setup

The example begins with a foundational XML structure that incorporates Precision Time Protocol (PTP) instance data, defined within the ptp element. This element is identified by its namespace urn:ieee:std:1588:yang:ieee1588-ptp-tt, indicating its adherence to the IEEE 1588 PTP standard tailored for telecommunications.

```
<ptp xmlns="urn:ieee:std:1588:yang:ieee1588-ptp-tt">
  <common-services>
    <fault-tolerant-timing-module-service
xmlns="urn:ieee:std:802.1AS:yang:ieee802-dot1as-fttm">
```

J.2 Fault-Tolerant Timing Module Management

Within this structure, we delve into the configuration of a Fault-Tolerant Timing Management (FTTM) system. The XML snippet introduces a fttm-system element, which houses a unique fault-tolerant-timing-function-index. This example illustrates a scenario with a single FTTM instance, identified by index 1.

```
      fttm-system>
        <fttm-system-index>1</fttm-system-index>
```

J.3 FTTM Configuration Details

Further detailing the FTTM configuration, the fttm-system-ds element specifies the system's capabilities, including the maximum number of available input ClockTarget Interfaces (fttm-max-num-time-indexes), the maximum number of available DTSF instances (fttm-max-num-dtsfs), the maximum number of available input ClockTarget Interfaces on each DTSF instance (dtsf-max-num-time-indexes), and the ability of the FTTM to use a local oscillator for its time selection operations (fttm-use-osc-clk).

```
        <fttm-system-ds>
          <fttm-max-num-time-indexes>3</fttm-max-num-time-indexes>
          <fttm-max-num-dtsfs>1</fttm-max-num-dtsfs>
          <dtsf-max-num-time-indexes>2</dtsf-max-num-time-indexes>
          <fttm-use-osc-clk>true</fttm-use-osc-clk>
```

J.4 TSF Configuration

The configuration continues with the definition of TSFs within the FTTM, where the TSF instance that serves as the ITSF is identified as tsf-instance-number = 0 and where each TSF instance that serves as a DTSF is identified with a tsf-instance-number that ranges from 1 to fttm-num-active-dtsfs.

The time-index selection algorithm that is used by the two TSFs in this example are read, as shown below.

```
<fttm-tsf-algo-name-list>
  <tsf-instance-number>0</tsf-instance-number>
  <fttm-tsf-algo-name>MVTISA</fttm-tsf-algo-name>
</fttm-tsf-algo-name-list>
<fttm-tsf-algo-name-list>
  <tsf-instance-number>1</tsf-instance-number>
  <fttm-tsf-algo-name>MVTISA</fttm-tsf-algo-name>
</fttm-tsf-algo-name-list>
```

The time difference change threshold, which is used to determine whether a TSF continues to use its previously selected time index or to change to the current time index that best satisfies its selection criteria, is configured to 2 ns for both TSFs of this example, as shown below.

```
<fttm-sel-change-thresh-list>
  <tsf-instance-number>0</tsf-instance-number>
  <extended-timestamp-list>
    <seconds>0</seconds>
    <fractional-nanoseconds>131072</fractional-nanoseconds>
  </extended-timestamp-list>
</fttm-sel-change-thresh-list>
<fttm-sel-change-thresh-list>
  <tsf-instance-number>1</tsf-instance-number>
  <extended-timestamp-list>
    <seconds>0</seconds>
    <fractional-nanoseconds>131072</fractional-nanoseconds>
  </extended-timestamp-list>
</fttm-sel-change-thresh-list>
```

J.5 PTP Instance association to FTTM input indexes

The example then outlines the association of PTP Instances with the FTTM input ClockTarget interfaces using fttm-map-ptp-instance-to-index, emphasizing the unique indexing of each PTP Instance within the full ieee1588-ptp-tt module. The configuration of three PTP Instances, with instance indexes 123, 456, and 789 associated with FTTM input ClockTarget interfaces index numbers 1, 2, and 3, respectively, is illustrated in the code below and in Figure J-1.

```
<fttm-map-ptp-instance-to-index-list>
  <fttm-input-index-number>1</fttm-input-index-number>
  <instance-index>123</instance-index>
</fttm-map-ptp-instance-to-index-list>
<fttm-map-ptp-instance-to-index-list>
  <fttm-input-index-number>2</fttm-input-index-number>
  <instance-index>456</instance-index>
</fttm-map-ptp-instance-to-index-list>
<fttm-map-ptp-instance-to-index-list>
  <fttm-input-index-number>3</fttm-input-index-number>
  <instance-index>789</instance-index>
</fttm-map-ptp-instance-to-index-list>
```

.

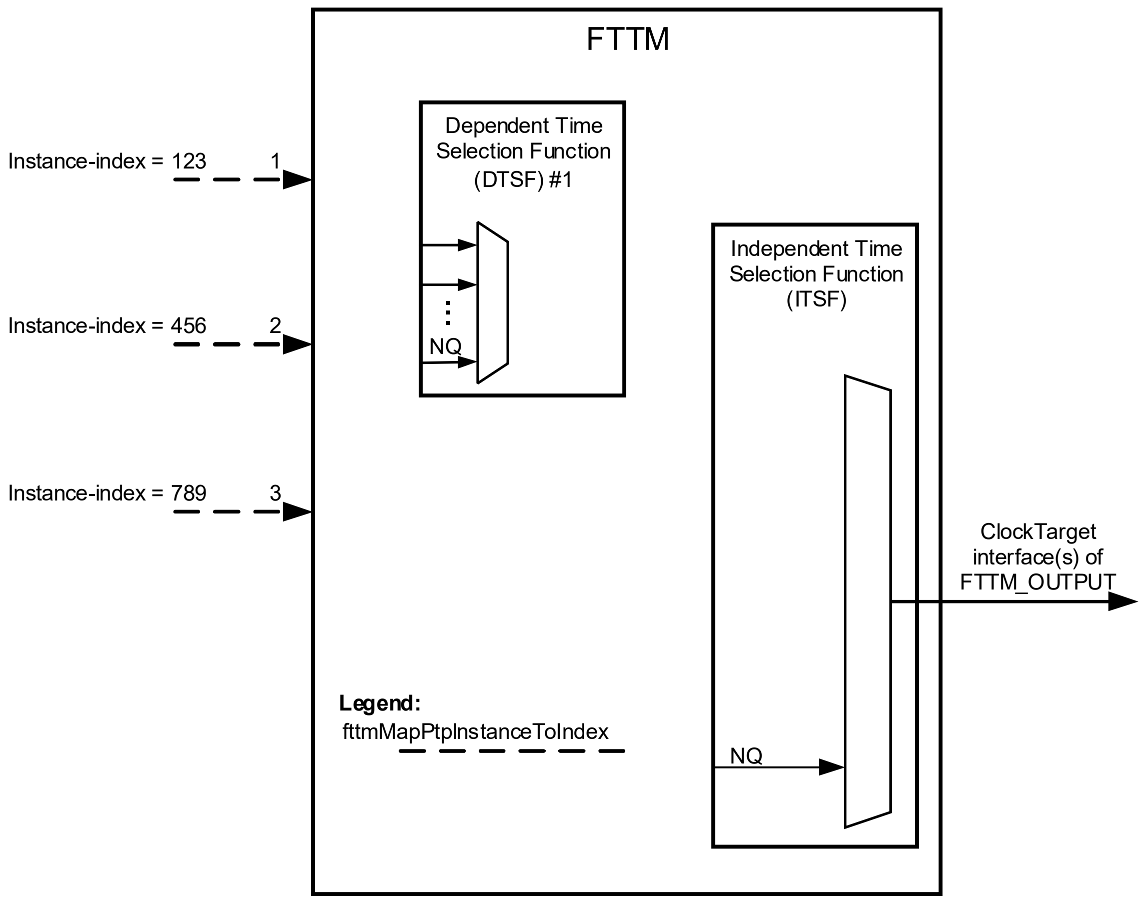


Figure J-1— fttmMapPtpInstanceToIndex example

J.6 FTTM input index connection to ITSF and DTSF input indexes

The example then outlines the connection of FTTM input indexes to TSF input indexes, using fttm-map-index-to-tsf, allowing the FTTM input ClockTarget Interfaces associated with the FTTM input indexes to be processed by an appropriate TSF; a DTSF (as a dependent time) or the ITSF (as an independent time).

The FTTM input index 1 is connected to input index 1 of TSF instance number 1 (a DTSF). The FTTM input index 2 is connected to index 2 of the same TSF. FTTM input index 3 is connected to input index 2 of TSF instance number 0 (the ITSF). This is illustrated in the following code and in Figure J-2.

The above connections insinuate that FTTM input indexes 1 and 2 share a dependence and that DTSF #1 will select between the two of them.

```
<fttm-map-index-to-tsf-list>
  <fttm-input-index-number>1</fttm-input-index-number>
  <tsf-instance-number>1</tsf-instance-number>
  <tsf-input-index-number>1</tsf-input-index-number>
</fttm-map-index-to-tsf-list>
<fttm-map-index-to-tsf-list>
  <fttm-input-index-number>2</fttm-input-index-number>
```

```
        <tsf-instance-number>1</tsf-instance-number>
        <tsf-input-index-number>2</tsf-input-index-number>
      </fttm-map-index-to-tsf-list>
      <fttm-map-index-to-tsf-list>
        <fttm-input-index-number>3</fttm-input-index-number>
        <tsf-instance-number>0</tsf-instance-number>
        <tsf-input-index-number>2</tsf-input-index-number>
      </fttm-map-index-to-tsf-list>
```

.

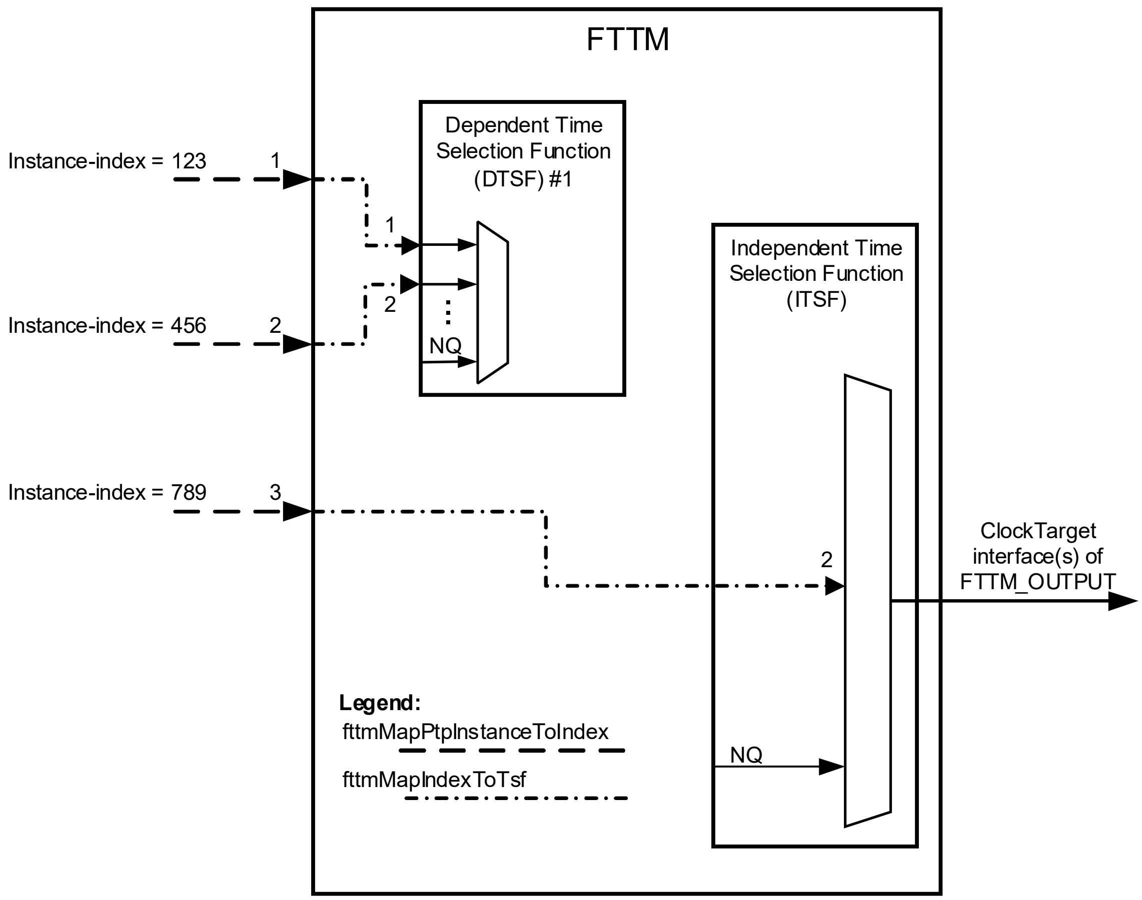


Figure J-2— fttmMapIndexToTsf example

J.7 DTSF outputs to ITSF input indexes

The example outlines the connection of DTSF output interfaces to ITSF input indexes, using fttm-map-dtsf-to-itsf, allowing the interface selected by the DTSF to undergo ITSF selection. Following the example from J.6, the output of DTSF #1 is to be connected to ITSF input index 1. This is illustrated in the code below and in Figure J-3.

With the addition of this connection, each of the 3 input ClockTarget Interfaces of the FTTM have a selectable path, via DTSF and ITSF state machines and selection algorithms, to the FTTM output ClockTarget Interface.

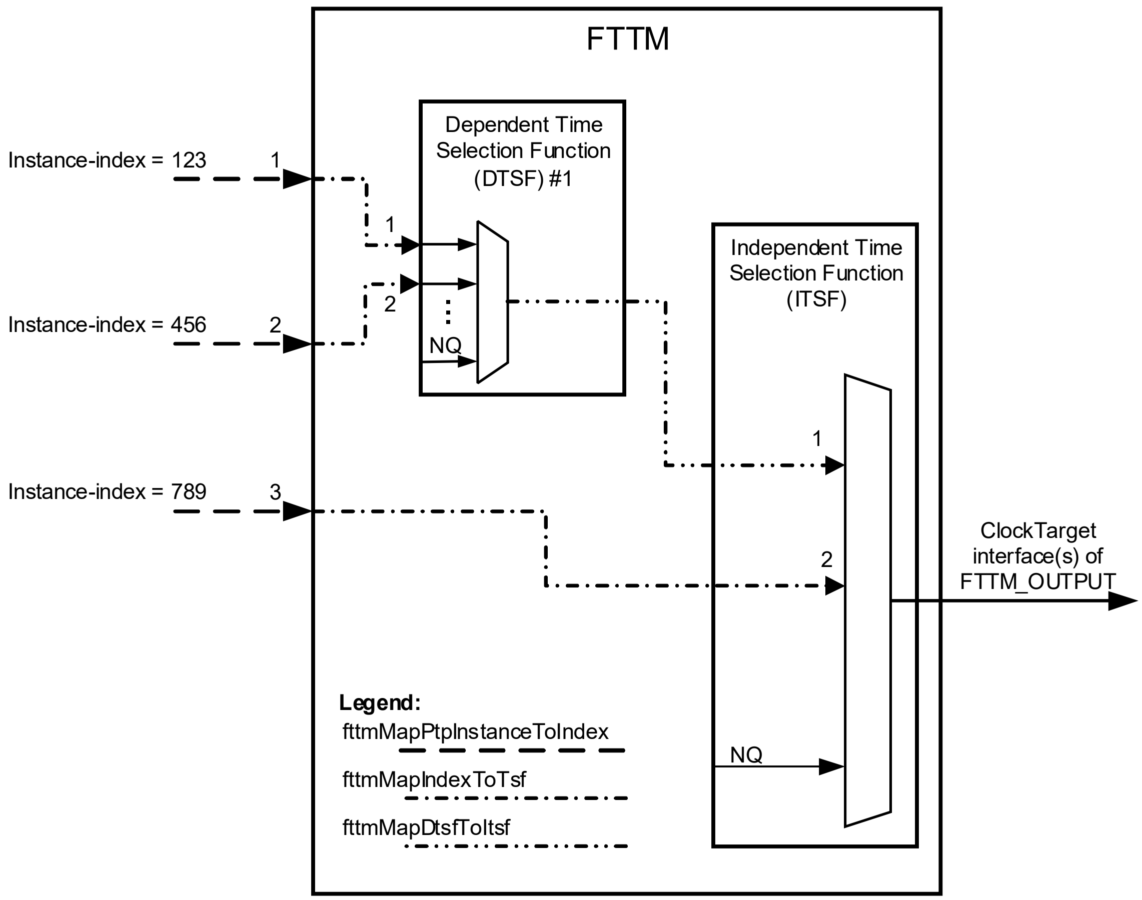


Figure J-3— fttmMapDtsfToItsf example

```
<fttm-map-dtsf-to-itsf-list>
  <dtsf-instance-number>1</dtsf-instance-number>
  <itsf-input-index-number>1</itsf-input-index-number>
</fttm-map-dtsf-to-itsf-list>
```

J.8 Inter-PTP Instance skew

The following XML instance data represents a two dimensional array holding the maximum magnitude of expected skew magnitude between times provided by the FTTM input ClockTarget interfaces of index x and index y when the times given by those interfaces are not faulty.

In this example, we have:

— $maxAs_{12}$= 11111 x 2^{-16} ns, between times provided by the FTTM input ClockTarget interfaces of index x = 1 and index y = 2 when the times given by those interfaces are not faulty.
— $maxAs_{13}$= 22222 x 2^{-16} ns, between times provided by the FTTM input ClockTarget interfaces of index x = 1 and index y = 3 when the times given by those interfaces are not faulty.
— $maxAs_{23}$= 33333 x 2^{-16} ns, between times provided by the FTTM input ClockTarget interfaces of index x = 1 and index y = 3 when the times given by those interfaces are not faulty.

```
<fttm-max-as-lists>
  <fttm-input-index-number>1</fttm-input-index-number>
    <fttm-max-as-list>
      <fttm-input-index-number>2</fttm-input-index-number>
      <fttm-max-as>11111</fttm-max-as>
    </fttm-max-as-list>
    <fttm-max-as-list>
      <fttm-input-index-number>3</fttm-input-index-number>
      <fttm-max-as>22222</fttm-max-as>
    </fttm-max-as-list>
</fttm-max-as-lists
<fttm-max-as-lists>
  <fttm-input-index-number>2</fttm-input-index-number>
  <fttm-max-as-list>
    <fttm-input-index-number>3</fttm-input-index-number>
    <fttm-max-as>33333</fttm-max-as>
  </fttm-max-as-list>
</fttm-max-as-lists>
```

J.9 Hysteresis

The following XML instance data represents a two-dimensional array holding the assigned hysteresis values, used on the skews detected on the FTTM input ClockTarget interfaces of index x and index y.

```
<fttm-hyst-lists>
  <fttm-input-index-number>1</fttm-input-index-number>
  <fttm-hyst-list>
    <fttm-input-index-number>2</fttm-input-index-number>
    <fttm-hyst>9999</fttm-hyst>
  </fttm-hyst-list>
  <fttm-hyst-list>
    <fttm-input-index-number>3</fttm-input-index-number>
    <fttm-hyst>8888</fttm-hyst>
  </fttm-hyst-list>
</fttm-hyst-lists>
<fttm-hyst-lists>
  <fttm-input-index-number>2</fttm-input-index-number>
  <fttm-hyst-list>
    <fttm-input-index-number>3</fttm-input-index-number>
    <fttm-hyst>7777</fttm-hyst>
  </fttm-hyst-list>
</fttm-hyst-lists>
```

In this example, we have:

- — $Hyst_{12}$= 999 x 2^{-16} ns, between times provided by the FTTM input ClockTarget interfaces of index x = 1 and index y = 2 when the times given by those interfaces are not faulty.
- — $Hyst_{13}$= 888 x 2^{-16} ns, between times provided by the FTTM input ClockTarget interfaces of index x = 1 and index y = 3 when the times given by those interfaces are not faulty.
- — $Hyst_{23}$= 777 x 2^{-16} ns, between times provided by the FTTM input ClockTarget interfaces of index x = 1 and index y = 3 when the times given by those interfaces are not faulty.

J.10 FTTM RateRatio offsets

The following XML instance data represents the fttm-sel-time-rate-ratio-offset value and the fttm-sel-time-std-dev-rate-ratio-offset value. The former is equal to the maximum measured rate ratio offset of the normalized clock rate of the time arriving on the input interfaces to the normalized clock rate of the FTTM's local clock, OSC_CLK, that allows the FTTM to remain in the FREQ_TRUSTED state when the TIME_TRUSTED state cannot be satisfied. The latter is equal to the maximum standard deviation of the measured rate ratio offset of the normalized clock rate of the time arriving on the input interfaces to the normalized clock rate of the FTTM's local clock, OSC_CLK, that allows the FTTM to remain in the FREQ_TRUSTED state when the TIME_TRUSTED state cannot be satisfied.

In this example, the fttm-sel-time-rate-ratio-offset value is 0.0001 (i.e., 100ppm) and the fttm-sel-time-std-dev-rate-ratio-offset value is 0.00002. Per 14.23.18 and 14.23.19, both values are multiplied by 2^{41} before into the managed object.

This example assumes that the fttm-use-osc-vlk object is TRUE, enabling OSC_CLK to be used by the FTTM.


```
<fttm-sel-time-rate-ratio-offset>219902326<fttm-sel-time-rate-ratio-offset>
<fttm-sel-time-std-dev-rate-ratio-offset>43980465<fttm-sel-time-std-dev-rate-ratio-offset>
```


J.11 FTTM status and statistics

The following XML instance data represents a reading of the status and statistics objects from the FTTM.

The status is given for the scenario in which the PTP Instance with instance index 456 is selected by the FTTM as the source for its trusted time. The selected time-index path is shown in Figure J-4.

The ToD returned by the three PTP Instances (i.e., the timeReceiverTimeCallback result) at the last ClockTargetEventCapure.invoke event sent to the PTP Instances is read from fttm-collected-tod. The validity of fttm-collected-tod is first checked by confirming fttm-invoke-status-avail is TRUE. After the ToD values are read for all PTP Instances, the state of fttm-invoke-status-avail is cleared to FALSE by setting fttm-invoke-status-avail-clr to TRUE and then back to FALSE.

The statistic on fttm-sel-time-index-change-cnt shows that the FTTM's counter value for the number of times it has changed its selected time-index is 2. The number of times the selected time-index has changed since the last time this counter value was read is given by the difference between the current value of the counter and the previous value of the counter (with a counter wrap-around to 0 after 65535).


```
          <fttm-num-active-dtsfs>1</fttm-num-active-dtsfs>
          <fttm-num-active-time-indexes>3</fttm-num-active-time-indexes>
          <fttm-tsf-sel-time-index-list>
            <tsf-instance-number>0</tsf-instance-number>
            <fttm-tsf-sel-time-index>1</fttm-tsf-sel-time-index>
          </fttm-tsf-sel-time-index-list>
          <fttm-tsf-sel-time-index-list>
            <tsf-instance-number>1</tsf-instance-number>
            <fttm-tsf-sel-time-index>2</fttm-tsf-sel-time-index>
          </fttm-tsf-sel-time-index-list>
          <fttm-sel-instance-index>456</fttm-sel-instance-index>
          <fttm-trust-state>TIME-TRUSTED</fttm-trust-state>

          <fttm-invoke-status-avail>TRUE</fttm-invoke-status-avail>
          <fttm-collected-tod-list>
            <fttm-input-index-number>1</fttm-input-index-number>
            <extended-timestamp-list>
              <seconds>0</seconds>
              <fractional-nanoseconds>1000</fractional-nanoseconds>
            </extended-timestamp-list>
          </fttm-collected-tod-list>
          <fttm-collected-tod-list>
            <fttm-input-index-number>2</fttm-input-index-number>
```

```
          <extended-timestamp-list>
            <seconds>0</seconds>
            <fractional-nanoseconds>1006</fractional-nanoseconds>
          </extended-timestamp-list>
        </fttm-collected-tod-list>
        <fttm-collected-tod-list>
          <fttm-input-index-number>3</fttm-input-index-number>
          <extended-timestamp-list>
            <seconds>0</seconds>
            <fractional-nanoseconds>1008</fractional-nanoseconds>
          </extended-timestamp-list>
        </fttm-collected-tod-list>
        <fttm-invoke-status-avail-clr>TRUE</fttm-invoke-status-avail-clr>
        <fttm-invoke-status-avail-clr>FALSE</fttm-invoke-status-avail-clr>

        <fttm-sel-time-index-change-cnt>2</fttm-sel-time-index-change-cnt>
```

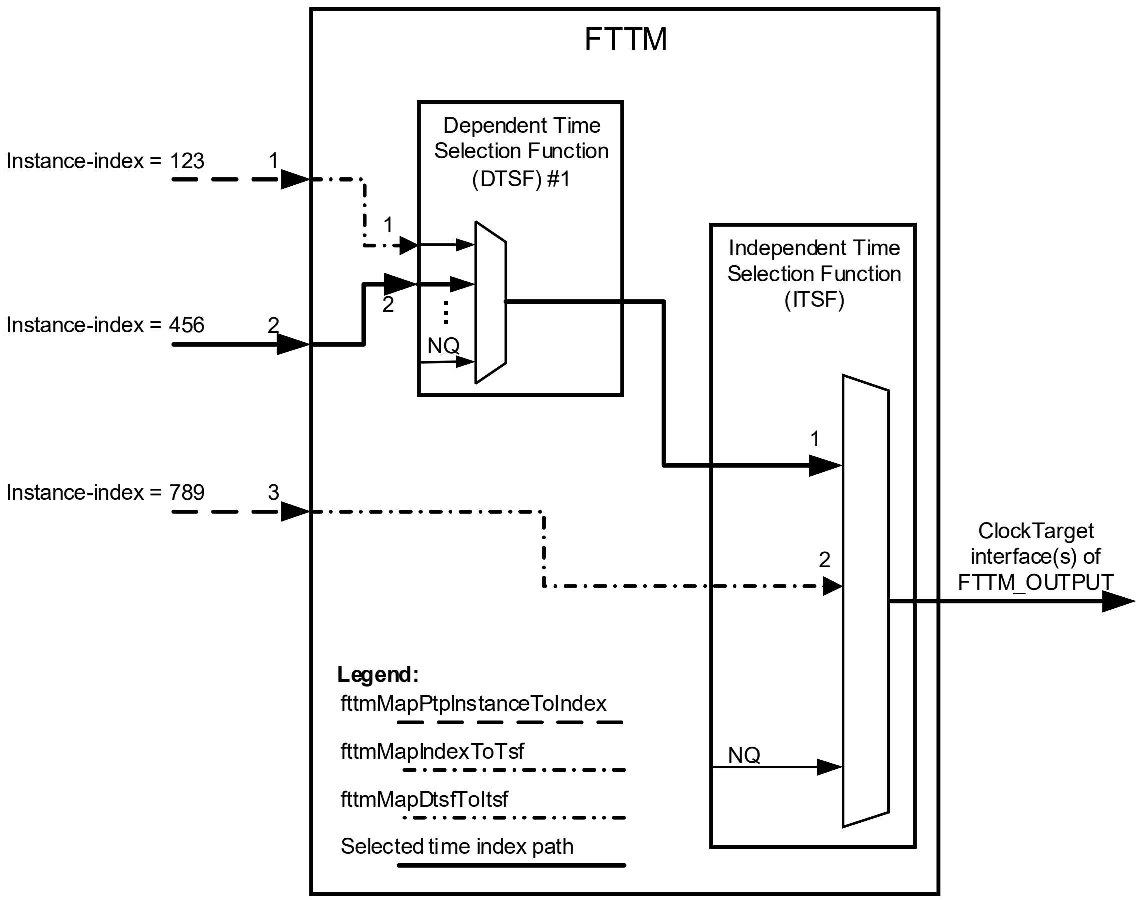


Figure J-4— fttmMapDtsfToItsf example

J.12 Conclusion

This example concludes with the proper closing tags, encapsulating the detailed configuration of an FTTM system within the common services of the ptp element. This walkthrough serves as a practical guide to understanding the XML configuration necessary for implementing the FTTM service.

```
        </fttm-system-ds>
        <fttm-system-description-ds>
          <user-description>This is a test instance.</user-description>
        </fttm-system-ds>
      </fttm-system>
    </fault-tolerant-timing-module-service>
  </common-services>
</ptp>
```

Insert the following annex.

Annex K (informative)

Fault-tolerance claims for the FTTM

The fault-tolerance claims that can be made for the FTTM are given in this annex. The bases for these claims are described in 20.2.2.

The FTTM can establish time integrity with NITS independent times and a maximum of NIFS independent input times with an independent fault when:

- — NITS = 2 and NIFS = 0
- — NITS (2 × NIFS + 1) if NITS > 2 and is odd
- — NITS (2 × NIFS) if NITS > 2 and is even, assuming the independent faults do not produce the same error
- — NITS (2 × NIFS + 2) if NITS > 2 and is even, assuming the independent faults can produce the same error

Table K-1 shows the integrity establishment ability of the FTTM for various combinations of NITS and NIFS, with the assumption that independent faults do not produce the same error.

Table K-1—FTTM Integrity establishment ability with independent faults that do not produce the same error

FTTM Integrity Establishment Ability							
		Number of Independent Times with an Indpendent Fault (NIFS)					
		0	1	2	3	4	5
Number of Independent Times (NITS)	1	✗	✗	✗	✗	✗	✗
	2	✓	✗	✗	✗	✗	✗
	3	✓	✓	✗	✗	✗	✗
	4	✓	✓	✓	✗	✗	✗
	5	✓	✓	✓	✗	✗	✗

Table K-2 shows the integrity establishment ability of the FTTM for various combinations of NITS and NIFS, with the assumption that independent faults can produce the same error.

Table K-2—FTTM Integrity establishment ability with independent faults that canproduce the same error

FTTM Integrity Establishment Ability							
		Number of Independent Times with an Indpendent Fault (NIFS)					
		0	1	2	3	4	5
Number of Independent Times (NITS)	1	✗	✗	✗	✗	✗	✗
	2	✓	✗	✗	✗	✗	✗
	3	✓	✓	✗	✗	✗	✗
	4	✓	✓	✗	✗	✗	✗
	5	✓	✓	✓	✗	✗	✗

Insert the following annex.

Annex L (informative)

Time error accumulation examples

L.1 General

As gPTP time is distributed through a network from a GM to PTP Relay Instances and/or PTP Instances, time error accumulates. Some of the reasons for this time error accumulation are listed below:

— Timing errors at the GM (TEgm)
— Timing errors at intermediate PTP Relay Instances (TErly)
— Timing errors at the PTP End Instance (TEend)
— Link asymmetry between PTP Instances (TElnk)

The above time errors are illustrated inFigure L-1.

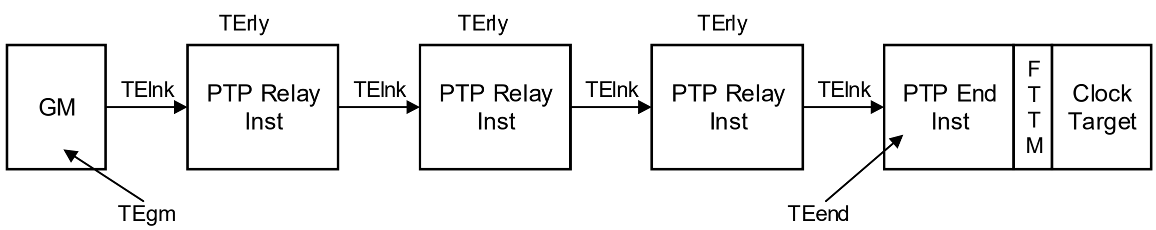


Figure L-1—Time error accumulation across a network

It is possible to determine the potential maximum absolute value of each of the above time errors and, thus, the maximum potential time error at the PTP End Instance. This result, maxAccumTE (see L.2), if available, can be used by the FTTM in its selection process for a trusted gPTP time to present to the ClockTarget.

NOTE—The potential maximum absolute values of TEgm, TErly, TEend, and TElnk are expected to be calculated or measured prior to operational usage. This could be done at the design, characterization, or certification phase. The specific methods and procedures to do this are not defined in this standard.

The ENHANCED_ACCURACY_METRICS TLV from IEEE Std 1588a-2023 [B6] can be used to accumulate the maximum constant and dynamic time errors of each PTP Instance and the connecting links, on a hop-by-hop basis, in the path from the GM to, but not including, the final PTP End Instance. This TLV is carried in Announce messages.

Time skew between two time distribution paths is shown in Figure M.2 . The time error contributors across each path are the same as shown in 0, but the contributors on each path are marked with the path's identifying suffix, x or y.

The various time skew results are described in L.2, L.3, L.4, and L.5.

L.2 maxAccumTE$_x$

For the list of reasons for time error accumulation given in L.1, the parameter maxAccumTEx is the maximum non-faulty accumulated time error magnitude for the time distributed on path x, from its GM

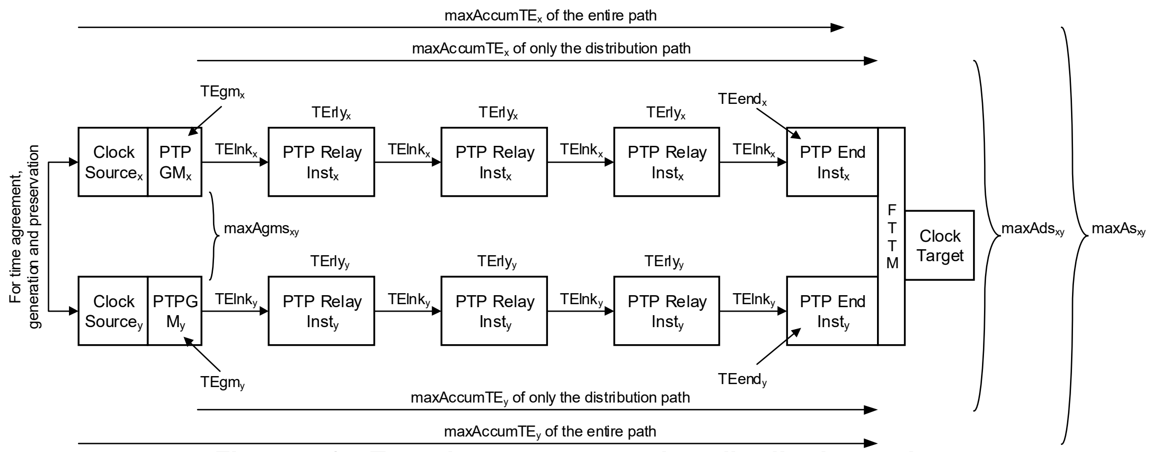


Figure L-2—Tme skew across two time distribution paths

(TEgm), through all intermediate PTP Relay Instances (TErly) and the corresponding links (TElnk), to the PTP End Instance (TEend) that is connected to the FTTM..

$$maxAccumTE_x = max(|TEgm_x|) + \sum max(|TErly_x|) + \sum max(|TE\ln k_x|) + max(|TEend_x|)$$

Where Σ is the summation symbol and represents a summation of all instances of the term adjacent to it

L.3 maxAgms$_{xy}$

The parameter maxAgms$_{xy}$ is the maximum accepted time skew magnitude between two non-faulty PTP GMs, GM$_x$ and GM$_y$. This value is equal to the worst-case time error magnitude between the two GMs when they are not faulty.

$$maxAgms_{xy} = max(|TEgm_x|) + max(|TEgm_y|)$$

L.4 maxAds$_{xy}$

The parameter maxAds$_{xy}$ is the maximum accepted distribution skew magnitude between the time of two non-faulty times, distributed on path x and path y. This value is equal to the worst-case time error magnitude between the two times, from the perspective of the FTTM, resulting from their distribution paths when they are not faulty.

$$\begin{aligned} maxAds_{xy} = {} & \sum max(|TErly_x|) + \sum max(|TE\ln k_x|) + max(|TEend_x|) + \\ & \sum max(|TErly_y|) + \sum max(|TE\ln k_y|) + max(|TEend_y|) \end{aligned}$$

L.5 maxAs$_{xy}$

The parameter maxAs$_{xy}$ is the maximum accepted skew magnitude between two non-faulty times, distributed on path x and path y. This value is equal to the worst-case time error magnitude between two synchronized times, from the perspective of the FTTM, when they are not faulty. This value can be used as a criterion to determine the trustworthiness of the times being compared.

$$\begin{aligned} maxAs_{xy} &= maxAgms_{xy} + maxAds_{xy} \\ &= maxAccumTE_x + maxAccumTE_y \end{aligned}$$

System integrators have to design their TSN network, which supports fault-tolerant timing and time integrity, such that synchronization-dependent nodes can withstand a drift equal to the magnitude of this maximum accepted skew.

Insert the following annex.

Annex M (informative)

Bridging alternate ClockTarget interface types to the FTTM

This annex discusses concepts for interworking alternate ClockTarget interfaces (e.g., ClockTargetTriggerGenerate and ClockTargetClockGenerator interfaces from 9.4 and 9.5, respectively) from a PTP Instance to a ClockTargetEventCapture interface (see 9.3) of the FTTM and from the ClockTargetEventCapture interface of the FTTM to a ClockTarget entity.

Figure M-1 shows an interworking function between PTP Instances, the FTTM, and the ClockTarget entity, where the PTP Instances and the ClockTarget entity do not use the ClockTargetEventCapture interface. The interworking function has multiple proxy timers, each of which holds the PTP time that is recovered from a corresponding PTP Instance using the information provided by its alternate ClockTarget interface. These proxy timers are used as sources of time for the ClockTargetEventCapture interfaces to the FTTM. The FTTM's fttmTrustState (see 14.23.20) and fttmSelInstanceIndex (see 14.23.16) management objects are then used to select which proxy timer's time, or the NQ time (see 20.3.2.3) if trust is not found, is sent to a functional block that generates the alternate ClockTarget interface to the ClockTarget entity.

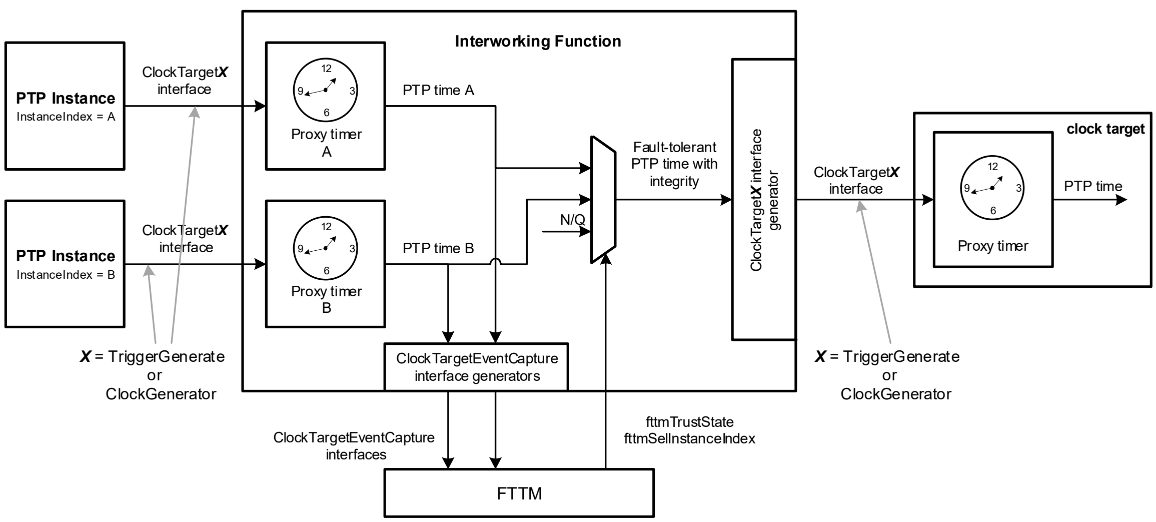


Figure M-1— Conceptual interworking function to alternate ClockTarget interfaces for the FTTM