Draft for Local and Metropolitan Area Networks-Timing and Synchronization for Time-Sensitive Applications

Amendment: Fault-Tolerant Timing with Time Integrity

Prepared by: Richard Tse, Microchip Technology

Abstract:

Keywords:

Introduction

This introduction is not part of P<designation>/D0, Draft <Gde./Rec. Prac./Std.> for Fault-Tolerant Timing with Time Integrity.

Contents

1. Overview 1
2. Normative references 1
3. Definitions 1
4. Abbreviations and acronyms 1
5. Conformance 1
6. Conventions 2
7. Time-synchronization model for a packet network 2
8. IEEE 802.1AS concepts and terminology 2
9. Application Interfaces 2
10. Media-independent layer specification 2
11. Media-dependent layer specification for full-duplex point-to-point links 2
12. Media-dependent layer specification for IEEE 802.11 links 2
13. Media-dependent layer specification for interface to IEEE 802.3 Ethernet passive optical network link 2
14. Timing and synchronization management 3
14.1 General 3
14.2 Default Parameter Data Set (defaultDS) 3
14.3 Current Parameter Data Set (currentDS) 3
14.4 Parent Parameter Data Set (parentDS) 3
14.5 Time Properties Parameter Data Set (timePropertiesDS) 3
14.6 Path Trace Parameter Data Set (pathTraceDS) 3
14.7 Acceptable TimeTransmitter Table Parameter Data Set (acceptableTimeTransmitterTableDS) 3
14.8 PTP Instance Synchronization Parameter Data Set (ptpInstanceSyncDS) 3
14.9 Drift Tracking Parameter Data Set (driftTrackingDS) 3
14.10 Port Parameter Data Set (portDS) 3
14.11 Description Port Parameter Data Set (descriptionPortDS 3
14.12 Port Parameter Statistics Data Set (portStatisticsDS) 3
14.13 Acceptable TimeTransmitter Port Parameter Data Set (acceptableTimeTransmitterPortDS) 3
14.14 External Port Configuration Port Parameter Data Set (externalPortConfigurationPortDS) 4
14.15 Asymmetry Measurement Mode Parameter Data Set (asymmetryMeasurementModeDS) 4
14.16 Common Services Port Parameter Data Set (commonServicesPortDS) 4
14.17 Common Mean Link Delay Service Default Parameter Data Set (cmldsDefaultDS) 4
14.18 Common Mean Link Delay Service Link Port Parameter Data Set (cmldsLinkPortDS) 4
14.19 Common Mean Link Delay Service Link Port Parameter Statistics Data Set (cmldsLinkPortStatisticsDS) 4
14.20 Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set (cmldsAsymmetryMeasurementModeDS) 4
14.21 Hot Standby System Parameter Data Set (hotStandbySystemDS) 4
14.22 Hot Standby System Description Parameter Data Set (hotStandbySystemDescriptionDS) 4

14.23 Fault Tolerant Timing Module System Parameter Data Set (fttmSystemDS) 4

15. Managed object definitions 5

16. Media-dependent layer specification for CSN... 5

17. YANG Data Model... 5

18. Hot Standby.. 5

19. Fault-tolerant timing with time integrity 6
19.1 General 6
19.2 Fault-tolerant time synchronization concepts 6
19.3 Fault-Tolerant Timing Module 10

Annex A (normative) Protocol Implementation Conformance Statement (PICS) proforma 26

Annex B (informative) Performance Requirements 27

Annex C (Informative) Timescales and epochs..28

Annex D Reserved for future use 29

Annex E Reserved for future use..30

Annex F (informative) PTP profile included in this standard..31

Annex G (informative) The asymmetry compensation measurement procedure based on line-swapping....32

Annex H (informative) Bibliography 33

Annex I (informative) Example Aerospace Configuration...34

Annex J (informative) Time synchronization with Fault Tolerance and Integrity 35
J.1 Introduction..35
J.2 Clock domain management...35
J.3 Time agreement generation examples...35
J.4 Balancing availability and integrity 36
J.5 FTTM operation in example network topologies...36

Amendment: Fault-Tolerant Timing with Time Integrity

1. Overview

2. Normative references

Insert the following normative reference:

IEEE Std 1588a™-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.

3. Definitions

4. Abbreviations and acronyms

Insert the following acronyms into the existing list of acronyms

DTSA	Dependent Time Selection Algorithm
FTTM	Fault-Tolerant Timing Module
ITSA	Independent Time Selection Algorithm
MVTISP	mid-value time-index selection process

5. Conformance

6. Conventions

7. Time-synchronization model for a packet network

8. IEEE 802.1AS concepts and terminology

9. Application Interfaces

10. Media-independent layer specification

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

12. Media-dependent layer specification for IEEE 802.11 links

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

14. Timing and synchronization management

14.1 General

14.1.1 Data set hierarchy

14.1.2 Data set descriptions

14.2 Default Parameter Data Set (defaultDS)

14.3 Current Parameter Data Set (currentDS)

14.4 Parent Parameter Data Set (parentDS)

14.5 Time Properties Parameter Data Set (timePropertiesDS)

14.6 Path Trace Parameter Data Set (pathTraceDS)

14.7 Acceptable TimeTransmitter Table Parameter Data Set (acceptableTimeTransmitterTableDS)

14.8 PTP Instance Synchronization Parameter Data Set (ptpInstanceSyncDS)

14.9 Drift Tracking Parameter Data Set (driftTrackingDS)

14.10 Port Parameter Data Set (portDS)

14.11 Description Port Parameter Data Set (descriptionPortDS

14.12 Port Parameter Statistics Data Set (portStatisticsDS)

14.13 Acceptable TimeTransmitter Port Parameter Data Set (acceptableTimeTransmitterPortDS)

14.14 External Port Configuration Port Parameter Data Set (externalPortConfigurationPortDS)

14.15 Asymmetry Measurement Mode Parameter Data Set (asymmetryMeasurementModeDS)

14.16 Common Services Port Parameter Data Set (commonServicesPortDS)

14.17 Common Mean Link Delay Service Default Parameter Data Set (cmldsDefaultDS)

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

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

14.20 Common Mean Link Delay Service Asymmetry Measurement Mode Parameter Data Set (cmldsAsymmetryMeasurementModeDS)

14.21 Hot Standby System Parameter Data Set (hotStandbySystemDS)

14.22 Hot Standby System Description Parameter Data Set (hotStandbySystemDescriptionDS)

Insert the following subclause

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

14.23.1 General

< The list below consists of initial ideas of potential FTTM managed objects. More thought and discussion is needed.>

14.23.2 fttmNumTimeIndexes (Uinteger8)

The parameter fttmNumTimeIndexes is equal to the number of input ClockTarget Interfaces connected to the FTTM.

14.23.3 fttmNumDTSAs (UInteger8)

The parameter fttmNumDTSAs is equal to the number of DTSA instances in the FTTM.

14.23.4 fttmNumDTSATimeIndexes[n] (Vector of UInteger8)

The parameter fttmNumDTSATimeIndexes[n] is the number of input ClockTarget Interfaces connected to DTSA Instance x.

14.23.5 fttmDTSATimeIndexMap[fttmNumDTSAs][fttmNumDTSATimeIndexes]

The parameter fttmDTSATimeIndexMap[x][y] maps the FTTM's input ClockTarget interfaces to DTSA instances' input ClockTarget interfaces, where appropriate.

14.23.6 fttmNumITSATimeIndexes (Vector of UInteger8)

The parameter fttmNumITSATimeIndexes is the number of input ClockTarget Interfaces on the ITSA Instance.

14.23.7 fttmITSAMap[fttmNumITSATimeIndexes]

The parameter fttmItsaMap maps either a FTTM's input ClockTarget interface or a DTSA output ClockTarget interface is each of the ITSA's input ClockTarget interfaces.

15. Managed object definitions

16. Media-dependent layer specification for CSN

17. YANG Data Model

18. Hot Standby

19. Fault-tolerant timing with time integrity

19.1 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 provide 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 gPTP instances per port in Bridges and end stations, and the external port configuration mode (see 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 multiple (typically 2) simultaneous 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 fault tolerance supporting gPTP features listed above. The concepts used by the FTTM are described in 19.2. Details on the FTTM and its default operations are given in 19.3. General information about fault-tolerant timing with time integrity can be found in Annex J.

19.2 Fault-tolerant time synchronization concepts

19.2.1 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 gPTP instances in Bridges and end stations.

19.2.2 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 19.2.7.4 for an example of such a criterion.

19.2.3 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 is outside the scope of this standard.

19.2.4 Time agreement distribution

Time agreement distribution is the process of distributing the time established by time agreement generation from time source nodes (GMs) to time destination nodes (gPTP End Instances). Time agreement distribution is performed using gPTP, per the models described in clause 7 and the mechanisms specified in clauses 8 to 16 of this standard.

19.2.5 Dependent gPTP times

Dependent gPTP times share one or more common time source components. This could be a common GM, continuously synchronized GMs, GMs that share a common (continuously connected) ClockSource, or a common gPTP 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, which are independent (see 19.2.6), 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 19.3).

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

— They have the same gPTP domainNumber, majorSdoID, and minorSdoID. This indicates that gPTP messages from the same gPTP GM are received by two gPTP 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 gPTP 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 19.3).

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

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.

19.2.6 Independent gPTP times

Independent gPTP times do not share any common time source 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 gPTP domains need to be aligned to each other in a manner that is resilient to faults (i.e., achieve time agreement and preservation, see 19.2.3), including Byzantine faults. For example, time agreement mechanisms can be used to align the clocks of two independent GMs.

Because the independent gPTP 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.

19.2.7 Time error accumulation

As gPTP time is distributed through a network from a GM to a gPTP End Instance, time error accumulates due to the following reasons:

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

The above time errors are illustrated in Figure 1.

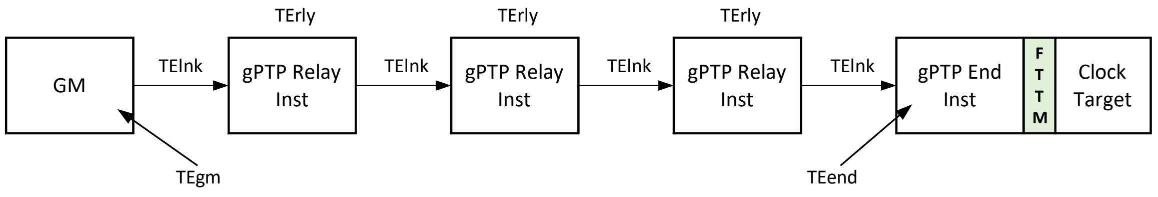

Figure 1—Time error accumulation across a network

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 gPTP End Instance. This result, maxAccumTE (see 19.2.7.1), if available, can be used by the FTTM to select the best 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 can be used to accumulate the maximum constant and dynamic time errors of each gPTP instance and the connecting links, on a hop-by-hop basis, in the path from the GM to, but not including, the final gPTP End Instance. This TLV is carried in gPTP Announce messages.

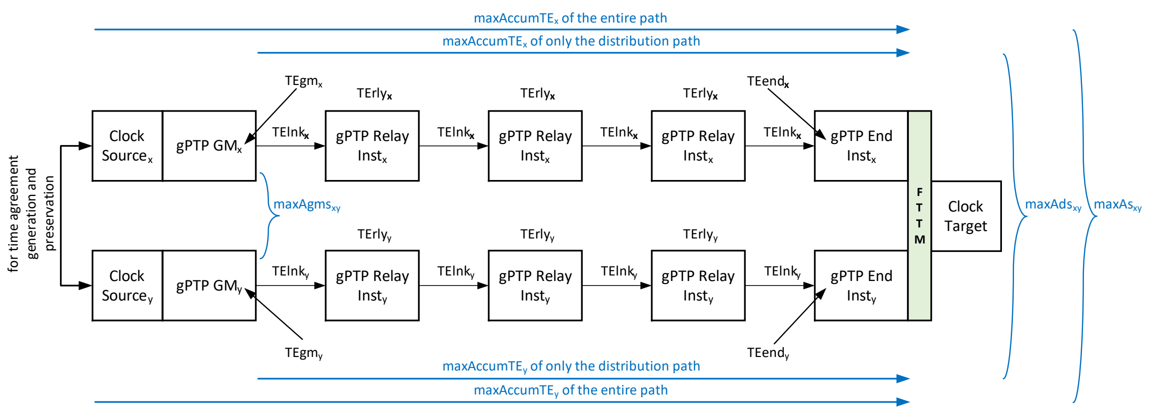
maxAccumTEx of the entire path
maxAccumTEx of only the distribution path
for time agreement generation and preservation
Clock Sourcex
gPTP GMx
TEgmx
TElnkx
gPTP Relay Instx
TErlyx
TEendx
gPTP End Instx
FTTM
Clock Target
maxAgmsxy
maxAdsxy
maxAsxy
Clock Sourcey
gPTP GMy
TEgmy
TElnky
gPTP Relay Insty
TErlyy
TEendy
gPTP End Insty
maxAccumTEy of only the distribution path
maxAccumTEy of the entire path

Figure 2—Time skew across two time distribution paths

Time skew between two time distribution paths is shown in Figure 2. The time error contributors across each path are the same as shown in Figure 1, 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 19.2.7.1, 19.2.7.2, 19.2.7.3, and 19.2.7.4.

19.2.7.1 maxAccumTE$_x$

The parameter maxAccumTE$_x$ is the maximum non-faulty accumulated time error magnitude for the time distributed on path x, from its GM (TEgm), through all intermediate gPTP Relay Instances (TErly) and the corresponding links (TElnk), to the gPTP End Instance (TEend) that is connected to the FTTM.

$$\text{maxAccumTE}_x = \max(|\text{TEgm}_x|) + \sum\max(|\text{TErly}_x|) + \sum\max(|\text{TElnk}_x|) + \max(|\text{TEend}_x|)$$

where ∑ is the summation symbol and represents a summation of all instances of the term adjacent to it

19.2.7.2 maxAgms$_{xy}$

The parameter maxAgms$_{xy}$ is the maximum accepted time skew magnitude between two non-faulty gPTP 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.

$$\text{maxAgms}_{xy} = \max(|\text{TEgm}_x|) + \max(|\text{TEgm}_y|)$$

19.2.7.3 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.

$$\begin{aligned}\text{maxAds}_{xy} = {} & \sum\max(|\text{TErly}_x|) + \sum\max(|\text{TElnk}_x|) + \max(|\text{TEend}_x|) + \\ & \sum\max(|\text{TErly}_y|) + \sum\max(|\text{TElnk}_y|) + \max(|\text{TEend}_y|)\end{aligned}$$

19.2.7.4 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 should 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.

19.3 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, is to be implemented in all time-aware bridges and end stations that receive multiple gPTP times. The FTTM manages the selection of a time source from amongst two or more gPTP times (and gPTP instances) to support increased availability (see 19.2.1) and integrity (see 19.2.2). 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.

Figure 3 illustrates the FTTM operating with three gPTP instances. The FTTM can also use the local oscillator’s clock (OSC_CLK) as an input to its selection algorithm.

Because the FTTM resides between gPTP Instances and a ClockTarget, its effect is localized. This allows different algorithms, each of which specifically serves the timing requirements of the corresponding ClockTarget, to be used by each FTTM. The default algorithms for the FTTM are defined in 19.3.3.

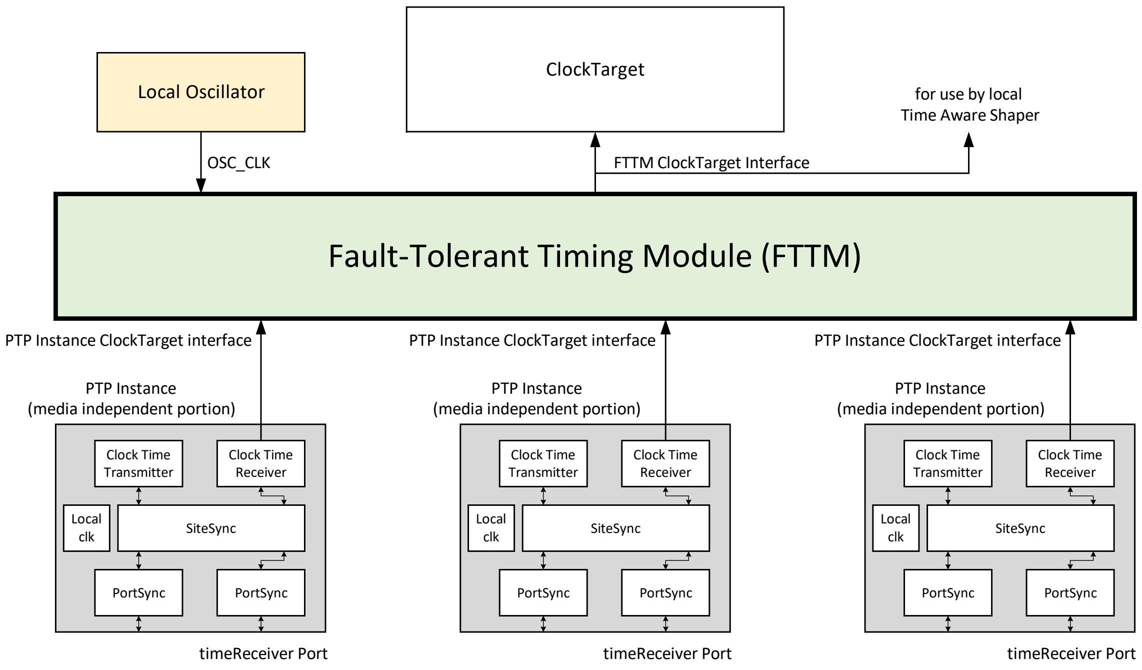
Local Oscillator
ClockTarget
for use by local Time Aware Shaper
OSC_CLK
FTTM ClockTarget Interface
Fault-Tolerant Timing Module (FTTM)
PTP Instance ClockTarget interface
PTP Instance (media independent portion)
Clock Time Transmitter
Clock Time Receiver
Local clk
SiteSync
PortSync
timeReceiver Port

Figure 3— Fault-Tolerant Timing Module in operation

19.3.1 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 one time and one domain is supported by the FTTM, more than one time and 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 19.2.5 and 19.2.6, respectively.

19.3.2 Functional description

The FTTM shall consist of the following functions:

- A local oscillator clock (OSC_CLK).
- ClockTarget application interfaces (see cluase 9) providing time information to the FTTM, where gPTP End Instances serve as the ClockTimeReceiver entities and the FTTM serves as the

ClockTarget entity. Time is passed to the FTTM via each ClockTarget application interface's timeReceiverTimeCallback parameter.

- Zero or more instances of a Dependent Time Selection Algorithm (DTSA).
- Zero or more instances of ClockTarget application interface(s) (see clause 9) providing time information from DTSA(s) to the ITSA, where each DTSA serves as a ClockTimeReceiver entity and the ITSA serves as a ClockTarget entity. Time is passed to the ITSA via each ClockTarget application interface's timeReceiverTimeCallback parameter.
- One instance of an Independent Time Selection Algorithm (ITSA).
- ClockTarget application interface(s) (see clause 9) 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.

A functional block diagram of the FTTM is shown in Figure 4.

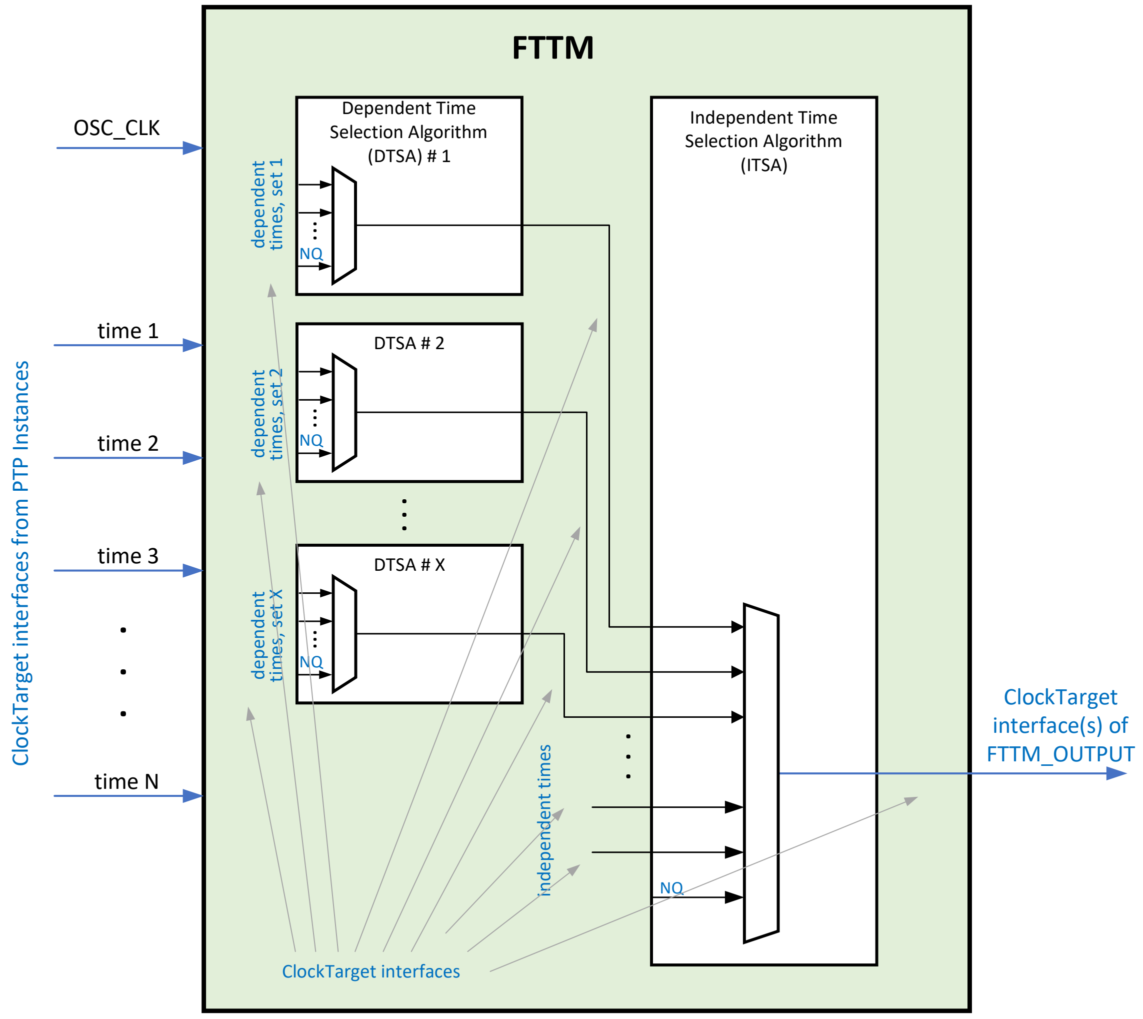
FTTM
OSC_CLK
Dependent Time Selection Algorithm (DTSA) # 1
Independent Time Selection Algorithm (ITSA)
dependent times, set 1
NQ
time 1
DTSA # 2
dependent times, set 2
time 2
ClockTarget interfaces from PTP Instances
time 3
DTSA # X
dependent times, set X
ClockTarget interface(s) of FTTM_OUTPUT
time N
independent times
NQ
ClockTarget interfaces

Figure 4—FTTM functional block diagram

19.3.2.1 Input ClockTarget interfaces

The ClockTarget interfaces that pass time information from the gPTP End Instances to the FTTM are designated as, from the FTTM's perspective, input ClockTarget interfaces. They may be any of the following types defined in clause 9, ClockTargetEventCapture, ClockTargetTriggerGenerate, and ClockTargetClockGenerator, or they may be of another type. However, they should all be of the same type. The ClockTargetPhaseDiscontinuity interface should also be provided by the gPTP End Instances to the FTTM.

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.

19.3.2.2 DTSA and ITSA

The DTSA and the ITSA each analyze their own set of input ClockTarget interfaces and, based on a specified criteria (e.g., see 19.2.7.4, 19.2.5, 19.2.6 for examples), determine which corresponding time(s) can be trusted (see 19.2.2). Being trusted does not mean the time is definitely 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 the DTSA or ITSA.

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 shall be declared to be untrusted. Other conditions for determining whether a time is trusted are determined by the specified criterion.

19.3.2.2.1 DTSA

Each set of input ClockTarget interfaces that share a common dependency shall be processed as a group by one instance of the DTSA. This grouping allows each DTSA instance to produce an output that is independent from the outputs of the other DTSAs and from the other ClockTarget interfaces that are connected as inputs to the ITSA.

Each instance of the DTSA shall select one of its input times, or the Not Qualified (NQ) time (see 19.3.2.2.3) if none of its input times can be determined to be trusted, as its result. The selected time's ClockTarget interface is passed to the output of the DTSA. This output is passed to the ITSA as an independent time.

The default DTSA is described in 19.3.3.3. Other algorithms for the DTSA may also be used by the FTTM. Other algorithms for the DTSA are not required to produce the same result as the default DTSA to achieve time convergence.

19.3.2.2.2 ITSA

When operating in the enhanced availability and integrity scenario (see 19.3), the set of input ClockTarget interfaces that share no dependency with each other, which include the output ClockTarget interfaces of all instances of the DTSA, shall be processed by the ITSA. The ITSA 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 19.3), it is expected that all the input times to the FTTM are connected to the ITSA regardless of their independence from each other (i.e., DTSAs are not used in this scenario). The ITSA 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 19.3), the ITSA receives the sole input time to the FTTM. In this special scenario, the ITSA shall always select the sole input time as its result.

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

The selected time's ClockTarget interface shall be passed to the output of the ITSA and becomes the output of the FTTM, FTTM_OUTPUT. This output is passed to the application ClockTarget entity.

The default ITSA is described in 19.3.3.4. Other algorithms for the ITSA may also be used by the FTTM. Other algorithms for the ITSA are not required to produce the same result as the default ITSA to achieve time convergence.

19.3.2.2.3 Not Qualified (NQ) time

The Not Qualified (NQ) time is used to represent the condition where none of the input times to the DTSA or to the ITSA 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 algorithm but shall have the isSynced status (per P802.3ASdm) 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

19.3.2.3 Output ClockTarget Interfaces

The ClockTarget interfaces that pass time information from the FTTM to the application's ClockTarget entity are designated as, from the FTTM's perspective, output ClockTarget interfaces. They may be any of the following types defined in clause 9, ClockTargetEventCapture, ClockTargetTriggerGenerate, and ClockTargetClockGenerator, or they may be of another type. However, they should be consistent with the types on the FTTM's input ClockTarget interfaces. all be of the same type. The ClockTargetPhaseDiscontinuity interface should also be provided by the FTTM to the application's ClockTarget entity.

19.3.3 Default Operations

19.3.3.1 General

The default DTSA state machine, the default ITSA state machine, and the default selection process used by both state machines are defined in this subclause.

Subclause 19.3.3.2 describes the default selection process, with its variables and pseudo-code.

Subclause 19.3.3.3 describes the default DTSA state machine, with its variables and its state diagram.

Subclause 19.3.3.4 describes the default ITSA state-machine, with its variables and its state diagram.

19.3.3.2 Mid-value time-index selection process

The mid-value trusted time-index selection process (MVTISP) 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 MVTISP shall be used by the default DTSA and by the default ITSA.

The MVTISP 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 19.2.7.4). 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.

19.3.3.2.1 Process variables

The variables used in the MVTISP are described in this subclause.

19.3.3.2.1.1 exclude_time_index[num_time_indexes] (Uinteger8)

This variable exclude_time_index[x] is a vector, of size num_time_indexes, of Boolean values that temporarily holds the index values of the trusted input ClockTarget interfaces as they are sorted into ascending order.

19.3.3.2.1.2 hyst[num_time_indexes][num_time_indexes]

This variable is a two-dimensional array of fractional nanosecond values. The array has a size of num_time_indexes in each dimension. This parameter holds the hysteresis added to maxAS[x][y] (see 19.3.3.2.1.3) for the times of the two ClockTarget interfaces with index numbers x and y.

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.

19.3.3.2.1.3 maxAs[x][y] (array of Uinteger32)

The variable maxAs[x][y] is a two-dimensional array of values, where each value gives the maximum magnitude of expected skew between times provided by the ClockTarget interface of index x and index y when those times are not faulty. The variable maxAS[x][y] is given in units of 2^{-16} ns. This value is used as the criteria to determine the trustworthiness of the times being compared. See $maxAS_{xy}$ in 19.2.7.4.

19.3.3.2.1.4 med_time_index (Uinteger8, NQ)

This variable is the selected input ClockTarget interface index number. The range of values for the input ClockTarget interfaces ranges from 1 to 255. The value of NQ represents a not-qualified time.

19.3.3.2.1.5 med_next_time_index (Uinteger8,NQ)

This variable is the index number for the input ClockTarget interface that is deemed to be the trusted partner for the selected input ClockTarget interface. The range of values for the input ClockTarget interfaces ranges from 1 to 255. The value of NQ represents a not-qualified time.

19.3.3.2.1.6 min_value (ExtendedTimestamp)

The variable min_value is a temporary value used for sorting the trusted ClockTarget interfaces from their times, lowest to highest.

19.3.3.2.1.7 num_sorted (Uinteger8)

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

19.3.3.2.1.8 num_time_indexes (Uinteger8)

This variable num_time_indexes contains the number of input ClockTarget interfaces connected to the DTSA instance. This parameter has a minimum value of 1.

The value of num_time_indexes does not include the NQ time (see 19.3.2.2.3).

19.3.3.2.1.9 ordered_time_index[num_sorted] (vector of Uinteger8)

The variable ordered_time_index[x] contains the Xth entry, starting from 1, of the trusted time indexes ordered from lowest to highest ToD value.

19.3.3.2.1.10 prev_trust_status

The variable prev_trust_status holds the last trust status of the DTSA. Valid values are NOT_TRUSTED and NO_TRUST, TIME_TRUST, and FREQ_TRUST.

19.3.3.2.1.11 prev_time_index_pair_status[x][y]

The variable prev_time_index_pair_status[x]y[] holds the trust status between ClockTarget interface s and ClockTarget interface y. Valid values are NOT_TRUSTED and TRUSTED.

19.3.3.2.1.12 trust_status (???)

The variable trust_status holds the type of trust the DTSA has currently advanced to. Valid values are NO_TRUST, TIME_TRUST, and FREQ_TRUST.

19.3.3.2.1.13 time_index_pair_status[x][y] (???)

The variable time_index_pair_status[x][y] holds the trust status between ClockTarget Interfaces x and y. Valid values are UNTRUSTED and TRUSTED.

19.3.3.2.1.14 time_index_status[x] (???)

The variable time_index_status[x] holds the trust status of ClockTarget Interface x, Valid values are UNTRUSTED and TRUSTED.

19.3.3.2.1.15 ToD[x] (vector of ExtendedTimestamp)

The variable ToD[x] holds the latest timeReceiverTimeCallback result for input ClockTarget interface x.

19.3.3.2.1.16 ToD_Diff[x][y] (array of ExtendedTimestamp)

The variable ToD_Diff[x][y] gives the time skew between the two ClockTarget interfaces, x and y.

19.3.3.2.1.17 x (Uinteger8)

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

19.3.3.2.1.18 y (Uinteger8)

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

19.3.3.2.2 Process pseudo-code

Pseudo-code that represents the MVTISP is given below.

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


// ######################################################################
// Clear status before starting a new round of time index comparisons.
// ######################################################################
trust_status = NO_TRUST
time_index_pair_status[x][y] = UNTRUSTED for all x and all y
time_index_status[x] = UNTRUSTED for all x
num_sorted = 1
exclude_time_index[x] = FALSE for all x


// ######################################################################
// Find all trusted time indexes, considering hysteresis.
// ######################################################################
For (x = 1, x <= num_time_indexes - 1, x++) {
    For (y = x + 1, y <= num_time_indexes, y++) {
        if ((ToD_Diff[x][y] <= maxAs[x][y] &&
             prev_time_index_pair_status[x][y] == UNTRUSTED) ||
             (ToD_Diff[x][y] <= (maxAs[x][y] + hyst[x][y]) &&
              prev_time_index_pair_status[x][y] == TRUSTED)) &&
            (isSynced_status[x] && gmPresent_status[x]) &&
            (isSynced_status[y] && gmPresent_status[y]))
        {
             // trust found for the pair
             trust_status = TIME_TRUST
             time_index_pair_status[x][y] = TRUSTED
             time_index_status[x] = TRUSTED
             time_index_status[y] = TRUSTED
             prev_time_index_pair_status[x][y] = TRUSTED
        }
        else
        {
                // trust not found for the pair
                prev_time_index_pair_status[x][y] = UNTRUSTED
          }
      }
}


// ########################################################################
// If trust_status = TIME_TRUST, find time index with the mid-value ToD.
// ########################################################################
// Trusted times detected.
If {trust_status == TIME_TRUST)
{
  // Update previous trust status.
  prev_trust_status = TIME_TRUST

  // 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 <= num_time_indexes, x++) {
    min_value = 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 <= num_time_indexes, y++) {
       if (time_index_status[y] == TRUSTED &&
           exclude_time_index[y] == FALSE &&
           ToD[y] <= min_value)
       {
         min_value = ToD[y] // record latest min ToD value found in the
                            // inner loop
         ordered_time_index[num_sorted] = y  // record latest time index
                                             // with min ToD value
       }
    }
    // Exclude latest time index with the min ToD value and add to sort index.
    exclude_time_index[ordered_time_index[num_sorted]] = TRUE
    num_sorted = num_sorted + 1
  }

  // Get median trusted time index and the trusted time index with
  // the next higher ToD.
  // The lower time index is selected if the number of trusted time
  // indexes is even.
      med_time_index      = ordered_time_index[INT((num_sorted)/2)]
      med_next_time_index = ordered_time_index[INT((num_sorted)/2)+1]
}

// If itsa_mode, check for transition to or sustaining frequency trust state.
// If time trust just lost or if already using frequency trust, keep
// existing median time index values if corresponding PTP Instance
// is still valid.
// Set previous trust status to FREQ_TRUST.
else if {itsa_mode == TRUE &&
         prev_trust_status == (TIME_TRUST || FREQ_TRUST) &&
         isSynced_status[med_time_index) == TRUE &&
         gmPresent_status(med_time_index) == TRUE)
{
  med_time_index = med_time_index
  med_next_time_index = med_next_time_index
  prev_trust_status = FREQ_TRUST
}
```

```
// No time trust or frequency trust so set output time indexes to NQ
// and clear previous trust status to NO_TRUST.
else
{
  med_time_index = NQ
  med_next_time_index = NQ
  prev_trust_status = NO_TRUST
}
```

19.3.3.3 Default DTSA state machine

The default DTSA state machine uses the ClockTargetEventCapture interface.

19.3.3.3.1 State machine variables

Still to be added…

19.3.3.3.2 State diagram

The default DTSA state machine is shown in Figure 5. The default DTSA shall use the MVTISP of 19.3.3.2 to find all the trusted input times and to select the index that corresponds to ClockTarget interface that holds the median ToD value from the trusted input times. The ClockTarget interface of the selected input dependent time shall be presented as the output of the default DTSA.

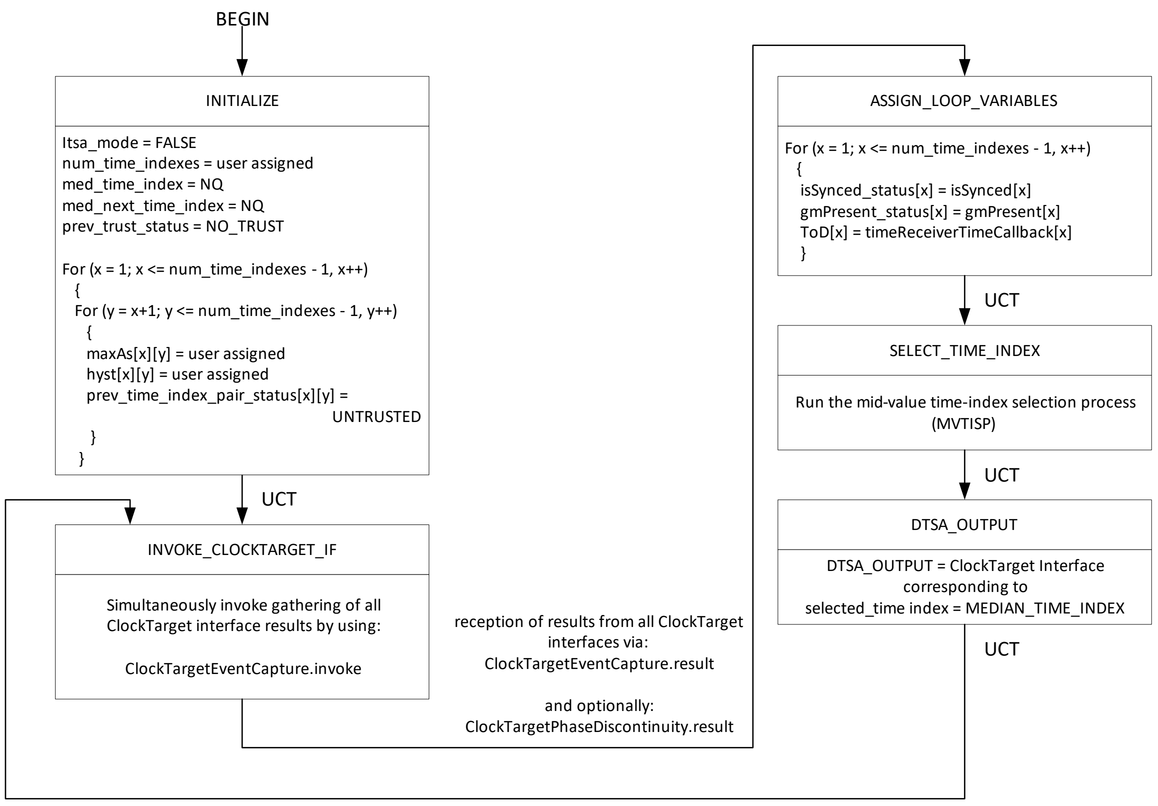
BEGIN
INITIALIZE
ltsa_mode = FALSE
num_time_indexes = user assigned
med_time_index = NQ
med_next_time_index = NQ
prev_trust_status = NO_TRUST
For (x = 1; x <= num_time_indexes - 1, x++) { For (y = x+1; y <= num_time_indexes - 1, y++) { maxAs[x][y] = user assigned hyst[x][y] = user assigned prev_time_index_pair_status[x][y] = UNTRUSTED } }
UCT
INVOKE_CLOCKTARGET_IF
Simultaneously invoke gathering of all ClockTarget interface results by using:
ClockTargetEventCapture.invoke
reception of results from all ClockTarget interfaces via:
ClockTargetEventCapture.result
and optionally:
ClockTargetPhaseDiscontinuity.result
ASSIGN_LOOP_VARIABLES
For (x = 1; x <= num_time_indexes - 1, x++) { isSynced_status[x] = isSynced[x] gmPresent_status[x] = gmPresent[x] ToD[x] = timeReceiverTimeCallback[x] }
UCT
SELECT_TIME_INDEX
Run the mid-value time-index selection process (MVTISP)
UCT
DTSA_OUTPUT
DTSA_OUTPUT = ClockTarget Interface corresponding to selected_time index = MEDIAN_TIME_INDEX
UCT

Figure 5— Default DTSA state machine

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

The INVOKE_CLOCKTARGET_IF state of the default DTSA state machine simultaneously invokes all the ClockTarget interfaces connected to the DTSA instance to provide timing information. Once all the ClockTarget interfaces respond to the invocation with their timing information results, this state transitions to the ASSIGN_LOOP_VARIABLES state.

The ASSIGN_LOOP_VARIABLES state of the default DTSA state machine assigns three timing result parameters (isSynced, gmPresent, and timeReceiverTimeCallback see clause 9) from each of the ClockTarget interfaces to DTSA variables. Once this assignment is finished, this state unconditionally transfers to the SELECT_TIME_INDEX state.

The SELECT_TIME_INDEX state of the default DTSA state machine runs the MVTISP (see 19.3.3.2) to find the trusted times, if any, from the input ClockTarget interfaces, sort them in ascending ToD order, and identify the index number of the input ClockTarget interface with the median ToD value. If no trusted time is found, then the index number for the NQ time is identified. This index number is then passed to the DTSA_OUTPUT state.

The DTSA_OUTPUT state of the default DTSA state machine passes the ClockTarget interface identified by the MVTISP of the SELECT_TIME_INDEX state to the output of the DTSA’s output.

19.3.3.4 Default ITSA state machine

The default ITSA state machine uses the ClockTargetEventCapture interface.

19.3.3.4.1 State machine variables

Still to be added…

19.3.3.4.1.1 itsaRateRatio[num_input_times] (vector of Float64)

The variable itsaRateRatio[x] is equal to the ratio of the clock frequency of the time arriving on input ClockTarget interface x to the frequency of the FTTM's local clock, OSC_CLK.

19.3.3.4.2 State diagram

The default ITSA state machine is shown in Figure 6 and Figure 7. The default ITSA shall use the MVTISP of 19.3.3.2 to find all the trusted input times and to select the index that corresponds to ClockTarget interface that holds the median ToD value from the trusted input times or the index that corresponds to the NQ time if no trusted time is available. The ClockTarget interface of the selected time shall be presented as the output of the default ITSA and, hence, as the output of the FTTM.

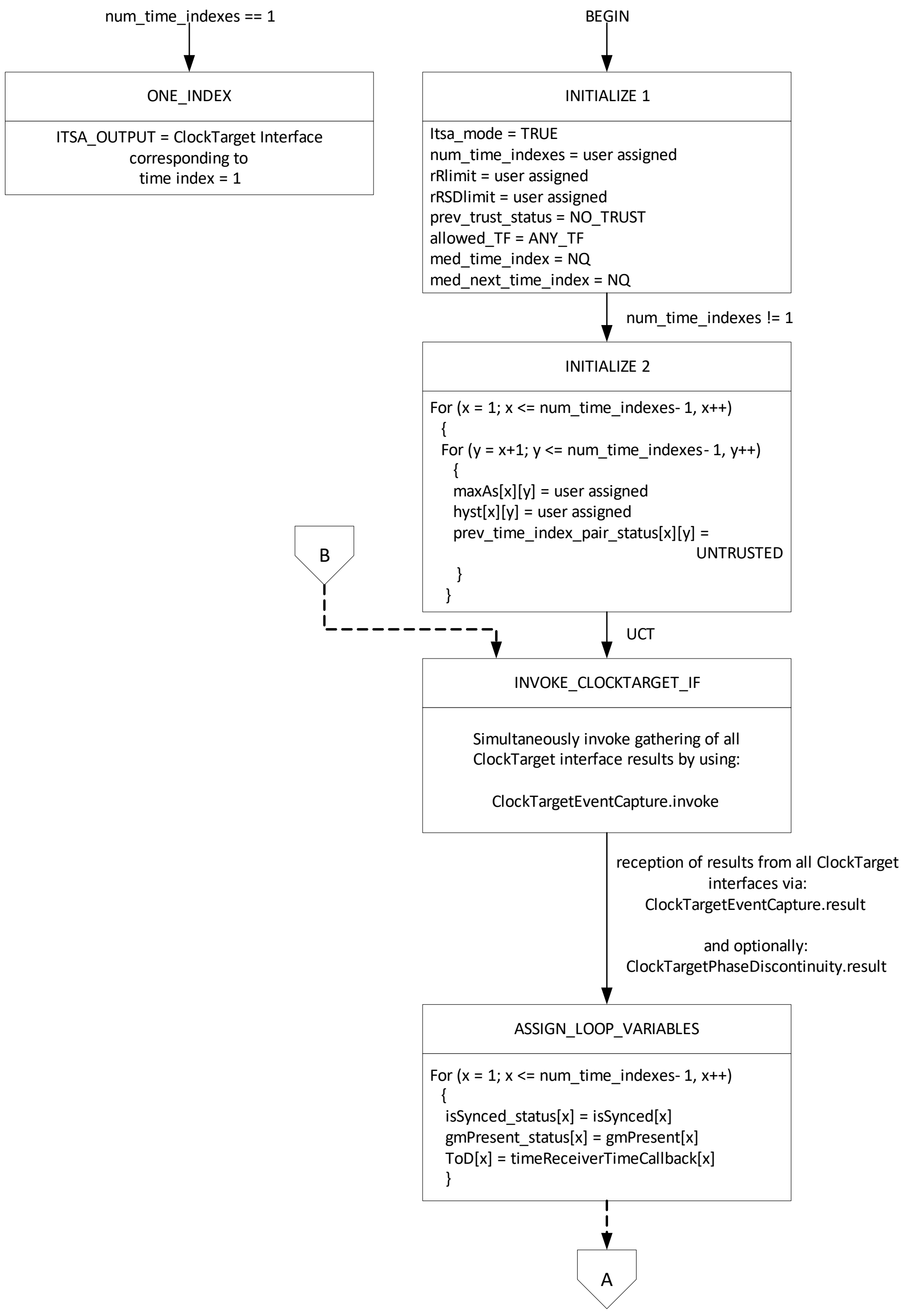
num_time_indexes == 1
BEGIN
ONE_INDEX
ITSA_OUTPUT = ClockTarget Interface corresponding to time index = 1
INITIALIZE 1
Itsa_mode = TRUE
num_time_indexes = user assigned
rRlimit = user assigned
rRSDlimit = user assigned
prev_trust_status = NO_TRUST
allowed_TF = ANY_TF
med_time_index = NQ
med_next_time_index = NQ
num_time_indexes != 1
INITIALIZE 2
For (x = 1; x <= num_time_indexes- 1, x++) { For (y = x+1; y <= num_time_indexes- 1, y++) { maxAs[x][y] = user assigned hyst[x][y] = user assigned prev_time_index_pair_status[x][y] = UNTRUSTED } }
B
UCT
INVOKE_CLOCKTARGET_IF
Simultaneously invoke gathering of all ClockTarget interface results by using:
ClockTargetEventCapture.invoke
reception of results from all ClockTarget interfaces via: ClockTargetEventCapture.result
and optionally: ClockTargetPhaseDiscontinuity.result
ASSIGN_LOOP_VARIABLES
For (x = 1; x <= num_time_indexes- 1, x++) { isSynced_status[x] = isSynced[x] gmPresent_status[x] = gmPresent[x] ToD[x] = timeReceiverTimeCallback[x] }
A

Figure 6—ITSA state machine, part 1

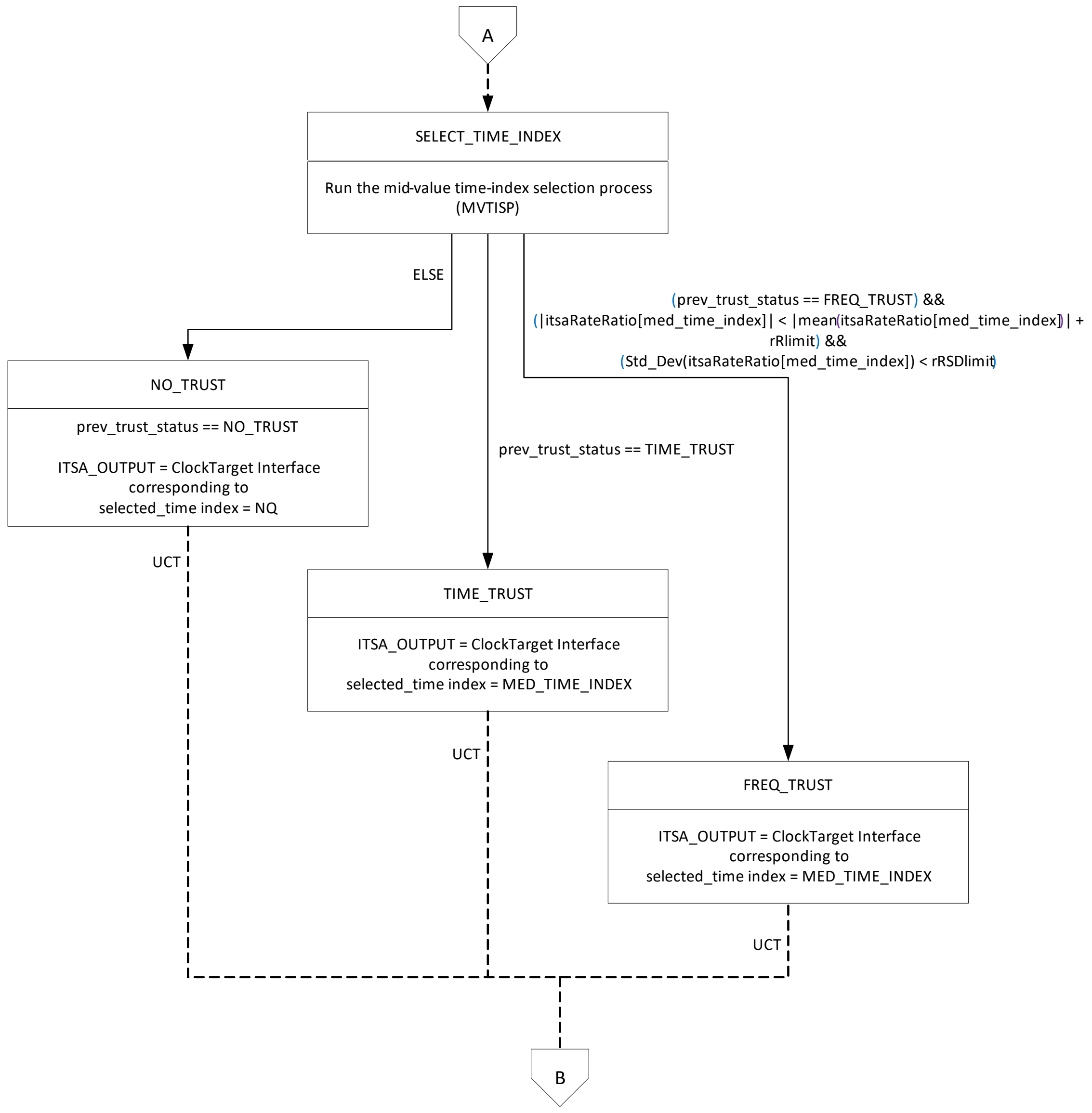
A
SELECT_TIME_INDEX
Run the mid-value time-index selection process (MVTISP)
ELSE
(prev_trust_status == FREQ_TRUST) && (|itsaRateRatio[med_time_index]| < |mean(itsaRateRatio[med_time_index])| + rRlimit) && (Std_Dev(itsaRateRatio[med_time_index]) < rRSDlimit)
NO_TRUST
prev_trust_status == NO_TRUST
ITSA_OUTPUT = ClockTarget Interface corresponding to selected_time index = NQ
prev_trust_status == TIME_TRUST
UCT
TIME_TRUST
ITSA_OUTPUT = ClockTarget Interface corresponding to selected_time index = MED_TIME_INDEX
UCT
FREQ_TRUST
ITSA_OUTPUT = ClockTarget Interface corresponding to selected_time index = MED_TIME_INDEX
UCT
B

Figure 7—ITSA state machine, part 2

The INITIALIZE 1 state of the default ITSA state machine sets up the starting values of some variables. Once these starting values are configured, this state unconditionally transfers to the INITIALIZIZE 2 state except for the case in which the assigned value of num_time_indexes is equal to one, in which the state instead transfers to the ONE_INDEX state.

The ONE_INDEX state of the default ITSA state machine is used when there is only one input ClockTarget interface connected to the ITSA. This corresponds to the regular availability and no integrity operational scenario described in 19.3. The state machine can only exit this state via the BEGIN condition.

The INITIALIZE 2 state of the default ITSA state machine sets up the starting values of variables that only make sense when the assigned value of num_time_indexes is greater than one. Once these starting values are configured, this state unconditionally transfers to the INVOKE_CLOCKTARGET_IF state.

The INVOKE_CLOCKTARGET_IF state of the default ITSA state machine simultaneously invokes all the ClockTarget interfaces connected to the ITSA instance to provide timing information. Once all the ClockTarget interfaces respond to the invocation with their timing information results, this state transitions to the ASSIGN_LOOP_VARIABLES state.

The ASSIGN_LOOP_VARIABLES state of the default ITSA state machine assigns three timing result parameters (isSynced, gmPresent, and timeReceiverTimeCallback see clause 9) from each of the ClockTarget interfaces to ITSA variables. Once this assignment is finished, this state unconditionally transfers to the SELECT_TIME_INDEX state.

The SELECT_TIME_INDEX state of the default ITSA state machine runs the MVTISP (see 19.3.3.2) to find the trusted times, if any, from the input ClockTarget interfaces. If trusted times are found, it sorts them in ascending ToD order and identifies the index number of the input ClockTarget interface that has the median ToD value.

— If the MVTISP returns a prev_trust_status = TIME_TRUST, this means that it has found a trusted time and selected a corresponding ClockTarget interface. The index of the selected ClockTarget interface is provided as the output of the SELECT_TIME_INDEX state and the state machine transitions to the TIME_TRUST state.
— If the MVTISP returns a prev_trust_status = FREQ_TRUST, this means that it has trust time but is trying to keep operating with the last selected ClockTarget interface, which still has isSynced and gmPresent asserted as TRUE. This last selected ClockTarget interface is presented as the output of the SELECT_TIME_INDEX state. If this ClockTarget interface is still providing a time that is incrementing consistently with respect to the FTTM’s local clock (OSC_CLK), as determined by the itsaRateRatio[x] measurement for the ClockTarget interface, then the state machine transitions to the FREQ_TRUST state.
— If neither of the above cases is true, then the ITSA has no trust in any of its input times. The state machine transitions to the NO_TRUST state.

The NO_TRUST state of the default ITSA state machine passes the NQ index to the output of the ITSA’s output and, hence, the FTTM’s output. This indicates that no trusted time is being passed from the FTTM. The state machine unconditionally transitions back to the INVOKE_CLOCKTARGET_IF state to start another round of integrity checking.

The TIME_TRUST and the FREQ_TRUST states of the default ITSA state machine both pass the ClockTarget interface identified by the MVTISP of the SELECT_TIME_INDEX state to the output of the ITSA’s output and, hence, the FTTM’s output. This indicates that a trusted time is being passed from the FTTM. The state machine unconditionally transitions back to the INVOKE_CLOCKTARGET_IF state to start another round of integrity checking.

Annex A (normative) Protocol Implementation Conformance Statement (PICS) proforma

Annex B (informative) Performance Requirements

Annex C (Informative) Timescales and epochs

Annex D Reserved for future use

Annex E Reserved for future use

Annex F (informative) PTP profile included in this standard

Annex G (informative) The asymmetry compensation measurement procedure based on line-swapping

Annex H (informative) Bibliography

Annex I (informative) Example Aerospace Configuration

Annex J (informative) Time synchronization with Fault Tolerance and Integrity

J.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 address the reliable and accurate transmission of time values and the associated sync and follow-up messages 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.

J.2 Clock domain management

As described in 19.2, clock domains can be considered dependent or independent. Independent clock domains, where ClockSources are independent, are expected to present problems to the integrator because, at the time of writing, commercially available devices cannot be relied upon to support multiple independent gPTP Instances at a single port. This makes it problematic to bridge synchronized traffic between domains. In Figure J-1, two clock domains D1 and D2 are shown overlapping at Bridges B2 and B3 with streams S1 and S2 sharing a common output port, P4, on B2. If the two clock domains are synchronized, Bridges B2 and B3 will synchronize to the common domain time and will be able to forward both streams to the downstream end stations. However, if the two clock domains are not synchronized, then a conflict can occur on the shared output port of B2 such that it must either maintain two PTP Instances on the shared output port and widen the output windows to accommodate a potential conflict or must forward one of the streams in an unsynchronized manner.

It is not possible for a device to support multiple unsynchronized gate schedules on a single output port, and aerospace networks using multiple PTP domains should therefore ensure that the clock domains are either dependent on a common ClockSource or are synchronized to each other by some other means (see 19.2.1).

Management of multiple PTP instances using a fault-tolerant timing module (FTTM) is discussed in 19.3.

<Insert Figure D-1 from P802.1DP/D1.1)

Figure J-1 – Multiple gPTP domains with shared port

J.3 Time agreement generation examples

PPS-based implementations.

TTE-based implementations.

J.4 Balancing availability and integrity

A common time source is used for all GMs.

Only 2 domains are used, instead of 3 or more.

An insufficient number of independent paths are available in the network for all PTP domains.

J.5 FTTM operation in example network topologies

A.1.1 N x point-to-point network (one for each gPTP domain)

A.1.2 Dual-homed network

A.1.3 Multi-homed network

A.1.4 Dual-star network

A.1.5 Bidirectional ring network

A.1.6 Mesh network