IEEE Std 1588™-2002

IEEE Standards

1588™

IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

IEEE Instrumentation and Measurement Society

Sponsored by the
TC9—Technical Committee on Sensor Technology

IEEE

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

8 November 2002

Print: SH95023
PDF: SS95023

IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

Sponsor

TC9—Technical Committee on Sensor Technology
of the
IEEE Instrumentation and Measurement Society

Approved 12 September 2002

IEEE-SA Standards Board

Abstract: A protocol to synchronize independent clocks running on separate nodes of a distributed measurement and control system to a high degree of accuracy and precision is specified. The protocol is independent of the networking technology, and the system topology is self-configuring.
Keywords: clocks, distributed system, master clock, measurement and control systems, real-time clock, synchronized clocks

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

Print: ISBN 0-7381-3369-8 SH95023
PDF: ISBN 0-7381-3370-1 SS95023

Patent notice

Introduction

(This introduction is not a part of IEEE Std 1588-2002, IEEE Standard for for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.)

The objective of this standard is to specify a protocol to synchronize independent clocks running on separate nodes of a distributed measurement and control system to a high degree of accuracy and precision. The clocks communicate with each other over a communication network. In its basic form, this protocol is intended to be administration free. The protocol generates a master slave relationship among the clocks in the system. Within a given subnet of a network, there will be a single master clock. All clocks ultimately derive their time from a clock known as the grandmaster clock. The communication path between any clock and its grandmaster clock is part of a minimum spanning tree.

Participants

The following persons contributed to earlier drafts of this standard:

Volker Arlt
Robert Johnson
Kang Lee
Heinrich Munz
Ed Powers
Jim Read
Richard Schmidt

The following persons were members of the committee at the time of publication:

John C. Eidson, *Chair*
Bruce Hamilton, *Editor*
Steven Jennings, *Secretary*

Scot Carter
Richard Hambly
R. William Kneifel, II
Jürgen Knopke
Jack Kusters
Judah Levine
Anatoly Moldovansky
Stephen Smith
Joe White
Stan P. Woods

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

Chris Bagge
Edul Batliwala
Jacob Benary
L. Wayne Catlin
Keith Chow
John C. Eidson
Patrick Gonia
Erich Gunther
Richard Hambly
Bruce Hamilton
Kang Lee
Yeou-Song Lee
Gregory Luri
Andrea Mariscotti
Peter Martini
David Miller
Anatoly Moldovansky
Andrew Thomson
Steven Tilden
Jay Warrior
Stephen C. Webb
John Westmoreland
Ronald Wolfe
Stan P. Woods
Hadrian Zbarcea

Contents

IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

1. Overview

This standard is divided into nine clauses. They are as follows:

Clause	Purpose
1	Provides the scope and benefits of this standard
2	Lists references to other standards that are referenced by this standard
3	Provides definitions that are either not found in other standards or have been modified for use with this standard
4	Provides conventions for the notation used in this standard
5	Defines the datatypes used in this standard
6	Provides an overview of the protocol specified by the standard
7	Defines the precision time protocol (PTP)
8	Defines the format of messages passed between participating clocks
9	Defines requirements for conformance

Annexes are provided as follows:

Annex	Purpose
A	Using the PTP
B	Defines time scales and epochs in PTP
C	Defines subdomain_name to address mappings
D	Defines the Ethernet implementation of PTP
E	Bibliography

Annexes defining communication-medium-specific implementation details for additional network implementations are expected to be provided in future revisions of this standard.

1.1 Scope

This standard defines a protocol enabling precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The protocol will be applicable to systems communicating by local area networks supporting multicast messaging including, but not limited to, Ethernet. The protocol will enable heterogeneous systems that include clocks of various inherent precision, resolution, and stability to synchronize. The protocol will support systemwide synchronization accuracy in the submicrosecond range with minimal network and local clock computing resources. The default behavior of the protocol will allow simple systems to be installed and operated without requiring the administrative attention of users.

1.2 Purpose

Measurement and control applications are increasingly using distributed system technologies such as network communication, local computing, and distributed objects. Many of these applications will be enhanced by having an accurate systemwide sense of time achieved by having local clocks in each sensor, actuator, or other system device. Without a standardized protocol for synchronizing these clocks, it is unlikely that the benefits will be realized in the multivendor system component market. Existing protocols for clock synchronization are not optimum for these applications. For example, Network Time Protocol (NTP) targets large distributed computing systems with millisecond synchronization requirements. The protocol proposed in this standard specifically addresses the needs of measurement and control systems:

— Spatially localized
— Microsecond to submicrosecond accuracy
— Administration free
— Accessible for both high-end devices and low-cost, low-end devices

2. References

This standard shall be used in conjunction with the standards listed in this clause. When the following standards are superseded by an approved revision, the revision shall apply.

IEEE Std 802.3™-2002, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.[1]

ISO 8601:2000, Data elements and interchange formats—Information interchange—Representation of dates and times.[2]

ISO/IEC 8859-1:1998, Information technology—8-bit single-byte coded graphic character sets—Part 1: Latin alphabet No. 1

[1]IEEE Publications are available from the Institute of Electrical and Electronics Engineers. 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA and <http://standards.ieee.org/sds/index.html>.

[2]ISO publications are available from the International Organization for Standardization, <http://www.iso.ch/iso/en/prods-services/ISOstore/store.html>.

3. Definitions

This clause defines terms that have specific meanings in the context of the PTP.

3.1 boundary clock: A clock with more than a single PTP port, with each PTP port providing access to a separate PTP communication path.

3.2 clock: A node that is capable of providing a measurement of the passage of time since a defined epoch. In this standard, the term *clock* is interpreted to mean either an ordinary clock or a PTP port of a boundary clock unless otherwise stated or obvious from the context.

3.3 clock timestamp point: A point in the network protocol stack of a clock at which timestamps are generated for Sync and Delay_Req messages sent or received by the clock. This point is defined for both the inbound and outbound paths in the protocol stack.

3.4 direct communication: A communication of PTP information between two PTP clocks with no intervening boundary clock.

3.5 epoch: The reference time defining the origin of a time scale.

3.6 event: An abstraction of the mechanism by which signals or conditions are generated and represented.

3.7 external synchronization: The process of bringing two clocks into a synchronized state by means other than the use of the PTP. *See also:* **synchronized clocks.**

3.8 grandmaster clock: Within a PTP subdomain, a PTP clock that is the ultimate source of time for clock synchronization using the PTP protocol.

3.9 interface definition language (IDL): A programming-language-independent method of specifying operation syntax [B1].[3]

3.10 master clock: In the context of a single PTP communication path, a clock that when viewed from the path appears to be the source of time to which all other clocks synchronize.

3.11 message timestamp point: A distinguished feature of Sync and Delay_Req messages serving as a reference point in these messages. A timestamp is defined by the instant a message timestamp point passes the clock timestamp point in a protocol stack.

3.12 network: A communication mechanism for passing PTP messages among multiple clocks.

3.13 node: A device that can issue or receive PTP communications on a network.

3.14 ordinary clock: A PTP clock with a single PTP port.

3.15 ordinary communication: A communication of non-PTP information between two nodes.

3.16 port number: An index identifying a specific PTP port on a PTP clock.

3.17 preferred master clock set: A set of clocks that will be favored over those not so designated in the selection of the grandmaster clock within a subdomain.

[3]The numbers in brackets correspond to those of the bibliography in Annex E.

3.18 precision time protocol (PTP): The protocol defined by this standard. As an adjective, it indicates that the modified noun is specified in or interpreted in the context of this standard.

3.19 PTP clock: A clock that participates in the PTP protocol.

3.20 PTP communication: Information used in the operation of the PTP protocol, transmitted in a PTP message over a PTP communication path.

3.21 PTP communication path: A segment of a network enabling direct communication between two PTP clocks.

3.22 PTP domain: A collection of one or more PTP subdomains.

3.23 PTP message: One of the five designated messages types defined in this standard: Sync, Delay_Req, Follow-up, Delay_Resp, and Management.

3.24 PTP multicast communication: A PTP message sent from any PTP port and received and processed by all PTP ports on the same PTP communication path. PTP multicast communications are not automatically forwarded to other PTP communication paths by routers or other similar network components.

3.25 PTP node: A node that issues any message that will be received by a second node resulting in any change in state in the second node that influences any aspect of the PTP protocol.

3.26 PTP port: The logical access point to a PTP clock for PTP communications on a single PTP communication path.

3.27 PTP subdomain: A logical grouping of PTP clocks that synchronize to each other using the PTP protocol but that are not necessarily synchronized to PTP clocks in another PTP subdomain.

NOTE—It should be emphasized that this grouping is logical. Clocks sharing a PTP communication path may or may not be in the same PTP subdomain, and clocks in the same PTP subdomain may or may not share a PTP communication path.

3.28 state transition diagram: A graphical means of expressing the allowed states of an object and the allowed transitions from one state to another (see 4.3).

3.29 subnet: A subset of a network. If a network contains devices whose function is to pass messages and these devices do not pass PTP non-management messages, then the network can be partitioned into regions in such a way that:

— No two nodes in a region communicate via one of these devices, and
— All communication between regions is via one or more of these devices.

Each such region is a subnet. If a network does not contain any such devices, it is a subnet.

3.30 synchronized clocks: Two clocks are synchronized to a specified uncertainty if they have the same epoch, and measurements of any time interval by both clocks differ by no more than the specified uncertainty. The timestamps generated by two synchronized clocks for the same event will differ by no more than the specified uncertainty.

3.31 timeout: A mechanism for terminating requested activity, at least from the requester's perspective, that does not complete within the time specified.

3.32 universally unique identifier (UUID): An identifier that has a unique value within some defined universe. For purposes of this standard, the universe consists of all possible PTP artifacts having a UUID unless otherwise stated.

4. Conventions

The specifications of this clause shall apply to all artifacts defined in this standard.

4.1 Descriptive syntax

The syntax conventions specified in the following subclauses are used in this standard.

4.1.1 Lexical form syntax

A lexical form refers to:

— A name
— A datatype

The conventions illustrated in the following list regarding lexical forms shall be used:

— *Type names:* e.g., TimeRepresentation (no word separation, initial letter of each word capitalized)
— *Enumeration members and global constants:* e.g., PTP_SYNC_MESSAGE (underscore word separation, all letters capitalized)
— *Fields of structures or messages:* e.g., seconds, specialElement (no word separation, initial word not capitalized, initial letter capitalization on subsequent words)
— *Data set members and all other variables:* new_master (underscore word separation, no letters capitalized)
— *<local name for something>:* text enclosed in angle brackets, < >, is used where the standard needs to refer to something whose syntax or lexical form is dependent on the local implementation and language.

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

4.2 Word usage

4.2.1 Shall

The word *shall*, equivalent to *is required to,* is used to indicate mandatory requirements, strictly to be followed in order to conform to the standard and from which no deviation is permitted.

4.2.2 Recommended

The word *recommended* is used to indicate flexibility of choice with a strong preference alternative.

4.2.3 Must

The use of the word *must* is deprecated and shall not be used when stating mandatory requirements. The word *must* is only used to describe unavoidable situations.

4.2.4 Should

The word *should*, equivalent to *is recommended that,* is used to indicate:

— Among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others.
— A certain course of action is preferred but not required.
— In the negative form, a certain course of action is deprecated but not prohibited.

4.2.5 May

The word *may*, equivalent to *is permitted,* is used to indicate a course of action permissible within the limits of the standard.

4.2.6 Can

The word *can*, equivalent to *is able to,* is used to indicate possibility and capability, whether material or physical.

4.3 Behavioral specification notation

State transition diagrams are used to specify behavioral characteristics as illustrated in Figure 1. Each state transition diagram is composed of the following components:

— *Named boxes,* representing states
— *Directed arrows,* indicating transitions from one state to the next

Each transition is labeled with:

— The enabling event or predicate label for a transition
— The transition action label for a transition

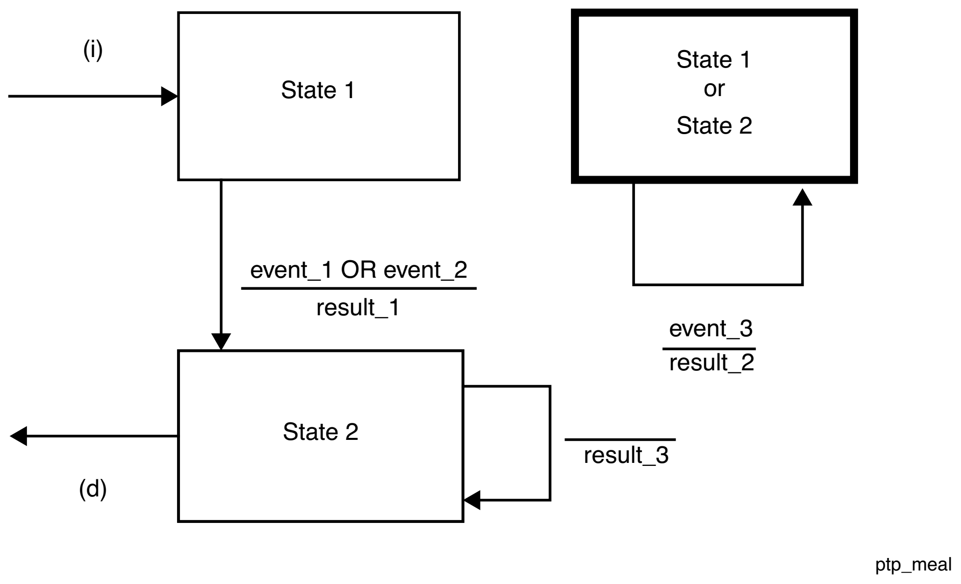

Figure 1—Mealy state transition diagram

The notation used describes state transition diagrams using the Mealy style, where actions are associated with the transition from one state to another.

Events—for example, event_1, event_2, and event_3—identify the inputs to the state machine. They can be operation requests and responses, or internal occurrences such as timer expirations.

Predicates—for example, event_1 OR event_2—identify enabling conditions for transitions. The first predicate encountered, evaluating from left to right, that is TRUE selects the transition to execute and therefore the next state.

Transition actions—for example, result_1—are the actions that are executed before transitioning to the next state.

The next state identifies the state for the state machine after the selected transition action completes. As the transition to the next state occurs, the value of the current state changes.

A bold line for a state box indicates that the box represents multiple states. Any transition shown that begins and terminates in such a state box indicates that there has been no change in state.

Transitions—for example, the transition resulting in result_3—that have no indicated enabling conditions, occur via unspecified mechanisms. Unless otherwise stated, in PTP these mechanisms are implementation-specific and outside the scope of the standard.

A transition into a state machine—for example, (i)—is indicated by a transition arrow that has no source state. A transition out of a state machine—for example, (d)—is indicated by a transition arrow with no destination state.

Example: As a result of either event_1 or event_2 becoming TRUE, State 1 is replaced with the value of the next state. In this example, the next state is State 2, which is specified as the name of the state box that is the target of the transition arrow. Before the transition, result_1 occurs. event_3 can occur in either State 1 or State 2. The state is unchanged but an action, result_2, occurs as the result of event_3.

5. Datatypes in a PTP system

For every datatype defined in a PTP system there shall be a default value designated. Quantities declared as having a type shall be instantiated with the default value unless otherwise specified.

The datatypes specified for the various PTP variables and message fields define logical properties, such as rollover for integers, necessary for correct operation of the protocol or interpretation of PTP message content.

The definition of data of types defined in this standard shall be correctly represented in the local system after such data has been communicated over the network. In any PTP implementation, these datatypes shall be mapped to implementation language-dependent primitives.

Implementations are free to use any internal representation preserving these characteristics provided that the internal representation shall not change the semantics of any quantity visible via PTP communications.

The mapping of datatypes into their on-the-wire format is network specific. For each network used in a PTP system, an annex to this standard shall be required that includes this mapping. Unless otherwise specified in this standard, the mapping in the network specific annexes shall be used for all PTP communications.

The following normative definitions are used in this clause.

Definition	Meaning
IDL: typedef <datatype> <datatypeArray>[ ];	An implementation of PTP Array of the datatype: <datatype>
IDL: typedef <datatype1> <datatype2>;	Where <datatype1> is a primitive or previously defined PTP datatype, shall mean that <datatype2> is a synonym for <datatype1>

5.1 Primitive datatypes

The datatypes defined in Table 1 shall be defined in all PTP systems. All other datatypes shall be derived from these primitive types.

Table 1—Primitive PTP datatypes

Datatype	Default value	Definition
Boolean	FALSE	TRUE or FALSE
Integer8	0	8-bit signed integer
UInteger8	0	8-bit unsigned integer
Integer16	0	16-bit signed integer
UInteger16	0	16-bit unsigned integer
Integer32	0	32-bit signed integer
UInteger32	0	32-bit unsigned integer
Integer64	0	64-bit signed integer
UInteger64	0	64-bit unsigned integer
Octet	All bits set to 0	8-bit field not interpreted as a number

5.2 Derived datatypes

The datatypes in this clause are based on the primitive datatypes. Only the primitive or derived datatypes defined in this standard are allowed as part of a PTP message.

5.2.1 Specification of arrays of primitive types

All arrays defined by this standard shall be of fixed length. Any of the following array types may be used provided the length is specified. Unless otherwise specified in this standard or network specific annexes, all arrays shall be marshaled into an on-the-wire format with the member having the lowest numerical index first.

IDL: typedef Boolean	BooleanArray[];
IDL: typedef Integer8	Integer8Array[];
IDL: typedef UInteger8	UInteger8Array[];
IDL: typedef Integer16	Integer16Array[];
IDL: typedef UInteger16	UInteger16Array[];
IDL: typedef Integer32	Integer32Array[];
IDL: typedef UInteger32	UInteger32Array[];
IDL: typedef Integer64	Integer64Array[];
IDL: typedef UInteger64	UInteger64Array[];
IDL: typedef Octet	OctetArray[];

5.2.2 Time type specifications

The TimeRepresentation type shall be used to specify both:

— Timestamps (Time with respect to an epoch)
— Time increments

IDL: struct TimeRepresentation

```
{
        UInteger32 seconds;
        Integer32 nanoseconds;
};
```

The range of the absolute value of the nanoseconds member shall be restricted to:

$$0 \leq |\text{nanoseconds}| < 10^9$$

The sign of the nanoseconds member shall be interpreted as the sign of the entire representation.

A negative timestamp shall indicate time prior to the epoch.

A negative increment shall result for example from subtracting a timestamp A from a timestamp B, where A is an instant later in time than B.

Example: A timestamp whose seconds member value is 10 and whose nanoseconds member value is -5×10^8 represents an instant 10.5 seconds before the epoch.

5.2.3 Enumeration type specifications

An enumeration member of a PTP enumeration type shall be represented as a UInteger8. Unless otherwise noted, the assignment of values to the enumeration members shall start at 0 (zero) for the first member listed of an enumeration and shall increment by +1 (one) for each succeeding member. Unless otherwise stated, the default value of an enumeration member shall be 0.

Any enumeration used in a PTP system shall have members of the form PTP_<suffix>. All PTP enumerations shall be unique.

5.2.3.1 Enumeration semantics

Enumerations defined in this standard always appear in message formats as UInteger8 types. For each enumeration, the value of the UInteger8 has been assigned for each member of the enumeration. Enumerations have no print form specified. Implementations shall not include any operational dependency on the particular textual description used in the definition of the enumeration. Any textual description of the semantics associated with the enumeration value shall not appear in the signature of any operation, or in the format of any PTP message, except for informational purposes.

5.2.3.2 Enumeration summary list

Most enumerations are defined in the clause where they are first used. This clause provides a list of all enumerations defined in this standard along with a reference to the defining clause.

Enumeration name	Reference clause
CommunicationId	6.2.4.1
ControlField	8.2.9
ManagementMessage	7.12
PTPState	7.3.1

6. PTP Clock synchronization model

6.1 Overview

6.1.1 General overview

The PTP standard specifies a clock synchronization protocol. This protocol is applicable to distributed systems consisting of one or more nodes, communicating over some set of communication media. Nodes are modeled as containing a real-time clock that may be used by applications within the node for various purposes, such as generating timestamps for data or ordering events managed by the node. The PTP protocol provides a mechanism for synchronizing the clocks of participating nodes to a high degree of accuracy. Guidelines for the design and use of systems supporting PTP are given in Annex A.

The standard specifies the following:

— The PTP protocol
— The node and communication properties necessary to support PTP

A PTP system is a distributed system consisting of ordinary clocks, possibly boundary clocks, and possibly administrative nodes. A boundary clock is a clock with a clock port for each of two or more distinct PTP communication paths. There are two aspects of the PTP protocol: the synchronization aspect and the administrative aspect.

Externally, all ordinary clocks appear identical in all aspects of the protocol. Furthermore, each PTP port of a boundary clock externally appears to be an ordinary clock in terms of the protocol. An administrative node is not required to implement the synchronization aspect of the protocol unless it is also a clock node.

6.1.2 Operational overview

A PTP communication path supports the direct communication among the PTP ports of a set of ordinary clocks and the PTP ports of any boundary clocks accessing the communication path. The general behavior of the clocks accessing a given communication path follows. In this discussion, the term *ports* will include the ports of any boundary clocks attached to the communication path.

A clock containing one of these ports will be selected as the *master clock* for the path. This selection is made by each port examining information contained in special messages termed Sync messages. A Sync message is sent periodically by any port associated with a clock claiming to be the master clock. All ports use the same algorithm, termed the *best master clock* algorithm. If a port of a master clock receives a Sync message from a *better* clock then that clock will cease to claim to be a master and the receiving port will assume the status of a *slave*. Likewise if a clock with a port acting as a slave determines that it would make a better master than the current master clock, it assumes the status of *master* and begins to send Sync messages.

To provide more orderly behavior when a clock comes on line, a clock will listen for Sync messages from a master clock for a system-specified time. If no Sync message is received within this time, the clock will assume it is the master clock until such time as a better clock appears. An additional mechanism to support more orderly reconfiguration of systems when clocks are added or deleted, clock characteristics change or connection topology changes is embodied in the PTP_PRE_MASTER state. In this state a clock port behaves exactly as it would if it were in the PTP_MASTER state except that it does not place certain classes of messages on the port communication path. A clock port remains in this premaster state long enough to allow changes to occur at points in the system between the local clock and possible master clocks visible from the port.

Sync messages from a port of the master also contain an estimate of the time the Sync message will be placed on the PTP communication path. This estimate is based on the time kept by the local clock of the master.

Master clock ports may also send Follow_Up messages. A Follow_Up message is always associated with a specific Sync message and contains a more precise estimate of the time the Sync message was placed on the PTP communication path.

Slave ports use the information contained in the Sync or Follow_Up messages to correct their local clock in such a way as to synchronize the local clock to the clock of the master.

This correction may be improved by accounting for the propagation delay experienced by messages in traveling from the master to the slave. This additional correction may be estimated by reversing the previous process. That is, the slave sends a message termed a Delay_Req message to the master. No Follow_Up message is used for this procedure. The master then reports the time of receipt back to the slave in a Delay_Resp message. This process is normally exercised infrequently to reduce network traffic.

In addition to issuing Sync messages, a master clock, and only a master clock, may also provide an external timing signal. These external timing signals shall be issued on the seconds' transition of the local clock. The external timing signal is transmitted only to slaves in the same subdomain, preferably in the same PTP communication path, and is transmitted on a medium other than the medium communicating the Sync messages. The behavior of systems where external timing signals are propagated to clocks on PTP communication paths not associated with a port on the issuing master clock is beyond the scope of this standard.

Boundary clocks serve to implement a time distribution tree (or set of trees). A boundary clock is required wherever there is a change of communication technology or a network element which blocks Sync, Follow_Up, Delay_Req, or Delay_Response messages. It is recommended that a boundary clock be used wherever there is a network element that inserts significant delay fluctuation. The boundary clock is a slave to the path containing the best clock it can see and asserts that it is the master for all other accessed paths. Since boundary clocks like ordinary clocks contain an internal clock, it is possible that this internal clock is the best clock it can see. In this case the boundary clock itself will be the master in all accessed paths. Thus in a system with boundary clocks, there may be several masters, one for each communication path. The best clock of all is designated the grandmaster. Sufficient information is contained in each Sync message to allow the best master clock algorithm to produce the correct overall system configuration and behavior.

6.1.3 Assumptions

The PTP standard makes several assumptions about the environment in which it operates. The following assumptions must be met to ensure correct operation of the protocol:

- The network must support multicast communication (see 6.2.2.2).
- It must be possible to prevent multicast messages from propagating beyond a subnet.
- Each clock implementing the protocol must meet the performance requirements of 7.10.
- A clock's stated properties, including its stratum (see 6.2.4.3) and identifier (see 6.2.4.5), must accurately describe the clock.

The following assumptions must be met to achieve optimal clock synchronization performance:

- Network delay between master and slave on a subnet must be symmetric (see 7.8.1.2).
- A clock may contain asymmetric delays in its timestamping mechanism or protocol path. If these asymmetries are not negligible, they must be correctly accounted for (see 6.2.4.9).
- Network delay between master and slave on a subnet must be constant over a PTP_DELAY_REQ_INTERVAL.
- Boundary clocks must be used to synchronize across subnets (see 6.2.1).
- Delay fluctuation due to network components and due to the protocol stack within clocks must be reduced by two techniques:
    1) The timestamps used in PTP are generated as close to the physical layer as practical for a given clock implementation. In cases where the most accurate timestamps can be generated only after a message has actually been transmitted, the actual value is communicated in the Follow_Up message from the master clock (see 8.4.1 and Annex A). See [B4], [B7], [B8] for mechanisms to aid in generating these timestamps.
    2) Remaining delay fluctuation introduced by the protocol stack and by network components not isolated by a boundary clock can be reduced by averaging (see A.5.2). The averaging algorithms are outside the scope of this standard.

— The computing power of clocks implementing the protocol must be great enough, and the number of clocks per subnet must be small enough, to meet the constraints of 7.11.

— The inherent stability of a clock's oscillator must be adequate (see A.4 and A.5.4).

6.2 PTP systems

A diagram of a typical distributed system of nodes containing clocks is shown in Figure 2.

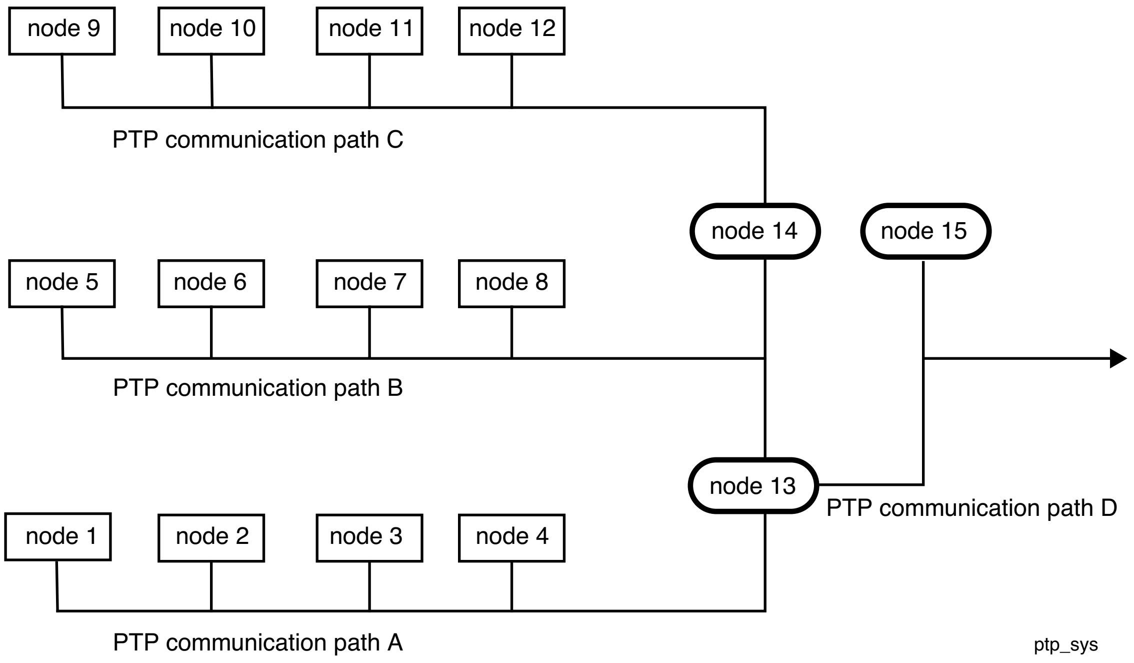

Figure 2—Typical system of synchronized clocks

In Figure 2, the rectangles represent nodes each containing a clock; rounded rectangles represent nodes each containing a boundary clock. These clocks communicate PTP messages with each other via the communication paths illustrated by the solid lines in the figure.

6.2.1 Clock types

There are two types of clocks in a PTP system:

— *Ordinary clocks:* Ordinary clocks shall communicate with other clocks over a single communication path. For example in Figure 2, clocks in nodes 1 through 4 communicate over a single communication path, path A.

— *Boundary clocks:* Boundary clocks, such as node 13 in Figure 2, may communicate with multiple sets of clocks (for example, node set 1, 2, 3, and 4 via A, set 5, 6, 7, 8, and 14 via B, and 15 via D). Each set may contain a mixture of ordinary and boundary clocks. A boundary clock shall communicate with each set using a distinct communication path. Communication between clocks (for example, 9 and 5) in these disjoint sets shall only occur via the boundary clock mechanisms specified in the protocol. Boundary clock to boundary clock communication is illustrated in Figure 2 by path B between boundary clocks 13 and 14 and path D between 13 and 15.

The logical communication access point of a clock to a PTP communication path is termed a PTP port. Ordinary clocks have a single PTP port while boundary clocks have two or more PTP ports.

6.2.2 PTP communications

6.2.2.1 PTP communication topology

The operation of the PTP protocol generates a topology of PTP communication paths forming an acyclic graph structure. That is, there will be no alternate PTP communication paths between any pair of PTP clocks. An example of prohibited path topology is the cyclic path that includes nodes 13, 14, and 15 in Figure 3. Again, rectangles represent nodes each containing a clock; rounded rectangles represent nodes each containing a boundary clock.

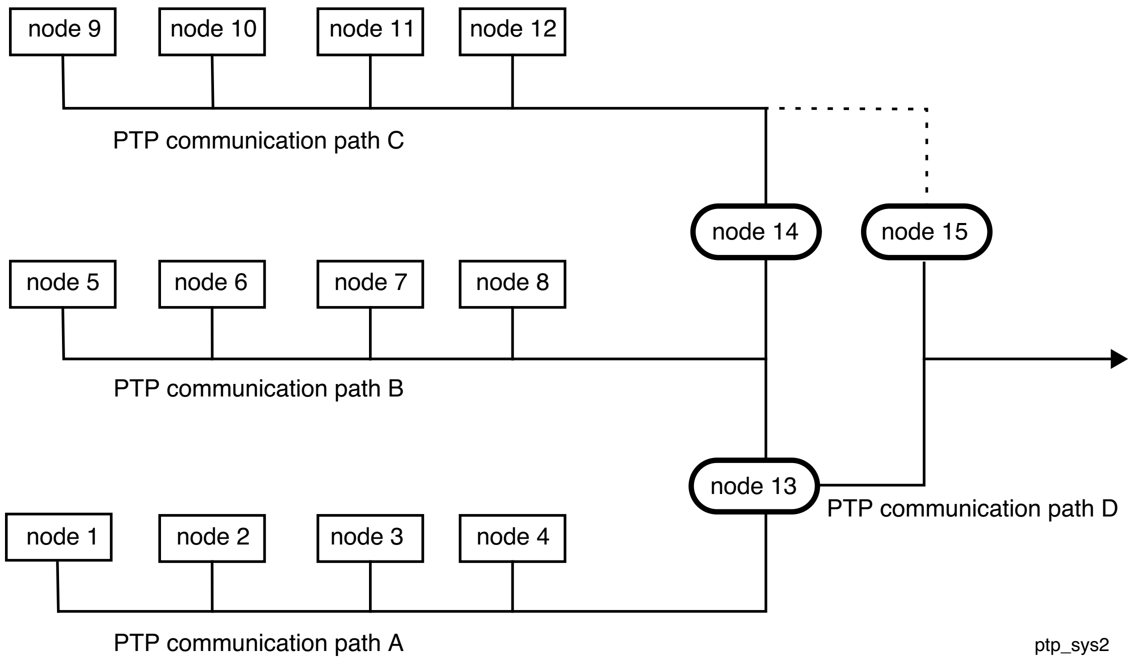

Figure 3—Prohibited topology

The PTP protocol detects such cyclic PTP communication paths. The PTP protocol changes cyclic graphs into acyclic graphs by changing the state of ports on the involved boundary clocks. In this example, the protocol has disabled the communication between nodes 14 and 15.[4] These state changes result in actual communication being over an acyclic topology even though the physical connection topology is cyclic.

The operation of the PTP protocol to produce this topology follows techniques described by Perlman [B2].

There is no requirement that all of the PTP communication paths use the same underlying communication media or technology. However, if different communication media or technologies are present, a set of clocks communicating with each other using a given communication medium or technology shall be separated from a set using a different communication medium or technology by a boundary clock.

[4]More precisely, it has put the appropriate port on node 14 or 15 into the PTP_PASSIVE state. Which node is put into that state depends on details of the network configuration and properties of the clocks and cannot be determined in this simplified example.

Boundary clocks separate PTP subdomains into distinct PTP communication paths. Communications involving boundary clocks shall be restricted as follows.

— PTP messages of type Sync, Delay_Req, Follow_Up, and Delay_Resp shall not be propagated between PTP communication paths separated by a boundary clock. This restriction applies to boundary clocks and any communication technology specific components enabling ordinary communication between devices on the separate PTP communication paths.

— PTP Management messages shall be communicated between PTP communication paths separated by a boundary clock. These PTP Management messages shall be communicated by the boundary clocks but shall not be communicated by any communication technology specific components enabling ordinary communication between devices on the separate PTP communication paths. The absolute value of the boundaryHops field of the management message shall be decremented by 1 by the boundary clock before retransmitting the message. If the boundaryHops field value is zero, the management message shall not be retransmitted. If the boundaryHops field value is positive the message shall be retransmitted only via boundary clock ports not in the PTP_INITIALIZING, PTP_FAULTY, or PTP_DISABLED states. If the boundaryHops field value is negative the message shall be retransmitted only via boundary clock ports not in the PTP_INITIALIZING state. In no case shall the message be retransmitted on the port on which it was received. Ordinary clocks shall not retransmit management messages.

  NOTE—Management messages will be retransmitted by clock ports in the PTP_PASSIVE state and may thus be delivered to some clocks more than once (see Figure 8). Management applications should take this fact into account.

6.2.2.2 PTP communication protocol

All PTP non-management messages shall be communicated as multicast communications. A multicast communication results in a single transmission of a PTP non-management message being received by all clocks and only those clocks sharing a PTP communication path. Management messages may be communicated using either a multicast or a point-to-point communication. All clocks shall be capable of receiving Management messages via multicast communication.

It is recommended that all implementations use a true multicast implementation as the actual communication protocol for PTP multicast communications. The behavior of systems building multicast out of point-to-point communications will:

— Not scale well as the number of clocks increases

— Require more complex logic in clocks to implement the requirements of the protocol

PTP communications take place in PTP domains characterized by a subdomain name and a port category (see 6.2.3, 6.2.5.1, and 6.2.5.3). Each message contains this information and each clock maintains values of these quantities that pertain to it. A clock shall accept and process a message if and only if the subdomain name and port category associated with the message are identical to the corresponding values pertaining to the receiving clock.

6.2.2.3 PTP message model, message timestamp point, and clock timestamp point

PTP communications are implemented with PTP messages. In all communication technologies, PTP communication paths are modeled to contain distinguishing locations known as the *clock timestamp points,* and certain messages are modeled to contain a distinguishing feature known as the *message timestamp point* (see 6.2.4.9). The precise definition of these terms shall be defined in this standard or its revisions for each network protocol supporting PTP. All PTP time stamps shall refer to the time at which the message timestamp point of either a Sync or a Delay_Req message passes a clock timestamp point. This time shall be corrected for latency (see 7.8.1.3).

6.2.3 PTP domains

A PTP domain, hereafter referred to as simply a *domain,* is a set of one or more PTP subdomains. A PTP subdomain, hereafter referred to as simply a *subdomain,* consists of one or more clocks communicating with each other, as defined by the PTP protocol, in order to synchronize these clocks. Except for certain PTP Management messages, the nodes in one subdomain shall not communicate with the nodes in another subdomain for purposes related to PTP. Multiple subdomains may be used to create independent collections of synchronized clocks sharing common PTP communication paths. The clocks within a subdomain will synchronize with each other, but there is no requirement that the clocks in one subdomain be synchronized with the clocks in another subdomain. The purpose of subdomains is to allow users to create localized sets of clocks, typically sharing a single communication path, that maintain a time base independent of the rest of the domain.

There are four subdomains defined by this standard. These subdomains shall be:

— DefaultPTPdomain: This shall be the default subdomain if a domain consists of only a single subdomain. It may be a subdomain in a domain consisting of one or more subdomains. Unless modified by a PTP Management message or clock-specific means outside the scope of this standard (e.g., a switch on the clock), the clock shall use this subdomain.
— AlternatePTPdomain1: This may be a subdomain in a domain consisting of one or more subdomains.
— AlternatePTPdomain2: This may be a subdomain in a domain consisting of one or more subdomains.
— AlternatePTPdomain3: This may be a subdomain in a domain consisting of one or more subdomains.

A domain may contain subdomains, termed *optional subdomains,* other than those given previously. Such optional subdomains shall implement all other requirements of this standard. The administration of domains containing optional subdomains is beyond the scope of this standard.

Examples:

— In Figure 2, if boundary clock 14 and its connections are removed, there are two PTP domains. One consists of the set 9, 10, 11, and 12, while the other consists of the set 1, 2, 3, 4, 5, 6, 7, 8, 13, and 15.
— If nodes 14 and 15 are present as shown, then there is a single PTP domain consisting of all nodes.
— If nodes 14 and 15 are present and nodes 1, 2, and 3 are designated as belonging to AlternatePTPdomain1 while the remaining nodes belong to the DefaultPTPdomain, then the system consists of two subdomains.

There is no requirement that a subdomain be implemented in a single communication medium or technology. However, by 6.2.2, if more than a single communication medium or technology exists within a subdomain, the clocks in the subdomain must consist of two or more disjoint sets, each with its own PTP communication path, communicating via one or more boundary clocks.

Boundary clocks shall be capable of implementing all aspects of the PTP protocol for the DefaultPTPdomain. Boundary clocks may in addition be capable of implementing the PTP protocol for additional PTP subdomains.

Within a subdomain a master clock, in addition to issuing a Sync message, may issue an external timing signal to any or all of clocks within the subdomain having ports in the PTP_SLAVE state, *slave clocks*. This external timing signal shall be transmitted on a medium other than the medium communicating the Sync messages. This external timing signal shall be issued on the seconds' transition of the local clock. Slave clocks receiving the external timing signals may use this information to synchronize to the master. It is recommended that slave clocks be configured to accept external timing signals only from a master clock sharing the PTP communication path with the slave clock. Because the external timing signal uses communication media other than the media used for PTP communications, such configuration is outside of the scope of this standard. The behavior of systems allowing the external timing signal to be delivered to clocks outside of the subdomain or not on the same PTP communication path of the issuing master clock is outside the scope of this standard.

6.2.4 Clock properties

The clocks in a PTP system are characterized as defined in the following subclauses.

6.2.4.1 UUID

Each clock and each port in a PTP system shall have an identifier, called a UUID. A port's identifier is termed the *Port-UUID,* and a clock's identifier is termed the *Clock-UUID*. UUIDs shall be implemented as three fields, as follows:

— communication_technology_field: This field shall specify the communication media and technology used in PTP communications of the port. The values shall be selected from the CommunicationId enumeration.

— uuid_field: This field shall consist of an OctetArray of length PTP_UUID_LENGTH. The value of this field for a given clock shall be unique within the communication technology defined by the communication_technology_field value. The value consisting of all zeros for this field shall be reserved in all technologies as a special value used by the PTP protocol. Technologies for which the length of the unique identifier is less than PTP_UUID_LENGTH shall left-justify the identifier in the field and pad the unused octets with zeroes. That is, the most significant octet of uuid_field shall be significant in all technologies. UUIDs longer than PTP_UUID_LENGTH shall not be permitted.

— port_id_field: The datatype of this field shall be UInteger16. The value of this field for a port on an ordinary clock shall be 1. The value of this field for the N ports on a boundary clock shall have the values 1, 2, …N. The value of this field shall be zero when used as part of the Clock-UUID. Note that the type of this field limits the number of ports on a boundary clock to 65 535.

There is no requirement on the in-memory layout of these fields. Their on-the-wire layout for Ethernet is given in D.1, and the algorithm for comparing two UUIDs is given later in this clause.

The default value of any UUID shall have the following field values:

— communication_technology_field: PTP_DEFAULT

— uuid_field: Each octet shall have all bits zero.

— port_id_field: Zero

The three fields of a Port-UUID shall be selected such that no two ports in a PTP domain have the same Port-UUID.

For an ordinary clock, the Clock-UUID shall be the Port-UUID of its single PTP port, but with a port_id_field value of zero.

For a boundary clock C, the Clock-UUID shall be constructed as follows:

— Find the smallest Port-UUID of C's ports, using the comparison algorithm of this clause (note that these ports may have different communication_technology_fields). Call this smallest Port-UUID S.

— C's Clock-UUID shall consist of:

   1) A communication_technology_field equal to that of S
   2) A uuid_field equal to that of S
   3) A port_id_field of zero

It is recommended that all clocks in a PTP domain using a given technology use the same specification for determining the Clock-UUID. The behavior of systems where this recommendation is not followed is outside the scope of this standard.

It is expected that the organization responsible for standardizing each network technology will provide a method for choosing a uuid_field meeting the requirements of this clause. The method for Ethernet is given in D.2. If a UUID meeting the requirements of this clause is not specified by the organization responsible for standardizing a network technology, then the specification of the uuid_field shall be as specified in D.2, and the communication_technology_field shall be PTP_ETHER.

IDL: enumeration CommunicationId;

Table 2—CommunicationId enumeration

Enumeration	Value	Communication protocol
PTP_CLOSED	0	Closed system outside the scope of this standard
PTP_ETHER	1	IEEE 802.3™ (Ethernet)[a]
reserved	2	Reserved
reserved	3	Reserved
PTP_FFBUS	4	FOUNDATION fieldbus
PTP_PROFIBUS	5	PROFIBUS
PTP_LON	6	LonTalk® protocol[b]
PTP_DNET	7	DeviceNet
PTP_SDS	8	SmartDistributedSystem™ [c]
PTP_CONTROLNET	9	ControlNet™ [d]
PTP_CANOPEN	10	CANopen
reserved	11 – 242	Reserved
PTP_IEEE1394	243	IEEE 1394™
PTP_IEEE802.11A	244	IEEE 802.11a™
PTP_IEEE_WIRELESS	245	IEEE 802.11b™
PTP_INFINIBAND	246	InfiniBand™ [e]
PTP_BLUETOOTH	247	Bluetooth™ wireless[f]
PTP_IEEE802.15.1	248	IEEE 802.15.1™
PTP_IEEE1451.3	249	IEEE 1451.3™
PTP_ IEEE1451.5	250	IEEE 1451.5™
PTP_USB	251	USB bus
PTP_ISA	252	ISA bus
PTP_PCI	253	PCI bus
PTP_VXI	254	VXI bus
PTP_DEFAULT	255	Default value

[a]The IEEE products referred to in Table 2 are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.
[b]LonTalk is a registered trademark of Echelon Corporation.
[c]SmartDistributedSystem is a trademark owned by Honeywell, Inc.
[d]ControlNet is a trademark owned by ControlNet International, Ltd.
[e]InfiniBand is a trademark owned by the InfiniBand Trade Association.
[f]Bluetooth is a trademark owned by Bluetooth SIG, Inc.

The following algorithm shall define the ordering properties of all UUIDs:

— communication_technology_field and port_id_field values shall be treated as unsigned integers.
— uuid_fields shall be compared as follows:
  1) Let X and Y be two uuid_fields.
  2) If every octet in X is equal to the corresponding octet in Y, $X = Y$.
  3) Otherwise, consider the most significant position in which the octets differ, and treat the octets in that position as unsigned integers. If the octet belonging to X is smaller than the octet belonging to Y, $X < Y$; otherwise $X > Y$.

Now, given UUIDs A and B:

— If the communication_technology_field of A is less than the communication_technology_field of B, $A < B$.
— Otherwise, if the communication_technology_field of A is greater than the communication_technology_field of B, $A > B$.
— Otherwise, if the uuid_field of A is greater than the uuid_field of B, $A > B$.
— Otherwise, if the uuid_field of A is less than the uuid_field of B, $A < B$.
— Otherwise, if the port_id_field of A is greater than the port_id_field of B, $A > B$.
— Otherwise, if the port_id_field of A is less than the port_id_field of B, $A < B$.
— Otherwise, $A = B$.

6.2.4.2 Clock type

Each clock shall be typed as either an ordinary clock or a boundary clock (see 7.4.2.11).

6.2.4.3 Clock stratum

The clock stratum, or stratum number, describes one measure of the quality of a clock. Each clock shall be characterized by a stratum number to be used by the best master clock algorithm as one parameter of clock quality (see 7.4.2.4).

The interpretation and allowed values of stratum numbers shall be as defined in Table 3. The clock stratum shall have type UInteger8.

Unless a clock is specifically designed to maintain clock accuracy in the face of power loss, a power cycle on the clock shall preclude assigning stratum numbers less than 3 on powerup.

If the inherent characteristics of a clock degrade such that the clock stratum number and clock identifier designations no longer apply, the clock shall either:

— Degrade its stratum numbers and clock identifiers in such a way as to correctly specify the current clock characteristics, or
— Be placed in the PTP_FAULTY state.

If the inherent characteristics of a clock improve such that the clock stratum and clock identifier designations no longer apply, the clock may upgrade its stratum numbers and clock identifiers. Such upgrades shall remain consistent with all requirements of this standard. Improved characteristics may result from a variety of causes. For example, the clock becomes directly (not via PTP) synchronized to a primary reference clock that is traceable to a recognized standard source of time.

Table 3—Stratum number definitions

Stratum number	Specification
0	May be used temporarily for special purposes by PTP implementations to force a clock to be deemed better than other clocks in the system.
1	Designates the clock as a primary reference standard traceable to a recognized standard source of time. A stratum 1 clock may be either a boundary clock or an ordinary clock. (NOTE—GPS clocks, calibrated atomic clocks, etc. fall into this stratum). A stratum 1 clock shall not be synchronized using the PTP protocol to another clock in a PTP system.
2	Designates the clock as a secondary standard reference clock. The clock shall be: — Directly (not via PTP) synchronized to a stratum 1 clock or another source deemed to be a correct source of time for the PTP subdomain or — Previously directly synchronized to a stratum 1 clock or another source deemed to be a correct source of time for the PTP subdomain and is still providing time information consistent with this clock or source as specified by the clock_identifier associated with the clock (see 6.2.4.5).
3	The lowest possible clock_stratum value if not 1 or 2 for a clock that is capable of issuing external timing signals and possibly setting the PTP_EXT_SYNC flag to TRUE (see 8.2.10).
4	The lowest possible clock_stratum value if not 1 or 2 for a clock that does not have the capability of issuing external timing signals and therefore sets the PTP_EXT_SYNC flag to FALSE (see 8.2.10).
5–254	Reserved.
255	The default value. A clock with this stratum number shall never be the best master clock.

For boundary clocks, the clock stratum number shall be identical for each PTP port.

While not required, it is recommended that in a given PTP subdomain clocks with clock stratum numbers less than 4 be inherently at least as stable (low default variance) as any clock with a greater clock stratum number.

6.2.4.4 Preferred master clock

A clock may be administratively designated as part of a *preferred master clock set*. This creates a set of clocks that will be favored over those not so designated in the selection of master clocks within a subdomain. The purpose of this designation is to allow users to specify a clock that will remain master in the presence of disconnection or connection of other clocks.

6.2.4.5 Clock identifier

The clock_identifier shall indicate the nature and expected absolute accuracy and epoch of a given clock. The datatype of the clock_identifier shall be an OctetArray of length PTP_CODE_STRING_LENGTH. Each octet of the field shall be interpreted as an ASCII character, with the most significant octet the leading character. The interpretation and allowed values of clock identifiers shall be in accordance with Table 4. For clock identifiers of less than PTP_CODE_STRING_LENGTH characters, the clock_identifier is left justified (toward the most significant octet) in the field, with the unused octets having value \0, (hex value 0x0). Clock identifiers shall be used to establish which of several clocks with identical clock stratum numbers is selected as the best master clock.

Clock identifiers indicating that the time scale is universal coordinated time (UTC) (see B.2) may become invalid for clocks with clock stratum values of 2 or greater under the following conditions:

- The possible drift of the clock since last synchronized to a UTC source of time exceeds the given specification, or
- The time interval since last synchronized to a UTC source of time includes times when possible leap second adjustments occur (see Annex B), or
- A fault potentially degrading these specifications occurs.

The ordering of clock identifiers used in the selection of the best master clock (see 7.6) is summarized in Table 5.

Unless specifically designed to maintain clock accuracy, a power cycle on a clock shall preclude clock identifiers other than DFLT on powerup.

For boundary clocks, the clock identifier shall be the same for each PTP port.

In Table 4, the listed accuracy specifications shall be interpreted as the sum of the mean and standard deviation from the applicable time base.

Table 4—Clock identifier definitions

Clock identifier (ASCII)	Applicable to clock_stratum number	Specification
ATOM	1	Time is derived from a calibrated atomic clock maintaining a UTC time base accurate to better than 25 ns
GPS	1	Time is derived from a correctly operating GPS receiver maintaining a UTC time base accurate to better than 100 ns.
ATOM	2	The stability of the clock is such that it is accurate to within 100 ns of the UTC time base established the last time it was synchronized directly to a stratum 1 clock with clock_identifier ATOM. A power cycle may preclude this designation.
GPS	2	The stability of the clock is such that it is accurate to within 100 ns of the UTC time base established the last time it was synchronized directly to a stratum 1 clock with clock_identifier GPS. A power cycle may preclude this designation.
NTP	2	The clock shall meet one of the following specifications: — The clock is correctly and actively participating in a suite of clocks using the NTP or equivalent protocol to maintain a UTC time base accurate to better than 15 ms, or — The stability of the clock is such that it is consistent to within 50ms of the time base established the last time it was correctly and actively participating in a suite of clocks using the NTP or equivalent protocol to maintain time consistent with UTC. A power cycle may preclude this designation. Examples of protocols providing time bases and accuracies equivalent to NTP are SNTP, link to NIST time server, etc.
HAND	2 or greater	The clock has been set to the correct UTC time to accuracy better than 10 seconds by an administrative procedure and is consistent with that time except for normal drift of this clock. A power cycle may preclude this designation.
INIT	2 or greater	The clock has been set with unspecified accuracy to an arbitrary or user-defined time by an administrative procedure and is consistent with that time except for normal drift of this clock. A power cycle may preclude this designation.
DFLT	3 or greater	Applicable if none of the other clock_identifiers apply.

Table 5—Clock identifier precedence

Stratum	Clock identifier ordering
1	ATOM equivalent to GPS
2	ATOM equivalent to GPS takes precedence over NTP takes precedence over HAND takes precedence over INIT.
3 or greater	HAND takes precedence over INIT takes precedence over DFLT.

6.2.4.6 Follow_Up capability

Each clock shall be designated as either supporting or not supporting the capability of providing, for each Sync message issued, a Follow_Up message conveying a more accurate value for the originTimestamp, namely the preciseOriginTimestamp (see 8.3.1.2 and 8.4.1.3).

For boundary clocks, this capability shall be separately specified for each PTP port potentially used by the clock. It is recommended that all PTP ports for a boundary clock carry the same specifications.

This designation shall be encoded into the flags field of all Sync or Delay_Req messages issued by the clock (see 8.2.10) as follows:

— If the PTP port supports the issuing of Follow_Up messages, the PTP_ASSIST flag shall be TRUE.
— Otherwise the PTP_ASSIST flag shall be FALSE.

6.2.4.7 State

For each ordinary clock and for each PTP port of a boundary clock, the state machine of 7.3 shall be supported.

6.2.4.8 Variance

Two variance estimates, as specified in 7.7, characterize clocks in a PTP system:

— Each clock shall maintain an estimate, the clock_variance (see 7.4.2.6) of its inherent stability properties (that is, its stability when it is not synchronized to another PTP clock using the PTP protocol).
— If a clock is synchronized to another using the PTP protocol, it may maintain an estimate, the observed_variance (see 7.4.4.9) of the stability properties of the clock to which it is synchronized.

6.2.4.9 Latency

Each PTP port shall be characterized by two constants known as outbound_latency and inbound_latency, used in the delay correction process specified in 7.8.1.3. In a clock, Sync and Delay_Req messages shall be assigned a timestamp on both issuance and receipt. These timestamps are instantiated at a point, the clock timestamp point, in the path between the code executing the PTP protocol and the communication media, as shown in Figure 4.

The outbound_latency constant shall be the propagation time between the clock timestamp point and the communication medium for outbound Sync or Delay_Req messages. The inbound_latency constant shall be the propagation time between the communication medium and the clock timestamp point for inbound Sync or Delay_Req messages.

The variance in these values contributes to the variance of the clock and shall be included in the computed clock variance (see 7.7).

Note that in general, the values of outbound_latency and inbound_latency will not be identical. For both inbound and outbound Sync and Delay_Req messages, timestamps are generated at the instant the message timestamp point passes the corresponding clock timestamp point.

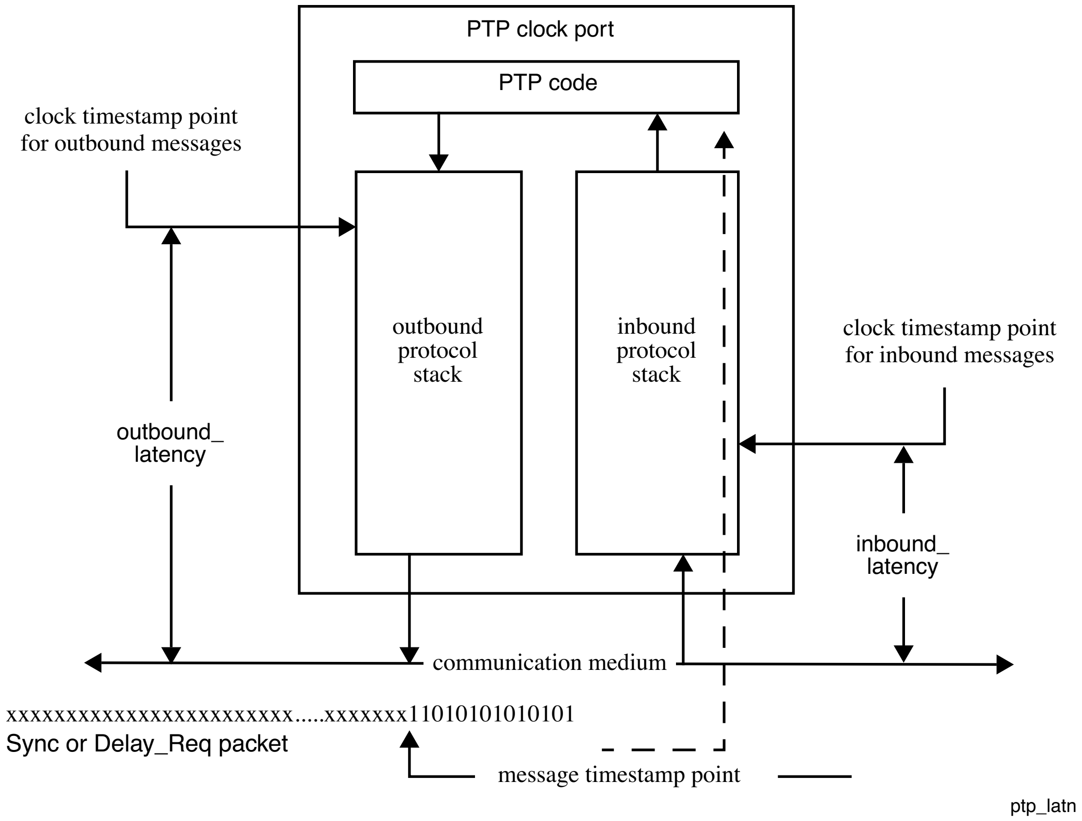

Figure 4—Definition of latency constants

The message timestamp point shall be a distinguished feature of both Sync and Delay_Req messages that can be recognized as each passes the clock timestamp point. Figure 4 shows a typical Sync or Delay_Req message entering the inbound protocol stack (dashed arrow) such that the message timestamp point (in the example the first "11" after a series of leading "10" patterns) passes the clock timestamp point inbound_latency seconds after leaving the communication medium and entering the bottom of the stack. The mechanisms for generating these timestamps are outside the scope of this standard.

Clause 6.2.2.3 requires that all timestamps reflect the time at which a message timestamp point passes a clock timestamp point. If an implementation detects Sync or Delay_Req messages at a point other than the message timestamp point, then the generated timestamps shall be appropriately corrected by the time interval between the time of detection and the time the message timestamp point passed a clock timestamp point.

6.2.5 PTP subdomain properties

A PTP subdomain is characterized as defined in the following subclauses.

6.2.5.1 Subdomain name

Each subdomain shall be characterized by a name recognized by all PTP clocks in the subdomain as the basis for sending and receiving PTP communications. The subdomains shall be as defined in 6.2.3. The combination of the subdomain name and the port category shall specify the end points for a PTP communication. All PTP clocks in the subdomain shall use this name as the basis for all PTP communications.

The value of subdomain name member shall be of type OctetArray [PTP_SUBDOMAIN_NAME_LENGTH]. The actual value shall be restricted as follows:

- The default values of the octets shall correspond to the hex values of the five ASCII characters _DFLT followed by (PTP_SUBDOMAIN_NAME_LENGTH – 5) null characters, 0x00. This shall result in the clock operating in the subdomain DefaultPTPdomain (see 6.2.3).
- The values of the octets corresponding to the hex values of the five ASCII characters _ALT1, _ALT2, and _ALT3 followed by (PTP_SUBDOMAIN_NAME_LENGTH – 5) null characters, 0x00, may be used and shall result in the clock operating in the subdomains AlternatePTPdomain1, AlternatePTPdomain2, and AlternatePTPdomain3 respectively (see 6.2.3).
- Other values of the octets may be chosen from the set consisting of the values of the printable ASCII characters starting with hex value 0x21 (!) up to and including hex value 0x7E (~). If the number of desired characters for a subdomain name is less than PTP_SUBDOMAIN_NAME_LENGTH, the desired characters shall be augmented by appending, in the least significant octets, sufficient value 0x00 octets to bring the total to PTP_SUBDOMAIN_NAME_LENGTH. The selection of such a subdomain name shall result in the clock operating in one of the domains AlternatePTPdomain1, AlternatePTPdomain2, or AlternatePTPdomain3 or any optionally named subdomains that exist in a PTP domain (see 6.2.3). The mapping of these octets onto the subdomain addresses shall be as specified in Annex C.

Modification of this member's value shall not take effect until PTP enters the Initialization state. Subclause 7.4, Note 1 applies.

Two subdomain names shall match if and only if an octet-by-octet comparison for all PTP_SUBDOMAIN_NAME_LENGTH octets results in a match for all comparisons.

6.2.5.2 Subdomain address

Each subdomain_name shall have a corresponding subdomain address. The representation of this address and the implementation of PTP communication based on this address are communication technology specific.

6.2.5.3 Port category

PTP communications shall be characterized by two port categories recognized by all PTP clocks in the subdomain as the basis for specifying the contents of PTP communications. The names shall be:

- EventPort: This name shall designate the access point for communication of PTP Sync or Delay_Req messages.
- GeneralPort: This name shall designate the access point for communication of PTP Follow_Up, Delay_Resp and Management messages.

The combination of the subdomain name and the port category shall specify the end points for a PTP communication. All PTP clocks shall use these names as the basis for all PTP communications.

6.2.5.4 Port address

Each communication technology shall have an address corresponding to each port category. The representation of these addresses and the implementation of PTP communication based on these addresses are communication technology specific.

6.2.5.5 Sync interval

The sync interval shall be the interval in seconds between successive Sync messages issued by master clocks. It shall have the same value for all clocks in a subdomain.

The value of sync interval is a compromise between the inherent stability of the clocks, the responsiveness of the clocks in a subdomain to change, and the communication load imposed by PTP.

The values of sync interval shall be taken from the set {1, 2, 8, 16, and 64 seconds}. Specific communication technologies may designate a subset of these values for use in that technology. Within a technology, all clocks shall be capable of correct PTP operation in a system with any of these designated sync intervals.

The behavior of PTP subdomains where clocks do not all have the same value of sync interval is outside the scope of this standard.

6.2.5.6 Epoch

The epoch is the origin of the timescale supported in a subdomain. The timescale of a subdomain shall be measured in cumulative seconds and nanoseconds since the epoch. Since all times within a subdomain are ultimately derived from the grandmaster clock in the subdomain, the subdomain epoch is the epoch of the grandmaster clock.

The clock_identifier of the grandmaster clock shall specify the value of the epoch as follows:

Clock identifier (ASCII)	Epoch specification
ATOM	PTP epoch
GPS	PTP epoch
NTP	PTP epoch
HAND	PTP epoch
INIT	The epoch shall be as defined by the initialization procedure.
DFLT	The epoch is unknown.

Where the epoch is the PTP epoch, the cumulative seconds and nanoseconds supported by the PTP protocol may be converted into UTC current month, day, year, time of day format by utilizing the current_utc_offset field value of the global time properties data set. The PTP epoch began at 0 hours on 1 January 1970. See Annex B for information on how to convert times measured from the PTP epoch to UTC and other commonly used time scales.

6.2.5.7 Epoch number

When the epoch is the PTP epoch, the value of the epoch number shall be the current number of times the 32-bit seconds clock has rolled over since the PTP epoch. For any epoch, the epoch number may be treated as the most significant part of the total number of seconds since the epoch where the least significant part is the 32-bit integer seconds portion of the TimeRepresentation of PTP timestamps.

7. PTP protocol specification

7.1 Protocol model of a clock

The model of an ordinary clock and of a boundary clock used in the definition of the protocol is illustrated in Figure 5 and Figure 6, respectively.

Figure 5 illustrates an ordinary clock. This clock is presumably one of several using a communication path implemented using communication technology *q*. The clock has two access points to the communication path modeled by the event and general ports with their associated addresses, together termed the *PTP port* or more simply, *port*. This port communicates with the protocol engine of the clock. The protocol engine in turn has access to port-specific information as well as the general information about the clock. The protocol engine can also read the clock's current time as well as adjust the time behavior of the local clock.

For all clocks, both ordinary and boundary:

— Each PTP port of the clock is modeled to have an index represented by the port_id_field. For ordinary clocks that have only a single port, the port_id_field shall have the value 1. For boundary clocks, each port has a distinct port_id_field with the ports numbered sequentially starting with port_id_field =1.
— The *local clock* of Figure 5 and Figure 6 is assigned an *internal* port_id_field value of 0.
— All clocks have at most one port in the PTP_SLAVE state during normal operation. The local clock shall be synchronized based on the information received on this port from the clock recognized as the master clock visible from that port. In the absence of a port in the PTP_SLAVE state, the local clock shall not synchronize to any other clock via the PTP protocol. It may synchronize to a local oscillator or to another clock, usually a UTC time source such as GPS via means outside the scope of this standard.
— Each clock maintains the data sets shown.

Figure 6 illustrates a boundary clock having several PTP ports {*p*}, each port being one of several using communication paths implemented using communication technologies {*q*}. Each PTP port is implemented as in an ordinary clock with the following exceptions:

— A separate set of clock port configuration information and foreign master information is maintained for each PTP port.
— A set of configuration information is maintained for the boundary clock itself. Any PTP port may access this data. If each port maintains a local copy of the clock datasets, implementations shall ensure that all such copies are consistent whenever protocol decisions are made at any port.
— All ports can read a single, common local clock.

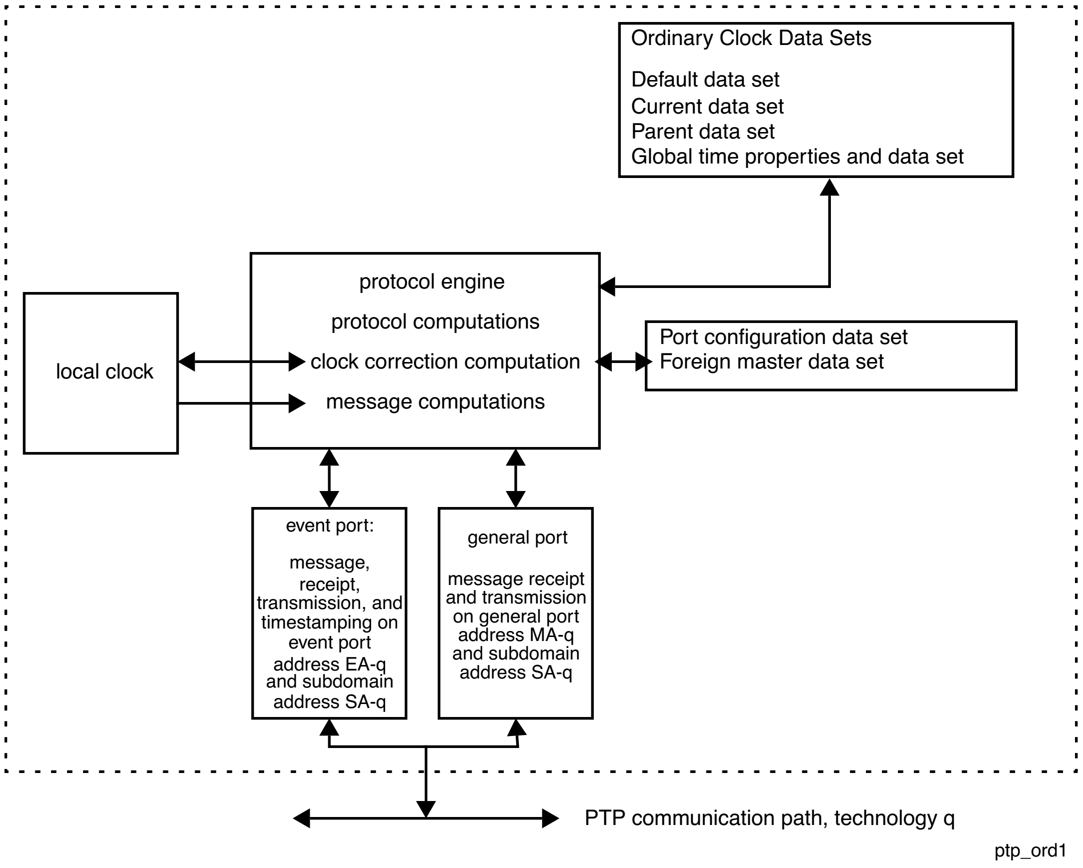
Ordinary Clock Data Sets
Default data set
Current data set
Parent data set
Global time properties and data set
protocol engine
protocol computations
clock correction computation
message computations
local clock
Port configuration data set
Foreign master data set
event port:
message, receipt, transmission, and timestamping on event port address EA-q and subdomain address SA-q
general port
message receipt and transmission on general port address MA-q and subdomain address SA-q
PTP communication path, technology q
ptp_ord1

Figure 5—Ordinary clock protocol model

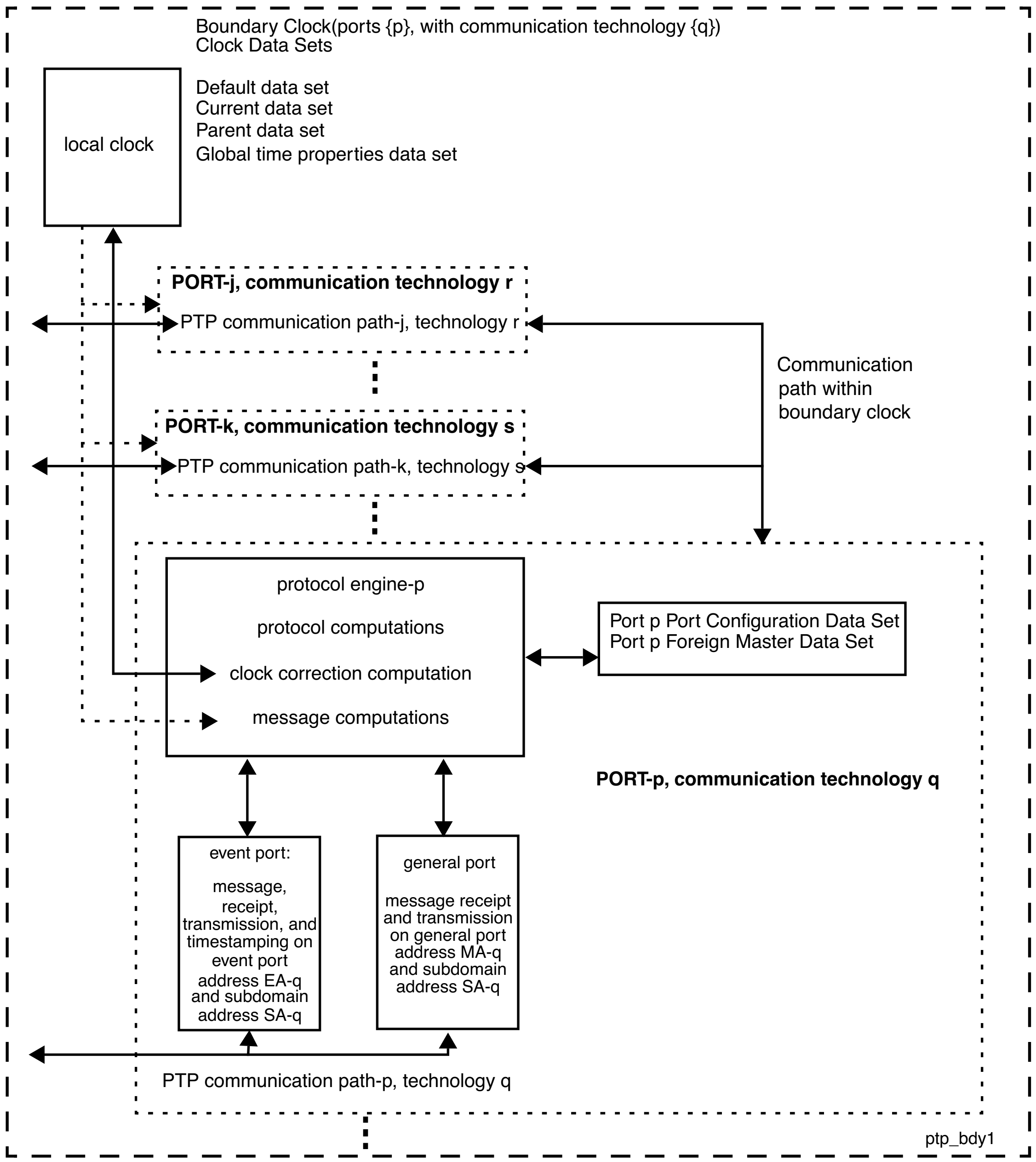
Boundary Clock(ports {p}, with communication technology {q})
Clock Data Sets
Default data set
Current data set
Parent data set
Global time properties data set
local clock
PORT-j, communication technology r
PTP communication path-j, technology r
Communication path within boundary clock
PORT-k, communication technology s
PTP communication path-k, technology s
protocol engine-p
protocol computations
clock correction computation
message computations
Port p Port Configuration Data Set
Port p Foreign Master Data Set
PORT-p, communication technology q
event port: message, receipt, transmission, and timestamping on event port address EA-q and subdomain address SA-q
general port message receipt and transmission on general port address MA-q and subdomain address SA-q
PTP communication path-p, technology q
ptp_bdy1

Figure 6—Boundary clock protocol model

7.2 Protocol model of a subdomain of PTP clocks

In a subdomain, the PTP protocol recognizes clocks with five categories of ports. Excluded from this discussion are clock ports in the PTP_INITIALIZING, PTP_DISABLED, or PTP_FAULTY states. These categories are as follows:

— Slave ports: These may be the ports of ordinary clocks or the external view of a single PTP port of a boundary clock that synchronize to a particular master port. A master port to which a slave is synchronized is termed the *slave's parent clock port*. A slave port must be in the PTP_SLAVE state specified in 7.3.

- — Master ports: These may be the ports of ordinary clocks or the external view of a single PTP port of a boundary clock that serve as the parent clock port to zero or more slave ports sharing a PTP communication path with the master. A master port must be in the PTP_MASTER or PTP_PRE_MASTER states specified in 7.3.
- — Grandmaster ports: These may be the ports of an ordinary clock or the external view of all PTP master ports of a boundary clock. There is a single set of grandmaster ports, all on the same clock in a subdomain. This set serves as the master clock ports to zero or more slave ports sharing the PTP subdomain with the grandmaster. A grandmaster port must be in the PTP_MASTER state specified in 7.3.
- — Uncalibrated ports: These may be the ports of an ordinary clock or the external view of a single PTP port of a boundary clock. These are ports for which the protocol has not yet assigned a master or designated the port as a master port. An uncalibrated port must be in the PTP_LISTENING or PTP_UNCALIBRATED state specified in 7.3.
- — Passive ports: These may be the ports of an ordinary clock or the external view of a single PTP port of a boundary clock. These ports may be associated with a stratum 1 or 2 clock that has detected the presence of other stratum 1 or 2 clocks in the system. One of the other stratum 1 or 2 clocks has been designated by the protocol as the master clock. Alternatively, a port may be designated as passive by the protocol to avoid cyclic topology. A passive port must be in the PTP_PASSIVE state specified in 7.3.

Every stable subdomain forms a parent-child hierarchy of clock ports. The root of this hierarchy is termed the *grandmaster clock*. At each branch point (necessarily a boundary clock), a clock port must be the parent and master port for all clock ports on the branch further removed from the root and have a slave port synchronizing to the next clock closer to the root. A port, not the root, at the extreme of any branch of the hierarchy must be a slave or passive port. A branch port must be a boundary clock. A stable subdomain is a subdomain in which all ports participating in PTP have been designated by the protocol as either a master, passive, or a slave and a single grandmaster clock has been designated.

For subdomains containing multiple stratum 1 or 2 clocks, the protocol will segment the subdomain into multiple disjoint parent-child hierarchies, each containing a single stratum 1 or 2 clock.

Examples of various subdomain master-slave configurations are illustrated in Figure 7 and Figure 8. In both figures, the state of each clock port is indicated by a M, S, or P for master, slave, and passive, respectively.

Figure 7 illustrates a subdomain in which one of the ordinary clocks, node-5, is designated as the grandmaster clock. In this case, all of the other clock ports will form an acyclic parent-child relationship with node-5 as the root.

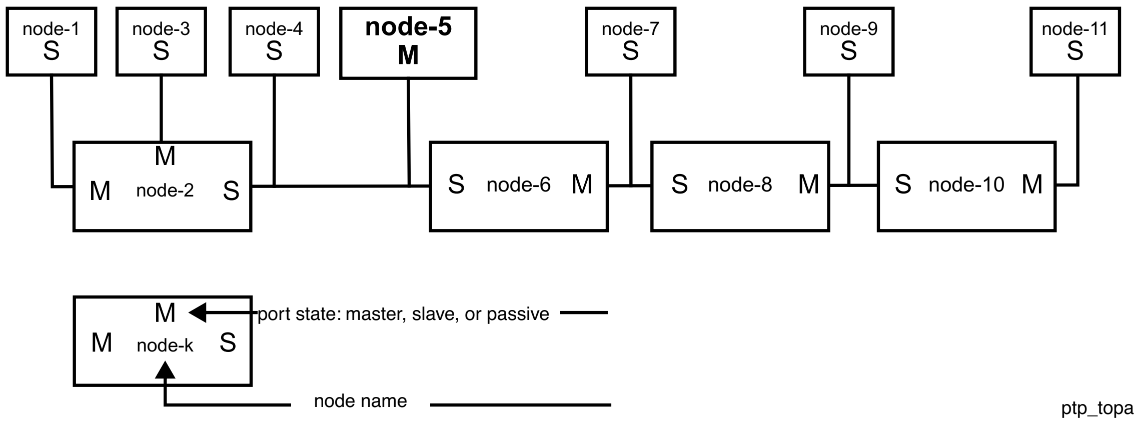

Figure 7—Master-slave configuration, example 1

Figure 8 illustrates a subdomain in which two of the clocks; C-1 and BC-9 are both stratum 1 clocks. In this case the system will segment into two disjoint subdomains, one with C-1 as grandmaster the other with BC-9 as grandmaster, as shown. Within each of the subdomains, the clock ports form an acyclic parent-child hierarchy. Where multiple paths occur, all but one of the paths will be cut by a port in the passive state, for example, between BC-11 and BC-12.

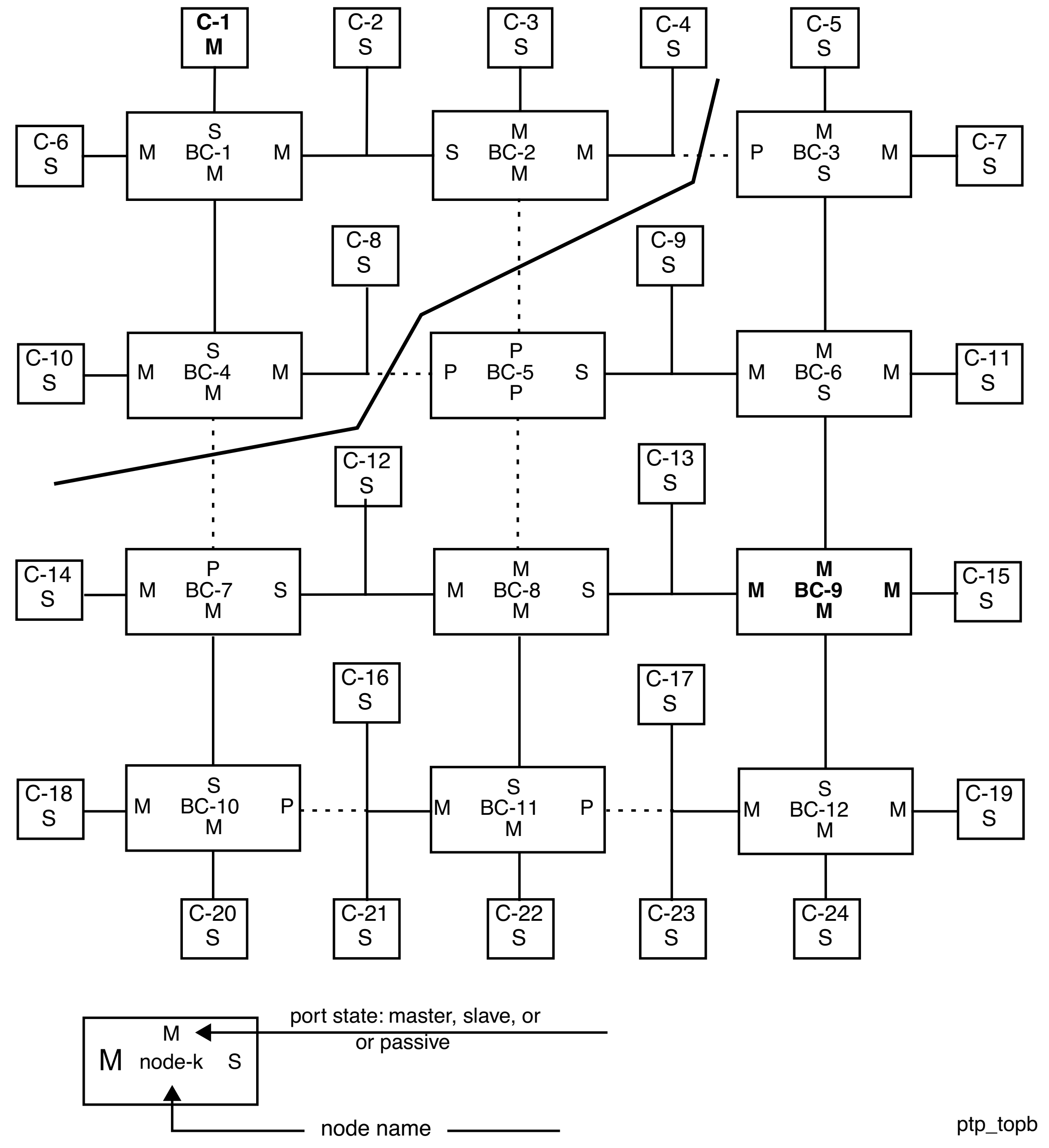

Figure 8—Master-slave configuration, example 2

The allowed possibilities for ordinary clocks and for each PTP port of a boundary clock are summarized in Table 6. The clock numbers refer to clocks in Figure 8 as an example.

Table 6—Allowed combinations of port categories

Port State	Ordinary clock	Boundary clock port
Grandmaster (and therefore a master and potentially a parent)	Possible (C-1)	Possible only if no port is a slave (BC-9)
Master (and therefore potentially a parent)	Possible and if so it is also a grandmaster (C-1)	Possible if one (BC-6) or no (BC-9) port is a slave.
Slave	Possible, (C-10)	Possible only for a single port, (all but BC-9)
Uncalibrated	Possible (not shown)	Possible (not shown)
Passive	Possible (not shown)	Possible (BC-11)

The protocol determines which of the ports in a system are masters, slaves, uncalibrated, passive, and which clock is the grandmaster. The transition from one stable configuration to another is governed by the protocol rules. Such transitions may be caused by the introduction or removal of a clock or the change in the properties, such as the clock identifier of one of the clocks. The addition or removal of a connection to a boundary clock may respectively join two subdomains or segment a single subdomain, thereby causing a transition between two stable configurations.

7.3 State behavior of clocks

This subclause describes the state machine which ordinary clocks and boundary clocks shall implement.

7.3.1 Protocol engine state machine

The protocol engine of each ordinary clock or each PTP port of a boundary clock shall implement the state machine illustrated in Figure 9 (see 4.3 for a key to interpreting the figure).

When a fault is cleared, the engine makes a transition to the PTP_INITIALIZING state. Before making the transition to PTP_LISTENING, an implementation is not required to perform all the steps that it would after an INITIALIZE command or a powerup or reset. It is required only to achieve the *effect* of performing those steps.

The behavior of the seven major states of a port associated with the protocol engine shall be as defined in the PTPState enumeration.

IDL: enumeration PTPState;

Table 7—PTP State enumeration

State enumeration	Description
PTP_INITIALIZING	The associated port may initialize the data sets, hardware, and communication properties of the clock.
PTP_FAULTY	The fault state of the protocol. The associated port shall not participate in the synchronization aspects of the protocol but may take implementation-specific measures to clear the fault.
PTP_DISABLED	The associated port shall not place any messages on its communication path. In a boundary clock, no activity at the associated port shall be allowed to affect the activity at any other port of the boundary clock.
PTP_LISTENING	The associated port is waiting for the Sync message receipt timeout to expire or to receive a Sync message from a master. The purpose of this state is to allow orderly addition of clocks to a subdomain.
PTP_PRE_MASTER	The associated port shall behave in all respects as though it were in the PTP_MASTER state except that it shall not place any non-Management messages on its communication path.
PTP_MASTER	The associated port is behaving as a master port. The local clock is used to timestamp the receipt or departure of messages associated with the port.
PTP_PASSIVE	The associated port shall not place any messages on its communication path unless otherwise specified.
PTP_UNCALIBRATED	One or more master ports have been detected in the subdomain. The appropriate master port is being selected and the local port is preparing to synchronize to the selected master port. This is a transient state to allow initialization of synchronization servos, updating of data sets when a new master port has been selected, and other implementation-specific activity.
PTP_SLAVE	The associated port shall synchronize to the selected master port.

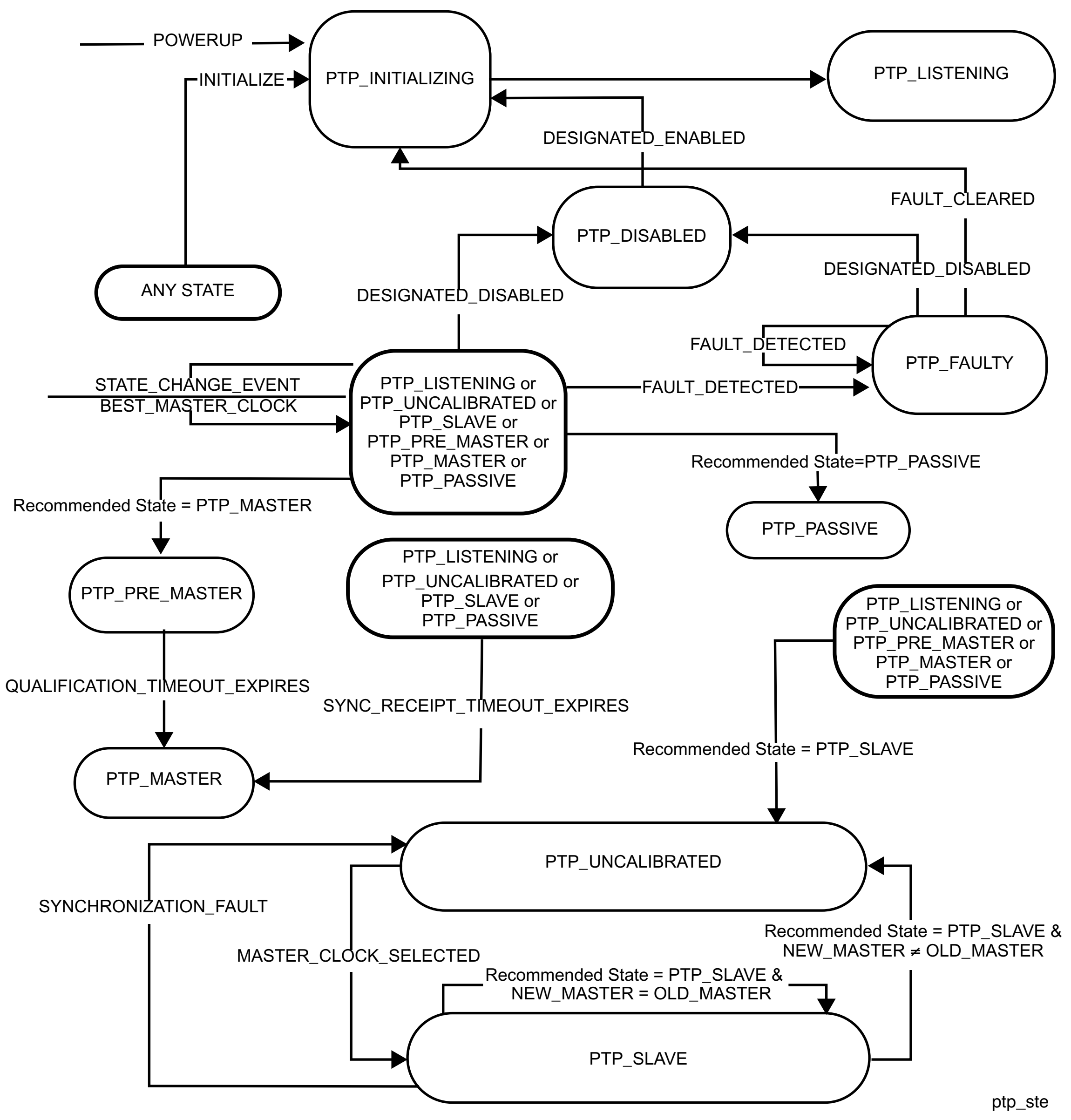
POWERUP
INITIALIZE
PTP_INITIALIZING
PTP_LISTENING
DESIGNATED_ENABLED
FAULT_CLEARED
PTP_DISABLED
DESIGNATED_DISABLED
ANY STATE
DESIGNATED_DISABLED
FAULT_DETECTED
PTP_FAULTY
STATE_CHANGE_EVENT
BEST_MASTER_CLOCK
PTP_LISTENING or PTP_UNCALIBRATED or PTP_SLAVE or PTP_PRE_MASTER or PTP_MASTER or PTP_PASSIVE
FAULT_DETECTED
Recommended State=PTP_PASSIVE
Recommended State = PTP_MASTER
PTP_PASSIVE
PTP_PRE_MASTER
PTP_LISTENING or PTP_UNCALIBRATED or PTP_SLAVE or PTP_PASSIVE
PTP_LISTENING or PTP_UNCALIBRATED or PTP_PRE_MASTER or PTP_MASTER or PTP_PASSIVE
QUALIFICATION_TIMEOUT_EXPIRES
SYNC_RECEIPT_TIMEOUT_EXPIRES
Recommended State = PTP_SLAVE
PTP_MASTER
PTP_UNCALIBRATED
SYNCHRONIZATION_FAULT
Recommended State = PTP_SLAVE & NEW_MASTER ≠ OLD_MASTER
MASTER_CLOCK_SELECTED
Recommended State = PTP_SLAVE & NEW_MASTER = OLD_MASTER
PTP_SLAVE
ptp_ste

Figure 9—Protocol engine state machine

7.3.2 State machine constraints for boundary clocks

Boundary clocks shall be implemented such that the requirements of this clause are met independently for each subdomain.

It is recommended that only the DefaultPTPdomain include boundary clocks.

A boundary clock consists of a single local clock and N PTP ports.

Each PTP port of a boundary clock shall have its own value for state. This state shall be governed by a copy of the state machine of Figure 9 associated with each port.

Only the combinations of states and behaviors for the PTP ports of a boundary clock shown in Table 8 shall be allowed in a subdomain. For purposes of this clause, an active port is a port not in the PTP_INITIALIZING, PTP_FAULTY, PTP_DISABLED, or PTP_LISTENING states.

Table 8—Boundary clock state constraints

Port state	Constraints on a boundary clock with N ports
PTP_INITIALIZING	If one port is in this state, all ports shall be in this state.
PTP_FAULTY	A port in this state is considered inactive. No activity on this port shall affect the active ports of the boundary clock or of any other port in the subdomain. If fault activity on a port in this state cannot be confined to the port, then all ports shall be in this state.
PTP_DISABLED	A port in this state is considered inactive. No activity on this port shall affect the active ports of the boundary clock or of any other port in the subdomain.
PTP_LISTENING	Any or all ports may be in this state.
PTP_PRE_MASTER	Same as for PTP_MASTER, except that the port shall not place any non-management messages on its associated communication path.
PTP_MASTER	If one port is in this state either: — No active port is in the PTP_SLAVE state and the boundary clock is also the grandmaster clock of the subdomain for all communication paths for which the boundary clock communicates via a port in the PTP_MASTER state, or — One active port is in the PTP_SLAVE state and the grandmaster clock of the subdomain is a clock communicating with the boundary clock directly or indirectly via the slave port. The boundary clock will be the master (but not the grandmaster) for all communication paths for which the boundary clock communicates via a port in the PTP_MASTER state.
PTP_PASSIVE	Any or all ports may be in this state.
PTP_UNCALIBRATED	Any port may be in this state.
PTP_SLAVE	At most, one port shall be in this state. The local clock of the boundary clock shall be synchronized to the master port in the communication path associated with the port in the PTP_SLAVE state.

The events shown Figure 9 shall be applied to the state machine's ports of the boundary clock as specified in Table 9.

Table 9—State machine event applicability

Event name	Applicability
POWERUP	State machines of all ports.
INITIALIZE	State machine of all ports.
FAULT_DETECTED	State machine of all ports affected by the fault.
FAULT_CLEARED	State machine of all ports affected by the fault.
STATE_CHANGE_EVENT	State machine associated with the port signaling the event. The information may affect all state machines due to other constraints of this clause.
BEST_MASTER_CLOCK	The execution of the best master clock algorithm associated with a port directly affects only that port. The execution potentially affects the states of other ports if required by the other constraints of this clause.
SYNC_RECEIPT_TIMEOUT_EXPIRES QUALIFICATION_TIMEOUT_EXPIRES	State machine associated with the expiring timeout mechanism.
DESIGNATED_ENABLED DESIGNATED_DISABLED	State machine specified by the initiating management message.
MASTER_CLOCK_SELECTED	State machine executing the best master clock algorithm making this decision.
SYNCHRONIZATION_FAULT	State machine associated with the port experiencing the fault.

7.4 Clock data set

For each clock, the following clock data sets specified in the subclauses of this clause shall be maintained locally as the basis for protocol decisions and for providing values for message fields:

— Default data set
— Current data set
— Parent data set
— Global time properties data set
— Port configuration data set (one data set for each port of a boundary clock)
— Foreign master data set (one data set for each port of a boundary clock)

Each data set member specification includes the following:

— The formal name for the member
— Semantics associated with the member
— The value assigned at initialization

In these data sets, the following notes apply:

NOTES

1—These values may change infrequently as a result of management messages or degradation of the clock implementation, for example, degradation in the stability of the local clock over time or changes in subdomain topology.

2—These values change frequently during the normal operation of the protocol.

3—This information does not directly affect the PTP protocol but may be used by services built on top of the protocol. This information is typically derived either from management messages or from a stratum 1 clock in the system, for example, a clock synchronized to a GPS receiver or participating in the NTP protocol with other computers with access to standard UTC time sources.

7.4.1 Clock data set initialization properties

There are two kinds of members for the clock data sets: fixed and modifiable.

The values of fixed members are inherent properties of the clock. These members shall have the values as specified in the defining subclause.

The modifiable members may change as a result of management messages, changes in internal properties of the clock resulting, for example, from local synchronization to a standard clock, parametric changes due to aging or temperature, or fault detection and recovery.

Modifiable members shall be initialized to one of the following values:

— The values indicated in the member specifications as initialization values. If no initialization value is specified, the default value for the specified datatype shall be used as the initialization value. This set of values is designated the specification initialization set.
— A set of values established by implementation-specific means such as management messages, non-volatile storage of current operating values, etc. If this option is implemented, the management message PTP_MM_INITIALIZE_CLOCK shall be implemented (see 7.12.4), and the value of the member, initializable, of the default data set shall be TRUE (see 7.4.2.9).

Modifiable members shall be initialized:

— As a result of the initialization event indicated in the state diagram Figure 9.
— As a result of the powerup event indicated in the state diagram Figure 9.
— As a result of the management message PTP_MM_INITIALIZE_CLOCK (see 7.12.4).
— As specified in other clauses of this standard.

7.4.2 Default data set

This data set defines inherent or assumed properties of the local clock. These values are used when the local clock becomes the grandmaster clock in a subdomain. They depend on the specific source of time for the local clock when it is the grandmaster.

The possibilities for the source of time to a grandmaster clock include:

— Directly synchronized (not via PTP) to an atomic clock or a GPS receiver
— Directly synchronized (not via PTP) to a clock participating in NTP
— A local oscillator

The members of this data set are:

— clock_communication_technology
— clock_uuid_field
— clock_port_field
— clock_stratum
— clock_identifier
— clock_variance
— clock_followup_capable
— preferred

— initializable
— external_timing
— is_boundary_clock
— sync_interval
— subdomain_name
— number_ports
— number_foreign_records

7.4.2.1 clock_communication_technology

This member's value shall be the communication_technology_field of the Clock-UUID of the local clock. The datatype shall be as specified in 6.2.4.1. This is a static property of the local clock and shall not be modifiable.

7.4.2.2 clock_uuid_field

This member's value shall be the uuid_field of the Clock-UUID of the local clock. The datatype shall be as specified in 6.2.4.1. This is a static property of the local clock and shall not be modifiable.

7.4.2.3 clock_port_field

This member's value and datatype shall be as specified in 6.2.4.1. This is a static property of the local clock and shall not be modifiable.

7.4.2.4 clock_stratum

This member's value shall characterize those aspects of the accuracy defined in 6.2.4.3. The datatype shall be as specified in 6.2.4.3. Depending on the implementation of the local clock, this value may or may not be modifiable.

7.4.2.5 clock_identifier

This member's value shall be the clock identifier of the local clock when it is the grandmaster clock of the system. The allowed initialization values depend on the value of the member clock_stratum and on the nature of the hardware implementation (see 6.2.4.5). The datatype shall be as specified in 6.2.4.5. Depending on the implementation of the local clock this value may or may not be modifiable. Subclause 7.4, Note 1 applies.

7.4.2.6 clock_variance

This member's value shall be the variance of the local clock when it is the grandmaster clock of the system (see 6.2.4.8). The datatype and computation of variances shall be as specified in 7.7. The initialization value shall reflect the inherent characteristics of the clock. Depending on the implementation of the local clock this value may or may not be modifiable. Subclause 7.4, Note 1 applies.

7.4.2.7 clock_followup_capable

This member's value shall indicate whether the clock is capable of providing Follow_Up messages. The datatype shall be Boolean. The value shall be TRUE if the local clock supports the issuing of Follow_Up messages per 6.2.4.6, and FALSE otherwise. This is a static property of the local clock and shall not be modifiable.

7.4.2.8 preferred

This member's value shall indicate whether the clock is to be preferred in the selection of the grandmaster clock. The datatype shall be Boolean. The value shall be TRUE if the local clock is a member of the preferred master clock set and FALSE if it is not a member of this set (see 6.2.4.4). This value shall be modifiable by the user. The default value shall be TRUE if the clock clock_stratum is 1 or 2 and FALSE otherwise. Subclause 7.4, Note 1 applies.

7.4.2.9 initializable

This member's value shall indicate whether the receipt of a management message with a managementMessageKey field value PTP_MM_INITIALIZE_CLOCK (see 7.12.4) causes the clock to execute the behavior normally executed on a reboot or power cycle of the node containing the clock (see 7.3). The datatype shall be Boolean. The value shall be TRUE if the receipt of a management message with a managementMessageKey field value PTP_MM_INITIALIZE_CLOCK (see 7.12.4) causes the clock to execute the behavior normally executed on a reboot or power cycle of the node containing the clock (see 7.3). Otherwise, the value shall be FALSE. This is a static property of the local clock and shall not be modifiable.

7.4.2.10 external_timing

This member's value shall indicate whether the clock is capable of sending external timing signals. The datatype shall be Boolean. The value shall be TRUE if the clock is capable of sending external timing signals when it has a port in the master state. Otherwise, the value shall be FALSE. This is a static property of the local clock and shall not be modifiable.

7.4.2.11 is_boundary_clock

This member's value shall indicate whether the clock is a boundary clock. The datatype shall be Boolean. The value shall be TRUE if the clock is a boundary clock. Otherwise, the value shall be FALSE. This is a static property of the local clock and shall not be modifiable.

7.4.2.12 sync_interval

This member's value shall specify the current sync interval (see 6.2.5.5). The datatype shall be Integer8. The value shall be the logarithm base 2 of the current sync interval in seconds. This value may be defined administratively by means of a management message. The default sync interval shall be two seconds, that is, sync_interval = 1, irrespective of the communication technology. The sync_interval value shall be the default value unless changed by an administrative procedure.

7.4.2.13 subdomain_name

This member's value shall specify the subdomain name. The datatype shall be as specified in 6.2.5.1. This value may be defined administratively by means of a management message. This member's initialization value may be the name of the last PTP subdomain defined administratively. In the absence of prior modification by means of a management message or by implementation-specific reset means, the default shall be the default value defined in 6.2.5.1.

7.4.2.14 number_ports

This member's value shall specify the number of ports on the device. For an ordinary clock this shall be the value 1. The datatype shall be UInteger16. This is a static property of the local clock and shall not be modifiable.

7.4.2.15 number_foreign_records

This member's value shall specify the maximum number of records maintained in the foreign master data set of each port. The datatype shall be UInteger16. This is a static property of the local clock and shall not be modifiable.

7.4.3 Current data set

This data set defines members whose values characterize the current properties of the local clock that describe the source and quality of the local time. These values depend on the specific source of time for the local clock and whether it is the grandmaster. Various possibilities for the source of time are (not an exhaustive list):

— Locally synchronized (not via PTP) to an atomic clock, a GPS receiver
— Locally synchronized (not via PTP) to a clock participating in NTP
— A local oscillator
— Another PTP clock in the subdomain

The members of this data set are:

— steps_removed
— offset_from_master
— one_way_delay

7.4.3.1 steps_removed

This member's value shall be the number of communication paths traversed between the local clock and the grandmaster clock. For example, steps_removed in a slave clock on the same PTP communication path as the grandmaster clock will have a value of 1, indicating that a single path was traversed. The datatype shall be UInteger16.

7.4.3.2 offset_from_master

When the local clock is slaved to another PTP clock using the PTP protocol, this member's value shall be the local clock's estimate of the current time offset between the local clock and the PTP clock to which is directly synchronized. If the local clock is not slaved to another PTP clock using the PTP protocol, the value shall be 0. This value shall be computed as specified in 7.8.1.1. The datatype shall be TimeRepresentation. Subclause 7.4, Note 2 applies.

7.4.3.3 one_way_delay

When the local clock is slaved to another PTP clock using the PTP protocol, this member's value shall be the local clock's estimate of the current one-way propagation delay between the local clock and the PTP clock to which it is synchronized via PTP. Otherwise, the value shall be 0. This value shall be computed as specified in 7.8.1.2. The datatype shall be TimeRepresentation. Subclause 7.4, Note 2 applies.

7.4.4 Parent data set

This data set defines quantities whose values characterize the current properties of the PTP port, the parent or parent port, serving as the source for the time for a PTP slave port via the PTP protocol.

The members of this data set are:

- parent_communication_technology
- parent_uuid
- parent_port_id
- parent_last_sync_sequence_number
- parent_followup_capable
- parent_external_timing
- parent_variance
- parent_stats
- observed_variance
- observed_drift
- utc_reasonable
- grandmaster_communication_technology
- grandmaster_uuid_field
- grandmaster_port_id_field
- grandmaster_stratum
- grandmaster_identifier
- grandmaster_variance
- grandmaster_preferred
- grandmaster_is_boundary_clock
- grandmaster_sequence_number

The fields described in 7.4.4.9 through 7.4.4.11 are intended for use in detecting a false-ticking master (see A.4). A clock may compute these. If it does, it shall compute all of them. If it does not, these values shall be set to their defaults.

7.4.4.1 parent_communication_technology

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the source-CommunicationTechnology field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_communication_technology of the default data set. The initialization value shall be the value of the member clock_communication_technology of the default data set. The datatype shall be as specified in 6.2.4.1. Subclause 7.4, Note 1 applies.

7.4.4.2 parent_uuid

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the source-Uuid field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_uuid_field of the default data set. The initialization value shall be the value of the member clock_uuid_field of the default data set. The datatype shall be as specified in 6.2.4.1. Subclause 7.4, Note 1 applies.

7.4.4.3 parent_port_id

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the source-PortId field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_port_field of the default data set. The initialization value shall be the value of the member clock_port_field of the default data set. The datatype shall be as specified in 6.2.4.1. Subclause 7.4, Note 1 applies.

7.4.4.4 parent_last_sync_sequence_number

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the sequenceId field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be 0. The datatype shall be UInteger16. Subclause 7.4, Note 2 applies.

7.4.4.5 parent_followup_capable

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the PTP_ ASSIST bit of the flags field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of member clock_ followup_capable of the default data set. The datatype shall be Boolean. Subclause 7.4, Note 1 applies.

7.4.4.6 parent_external_timing

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the PTP_ EXT_SYNC bit of the flags field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of member external_ timing of the default data set. The datatype shall be Boolean. Subclause 7.4, Note 1 applies.

7.4.4.7 parent_variance

When the port is in the PTP_SLAVE state, this member's value shall be the value of the localClockVariance field of the last Sync message received from the parent of the local port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_variance of the default data set. The datatype shall be as specified in 7.7. Subclause 7.4, Note 1 applies.

7.4.4.8 parent_stats

If all of the following are true:

— The clock has a port in the PTP_SLAVE state.
— The clock has computed statistically valid estimates of its parent's variance and drift rate.
— Either the clock has determined whether its parent's UTC information is reasonable (see 7.4.4.9 through 7.4.4.11), or the time scale of the clock's parent is not UTC (see 6.2.5.6).

then the value of this field shall be TRUE. Otherwise the value shall be FALSE. The data type shall be boolean. Subclause 7.4, Note 3 applies.

How a clock makes the estimates, determines their validity, and decides the reasonableness of the UTC information is outside the scope of this standard.

7.4.4.9 observed_variance

If the value of the parent_stats field in the Parent data set is FALSE, the value of this field shall be 0. Otherwise, this member's value shall be an estimate of the parent clock's variance as observed by the slave clock, computed and represented as described in 7.7. The datatype shall be as specified in 7.7. Subclause 7.4, Note 3 applies.

7.4.4.10 observed_drift

If the value of the parent_stats field in the Parent data set is FALSE, the value of this field shall be 0. Otherwise, this member's value shall be an estimate of the parent clock's drift rate as observed by the slave clock, in nanoseconds per second. If the estimate exceeds the capacity of its datatype, this value shall be set to the largest or smallest allowable value, as appropriate. The datatype shall be Integer32. Subclause 7.4, Note 3 applies.

7.4.4.11 utc_reasonable

If the value of the parent_stats field in the Parent data set is TRUE and all of the following values are determined by the slave clock to be reasonable, the value of this field shall be TRUE:

— Value of the currentUTCOffset field of the last Sync message received from the parent of the local clock
— Values of the PTP_LI_59 and PTP_LI_61 bits of the flags field of the last Sync message received from the parent of the local clock
— Value of the epochNumber field of the last Sync message received from the parent of the local clock

Otherwise, the value of this field shall be FALSE. The datatype shall be Boolean. Subclause 7.4, Note 3 applies.

7.4.4.12 grandmaster_communication_technology

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterCommunicationTechnology field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_communication_technology of the default data set. The datatype shall be as specified in 6.2.4.1. The initialization value shall be 255. Subclause 7.4, Note 1 applies.

7.4.4.13 grandmaster_uuid_field

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterClockUuid field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_uuid_field of the default data set. The datatype shall be as specified in 6.2.4.1. Subclause 7.4, Note 1 applies.

7.4.4.14 grandmaster_port_id_field

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterPortId field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_port_field of the default data set. The datatype shall be as specified in 6.2.4.1. Clause 7.4, Note 1 applies.

7.4.4.15 grandmaster_stratum

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterClockStratum field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_stratum of the default data set. The datatype shall be as specified in 6.2.4.3. The initialization value shall be 255. Subclause 7.4, Note 1 applies.

7.4.4.16 grandmaster_identifier

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterClockIdentifier field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_identifier of the default data set. The datatype shall be as specified in 6.2.4.5. The initialization value shall be the clock identifier DFLT (see 6.2.4.5). Subclause 7.4, Note 1 applies.

7.4.4.17 grandmaster_variance

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterClockVariance field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member clock_variance of the default database. The datatype shall be as specified in 7.7. The initialization value shall be 0. Subclause 7.4, Note 1 applies.

7.4.4.18 grandmaster_preferred

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterIsPreferred field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member preferred of the default data set. The datatype shall be Boolean. Subclause 7.4, Note 1 applies.

7.4.4.19 grandmaster_is_boundary_clock

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterIsBoundaryClock field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the value of the member is_boundary_clock of the default data set. The datatype shall be Boolean. Subclause 7.4, Note 1 applies.

7.4.4.20 grandmaster_sequence_number

When the clock has a port in the PTP_SLAVE state, this member's value shall be the value of the grandmasterSequenceId field of the last Sync message received from the parent of the slave port. When the local clock is the grandmaster clock of a subdomain, this value shall be the last_sync_event_sequence_number of the port configuration data set for the port with the lowest value of port_id_field that is also in the PTP_MASTER state. Note that a clock is the grandmaster clock of a subdomain if it has no ports in the PTP_SLAVE state (see 7.2). The datatype shall be UInteger16. Subclause 7.4, Note 1 applies.

7.4.5 Global time properties data set

This data set defines members needed to interpret the time provided in PTP messages.

The members of this data set are:

— current_utc_offset
— leap_59
— leap_61
— epoch_number

7.4.5.1 current_utc_offset

This member's value shall be as specified in Table 10. The datatype shall be Integer16. Subclause 7.4, Note 3 applies.

Table 10—current_utc_offset values

Condition	current_utc_offset value
Ordinary clock in PTP_SLAVE state or Boundary clock with one port in PTP_SLAVE state	Value of the currentUTCOffset field of the last Sync message received from the parent of the local clock
The local clock is of stratum 1 or 2, or has a clock_identifier = HAND	The value shall be the offset in seconds between the UTC and TAI time scales, i.e., the current number of leap seconds. (See Annex B)
No clock port is in PTP_SLAVE state and the local clock is not of stratum 1 or 2, nor has a clock_identifier = HAND	0
Otherwise	0

7.4.5.2 leap_59

This member's value shall be as specified in Table 11. The datatype shall be Boolean. Subclause 7.4, Note 3 applies.

Table 11—Current leap_59 values

Condition	Current leap_59 value
Ordinary clock in PTP_SLAVE state or Boundary clock with one port in PTP_SLAVE state	Value of the PTP_LI_59 bit of the flags field of the last Sync message received from the parent of the local clock
The local clock is of stratum 1 or 2, or has a clock_identifier = HAND	The value shall be TRUE when the last minute of the current day will contain 59 seconds. (See Annex B).
No clock port is in PTP_SLAVE state and the local clock is not of stratum 1 or 2, nor has a clock_identifier = HAND	FALSE
Otherwise	FALSE

7.4.5.3 leap_61

This member's value shall be as specified in Table 12.The datatype shall be Boolean. Subclause 7.4, note 3 applies.

7.4.5.4 epoch_number

This member's value shall be as specified in Table 13.The datatype shall be UInteger16. Subclause 7.4, Note 3 applies.

Table 12—Current leap_61 values

Condition	Current leap_61 value
Ordinary clock in PTP_SLAVE state or Boundary clock with one port in PTP_SLAVE state	Value of the PTP_LI_61 bit of the flags field of the last Sync message received from the parent of the local clock
The local clock is of stratum 1 or 2, or has a clock_identifier = HAND	The value shall be TRUE when the last minute of the current day will contain 61 seconds. (See Annex B)
No clock port is in PTP_SLAVE state and the local clock is not of stratum 1 or 2, nor has a clock_identifier = HAND	FALSE
Otherwise	FALSE

Table 13—Current epoch_number values

Condition	Current epoch_number value
Ordinary clock in PTP_SLAVE state or Boundary clock with one port in PTP_SLAVE state	Value of the epochNumber field of the last Sync message received from the parent of the local clock
The local clock is of stratum 1 or 2, or has a clock_identifier = HAND	The value shall be as defined in 6.2.5.7
No clock port is in PTP_SLAVE state and the local clock is not of stratum 1 or 2, nor has a clock_identifier = HAND	0 unless another value is known to be correct for the time scale in use
Otherwise	0

7.4.6 Port configuration data set

For the single port of an ordinary clock and for each port of a boundary clock, the following *port configuration data set* shall be maintained as the basis for protocol decisions and providing values for message fields. The number of such records shall be the value of number_ports of the default data set.

The members of this data set are:

- port_state
- last_sync_event_sequence_number
- last_general_event_sequence_number
- subdomain_address
- event_port_address
- general_port_address
- port_communication_technology
- port_uuid_field
- port_id_field
- burst_enabled

7.4.6.1 port_state

This member's value shall be the value of the current state of the protocol engine associated with this port (see 7.3). The datatype shall be UInteger8. The initialization value shall be PTP_INITIALIZING. Subclause 7.4, Note 2 applies.

7.4.6.2 last_sync_event_sequence_number

This member's value shall be the value of the sequenceId field of the last PTP Sync or Delay_Req message sent from this port. The datatype shall be UInteger16. Subclause 7.4, Note 2 applies.

7.4.6.3 last_general_event_sequence_number

This member's value shall be the value of the sequenceId field of the last PTP Delay_Resp, Follow_Up or Management message sent from this port. The datatype shall be UInteger16. Subclause 7.4, Note 2 applies.

7.4.6.4 subdomain_address

This member's value shall be the communication technology specific address (see 6.2.5.2) corresponding to the value of the member subdomain_name of the default data set (see 7.4.2.13). Modification of this member's value shall not take effect until the clock enters the PTP_INITIALIZING state. The datatype shall be OctetArray[*k*] where *k* is the length in octets of the representation in the communication technology of the port.

7.4.6.5 event_port_address

This member's value shall be the communication technology specific port address (see 6.2.5.4) for event ports. This is a static property of the local clock and shall not be modifiable. The datatype shall be OctetArray[*k*] where *k* is the length in octets of the representation in the communication technology of the port.

7.4.6.6 general_port_address

This member's value shall be the communication technology specific port address (see 6.2.5.4) for general ports. This is a static property of the local clock and shall not be modifiable. The datatype shall be OctetArray[*k*] where *k* is the length in octets of the representation in the communication technology of the port.

7.4.6.7 port_communication_technology

This member's value is the communication_technology_field of the Port-UUID of the local port (see 6.2.4.1). This is a static property of the local clock and shall not be modifiable. The datatype shall be as specified in 6.2.4.1.

7.4.6.8 port_uuid_field

This member's value is the uuid_field of the Port-UUID of the local port (see 6.2.4.1). This is a static property of the local clock and shall not be modifiable. The datatype shall be as specified in 6.2.4.1.

7.4.6.9 port_id_field

This member's value is the port_id_field of the Port-UUID of the local port (see 6.2.4.1). This is a static property of the local clock and shall not be modifiable. The datatype shall be as specified in 6.2.4.1.

7.4.6.10 burst_enabled

This member's value shall indicate whether the clock is capable of requesting and providing a burst of Sync messages (see 7.5.5). The datatype shall be Boolean. The default value shall be FALSE.

7.4.7 Foreign master data set

For the single port of an ordinary clock and for each port of a boundary clock, the following *foreign master data set* shall be maintained as the basis for protocol decisions and providing values for message fields. The number of such data sets shall be the value of number_ports of the default data set.

This data set is used to qualify messages from foreign master clocks received by the port. A separate data record consisting of the members specified in this clause shall be maintained for each foreign master identified. The capacity, initialization, and overflow properties of this set of data records are implementation-specific. The maximum number of records in each data set maintained by the implementation shall be the value of number_foreign_records of the default data set. Externally, these records appear to be numbered from 1 to number_foreign_records. Subclause 7.4, Note 2 applies.

The members of this data set are:

— foreign_master_communication_technology
— foreign_master_uuid_field
— foreign_master_port_id_field
— foreign_master_syncs

7.4.7.1 foreign_master_communication_technology

This member's value shall be the sourceCommunicationTechnology field value of Sync messages from the foreign master. The datatype shall be as specified in 6.2.4.1.

7.4.7.2 foreign_master_uuid_field

This member's value shall be the sourceUuid field value of Sync messages from the foreign master. The datatype shall be as specified in 6.2.4.1.

7.4.7.3 foreign_master_port_id_field

This member's value shall be the sourcePortId field value of Sync messages from the foreign master. The datatype shall be as specified in 6.2.4.1.

7.4.7.4 foreign_master_syncs

This member's value shall be the number of Sync messages received from the foreign master within a time window PTP_FOREIGN_MASTER_TIME_WINDOW. The datatype shall be UInteger16.

7.5 Messaging and internal event behavior of clocks

This subclause specifies the behavior for the following events that may occur in an ordinary clock or for each PTP port of a boundary clock:

— Initialization
— Receipt of any message

— Occurrence of the STATE_CHANGE_EVENT
— Transmission of messages
— Expiration of the sync-event receipt timeout mechanism
— Expiration of the sync-event interval timeout mechanism
— Expiration of the qualification-timeout mechanism
— Completion of the best master clock algorithm
— Detection of an internal fault
— Synchronization changes of the local clock
— Events related to an external timing signal

The behavior resulting from these events may depend on the current state of the PTP port of an ordinary clock or boundary clock PTP port as specified in 7.3.

The PTP protocol shall not result in any communication on the communication paths except as specified in the following subclauses.

Any state or data changes or data set updates of the local clock resulting from the occurrence of any event or qualifying sequence of events defined in this clause shall be atomic. That is, all such changes resulting from an instance of any event or qualifying sequence of events shall be processed before considering changes due to a different instance. The only qualifying sequence of events shall be the receipt of multiple Sync messages within the same sync_interval.

If state or data change or the occurrence of an event results in a request to issue a PTP message, the requests shall be processed in First-In-First-Out (FIFO) order by message type. A logically separate FIFO ordering shall be maintained for each type of message. Once a message request is queued, the message shall be issued irrespective of any subsequent event occurrences.

All events defined in this clause shall obey the timing and ordering constraints of 7.11.

No received PTP message shall be processed unless the default data set subdomain_name value is identical, octet-by-octet for all octets, with the subdomain field of the PTP message header (see 8.2.3).

7.5.1 Initialization

By 7.3.2, the PTP_INITIALIZATION state occurs simultaneously in all ports of a clock. While a clock is in the PTP_INITIALIZATION state, implementation-specific initialization of data sets, hardware, and communication facilities needed for the execution of the PTP protocol shall occur. There shall be no issuing of or responses to PTP messages while in the PTP_INITIALIZATION state.

Where there are dependencies in the initialization, the independent values shall be initialized prior to the initialization of any dependent values.

7.5.2 Receipt by a clock of any message from itself

The PTP protocol makes no use of messages received by the same clock that sent them, and implementations should prevent messages from being delivered to the sender. This clause specifies what to do when such messages are received.

7.5.2.1 Ordinary clocks and any single port of a boundary clock

A message received at a port that was previously issued by the same port shall be ignored. An exception may be made for implementation-specific diagnostic purposes beyond the scope of this standard.

This situation may be identified by comparing the sourceCommunicationTechnology, sourceUuid, and sourcePortId fields of the received message and the port_communication_technology, port_uuid_field, and port_id_field of the port configuration data set of the receiving port. The possibilities and interpretations are shown in Table 14.

NOTE—This condition may occur due to normal or abnormal properties of communication paths.

7.5.2.2 Additional constraints for boundary clocks

A port *n* on a boundary clock may directly receive messages issued by a different port *m* of the same boundary clock. This occurs if both ports *n* and *m* communicate to the same communication path. This is an abnormal situation not detected by the best master clock algorithm and must be handled independently. When this condition is detected in a set of the ports, the boundary clock shall place all of the involved ports except the port with the lowest port_id_field (see 6.2.4.1) (say, *n*) in the PTP_PASSIVE state until such time as port *n* is no longer in the PTP_MASTER state by virtue of normal operation of the protocol.

This situation may be identified by comparing the sourceCommunicationTechnology, sourceUuid, and sourcePortId fields of the received message and the port_communication_technology, port_uuid_field, and port_id_field of the port configuration data set of the receiving port. The possibilities and interpretations are shown in Table 14.

Table 14—Source UUID comparisons

Given a message *m* arriving at port *n* of clock *c*, where *n* has port_communication_technology *a* and port_uuid_field *b*:	
If *m* contains:	**The interpretation is:**
sourceCommunicationTechnology *a*, sourceUuid *b*, sourcePortId *n*	*m* was sent from port *n* on clock *c*.
sourceCommunicationTechnology *a*, sourceUuid $f \neq b$, sourcePortId *g*	*m* was sent from a port on a clock other than *c*.
sourceCommunicationTechnology *a*, sourceUuid *b*, sourcePortId $o \neq n$	*m* was sent from clock *c*, but from a port other than *n*.

7.5.3 Receipt of a Sync message from another clock

The logic for processing a Sync message shall be as defined in Figure 10. The states indicated in this figure refer to the current state of the port receiving the Sync message.

If the port receiving a Sync message is in the PTP_INITIALIZING or PTP_DISABLED states, the message shall be disregarded. If the port receiving a Sync message is in the PTP_FAULTY state, the message shall be disregarded except for implementation-specific purposes meeting the requirements of 7.3.

If the PTP_SYNC_BURST flag in the Sync message is TRUE, the port may disregard the message. If the flag is FALSE, or the port does not disregard the message, the rest of this clause shall apply.

NOTE—An implementation which makes use of Sync messages with the PTP_SYNC_BURST flag TRUE should exercise caution, since the interarrival interval of these messages is shorter than normal.

If the sourceCommunicationTechnology, sourceUuid, and sourcePortId fields of the Sync message are identical with the parent_communication_technology, parent_uuid_field, and parent_port_id_field fields, respectively, of the parent data set, then the message is from the current master clock.

When a Sync message from port *F* is received by port *N* from the communication path associated with *N*, *F* is designated as a foreign master clock if any of the following is true:

— The port *N* is not in the PTP_SLAVE state, or
— The port *N* is in the PTP_SLAVE state and the sourceCommunicationTechnology, sourceUuid, and sourcePortId fields of the message are not all identical with the respective parent_communication_technology, parent_uuid_field, and parent_port_id_field of the parent data set.

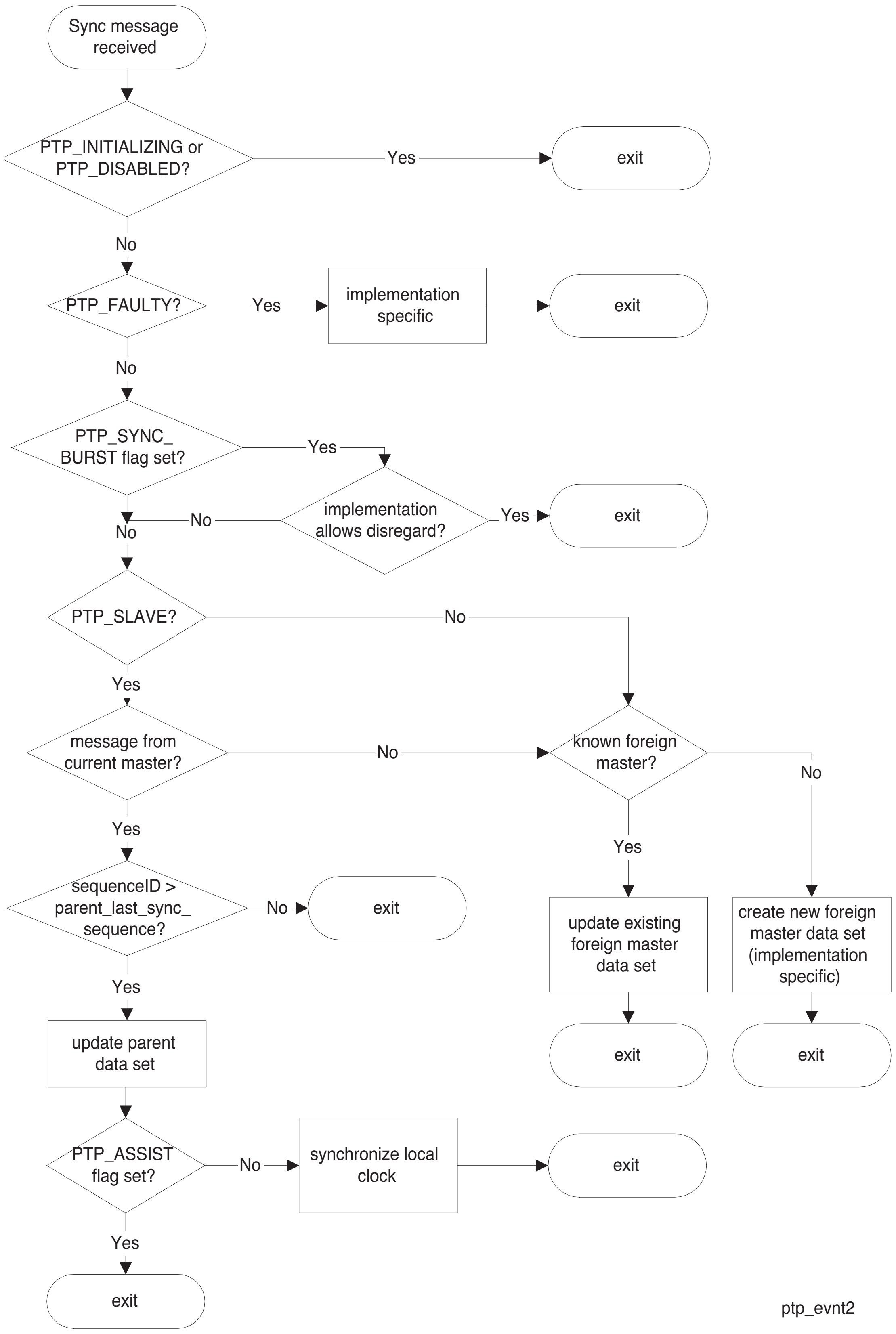
Sync message received
PTP_INITIALIZING or PTP_DISABLED?
Yes
exit
No
PTP_FAULTY?
Yes
implementation specific
exit
No
PTP_SYNC_ BURST flag set?
Yes
implementation allows disregard?
Yes
exit
No
No
PTP_SLAVE?
No
Yes
message from current master?
No
known foreign master?
No
Yes
Yes
sequenceID > parent_last_sync_ sequence?
No
exit
update existing foreign master data set
create new foreign master data set (implementation specific)
exit
exit
Yes
update parent data set
PTP_ASSIST flag set?
No
synchronize local clock
exit
Yes
exit
ptp_evnt2

Figure 10—Receipt of Sync message logic

If a Sync message is received from a foreign master clock, the foreign master data set of the receiving port shall be updated as follows:

— If the sourceCommunicationTechnology, sourceUuid, and sourcePortId fields of the message are identical with the foreign_master_communication_technology, foreign_master_uuid_field, and foreign_master_port_id_field fields respectively of a record in the foreign master data set then the foreign_master_syncs field of that record shall be incremented.
— If the sourceCommunicationTechnology, sourceUuid, and sourcePortId fields of the message are not identical with the foreign_master_communication_technology, foreign_master_uuid_field, and foreign_master_port_id_field fields respectively of a record in the foreign master data set then a new record reflecting the respective field values of the message shall be created with the foreign_master_ syncs field having value 0. Implementation-specific limitations on the capacity of the foreign master data set will limit the number of such records.

If a Sync message is received from the current master clock of record, the parent data set for the receiving port shall be updated according to Table 21, except that the source of each field shall be the received Sync message rather than E_{best}. Note that implementation-specific additional information from this message beyond the fields defined in the parent data set may need to be maintained and updated to meet the requirements of 7.8.

If a Sync message is received from the current master clock of record, the local clock shall be synchronized based the contents of the message in accordance with 7.8, provided all of the following conditions are met. Otherwise, no additional action shall be taken.

a) The port receiving the Sync message is in the PTP_SLAVE state.
b) The PTP_ASSIST bit of the flags field of the received Sync message is FALSE (indicating that a Follow_Up message will not be received).

7.5.4 Receipt of a Follow_Up message from another clock

The logic for processing a Follow_Up message shall be as defined in Figure 11. The states indicated in this figure refer to the current state of the port receiving the Follow_Up message.

If the port receiving a Follow_Up message is in the PTP_INITIALIZING or PTP_DISABLED states, the message shall be disregarded. If the port receiving a Follow_Up message is in the PTP_FAULTY state, the message shall be disregarded except for implementation-specific purposes meeting the requirements of 7.3.

If the PTP_SYNC_BURST flag in the Follow_Up message is TRUE, the port may disregard the message. If the flag is FALSE or the port does not disregard the message, the rest of this clause shall apply.

NOTE—An implementation which makes use of Follow_Up messages with the PTP_SYNC_BURST flag TRUE should exercise caution, since the interarrival interval of these messages is shorter than normal.

If the sourceCommunicationTechnology, sourceUuid, and sourcePortId fields of the Follow_Up message are identical with the parent_communication_technology, parent_uuid_field, and parent_port_id_field fields, respectively, of the parent data set, then the message is from the current master clock. Follow_Up messages not from the current master clock shall be disregarded.

A Follow_Up message from the current master clock shall be qualified if all of the following conditions are met:

c) The port receiving the Follow_Up message is in the PTP_SLAVE state.
d) The associatedSequenceId field of the Follow_Up message is identical to the parent_last_sync_ sequence_number field of the parent data set.

The receipt of a qualified Follow_Up message shall cause the local clock to be synchronized. When so indicated, the local clock shall synchronize to the current master clock based on information in the qualified Follow_Up message and the Sync message designated by parent_communication_technology, parent_uuid_field, parent_port_id_field, and parent_last_sync_sequence_number fields of the parent data set. Synchronization shall be based on the fields in a Follow_Up message and the matching Sync message from the current master clock. Synchronization shall be in accordance with 7.8.

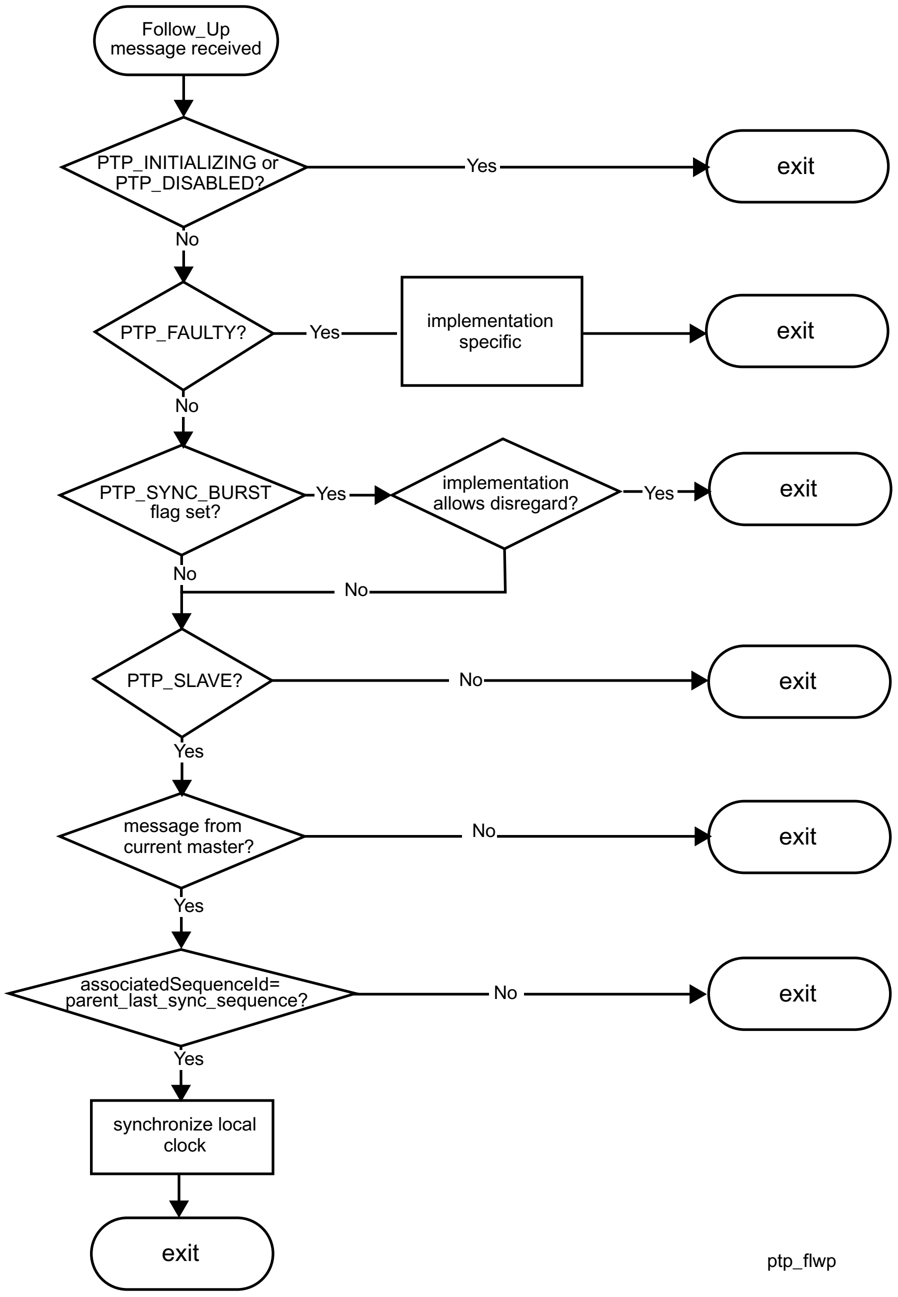
Follow_Up message received
PTP_INITIALIZING or PTP_DISABLED?
Yes
exit
No
PTP_FAULTY?
Yes
implementation specific
exit
No
PTP_SYNC_BURST flag set?
Yes
implementation allows disregard?
Yes
exit
No
No
PTP_SLAVE?
No
exit
Yes
message from current master?
No
exit
Yes
associatedSequenceId= parent_last_sync_sequence?
No
exit
Yes
synchronize local clock
exit
ptp_flwp

Figure 11—Receipt of Follow_Up message logic

7.5.5 Receipt of a Delay_Req message from another clock

The logic for processing a Delay_Req message shall be as defined in Figure 12. The states indicated in this figure refer to the current state of the port receiving the Delay_Req message.

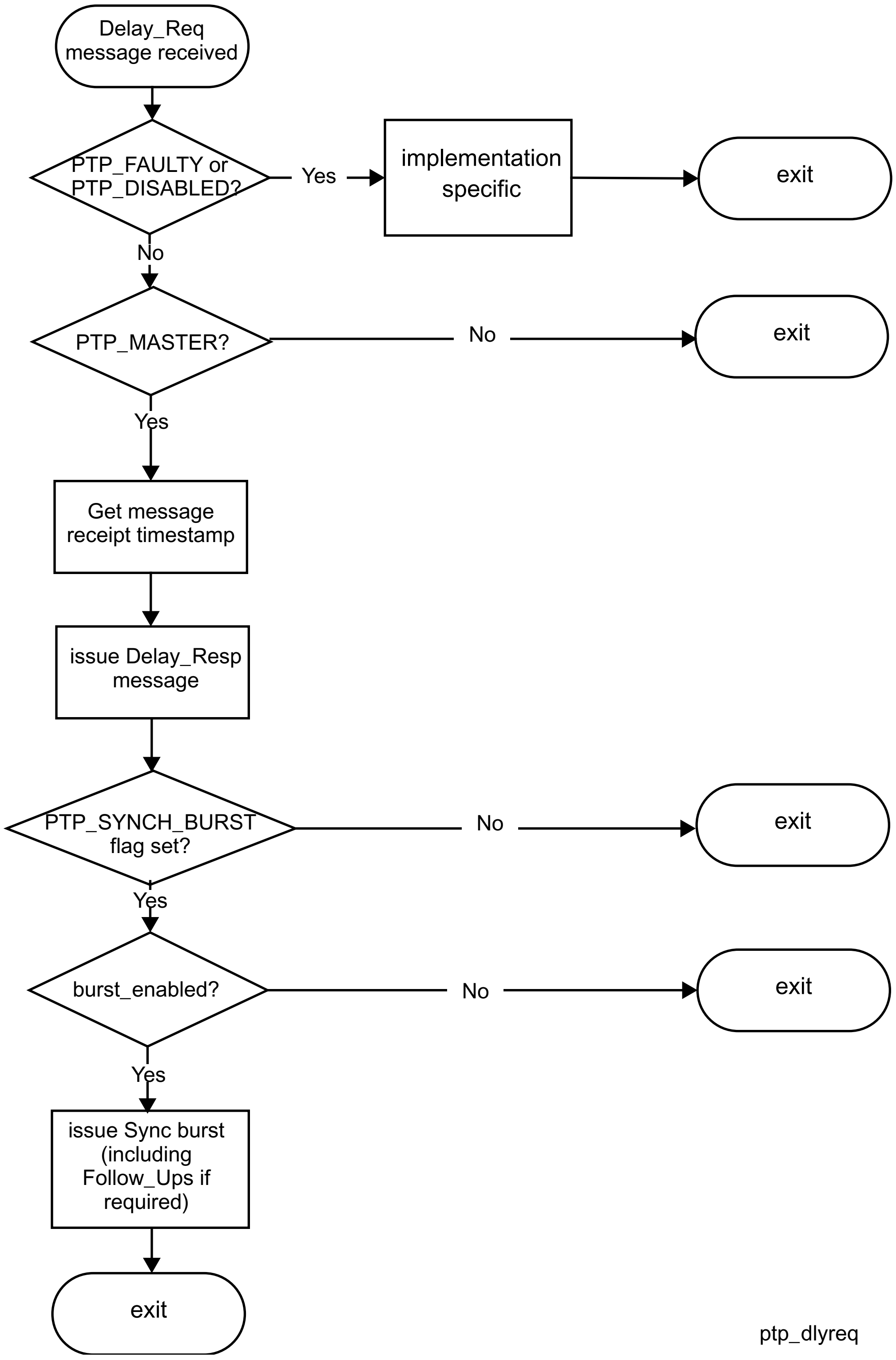
Delay_Req message received
PTP_FAULTY or PTP_DISABLED?
Yes
implementation specific
exit
No
PTP_MASTER?
No
exit
Yes
Get message receipt timestamp
issue Delay_Resp message
PTP_SYNCH_BURST flag set?
No
exit
Yes
burst_enabled?
No
exit
Yes
issue Sync burst (including Follow_Ups if required)
exit
ptp_dlyreq

Figure 12—Receipt of Delay_Req message logic

If the port receiving the Delay_Req message is in the PTP_INITIALIZING or PTP_DISABLED states, the message shall be disregarded. If the port receiving a Delay_Req message is in the PTP_FAULTY state, the message shall be disregarded except for implementation-specific purposes meeting the requirements of 7.3.

Otherwise, if the port receiving the Delay_Req message is not in the PTP_MASTER state, the message shall not result in any change in an active port (see 7.3.2). Otherwise, the following behavior shall be implemented by the receiving port.

The receiving port shall capture the time, as measured by the local clock, at which the message timestamp point passed the inbound clock timestamp point (see Figure 4). It shall correct this value for latency (see 7.8.1.3) and retain the result for use as delayReceiptTimestamp (see 8.5.1.2).

The receiving master clock shall issue a Delay_Resp message on the receiving port (see 7.5.12).

If the PTP_SYNC_BURST flag in the Delay_Req message is TRUE and the receiving port's value of burst_enabled is TRUE, it is recommended that the receiving master clock issue a sequence of Sync messages (hereafter called a *burst*) on the receiving port. The Sync messages in a burst are excluded from the definition of *sync interval* in 6.2.5.5. If issued, a burst shall have these properties:

— It shall consist of Sync messages (and possibly Follow_Up messages) (see 7.5.9). These messages may include those sent as a result of the SYNC_INTERVAL_TIMEOUT_EXPIRES event (see 7.5.15).
— The messages shall be sent at approximately evenly-spaced times. The spacing shall be less than the sync_interval (see 6.2.5.5).
— The number of messages in the burst shall be >1 and ≤8.
— If a message in the burst is being sent as a result of the SYNC_INTERVAL_TIMEOUT_EXPIRES event, the PTP_SYNC_BURST flag in the message shall be FALSE; otherwise it shall be TRUE.

If a port receives one or more Delay_Req messages in which the PTP_SYNC_BURST flag is TRUE while it is in the process of sending a burst, it may choose separately for each such message whether to send a burst in response. Whatever choice it makes, it shall finish each burst before beginning the next one.

If burst_enabled is FALSE on a port, the clock shall not issue a burst on that port.

7.5.6 Receipt of a Delay_Resp message from another clock

The logic for processing a Delay_Resp message shall be as defined in Figure 13. The states indicated in this figure refer to the current state of the port receiving the Delay_Resp message.

If the port receiving a Delay_Resp message is in the PTP_INITIALIZING or PTP_DISABLED states, the message shall be disregarded. If the port receiving a Delay_Resp message is in the PTP_FAULTY state, the message shall be disregarded except for implementation-specific purposes meeting the requirements of 7.3.

If the sourceCommunicationTechnology, sourceUuid, and sourcePortId fields of the Delay_Resp message are identical with the parent_communication_technology, parent_uuid_field, and parent_port_id_field fields, respectively, of the parent data set, then the message is from the current master clock.

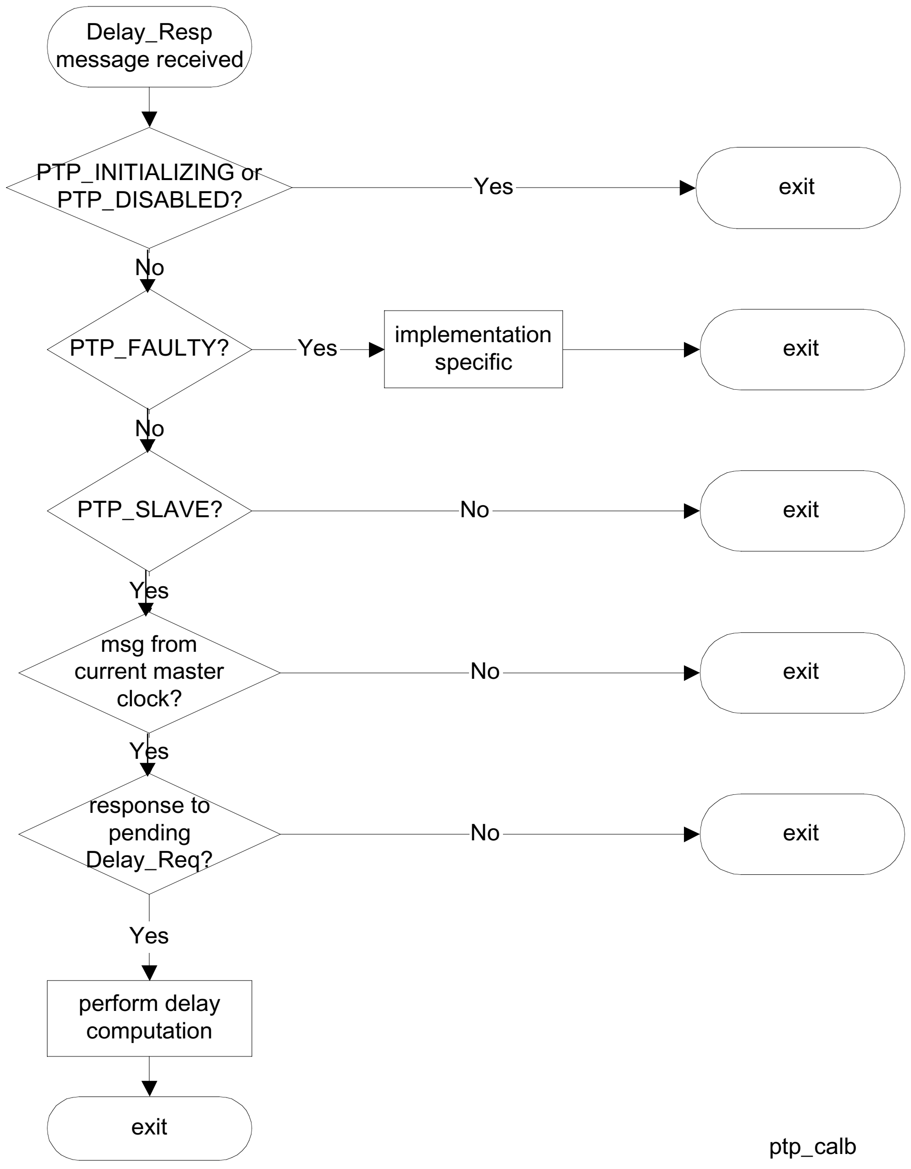
Delay_Resp message received
PTP_INITIALIZING or PTP_DISABLED?
Yes
exit
No
PTP_FAULTY?
Yes
implementation specific
exit
No
PTP_SLAVE?
No
exit
Yes
msg from current master clock?
No
exit
Yes
response to pending Delay_Req?
No
exit
Yes
perform delay computation
exit
ptp_calb

Figure 13—Receipt of Delay_Resp message logic

If a Delay_Resp message is not from the current master clock, it shall be disregarded. Otherwise, delay computations in accordance with 7.8 shall be executed if and only if the values of the requestingCommunicationTechnology, requestingClockUuid, requestingPortId and requestingSequenceId fields of the Delay_Resp message are identical with the respective values of the sourceCommunicationTechnology, sourceUuid, sourcePortId, and sequenceId fields of the last Delay_Req message sent from the clock.

7.5.7 Receipt of a Management message from another node

If the port receiving the Management message is in the PTP_INITIALIZING state, the message shall not result in any change in an active port (see 7.3.2).

A received Management message shall be accepted if one of the following conditions is met:

— The targetCommunicationTechnology and targetUuid fields of the received Management message respectively are identical to the clock_communication_technology and clock_uuid_field of the default data set of the receiving clock, or
— The targetCommunicationTechnology and targetUuid fields of the received Management message are identical to the respective default values for the respective fields of a Clock-UUID (see 6.2.4.1).

For accepted management Messages, the behavior specified in 7.12 shall be implemented.

7.5.8 Occurrence of a STATE_CHANGE_EVENT

The STATE_CHANGE_EVENT is the mechanism for using the data in received Sync messages to determine which is the best master clock, and whether the clock receiving the Sync message needs to change its state.

Every clock shall implement a mechanism generating the STATE_CHANGE_EVENT, and the occurrence of one shall implement the logic of Figure 14.

The STATE_CHANGE_EVENT shall:

— Logically occur simultaneously on all ports of a clock.
— Occur at least once per sync interval.
— Not occur when any port is in the PTP_INITIALIZING state.

Prior to or as the first action of the STATE_CHANGE_EVENT logic, each port *N*, not in the PTP_DISABLED or PTP_FAULTY states, shall compute an updated value of E_{rbest} (see 7.6), reflecting the receipt of Sync messages since the last STATE_CHANGE_EVENT.

Following the computation of the set of E_{rbest} for all ports, the clock shall complete the following tasks, in the order given:

a) Compute E_{best}.
b) Apply the best master clock algorithm.
c) Update the appropriate data sets.
d) Make the required state changes in all ports.

These tasks shall be carried out atomically—that is, all tasks and updates shall complete before any information serving as inputs to these tasks is changed. This input information shall include the set of E_{rbest} values. These tasks shall be executed as defined in 7.6.

NOTE—The arrival of a Sync message is a convenient time to signal a STATE_CHANGE_EVENT. However, implementations must assure that STATE_CHANGE_EVENTs occur frequently enough, even if no Sync messages arrive, and should operate efficiently in the presence of Sync message bursts. The SYNC_INTERVAL_TIMEOUT_EXPIRES event is also a convenient time to signal a STATE_CHANGE_EVENT, but it useful only when a clock is in the PTP_MASTER state.

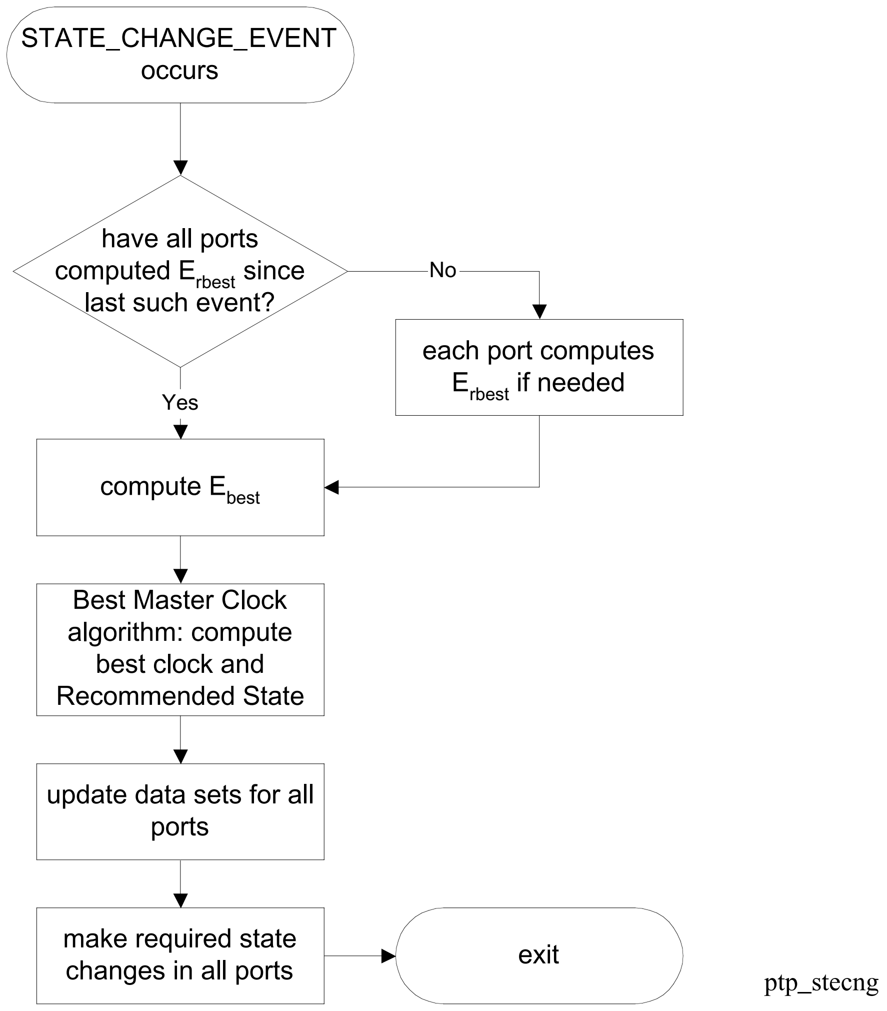
STATE_CHANGE_EVENT occurs
have all ports computed E_{rbest} since last such event?
No
Yes
each port computes E_{rbest} if needed
compute E_{best}
Best Master Clock algorithm: compute best clock and Recommended State
update data sets for all ports
make required state changes in all ports
exit
ptp_stecng

Figure 14—STATE_CHANGE_EVENT logic

7.5.9 Transmission of a Sync message

A port shall issue a Sync message on a port if:

— Required by a Management message if specified in 7.12, or
— The port is in the PTP_MASTER state and the port's SYNC_INTERVAL_TIMEOUT_EXPIRES event occurs (see 7.5.15).

It is recommended that a clock issue a Sync message on a port in the PTP_MASTER state if the port has received a Delay_Req message in which the PTP_SYNC_BURST flag is TRUE (see 7.5.5), and the port's value of burst_enabled is TRUE.

A clock shall not issue a Sync message under any circumstances not mentioned in this clause.

The fields of the issued Sync message shall conform to 8.3.1.

The value of port configuration data set member last_sync_event_sequence_number shall be incremented by +1 before transmission of a Sync message by the port.

The port shall update the value of grandmaster_sequence_number of the parent data set to the incremented value of the last_sync_event_sequence_number of the port data set if and only if the following constraints are met:

— The clock is a grandmaster clock. Note that a clock is the grandmaster clock of a subdomain if it has no ports in the PTP_SLAVE state (see 7.2).
— The port is the port with the lowest value of port_id_field that is also in the PTP_MASTER state (see 7.4.4.20).

If the default data set member clock_followup_capable is TRUE, then the port shall also issue a corresponding Follow_Up message with fields conforming to 8.4.1, and with the timing conforming to 7.11. In this case the port shall capture the sequenceId value of the Sync message as an input to the associatedSequenceId field of the Follow_Up message. The mechanism for obtaining the value for the preciseOriginTimestamp field of the associated Follow_Up message shall be started.

7.5.10 Transmission of a Follow_Up message

A port shall issue a Follow_Up message if and only if:

— Required by a Management message if specified in 7.12, or
— The port is in the PTP_MASTER state, the default data set member clock_followup_capable is TRUE, and a Follow-Up message is required under the terms of 7.5.9.

The fields of the issued Follow_Up message shall conform to 8.4.1, and with the timing conforming to 7.11.

The value of port configuration data set member last_general_event_sequence_number shall be incremented by +1 before transmission of a Follow_Up message by the port.

7.5.11 Transmission of a Delay_Req message

A clock shall issue a Delay_Req message on a port if:

— Required by a Management message if specified in 7.12, or
— The port is in the PTP_SLAVE state and the information obtained in the resulting Delay_Resp message from the parent is required for the delay computations specified by 7.8.1.2. The issuance of Delay_Req messages under this condition shall be limited to the times specified in 7.11. If the value of burst_enabled for the port is TRUE, the clock may set the PTP_SYNC_BURST flag in the Delay_Req message to TRUE.

NOTE—The behavior with the PTP_SYNC_BURST flag set to TRUE is useful if the clock needs to get its statistics established quickly if it is starting up and the sync_interval is long. However, it consumes network bandwidth and master clock cycles and should be used with caution.

A clock may issue a Delay_Req message on a port at any time if the port is in the PTP_SLAVE state and the value of burst_enabled for the port is TRUE. The value of the PTP_SYNC_BURST flag for a Delay_Req message issued under this condition shall be FALSE.

NOTE—Sending several Delay_Req messages in quick succession is a way for the clock to get its statistics established quickly if it is starting up and the sync_interval is long. However, this behavior consumes network bandwidth and master clock cycles and should be used with caution.

A clock shall not issue a Delay_Req message on a port under any circumstances not mentioned in this clause.

The value of port configuration data set member last_sync_event_sequence_number shall be incremented by +1 before transmission of a Delay_Req message by the port.

The fields of the issued Delay_Req message shall conform to 8.3.1. The timing shall conform to 7.11.

7.5.12 Transmission of a Delay_Resp message

A port shall issue a Delay_Resp message if and only if:

— Required by a Management message if specified in 7.12, or
— The port is in the PTP_MASTER state and has received a Delay_Req message (see 7.5.5).

The fields of the issued Delay_Resp message shall conform to 8.5.1, and with the timing conforming to 7.11.

The value of port configuration data set member last_general_event_sequence_number shall be incremented by +1 before transmission of a Delay_Resp message by the port.

7.5.13 Transmission of a Management message

A port shall issue a Management message if and only if:

— Required by a received Management message if specified in 7.12, or
— Required by implementation-specific considerations outside the scope of this standard.

The fields of the issued Management message shall conform to 8.6.1, and with the timing conforming to 7.11.

The value of port configuration data set member last_general_event_sequence_number shall be incremented by +1 before transmission of a Management message by the port.

7.5.14 Sync-event receipt timeout mechanism events

Each protocol engine shall support a timeout mechanism known as the *sync-event receipt timeout*.

This timeout mechanism for a port shall be set to zero and (re)started when any of the following occur:

— When a Sync message is received from the current parent clock port of record per 7.5.3, or
— When the sync-event receipt timeout timer expires after an interval PTP_SYNC_RECEIPT_TIMEOUT seconds, or
— When leaving the PTP_PRE_MASTER (except when the next state is PTP_MASTER), or the PTP_MASTER states, or
— When entering the PTP_LISTENING state

This timeout mechanism for a port shall be set to zero and not restarted when entering the PTP_INITIALIZING, PTP_PRE_MASTER, PTP_FAULTY, PTP_DISABLED, or the PTP_MASTER states.

The expiration of the sync-event receipt timeout mechanism after an interval PTP_SYNC_RECEIPT_TIMEOUT seconds defines the SYNC_RECEIPT_TIMEOUT_EXPIRES event specified in the protocol engine state machine (see 7.3).

The logic for processing a SYNC_RECEIPT_TIMEOUT_EXPIRES event on a port shall be as defined in Table 15.

Table 15—SYNC_RECEIPT_TIMEOUT_EXPIRES logic

State	SYNC_RECEIPT_TIMEOUT_EXPIRES event: resulting actions based on state
PTP_INITIALIZING	Does not occur per this clause.
PTP_FAULTY	Does not occur per this clause.
PTP_DISABLED	Does not occur per this clause.
PTP_PRE_MASTER	Does not occur per this clause.
PTP_MASTER	Does not occur per this clause.
PTP_LISTENING PTP_PASSIVE PTP_UNCALIBRATED PTP_SLAVE	In order: a) Update the port's data sets to the PTP_MASTER state configuration (see 7.6.5) as specified for decision M1. b) Set the port's state to PTP_MASTER.

7.5.15 Sync-event interval timeout mechanism events

Each protocol engine shall support a timeout mechanism known as the *sync-event interval timeout*. Nothing in this clause shall prevent implementations of boundary clocks from using a common mechanism for all ports.

This timeout mechanism for a port shall be set to zero and (re)started when any of the following occur:

— When the sync-event interval timeout timer expires after an interval PTP_SYNC_INTERVAL_TIMEOUT seconds, or

— When entering the PTP_MASTER state

This timeout mechanism for a port shall be set to zero and not restarted when leaving the PTP_MASTER state.

This timeout mechanism for a port may be set to zero and (re)started when entering the PTP_PRE_MASTER state. In this case the timeout mechanism for a port shall be set to zero and not restarted when leaving the PTP_PRE_MASTER state unless the next state is PTP_MASTER.

The expiration of the sync-event interval timeout mechanism after an interval PTP_SYNC_INTERVAL_TIMEOUT seconds defines the SYNC_INTERVAL_TIMEOUT_EXPIRES event. The effects of a SYNC_INTERVAL_TIMEOUT_EXPIRES event are specified in 7.5.8 and 7.5.9.

7.5.16 Qualification-timeout mechanism events

The expiration of the qualification-timeout mechanism defines the QUALIFICATION_TIMEOUT_EXPIRES event of Figure 9.

The Qualification-Timeout mechanism shall start when the port enters the PTP_PRE_MASTER state. The expiration shall occur after a delay computed as follows:

— The delay shall be for *N* multiplied by PTP_SYNC_INTERVAL_TIMEOUT seconds.

— If the BEST_MASTER_CLOCK = PTP_MASTER event of Figure 9 was based on decision points M1 or M2 of Figure 16, *N* shall be 0.

— If the BEST_MASTER_CLOCK = PTP_MASTER event of Figure 9 was based on decision point M3 of Figure 16, *N* shall be the value incremented by 1 (one) of the steps_removed field of the current data set.

7.5.17 Execution of the best master clock algorithm

The best master clock algorithm shall be executed once for each occurrence specified in 7.5.8. The execution of the best master clock algorithm may result in a change of protocol engine clock state as specified in the protocol engine state machine (see 7.3).

7.5.18 Detection of an internal fault

An internal fault shall be any condition that if persistent will:

— Prevent the correct execution of the protocol defined by this standard, or
— Prevent the correct implementation of any APIs defined by this standard, or
— Produce or continue faulty behavior of the local clock

The enumeration of all such faults and the mechanisms for detecting them is outside the scope of this standard.

It is recommended that the following condition be detected and identified as a fault for purposes of this clause:

— Inconsistency between the values of the sync_interval from the default data set and the syncInterval field of any received Sync or Delay_Req messages.

The detection of such a fault defines the FAULT_DETECTED event. The occurrence of a FAULT_DETECTED event shall result in the state change defined in the protocol engine state machine (see 7.3).

The clearing of all detected fault conditions, either as a result of Management messages or internal procedures, defines the FAULT_CLEARED event. The occurrence of a FAULT_CLEARED event shall result in the state change defined in the protocol engine state machine (see 7.3).

7.5.19 Effects of changes in the specifications of the local clock

The performance of a local clock may change due to internal changes or due to the interaction with other clocks via the exchange of PTP messages.

7.5.19.1 Changes in the local clock due to PTP messages

Changes in internal state resulting from the exchange of non-Management PTP messages shall be governed by the operation of the protocol engine state machine (see 7.3), and the specifications for the best master clock selection (see 7.6).

Any changes in internal state resulting from the exchange of PTP Management messages are specified for each such message.

7.5.19.2 Changes in the local clock due to other causes

Changes due to internal properties of the local clock (for example, a drift of a local oscillator or changes in information received from a GPS receiver to which the local clock is directly synchronized) may result in updates to the data sets.

All resulting changes to data sets shall be treated atomically with respect to any other activity accessing the data sets including the activity specified in 7.5.8.

Any changes in the performance of the local clock that affect the default data set shall cause the default data set to be updated to reflect the current properties of the local clock.

For clocks that are the grandmaster clock of a subdomain, any changes in the performance of the local clock that affect the global time properties data set shall cause the global time properties data set to be updated to reflect the current properties of the local clock.

7.5.20 Events related to an external timing signal

A clock may have an external timing signal (see 6.2.3). This subsection specifies the events related to such a signal.

7.5.20.1 Transition on the seconds boundary of the local clock

If the PTP_EXT_SYNC flag is TRUE for the local clock and the clock has a port in the PTP_MASTER state, the clock shall issue an external timing signal on each seconds boundary of the local clock (the instant when the nanoseconds portion of the local clock transitions from a very large to a very small number due to normal advancing of the local clock).

7.5.20.2 Receipt of an external timing signal

The receipt of an external timing signal may be used for implementation-specific correction to the time of the local clock if and only if the local clock has a port in the PTP_SLAVE state. It is recommended that this correction not occur if the PTP_EXT_SYNC flag of Sync messages from the Parent clock of the local clock is FALSE. The behavior of PTP systems in which the external timing signal is received from a clock other than the parent clock of the receiving clock is outside the scope of this standard.

Note that there is no mechanism specified by this standard for correcting for latency in the receipt of the external timing signal analogous to the latency correction mechanism of 7.8.1.2 for Sync events. The correction for latency and other effects associated with the external timing signal is outside the scope of this standard.

7.5.20.3 Specification of the external timing signal

The external timing signal shall consist of a 10 Mb/s Manchester encoded alternating 1s and 0s bit pattern with a distinguished signal at the seconds boundary of the master's local clock. The distinguished signal shall be the bit pattern 1,1, with the start of this pattern on the seconds boundary, as illustrated in Figure 15.

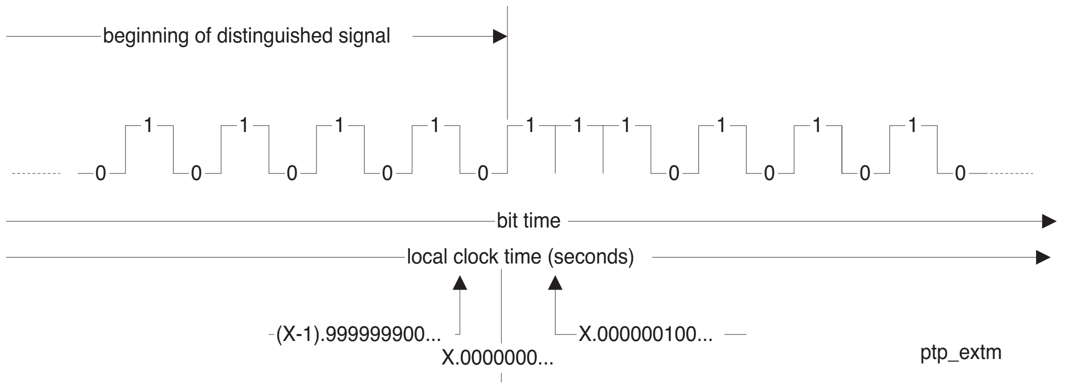

Figure 15—External timing signal specification

It is recommended that whatever medium is used to convey this external timing signal that the signal levels and physical characteristics conform to those for 10BaseT Ethernet technology.

7.6 Best master clock algorithm

This clause specifies the way that a local clock determines which of all the clocks it can see (including itself) is the *best*. From that it determines what state it should be in (Table 7). The algorithm runs independently on each clock in a PTP system. That is, clocks do not negotiate which should be master and which should be slave—instead, each computes only its own state. The algorithm avoids configurations with two masters, or none, or which oscillate.

7.6.1 Overview and definition of terms

The best master clock (BMC) algorithm consists of two parts:

— A data set comparison algorithm that computes a binary relation on data sets associated with two clock ports. One of these data sets may represent the default characteristics of the local clock (see 7.4.2). Otherwise the data sets are based on information contained in Sync messages received from ports in the subdomain. This algorithm shall be as specified in 7.6.2.

— An algorithm that computes a recommendation for the state of each port involved under the terms of 7.3. This algorithm shall be as specified in 7.6.3. The recommendation is called *Recommended State* in Figure 9, Figure 14, and Figure 16.

The purpose of the BMC is to determine the state of each port of a PTP system. This determination is based on information contained in Sync messages received at the ports of a given clock and on the default data set values of the given clock.

Given a PTP subdomain in which the network and clock configurations do not change with time, the BMC is designed to produce the following characteristics:

a) If the subdomain has 0 or 1 stratum-1 or stratum-2 clocks, whether or not designated as preferred, and no clocks with stratum >2 that are designated as preferred, there will be exactly one grandmaster clock (see 6.2.4.4).

b) If the subdomain has N>1 stratum-1 or stratum-2 clocks, whether or not designated as preferred, and no clocks with stratum >2 that are designated as preferred, the subdomain will be partitioned into N disjoint subdomains, each with exactly one of the stratum-1 or stratum-2 clocks as the grandmaster.

c) If the subdomain has N>0 stratum-1 or stratum-2 clocks, and additional clocks with stratum >2 that are designated as preferred, the subdomain will be partitioned into at least N disjoint subdomains, each with exactly one of the stratum-1 or stratum-2 clocks as the grandmaster. Depending on the details of the topology and whether any of the stratum-1 or stratum-2 clocks is designated as preferred, there may be additional disjoint subdomains, each with exactly one of the preferred stratum >2 clocks as the grandmaster.

d) Within each of the disjoint subdomains described in items b) and c), the parent-child hierarchy between the grandmaster and all of the other clocks will be a spanning tree with the grandmaster as root and a minimum number of segments (separated by boundary clocks) between any clock and its grandmaster.

The BMC algorithm is run locally on all ports of every clock in a subdomain and results in the previous properties a) through d). Since it runs continually, it continually readapts to changes in the network or the clocks.

7.6.2 BMC algorithm

The BMC algorithm on a typical clock C_0 characterized by a default data set D_0 and with N ports shall be as follows.

— For each port r, qualified Sync messages received from other ports of other clocks connected to the communication path used by port r shall be compared.

— For each port r, the best of these messages E_{rbest} shall be determined using the data set comparison algorithm.

— For the set of N ports of clock C_0, the best of all messages, E_{best}, shall be determined from the N E_{rbest} messages using the data set comparison algorithm.

— For each of the N ports of clock C_0, the messages E_{best} and E_{rbest} and the default data set D_0 shall be used with the state decision algorithm to determine the BMC event applicable to the state machine of each port (see 7.3).

Each port may determine E_{rbest} independently of activity on other ports. The determination of E_{best}, the application of the state decision algorithm, and the instantiation of any state changes required by the application of the results of these determinations shall be atomic. That is, during this process the set of values of E_{rbest} from all ports and the contents of the default data set being used shall not be updated.

In computing E_{rbest} a port r shall:

— Consider only qualified Sync messages received on port r, and only the most recent one from each sending port. The definition of *more recent* follows.

— Include the results of the previous computation of E_{rbest} on port r. However, if there is a more recent qualified Sync message from the same sending port, the values from that message shall be considered instead. If a SYNC_RECEIPT_TIMEOUT_EXPIRES event occurs (see 7.5.14) for the clock selected during the previous computation of E_{rbest} on port r, then the previous computation of E_{rbest} on port r shall not be included.

— Include at least one qualified Sync message from a foreign master clock if such a message exists. In this case, r shall delete from its foreign master data set all such records that were considered but not selected as E_{rbest}.

NOTE—It is recommended that Sync messages be included from as many foreign master clocks as port resources permit.

A Sync message *S* received by a port *r* shall be qualified for consideration by the BMC algorithm as follows:

— If *S* was sent from port *r* or from any other port on the clock containing port *r* (see 7.5.2), *S* shall not be qualified.
— If *S* is not the most recent Sync message received from a given clock, *S* shall not be qualified.
— If port *r* is in the PTP_DISABLED or PTP_FAULTY states, *S* shall not be qualified.
— If *r* is in the PTP_SLAVE state, and *S* is from the current master clock, and *S* is not the most recent Sync message from the current master clock, *S* shall not be qualified. The most recent Sync message from the current master clock shall be identified by the parent_communication_technology, parent_uuid, parent_port_id and parent_last_sync_sequence_number fields of the parent data set.
— If the sender of *S* is a foreign master clock *F*, and fewer than PTP_FOREIGN_MASTER_THRESHOLD nonidentical Sync messages from *F* have been received within the most recent PTP_FOREIGN_MASTER_TIME_WINDOW, *S* shall not be qualified.

  NOTE—The purpose of this window and threshold is to disqualify a potential master clock if we have not heard from it in too long, without requiring us to receive every one of its Sync messages.

— Otherwise, *S* shall be qualified.

In computing E_{rbest}, a port may collect qualified Sync messages for batch processing to determine E_{rbest} provided that the determination of E_{rbest} shall be made at least once per PTP_SYNC_INTERVAL_TIMEOUT if a qualified Sync message has been received. If available, at least two qualified Sync messages from different sources shall be included in each determination of E_{rbest}.

A Sync message *A* is more recent than a Sync message *B* if:

— The grandmasterSequenceId field of *A* is greater than the grandmasterSequenceId field of *B* appropriately taking into account the rollover properties of the datatype, or
— The grandmasterSequenceId field of *A* is the same as the grandmasterSequenceId field of *B*, and the sequenceId field of *A* is greater than the sequenceId field of *B* appropriately taking into account the rollover properties of the datatype.

7.6.3 State decision algorithm

The state decision algorithm used in the best master clock algorithm shall be as defined by Figure 16. After a decision is reached by use of this algorithm, the data sets of the local clock shall be updated as specified in 7.6.5.

D_0 represents the characteristics of clock C_0 as specified in the default data set. Determinations of D_0 relative to E_{best} or E_{rbest} shall be made using the data set comparison algorithm. Determinations of E_{best} relative to E_{rbest} shall be made using the data set comparison algorithm. The determination of whether D_0 is stratum 1 or 2 shall be made based on the value of the clock_stratum value of D_0.

The values for Recommended State used in the protocol engine state machine of Figure 9 shall be the values for Recommended State given in Figure 16.

7.6.4 Data set comparison algorithm

The best master clock algorithm compares one clock to another by comparing data sets that represent those clocks. The data set comparison algorithm shall be as defined by Figure 17, Figure 18, Figure 19, and Figure 20. The data sets are indicated in these figures as set *A* and set *B*. The sources for data set values are given in Table 16.

NOTE—The general process followed in this comparison is as follows:

a) Find which clock derives its time from the better grandmaster. Choosing this, rather than which is the better clock, is essential for the stability of the algorithm.
b) If the grandmasters are equal or equivalent, choose the clock which is nearer in path length to its grandmaster, or which has heard from its grandmaster more recently.
c) If those properties are equivalent, use tie-breaking techniques.

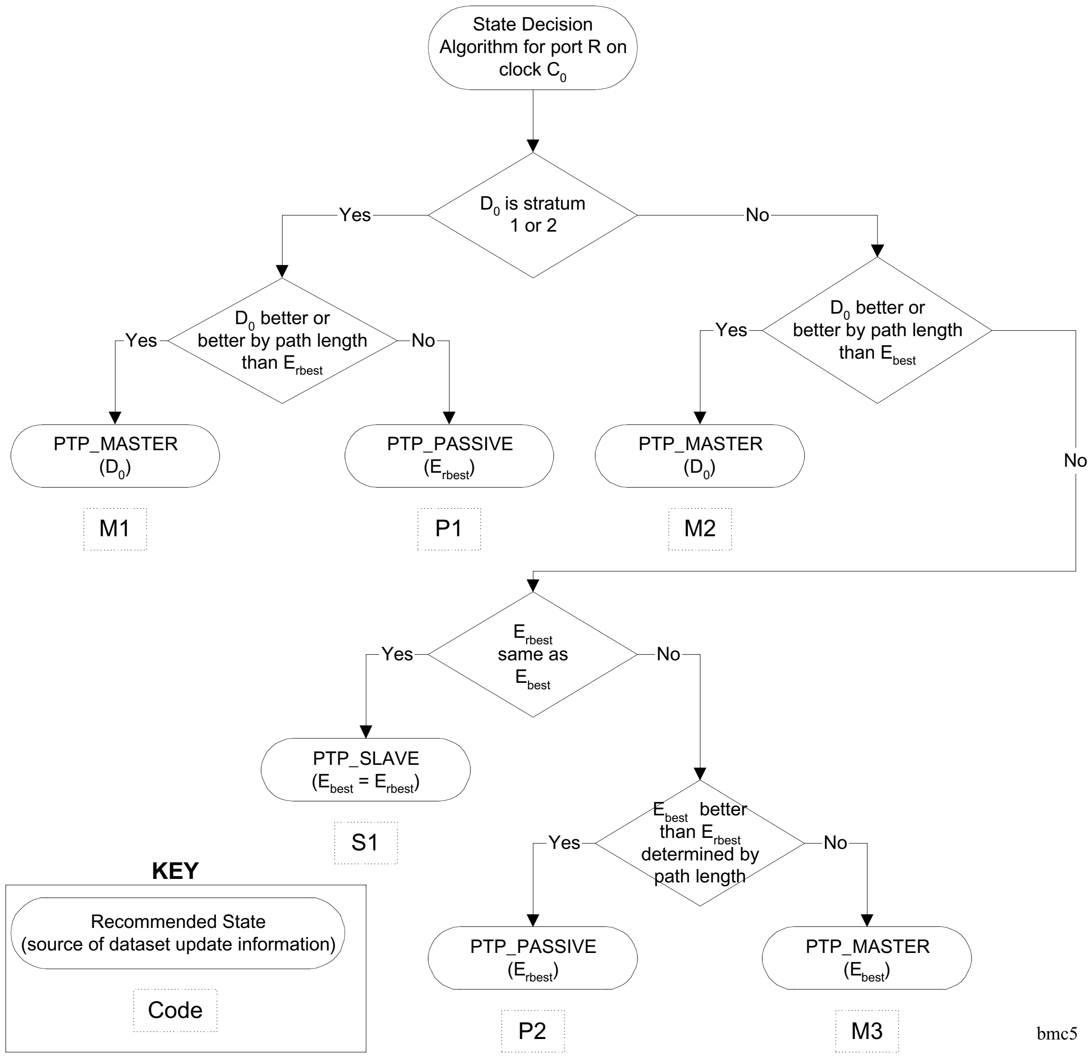
State Decision Algorithm for port R on clock C_0
D_0 is stratum 1 or 2
Yes
No
D_0 better or better by path length than E_{rbest}
Yes
No
D_0 better or better by path length than E_{best}
Yes
No
PTP_MASTER (D_0)
M1
PTP_PASSIVE (E_{rbest})
P1
PTP_MASTER (D_0)
M2
E_{rbest} same as E_{best}
Yes
No
PTP_SLAVE ($E_{best} = E_{rbest}$)
S1
E_{best} better than E_{rbest} determined by path length
Yes
No
PTP_PASSIVE (E_{rbest})
P2
PTP_MASTER (E_{best})
M3
KEY
Recommended State (source of dataset update information)
Code
bmc5

Figure 16—State decision algorithm

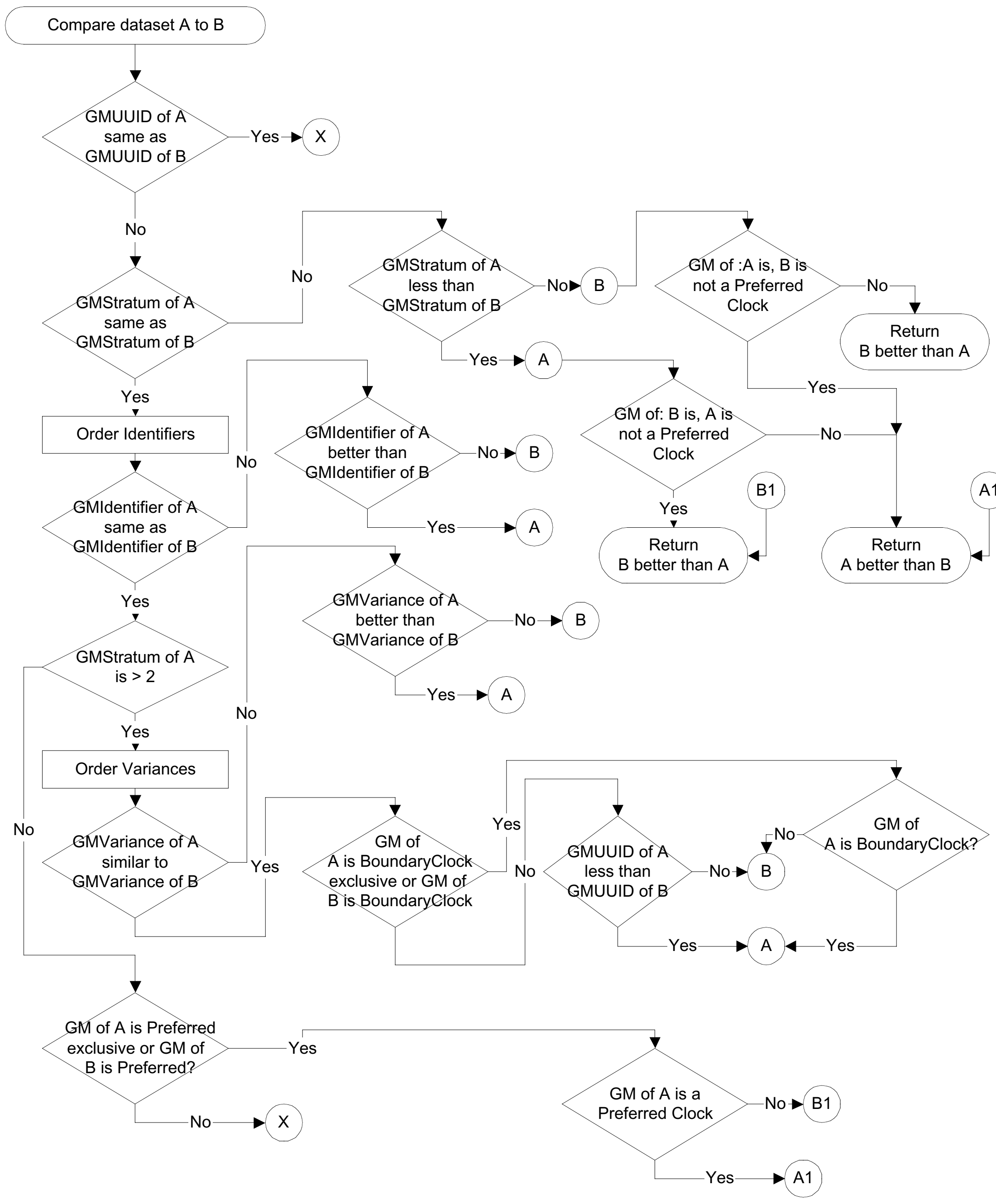
Compare dataset A to B
GMUUID of A same as GMUUID of B
Yes
X
No
GMStratum of A same as GMStratum of B
No
GMStratum of A less than GMStratum of B
No
B
Yes
A
GM of :A is, B is not a Preferred Clock
No
Return B better than A
Yes
GM of: B is, A is not a Preferred Clock
No
Yes
Return B better than A
B1
Return A better than B
A1
Yes
Order Identifiers
GMIdentifier of A same as GMIdentifier of B
No
GMIdentifier of A better than GMIdentifier of B
No
B
Yes
A
Yes
GMVariance of A better than GMVariance of B
No
B
Yes
A
GMStratum of A is > 2
Yes
Order Variances
No
GMVariance of A similar to GMVariance of B
Yes
GM of A is BoundaryClock exclusive or GM of B is BoundaryClock
Yes
No
GMUUID of A less than GMUUID of B
No
B
Yes
A
GM of A is BoundaryClock?
No
Yes
No
GM of A is Preferred exclusive or GM of B is Preferred?
Yes
No
X
GM of A is a Preferred Clock
No
B1
Yes
A1

Figure 17—Data set comparison algorithm, part 1

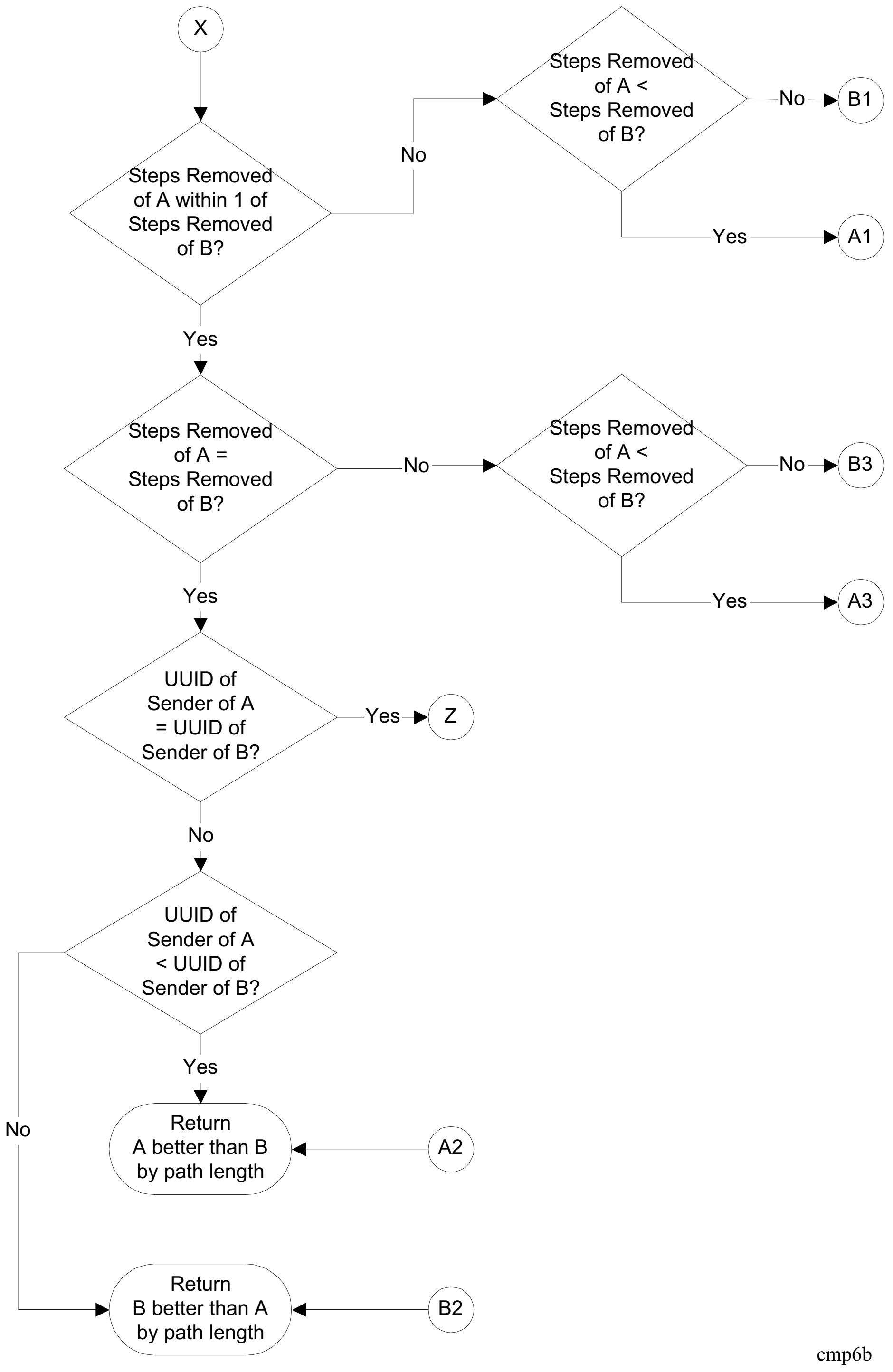
X
Steps Removed of A within 1 of Steps Removed of B?
No
Steps Removed of A < Steps Removed of B?
No
B1
Yes
A1
Yes
Steps Removed of A = Steps Removed of B?
No
Steps Removed of A < Steps Removed of B?
No
B3
Yes
A3
Yes
UUID of Sender of A = UUID of Sender of B?
Yes
Z
No
UUID of Sender of A < UUID of Sender of B?
Yes
No
Return A better than B by path length
A2
Return B better than A by path length
B2
cmp6b

Figure 18—Data set comparison algorithm, part 2

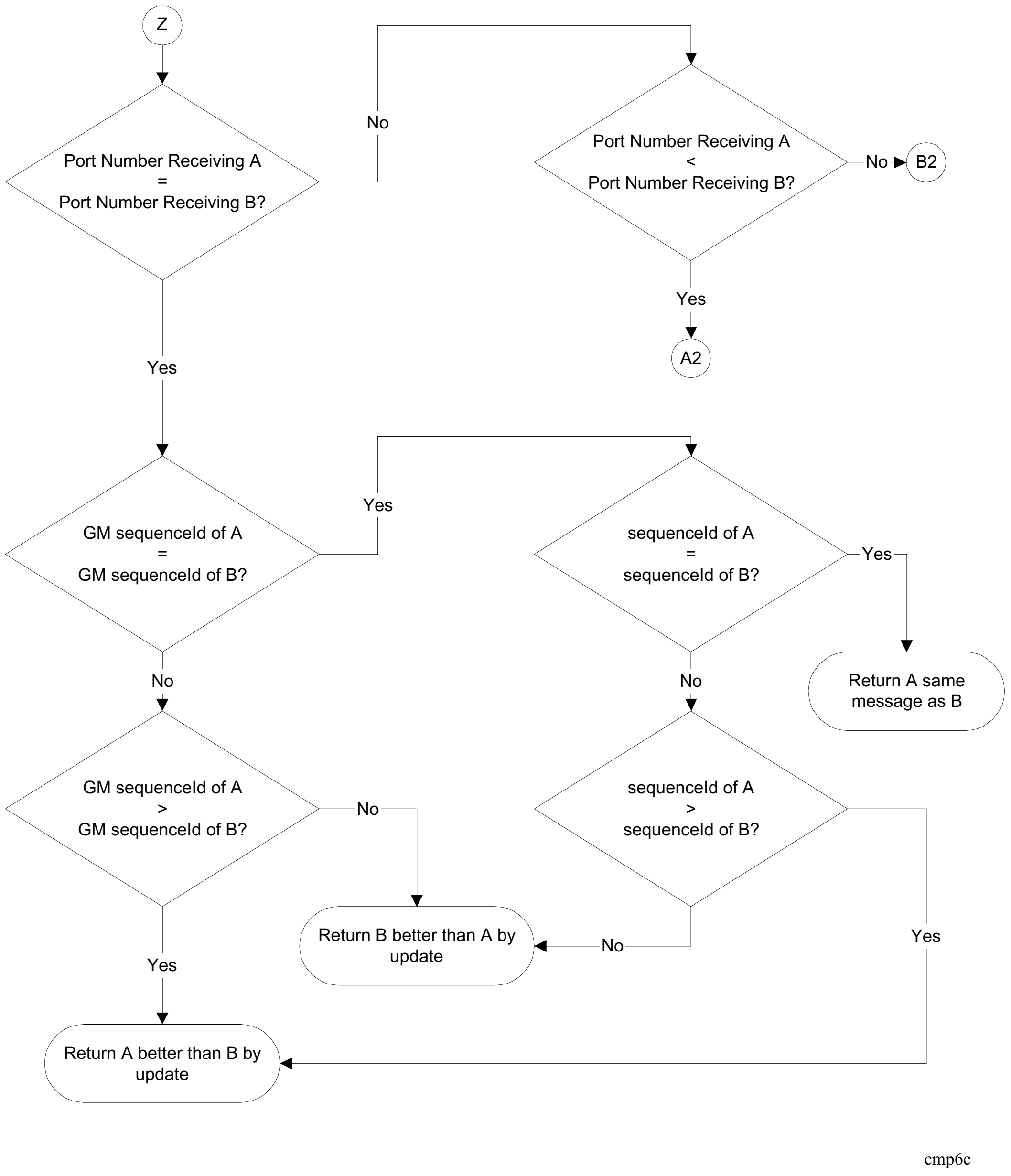
Z
Port Number Receiving A = Port Number Receiving B?
No
Port Number Receiving A < Port Number Receiving B?
No
B2
Yes
A2
Yes
GM sequenceId of A = GM sequenceId of B?
Yes
sequenceId of A = sequenceId of B?
Yes
Return A same message as B
No
No
GM sequenceId of A > GM sequenceId of B?
No
sequenceId of A > sequenceId of B?
Return B better than A by update
No
Yes
Yes
Return A better than B by update
cmp6c

Figure 19—Data set comparison algorithm, part 3

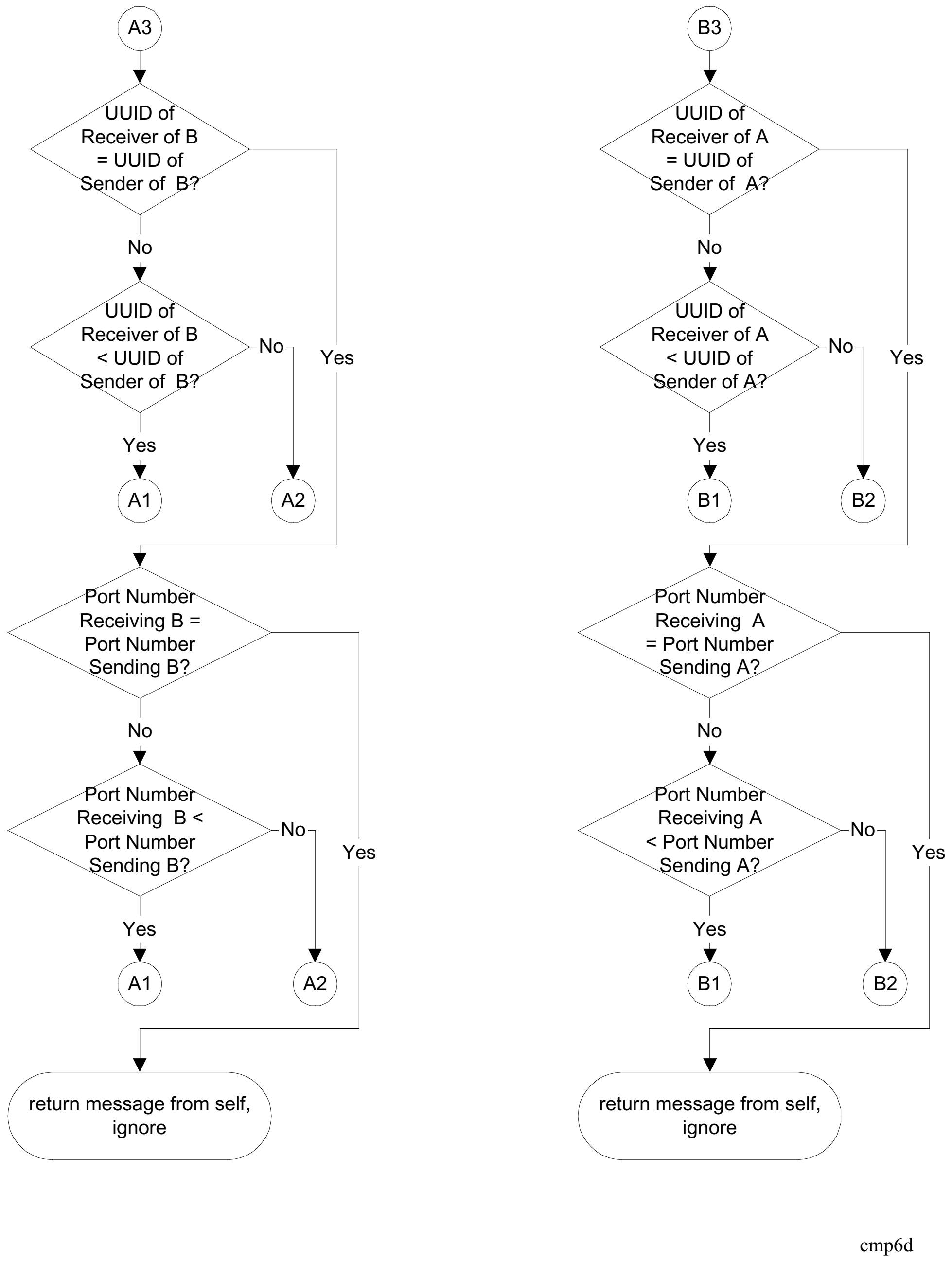
A3
UUID of Receiver of B = UUID of Sender of B?
No
Yes
UUID of Receiver of B < UUID of Sender of B?
No
Yes
A1
A2
Port Number Receiving B = Port Number Sending B?
No
Yes
Port Number Receiving B < Port Number Sending B?
No
Yes
A1
A2
return message from self, ignore
B3
UUID of Receiver of A = UUID of Sender of A?
No
Yes
UUID of Receiver of A < UUID of Sender of A?
No
Yes
B1
B2
Port Number Receiving A = Port Number Sending A?
No
Yes
Port Number Receiving A < Port Number Sending A?
No
Yes
B1
B2
return message from self, ignore
cmp6d

Figure 20—Data set comparison algorithm, part 4

Table 16—Information sources for data set comparison algorithm

When considering this property:	If the data set is E_{best} or E_{rbest}, use these fields of the associated message	If the data set is D_0, use these fields of the local clock
GMUUID	grandmasterCommunicationTechnology, grandmasterClockUuid, grandmasterPortId	clock_communication_technology, clock_uuid_field, clock_port_field
GMStratum	grandmasterClockStratum	clock_stratum
GMIdentifier	grandmasterClockIdentifier	clock_identifier
GMVariance	grandmasterClockVariance	clock_variance
GM is Boundary Clock	grandmasterIsBoundaryClock	is_boundary_clock
GM is Preferred Clock	grandmasterPreferred	preferred
Steps Removed	stepsRemoved	steps_removed
UUID of Sender	sourceCommunicationTechnology, sourceClockUuid, sourcePortId	clock_communication_technology, clock_uuid_field, clock_port_field
Port Number Receiving	port_id_field (of port database of port receiving message)	clock_port_field
GM sequenceId	grandmasterSequenceId	grandmaster_sequence_number of parent data set
sequenceId	sequenceId	grandmaster_sequence_number of parent data set
UUID of Receiver	port_communication_technology, port_uuid_field, port_id_field (all of port database of port receiving message)	clock_communication_technology, clock_uuid_field, clock_port_field
Port Number Sending	sourcePortId	clock_port_field

7.6.5 Update of data sets

The update of the current, parent, and global time data sets shall be as defined in Table 18, Table 19, Table 20, and Table 21 for the state decision codes, which are indicated by *Code* in Figure 16. The sources of update information are summarized in Table 17.

Table 17—Source of data set updates associated with state decision algorithm

State decision code	Source of update information
M1, M2	Default data set
M3	E_{best}
P1, P2, S1	E_{rbest}

Member port_state of the port configuration data set shall be updated as changes are made by the protocol engine of Figure 9 associated with each port in accordance with 7.5.8.

Data set fields which are not included in Table 18, Table 19, Table 20, and Table 21 are not updated due to requirements of this clause.

Table 18—Updates for state decision code M1 and M2

Update this field	From the indicated field of the default data set unless otherwise stated
Current data set	
steps_removed	set to 0
offset_from_master	set to 0
one_way_delay	set to 0
Parent data set	
parent_communication_technology	clock_communication_technology
parent_uuid	clock_uuid_field
parent_port_id	clock_port_field
parent_last_sync_sequence_number	set to 0
parent_followup_capable	clock_followup_capable
parent_external_timing	external_timing
parent_variance	clock_variance
grandmaster_communication_technology	clock_communication_technology
grandmaster_uuid_field	clock_uuid_field
grandmaster_port_id_field	clock_port_field
grandmaster_stratum	clock_stratum
grandmaster_identifier	clock_identifier
grandmaster_variance	clock_variance
grandmaster_preferred	preferred
grandmaster_is_boundary_clock	is_boundary_clock
grandmaster_sequence_number	see 7.4.4.20
Port configuration data set	
port_state	see 7.3.1

Table 19— Updates for state decision code M3

Update this field	From the indicated source
Port configuration data set	
port_state	see 7.3.1

Table 20—Updates for state decision code P1 and P2

Update this field	From the indicated source
Port configuration data set	
port_state	see 7.3.1

Table 21—Updates for state decision code S1

Update this field	From the indicated source
Current data set	
steps_removed	1 + value of localStepsRemoved of E_{best}
Parent data set	
parent_communication_technology	sourceCommunicationTechnology of E_{best}
parent_uuid	sourceUuid of E_{best}
parent_port_id	sourcePortId of E_{best}
parent_last_sync_sequence_number	sequenceId of E_{best}
parent_followup_capable	logical value of PTP_ASSIST bit of flags field of E_{best}
parent_external_timing	logical value of PTP_EXT_SYNC bit of flags field of E_{best}
parent_variance	localClockVariance of E_{best}
grandmaster_communication_technology	grandmasterCommunicationTechnology of E_{best}
grandmaster_uuid_field	grandmasterClockUuid of E_{best}
grandmaster_port_id_field	grandmasterPortId of E_{best}
grandmaster_stratum	grandmasterClockStratum of E_{best}
grandmaster_identifier	grandmasterClockIdentifier of E_{best}
grandmaster_variance	grandmasterClockVariance of E_{best}
grandmaster_preferred	grandmasterPreferred of E_{best}
grandmaster_is_boundary_clock	grandmasterIsBoundaryClock of E_{best}
grandmaster_sequence_number	grandmasterSequenceId of E_{best}
Global time properties data set	
current_utc_offset	currentUTCOffset of E_{best}
leap_59	logical value of PTP_LI_59 bit of flags field of E_{best}

Table 21—Updates for state decision code S1 ***(continued)***

Update this field	From the indicated source
leap_61	logical value of PTP_LI_61 bit of flags field of E_{best}
epoch_number	epochNumber of E_{best}
Port configuration data set	
port_state	see 7.3.1

7.6.6 Ordering in the data set comparison algorithm

The ordering of two UUIDs shall be as specified in 6.2.4.1.

The ordering of two clock_identifiers shall be as specified in Figure 21. The clock_identifiers specified in Figure 21 are the only clock_identifiers allowed by 6.2.4.5. Implementations using other clock_identifiers shall assign an order number of 4 (see Figure 21). The behavior of systems containing clocks with non-conformant clock_identifiers is outside the scope of this standard.

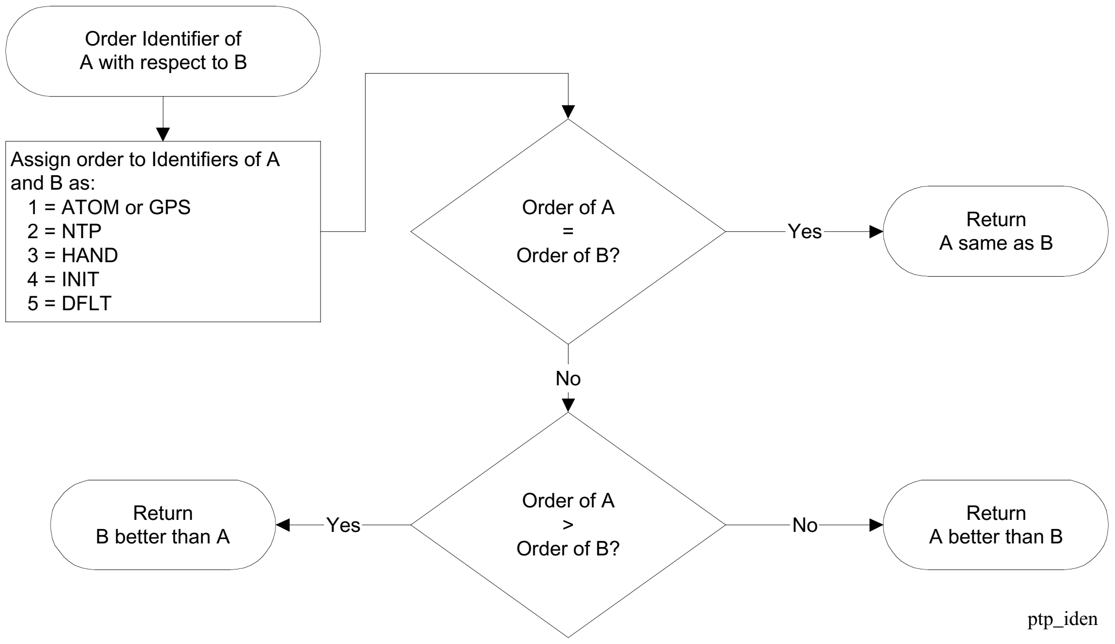

Figure 21—Ordering of clock_identifiers

The ordering of two variances shall be as specified in Figure 22. Note that the ordering algorithm operates on the scaled, logarithmic representations of the variances (see 7.7.2).

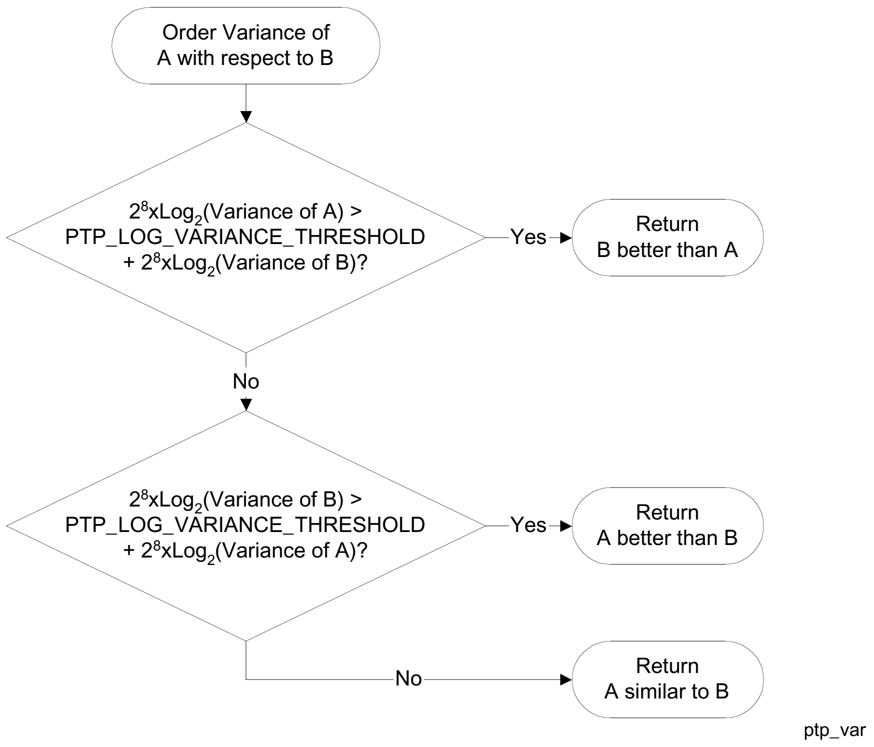
Order Variance of A with respect to B
2⁸xLog₂(Variance of A) > PTP_LOG_VARIANCE_THRESHOLD + 2⁸xLog₂(Variance of B)?
Yes
Return B better than A
No
2⁸xLog₂(Variance of B) > PTP_LOG_VARIANCE_THRESHOLD + 2⁸xLog₂(Variance of A)?
Yes
Return A better than B
No
Return A similar to B
ptp_var

Figure 22—Ordering of variances

7.7 Clock variance computation

This clause defines the methods of computing the statistical properties of PTP clocks used in certain message field values and algorithms.

7.7.1 Variance algorithm

The PTP variance representing the performance of the appropriate clock shall be computed based on the theory of Allan deviation as follows:

The Allan deviation $\sigma_y(\tau)$ is defined as follows [B5]:

$$\sigma_y(\tau) = \left[\frac{1}{2(N-2)\tau^2} \times \sum_{k=1}^{N-2} (x_{k+2} - 2x_{k+1} + x_k)^2\right]^{\frac{1}{2}}$$

where x_k, x_{k+1}, and x_{k+2} are time residual measurements made at times t_k, $t_k + \tau$, and $t_k + 2\tau$ and τ is the sample period between measurements. The term residual implies that any consistent systematic effects have been removed from the data.

The Allan deviation as stated gives a statistic on the variation of the frequency of the oscillator used as the basis of the time base. PTP algorithms are based on statistics of the time differences as measured against a reference clock, rather than Allan frequency statistics.

The PTP variance, $\sigma^2_{PTP} = \tau^2 \times \frac{1}{3}\sigma_y^2$, shall be as follows:

$$\sigma^2_{PTP} = \frac{1}{3}\left[\frac{1}{2(N-2)} \times \sum_{k=1}^{N-2} (x_{k+2} - 2x_{k+1} + x_k)^2\right]$$

where x_k, x_{k+1}, and x_{k+2} are time residual measurements, made at times t_k, $t_k + \tau$, and $t_k + 2\tau$, between the time provided by the measured clock and a local reference clock. For PTP variances τ, the sample period shall be the sync interval.

Implementations may compute a conservative estimate of the PTP variance rather than computing the exact value given here. Note that this may be necessary in implementations with limited computational or memory resources [B9].

NOTE—The dependence of the Allan deviation on the sample period provides information on the type of the underlying noise processes. The Allan deviation is not sensitive to systematic offsets in time or in frequency, even though those offsets may be important in some applications of this standard. In addition, the Allan deviation does not provide a useful diagnostic when the noise spectrum contains *bright lines*—power line– induced variations at 60 Hz, for example. Finally, the Allan deviation is computed as an average over the ensemble of observations, and it is most useful when the data are statistically stationary. The deviation does not provide a good measure of the frequency or amplitude of occasional glitches, even though those sorts of events might also be important in some applications of this standard.

7.7.2 Variance representation

Variances shall be represented as follows:

- An estimate of the variance σ^2_{PTP} specified in 7.7.1 shall be computed.
- The value of the logarithm to the base 2 of this computed estimate shall be computed. The computation of the logarithm need not be more precise than the precision of the estimate of the variance.
- This value shall be multiplied by 2^8 to produce a scaled value.
- This scaled value shall be modified per the hysteresis specification of this clause to produce the reported value.
- This reported value, represented as an Integer16, shall be the value of the variances specified in 7.4.2.6 and 7.4.4.9.

This representation ensures that the ordering of variances algorithm of 7.6.6 produces identical results in all implementations. (This cannot be guaranteed with a floating-point representation).

Since variance values are used in the selection of the best master clock (see 7.6.6), implementations shall include hysteresis in the estimation of the variance to preclude thrashing in the process of selecting the master clock. The magnitude of this hysteresis shall be PTP_LOG_VARIANCE_HYSTERESIS and shall be applied to the scaled values of the logarithm of the estimate to generate the reported scaled values of the logarithm of the estimate as shown in Figure 23. Sufficient local state shall be maintained to allow correct implementations of the hysteresis properties for both increasing and decreasing trends in the actual variance estimate.

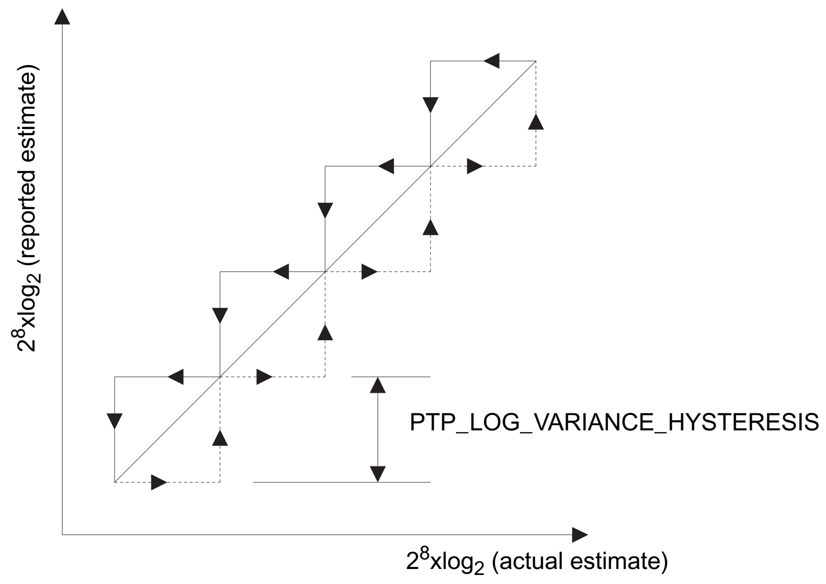
$2^8 x\log_2$ (reported estimate)
PTP_LOG_VARIANCE_HYSTERESIS
$2^8 x\log_2$ (actual estimate)

Figure 23—Log variance hysteresis

7.7.3 Computation of clock_variance

The computation of the value of clock_variance for the default data set shall be based on the characteristics of the local clock. The value of clock_variance may:

— Be a static constant determined by the manufacture, or
— Be computed based on measured or modeled behavior of the components of the local clock and its environment.

The value of clock_variance shall be computed and represented per 7.7.1 and 7.7.2.

The value clock_variance shall be an estimate of the variations of the local clock from a linear time scale when it is not synchronized to another PTP clock using the PTP protocol. The primary source of time in this case may be an atomic clock, a GPS receiver, a stable local oscillator, a suite of clocks synchronized via NTP, etc. These sources may contribute to the variance estimate.

If the clock_identifier indicates that the time scale is UTC, the linear time scale used to estimate the clock_ variance shall be UTC time.

The value clock_variance shall be the variance applicable to an ensemble of measurements that include error contributions from:

— Synchronization to its primary source of time
— Drift and noise characteristics of the local oscillator
— Sampling quantization errors, fluctuations in inbound and outbound latency, etc. of the local clock

This field value may be a static value determined by the manufacturer based on measurements or analysis of the specific implementation when the local clock is not synchronized to some external source of time via a non-PTP protocol. If the local clock is synchronized to some external source of time via a non-PTP protocol, then the value may be static or dynamic depending on the implementation, but in any case shall represent the current statistical performance of the clock.

7.8 Local clock synchronization

This clause discusses synchronization of the local clock, either via PTP or by other means.

7.8.1 Local clock synchronized via PTP to another PTP clock

When an ordinary clock or a port of a boundary clock is in the PTP_SLAVE state, the local clock of the slave shall be synchronized to the local clock of the slave's master by minimizing the offset_from_master value of the current data set. The time and rate characteristics of the local clock shall be modified for purposes of synchronization upon receipt of either a Sync message or a Follow_Up message as specified in 7.5.3 and 7.5.4. The techniques used to modify the time and rate characteristics of the local clock shall be based on the offset_from_master and one_way_delay fields of the current data set but are otherwise outside the scope of this standard.

The offset_from_master and one_way_delay fields of the current data set shall be updated whenever new values of these fields are computed as required by this clause.

Note that many factors may affect the accuracy in synchronizing a local clock to its parent including:

— The choice of sync interval (see 6.2.5.5). This choice represents a trade-off between network and computational load and the inherent stability of the local and parent clocks.
— The stability of the local oscillator driving the local and parent clocks. This stability may be adversely affected by variation in supply voltages, temperature, acceleration and possibly other environmental and implementation factors.

7.8.1.1 Computation of offset_from_master

The offset_from_master value shall be computed as follows.

If the computation is caused as a result of the receipt of a Sync message (see 7.5.3), then

offset_from_master = (sync_receipt_time) – (originTimestamp) – (one_way_delay).

If the computation is caused as a result of the receipt of a Follow_Up message (see 7.5.4), then

offset_from_master = (sync_receipt_time) – (preciseOriginTimestamp) – (one_way_delay).

Where:

— (sync_receipt_time) shall be the time at which the Sync message was received by the slave clock as measured by the local clock of the slave at the designated point in the inbound protocol stack (see Figure 4).
— (originTimestamp) shall be the value of the originTimestamp field in the received Sync message (see 8.3.1.2).
— (preciseOriginTimestamp) shall be the value of the field with that name in the received Follow_Up message (see 8.4.1.3).
— (one_way_delay) shall be as specified in 7.8.1.2.

7.8.1.2 Computation of one_way_delay

The one_way_delay value shall be computed as follows.

(one_way_delay) = {(master_to_slave_delay) + (slave_to_master_delay)}/2.

This computation assumes that the path lengths are identical in the two directions. If additional knowledge is available concerning any possible path asymmetry, this computation shall be corrected so that the value of (one_way_delay) more accurately represents the time of propagation of a Sync message from the master to the slave clock.

7.8.1.2.1 Computation of master_to_slave_delay

The master_to_slave_delay value shall be computed as follows.

If the computation is caused as a result of the receipt of a Sync message (see 7.5.3), then

master_to_slave_delay = (sync_receipt_time) – (originTimestamp).

If the computation is caused as a result of the receipt of a Follow_Up message (see 7.5.4), then

master_to_slave_delay = (sync_receipt_time) – (preciseOriginTimestamp).

Where:

- (sync_receipt_time) shall be the time at which the Sync message was received by the slave clock as measured by the local clock of the slave at the designated point in the inbound protocol stack (see Figure 4).
- (originTimestamp) shall be the value of the originTimestamp field in the received Sync message (see 8.3.1.2).
- (preciseOriginTimestamp) shall be the value of the field with that name in the received Follow_Up message (see 8.4.1.3).

7.8.1.2.2 Computation of slave_to_master_delay

The slave_to_master_delay value shall be computed as follows:

The computation is caused as a result of the receipt of a Delay_Resp message.

slave_to_master_delay = (delayReceiptTimestamp) – (delay_req_sending_time).

Where:

- (delayReceiptTimestamp) shall be the value of the field with that name in the received Delay_Resp message (see 8.5.1.2).
- (delay_req_sending_time) shall be the time at which the Delay_Req message was sent by the slave clock as measured by the local clock of the slave at the designated point in the outbound protocol stack (see Figure 4).

7.8.1.2.3 Example computation of one_way_delay

The computation of one_way_delay is illustrated in Figure 24 and Table 22 with the assumption that the PTP communication path delay is the same in both directions.

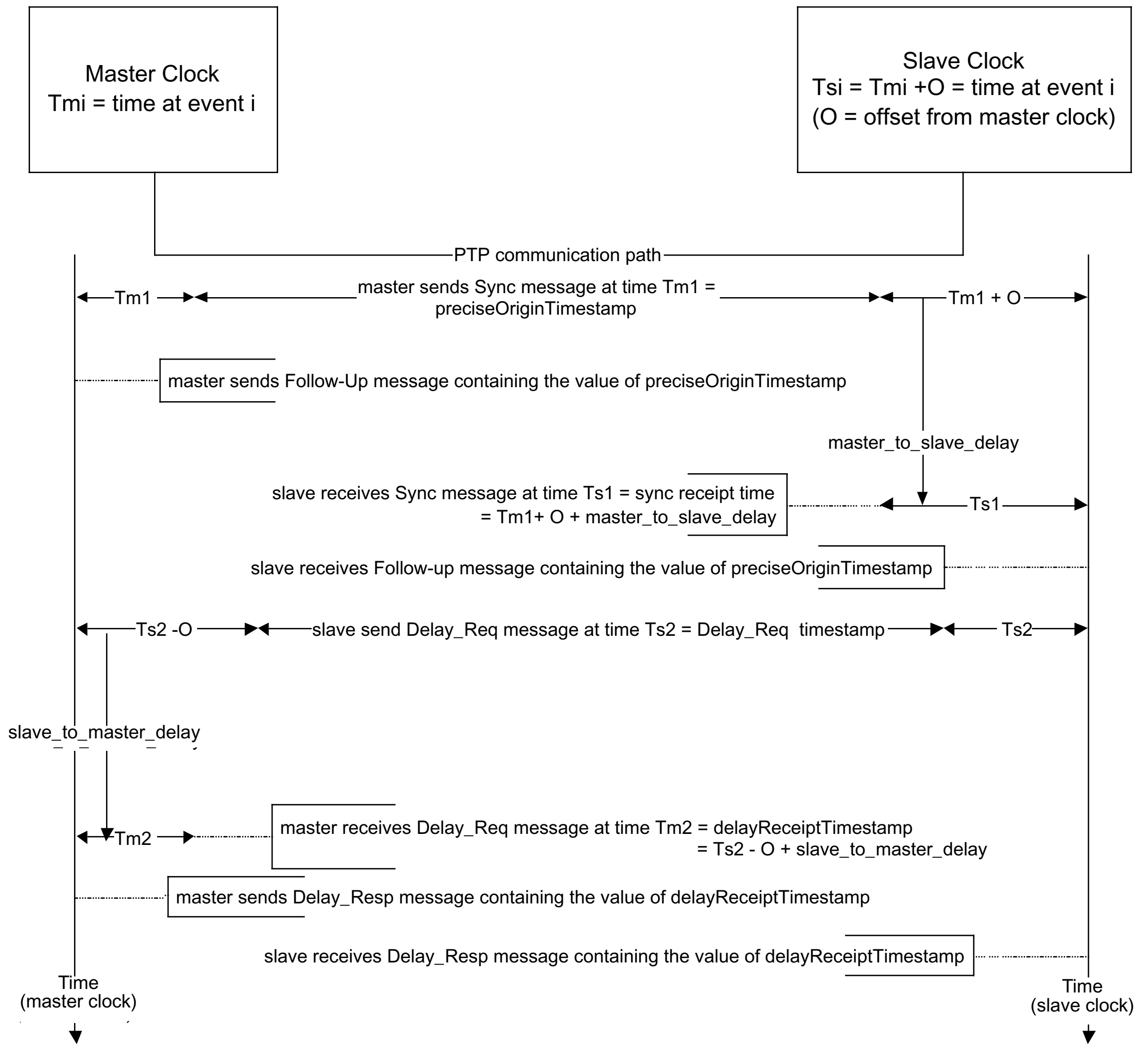
Master Clock
Tmi = time at event i
Slave Clock
Tsi = Tmi +O = time at event i
(O = offset from master clock)
PTP communication path
Tm1
master sends Sync message at time Tm1 = preciseOriginTimestamp
Tm1 + O
master sends Follow-Up message containing the value of preciseOriginTimestamp
master_to_slave_delay
slave receives Sync message at time Ts1 = sync receipt time = Tm1+ O + master_to_slave_delay
Ts1
slave receives Follow-up message containing the value of preciseOriginTimestamp
Ts2 -O
slave send Delay_Req message at time Ts2 = Delay_Req timestamp
Ts2
slave_to_master_delay
Tm2
master receives Delay_Req message at time Tm2 = delayReceiptTimestamp = Ts2 - O + slave_to_master_delay
master sends Delay_Resp message containing the value of delayReceiptTimestamp
slave receives Delay_Resp message containing the value of delayReceiptTimestamp
Time (master clock)
Time (slave clock)

Figure 24—Current delay to master computation example

7.8.1.3 Latency corrections

All timestamps associated with outbound Sync or Delay_Req messages shall be corrected for outbound_ latency (see 6.2.4.9). These timestamps are:

— originTimestamp (see 8.3.1.2)
— preciseOriginTimestamp (see 8.4.1.3)
— delay_req_sending_time (see 7.8.1.2.2)

The reported value for these timestamps shall be computed as follows.

— The Sync or Delay_Req sending timestamp measured by the local clock at the clock timestamp point of the outbound protocol stack shall be recorded as <measured value>.
— The reported value of the timestamp shall be <measured value> + outbound_latency.

Table 22—Current delay to master computation example

Term	Value
sync_receipt_time = Ts1	Ts1 =Tm1+O+master_to_slave_delay
preciseOriginTimestamp = Tm1	Tm1
master_to_slave_delay (computed)	Ts1–Tm1
delay_req_sending_time = Ts2	Ts2
delayReceiptTimestamp = Tm2	Tm2 = Ts2 – O + slave_to_master_delay
slave_to_master_delay (computed)	= Tm2 – Ts2
one_way_delay	= {(master_to_slave_delay as computed) + (slave_to_master_delay as computed)}/2 = {(Ts1-Tm1) + (Tm2-Ts2)}/2 ={(O + master_to_slave_delay) + (–O +slave_to_master_delay)}/2 ={(master_to_slave_delay) + (slave_to_master_delay)}/2 = master_to_slave_delay if path is symmetrical

All timestamps associated with inbound Sync or Delay_Req messages shall be corrected for inbound_ latency (see 6.2.4.9). These timestamps are:

— sync_receipt_time (see 7.8.1.1)
— delayReceiptTimestamp (see 8.5.1.2)

The reported value for these timestamps shall be computed as follows:

— The receipt timestamp measured by the local clock at the clock timestamp point of the inbound protocol stack shall be recorded as <measured value>.
— The reported value of the timestamp shall be <measured value> – inbound_latency.

NOTE—The failure to correct for latency will not be apparent if all clock implementations have identical values for the latency constants. However, if two clock implementations do not have the same values of the latency constants, difference will appear as offset between the two clocks.

7.8.2 Local clock synchronized directly to another clock

A local clock with no ports in the PTP_SLAVE state may synchronize directly with an external clock (see 7.1 and 7.4.3). If such external synchronization is in effect, the data set updates shall be accomplished as in Table 23, as part of the synchronization mechanism.

7.9 Values for system and clock constants.

The several PTP defined constants shall have values indicated in Table 24.

Note that the value of sync_interval may be modified (see 7.4.2.12 and 6.2.5.5). The values of PTP_SYNC_ INTERVAL_TIMEOUT, PTP_SYNC_RECEIPT_TIMEOUT, and PTP_FOREIGN_MASTER_TIME_ WINDOW shall be updated whenever the underlying value of sync_interval is changed or set to the default value. From 7.4.2.12, sync_interval is the logarithm to the base 2 of the sync interval when the sync interval is measured in seconds.

Table 23— External synchronization updates

Update the indicated fields	Comment
Default data set	
clock_stratum	To reflect the characteristics of the external clock and synchronization mechanism
clock_identifier	
clock_variance	
Global time properties data set	
current_utc_offset	If available, update based on information provided by the external clock
leap_59	
leap_61	
epoch_number	

Table 24—PTP constants

PTP constant	Value	Units	Reference(s)
PTP_UUID_LENGTH	6	octets	6.2.4.1
PTP_CODE_STRING_LENGTH	4	octets	6.2.4.5
PTP_SUBDOMAIN_NAME_LENGTH	16	octets	6.2.5.1
PTP_MAX_MANAGEMENT_PAYLOAD_SIZE	90	octets	8.6.1.8
PTP_SYNC_INTERVAL_TIMEOUT	$2^{\text{sync\_interval}}$	seconds	7.5.15
PTP_SYNC_RECEIPT_TIMEOUT	$10 \times 2^{\text{sync\_interval}}$	seconds	7.5.14
PTP_DELAY_REQ_INTERVAL	30	n/a	7.11
PTP_FOREIGN_MASTER_THRESHOLD	2	n/a	7.6.2
PTP_FOREIGN_MASTER_TIME_WINDOW	$4 \times 2^{\text{sync\_interval}}$	seconds	7.6.2
PTP_RANDOMIZING_SLOTS	18	N/A	7.11
PTP_LOG_VARIANCE_THRESHOLD	2^8	$2^8\log_2(\text{second}^2)$	7.6.6
PTP_LOG_VARIANCE_HYSTERESIS	2^7	$2^8\log_2(\text{second}^2)$	7.7.2

The behavior of PTP systems with constant values other than those specified in Table 24 is outside the scope of this standard. The following constraints on these constants shall be observed even in systems with constant values other than those specified in Table 24.

— PTP_FOREIGN_MASTER_TIME_WINDOW ≥ PTP_FOREIGN_MASTER_THRESHOLD × PTP_SYNC_INTERVAL_TIMEOUT

— PTP_SYNC_RECEIPT_TIMEOUT ≥ (PTP_FOREIGN_MASTER_TIME_WINDOW) + 2(PTP_SYNC_INTERVAL_TIMEOUT)

— PTP_MAX_MANAGEMENT_PAYLOAD_SIZE ≥ 32

7.10 Physical requirements for PTP implementations

A clock participating in the PTP protocol shall meet the physical constraints specified in the subclauses of this clause.

7.10.1 Oscillator frequency

The oscillator serving as the time base for a PTP clock shall have an accuracy and stability such that a second measured in the time base of the PTP clock is within ± 0.01% of the SI second (see Annex B). This specification shall apply over the rated temperature range of the PTP clock.

The stability of the oscillator shall be consistent with the values of clock_identifier and clock_variance of the PTP clock.

7.10.2 Time adjustment range

A PTP clock in the PTP_SLAVE state shall be able to correct its time base to match any master clock meeting the requirements of 7.10.1.

NOTE—This requires an adjustment range of at least ± 0.02% on the rate of the clock.

7.11 PTP timing and event ordering constraints

A clock participating in the PTP protocol shall meet the processing constraints specified in the subclauses of this clause.

7.11.1 Processing of Sync messages

Received Sync messages shall be processed under the terms of 7.5.3 at a rate no less than two per PTP_SYNC_INTERVAL_TIMEOUT seconds.

When required under the terms of 7.5.9, a clock shall issue a Sync message within PTP_SYNC_INTERVAL_TIMEOUT/(2 × PTP_RANDOMIZING_SLOTS) seconds after the event requiring the issuance of the Sync message.

7.11.2 Processing of Follow_Up messages

Received Follow_Up messages shall be processed under the terms of 7.5.4, at a rate no less than one per PTP_SYNC_INTERVAL_TIMEOUT seconds.

When required under the terms of 7.5.10, a clock shall issue a Follow_Up message within PTP_SYNC_INTERVAL_TIMEOUT / PTP_RANDOMIZING_SLOTS seconds after the issuance of the corresponding Sync message.

7.11.3 Processing of Delay_Req messages

Received Delay_Req messages shall be processed under the terms of 7.5.5, at an average rate no less than (PTP_RANDOMIZING_SLOTS-2) messages per PTP_SYNC_INTERVAL_TIMEOUT seconds. The averaging interval shall be PTP_DELAY_REQ_INTERVAL × PTP_SYNC_INTERVAL_TIMEOUT seconds.

Note that this rate may be a limiting factor in the performance within a single PTP communication path of a PTP system.

A system of (PTP_RANDOMIZING_SLOTS-2) × PTP_DELAY_REQ_INTERVAL clocks within a single PTP communication path, implementing all of the functionality specified by this standard, will together generate Delay_Req messages at this average rate.

When required by a Management message under the terms of 7.5.11, a clock shall issue a Delay_Req message within PTP_SYNC_INTERVAL_TIMEOUT / PTP_RANDOMIZING_SLOTS seconds after the event requiring the issuance of the Delay_Req message.

When required for the delay computation under the terms of 7.5.11, a clock shall issue a Delay_Req message within a time window *T*, specified as follows:

- A clock shall generate two integer random numbers *R* and *Q* at entry into the PTP_SLAVE state and after issuing a Delay_Req message.
- *R* shall be in the closed interval 2 to PTP_DELAY_REQ_INTERVAL.
- *Q* shall be in the closed interval 2 to PTP_RANDOMIZING_SLOTS.
- The window shall occur in the time interval *P* beginning at the *R*th receipt after the generation of *R*, of a Sync message from the current master clock.
- Within the time interval *P*, the window *T* shall be the closed interval *Q*×Delta*T* to (*Q*+1) × Delta*T*.
- The time window width Delta*T* shall be PTP_SYNC_INTERVAL_TIMEOUT / PTP_RANDOMIZING_SLOTS seconds.

7.11.4 Processing of Delay_Resp messages

Received Delay_Resp messages shall be processed under the terms of 7.5.6 at an average rate no less than (PTP_RANDOMIZING_SLOTS-2) messages per PTP_SYNC_INTERVAL_TIMEOUT seconds. The averaging interval shall be PTP_DELAY_REQ_INTERVAL × PTP_SYNC_INTERVAL_TIMEOUT seconds.

Note that this rate may be a limiting factor in the performance within a single PTP communication path of a PTP system.

A system of (PTP_RANDOMIZING_SLOTS-2) × PTP_DELAY_REQ_INTERVAL clocks within a single PTP communication path implementing all of the functionality specified by this standard will together result in the master clock for the PTP communication path generating Delay_Resp messages at this average rate.

When required under the terms of 7.5.5 and 7.5.12, a clock shall issue a Delay_Resp message within PTP_SYNC_INTERVAL_TIMEOUT / (2 × PTP_RANDOMIZING_SLOTS) seconds after the event requiring the issuance of the Delay_Resp message.

7.11.5 Processing of Management messages

Received Management messages shall be processed under the terms of 7.5.7, at a rate no less than one per PTP_SYNC_INTERVAL_TIMEOUT seconds.

7.11.6 Timing and ordering summary (informative)

The time and order constraints of 7.11 are illustrated in Figure 25, Figure 26, and Figure 27.

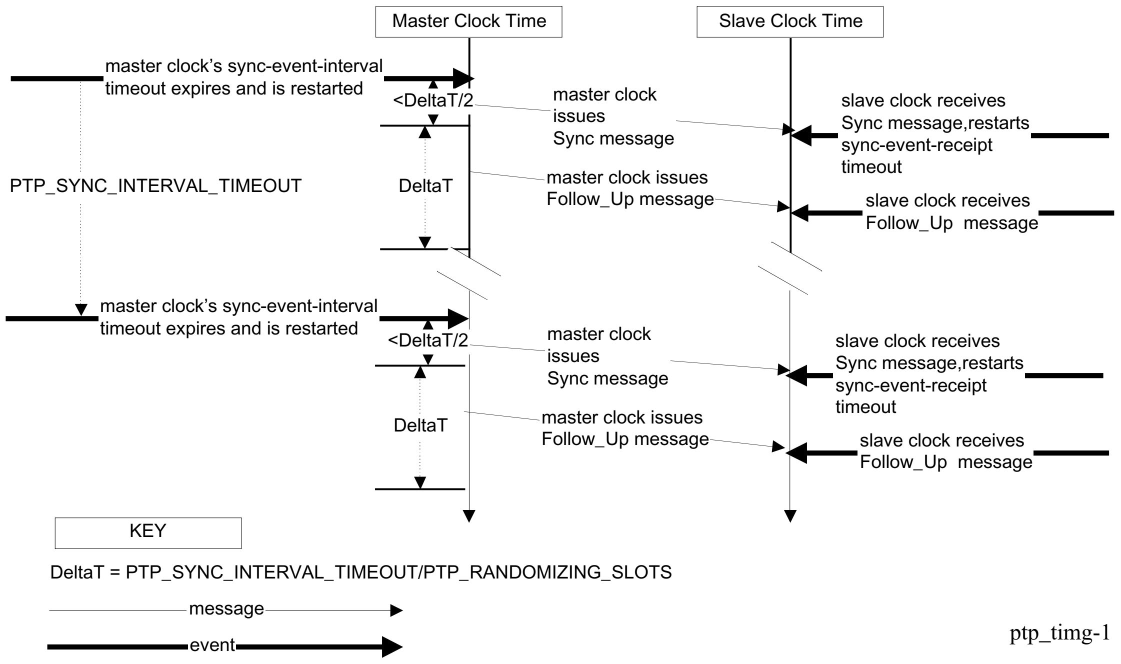
Master Clock Time
Slave Clock Time
master clock's sync-event-interval timeout expires and is restarted
<DeltaT/2
master clock issues Sync message
slave clock receives Sync message,restarts sync-event-receipt timeout
PTP_SYNC_INTERVAL_TIMEOUT
DeltaT
master clock issues Follow_Up message
slave clock receives Follow_Up message
master clock's sync-event-interval timeout expires and is restarted
<DeltaT/2
master clock issues Sync message
slave clock receives Sync message,restarts sync-event-receipt timeout
DeltaT
master clock issues Follow_Up message
slave clock receives Follow_Up message
KEY
DeltaT = PTP_SYNC_INTERVAL_TIMEOUT/PTP_RANDOMIZING_SLOTS
message
event
ptp_timg-1

Figure 25—Timing and Ordering of Sync and Follow_Up messages

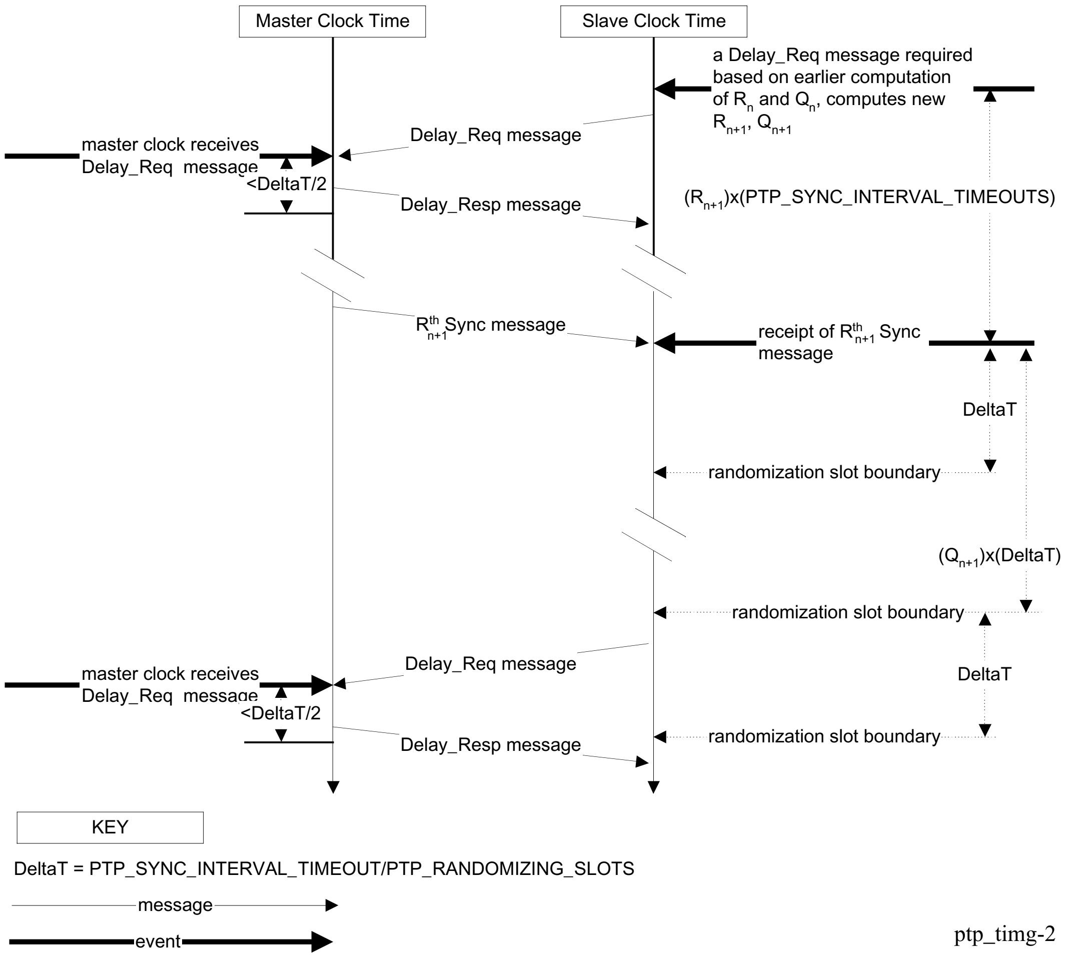
Master Clock Time
Slave Clock Time
a Delay_Req message required based on earlier computation of R_n and Q_n, computes new R_{n+1}, Q_{n+1}
master clock receives Delay_Req message
Delay_Req message
<DeltaT/2
Delay_Resp message
(R_{n+1})x(PTP_SYNC_INTERVAL_TIMEOUTS)
R^{th}_{n+1} Sync message
receipt of R^{th}_{n+1} Sync message
DeltaT
randomization slot boundary
(Q_{n+1})x(DeltaT)
randomization slot boundary
Delay_Req message
master clock receives Delay_Req message
<DeltaT/2
DeltaT
Delay_Resp message
randomization slot boundary
KEY
DeltaT = PTP_SYNC_INTERVAL_TIMEOUT/PTP_RANDOMIZING_SLOTS
message
event
ptp_timg-2

Figure 26—Timing and ordering of Delay_Req and Delay_Resp messages

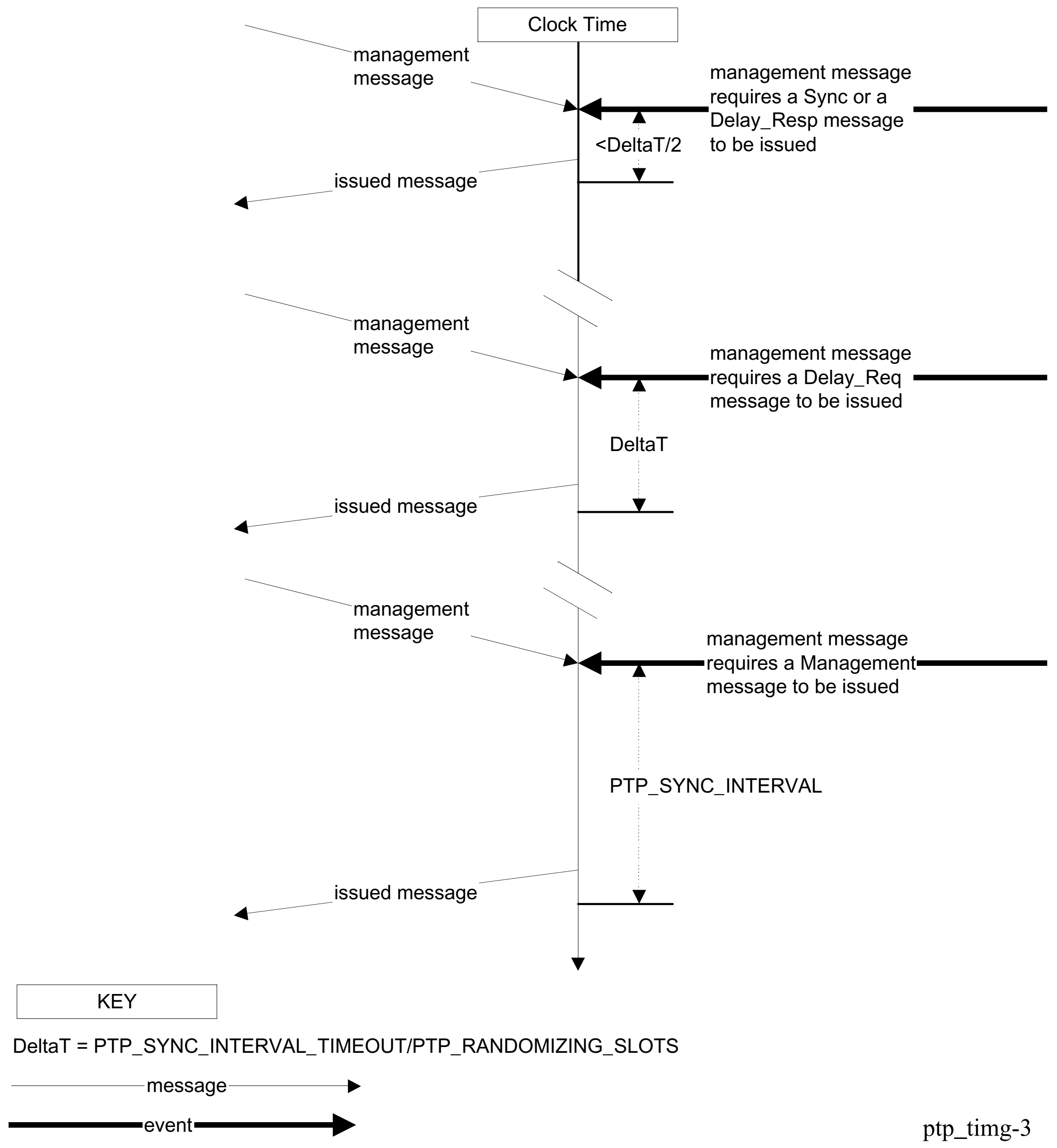
Clock Time
management message
management message requires a Sync or a Delay_Resp message to be issued
<DeltaT/2
issued message
management message
management message requires a Delay_Req message to be issued
DeltaT
issued message
management message
management message requires a Management message to be issued
PTP_SYNC_INTERVAL
issued message
KEY
DeltaT = PTP_SYNC_INTERVAL_TIMEOUT/PTP_RANDOMIZING_SLOTS
message
event
ptp_timg-3

Figure 27—Timing and ordering of management messages

7.12 Management message semantics

This clause defines the management messages that shall be supported in a PTP clock. Management message syntax is specified in 8.6. Management messages are distinguished by the value of the managementMessageKey field, 8.6.1.7, which is taken from the following enumeration.

IDL: enumeration ManagementMessage;

Table 25—ManagementMessage enumeration

Enumeration	Value (decimal)
PTP_MM_NULL	0
PTP_MM_OBTAIN_IDENTITY	1
PTP_MM_CLOCK_IDENTITY	2
PTP_MM_INITIALIZE_CLOCK	3
PTP_MM_SET_SUBDOMAIN	4
PTP_MM_CLEAR_DESIGNATED_PREFERRED_MASTER	5
PTP_MM_SET_DESIGNATED_PREFERRED_MASTER	6
PTP_MM_GET_DEFAULT_DATA SET	7
PTP_MM_DEFAULT_DATA SET	8
PTP_MM_UPDATE_DEFAULT_DATA SET	9
PTP_MM_GET_CURRENT_DATA SET	10
PTP_MM_CURRENT_DATA SET	11
PTP_MM_GET_PARENT_DATA SET	12
PTP_MM_PARENT_DATA SET	13
PTP_MM_GET_PORT_DATA SET	14
PTP_MM_PORT_DATA SET	15
PTP_MM_GET_GLOBAL_TIME_DATA SET	16
PTP_MM_GLOBAL_TIME_DATA SET	17
PTP_MM_UPDATE_GLOBAL_TIME_PROPERTIES	18
PTP_MM_GOTO_FAULTY_STATE	19
PTP_MM_GET_FOREIGN_DATA SET	20
PTP_MM_FOREIGN_DATA SET	21
PTP_MM_SET_SYNC_INTERVAL	22
PTP_MM_DISABLE_PORT	23
PTP_MM_ENABLE_PORT	24
PTP_MM_DISABLE_BURST	25
PTP_MM_ENABLE_BURST	26
PTP_MM_SET_TIME	27
Reserved	28–127
Implementation-specific	128–255

Management messages may be directed to all clocks or a specific clock within a subdomain by setting appropriate values for the targetUuid and targetCommunicationTechnology fields of the Management message. In the following subclauses the term *receipt* shall be interpreted to mean that the receiving clock act on the message provided that the requirements of 7.5.7 are met.

The interpretation, if required, of the targetPortId field of the Management message shall be as defined in the following subclauses.

The interpretation of the boundaryHops field of the Management message shall be as specified in 6.2.2.1.

The semantics of management messages are defined in the following subclauses.

7.12.1 PTP_MM_NULL

The receipt of a management message with a managementMessageKey field value PTP_MM_NULL shall have no effect on a clock.

7.12.2 PTP_MM_OBTAIN_IDENTITY

The receipt of a management message with a managementMessageKey field value PTP_MM_OBTAIN_IDENTITY shall cause the clock to issue a Management message with a managementMessageKey field of PTP_MM_CLOCK_IDENTITY.

7.12.3 PTP_MM_CLOCK_IDENTITY

The result of receiving a management message with a managementMessageKey field value PTP_MM_CLOCK_IDENTITY is implementation dependent.

7.12.4 PTP_MM_INITIALIZE_CLOCK

If the initializable field in the receiving clock's default data set is TRUE, the receipt of a management message with a managementMessageKey field value PTP_MM_INITIALIZE_CLOCK shall cause the clock to execute the behavior normally executed on a reboot or power cycle of the node containing the clock (see 7.3). Otherwise the receipt of this message shall have no effect.

The messageParameters field, initializationKey, of this message shall be interpreted as follows:

— *Value 0:* The specification initialization set shall be used (see 7.4.1).
— *Value 1:* The initialization set stored in nonvolatile storage for recovery after a power fail shall be used.
— *Values >1:* implementation specific
— If the value of the initializationKey is not interpretable by the receiving clock, the specification initialization set shall be used.

7.12.5 PTP_MM_SET_SUBDOMAIN

The receipt of a management message with a managementMessageKey field value PTP_MM_SET_SUBDOMAIN shall cause the clock to update the value of the subdomain_name in its default data set to the value given in the messageParameters field, subdomainName. The update of the value of subdomain_name shall not take effect until one of the following events occurs:

— Receipt of a management message with a managementMessageKey field value PTP_MM_INITIALIZE_CLOCK

— A reboot or power cycle of the node containing the clock.

7.12.6 PTP_MM_CLEAR_DESIGNATED_PREFERRED_MASTER

The receipt of a management message with a managementMessageKey field value PTP_MM_CLEAR_ DESIGNATED_PREFERRED_MASTER shall cause the clock to set the value of preferred in the default data set to FALSE.

7.12.7 PTP_MM_SET_DESIGNATED_PREFERRED_MASTER

The receipt of a management message with a managementMessageKey field value PTP_MM_SET_ DESIGNATED_PREFERRED_MASTER shall cause the clock to set the value of preferred in the default data set to TRUE.

7.12.8 PTP_MM_GET_DEFAULT_DATA_SET

The receipt of a management message with a managementMessageKey field value PTP_MM_GET_ DEFAULT_DATA_SET shall cause the clock to issue a management message with a managementMessageKey field of PTP_MM_DEFAULT_DATA_SET.

7.12.9 PTP_MM_DEFAULT_DATA_SET

The result of receiving a management message with a managementMessageKey field value PTP_MM_ DEFAULT_DATA_SET is implementation dependent.

7.12.10 PTP_MM_UPDATE_DEFAULT_DATA_SET

The receipt of a management message with a managementMessageKey field value PTP_MM_UPDATE_ DEFAULT_DATA_SET shall cause the receiving clock to update the default data set fields as defined in Table 26.

Table 26—messageParameters fields for PTP_MM_UPDATE_DEFAULT_DATA_SET

Update default data set member	Using messageParameters field
clock_stratum	clockStratum
clock_identifier	clockIdentifier
clock_variance	clockVariance
preferred	preferred
sync_interval	syncInterval
subdomain_name	subdomainName

The update of the value of subdomain_name and sync_interval shall not take effect until one of the following events occur:

— Receipt of a management message with a managementMessageKey field value PTP_MM_ INITIALIZE_CLOCK

— A reboot or power cycle of the node containing the clock.

7.12.11 PTP_MM_GET_CURRENT_DATA_SET

The receipt of a management message with a managementMessageKey field value PTP_MM_GET_CURRENT_DATA_SET shall cause the clock to issue a management message with a managementMessageKey field of PTP_MM_CURRENT_DATA_SET.

7.12.12 PTP_MM_CURRENT_DATA_SET

The result of receiving a management message with a managementMessageKey field value PTP_MM_CURRENT_DATA_SET is implementation dependent.

7.12.13 PTP_MM_GET_PARENT_DATA_SET

The receipt of a management message with a managementMessageKey field value PTP_MM_GET_PARENT_DATA_SET shall cause the clock to issue a management message with a managementMessageKey field of PTP_MM_PARENT_DATA_SET.

7.12.14 PTP_MM_PARENT_DATA_SET

The result of receiving a management message with a managementMessageKey field value PTP_MM_PARENT_DATA_SET is implementation dependent.

7.12.15 PTP_MM_GET_PORT_DATA_SET

The receipt of a management message with a managementMessageKey field value PTP_MM_GET_PORT_DATA_SET shall cause the clock to issue a management message with a managementMessageKey field of PTP_MM_PORT_DATA_SET. The value of the targetPortId field shall be interpreted as the port number identifying the specific port configuration data set to be sent in the subsequent PTP_MM_PORT_DATA_SET message. Values of targetPortId greater than the total number of ports, number_ports of the default data set, or zero shall cause this message to be ignored.

7.12.16 PTP_MM_PORT_DATA_SET

The result of receiving a management message with a managementMessageKey field value PTP_MM_PORT_DATA_SET is implementation dependent.

7.12.17 PTP_MM_GET_GLOBAL_TIME_DATA_SET

The receipt of a management message with a managementMessageKey field value PTP_MM_GET_GLOBAL_DATA_SET shall cause the clock to issue a management message with a managementMessageKey field of PTP_MM_GLOBAL_TIME_DATA_SET.

7.12.18 PTP_MM_GLOBAL_TIME_DATA_SET

The result of receiving a management message with a managementMessageKey field value PTP_MM_GLOBAL_TIME_DATA_SET is implementation dependent.

7.12.19 PTP_MM_UPDATE_GLOBAL_TIME_PROPERTIES

The receipt a management message with a managementMessageKey field value PTP_MM_UPDATE_GLOBAL_TIME_PROPERTIES shall cause the receiving clock to update the global time properties data set fields as defined in Table 27.

Table 27—messageParameters fields for PTP_MM_UPDATE_GLOBAL_TIME_PROPERTIES

Update global time properties data set member	Using messageParameters field
current_utc_offset	currentUtcOffset
leap_59	leap59
leap_61	leap61
epoch_number	epochNumber

NOTE—Normally this message is designated to be acted on by the grandmaster clock of a subdomain. While it may be sent to and received by any clock in the subdomain, any changes in the data set of a clock that is not the grandmaster will be overwritten as the information from the grandmaster clock is propagated down the clock hierarchy on subsequent synchronization cycles.

7.12.20 PTP_MM_GOTO_FAULTY_STATE

The receipt of a management message with a managementMessageKey field value PTP_MM_GOTO_ FAULTY_STATE shall cause the port whose port_id_field of the port configuration data set matches the targetPortId of the received Management message to process a FAULT_DETECTED event transition in the state machine associated with the port (see 7.3). A port_id_field value of 0 shall cause all ports of a clock to respond to the message.

7.12.21 PTP_MM_GET_FOREIGN_DATA_SET

The receipt of a management message with a managementMessageKey field value PTP_MM_GET_ FOREIGN_DATA_SET shall cause the clock to issue a management message with a managementMessageKey field of PTP_MM_FOREIGN_DATA_SET.

The value of the targetPortId field shall be interpreted as the port number identifying the specific foreign master data set to be sent in the subsequent PTP_MM_FOREIGN_DATA_SET message. Values of targetPortId greater than the total number of ports, number_ports of the default data set, or zero shall cause this message to be ignored.

The value of the messageParameters field, recordKey, shall be interpreted as the record number identifying the specific record of the foreign master data set of the identified port to be sent in the subsequent PTP_ MM_FOREIGN_DATA_SET message. Values of recordKey greater than the total number of records, number_foreign_records of the default data set, or zero shall cause this message to be ignored.

7.12.22 PTP_MM_FOREIGN_DATA_SET

The result of receiving a management message with a managementMessageKey field value PTP_MM_ FOREIGN_DATA_SET is implementation dependent.

7.12.23 PTP_MM_SET_SYNC_INTERVAL

The receipt of a management message with a managementMessageKey field value PTP_MM_ SET_SYNC_ INTERVAL shall cause the sync_interval in the default data set of the receiving clock to be updated to the value of the syncInterval field of this message.

The update of the value of sync_interval shall not take effect until one of the following events occur:

— Receipt of a management message with a managementMessageKey field value PTP_MM_INITIALIZE_CLOCK
— A reboot or power cycle of the node containing the clock

7.12.24 PTP_MM_DISABLE_PORT

The receipt of a management message with a managementMessageKey field value PTP_MM_DISABLE_PORT shall cause the port whose port_id_field of the port configuration data set matches the targetPortId of the received Management message to process a DESIGNATED_DISABLED event transition in the state machine associated with the port (see 7.3). A port_id_field value of 0 shall cause all ports of a clock to respond to the message.

7.12.25 PTP_MM_ENABLE_PORT

The receipt of a management message with a managementMessageKey field value PTP_MM_ENABLE_PORT shall cause the port whose port_id_field of the port configuration data set matches the targetPortId of the received Management message to process a DESIGNATED_ENABLED event transition in the state machine associated with the port (see 7.3). A port_id_field value of 0 shall cause all ports of a clock to respond to the message.

7.12.26 PTP_MM_DISABLE_BURST

The receipt of a management message with a managementMessageKey field value PTP_MM_DISABLE_BURST shall cause the value of burst_enabled for the port whose port_id_field of the port configuration data set matches the targetPortId of the received Management message to be set to FALSE. If the port_id_field value is 0, the value of burst_enabled shall be set to FALSE for all ports of the receiving clock.

7.12.27 PTP_MM_ENABLE_BURST

The receipt of a management message with a managementMessageKey field value PTP_MM_ENABLE_BURST shall cause the value of burst_enabled for the port whose port_id_field of the port configuration data set matches the targetPortId of the received Management message to be set to TRUE. If the port_id_field value is 0, the value of burst_enabled shall be set to TRUE for all ports of the receiving clock.

7.12.28 PTP_MM_SET_TIME

The receipt of a management message with a managementMessageKey field value PTP_MM_SET_TIME shall cause the time of the receiver's local clock to be set to the value defined by the localTimeSeconds and localTimeNanoseconds fields of the received management message.

NOTE—Normally this message is designated to be acted on by the grandmaster clock of a subdomain. While it may be sent to and received by any clock in the subdomain, any changes in the local time of a clock that is not the grandmaster will be overwritten as the information from the grandmaster clock is propagated down the clock hierarchy on subsequent synchronization cycles.

The delay between the sending of this management message and the receipt and processing by the receiving clock will depend on the network technology and the characteristics of the operating systems of the two clocks. In general, the fluctuations in this delay time will limit the accuracy in the setting of system time by use of this message.

7.12.29 Unrecognized Management Messages

Receipt of a management message with a managementMessageKey field value which is reserved (see Table 25) in the version of this standard to which the clock conforms shall not cause any action which influences the operation of the protocol.

Receipt of a management message with a managementMessageKey field value which is reserved for implementation-specific purposes (see Table 25) but which the clock does not recognize shall not cause any action which influences the operation of the protocol.

8. PTP inter-clock message formats

8.1 Inter-clock messages

There are five kinds of messages passed between clocks participating in the PTP protocol.

— *Sync messages:* The appearance of a Sync message at the clock timestamp point interface of a clock shall be an event to which a local clock must assign a timestamp, the sync-event-timestamp, based on the value of the local clock.

— *Delay_Req messages:* The appearance of a Delay_Req message at the clock timestamp point interface of a clock shall be an event to which a local clock must assign a timestamp, the delay-event-timestamp, based on the value of the local clock.

— *Follow_Up messages:* Follow_Up messages communicate the local value of the sync-event-timestamp for a particular Sync message to another clock in the PTP system.

— *Delay_Resp messages:* Delay_Resp messages communicate the delay-event-timestamp marking the receipt by a master clock of a slave's Delay_Req message from the slave clock.

— *Management messages:* Management messages communicate information used to manage individual clocks and a system of clocks.

All timestamps generated shall be of type TimeRepresentation. Unless otherwise specified, all fields defined by this clause or subclauses shall have the datatype of the member or source from which the field value is obtained.

8.2 PTP message header specification

The field specifications for the first 11 fields of all PTP messages are identical. These fields shall be marshaled into their on-the-wire format in the following order:

a) versionPTP
b) versionNetwork
c) subdomain
d) messageType
e) sourceCommunicationTechnology
f) sourceUuid
g) sourcePortId
h) sequenceId
i) control
j) flags
k) reserved

These fields are specified in the following subclauses.

8.2.1 versionPTP

The versionPTP field shall be of type UInteger16. The value of the versionPTP field shall be the version number of the PTP standard implemented by the clock issuing the message. For this edition of the standard, the version number shall be 1.

8.2.2 versionNetwork

The versionNetwork field shall be of type UInteger16. The value of the versionNetwork field shall be the version number of the network specific portions of the PTP standard implemented by the clock issuing the message.

8.2.3 subdomain

The value of the subdomain field shall be the value of subdomain_name from the default data set.

The subdomain field may be used in an implementation for two purposes:

— As a "magic number" to increase the confidence that the message is a PTP message.
— As a unique identifier of the subdomain.

The Octet[PTP_SUBDOMAIN_NAME_LENGTH+1] array formed by appending the messageType field to the subdomain field may be used in an implementation for two purposes:

— As a "magic number" to increase the confidence that the message is a PTP event or a PTP general message.
— As a unique identifier distinguishing event or general messages of a particular PTP subdomain.

8.2.4 messageType

The messageType field shall be of type UInteger8. The value of the messageType field shall be as follows:

— If the message is of type Sync or Delay_Req, the value shall be 0x01. These messages shall be known as event messages.
— If the message is of type Delay_Resp, Follow_Up, or Management, the value shall be 0x02. These messages shall be known as general messages.

8.2.5 sourceCommunicationTechnology

The value of the sourceCommunicationTechnology field shall be the value of port_communication_technology of the port data set of the port issuing this message.

8.2.6 sourceUuid

The value of the sourceUuid field shall be the value of port_uuid_field of the port data set of the port issuing this message.

8.2.7 sourcePortId

The value of the sourcePortId field shall be the value of port_id_field of the port data set of the port issuing this message.

8.2.8 sequenceId

The value of the sequenceId field shall be one of the following:

— If the messageType indicates the message is of type Sync or Delay_Req, the value shall be last_sync_event_sequence_number member of the port data set of the port issuing this message.
— If the messageType indicates the message is of type Delay_Resp, Follow_Up or Management, the value shall be last_general_event_sequence_number member of the port data set of the port issuing this message.

8.2.9 control

The control field shall be of type UInteger8. This value of this field shall be taken from the ControlField enumeration (see Table 28). The values shall be selected as follows:

— For a Sync message, the value shall be PTP_SYNC_MESSAGE.
— For a Delay_Req message, the value shall be PTP_DELAY_REQ_MESSAGE.
— For a Follow_Up message, the value shall be PTP_FOLLOWUP_MESSAGE.
— For a Delay_Resp message, the value shall be PTP_DELAY_RESP_MESSAGE.
— For a Management message, the value shall be PTP_MANAGEMENT_MESSAGE.

IDL: enumeration ControlField;

Table 28—ControlField enumeration

Enumeration	Value
PTP_SYNC_MESSAGE	0
PTP_DELAY_REQ_MESSAGE	1
PTP_FOLLOWUP_MESSAGE	2
PTP_DELAY_RESP_MESSAGE	3
PTP_MANAGEMENT_MESSAGE	4
reserved	5–255

8.2.10 flags

The flag field shall be of type Octet[2]. The value of this field shall be interpreted bit wise with each bit interpreted as a Boolean as follows:

Bit position	Name	Interpretation
most significant (15)		reserved
(14)		reserved
(13)		reserved
(12)		reserved
(11)		reserved
(10)		reserved
(9)		reserved
(8)		reserved
(7)		reserved
(6)	PTP_SYNC_BURST	In a Delay_Req message, TRUE iff the sender is requesting a burst of Sync messages. In a Sync or Follow_Up message, TRUE iff this is part of such a burst (and can thus be ignored by clocks not requesting it).
(5)	PARENT_STATS	Value of parent_stats of Parent data set
(4)	PTP_EXT_SYNC	Value of external_timing of default data set
(3)	PTP_ASSIST	Value of clock_followup_capable of default data set
(2)	PTP_BOUNDARY_CLOCK	Value of is_boundary_clock of default data set
(1)	PTP_LI_59	Value of leap_59 of global time properties data set
least significant (0)	PTP_LI_61	Value of leap_61 of global time properties data set

8.2.11 reserved

This field shall consist of four octets. The four octets of this field shall be set to the null value.

8.3 Sync and Delay_Req messages

In the protocol specification, event messages appear in two forms distinguished by the value of their control field.

Both forms shall share the same field specifications. Both forms shall communicate via the EventPort.

8.3.1 Sync and Delay_Req message field specifications

The field specifications for both Sync and Delay_Req messages are identical except as noted in the following subclauses. The fields of both Sync and Delay_Req messages shall be marshaled into their on-the-wire format in the following order:

a) versionPTP
b) versionNetwork
c) subdomain
d) messageType
e) sourceCommunicationTechnology
f) sourceUuid
g) sourcePortId
h) sequenceId
i) control
j) flags
k) reserved
l) originTimestamp
m) epochNumber
n) currentUTCOffset
o) grandmasterCommunicationTechnology
p) grandmasterClockUuid
q) grandmasterPortId
r) grandmasterSequenceId
s) grandmasterClockStratum
t) grandmasterClockIdentifier
u) grandmasterClockVariance
v) grandmasterIsPreferred
w) grandmasterIsBoundaryClock
x) syncInterval
y) localClockVariance
z) localStepsRemoved
aa) localClockStratum
ab) localClockIdentifier
ac) parentCommunicationTechnology
ad) parentUuid
ae) parentPortField
af) estimatedMasterVariance
ag) estimatedMasterDrift
ah) utcReasonable

8.3.1.1 control

The value of the control field indicates that this message is a Sync or a Delay_Req message. The value of this field shall be the value of the ControlField enumeration member:

— PTP_SYNC_MESSAGE for a Sync message.
— PTP_DELAY_REQ_MESSAGE for a Delay_Req message

8.3.1.2 originTimestamp

The originTimestamp field shall be of type TimeRepresentation. The value of the originTimestamp field shall be the time, as estimated by the local clock of the clock issuing this Sync or Delay_Req message, at which the message timestamp point passed the outbound clock timestamp point (see 6.2.4.9), corrected for any latency per 7.8.1.3. These timestamps are the sync-event-timestamp and delay-event-timestamps respectively (see 8.1). The value of this field shall have an absolute error, relative to the local clock, of less than 0.25 seconds. The precise interpretation of this field shall depend on the value of the PTP_ASSIST bit of the flags field as follows:

— *PTP_ASSIST is FALSE*. In this case, there will not be a Follow_Up message issued by the clock. The originTimestamp field shall contain the latency corrected sync-event-timestamp or delay-event-timestamp. A negative or zero value of the timestamp is an error.
— *PTP_ASSIST is TRUE*. In this case, there will be a Follow_Up message issued by the clock for a Sync message (but not for a Delay_Req message). The originTimestamp field shall contain a timestamp estimating the latency corrected sync-event-timestamp or delay-event-timestamp. A negative value of the timestamp is an error. A zero value of the timestamp shall be used if no estimate is available at the time the Sync message is issued. The precise value of the time is conveyed in the Follow_Up message associated with the Sync message.

8.3.1.3 epochNumber

The value of the epochNumber field shall be the value of epoch_number of the global time properties data set of the clock issuing this message.

8.3.1.4 currentUTCOffset

The value of the currentUTCOffset field shall be the value of current_utc_offset of the global time properties data set of the clock issuing this message.

8.3.1.5 grandmasterCommunicationTechnology

The value of the grandmasterCommunicationTechnology field shall be the value of grandmaster_communication_technology of the parent data set of the clock issuing this message.

8.3.1.6 grandmasterClockUuid

The value of the grandmasterClockUuid field shall be the value of grandmaster_uuid_field of the parent data set of the clock issuing this message.

8.3.1.7 grandmasterPortId

The value of the grandmasterPortId field shall be the value of grandmaster_port_id of the parent data set of the clock issuing this message.

8.3.1.8 grandmasterSequenceId

The value of the grandmasterSequenceId field shall be the value of grandmaster_sequence_number of the parent data set of the clock issuing this message.

8.3.1.9 grandmasterClockStratum

The value of the grandmasterClockStratum field shall be the value of grandmaster_stratum of the parent data set of the clock issuing this message.

8.3.1.10 grandmasterClockIdentifier

The value of the grandmasterClockIdentifier field shall be the value of grandmaster_identifier of the parent data set of the clock issuing this message.

8.3.1.11 grandmasterClockVariance

The value of the grandmasterClockVariance field shall be the value of grandmaster_variance of the parent data set of the clock issuing this message.

8.3.1.12 grandmasterPreferred

The value of the grandmasterClockPreferred field shall be the value of grandmaster_preferred of the parent data set of the clock issuing this message.

8.3.1.13 grandmasterIsBoundaryClock

The value of the grandmasterIsBoundaryClock field shall be the value of grandmaster_is_boundary_clock of the parent data set of the clock issuing this message.

8.3.1.14 syncInterval

The value of the syncInterval field shall be the value of sync_interval of the default data set of the clock issuing this message.

8.3.1.15 localClockVariance

The value of the localClockVariance field shall be the value of clock_variance of the default data set of the clock issuing this message.

8.3.1.16 localStepsRemoved

The value of the localStepsRemoved field shall be the value of steps_removed of the current data set of the clock issuing this message.

8.3.1.17 localClockStratum

The value of the localClockStratum field shall be the value of clock_stratum of the default data set of the clock issuing this packet.

8.3.1.18 localClockIdentifier

The value of the localClockIdentifier field shall be the value of clock_identifier of the default data set of the clock issuing this packet.

8.3.1.19 parentCommunicationTechnology

The value of the parentCommunicationTechnology field shall be the value of parent_communication_technology of the parent data set of the clock issuing this packet.

8.3.1.20 parentUuid

The value of the parentUuid field shall be the value of parent_uuid of the parent data set of the clock issuing this packet.

8.3.1.21 parentPortField

The value of the parentPortField field shall be the value of parent_port_id of the parent data set of the clock issuing this packet.

8.3.1.22 estimatedMasterVariance

The value of the estimatedMasterVariance field shall be the value of observed_variance of the parent data set of the clock issuing this packet.

8.3.1.23 estimatedMasterDrift

The value of the estimatedMasterDrift field shall be the value of observed_drift of the parent data set of the clock issuing this packet.

8.3.1.24 utcReasonable

The value of the utcReasonable field shall be the value of utc_reasonable of the parent data set of the clock issuing this packet.

8.4 Follow_Up messages

In the protocol specification, Follow_Up messages are distinguished by the value of their control field. Unless otherwise stated, the specification of the header fields shall be unchanged from 8.2.

8.4.1 Follow_Up message field specifications

The field specifications for Follow_Up messages are given in the following subclauses. The fields of Follow_Up messages shall be marshaled into its on-the-wire format in the following order:

a) versionPTP
b) versionNetwork
c) subdomain
d) messageType
e) sourceCommunicationTechnology
f) sourceUuid
g) sourcePortId
h) sequenceId
i) control
j) flags
k) reserved
l) associatedSequenceId
m) preciseOriginTimestamp

8.4.1.1 control

The value of the control field indicates that this message is a Follow_Up message. The value of this field shall be the value of the ControlField enumeration member PTP_FOLLOWUP_MESSAGE.

8.4.1.2 associatedSequenceId

The value of the associatedSequenceId field shall be the value of the sequenceId field of the Sync message associated with this Follow_Up message (see 7.5.9). The associatedSequenceId field shall have datatype UInteger16.

8.4.1.3 preciseOriginTimestamp

The preciseOriginTimestamp field shall have datatype TimeRepresentation. The value of the preciseOriginTimestamp field shall be the time, the sync-event-timestamp, at which the associated Sync message timestamp point passed the outbound clock timestamp point, as measured by the local clock of the port that issued the associated message, and corrected for latency per 7.8.1.3. The value of this field shall have an absolute error, relative to the local clock, of less than 0.25 seconds. The precise interpretation of this field depends on the value of the PTP_ASSIST bit of the flags field of the associated Sync message as follows:

— *PTP_ASSIST is FALSE*. In this case, there should not have been a Follow_Up message issued by the clock.
— *PTP_ASSIST is TRUE*. In this case, this message shall be the Follow_Up message associated with the last Sync message issued by the clock. The preciseOriginTimestamp field shall contain a precise value of the latency corrected sync-event-timestamp of the associated Sync message. This value shall be consistent with the statistics specified by the localClockVariance field of the associated Sync message. A negative or zero value of the timestamp is an error.

8.5 Delay_Resp messages

In the protocol specification, Delay_Resp messages are distinguished by the value of their control field. Unless otherwise stated, the specification of the header fields shall be unchanged from 8.2.

8.5.1 Delay_Resp message field specifications

The field specifications for Delay_Resp messages are given in the following subclauses. The fields of Delay_Resp messages shall be marshaled into its on-the-wire format in the following order:

a) versionPTP
b) versionNetwork
c) subdomain
d) messageType
e) sourceCommunicationTechnology
f) sourceUuid
g) sourcePortId
h) sequenceId
i) control
j) flags
k) reserved
l) delayReceiptTimestamp
m) requestingSourceCommunicationTechnology
n) requestingSourceUuid
o) requestingSourcePortId
p) requestingSourceSequenceId

The associated Delay_Req message shall be the message providing the source for the requestingSourceCommunicationTechnology, requestingSourceUuid, requestingSourcePortId, and requestingSourceSequenceId field values of this Delay_Resp message. The receipt of the associated Delay_Req message shall generate the delayReceiptTimestamp value (see 7.5.5). The combination of the requestingSourceCommunicationTechnology, requestingSourceUuid, requestingSourcePortId, and requestingSourceSequenceId fields shall uniquely distinguish the particular associated Delay_Req message using a particular PTP communication path modulo the rollover limitations of the datatype of the requestingSourceSequenceId.

8.5.1.1 control

The value of the control field indicates that this message is a Delay_Resp message. The value of this field shall be the value of the ControlField enumeration member PTP_DELAY_RESP_MESSAGE.

8.5.1.2 delayReceiptTimestamp

The value of the delayReceiptTimestamp field shall be the time, as measured by the local clock of the port sending this message, at which the message timestamp point of the associated Delay_Req message passed the inbound clock timestamp point, corrected for any latency (see 7.8.1.3).

8.5.1.3 requestingSourceCommunicationTechnology

The value of the requestingSourceCommunicationTechnology field shall be the value of the sourceCommunicationTechnology field of the associated Delay_Req message.

8.5.1.4 requestingSourceUuid

The value of the requestingSourceUuid field shall be the value of the sourceUuid field of the associated Delay_Req message.

8.5.1.5 requestingSourcePortId

The value of the requestingSourcePortId field shall be the value of the sourcePortId field of the associated Delay_Req message.

8.5.1.6 requestingSourceSequenceId

The value of the requestingSourceSequenceId field shall be the value of the sequenceId field of the associated Delay_Req message.

8.6 Management messages

In the protocol specification, Management messages are distinguished by the value of their control field. Unless otherwise stated, the specification of the header fields shall be unchanged from 8.2.

8.6.1 Management message field specifications

The field specifications for Management messages are given in the following subclauses. The fields of Management messages shall be marshaled into its on-the-wire format in the following order:

a) versionPTP
b) versionNetwork
c) subdomain
d) messageType
e) sourceCommunicationTechnology
f) sourceUuid
g) sourcePortId

h) sequenceId
i) control
j) flags
k) reserved
l) targetCommunicationTechnology
m) targetUuid
n) targetPortId
o) startingBoundaryHops
p) boundaryHops
q) managementMessageKey
r) parameterLength
s) messageParameters

8.6.1.1 control

The value of the control field indicates that this message is a Management message. The value of this field shall be the value of the ControlField enumeration member PTP_MANAGEMENT_MESSAGE.

8.6.1.2 targetCommunicationTechnology

The value of the targetCommunicationTechnology field shall be the value of the clock_communication_technology member of the default data set of the clock receiving this Management message. The targetCommunicationTechnology field shall have the datatype specified for the communication_technology_field in 6.2.4.1.

8.6.1.3 targetUuid

The targetUuid field shall be an OctetArray of length PTP_UUID_LENGTH. The value of the targetUuid field shall be the clock_uuid_field member of the default data set of the clock in the subdomain that is to receive and act on this Management message. The targetUuid field shall have the datatype specified for the uuid_field in 6.2.4.1. The default value for the targetUuid and targetCommunicationTechnology fields indicates that all clocks in the subdomain are to receive and act on this management message (see 7.5.7).

8.6.1.4 targetPortId

The value of the targetPortId field shall be the value of the port_id_field member of the port configuration data set of the target port of the clock receiving this management message. The action specified by the Management message shall be confined to the target port. The targetPortId field shall have the datatype specified for the port_id_field in 6.2.4.1.

8.6.1.5 startingBoundaryHops

The value of the startingBoundaryHops field shall conform to the semantics of the boundaryHops field given in 6.2.2.1.

The value of the startingBoundaryHops field is implementation-dependent for messages that are not issued in response to a request from another Management message. For Management messages that are issued in response to a request from another Management message, the value of startingBoundaryHops shall be the value computed from the startingBoundaryHops and boundaryHops fields of the requesting message as startingBoundaryHops – boundaryHops. Note that when a Management message is received, the absolute value of this difference indicates the number of retransmissions by boundary clocks that the message experienced. The startingBoundaryHops field shall have datatype Integer16.

8.6.1.6 boundaryHops

The value of the boundaryHops field shall indicate the remaining number of successive retransmissions of the management message by boundary clocks receiving the message per 6.2.2.1. The value of boundaryHops shall be identical to the value of the field startingBoundaryHops when first transmitted by the issuing clock. The boundaryHops field shall have datatype Integer16.

8.6.1.7 managementMessageKey

The value of the managementMessageKey field shall indicate the purpose of the message and shall be the key to interpreting the messageParameters field. The value of this field shall be the value of a member selected from the enumeration ManagementMessage. This enumeration and the semantics of each message type are specified in 7.12.

8.6.1.8 parameterLength

The value of the parameterLength field shall specify the actual number of octets in the messageParameters field as defined in 8.6.1.10. The value of this field shall be less than or equal to PTP_MAX_ MANAGEMENT_PAYLOAD_SIZE. The parameterLength shall have datatype UInteger16.

8.6.1.9 messageParameters

The messageParameters field shall be an OctetArray of length parameterLength. The interpretation of this field shall be determined by the value of the managementMessageKey field (see 7.12). Parameters shall be marshaled as specified in 8.6.1.10.

8.6.1.10 Specification of parameterLength values and messageParameter formats

The formats of the messageParameters field for the Management messages specified in the following subclauses shall be used unless superseded or otherwise specified by communication technology specific restrictions in an Annex. Boundary clocks or any other device accessing multiple communication paths for which the formats of Management messages differ shall translate these messages into the format specified for the communication technology of the target communication path before transmission. For messages where parameterLength is 0, there shall be no messageParameters field.

8.6.1.10.1 PTP_MM_NULL

The value of the parameterLength field shall be 0.

8.6.1.10.2 PTP_MM_OBTAIN_IDENTITY

The value of the parameterLength field shall be 0.

8.6.1.10.3 PTP_MM_CLOCK_IDENTITY

The value of the parameterLength field is communication technology specific.

Default data set values of the issuing clock shall be encoded into the corresponding messageParameters fields in the order shown in Table 29.

Table 29—messageParameters fields for PTP_MM_CLOCK_IDENTITY

messageParameters field	messageParameters field type	Default data set value
clockCommunicationTechnology	UInteger8	clock_communication_technology
clockUuidField	Octet[6]	clock_uuid_field
clockPortField	UInteger16	clock_port_field
manufacturerIdentity	Octet[48]	manufacturer_identity

Where the manufacturer_identity field value shall be up to 48 Latin-1 characters (ISO/IEC 8859-1:1998) packed one character per octet, left justified in the field, terminated by the null octet, 0x00, and with unused octets filled with the null octet. The manufacturerIdentity field shall indicate, in order:

a) The name of the manufacturer of the clock, followed by a semicolon (;)
b) The model number of the clock

8.6.1.10.4 PTP_MM_INITIALIZE_CLOCK

The value of the parameterLength field is communication technology specific.

There shall be a single member of the messageParameters field, initializationKey. This member shall contain two octets to be interpreted as a UInteger16. This value shall be the designator of a consistent set of initialization values (see 7.12.4).

8.6.1.10.5 PTP_MM_SET_SUBDOMAIN

The value of the parameterLength field is communication technology specific.

The messageParameters field, subdomainName, shall be an octet array of length PTP_SUBDOMAIN_NAME_LENGTH, containing a subdomain name subject to the restrictions of 6.2.5.1.

8.6.1.10.6 PTP_MM_CLEAR_DESIGNATED_PREFERRED_MASTER

The value of the parameterLength field shall be 0.

8.6.1.10.7 PTP_MM_SET_DESIGNATED_PREFERRED_MASTER

The value of the parameterLength field shall be 0.

8.6.1.10.8 PTP_MM_GET_DEFAULT_DATA_SET

The value of the parameterLength field shall be 0.

8.6.1.10.9 PTP_MM_DEFAULT_DATA_SET

The value of the parameterLength field is communication technology specific.

Default data set values of the issuing clock shall be encoded into the corresponding messageParameters fields in the order shown in the Table 30.

Table 30—messageParameters fields for PTP_MM_DEFAULT_DATA_SET

messageParameters field	messageParameters field type	Default data set value
clockCommunicationTechnology	UInteger8	clock_communication_technology
clockUuidField	Octet[6]	clock_uuid_field
clockPortField	UInteger16	clock_port_field
clockStratum	UInteger8	clock_stratum
clockIdentifier	Octet[4]	clock_identifier
clockVariance	Integer16	clock_variance
clockFollowupCapable	Boolean	clock_followup_capable
preferred	Boolean	preferred
initializable	Boolean	initializable
externalTiming	Boolean	external_timing
isBoundaryClock	Boolean	is_boundary_clock
syncInterval	Integer8	sync_interval
subdomainName	Octet[16]	subdomain_name
numberPorts	UInteger16	number_ports
numberForeignRecords	UInteger16	number_foreign_records

8.6.1.10.10 PTP_MM_UPDATE_DEFAULT_DATA_SET

The value of the parameterLength field is communication technology specific.

Update values for the default data set of the receiving clock shall be encoded into the corresponding messageParameters fields in the order shown in Table 31.

Table 31—messageParameters fields for PTP_MM_UPDATE_DEFAULT_DATA_SET

messageParameters field	messageParameters field type	Update value for default data set member of receiving clock
clockStratum	UInteger8	clock_stratum
clockIdentifier	Octet[4]	clock_identifier
clockVariance	Integer16	clock_variance
preferred	Boolean	preferred
syncInterval	Integer8	sync_interval
subdomainName	Octet[16]	subdomain_name

8.6.1.10.11 PTP_MM_GET_CURRENT_DATA_SET

The value of the parameterLength field shall be 0.

8.6.1.10.12 PTP_MM_CURRENT_DATA_SET

The value of the parameterLength field is communication technology specific.

Current data set values of the issuing clock shall be encoded into the corresponding messageParameters fields in the order shown in Table 32.

Table 32—messageParameters fields for PTP_MM_CURRENT_DATA_SET

messageParameters field	messageParameters field type	Current data set value
stepsRemoved	UInteger16	steps_removed
offsetFromMasterSeconds	UInteger32	offset_from_master (seconds)
offsetFromMasterNanoseconds	Integer32	offset_from_master (nanoseconds)
oneWayDelaySeconds	UInteger32	one_way_delay (seconds)
oneWayDelayNanoseconds	Integer32	one_way_delay (nanoseconds)

8.6.1.10.13 PTP_MM_GET_PARENT_DATA_SET

The value of the parameterLength field shall be 0.

8.6.1.10.14 PTP_MM_PARENT_DATA_SET

The value of the parameterLength field is communication technology specific.

Parent data set values of the issuing clock shall be encoded into the corresponding messageParameters fields in the order shown in Table 33.

Table 33—messageParameters fields for PTP_MM_PARENT_DATA_SET

messageParameters field	messageParameters field type	Parent data set value
parentCommunicationTechnology	UInteger8	parent_communication_technology
parentUuid	Octet[6]	parent_uuid
parentPortId	UInteger16	parent_port_id
parentLastSyncSequenceNumber	UInteger16	parent_last_sync_sequence_number
parentFollowupCapable	Boolean	parent_followup_capable
parentExternalTiming	Boolean	parent_external_timing
parentVariance	Integer16	parent_variance
parentStats	Boolean	parent_stats

Table 33—messageParameters fields for PTP_MM_PARENT_DATA_SET *(continued)*

messageParameters field	messageParameters field type	Parent data set value
observedVariance	Integer16	observed_variance
observedDrift	Integer32	observed_drift
utcReasonable	Boolean	utc_reasonable
grandmasterCommunicationTechnology	UInteger8	grandmaster_communication_technology
grandmasterUuidField	Octet[6]	grandmaster_uuid_field
grandmasterPortIdField	UInteger16	grandmaster_port_id_field
grandmasterStratum	UInteger8	grandmaster_stratum
grandmasterIdentifier	Octet[4].	grandmaster_identifier
grandmasterVariance	Integer16	grandmaster_variance
grandmasterPreferred	Boolean	grandmaster_preferred
grandmasterIsBoundaryClock	Boolean	grandmaster_is_boundary_clock
grandmasterSequenceNumber	UInteger16	grandmaster_sequence_number

8.6.1.10.15 PTP_MM_GET_PORT_DATA_SET

The value of the parameterLength field is 0.

8.6.1.10.16 PTP_MM_PORT_DATA_SET

The value of the parameterLength field is communication technology specific.

Port configuration data set values for the requested port of the issuing clock shall be encoded into the corresponding messageParameters fields in the order shown in Table 34.

Table 34—messageParameters fields for PTP_MM_PORT_DATA_SET

messageParameters field	messageParameters field type	Port data set and local values
returnedPortNumber	UInteger16	returned_port_number
portState	UInteger8	port_state
lastSyncEventSequenceNumber	UInteger16	last_sync_event_sequence_number
lastGeneralEventSequenceNumber	UInteger16	last_general_event_sequence_number
portCommunicationTechnology	UInteger8	port_communication_technology
portUuidField	Octet[6]	port_uuid_field
portIdField	UInteger16	port_id_field
burstEnabled	Boolean	burst_enabled
subdomainAddressOctets	UInteger8	subdomain_address_octets = h
eventPortAddressOctets	UInteger8	event_port_address_octets = k

Table 34—messageParameters fields for PTP_MM_PORT_DATA_SET *(continued)*

messageParameters field	messageParameters field type	Port data set and local values
generalPortAddressOctets	UInteger8	general_port_address_octets = m
subdomainAddress	Octet[h]	subdomain_address
eventPortAddress	Octet[k]	event_port_address
generalPortAddress	Octet[m]	general_port_address

Where:

— returned_port_number shall be the port number of the data set as requested in the corresponding PTP_MM_GET_PORT_DATA_SET request message.

— subdomain_address_octets = *h* shall be the number of octets in the subdomainAddress field.

— event_port_address_octets = *k* shall be the number of octets in the eventPortAddress field.

— general_port_address_octets = *m* shall be the number of octets in the generalPortAddress field.

8.6.1.10.17 PTP_MM_GET_GLOBAL_TIME_DATA_SET

The value of the parameterLength field shall be 0.

8.6.1.10.18 PTP_MM_GLOBAL_TIME_DATA_SET

The value of the parameterLength field is communication technology specific.

Global time data set values of the issuing clock shall be encoded into the corresponding messageParameters fields in the order shown in Table 35.

Table 35—messageParameters fields for PTP_MM_GLOBAL_TIME_DATA_SET

messageParameters field	messageParameters field type	Global time data set and local values
localTimeSeconds	UInteger32	local_time_seconds
localTimeNanoseconds	Integer32	local_time_nanoseconds
currentUtcOffset	Integer16	current_utc_offset
leap59	Boolean	leap_59
leap61	Boolean	leap_61
epochNumber	UInteger16	epoch_number

The fields local_time_seconds and local_time_nanoseconds shall be the current time value of the issuing clock.

8.6.1.10.19 PTP_MM_UPDATE_GLOBAL_TIME_PROPERTIES

The value of the parameterLength field is communication technology specific.

Update values for the global time properties data set shall be encoded into the corresponding messageParameters fields in the order shown in Table 36.

Table 36—messageParameters fields for PTP_MM_UPDATE_GLOBAL_TIME_PROPERTIES

messageParameters field	messageParameters field type	Update value for global time properties data set member of receiving clock
currentUtcOffset	Integer16	current_utc_offset
leap59	Boolean	leap_59
leap61	Boolean	leap_61
epochNumber	UInteger16	epoch_number

8.6.1.10.20 PTP_MM_GOTO_FAULTY_STATE

The value of the parameterLength field shall be 0.

8.6.1.10.21 PTP_MM_GET_FOREIGN_DATA_SET

The value of the parameterLength field is communication technology specific.

The contents of the messageParameters field, recordKey shall be the foreign data set record number for the returned record encoded as a UInteger16. The value shall be less than or equal to the total number of foreign records, number_foreign_records of the default data set and greater than 0.

8.6.1.10.22 PTP_MM_FOREIGN_DATA_SET

The value of the parameterLength field is communication technology specific.

The designated record of the foreign master data set and designated local variables of the selected port shall be encoded into the corresponding messageParameters fields in the order shown in Table 37.

Table 37—messageParameters fields for PTP_MM_FOREIGN_DATA_SET

messageParameters field	messageParameters field type	Foreign data set and local values
returnedPortNumber	UInteger16	returned_port_number
returnedRecordNumber	UInteger16	returned_record_number
foreignMasterCommunicationTechnology	UInteger8	foreign_master_communication_technology
foreignMasterUuidField	Octet[6]	foreign_master_uuid_field
foreignMasterPortIdField	UInteger16	foreign_master_port_id_field
foreignMasterSyncs	UInteger16	foreign_master_syncs

The returned_port_number shall be the port number of the foreign data set as requested in the PTP_MM_GET_FOREIGN_DATA_SET request.

The returned_record_number shall be the index of record returned from the foreign data set as requested in recordKey field of the PTP_MM_GET_FOREIGN_DATA_SET request.

8.6.1.10.23 PTP_MM_SET_SYNC_INTERVAL

The value of the parameterLength field is communication technology specific.

The update value sync_interval for the receiving clock shall be encoded into the messageParameters field syncInterval as an Integer16.

8.6.1.10.24 PTP_MM_DISABLE_PORT

The value of the parameterLength field shall be 0.

8.6.1.10.25 PTP_MM_ENABLE_PORT

The value of the parameterLength field shall be 0.

8.6.1.10.26 PTP_MM_DISABLE_BURST

The value of the parameterLength field shall be 0.

8.6.1.10.27 PTP_MM_ENABLE_BURST

The value of the parameterLength field shall be 0.

8.6.1.10.28 PTP_MM_SET_TIME

The value of the parameterLength field is communication technology specific.

The update value of the local time of the receiving clock shall be encoded into the corresponding messageParameters fields in the order shown in Table 38.

Table 38—messageParameters fields for PTP_MM_SET_TIME

messageParameters field	messageParameters field type	Update value for local time of receiving clock
localTimeSeconds	UInteger32	local_ time_seconds
localTimeNanoseconds	Integer32	local_time_nanoseconds

The fields local_time_seconds and local_time_nanoseconds shall be the update time value for the receiving clock.

9. Conformance

9.1 Conformance objective

The philosophy underlying the conformance requirements of this clause is to provide the structure necessary to raise the level of interoperability of systems built to this standard, while leaving opportunity open for continued technical improvement and differentiation.

9.2 PTP node conformance requirements

A conformant PTP node shall meet the requirements of one or more of the subclauses of this clause.

9.2.1 Fully conformant PTP node

A fully conformant PTP node shall meet all normative requirements of this standard.

9.2.2 Slave only conformant PTP node

A slave-only conformant PTP node shall meet all normative requirements of this standard with the following exceptions:

— It shall be configured internally such that the best master clock algorithm always selects the clock identified by E_{best} (see 7.6) as the best master clock.
— It shall not issue Sync, Delay_Resp, or Follow_Up messages under any circumstance.
— It shall report a clock_stratum value of 255 in response to any Management message requesting this property.

9.2.3 Management only conformant PTP node

A management-only conformant PTP node shall meet all normative requirements of this standard with the following exceptions:

— It shall not issue Sync, Delay_Req, Delay_Resp, or Follow_Up messages under any circumstance.
— It shall report a clock_stratum value of 255 in response to any Management message requesting this property.

9.3 PTP system conformance requirements

A PTP system shall be deemed conformant if and only if all of the following are true:

— All PTP nodes in the system meet the requirements of 9.2.
— There is at least one PTP clock in each PTP subdomain of the system meeting the requirements of 9.2.1.
— All normative requirements of this standard regarding PTP communications, PTP communication topology, PTP timing constraints, PTP message syntax, and PTP message semantics are met.
— The values of all PTP defined values and constants of 7.9 are identical in all PTP nodes.
— For each communication technology supporting a PTP communication path, a PTP communication technology specific specification exists, and the requirements of that specification are met.
— No node in the system issues PTP messages unless that node is conformant to the terms of 9.2.

The communication technology specific specification shall define all terms and resolve all issues noted in this standard as being communication technology specific.

Annex A

(informative)

Using the PTP protocol

A.1 Overview

PTP provides a simple methodology for accurately synchronizing clocks in a distributed system. When designing such a system the following questions need to be answered.

Physical layout issues:

— How physically dispersed are the clocks?
— What network technology is to be used?

Logical issues:

— Is the system a single collection of clocks, or are the clocks divided into logical groupings each with their own sense of time?

Component issues:

— How accurately do the clocks need to be synchronized?
— What is the primary source of time for the system? Must it be traceable to UTC?

Local implementation issues:

— How are timing requirements to be met?
— How do other applications sharing the communication network affect PTP?
— How do accuracy requirements affect the implementation?
— What are the design issues for local oscillators?

The following clauses of this annex address each of these topics.

A.2 Physical layout

PTP clocks communicate with each other over a network. Typically, the selection of the network technology will be based on the primary application. PTP will work on any packet-based system. PTP is designed to work in a multicast environment although it is possible to design unicast PTP components and systems. Ethernet is an ideal network for implementing PTP, and the rest of this annex uses Ethernet as an example.

All networks have limitations on distance, number of allowed nodes, and traffic. If the clocks to be synchronized are dispersed beyond the range of the network technology then the system must be designed as separate *islands of time* with provision outside of PTP for synchronizing these islands.

For example, if the system consists of two compact sites separated by several miles, PTP can be used within each site with site-to-site synchronization provided by another technology such as GPS.

Within a site, distance, traffic, and number of node issues are usually addressed by special network components. For Ethernet, localized nodes typically communicate via simple repeaters. For more complex applications, groups of nodes each using a repeater or even individual nodes may be connected using switched hubs. For larger and more complex systems, routers are used to separate the system into groupings each group using switches or repeaters. In general, each level of separation using these devices introduces additional statistical delay and delay fluctuation in the message transmission times between nodes.

PTP is designed to minimize the effects of delay and delay fluctuation. To get the best PTP performance, the network topology should have as a constraint the minimization of the number of such separating devices between clocks with the most critical synchronization requirements.

For Ethernet technology, the following guidelines are useful:

— The least delay and delay fluctuation will be observed between clocks communicating via repeaters. Ethernet repeaters do not use any store and forward techniques nor do they involve any parsing of message contents. Repeater delay and delay fluctuation can be corrected quite well by local synchronization algorithms.
— Switched hubs introduce considerable delay and increased delay fluctuation compared to repeaters. However, depending on the desired accuracy, a PTP system containing switches will produce satisfactory results. The increased delay and delay fluctuation results from the requirement that the switch parse a portion of the message header and possibly queue messages since each network connection to a switch is a separate Ethernet collision domain.
— Routers are usually the top-level component in an Ethernet network. Routers do considerable parsing on a message and involve extensive store and forward resulting in considerable delay and delay fluctuation. Without special techniques, clocks synchronized via routers will be limited in synchronization accuracy, generally to the order of several milliseconds.

Boundary clocks may be used to improve the performance across separations in the network defined by routers.

A.3 Logical layout

Most applications consist of a single set of clocks to be synchronized. For this case, all the clocks can use a single subdomain. If the DefaultPTPdomain is used, then no configuration of the clocks is necessary.

If the application requires several groups of clocks with each group maintaining a different self-consistent time base, then one of two solutions may be used.

— If the rest of the application is segmented into the same groups, it may be possible to use separate noncommunicating networks in which case each can group can use the DefaultPTPdomain. Network routers are often used for this purpose.
— If the groups must share a common network, then each group may be assigned a different subdomain_name. This will logically divide the PTP clocks as desired. Depending on the mapping to the underlying physical addressing of the network, the processing load on each clock may or may not be affected. In the case of Ethernet, PTP has reserved four multicast addresses for PTP domains. Depending on the hardware of each node, these addresses may be selected by the network hardware thereby reducing unwanted loading of the node's processor.

With the exception of the assignment of PTP nodes to a subdomain, PTP defines an administration free system. Within a subdomain, PTP nodes may be added or removed without any requirement for modification of address tables, etc., provided components use the recommended multicast communication model. Addition or removal of PTP nodes may cause a different PTP clock to become the grandmaster clock in the system.

This may cause a transient in the time base as the system automatically recalibrates for the new delay patterns to the new grandmaster clock.

A.4 Component issues

The primary issue in the selection of PTP system components is the required synchronization accuracy.

— PTP clocks should be selected that are designed to support those features of the PTP protocol required for a given accuracy. PTP clocks with the highest inherent accuracy should support the use of the Follow_Up messages.
— Network components and physical design decisions also affect the accuracy as outlined in the previous sections.

Properly designed Ethernet PTP systems can readily achieve submicrosecond accuracy.

A second issue is the technique for establishing the PTP system epoch. In every PTP subdomain, the epoch is defined by the grandmaster clock that is selected according to the best master clock algorithm.

If UTC time is a requirement, then the grandmaster clock must maintain a UTC time base.

If a PTP subdomain contains a stratum 1 clock the time base will be UTC. Such systems may or may not maintain the epoch after a power outage.

If a PTP subdomain contains a stratum 2 or greater clock, the time base will be either UTC or a time base established by the user. Such systems may or may not maintain the epoch after a power outage.

In addition to introducing stratum 1 or 2 PTP clocks into a system, control over the selection of the grandmaster clock may be achieved by designating one or more PTP clocks as belonging to a preferred master clock set. If the clocks so designated are selected to be clocks that maintain their time base over a power outage a more robust system will result.

A master clock may fail in such a way that its time or frequency become incorrect. Detection of this problem and recovery from it are outside the scope of this standard. However, the Delay_Req message's flags field, fields (see 8.3.1.3–8.3.1.4 and 8.3.1.17–8.3.1.24) are available to aid in detecting a *false-ticking* master. Implementers are advised to consider information from as many clocks as possible and to weigh the information from each clock according to that clock's inherent stability.

Note that detection and recovery of a false-ticking master can be done by a third party, by one or more of the slaves (slaves can see each others' Delay_Req messages), or by the master itself.

The PTP_MM_DISABLE_PORT and PTP_MM_GOTO_FAULTY_STATE management messages are available to aid in recovering from a false-ticking master. Note that disabling or demoting a master has side effects (especially if it is a boundary clock), so the decision to do that may depend on factors besides its timekeeping quality. That decision is outside the scope of this standard.

A.5 Local implementation issues

This clause provides some guidelines for implementers of PTP ordinary and boundary clocks. While not in the scope of this standard, implementations should take care that services built on top of clocks synchronized via PTP (or any other protocol) do not degrade the accuracy.

A.5.1 Timing issues

The timing requirements of an implementation are defined in 7.11. Implementations must meet these requirements and must also meet whatever timing requirements are needed to operate the servomechanism that synchronizes the local clock based on information in PTP messages.

Implementations must ensure that adequate computing and memory resources are available to meet these requirements. Implementations must also ensure that the resources needed by the PTP implementation have adequate priority over other applications sharing these resources to meet the PTP and servomechanism timing requirements. It is recommended that PTP tasks be assigned the highest priority in an implementation, similar to priorities assigned to the protocol stack and other operating system resources.

PTP implementations normally require resources for a short time in every sync interval. The selection of the sync interval for a system must be consistent with the available resources in all system components.

The use of network resources by other applications can affect PTP accuracy as discussed in A.5.2.

A.5.2 Accuracy issues

The achievable accuracy of a PTP system is limited by the following:

— The delay fluctuation in the protocol stacks of PTP clocks
— The delay fluctuation in network components
— Timestamping accuracy
— Stability issues

A.5.2.1 Protocol stack delay fluctuation

The simplest implementations of PTP operate as ordinary applications at the top of the network protocol stack. Timestamps are generated at the application level. Protocol stack delay fluctuation will cause errors in these timestamps. These errors are typically in the hundred microseconds to milliseconds range depending on the operating system.

Implementations may generate timestamps at the interrupt level rather than at the application level. In this case delay fluctuation will typically be reduced to tens of microseconds, depending on other use of interrupts by other applications.

The greatest reduction in errors due to protocol stack delay fluctuation is achieved with hardware assist techniques that generate timestamps at the physical layer of the protocol stack. Delay fluctuation at this point is typically in the nanoseconds range. For example, in an Ethernet system, these errors result from the phase lock characteristics of the PHY chips that recover the clock and data synchronization from the incoming data streams. The effect of this delay fluctuation may be reduced by suitable design of the clock servo algorithms.

A.5.2.2 Network component delay fluctuation

Network components introduce fluctuation in the propagation time of messages. This directly affects the accuracy of the offset_from_master and one_way_delay values of the current data set.

Routers typically store and partially parse each message and retransmit when other subnets allow. This delay fluctuation is typically many milliseconds. Protocols such as the Ethernet Network Time Protocol [B9] are designed to manage large systems containing routers and wide area transmission components. PTP is intended for more local systems and therefore uses multicast communications not transmitted by routers.

PTP boundary clocks placed at routers allow PTP to cross router boundaries and replace router delay fluctuation with the much lower delay fluctuation of a normal PTP clock as discussed in A.5.2.1.

Network switches typically found in large Ethernet subnets are also subject to store and forward delay fluctuation. Typical Ethernet switches have input and output buffers communicating over a very high-speed back plane or switch fabric. Each port typically connects directly to an end device or a repeater supporting several end devices. The dominant contribution to delay fluctuation arises from the output buffering. If the output subnet is always available this delay fluctuation is typically in the nanoseconds range and reducible by averaging techniques. Intensive traffic directed at a node containing a PTP clock may cause increased delay fluctuation due to this output buffering. This delay fluctuation is much more difficult to reduce. The proper design of measurement and control systems must recognize this effect and take measures to reduce the impact. For example, short messages generally produce less delay fluctuation than long messages. Likewise, massive data transfer to a particular node may be either broken up or executed at application times when reduced clock synchronization accuracy is acceptable.

Repeaters typically introduce the same level and type of delay fluctuation as the PHY layer of the protocol stacks and can be managed the same way.

A.5.3 Timestamp accuracy

The resolution of the clock generating the timestamps required by PTP must be consistent with the desired accuracy. Note that this resolution is included in the characterization of variance and clock identifier.

A.5.4 Stability issues

As noted in previous sections of this annex, the delay fluctuation introduced into the computation of the offset_to_master and one_way_delay variables may be reduced by suitable design of the synchronization servo algorithms of the local clock. Engineering trade-offs must be made between the averaging times (number of samples) and the responsiveness to effects other than delay fluctuation, such as oscillator stability. Likewise a trade-off must be made between sampling rates (sync interval) and responsiveness to changes in topology (selection of master clocks) and the required computation and network bandwidth resources.

The fundamental time stability of the local clock must be consistent with the required sync interval and accuracy specifications. The algorithms used to reduce delay fluctuation will not correct for drifts of the local clocks during time intervals small compared with the averaging intervals of the algorithms. Servos cannot correct for random drifts occurring within a sync interval.

At high accuracy the specifications on the stability of the local oscillators driving the local clock can be quite difficult to meet. The trade-off will be between cost and stability. Local oscillators will typically be quartz crystals. Quartz crystals typically drift due to thermal, mechanical, and aging effects. Of these, thermal effects are the most difficult in most applications.

For example, a typical thermal specification for uncompensated crystals is 1 PPM per degree Celsius. A one degree temperature rise over a sync interval of 2 seconds will produce an error of roughly 2 microseconds. The thermal environment of the crystal would need to be controlled to this level. Accuracies in the tens of nanosecond range therefore imply that some combination of better thermal specifications on the crystal, reduced sync interval, or better thermal management combine to reduce the thermal drift by two orders of magnitude.

PTP allows sync intervals to be reduced to 1 second, with the corresponding increase in computation and network bandwidth requirements.

Thermal specifications on crystals become increasingly expensive below 1 PPM/degree. Control of the thermal environment must be carefully managed particularly in high accuracy implementations. Very long averaging times typically require oven-controlled crystals or the use of more stable oscillators. Thermal drift during the short intervals and averaging times typical of PTP systems can often be managed by attention to heat dissipation in surrounding devices, cooling patterns within the node, increasing the thermal mass of the oscillator, and similar techniques. See [B10] for a thorough discussion of clock characterization.

Annex B

(informative)

Time scales and epochs in PTP

A more detailed discussion of many of the topics in this annex may be found in [B9], [B11], and [B3].

B.1 General considerations

Within a PTP subdomain, the characteristics of the time available are determined by the grandmaster clock of the subdomain. The grandmaster determines:

- The rate at which time advances. This is measured by how well a time interval determined between any two events as measured by the grandmaster correspond to the same interval measured using a clock consistent with the internationally defined second. This internationally defined second is the measure of time defining the Temps Atomique International (TAI) time base maintained by the Bureau International des Poids et Mesures near Paris.
- The origin or epoch of the time scale.

The possible epochs available for use by the PTP grandmaster clock are as follows (see 6.2.5.6):

- PTP: indicated by a Grandmaster Identifier value of ATOM, GPS, NTP, or HAND. When using the PTP epoch, the time distributed by the grandmaster clock shall be in seconds since the PTP epoch (see 6.2.5.6).
- User specific: indicated by a Grandmaster Identifier value of INIT.
- Unknown or unreliable: indicated by a Grandmaster Identifier value of DFLT.

Which timescale is actually represented in a PTP subdomain depends entirely on the timescale implemented by the grandmaster clock. As will be seen, PTP itself tracks only elapsed time since the epoch. Any correction of PTP times for leap seconds in converting to or from UTC is the responsibility of the application.

All timestamps appearing in PTP messages are expressed in cumulative seconds and nanoseconds since the epoch. The epoch_number represents leading significant bits beyond the seconds field of the TimeRepresentation datatype.

B.2 UTC, TAI, and the PTP epoch

The PTP epoch began at 0 hours on 1 January 1970 (Modified Julian Day 40 587.0).[5] Times measured since this epoch are designated in this standard as PTP Seconds. This is the same epoch used by many computer system algorithms for converting between seconds from the epoch and UTC provided the leap seconds are known for the time of interest.

[5]The Julian Date, JD, is the Julian Day Number, JDN, followed by the fraction of the day elapsed since the preceding Greenwich mean noon. The Julian Day Number is a day count with the origin, JD = 0, at Greenwich mean noon on 1 January 4713 BC. The Modified Julian Date, MJD, is the Julian Date less 2 400 000.5 which shifts the origin to midnight on 17 November 1858. For example: at 0 hours on 1 January 1900, JD = 2 415 020.5, and MJD = 15 020.

In PTP, PTP Seconds are represented in terms of the epoch_number (if applicable), seconds and nanoseconds. The combination of the epoch_number and the seconds may be taken as a (16+32)-bit unsigned = 48-bit unsigned integer. This representation will overflow at 2^{48} seconds = 8 925 512 years, comfortably far into the future. Without the epoch_number, the seconds representation would overflow in 136 years or roughly January of the year 2106.

TAI is the international standard for time based on the SI second as realized on the rotating geoid. TAI is implemented by a suite of atomic clocks and forms the timekeeping basis for other time scales in common use. Of these, UTC is the time scale of most engineering and commercial interest. The UTC representation is specified by ISO 8601:2000 as YYYY-MM-DD for the date and hh:mm:ss for the time in each day.

The rate at which UTC time advances is identical to the rate of TAI. UTC time differs from the TAI time by a constant offset. This offset is modified on occasion by adding or subtracting leap seconds.

Starting on 0 hours of 1 January 1972 (MJD 41,317.0), the world's standard time systems began the implementation of leap seconds to allow only integral second correction between UTC Seconds and TAI, both of which are expressed in days, hours, minutes, and seconds. Leap second corrections, which are applied to UTC but not to TAI, are made preferably following second 23:59:59 of the last day of June or December. The first such correction, a single positive leap second correction, was made following 23:59:59 on 30 June 1972, and UTC was 11 seconds behind TAI following that instant.

PTP Seconds are a count of the elapsed seconds since the PTP epoch without regard to leap seconds. Since 1 January 1972, the PTP timescale corresponds to the TAI timescale with a constant offset of 10 seconds. (The offset between the PTP and TAI time scales is not a constant number of seconds before 1 January 1972.) The combination of PTP time (seconds, nanoseconds, epoch_number) and current number of leap seconds allows the computation of PTP, UTC, or TAI times, given one of them.

The current number of leap seconds is represented in PTP by the value of currentUTCOffset.

If the information is available, PTP distributes a warning prior to a pending change in the value of currentUTCOffset. These changes are indicated by bits in the flags field of Sync messages as follows (see 8.2.10):

— bit PTP_LI_59: When TRUE the last minute of the current day will contain 59 seconds.
— bit PTP_LI_61: When TRUE the last minute of the current day will contain 61 seconds

NOTE—The condition where PTP_LI_59 and PTP_LI_61 are both TRUE defines a state named PTP_ALARM in which the source of the leap second information is not to be trusted.

B.3 Standard time sources

There are two standard time sources of particular interest in implementing PTP systems for which UTC time is required by the application.

The first are systems implementing the NTP protocol widely used in synchronizing computer systems within a campus and around the world. A set of NTP servers to which NTP clients synchronize is maintained. These servers themselves are synchronized to timeservers traceable to international standards. UTC time accuracy from NTP systems is usually in the millisecond range. NTP provides the current time, the current number of leap seconds (supported only in NTP version 4), and the warning flags marking the introduction of a leap second correction, which will be inserted at the end of the current UTC day. NTP does not correct the number of NTP seconds since the NTP epoch whenever a leap second correction is made. (In other words, the NTP clock effectively stops during a leap second, and the time interval occupied by a leap second is effectively "forgotten" once it has been inserted.) The NTP epoch is 0 hours on 1 January 1900. NTP was set at 0 hours on 1 January 1972 to 2 272 060 800.0 to agree with UTC. Currently, NTP represents seconds as a 32-bit unsigned integer. NTP therefore rolls over every 2^{32} seconds = 136 years with the first such rollover occurring in approximately the year 2036.

The second system of interest is the global positioning satellite system, GPS, maintained by the U.S. Department of Defense. UTC time accuracy from the GPS system is usually in the 10–100 ns range. GPS system transmissions represent the time as {GPS Weeks, GPS SecondsInLastWeek}, the number of weeks since the GPS epoch and the number of seconds since the beginning of the current week. From this, GPS Seconds, the number of seconds since the GPS epoch, may be computed. GPS provides the current time, the current number of leap seconds, and the warning flags marking the introduction of a leap second correction. From GPS time, UTC, PTP, and TAI times may be computed using the information contained in the GPS transmissions. The GPS epoch began at 0 hours on 6 January 1980 (MJD 44 244). GPS Weeks are represented in the satellite transmissions modulo 1024 weeks = 19.7 years. The first such rollover occurred between the weeks of 15 August and 22 August 1999. Many but not all commercial systems are believed to have correctly managed this rollover.

Either of these systems may be conveniently used to provide a stratum-1 PTP clock. Relationships between the timescales discussed and examples of times in each system for interesting instants are given in Table B.1 and Table B.2.

Table B.1—Relationships between timescales

From	To	Formula
NTP Seconds	PTP Seconds	PTP Seconds = NTP Seconds –2 208 988 800 + currentUTCOffset
PTP Seconds	NTP Seconds	NTP Seconds = PTP Seconds + 2 208 988 800 – currentUTCOffset
GPS Seconds = (GPS Weeks × 7 × 86 400) + GPS SecondsInLastWeek (GPS Week number must include 1024 × number of rollovers)	PTP Seconds	PTP Seconds = GPS Seconds + 315 964 819
PTP Seconds	GPS Seconds	GPS Seconds = PTP Seconds – 315 964 819

Table B.2—Examples of timescale correspondence

Time/MJD	UTC	leap seconds	PTP seconds	NTP seconds	GPS seconds
0:00:00 15 020	0:00:00 1 Jan. 1900			0 (NTP epoch)	
0:00:00 40 587	0:00:00 1 Jan. 1970		0 (PTP epoch)		
0:00:00 41 317	0:00:00 1 Jan. 1972	10 (integral leap second corrections start)	63 072 000 + 10 = 63 072 010	2 272 060 800	
0:00:00 44 244	0:00:00 6 Jan. 1980	19	315 964 800 + 19 = 315 964 819	2 524 953 600	0 (GPS epoch)
23:59:59 48 256	23:59:59 31 Dec. 1990	25	662 601 600 + 86399 + 25 = 662 688 024	2 871 676 799	346 723 205
23:59:60 48 256	23:59:60 31 Dec. 1990	(leap)	662 601 600 + 86400 + 25 = 662 688 025	2 871 676 799	346 723 206
0:00:00 48 257	0:00:00 1 Jan. 1991	26	662 688 000 + 26 = 662 688 026	2 871 676 800	346 723 207
0:00:01 48 257	0:00:01 1 Jan. 1991	26	662 688 001 + 26 = 662 688 027	2 871 676 801	346 723 208
21:44:58 51 354	21:44:58 25 Jun. 1999	32	930 268 800 + 78 298 + 32 = 930 347 130	3 139 257 600 + 78 298 = 3 139 335 898	614 382 311
16:57:44 51357	16:57:44 28 Jun. 1999	32	930 528 000 + 61 064 + 32 = 930 589 096	3 139 516 800 + 61 064 =3 139 577 864	614 624 277

Annex C

(normative)

subdomain_name to subdomain_address mapping algorithm

The mapping of subdomain_names, with the exception of the names _DFLT, _ALT1, _ALT2, and _ALT3, each followed by (PTP_SUBDOMAIN_NAME_LENGTH -5) null octets, to subdomain_addresses shall be as follows (see 6.2.5.1):

Step1: Compute the CRC32 checksum using the following algorithm.

Step 2: Compute a handle equal to the checksum modulo *N* where *N* is the number of subdomain alternate addresses over which the subdomain_name is to be mapped. For the case where only the PTP specified alternate addresses exist, *N* is 3. All clocks in a PTP system shall have the same value for *N*. The behavior of systems containing clocks with *N* other than 3 is outside the scope of this standard.

Step 3: The handle shall map onto the alternate subdomain_addresses such that:

— handle = 0 corresponds to AlternatePTPdomain1
— handle = 1 corresponds to AlternatePTPdomain2
— handle = 2 corresponds to AlternatePTPdomain3

If additional alternate subdomain_addresses are provided, *N* shall be correspondingly increased. For *N*>3, handle values >2 shall correspond sequentially to the additional subdomain_addresses.

The CRC32 checksum for a given subdomain_name shall have the value produced by the following C code. Note that the length of the subdomain_name pointed to by pInput in the following routine must have a value of PTP_SUBDOMAIN_NAME_LENGTH (see 7.9).

NOTE—This code produces the CRC32 checksum specified by IEEE802.3 clause 3.1.1. As a check of the code, an input string "123456789," length 9, returns a CRC of 0xCBF43926.

In this code, *unsigned long* shall correspond to the PTP datatype UInteger32.

```
// compute the checksum of the subdomain_name
// inputs: pInput a pointer to the domain name character array
// returns the CRC checksum as an unsigned long
unsigned long crc_algorithm (
const void *pInput, unsigned int length)
{
        // pInput pointer to character string of interest,
        // length is strlen of string
        // CRC-32 802.3 polynomial is 0x04c11db7
        // bit-reverse 0x04c11db7:
        unsigned long polynomial = 0xedb88320;
        // crc initialized to all 1s
        unsigned long crc = 0xffffffff;
        unsigned char *blkPtr = (unsigned char *)pInput;
        // sequence thru each byte of the input sequence
        while(length--)
        {
                int i;
                unsigned char data = *(blkPtr++);

                // include each bit of the data, starting with the lsb
                for (i=0; i<8; i++)
                {
                        if((crc^data)&1)
                        {
                                crc = (crc>>1);
                                crc ^= polynomial;
                        }
                        else
                        {
                                crc = (crc>>1);
                        }
                        data >>= 1;
                }
        }
        return crc^0xffffffff;
}
```

For example, the previous algorithm produces the following combinations of subdomain_name, CRC32 checksum, handle, and subdomain_address for the *N*=3 case.

subdomain_name (hex)	**CRC32 checksum**	**handle**	**subdomain_address**
ABCDEFGHIJKLMNOP (0x4142434445464748494A4B4C4D4E4F50)	0xE0E8FF4D	2	AlternatePTPdomain3
abcdf <11 nulls> (0x6162636466000000000000000000000)	0x68B370E3	1	AlternatePTPdomain2
abcdg<11 nulls> (0x6162636467000000000000000000000)	0xF3163C8C	0	AlternatePTPdomain1
ABCDEF!#$%35wxyz (0x41424344454621232425333577 78797A)	0x2A53C933	2	AlternatePTPdomain3

Annex D

(normative)

Ethernet implementation of PTP

This clause specifies those portions of the PTP standard that are specific to Ethernet implementations. For this version of the standard the value of the versionNetwork field of PTP messages shall be 1. The specifications in this annex shall apply to all PTP implementations using Ethernet as a communication network. The specifications in this annex may in the future be superseded by specifications of the IETF. Such specifications superseding this annex shall meet all other requirements of this standard. Balloted revisions to this annex or being superceded by IETF specification shall cause the value of the field versionNetwork to be incremented by +1 (one) from the value existing at the time of superseding or revision.

D.1 Message on-the-wire formats

The on-the-wire formats for the five types of PTP messages are specified in this clause for an Ethernet implementation.

D.1.1 message timestamp point

The PTP message timestamp point (see 6.2.2.3) shall correspond to the leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet of an Ethernet packet (IEEE Std 802.3-2002). This point is illustrated in Figure D.1.

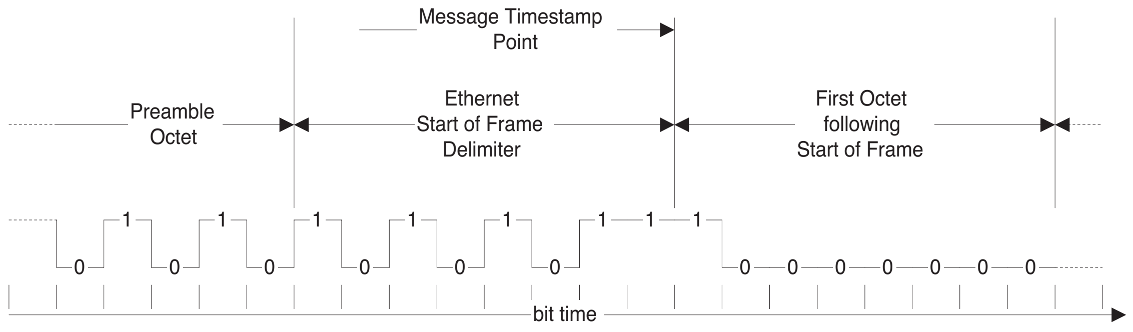
Message Timestamp Point
Preamble Octet
Ethernet Start of Frame Delimiter
First Octet following Start of Frame
1
0
bit time

Figure D.1—Message timestamp point

D.1.2 PTP defined data type mappings

Messages are placed on the wire, marshaled, starting with the 0th octet and proceeding in order until the last octet is transmitted. Within each octet, the least significant bit is transmitted first. All numeric data types are marshaled as big endian. Note that the only exception (not of concern to the PTP implementer) is the Ethernet Frame CRC (last four Octets) where the (hardware-computed) CRC bit order is reversed on the wire.

D.1.3 PTP message formats

D.1.3.1 Ethernet, IPV4, and UDP headers for all PTP messages

N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name
0	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	destination MAC address (see D.3.1)
4	h_8h_9	$h_{10}h_{11}$	k_0k_1	k_2k_3	Octet[6] (cont) | Octet[6]	destination MAC address (cont) | source MAC address
8	k_4k_5	k_6k_7	k_8k_9	$k_{10}k_{11}$	Octet[6] (cont)	source MAC address (cont)
12	0x08	00	0x45	00	UInteger16 | UInteger8 | UInteger8	type (0x0800 = IP datagram) | version IPV4, IP header length (5x4-32bit words)| type of service (0)
16	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	IP datagram length | datagram sequence number
20	00	00	0x01	0x11	Octet[2] | UInteger8 | UInteger8	flags/fragments | time to live (TTL) | Protocol: UDP
24	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | Octet[4]	IP header checksum | source IP address
28	k_4k_5	k_6k_7	h_0h_1	h_2h_3	Octet[4] (cont) | Octet[4]	source IP address (cont) | destination IP address (see D.3.1)
32	h_4h_5	h_6h_7	k_0k_1	k_2k_3	Octet[4] (cont) | UInteger16	destination IP address (cont) | source port number
36	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	destination port number (see D.3.1) | UDP length
40	h_0h_1	h_2h_3	N/A	N/A	UInteger16	UDP checksum

N is a count of octets starting with zero as the octet following the Start of Frame delimeter.

D.1.3.2 PTP Sync and Delay_Req message specification (UDP user payload portion)

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.2 and 8.3
42	0	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	versionPTP | versionNetwork
46	4	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[16]	subdomain
50	8	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[16](cont)	subdomain(cont)
54	12	$h_{16}h_{17}$	$h_{18}h_{19}$	$h_{20}h_{21}$	$h_{22}h_{23}$	Octet[16](cont)	subdomain(cont)
58	16	$h_{24}h_{25}$	$h_{26}h_{27}$	$h_{28}h_{29}$	$h_{30}h_{31}$	Octet[16](cont)	subdomain(cont)
62	20	k_0k_1	j_0j_1	h_0h_1	h_2h_3	UInteger8 | UInteger8 | Octet[6]	messageType | sourceCommunica-tionTechnology | sourceUuid
66	24	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	sourceUuid(cont)
70	28	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	sourcePortId | sequenceId

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.2 and 8.3
74	32	j_0j_1	00	h_0h_1	h_2h_3	UInteger8 | Octet | Octet[2]	control | reserved | flags
78	36	00	00	00	00	Octet[4]	reserved
82	40	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	originTimestamp (seconds)
86	44	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	originTimestamp (nanoseconds)
90	48	k_0k_1	k_2k_3	h_0h_1	h_2h_3	UInteger16 | Integer16	epochNumber | currentUTCOffset
94	52	00	j_0j_1	h_0h_1	h_2h_3	Octet | UInteger8 | Octet[6]	grandmasterCommunicationTechnology | grandmasterClockUuid
98	56	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	grandmasterClockUuid (cont)
102	60	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	grandmasterPortId | grandmasterSequenceId
106	64	00	00	00	h_0h_1	UInteger8	grandmasterClockStratum
110	68	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[4]	grandmasterClockIdentifier
114	72	00	00	h_0h_1	h_2h_3	Integer16	grandmasterClockVariance
118	76	00	k_0k_1	00	h_0h_1	Octet | UInteger8 | Octet | UInteger8	grandmasterPreferred | grandmasterIsBoundaryClock
122	80	00	00	00	h_0h_1	Integer8	syncInterval
126	84	00	00	h_0h_1	h_2h_3	Integer16	localClockVariance
130	88	00	00	h_0h_1	h_2h_3	UInteger16	localStepsRemoved
134	92	00	00	00	h_0h_1	UInteger8	localClockStratum
138	96	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[4]	localClockIdentifier
142	100	00	j_0j_1	h_0h_1	h_2h_3	Octet | UInteger8 | Octet[6]	parentCommunicationTechnology | parentUuid
148	104	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	parentUuid(cont)
152	108	00	00	h_0h_1	h_2h_3	UInteger16	parentPortField
156	112	00	00	h_0h_1	h_2h_3	Integer16	estimatedMasterVariance
160	116	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	estimatedMasterDrift
164	120	00	00	00	h_0h_1	Boolean	utcReasonable

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.3 Follow_Up messages specification (UDP user payload portion)

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.2 and 8.4
42	0	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	versionPTP | versionNetwork
46	4	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[16]	subdomain
50	8	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[16](cont)	subdomain(cont)
54	12	$h_{16}h_{17}$	$h_{18}h_{19}$	$h_{20}h_{21}$	$h_{22}h_{23}$	Octet[16](cont)	subdomain(cont)
58	16	$h_{24}h_{25}$	$h_{26}h_{27}$	$h_{28}h_{29}$	$h_{30}h_{31}$	Octet[16](cont)	subdomain(cont)
62	20	k_0k_1	j_0j_1	h_0h_1	h_2h_3	UInteger8 | UInteger8 | Octet[6]	messageType | sourceCommunicationTechnology | sourceUuid
66	24	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	sourceUuid(cont)
70	28	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	sourcePortId | sequenceId
74	32	j_0j_1	00	h_0h_1	h_2h_3	UInteger8 | Octet | Octet[2]	control | reserved | flags
78	36	00	00	00	00	Octet[4]	reserved
82	40	00	00	h_0h_1	h_2h_3	Octet[2] | UInteger16	associatedSequenceId
86	44	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	preciseOriginTimestamp (seconds)
90	48	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	preciseOriginTimestamp (nanoseconds)

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.4 Delay_Resp messages specification (UDP user payload portion)

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.2 and 8.5
42	0	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	versionPTP | versionNetwork
46	4	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[16]	subdomain
50	8	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[16](cont)	subdomain(cont)
54	12	$h_{16}h_{17}$	$h_{18}h_{19}$	$h_{20}h_{21}$	$h_{22}h_{23}$	Octet[16](cont)	subdomain(cont)
58	16	$h_{24}h_{25}$	$h_{26}h_{27}$	$h_{28}h_{29}$	$h_{30}h_{31}$	Octet[16](cont)	subdomain(cont)
62	20	k_0k_1	j_0j_1	h_0h_1	h_2h_3	UInteger8 | UInteger8 | Octet[6]	messageType | sourceCommunicationTechnology | sourceUuid
66	24	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	sourceUuid(cont)
70	28	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	sourcePortId | sequenceId
74	32	j_0j_1	00	h_0h_1	h_2h_3	UInteger8 | Octet | Octet[2]	control | reserved | flags
78	36	00	00	00	00	Octet[4]	reserved
82	40	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	delayReceiptTimestamp (seconds)

D.1.3.4 Delay_Resp messages specification (UDP user payload portion) *(continued)*

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.2 and 8.5
86	44	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	delayReceiptTimestamp (nanoseconds)
90	48	00	j_0j_1	h_0h_1	h_2h_3	Octet | UInteger8 | Octet[6]	requestingSourceCommunicationTechnology | requestingSourceUuid
94	52	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	requestingSourceUuid (cont)
98	56	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	requestingSourcePortId | requestingSourceSequenceId

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.5 Management message, parameterLength = 0, specification (UDP user payload portion)

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, clauses 8.2 and 8.6
42	0	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	versionPTP | versionNetwork
46	4	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[16]	subdomain
50	8	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[16](cont)	subdomain(cont)
54	12	$h_{16}h_{17}$	$h_{18}h_{19}$	$h_{20}h_{21}$	$h_{22}h_{23}$	Octet[16](cont)	subdomain(cont)
58	16	$h_{24}h_{25}$	$h_{26}h_{27}$	$h_{28}h_{29}$	$h_{30}h_{31}$	Octet[16](cont)	subdomain(cont)
62	20	k_0k_1	j_0j_1	h_0h_1	h_2h_3	UInteger8 | UInteger8 | Octet[6]	messageType | sourceCommunicationTechnology | sourceUuid
66	24	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	sourceUuid(cont)
70	28	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	sourcePortId | sequenceId
74	32	j_0j_1	00	h_0h_1	h_2h_3	UInteger8 | Octet | Octet[2]	control | reserved | flags
78	36	00	00	00	00	Octet[4]	reserved
82	40	00	j_0j_1	h_0h_1	h_2h_3	Octet | UInteger8 | Octet[6]	targetCommunicationTechnology | targetUuid
86	44	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	targetUuid (cont)
90	48	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | Integer16	targetPortId | startingBoundaryHops
94	52	k_0k_1	k_2k_3	00	h_0h_1	Integer16 | UInteger8	boundaryHops | managementMessageKey
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 0

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6 Management message, parameterLength >0, specification (UDP user payload portion)

The portion of management messages with nonzero parameterLength common to all such messages shall be as shown in the following table. The details for the parameterLength and messageParameters fields for these messages are shown in the subclauses.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, clauses 8.2 and 8.6
42	0	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	versionPTP | versionNetwork
46	4	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[16]	subdomain
50	8	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[16](cont)	subdomain(cont)
54	12	$h_{16}h_{17}$	$h_{18}h_{19}$	$h_{20}h_{21}$	$h_{22}h_{23}$	Octet[16](cont)	subdomain(cont)
58	16	$h_{24}h_{25}$	$h_{26}h_{27}$	$h_{28}h_{29}$	$h_{30}h_{31}$	Octet[16](cont)	subdomain(cont)
62	20	k_0k_1	j_0j_1	h_0h_1	h_2h_3	UInteger8 | UInteger8 | Octet[6]	messageType | sourceCommunicationTechnology | sourceUuid
66	24	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	sourceUuid(cont)
70	28	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | UInteger16	sourcePortId | sequenceId
74	32	j_0j_1	00	h_0h_1	h_2h_3	UInteger8 | Octet | Octet[2]	control | reserved | flags
78	36	00	00	00	00	Octet[4]	reserved
82	40	00	j_0j_1	h_0h_1	h_2h_3	Octet | UInteger8 | Octet[6]	targetCommunicationTechnology | targetUuid
86	44	h_4h_5	h_6h_7	h_8h_9	$h_{10}h_{11}$	Octet[6](cont)	targetUuid (cont)
90	48	h_0h_1	h_2h_3	k_0k_1	k_2k_3	UInteger16 | Integer16	targetPortId | startingBoundaryHops
94	52	k_0k_1	k_2k_3	00	h_0h_1	Integer16 | UInteger8	boundaryHops | managementMessageKey
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength (=N)
102	60	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[N]	messageParameters
106	64	…	…	…	…	Octet[N]	messageParameters(cont)
…	…	…	…	…	$h_{N-2}h_{N-1}$	Octet[N]	messageParameters(cont)

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.1 PTP_MM_CLOCK_IDENTITY

The value of the parameterLength field shall be 64.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.3
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 64
102	60	00	00	00	h_0h_1	UInteger8	clockCommunicationTechnology
106	64	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	clockUuidField
110	68	h_8h_9	$h_{10}h_{11}$	00	00	Octet[6] (cont)	clockUuidField(cont)
114	72	00	00	h_0h_1	h_2h_3	UInteger16	clockPortField
118	76	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[48]	manufacturerIdentity
122	80	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[48] (cont)	manufacturerIdentity (cont)
…	…	…					
162	120	$h_{88}h_{89}$	$h_{90}h_{91}$	$h_{92}h_{93}$	$h_{94}h_{95}$	Octet[48] (cont)	manufacturerIdentity (cont)

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.2 PTP_MM_INITIALIZE_CLOCK

The value of the parameterLength field shall be 4.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 7.12.4
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 4
102	60	00	00	h_0h_1	h_2h_3	UInteger16	initializationKey

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.3 PTP_MM_SET_SUBDOMAIN

The value of the parameterLength field shall be 16.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.5
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 16
102	60	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[16]	subdomainName
106	64	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[16] (cont)	subdomainName (cont)
110	68	$h_{16}h_{17}$	$h_{18}h_{19}$	$h_{20}h_{21}$	$h_{22}h_{23}$	Octet[16] (cont)	subdomainName (cont)
114	72	$h_{24}h_{25}$	$h_{26}h_{27}$	$h_{28}h_{29}$	$h_{30}h_{31}$	Octet[16] (cont)	subdomainName (cont)

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.4 PTP_MM_DEFAULT_DATA_SET

The value of the parameterLength field shall be 76.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.9
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 76
102	60	00	00	00	h_0h_1	UInteger8	clockCommunicationTechnology
106	64	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	clockUuidField
110	68	h_8h_9	$h_{10}h_{11}$	00	00	Octet[6](cont)	clockUuidField(cont)
114	72	00	00	h_0h_1	h_2h_3	UInteger16	clockPortField
118	76	00	00	00	h_0h_1	UInteger8	clockStratum
122	80	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[4].	clockIdentifier
126	84	00	00	h_0h_1	h_2h_3	Integer16	clockVariance
130	88	00	00	00	h_0h_1	Boolean	clockFollowupCapable
134	92	00	00	00	h_0h_1	Boolean	preferred
138	96	00	00	00	h_0h_1	Boolean	initializable
142	100	00	00	00	h_0h_1	Boolean	externalTiming
146	104	00	00	00	h_0h_1	Boolean	isBoundaryClock
150	108	00	00	00	h_0h_1	Integer8	syncInterval
154	112	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[16]	subdomainName
158	116	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[16](cont)	subdomainName(cont)
162	120	$h_{16}h_{17}$	$h_{18}h_{19}$	$h_{20}h_{21}$	$h_{22}h_{23}$	Octet[16](cont)	subdomainName(cont)
166	124	$h_{24}h_{25}$	$h_{26}h_{27}$	$h_{28}h_{29}$	$h_{30}h_{31}$	Octet[16](cont)	subdomainName(cont)
170	128	00	00	h_0h_1	h_2h_3	UInteger16	numberPorts
174	132	00	00	h_0h_1	h_2h_3	UInteger16	numberForeignRecords

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.5 PTP_MM_UPDATE_DEFAULT_DATA_SET

The value of the parameterLength field shall be 36.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.10
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 36
102	60	00	00	00	h_0h_1	UInteger8	clockStratum
106	64	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[4].	clockIdentifier
110	68	00	00	h_0h_1	h_2h_3	Integer16	clockVariance
114	72	00	00	00	h_0h_1	Boolean	preferred
118	76	00	00	00	h_0h_1	Integer8	syncInterval
122	80	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[16]	subdomainName
126	84	h_8h_9	$h_{10}h_{11}$	$h_{12}h_{13}$	$h_{14}h_{15}$	Octet[16](cont)	subdomainName(cont)
130	88	$h_{16}h_{17}$	$h_{18}h_{19}$	$h_{20}h_{21}$	$h_{22}h_{23}$	Octet[16](cont)	subdomainName(cont)
134	92	$h_{24}h_{25}$	$h_{26}h_{27}$	$h_{28}h_{29}$	$h_{30}h_{31}$	Octet[16](cont)	subdomainName(cont)

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.6 PTP_MM_CURRENT_DATA_SET

The value of the parameterLength field shall be 20.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.12
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 20
102	60	00	00	h_0h_1	h_2h_3	UInteger16	stepsRemoved
106	64	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	offsetFromMasterSeconds
110	68	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	offsetFromMasterNanoseconds
114	72	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	oneWayDelaySeconds
118	76	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	oneWayDelayNanoseconds

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.7 PTP_MM_PARENT_DATA_SET

The value of the parameterLength field shall be 90.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.14
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 90
102	60	00	00	00	h_0h_1	UInteger8	parentCommunicationTechnology
106	64	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	parentUuid
110	68	h_8h_9	$h_{10}h_{11}$	00	00	Octet[6](cont)	parentUuid (cont)
114	72	00	00	h_0h_1	h_2h_3	UInteger16	parentPortId
118	76	00	00	h_0h_1	h_2h_3	UInteger16	parentLastSyncSequenceNumber
122	80	00	00	00	h_0h_1	Boolean	parentFollowupCapable
126	84	00	00	00	h_0h_1	Boolean	parentExternalTiming
130	88	00	00	h_0h_1	h_2h_3	Integer16	parentVariance
134	92	00	00	00	h_0h_1	Boolean	parentStats
138	96	00	00	h_0h_1	h_2h_3	Integer16	observedVariance
142	100	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	observedDrift
146	104	00	00	00	h_0h_1	Boolean	utcReasonable
150	108	00	00	00	h_0h_1	UInteger8	grandmasterCommunicationTechnology
154	112	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	grandmasterUuidField
158	116	h_8h_9	$h_{10}h_{11}$	00	00	Octet[6](cont)	grandmasterUuidField(cont)
162	120	00	00	h_0h_1	h_2h_3	UInteger16	grandmasterPortIdField
166	124	00	00	00	h_0h_1	UInteger8	grandmasterStratum
170	128	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[4].	grandmasterIdentifier
174	132	00	00	h_0h_1	h_2h_3	Integer16	grandmasterVariance
178	136	00	00	00	h_0h_1	Boolean	grandmasterPreferred
182	140	00	00	00	h_0h_1	Boolean	grandmasterIsBoundaryClock
186	144	00	00	h_0h_1	h_2h_3	UInteger16	grandmasterSequenceNumber

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.8 PTP_MM_PORT_DATA_SET

The value of the parameterLength field shall be 52.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.16
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 52
102	60	00	00	h_0h_1	h_2h_3	UInteger16	returnedPortNumber
106	64	00	00	00	h_0h_1	UInteger8	portState
110	68	00	00	h_0h_1	h_2h_3	UInteger16	lastSyncEventSequenceNumber
114	72	00	00	h_0h_1	h_2h_3	UInteger16	lastGeneralEventSequenceNumber
118	76	00	00	00	h_0h_1	UInteger8	portCommunicationTechnology
122	80	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	portUuidField
126	84	h_8h_9	$h_{10}h_{11}$	00	00	Octet[6](cont)	portUuidField (cont)
130	88	00	00	h_0h_1	h_2h_3	UInteger16	portIdField
134	92	00	00	00	h_0h_1	Boolean	burstEnabled
138	96	00	h_0h_1	k_0k_1	m_0m_1	UInteger8| UInteger8| UInteger8	subdomainAddressOctets = 4 | eventPortAddressOctets = 2 | generalPortAddressOctets = 2
142	100	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[4]	subdomainAddress
146	104	00	00	h_0h_1	h_2h_3	Octet[2]	eventPortAddress
150	108	00	00	h_0h_1	h_2h_3	Octet[2]	generalPortAddress

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

The subdomain_address field shall have the value 0xE0000181 to 0xE0000184 for the subdomains _DFLT, _ALT1, _ALT2, and _ALT3 respectively. These correspond to the dotted decimal notations 224.0.1.129 to 224.0.1.132.

The event_port_address field shall have the value 0x013F corresponding to the decimal value 319.

The general_port_address field shall have the value 0x0140 corresponding to the decimal value 320.

D.1.3.6.9 PTP_MM_GLOBAL_TIME_DATA_SET

The value of the parameterLength field shall be 24.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.18
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 24
102	60	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	localTimeSeconds
106	64	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	localTimeNanoseconds
110	68	00	00	h_0h_1	h_2h_3	Integer16	currentUtcOffset
114	72	00	00	00	h_0h_1	Boolean	leap59
118	76	00	00	00	h_0h_1	Boolean	leap61
122	80	00	00	h_0h_1	h_2h_3	UInteger16	epochNumber

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.10 PTP_MM_UPDATE_GLOBAL_TIME_PROPERTIES

The value of the parameterLength field shall be 16.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.19
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 16
102	60	00	00	h_0h_1	h_2h_3	Integer16	currentUtcOffset
106	64	00	00	00	h_0h_1	Boolean	leap59
110	68	00	00	00	h_0h_1	Boolean	leap61
114	72	00	00	h_0h_1	h_2h_3	UInteger16	epochNumber

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.11 PTP_MM_GET_FOREIGN_DATA_SET

The value of the parameterLength field shall be 4.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.21
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 4
102	60	00	00	h_0h_1	h_2h_3	UInteger16	recordKey

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.12 PTP_MM_FOREIGN_DATA_SET

The value of the parameterLength field shall be 28.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.22
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 28
102	60	00	00	h_0h_1	h_2h_3	UInteger16	returnedPortNumber
106	64	00	00	h_0h_1	h_2h_3	UInteger16	returnedRecordNumber
110	68	00	00	00	h_0h_1	UInteger8	foreignMasterCommunicationTechnology
114	72	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Octet[6]	foreignMasterUuidField
118	76	h_8h_9	$h_{10}h_{11}$	00	00	Octet[6](cont)	foreignMasterUuidField (cont)
122	80	00	00	h_0h_1	h_2h_3	UInteger16	foreignMasterPortIdField
126	84	00	00	h_0h_1	h_2h_3	UInteger16	foreignMasterSyncs

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.13 PTP_MM_SET_SYNC_INTERVAL

The value of the parameterLength field shall be 4.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.23
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 4
102	60	00	00	h_0h_1	h_2h_3	Integer16	syncInterval

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.1.3.6.14 PTP_MM_SET_TIME

The value of the parameterLength field shall be 8.

SOF	N	Octet N	Octet N+1	Octet N+2	Octet N+3	Type (informative)	Field name, 8.6.1.10.28
98	56	00	00	h_0h_1	h_2h_3	UInteger16	parameterLength = 8
102	60	h_0h_1	h_2h_3	h_4h_5	h_6h_7	UInteger32	localTimeSeconds
106	64	h_0h_1	h_2h_3	h_4h_5	h_6h_7	Integer32	localTimeNanoseconds

SOF is a count of octets starting with zero as the octet following the Start of Frame delimeter.
N is a count of octets starting with zero as the octet following the last octet of the header defined in D.1.3.1.

D.2 Ethernet PTP UUIDs

Subclause 6.2.4.1 specifies that each clock supporting PTP has a universal unique identifier within a PTP synchronization domain. Nodes may be administrative nodes, clocks, or a combination of the two.

In an Ethernet node implementation, the uuid_field of a UUID meeting the requirements of 6.2.4.1 shall be an Ethernet MAC address uniquely associated with the node, whether administrative node only or an actual clock. If a node has multiple Ethernet MAC addresses associated with it, then one of these shall be selected to be the uuid_field for purposes of PTP. If the node has multiple clocks, the uuid_field for each shall meet the requirements of 6.2.4.1 and shall be an Ethernet MAC address uniquely associated with the respective clock.

D.3 IEEE 802.3 (Ethernet) addressing for PTP

Subclause 6.2.5 specified two PTP ports and a set of PTP multicast addresses for establishing the communication patterns within the PTP protocol. In an Ethernet implementation, the PTP ports and PTP multicast addresses are mapped onto UDP ports and IP multicast addresses. This annex specifies the assignment of IP destination addresses and UDP port numbers in the PTP protocol.

In an Ethernet implementation, all PTP messages shall be UDP messages.

The engineering trade-offs made in selecting the specified assignments are also given.

D.3.1 Addressing specification

Two port numbers are used in the PTP protocol. These port numbers are defined in the following table:

Port category	Purpose	IANA Name[a]	Value
EventPort	communicates PTP Sync or Delay_Req messages	ptp-event	319
GeneralPort	communicates PTP Follow_Up, Delay_Resp or Management messages	ptp-general	320

[a]The IANA, Internet Assigned Numbers Authority, assigned the dedicated multicast addresses and port numbers shown along with the IANA Names. These names appear in the IANA listings identifying multicast addresses and port numbers.

Four multicast addresses are specified in the PTP protocol. These addresses for an Ethernet implementation are defined in the following table. The *time to live* (IPV4 header byte 8, message byte 22) or *hop limit* (IPV6 header byte 7, message byte 21) value for all PTP messages shall be 0. That is, these messages shall not be forwarded by network routers. From 6.2.5, other optional multicast addresses may be used but the management of systems utilizing optional addresses is outside the scope of this standard.

Address name	Purpose	IANA Name[a]	Value
DefaultPTPdomain	Defines the default synchronization subdomain of a PTP system	PTP-primary	224.0.1.129
AlternatePTPdomain1	Defines an alternate synchronization subdomain	PTP-alternate1	224.0.1.130
AlternatePTPdomain2	Defines an alternate synchronization subdomain	PTP-alternate2	224.0.1.131
AlternatePTPdomain3	Defines an alternate synchronization subdomain	PTP-alternate3	224.0.1.132
(optional subdomains)	Implementation-specific subdomains	N/A	Outside the scope of this standard

[a]The IANA, Internet Assigned Numbers Authority, assigned the dedicated multicast addresses and port numbers shown along with the IANA Names. These names appear in the IANA listings identifying multicast addresses and port numbers.These listings can be found at http://www.iana.org.

D.3.2 Addressing rationale (Informative)

Five logical communication paths occur in a PTP system:

— Master clock to slave clock
— Master clock to master clock
— Slave clock to master clock
— Administration tool to either a master or a slave clock
— Master or slave clock to administration tool

Four types of messages are carried over these logical links:

— Sync or Delay_Req messages that are the events timestamped by the clocks
— Follow_Up messages that report the transmit time of a Sync message
— Delay_Resp messages used in the delay measurement
— Management messages used for monitoring and configuration of PTP clock systems

The useful combinations of communication paths and message types are indicated in Table D.1. Unused combinations are indicated by *N/A*. For the useful combinations, the normal fan-out (that is, the cardinality) of the communication is given. The frequency of occurrence for each message type during normal steady state operation with the default value of sync interval is also given.

During transients, such as startup or change of master clock, traffic will increase.

The semantics of the Follow_Up, Delay_Resp and Management messages are contained entirely in the message data. The semantics of the Sync or Delay_Req message are contained in the message data and in the time of message transmission and receipt by the sending and receiving clocks. PTP is predicated on the precise measurement of these times for Sync or Delay_Req messages, and it is therefore necessary to easily distinguish Sync or Delay_Req messages from all other network traffic. It is also desirable to distinguish between Follow_Up, general and management messages at as low a level as practical to eliminate unnecessary processor cycles devoted to PTP. Overlaid on all of this is the notion of multiple, independent PTP synchronization subdomains.

Table D.1—Cardinality of message passing

Path/Message type	Sync or Delay_Req (~1/sec +1/min/slave)	Follow_Up (~1/sec)	Delay_Resp (~1/min/slave)	Management (rarely)
Master to slave	1:N	1:N	1:1	N/A
Master to master	N:1	N/A	N/A	N/A
Slave to master	1:1	N/A	N/A	N/A
Administrator to master or slave	N/A	N/A	N/A	1:1 or 1:N
Master or slave to administrator	N/A	N/A	N/A	1:1

In the IP-based Ethernet protocols, there are two low-level mechanisms available for disambiguating messages: the IP destination address and the port number. In typical IP protocol stacks, destination addresses are resolved in hardware and ports are resolved in the lowest level of IP software without involving application level software (except for setup). In principle, either destination addresses or port numbers can be used singly or in combination to satisfy the requirements outline above.

Synchronization subdomains within the same Ethernet collision domain could be implemented in the application space of the PTP protocol but this would require all clocks to process messages for all subdomains. To minimize useless processor cycles it appears that subdomains should be implemented at the address level. If this is done at the address level, only unicast or multicast is adequate. Broadcast does not provide the needed differentiation. PTP specifies that in an administration free implementation, the default synchronization subdomain be used. PTP provides administrative facilities for configuring a clock to be in one of the alternate subdomains. Only three alternate subdomain_addresses are specified. The most probable use of subdomains is to allow multiple subsystems in the same collision domain to be self-consistent with respect to time but not depend on the presence of a master clock in one of the other subsystems. If more than four such subdomains appear to be needed, PTP recommends that the collision domain include an administratively designated master clock, independent of any of the subsystems, serving as the master clock for all. The alternative to using multicast addresses to implement subdomains is to use either port numbers or internal structure in the protocol user space of the message. A look at the IANA assigned port numbers and multicast addresses suggests that most protocols use addresses for similar purposes and that this would be more comfortable to the user community. In addition, the rejection of PTP messages from a foreign subdomain is slightly more efficient using addresses. Using protocol user space would unduly burden processors with the task of rejecting uninteresting messages.

From Table D.1, the only significant cases of one-to-one communication are slave to master for Delay_Req messages and master to slave for Delay_Resp messages. One-to-one communications could be implemented with unicast addresses, which are more efficient in terms of processor burden. However multicast addresses have been specified since they require less administration.

The slave to master one-to-one communications could be implemented with a unicast address for the master. This would require all clocks to listen both on a multicast address for messages from a master and on its own IP unicast address if it is a master. In addition, the slaves would need to be able to change the destination address of the Delay_Req messages it issues if the master changes. The advantage of such a scheme is that only the master need parse messages from a particular slave. The disadvantage is that the slave must be able to reconfigure a unicast port, and a master must listen on two addresses. Neither requires any action by the administration facilities of the protocol. In steady-state operation, these slave-issued messages are relatively infrequent. The number of such messages does scale linearly with the number of slaves. In synchronization subdomains of less than approximately 50 clocks, the PTP traffic is still on the order of two messages per second for the default value of sync interval. The added burden of multiple, configurable IP sockets does not seem worthwhile.

The master to slave one-to-one Delay_Resp communication case is slightly different. The traffic, scaling, and processing cycle penalty with the number of slaves is identical with the slave to master one-to-one communication. The use of one-to-one communications in this case would require the master to maintain and use the unicast address of all slaves. This does not seem worthwhile.

In principle, all 1:*N* communications could be handled by a sequence of unicast messages. However, the costs in maintaining address tables and network traffic that scales from the current two messages/second linearly with the number of clocks is not worthwhile. A unicast-based scheme would also either require administration or greatly complicate the current protocol.

Annex E

(informative)

Bibliography

E.1 General bibliography

[B1] IDL is specified by The Object Management Group (OMG). CORBA/IIOP 2.2 Specification: Framingham, MA, February 1998. See also http://www.opengroup.org/regproducts/obm0.htm.

A third-party reference discussing the same issues is *Inside CORBA,* Mowbray, Thomas and Ruh, William, Addison-Wesley, 1997, ISBN 0-2-1-89540-4.

[B2] Perlman, Radia, *Interconnections, Bridges and Routers*. Addison-Wesley, 1992, ISBN 0-201-56332-0.

E.2 References pertaining to time specification

[B3] Allan, David W., Ashby, Neil, and Hodge, Cliff, "Fine-tuning Time in the Space Age," *IEEE Spectrum,* March 1998.

[B4] Eidson, John C., et al., Method for Recognizing Events and Synchronizing Clocks, U.S. Patent 5,566,180. October 15, 1996.

[B5] "Draft Revision of IEEE Std 1139-1988, IEEE Standard Definitions of Physical Quantities For Fundamental Frequency and Time Metrology—Random Instabilities." E.S. Ferre-Pikal, J.R.Vig, J.C. Camparo, L.S. Cutler, L. Maleki, W.J. Riley, S.R. Stein, C. Thomas, F.L. Walls, and J.D. White, 1997 *IEEE International Frequency Control Symposium Proceedings*.

[B6] Mills, L. David, "On the Chronometry and Metrology of Computer Network Timescales and their Application to the Network Time Protocol," *ACM Computer Communications Review,* vol. 21, no. 5, pp. 8-17, October 1991.

[B7] Mills, L. David, "A Kernel Model for Precision Timekeeping," RFC 1589, March 1994.

[B8] Mogul, J., and Stone, J., Pulse-Per-Second API for UNIX-like Operating Systems, RFC 2783, March 2000.

[B9] Network Time Protocol (Version 3), RFC-1305, March 1992.

[B10] Sullivan, D.B., Allan, D.W., Howe, D.A., Walls, F.L., editors, *Characterization of Clocks and Oscillators,* NIST Technical Note 1337, March 1990.

[B11] U.S. Naval Observatory web page, items and pointers on <http://tycho.usno.navy.mil/time.html>.