State Machines and Message Formats for IEEE 802.1AS

Revision 0.0
2007.03.08

Geoffrey M. Garner
Samsung
gmgarner@comcast.net

Revision history:

0.0 – Initial document. Contains detailed state machines and corresponding C code for Pdelay requestor port, Pdelay responder port, peer-to-peer (P2P) transparent clock (TC) node, receipt of Sync and Follow_Up by P2P TC port, receipt of Sync and Follow_Up by ordinary clock (OC), and sending of Sync and Follow_Up by OC. Also contains message formats, copied and pasted from IEEE 1588 version 2, Draft D1-C [1], with material not of interest to IEEE 802.1AS omitted.

1. Introduction

This document provides state machines and message formats for IEEE 802.1AS. The actions in the state machines are given in the form of C code. The state machines include: (1) Pdelay responder port, (2) Pdelay requestor port, (3) peer-to-peer transparent clock node, (4) receipt of Sync and Follow_Up by a peer-to-peer transparent clock port, (5) receipt of Sync and Follow_Up by an ordinary clock, (6) sending of Sync and Follow_Up by an ordinary clock, and (7) receiving and sending of Announce by an ordinary clock. The latter state machine is a simplified version. Message formats are provided for Announce, Sync, Follow_Up, Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up. Each message contains a PTP common header, whose format is provided prior to the message formats (this is to avoid repeating the common header information with each message). Data types are summarized prior to the common header and messages.

2. State machines and corresponding C code

The state machines and corresponding C code for the various P2P TC and OC functions are given in the following subsections. The notation used describes state transition diagrams using the Mealy style, where actions are associated with the transition from one state to another. The following description is taken from [1] (with additional description added on how C code is used with the state machines).

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 and

— The transition action label for a transition

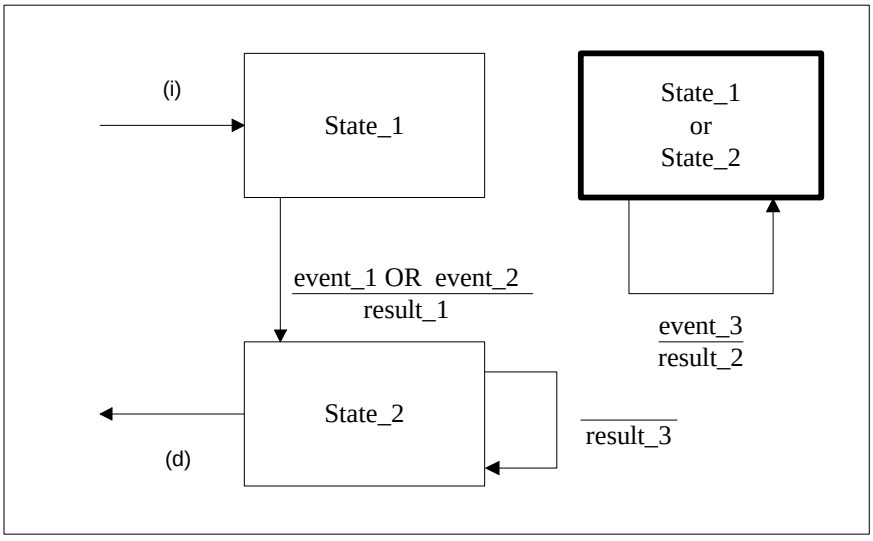

Figure 1 : Mealy state transition diagram

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. In the state machines here, the predicates are indicated using standard C language constructs, e.g., ‘&&’ is used for ‘AND’ and ‘||’ is used for ‘OR’.

Transition actions, for example “result_1,” are the actions that are executed before transitioning to the next state. These are indicated in the state machines here using C code. Small amounts of C code are shown next to the associated transition, below the horizontal line that separates the event and action. In cases where the amount of C code is too large to fit in the diagram next to the transition, the code is placed in a C function, and the function is indicated in the diagram next to the transition. The function is in turn defined in the same figure (i.e., the figure containing the state machine) off to the side.

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

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. Note that this construct is not used in the state machines of Revision 0.0 of this document, but may be used in future revisions.

Transitions, for example the transition resulting in result_3, that have no indicated enabling conditions, occur via unspecified mechanisms. Unless otherwise stated, in PTP the events giving rise to these mechanisms are implementation-specific and outside the scope of the standard. Note that this construct is not used in any of the state machines of Revision 0.0 of this document.

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. Note that these constructs are not used in any of the state machines of Revision 0.0 of this document.

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.

2.1 Pdelay responder port

The state machine and C code for a Pdelay responder P2P TC port are shown in Figure 2-1.

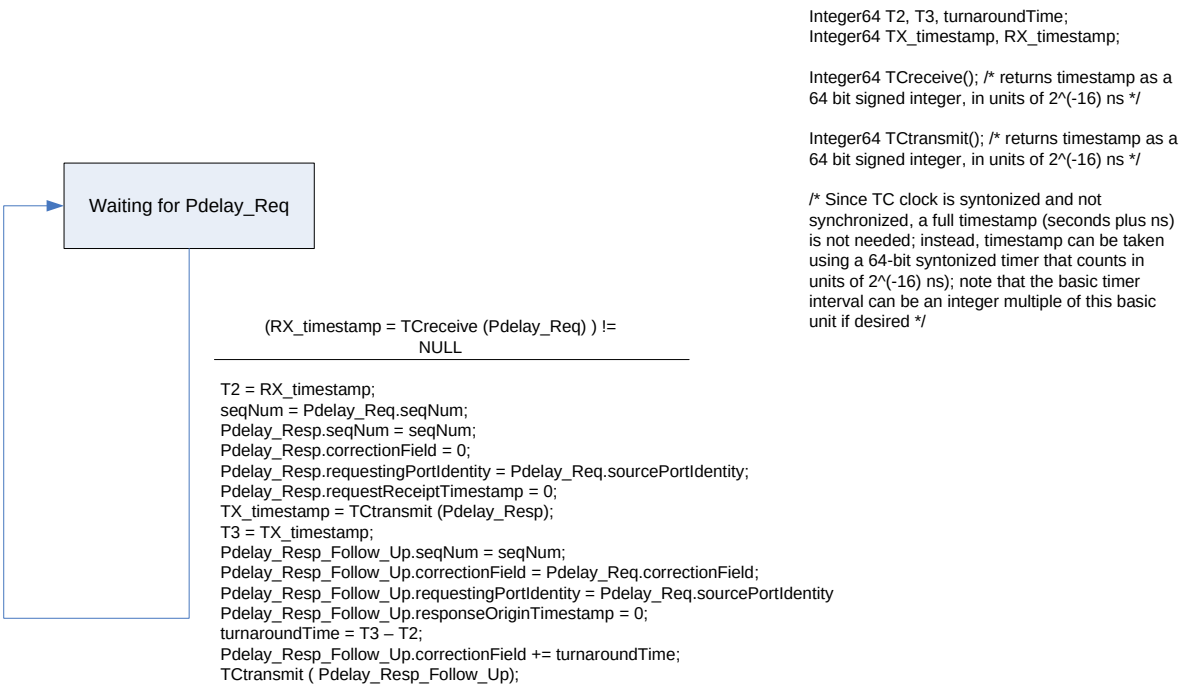

Figure 2-1. State machine, and corresponding C code, for Pdelay responder P2P TC port

2.2 Pdelay requester port

The state machine and C code for a Pdelay requestor P2P TC port are shown in Figure 2-2.

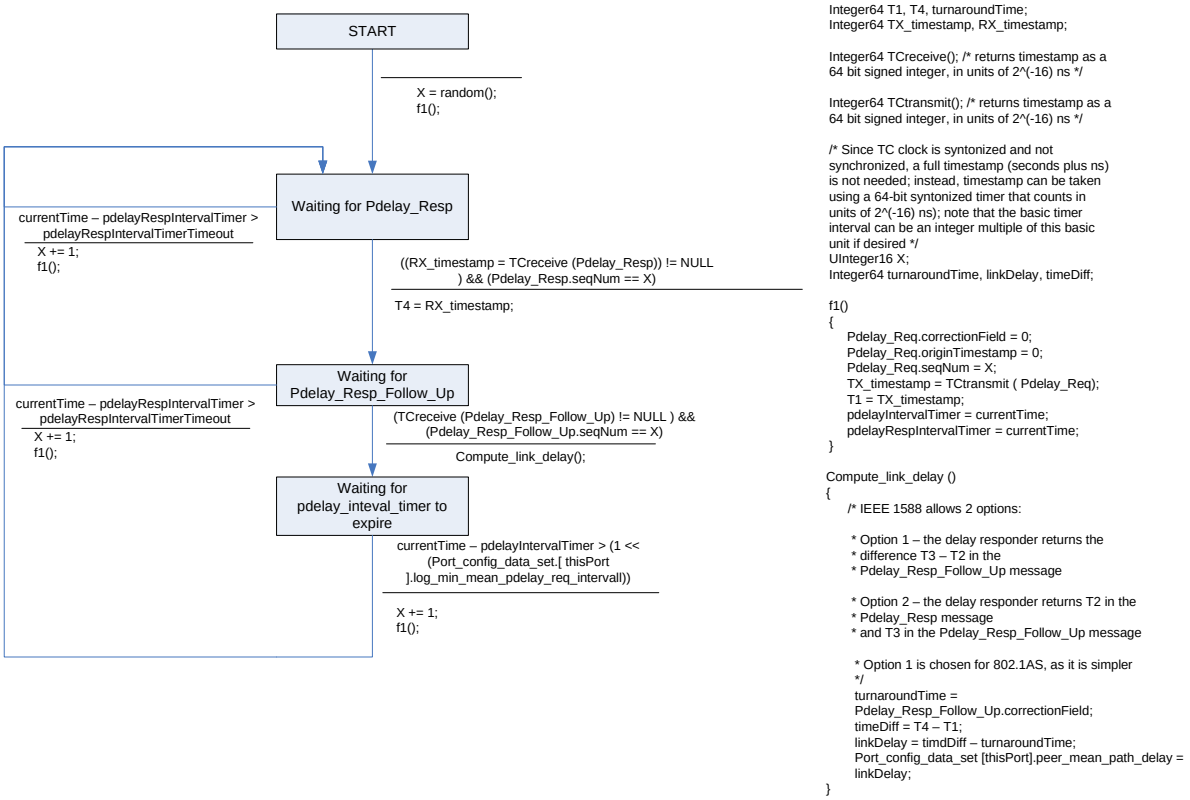

Figure 2-2. State machine, and corresponding C code, for Pdelay requestor P2P TC port

2.3 Peer-to-peer transparent clock node

The state machine and C code for a peer-to-peer transparent clock node are shown in Figure 2-3.

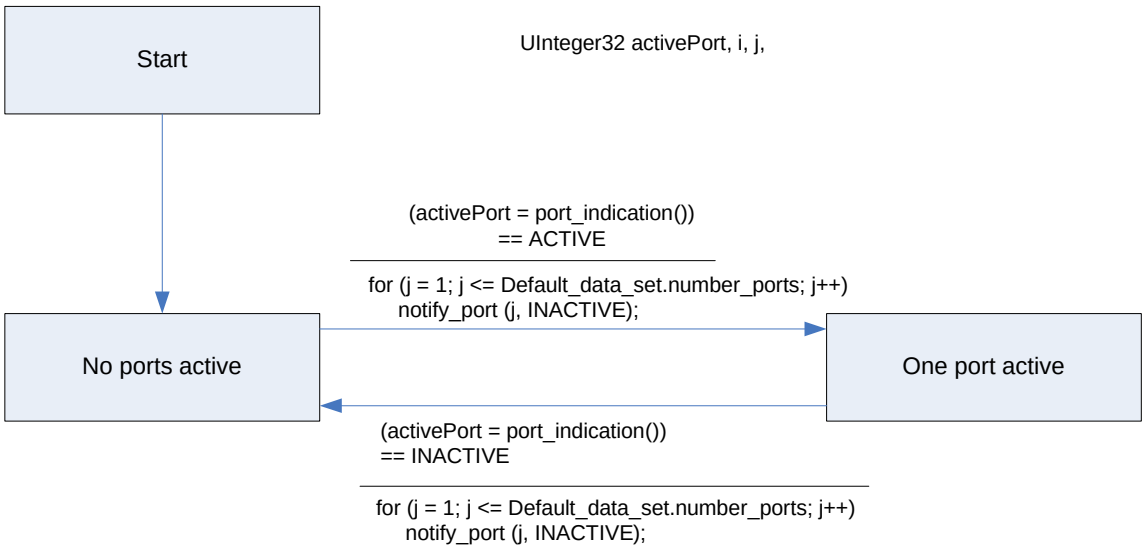

Figure 2-3. State machine, and corresponding C code, for a peer-to-peer transparent clock node

2.4 Receipt of Sync and Follow_Up by a peer-to-peer transparent clock port

The state machine and C code for receipt of Sync and Follow_Up by a peer-to-peer transparent clock port are shown in Figure 2-4.

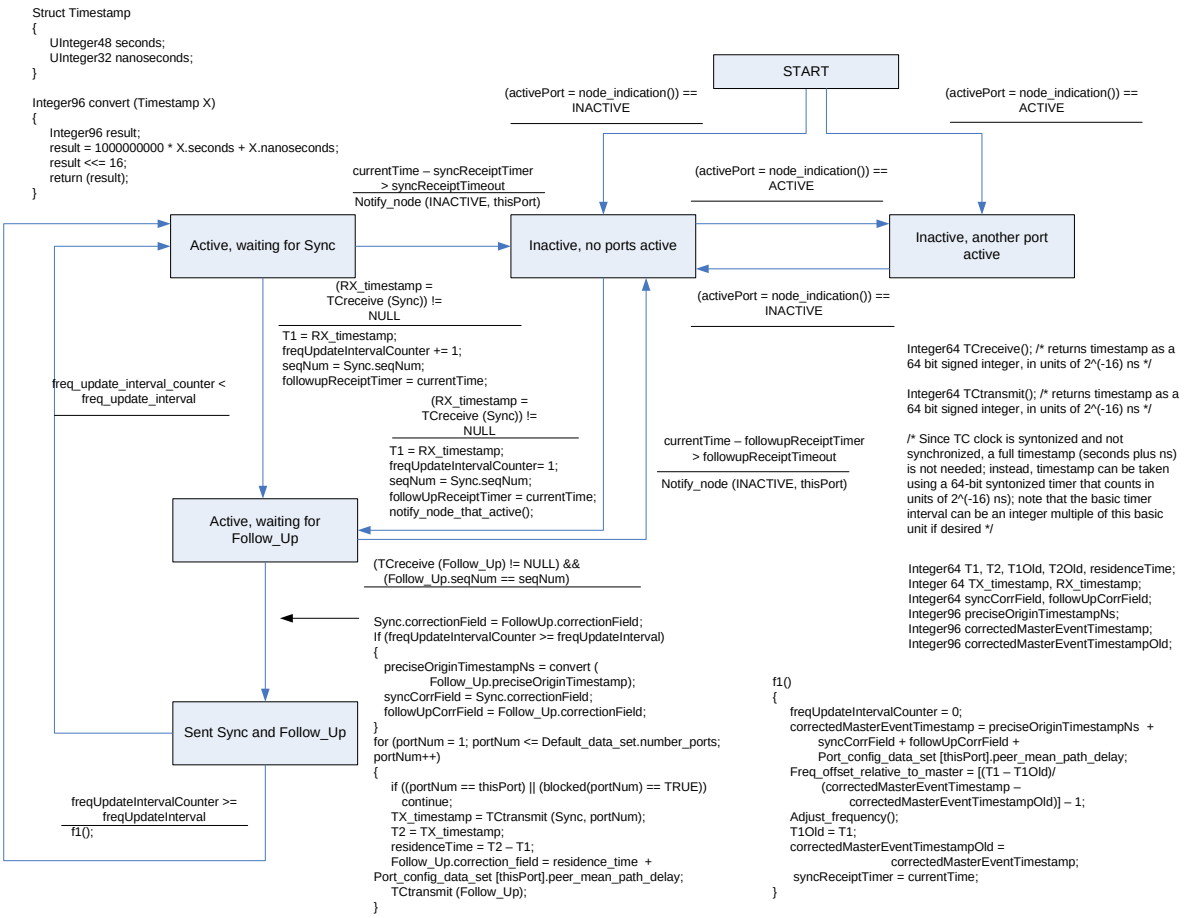

Figure 2-4. State machine, and corresponding C code, for receipt of Sync and Follow_Up by a peer-to-peer transparent clock port

2.5 Receipt of Sync and Follow_Up by an ordinary clock

The state machine and C code for receipt of Sync and Follow_Up by an ordinary clock are shown in Figure 2-5.

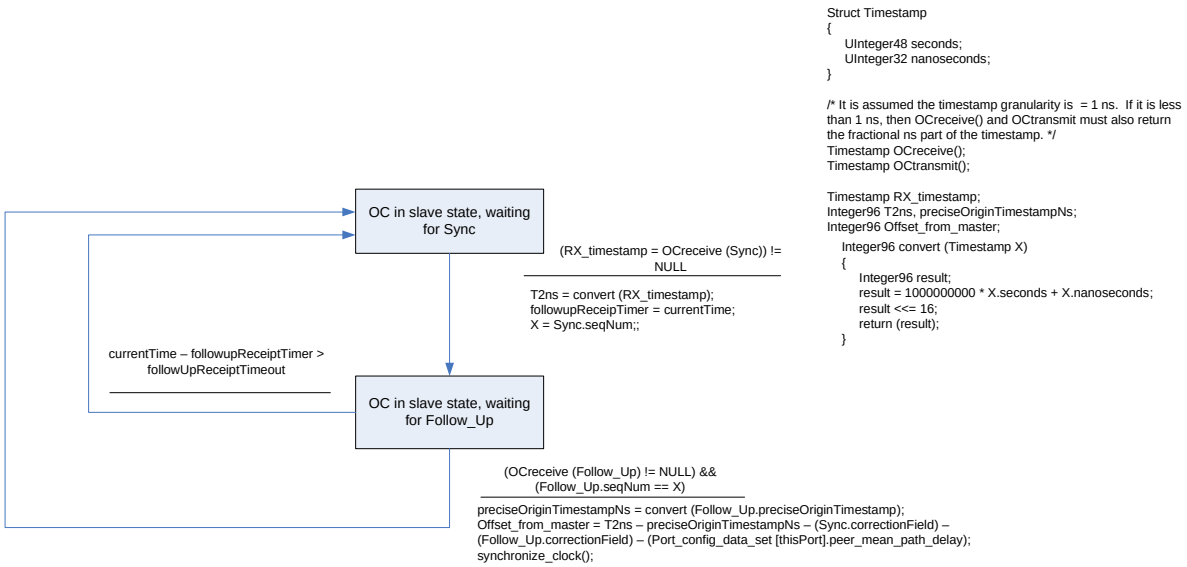

Figure 2-5. State machine, and corresponding C code, for receipt of Sync and Follow_Up by an ordinary clock

2.6 Sending of Sync and Follow_Up by an ordinary clock

The state machine and C code for sending of Sync and Follow_Up by an ordinary clock are shown in Figure 2-6.

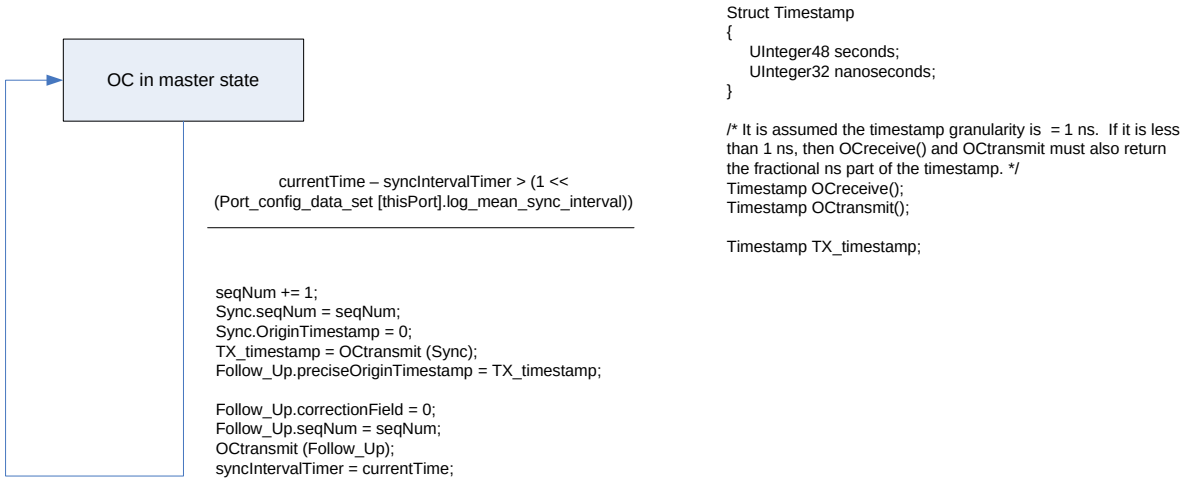

Figure 2-6. State machine, and corresponding C code, for sending of Sync and Follow_Up by an ordinary clock

2.7 Simplified state machine for receiving and sending of Announce by an ordinary clock

A simplified state machine for receiving and sending of Announce by an ordinary clock is shown in Figure 2-7.

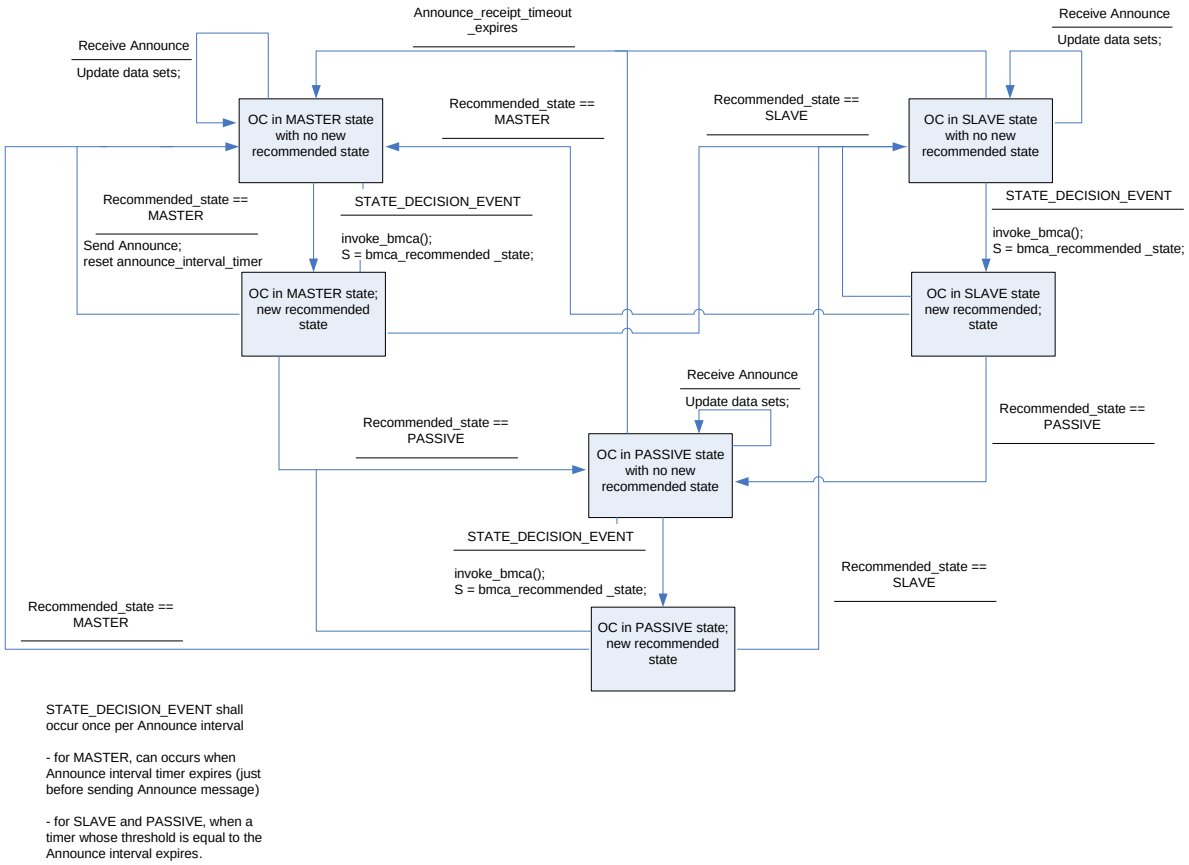

Figure 2-7. Simplified state machine for receiving and sending of Announce by an ordinary clock

3. Message formats

The subsections below contain the details of those 1588 messages that are used in IEEE 802.1AS. The material here is copied from clauses 5 and 13 of [1], with modifications for recent agreements in the P1588 committee.

In the tables in the subsections below, the ‘octets’ column indicates the size of the field in octets. The ‘offset’ column indicates the offset of the first octet of the field from the start of the PTP defined fields of the message.

3.1 Data types

3.1.1 Primitive data types specifications

All non-primitive data types are derived from these primitive types. Signed integers are represented in two’s complement form.

Table 1: Primitive PTP data types

Data type	Definition
Boolean	TRUE or FALSE.
Enumeration4	4-bit enumerated value
Enumeration8	8-bit enumerated value
Enumeration16	16-bit enumerated value
UInteger4	4-bit unsigned integer
Integer8	8-bit signed integer
UInteger8	8-bit unsigned integer
Integer16	16-bit signed integer
UInteger16	16-bit unsigned integer
Integer32	32-bit signed integer
UInteger32	32-bit unsigned integer
UInteger48	48-bit unsigned integer
Integer64	64-bit signed integer
Nibble	4-bit field not interpreted as a number
Octet	8-bit field not interpreted as a number

3.1.2 Derived data type specifications

3.1.2.1 TimeInterval

The TimeInterval type represents time intervals.

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

The scaledNanoseconds member is the time interval expressed in units of nanoseconds and multiplied by 2^{+16}. Positive or negative time intervals outside the maximum range of this data type shall be encoded as the largest positive and negative values of the data type respectively.

For example: 2.5 ns is expressed as:
(hex) 0x0000 0000 0002 8000

3.1.2.2 Timestamp

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

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

The seconds member is the integer portion of the timestamp in units of seconds.
The nanoseconds member is the fractional portion of the timestamp in units of nanoseconds.
The nanoseconds member is always less than 10^9.

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

3.1.2.3 ClockIdentity

The ClockIdentity type identifies a PTP clock.

typedef Octet[8] ClockIdentity;

3.1.2.4 PortIdentity

The PortIdentity type identifies a PTP port.

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

3.1.2.5 ClockQuality

The ClockQuality represents the quality of a clock.

```
struct ClockQuality
{
        UInteger8 clockClass;
        Enumeration8 clockAccuracy;
        Integer16 scaledLogVariance;
}
```

3.2 General message format requirements

All messages shall have a header and a body. Reserved fields shall be transmitted with the all bits of the field 0 and ignored by the receiver. The data type of the field shall be the type indicated in brackets in the title of each clause.

The standard Ethernet header and FCS (18 bytes total) must be added to each of the messages of sections 3.4 through 3.9.

3.3 Header

3.3.1 General header specifications

The common header for all PTP messages shall be as specified in Table 2.

Table 2: Common message header

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
reserved								1	5
flags								2	6
correctionField								8	8
reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
control								1	32
logMeanMessageInterval								1	33

3.3.2 Header field specifications

transportSpecific (Nibble)

The transportSpecific field may be used by a lower layer transport protocol and is defined by the mapping specification of that protocol in the respective annex of [1].

messageType (Enumeration4)

The value shall indicate the type of the message as defined in Table 3.

Table 3: Values of messageType field

Message Type	Message class	Value(hex)
SYNC_MESSAGE	Event	0
DELAY_REQ_MESSAGE	Event	1
PATH_DELAY_REQ_MESSAGE	Event	2
PATH_DELAY_RESP_MESSAGE	Event	3
Reserved		4-7
FOLLOWUP_MESSAGE	General	8
DELAY_RESP_MESSAGE	General	9
PATH_DELAY_FOLLOWUP_MESSAGE	General	A
ANNOUNCE_MESSAGE	General	B
SIGNALING_MESSAGE	General	C
MANAGEMENT_MESSAGE	General	D
Reserved		E-F

The Most Significant Bit of the message ID field divides this field in half between Event and General messages.

NOTE- The reserved nibble immediately following messageType is reserved for future expansion of the messageType field.

NOTE- The message types DELAY_REQ_MESSAGE, DELAY_RESP_MESSAGE, SIGNALING_MESSAGE, and MANAGEMENT_MESSAGE are not used in IEEE 802.1AS.

versionPTP (UInteger4)

The value of the versionPTP field shall be the value of the version_number member of the port data set. For the current version of IEEE 802.1AS, this value is 2.

messageLength (UInteger16)

The value of the messageLength shall be the total number of octets that form the PTP message. The counted octets start with the first octet of the header and include and terminate with the last octet of the body of the message as defined in the subsections below.

domainNumber (UInteger8)

For ordinary of boundary clocks, the value shall be the value of the domain_number member of the default data set of the originating ordinary or boundary clock. For IEEE 802.1AS, this value is 0.

Flags (Octet[2])

The value of the bits of the array shall be as defined in Table 4. For message types where the bit is not defined in Table 4, the values shall be FALSE.

The flags PTP_profile_Specific1 and PTP_profile_Specific2 are available for use by IEEE 802.1AS if needed.

Table 4: Values of flags field

Octet	Bit	Message Type	Name	Description
0	0	Announce	LI_61	Value of leap_61 of global time properties data set
0	1	Announce	LI_59	Value of leap_59 of global time properties data set
0	2	Announce, Sync, Follow_Up	ALTERNATE_MASTER (not used by IEEE 802.1AS)	FALSE if the port of the originator is in the MASTER state. Otherwise TRUE.
0	3	Sync, Pdelay_Resp	TWO_STEP	For an one-step clock, the value of TWO_STEP shall be FALSE. For a two-step clock, the value of TWO_STEP shall be TRUE.
0	4	Announce	TIME_TRACEABLE	The value of time_traceable of the global time properties data set.
0	5	Announce	PTP_TIMESCALE	The value of ptp_timescale of the global properties data set.
0	6	All	UNICAST (not used by IEEE 802.1AS)	TRUE, if the transport layer protocol address to which this message was sent is a unicast address. FALSE, if the transport layer protocol address to which this message was sent is a multicast address.
0	7	Announce	FREQUENCY_TRACEABLE	The value of frequency_traceable of the global time properties data set.
1	0	ALL	Reserved	NOTE-This bit is reserved for the experimental security mechanism of [1]
1	6	All	PTP_profile_Specific1	As defined by an alternate PTP profile
1	7	All	PTP_profile_Specific2	As defined by and alternate PTP profile

All unused flags are reserved.

correctionField (Integer64)

The correctionField is the value of the correction measured in nanoseconds and multiplied by 2^{16}. E.g. 2.5 ns is represented as 0x0000000000028000

A value of one in all bits, except the most significant, of the field, shall indicate that the correction is too big to be represented.

The value of the correctionField depends on the message type as described in Table 5.

Table 5: Correction field semantics

Message Type	correctionField description
SYNC_MESSAGE	Corrections for fractional nanoseconds, residence time and path delay in peer-to-peer transparent clocks, and asymmetry corrections
PATH_DELAY_REQ_MESSAGE	Corrections for fractional nanoseconds and asymmetry corrections
PATH_DELAY_RESP_MESSAGE	Corrections for fractional nanoseconds and asymmetry corrections
FOLLOWUP_MESSAGE	Corrections for fractional nanoseconds, residence time and path delay in peer-to-peer transparent clocks, and asymmetry corrections
PATH_DELAY_FOLLOWUP_MESSAGE	Corrections for fractional nanoseconds and asymmetry corrections
ANNOUNCE_MESSAGE	Zero

sourcePortIdentity (PortIdentity)

The value of the sourcePortIdentity field shall be the value of the port_identity member of the port data set of the port that originated this message.

sequenceId (UInteger16)

For Sync, Pdelay_Req, and Announce messages, the value of the sequenceId field shall be assigned by the originator of the message in conformance to subclause 7.3.7 of [1], i.e., a separate pool of sequence numbers shall be used for each of these message types, and the sequenceId of a message shall be one greater than the sequenceId of the previous message of the same type, subject to the constraints of rollover. The value of the sequenceId field of a Follow_Up message shall be equal to the value of the sequenceId field of the corresponding Sync message. The value of the sequenceId field of a Pdelay_Resp and of a Pdelay_Resp_Follow_Up message shall be equal to the value of the sequenceId field of the corresponding Pdelay_Req message.

control (UInteger8)

The value of control field depends on the message type defined in the messageType field, sub-clause 0, and shall have the value specified in Table 6. The use of this field by the receiver is deprecated. NOTE—This field is provided for compatibility with hardware designed based on version 1 of this standard. This field is not used by IEEE 802.1AS.

Table 6: control field enumeration

Message Type	control field Value
Sync	0x0
Delay_Req	0x1
Follow_Up	0x2
Delay_Resp	0x3
Management	0x4
All others	0x5
reserved	0x6-0xFF

logMeanMessageInterval (Integer8)

The value of the logMeanMessageInterval field is determined by the type of the message and shall be as defined in Table 7.

Table 7: Values of logMeanMessageInterval field

Message Type	Value of logMeanMessageInterval
Announce	The value of the log_mean_announce_interval member of the default data set
Sync, Follow_Up	The value of the log_mean_sync_interval member of the port data set
Pdelay_Req	0x7F
Pdelay_Resp	0x7F
Pdelay_Resp_Follow_Up	0x7F

3.4 Announce message

3.4.1 General Announce message specifications

The fields of Announce messages shall be as specified in Table 8.

Table 8: Announce message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (section 3.3)								34	0
originTimestamp								10	34
currentUTCOffset								2	44
reserved								1	46
timeSource								1	47
stepsRemoved								2	48
grandmasterPortIdentity								10	50
grandmasterClockQuality								4	60
grandmasterPriority1								1	64
grandmasterPriority2								1	65
parentPortIdentity								10	66

3.4.2 Announce message field specifications

originTimestamp (Timestamp)

The value shall be an estimate of the local time of the originating clock when the Announce message was transmitted.

currentUTCOffset (Integer16)

The value shall be the value of the current_utc_offset member of the global properties data set.

timeSource (Enumeration8)

The value shall be the value of the time_source member of the global properties data set.

stepsRemoved (UInteger16)

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

grandmasterPortIdentity (PortIdentity)

The value of the grandmasterPortIdentity field shall be the value of the grandmaster_port_identity member of the parent data set.

grandmasterClockQuality (ClockQuality)

The value of the grandmasterClockQuality field shall be the value of the grandmaster_clock_quality member of the parent data set.

grandmasterPriority1 (UInteger8)

The value of the grandmasterPriority1 field shall be the value of the grandmaster_priority1 member of the parent data set.

grandmasterPriority2 (UInteger8)

The value of the grandmasterPriority2 field shall be the value of the grandmaster_priority2 member of the parent data set.

parentPortIdentity (PortIdentity)

The value of the parentPortIdentity field shall be the value of the parent_port_identity member of the parent data set.

3.5 Sync message

3.5.1 General Sync message specifications

The fields of Sync and Delay_Req messages shall be as specified in Table 9.

Table 9: Sync and Delay_Req message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (section 3.3)								34	0
originTimestamp								10	34

originTimestamp (Timestamp)

The value of the originTimestamp field shall be an estimate of the correct time. For IEEE 802.1AS, zero is an acceptable estimate of the correct time.

3.6 Follow_Up message

3.6.1 General Follow_Up message specifications

The fields of Follow_Up message shall be as specified in Table 10.

Table 10: Follow_Up message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (section 3.3)								34	0
preciseOriginTimestamp								10	34

preciseOriginTimestamp (Timestamp)

The value of the preciseOriginTimestamp shall be the value of the timestamp for transmission of the associated Sync message, truncated to the next lowest nanosecond (i.e., excluding any fractional nanoseconds)

3.7 Pdelay_Req message

3.7.1 General Pdelay_Req message specifications

The fields of the Pdelay_Req message shall be as specified in Table 11.

Table 11: Pdelay_Req message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (section 3.3)								34	0
originTimestamp								10	34
Reserved								10	44

originTimestamp (Timestamp)

The value of the originTimestamp shall be 0.

3.8 Pdelay_Resp message

3.8.1 General Pdelay_Resp message specifications

The fields of the Pdelay_Resp message shall be as specified in Table 12.

Table 12: Pdelay_Resp message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (section 3.3)								34	0
requestReceiptTimestamp								10	34
requestingPortIdentity								10	44

requestReceiptTimestamp (Timestamp)

The value of the requestReceiptTimestamp shall be 0.

requestingPortIdentity (PortIdentity)

The value of the requestingPortIdentity shall be the value of the sourcePortIdentity of the corresponding Pdelay_Req message.

3.9 Pdelay_Resp_Follow_Up message

3.9.1 General Pdelay_Resp_Follow_Up message specifications

The fields of the Pdelay_Resp_Follow_Up message shall be as specified in Table 13.

Table 13: Pdelay_Resp_Follow_Up message fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (section 3.3)								34	0
responseOriginTimestamp								10	34
requestingPortIdentity								10	44

responseOriginTimestamp (Timestamp)

The value of the responseOriginTimestamp shall be 0.

requestingPortIdentity (PortIdentity)

The value of the requestingPortIdentity shall be the value of the sourcePortIdentity of the corresponding Pdelay_Req message.

5. References

[1] IEEE 1588™/D1-C 4 March 2007 Draft Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE, March 4, 2007.

[2] IEEE P802.1AS/D0.6, IEEE Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks, IEEE, January 22, 2007.