(Last updated: June 7, 2004)

Questions that require an interpretation of the meaning of IEEE-1588 or point out possible errors in the standard are submitted to a subcommittee of the IEEE-1588 working group for formal interpretation following IEEE guidelines. This file is a copy of these interpretations. By IEEE policy these do NOT represent a revision of the standard.

Submitted questions are in black. Responses are in red.

QUESTION 1: (submitted 15 Jan 2003)

On the machines I am targeting, the native Integer32 does not have unique representations for positive and negative zero. This implies that the struct TimeRepresentation for a positive whole number of seconds (nanoseconds == 0) is identical to the TimeRepresentation for the negative same whole number of seconds (nanoseconds == -0). This appears to be a flaw in the design. Am I missing something here?

RESPONSE:
Reference clause 5.2.2.
You are correct. There is a flaw in the design that results in an ambiguous interpretation of the struct TimeRepresentation for all negative time intervals or timestamps, the latter being unlikely in practice since it implies the epoch is in the future. It is also impossible to express negative time intervals or timestamps for which the nanoseconds portion is zero. Please note that by clause 5 implementers may use any internal representation provided it does not change the semantics of any network visible operation of the protocol. Representations specified in the standard must be used in any on-the-wire message defined in the standard.

Given the zero flaw, there is no possible interpretation that can resolve the issue. This will need to be corrected in the next version of the standard. The recommendation is to implement as though clause 5.2.2 read:

```
IDL: struct TimeRepresentation
{
        UInteger32 seconds;
        UInteger32 nanoseconds;
};
```

where the nanoseconds member is defined such that the most significant bit represents the sign bit, 1 indicating a negative number, and the least significant 31 bits represent the nanoseconds portion of the time being represented. TimeRepresentation thus defines a sign magnitude representation for time stamps and time intervals.

The range of the absolute value of the least significant 31 bits of the nanoseconds portion of the representation shall be restricted to:

$0 <= |$ least significant 31 bits of nanoseconds$| < 10^9$

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

For example:

+2.0 seconds is represented by seconds = 0X00000002 and nanoseconds = 0X00000000
-2.0 seconds is represented by seconds = 0X00000002 and nanoseconds = 0X80000000
+2.000000001 seconds is represented by seconds = 0X00000002 and nanoseconds = 0X00000001
-2.000000001 seconds is represented by seconds = 0X00000002 and nanoseconds = 0X80000001

Note: This change will require the corresponding changes of the type of the nanoseconds member from Integer32 to UInteger32 in numerous places in other clauses of the standard.

QUESTION 2:

In the PTP spec Annex D.3.1, it defines the UDP port numbers for messages as follows:

Event port = 319 (Sync, Delay_Req)
General port = 320 (Follow-up, Delay_Resp or Management message)

Are the ports for the listeners to bind (udp recvfrom port)? If that's the case, does it mean that the slave has to open two sockets to listen, one for event (e.g. Sync) messages and the other for other (e.g. Follow-up) messages? The spec only suggests having separate sockets for multicast and unicast messages (e.g. Sync).

RESPONSE:
Reference clauses 6.2.2.2, Annex D.3 and D.3.1.
Clause 6.2.2.2 specifies that all non-management messages shall be communicated via multicast communication. Management message may also use point-to-point communication. Thus a device must listen for all types of messages, management and non-management, as multicast communications. The devices may **in addition** accept management messages as a point-to-point communication. If the device knows that a target device will accept a point-to-point management message it may send it as point-to-point otherwise it must use multicast. Note that management messages are one of the

types of messages communicated over GeneralPorts (the other types being Follow_Up and Delay_Resp messages).

Annex D.3 and 3.1 specifies how this is to work in normal Ethernet implementations. All messages shall be UDP (clause D.3). Clause D.3.1 specifies that each domain shall use one of the defined Ethernet multicast addresses and that EventPorts shall use port number 319 and GeneralPorts port number 320.

Exactly how the above is implemented may vary with the operating system. In most operating systems this will require two sockets, one for the EventPort and one for the GeneralPort.

Most implementations will not make use of the additional option of unicast messages for management messages as allowed by clause 6.2.2.2. If they do an additional socket will need to be provided for this purpose.

QUESTION 3: (submitted March 19, 2003)

Page 54: 7.5.5 Receipt of a Delay_Req message from another clock:
The Text states that the message shall be disregarded in the PTP_INITIALIZING, PTP_DISABLED and the PTP_FAULTY state. Figure 13 depicts only the PTP_DISABLED and the PTP_FAULTY state. The PTP_INITIALIZING state is missed in figure 13.

RESPONSE:
Figure 12 is incorrect and the text is correct. The PTP_Faulty, Disabled, and Initializing states should be described in Figure 12 in the same way as in Figures 10,11 and 13 This will have to be corrected in the next revision. In the meantime implementers are urged to consider the suggested interpretation.

QUESTION 4: (submitted March 19, 2003)

Page 72: stepsRemoved must be localStepsRemoved (Table 16).

RESPONSE:
Correct. The entry ‘stepsRemoved’ in the second column of Table 16 should refer to the message field ‘localStepsRemoved’ of the Sync message. This will have to be corrected in the next revision. In the meantime implementers are urged to consider the suggested interpretation.

QUESTION 5: (submitted March 19, 2003)

Page 72:steps_removed is from current data set and not from the default data set (Table 16).

RESPONSE:

Several of the items including steps_removed, in the last column of Table 16 are from datasets other than the default data set. The source of the confusion is a misstatement in the first sentence of the second paragraph of 7.6.3 which should read "D_0 represents the characteristics of clock C_0 as specified in the data sets of C_0" In addition the first sentence of 7.6.2 should be modified to read "The BMC algorithm on a typical clock C_0 characterized by a default data set D_0 and other data sets and with N ports shall be as follows:" This will have to be corrected in the next revision. In the meantime implementers are urged to consider the suggested interpretation.

QUESTION 6: (submitted March 19, 2003)

Invalid grandmaster clock
Consider a boundary clock with 3 PTP ports, which is connected to the GM clock. Assume the GM clock changes its sync interval from the default value of 1 to 2. The BC receives the sync message from the current master (GM) and compares the sync interval of the received sync message with the sync interval of the default data set (7.5.18 Detection of an internal fault). The BC detects an error on the port connected to the GM and posts the FAULT_DETECTED event, which causes a state transition to the PTP_FAULTY state.

The problem is that the parent data set has the values of the GM clock, which will be invalid (since Faulty port state). The BC transmits sync messages with the values of the parent data set and will be therefore pretend to be the GM or more precisely is synchronized to the GM. The result is that the ptp domain is synchronized to the BC which pretends to be the GM and that the BMC cannot reconfigure the PTP domain.

Aside from this the grandmaster sequence id of the parent data set is incremented since no port of the BC is in the PTP_SLAVE state – 2 x PTP_MASTER, 1 x PTP_FAULTY – (The definition of a Grandmaster Clock in 7.4.4.20 and 7.5.9 "A clock is the GM clock of a subdomain if it has no ports in the PTP_SLAVE state.") which results that the BC transmits sync messages which pretend to be a "living" GM.

Conclusion:
The parent data set of a BC must be updated (preferable with the default data set of the PTP clock) if we loose the synchronization to the GM. Otherwise we distribute outdated sync messages. This problem occurs if the PTP_FAULTY or PTP_DISABLED state is entered.

Remedy:
If we enter the PTP_FAULTY or the PTP_DISABLED state we should update the parent data set with the update method M1 and M2 (Table 18) as it is described in Table 15.

The consequences are:

For a BC, which is also, GM that the parent data set is overwritten with the same information.

And for a BC which is not the GM, that the old GM is cleared.

The given scenario with the changed sync interval is only to show that there exist various scenarios where a BC juggles to be the GM.

RESPONSE:
The next 3 paragraphs address the points raised so far:
Changing of Sync interval is presumed to be a rare event that is managed as part of an administrative procedure, whether automated by some external tool or invoked manually, and not part of normal operation. In either case the expectation is that the appropriate management message will be used. Clause 6.2.5.5 explicitly states, "The behavior of PTP subdomains where clocks do not all have the same value of sync interval is outside the scope of this standard".

Also note that the changing of sync_interval by means of one of the management messages defined in clauses 7.12.10 or 7.12.23 also require that the node be reinitialized before the update of sync_interval takes effect. A well-designed tool for changing sync_interval will therefore issue to all nodes either the message of 7.12.10 or 7.12.23 followed by the multicast message of 7.12.4 to all nodes causing them to re-initialize.

If a node does detect such a situation this is certainly a situation within the meaning of FAULT_DETECTED of clause 7.5.18 since it prevents proper continuing operation of the protocol. This would result in the node transitioning to the PTP_FAULTY state. Note that by 7.3.1 table 7 and 7.3.2 table 8 a node in the PTP_FAULTY state is prohibited from taking action that would affect the operation of the protocol on any other port on the communication path. This includes the sending of any PTP messages on the faulty port. In the case of a boundary clock if the effects of the detected problem cannot be confined to the detecting port then the entire boundary clock must be in the PTP_FAULTY state. It is also true that the only exit from the PTP_FAULTY state is ultimately via the PTP_INITIALIZATION state, which will result in the initialization of all data structures. If the port declared faulty was previously in the PTP_SLAVE state then the information in the parent data set will reflect the characteristics of the node communicating to the port in question. Although not specifically stated in the standard the change of a port's state from PTP_SLAVE to either PTP_FAULTY or PTP_DISABLED would be a good time to update the parent data set as defined in Table 15 for nodes in the states listed in the last row of the table to keep the problem confined to the faulty or disabled port. Although implied by the standard in 7.3.1 table 7 and 7.3.2 table 8, this should be clarified in the next revision.

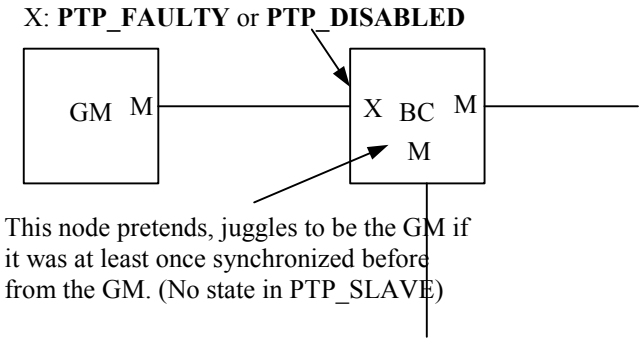
Question 6 continued:
The point here is either the definition of the GM 7.4.4.20/7.5.9 or the clearing of the outdated parent data set. This is due to no port of the BC is in the PTP_SLAVE state. If the clock was synchronized from a GM that does not longer synchronize the BC due to one of the port state either ***PTP_DISABLED*** or ***PTPT_FAULTY***.

Sending a sync message the respective port of a BC verifies whether there is a port in the PTP_SLAVE state. In this scenario there is no port in ***PTP_SLAVE*** state because the port that was connected to the GM is either in the ***PTP_FAULTY or PTP_DISABLED*** state. (The definition of a Grandmaster Clock in 7.4.4.20 and 7.5.9 ”A clock is the GM clock of a subdomain if it has no ports in the PTP_SLAVE state."). This results that the BC transmits sync messages, which ***pretend –mock –*** to ***be a ”living” GM***.

The update of the parent data set is not optimal considering cascaded BCs.

Another solution, which I think will be the best, is to ***redefine*** the ***Grandmaster Definition*** of 7.4.4.20 and 7.5.9.

A Clock is a GM clock if all ports are either in the PTP_MASTER, PTP_PASSIVE or PTP_PRE_MASTER state.



RESPONSE CONTINUED:
This last picture and its comment will not occur if the recommendation above to interpret a transition into either the PTP_FAULTY or PTP_DISABLED states results in a reinitialization of the parent data set as noted. The proposed change of definition of Grandmaster will not solve the problem. The key to the operation of the protocol state machine is the contents of the data sets, in particular the port configuration data set (where the port state is maintained), the parent data set and the default data set. Of these the port and parent data sets are the ones subject to update based on the protocol state machine (including transitions to faulty and disabled as noted above) and which are the key to determining the correct content of any Sync messages sent. The notion of Grandmaster is something DERIVED from this information and not the root cause. It is the data sets that drive all decision, not the fact that a particular clock is a grandmaster. The statement at several places in the standard that the grandmaster clock has no ports in the PTP_SLAVE state is correct. For example a boundary clock with all ports disabled, faulty or passive is the grandmaster for all of its ports, however ports in these states do not issue Sync messages and hence such a grandmaster would be in an isolated portion of the subdomain.

QUESTION 7: (submitted March 19, 2003)
Qualify Sync Messages- ‘most recent message’

The problem deals with the qualification of messages for use with the best master algorithm.

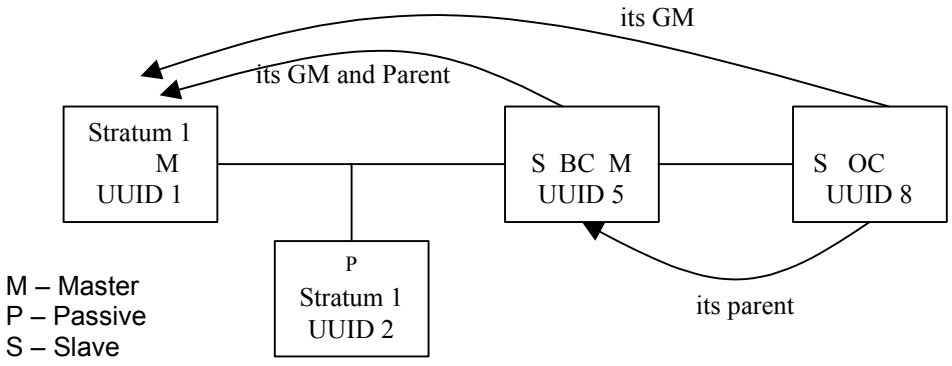
Section 7.5.3 Receipt of a Sync message from another clock (Figure 10)
A Sync message is processed if the received sequenceId from the current sync message is greater than the value of the parent_last_sync_sequence_number of the parent data set. As a result of that the parent data set is updated.

Section 7.6.2 BMC Algorithm:
A Sync message S received by a port 'r' shall be ***qualified*** if S is the most recent message received from a given clock. We will assume that the clock we receive the sync message S is the also from the current master.
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, or if 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.

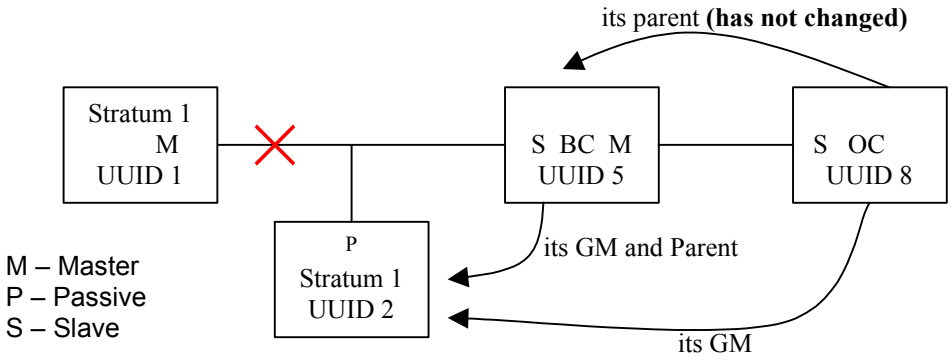
"C code":

```
if(   ( msg_a->grandmaster_sequence_id_ >  msg_b->grandmaster_sequence_id_ ) ||
   ( ( msg_a->grandmaster_sequence_id_ == msg_b->grandmaster_sequence_id_ )
&&
     ( msg_a->sequence_id_ > msg_b->sequence_id_ )                    ))
{
        /* received sync message is more recent than stored -> Sync message qualified */
}
else
{
        /* received old sync message-> disqualify sync message */
}
```

There is a discrepancy in the conditions 7.5.3 and 7.6.2. Consider the following example.
Initial condition:
The Ordinary clock with UUID 1 is the grandmaster clock.



If the GM clock UUID 1 fails or the link, the boundary clock (UUID 5) recognizes this after PTP_SYNC_RECEIPT_TIMEOUT and changes its state to PTP_MASTER. The same does the stratum 1 ordinary clock UUID 2. The BMC computes the new states and sets the state of the stratum 1 UUID 2 clock to PTP_MASTER and the BC UUID clock to PTP_SLAVE. For the ordinary clock UUID 8 the parent has not changed only the GM. The following figure illustrates this scenario.



Assume that the grandmasterSequenceId field of UUID 1 is 100 and the grandmasterSequenceId field of UUID 2 is 10.
The problem is the following, if we follow 7.5.3 receipt of a sync message from another clock, the parent data set is ***updated***. But the received sync message from the BC by the OC UUID 8 is ***not qualified*** due to the grandmasterSequenceId of the last qualified sync message is 100 (namely the grandmasterSequenceId from UUID 1) and the grandmasterSequenceId of the current sync message is 10. After 90 sync messages the sync messages will overcome this problem due to the grandmasterSequenceId.

Following behaviors are possible:
• If we implement 7.5.3 and the ”sync qualification mechanism” independent.
The received sync message updates always the parent data set. But the parent data set is overwritten by the last received sync message which was qualified (The last sync message from UUID 1). That is in our example, we overwrite the parent data set 90 times with the last qualified sync message (UUID 1) which results in an oscillating parent data set and a temporary deadlock depending on the difference between the grandmasterSequenceIds).

- Another possibility is to disregard the received sync message and do not restart the sync receipt timer if the grandmasterSequenceId of the last qualified sync message is greater than the grandmasterSequenceId of the received sync message. The result is that the parent data set is not updated for PTP_SYNC_RECEIPT_TIMEOUT but it causes a state transition, which updates the parent data set with the default data set (Table 7-10). Finally we can say that the deadlock is resolved after PTP_SYNC_RECEIPT_TIMEOUT seconds but we loose the synchronization and a cascade of boundary clocks loose the synchronization for the number of boundary clocks x PTP_SYNC_RECEIPT_TIMEOUT.

Assume 5 BC and a PTP_SYNC_RECEIPT_TIMEOUT of 20 seconds.

An ordinary clock connected to the 5th BC loose the synchronization for about 100 seconds.

Question: The question is why the grandmasterSequenceId field of A must be the same as the grandmasterSequenceId field of B (This disqualifies sync messages if the GM changes or the current GM reboots). ?
Is it not sufficient to compare only the sequenceId fields (as it is done in 7.5.3) because we will disregard the sync message if we receive an old message from the parent. If we compare only the sequenceId of the sync message we trust the parent and a cascade of boundary clocks are updated from the parent. The 1st BC is updated after PTP_SYNC_RECEIPT_TIMEOUT and propagates its new parent data set with the next sync message to all other BC.

Assume 5 BC and a PTP_SYNC_RECEIPT_TIMEOUT of 20 seconds.

An ordinary clock connected to the 5th BC loose the synchronization for about 30 seconds. Which is much better than 100 seconds.

Conclusion:
It would be the best to ***change*** the ***definition*** of the ***more recent message*** to

”The grandmasterSequenceId field of A is greater than the grandmasterSequenceId field of B, or if the sequenceId field of A is greater than the sequenceId field of B.”

and to remove the term ”grandmasterSequenceId field of A is the same as the grandmasterSequenceId field of B”. If we cannot remove this term please state why we cannot remove this term ?

RESPONSE:
There is a problem here but not in quite the way as stated by the questioner. First note that 7.5 requires that all ‘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’ to be atomic. The only ‘qualifying sequence of events’ it the receipt of multiple Sync messages within the same sync_interval. In addition 7.6.2 requires that during the operation of computing E_{rbest} and the resulting state changes, that the portion of Figure 14 starting with ‘compute E_{best}’ be atomic. In particular no updates due to receipt of a Sync message may be processed during this operation.

Next, clause 7.5.3 does NOT mention qualified Sync messages. This is correct. Any Sync message received by a port in the PTP_SLAVE state (at the time the message is processed due to the atomicity requirement above) will result in either an update to the foreign master data sets, rejection because it is a duplicate message, or an update of the parent data set of the receiving clock. Note that there is only a single parent data set even for a boundary clock, Figures 5 and 6. Thus a clock with a port in the PTP_SLAVE state

and receiving a Sync message from the current master clock, as identified in the parent data set, will have the parent data set updated with the values in the received Sync message. Note that the determination of the state of the port is made during the atomic process of processing the Sync message as described in 7.5.3 and not at the time of receipt (unless the atomic process is made to include the time of receipt).

The notion of qualified and most recent is correctly applicable only to clause 7.6.2, which specifies the best master clock algorithm and its effects on port state and clock data sets and which must be atomic with respect to other operations, in particular the operations of 7.5.3. 7.6.2 specifies how to compute the best message received on port r, E_{rbest}, and the best message received by the clock, E_{best}. Only 'qualified' Sync messages are to be used in this computation.

It also states in computing E_{rbest} that the results of the previous computation of E_{rbest} be included UNLESS there has been a more recent message from the same SENDING port or an expiration of the SYNC_RECEIPT_TIMEOUT for the receiving port. The inclusion of the prior E_{rbest} is to prevent thrashing if the computing capacity of the node cannot process all of the messages at hand within the allotted time. This assures that the 'best' so far is always included and the protocol converges. The more recent and timeout exclusions of the prior E_{rbest} is to eliminate information from a now silent or degraded clock from displacing a present and better clock. In computing E_{rbest} and E_{best} only messages are used and NO reference is made to the default or parent data sets. This implies that each port must maintain E_{rbest} from the previous round separate from any resulting update of the data sets defined in this standard.

The purpose of the 'qualification' specification is to eliminate:

- Stale information by using only the most 'recent' from a sending port
- Messages received by the clock due to abnormal network topology or operation
- Messages received while the clock is in inappropriate states
- Messages from foreign masters that have not appeared a certain number of times within a time window (to prevent thrashing).

The problem is with the definition of 'most recent', which is only applied when comparing two messages received on a given receiving port of the receiving clock and sent from the same port on the same sending clock. As noted by the questioner, clause 7.6.2 imposes two 'ORed' conditions on the compared messages. One based on grandmasterSequenceId and the other on grandmasterSequenceIds and sequenceId. As stated this definition may lead to problems with symptoms similar to those noted by the questioner although not for the reasons given.

Since the most 'recent' message sent by a port is defined by the sequenceId assigned by the sending port the last section of 7.6.2 starting with "A Sync message A is more recent than a Sync message B if " should read:

"When comparing two messages received from the same sending port, the message with the greatest value of sequenceId appropriately taking into account the rollover properties of the datatype is the most recent".

This will need to be corrected in the next version of the standard. To avoid degradation experienced under certain circumstances as noted by the questioner, implementers are urged to use the specification suggested above.

QUESTION 8: (submitted March 19, 2003)

Events
The standard does not state when several events must be posted. It would be a great help for the readers to know when these events are posted.

Question: When should the MASTER_CLOCK_SELECTED event be posted ?
We post the MASTER_CLOCK_SELECTED event in the update method S1 (recommended state is PTP_SLAVE) if the current state is PTP_UNCALIBRATED.

RESPONSE:
This refers to an event in Figure 9 of clause 7.3.2 and Table 9 of 7.3.2. Method S1 refers to figure 16 clause 7.6.4. The time of posting of this event, which can only result in a transition between the PTP_UNCALIBRATED and PTP_SLAVE states, is a local implementation issue and has no visible effect on the overall operation of the protocol in a system of clocks. As such it is not specified by the standard. The choice made by the questioner is one possible and reasonable implementation choice.

QUESTION 9: (submitted March 19, 2003)

When should the SYNCHRONIZATION_FAULT event be posted ?
We could post this event if the grandmasterSequenceId does not change over a defined period.

RESPONSE:
This refers to an event in Figure 9 of clause 7.3.2 and Table 9 of 7.3.2. See also the FAULT_DETECTED event with the same references as well as 7.5.18. Both FAULT_DETECTED and SYNCHRONIZATION_FAULT are intended to provide an orderly way for a node to deal with certain classes of problems. The difference is whether a complete re-initialization of the node is required after clearing the fault, the FAULT_DETECTED case, or whether if the node is a slave that a lesser measure, transition to PTP_UNCALIBRATED, until the fault is cleared is appropriate. 7.5.18 provides the only guidance on the nature of the faults, all of which are presumed to be detected by the node implementation presumably without recourse to external resources. Management messages are provided to allow similar efforts to be undertaken by external devices.
An example of a SYNCHRONIZATION_FAULT might be a detected temporary excursion in the local oscillator due to a power dip that has caused an excursion in the

local synchronization mechanism in a slave clock. There is no need to reselect a master, etc. but simply to allow the transient to die out before returning to the slave state.

An example of a FAULT_DETECTED event might be a detected internal events such as overflow in data buffers, thread failure, faulty power supply, etc. that will require reinitialization after clearing.

Internal observation of external events such as failure to receive Follow_Up messages when expected or unchanging sequenceIds from parent or grandmaster indicate potential problems in other nodes. An appropriate response could well be a FAULT_DETECTED event response. However unless the node is specifically designed to try and correct perceived faults in other nodes, which is outside of the scope of the standard, then only the FAULT_DETECTED, DESIGNATED_DISABLED or SYNCHRONIZATION_FAULT events are appropriate.

QUESTION 10: (submitted 3_31_03)

Inconsistent parent data set

Consider the case where the PTP Clock (Ordinary or Boundary) is in the PTP_MASTER state and there is no SYNC message at any port available. Therefore we cannot determine E_{rbest} and $E_{best,}$ which implies also, that we cannot run the state decision algorithm figure 16.

Without management:

This is no problem, since only SYNC message can cause a state transition (new data).

With management:

The problem is, that the parent data set is not updated from the state change decision algorithm, if no SYNC message is available. But if the default data set changes e.g. by the management messages PTP_MM_SET_DESGNATED_PREFERRED_MASTER or PTP_MM_CLEAR_DESGNATED_PREFERRED_MASTER, the parent data set must also be updated. Otherwise the default and parent data set are inconsistent.
For instance we clear the grandmaster preferred flag with the management message (default data set), but this is not updated at the parent data set since we have no SYNC message which runs the state decision algorithm. Therefore we emit SYNC messages where the grandmaster preferred flag is set. This cause problems because Clocks which are better do not sent SYNC messages, since the grandmaster preferred flag is set.

RESPONSE:

The question refers to clauses 7.5.8 and most of 7.6. The statement in the question that '…implies also, that we cannot run the state decision algorithm figure 16' is incorrect.

Clause 7.5.8 states that at least once per sync interval there will be a computation of $E_{best,}$ the best master clock algorithm will be applied, the appropriate data sets will be updated and the required state changes will be made.

The issue raised is in conjunction with figure 16 of 7.6.4. It is true that if there have been no Sync messages received on any port for a sufficiently long time that the port will be in the PTP_MASTER state and that there will be no inputs to the computation of E_{rbest} and E_{best}. During the application of the logic of figure 16 the relevant decision will be the comparison of D_0, the default data set, with either E_{rbest} or E_{best} depending on the stratum of the clock. Since in this case neither E_{rbest} nor E_{best} exists the correct conclusion is that D_0 is better. This will result in the data set update required for either M1 or M2, table 18. The update will make the parent data set consistent with the values of the default data set modified either due to internal operation of the clock or via a management message as posed by the questioner. Note that clause 7.6.2 requires that the execution of the terms of 7.6 be atomic, in particular with respect to updates of the default data set.

While correct, the standard could have been more explicit on the treatment of this case. This should be considered at the next revision.

QUESTION 11: (Submitted February 17, 2004 by two separate parties. The following is a synthesis of the submitted questions on this subject)

- The Ethernet appendix, D.1.2, clearly states that all numeric data types are placed on the wire 'big endian'.
- The specification does not define "numeric data types".
- All "octet" fields go out in low index octet to high index octet order, 5.2.1
- The specification is ambiguous in the assignment of values to the several octets for all IEEE 1588 defined fields defined as octet arrays of length greater than 1. These fields are:
    - subdomain, 8.2.3 and 6.2.5.1
    - Any uuid_field, 6.2.4.1
    - flags, 8.2.10
    - clock_identifier, 6.2.4.5
    - manufacturerIdentity, 8.6.1.10.3

RESPONSE:

The questioners are correct in pointing out an ambiguity in the specification.

The recommendation is that the following changes be made in the standard at next revision and that current implementations follow this recommendation:
5.1 Add a statement: The term ‘numeric data types’ shall be synonymous with the list of primitive PTP datatypes excluding the types Boolean and Octet.

D.1.2 Add a statement: Boolean data types shall be marshaled as a single octet with FALSE having all bits 0 and TRUE having a hex value of 0x01.

The ambiguities in the mapping of octet arrays will be resolved to be consistent with the mapping of the field ‘destination port number’ in the Ethernet header, see D.1.3.1 and clause D.3.1 In this case note that 7.4.6.5 and 7.4.6.6 specify port addresses to be octet arrays with the array length dependent on the communication technology. For Ethernet,

D.3.1, this is an octet array of length 2. D.3.1 defines the Ethernet event and general port numbers as numeric 319 and 320 or hex 0x013F and 0x0140 respectively. These will be mapped onto the wire in the order 0x01, 0x3F and 0x01, 0x40 respectively as defined in Ethernet specifications. To be consistent with 5.2.1 these port numbers when expressed as an octet array would be as follows:
event port number 319 => octet[0]= 0x01, octet[1]= 0x3F
general port number 320 => octet[0] = 0x01, octet[1] = 0x40
Based on this mapping defined by Ethernet, not 1588, specifications:

5.2.1 Add a statement: For octet arrays of length greater than 1, the octet with the lowest numerical index shall be termed the most significant octet or the left-most octet of the array.

6.2.5.1 Add a statement: The leading character of the subdomain name, for example the "_" in "_DFLT" shall be the most significant character as defined in 5.2.1.

6.2.4.1 Add a statement to the section on uuid_field: uuid fields shall be marshaled into their on-the-wire formats based on 5.2.1 with the most significant octet of the array containing the most significant octet of the uuid field as defined in this clause.

8.2.10 Add a statement: The most significant octet of the flags field shall be bits 8-15 and the least significant octet shall be bits 0-7 as defined in this clause.

6.2.4.5 Add a statement to the first paragraph: Thus for the identifier 'ATOM' the 'A' shall be considered the most significant or leading character and the 'M' the least significant character.

8.6.1.10.3 Add a statement after "…filled with the null octet": Thus for the manufacturerIdentity 'ABC' the 'A' shall be considered the left-most character of the field.

QUESTION 12: (submitted February 23, 2004)

The sync_interval value is listed as an Integer8 in every place I can find except the PTP_MM_SET_SYNC_INTERVAL description in the Ethernet appendix, where it is an Integer16. Why the difference?

RESPONSE:
There is an inconsistency. sync_interval is defined as:
-in 7.4.2.12 as Integer8
-in 8.6.1.10.23 and D.1.3.6.13 as an Integer16

For reference, the sync_interval as defined in 7.4.2.12 as the logarithm base 2 of the current sync interval as defined in 6.2.5.5 Thus for the Integer8 datatype the extreme values of sync_interval are positive 255 and negative 256 corresponding to values of sync interval of 2 raised to these powers. This is clearly enough dynamic range to cover all reasonable situations. Furthermore for the values currently allowed per 6.2.5.5 the value

of sync_interval will always be positive, (the smallest value of sync interval is 1 second), and therefore even in the current Integer16 of 8.6.1.10.23 all of the significant bits would be in the least significant byte. As an Integer16 these would map into octet 105 after SOF in D.1.3.6.13. This is also consistent with the other messages containing sync_interval defined in 8.6.1.10.9, 8.6.1.10.10, D1.3.6.4, and D.1.3.6.5.

To avoid any ambiguity resulting from the definitions in 8.6.1.10.23 and D.1.3.6.13 we recommend that the following changes be made in the standard at next revision and that current implementations follow this recommendation:

Change the datatypes for PTP_MM_SET_SYNC_INTERVAL in 8.6.1.10.23 and syncInterval in D.1.3.6.13 to Integer8. In D.1.3.6.13 the table entry for octets 102, 103, 104, and 105 after start of frame will be changed to 00, 00, 00, h_0h_1 respectively.

QUESTION 13: (submitted May 3, 2004)
There is an event in the Protocol Engine State Machine that I can't find a definition for:

- SYNCHRONIZATION_FAULT

This seems like the SYNC_RECEIPT_TIMEOUT_EXPIRES event, but that defined event already causes a transition from PTP_SLAVE to PTP_MASTER. SO...it seems like the SYNCHRONIZATION_FAULT event is really the SYNC_RECEIPT_TIMEOUT_EXPIRES when the clock is "slave only"? Regardless, the event needs a definition.

Looking closer, it also seems like there is some bouncing around between states when the 'current master' goes away. A PTP_SLAVE detects the timeout, goes to PTP_MASTER, runs the STATE_CHANGE_EVENT and goes to PTP_PRE_MASTER, and then either goes to PTP_MASTER again or heads off to PTP_UNCALIBRATED. That first transition to PTP_MASTER has me a bit confused, based on events afterward.

RESPONSE:
The questioner is correct there is no formal definition of SYNCHRONIZATION_FAULT other than what is implied by its appearance as an event in the protocol state machine of 7.3.1 Figure 9 as a transition from the PTP_SLAVE state to the PTP_UNCALIBRATED state.

7.3.1 Table 7 indicates that the PTP_UNCALIBRATED state 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.' The other transitions between these two states cover the case of new master clocks appearing.

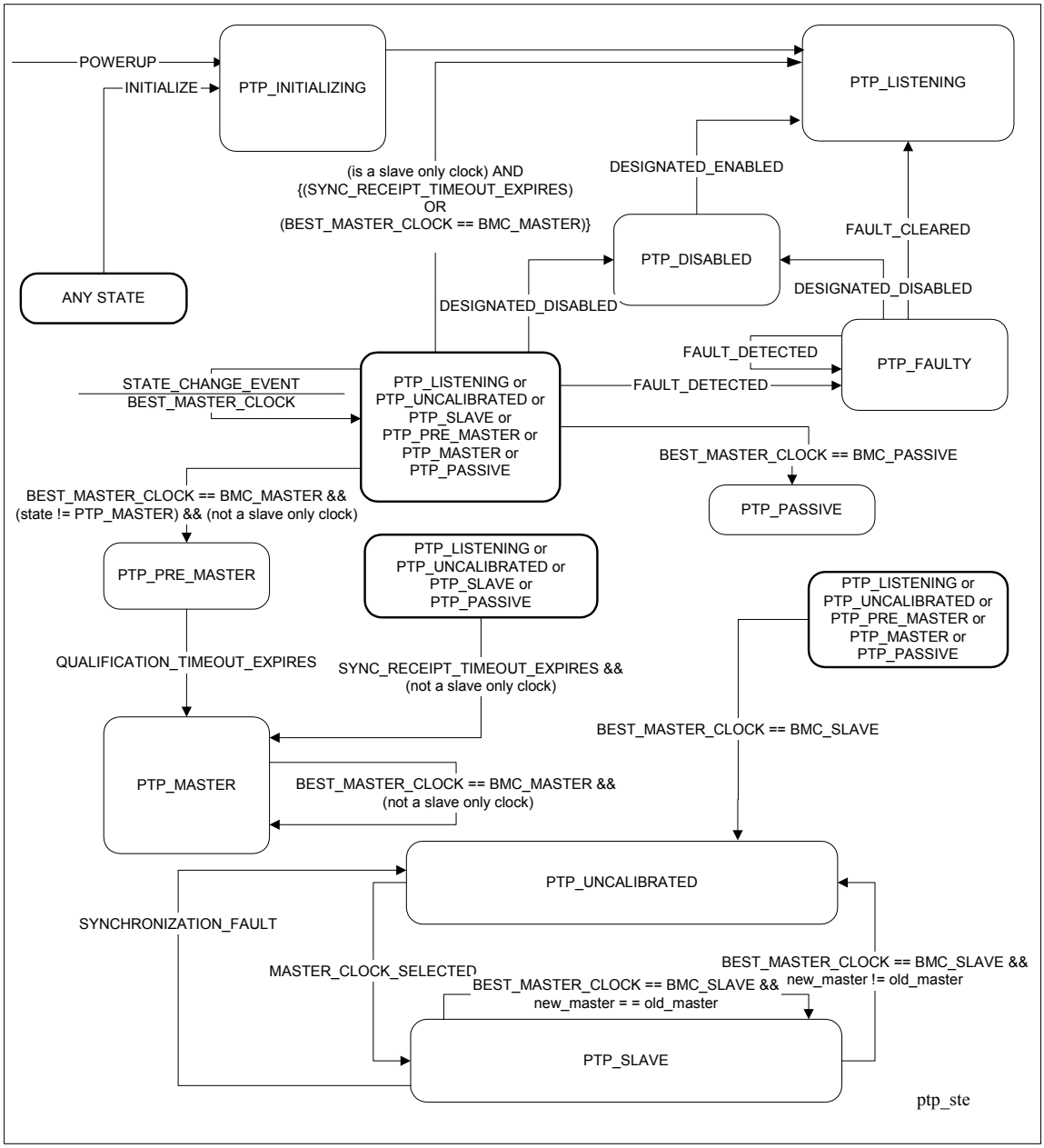
There may be other implementation specific slave issues that require revisiting the PTP_UNCALIBRATED state. A possible example is a transient condition in the slave's processor that prevents timely execution of the servo algorithms that temporarily takes the slave out of specification and requires re-initialization of implementation specific

servo parameters of data. Note that things like this that result in long term faulty behavior are covered by the FAULT_DETECTED event.

There is also an issue when the clock is a ‘slave only’ clock by clause 9.2.2. Such a clock is not permitted to issue messages normally issued by a clock in the PTP_MASTER state. Hence for a ‘slave only’ clock a SYNC_RECEIPT_TIMEOUT_EXPIRES message should take the slave only clock to the PTP_LISTENING state. (Note that it is not illegal, in this version of the standard, to go to the PTP_MASTER state as indicated in Figure 9 but once there it must not act like a master and suppress the messages specified in 9.2.2. The easiest way to implement this behavior is to transition to the PTP_LISTENING state.)

There is also the possibility of thrashing for any clock in the PTP_MASTER state due to an ambiguity in the definition of what constitutes a STATE_CHANGE_EVENT clause 7.5.8 and the state diagram of Figure 9. The STATE_CHANGE_EVENT causes the best master clock algorithm to execute. The common condition for a clock in the PTP_MASTER state when this occurs is for the Recommended State (of Figure 9) to be PTP_MASTER presumably causing the clock to remain in the PTP_MASTER state. Yet Figure 9 requires the clock to go to the PTP_PRE_MASTER state. This is incorrect. Figure 9 should indicate that the clock remains in the PTP_MASTER state.

In a future revision Figure 9 should be modified to the version shown below.



It is recommended that future versions of the standard make these points explicit.

QUESTION 14: (submitted May 3, 2004)
Section 8.6.1.10.3 specifies the manufacturer identity field can be up to 48 ‘characters’ (one ‘character’ per octet), terminated by the null octet. This seems to imply the name can be up to 48 non-null values, followed by a null. This is supported by referring to the null as an octet rather than a character (the spec does not say ‘terminated by the null character’). However, the array of octets containing this value is a length of 48. Thus, either the maximum ID length is 47 ‘characters’ or 48 characters should be allowed without a terminating null octet.

I support a simple specification of 'up to 48 Latin-1 characters, unused octets filled with the null octet'. With a fixed field, we do not need the terminating null when all octets are used.

RESPONSE:
The questioner is correct and poses a reasonable solution. However it is recommended that the clause be changed to read '...up to and including 47 Latin-1…' with the result that whatever string is used it will automatically be null terminated per the rest of the clause. In this way all values for this datum irrespective of length can be treated the same.

QUESTION 15: (submitted May 3, 2004)
The Sync Interval specification is inconsistent when the value is modified. This results in a master possibly sending a sync interval in a Sync message that does not match the current sync interval. The inconsistent specifications:

- 7.12.23: updates the value in the Default Data Set, does not take effect until init, or power up.
- 7.4.2.12: Default Data Set shall specify the current sync interval (may conflict with 7.12.23 when modified)
- 8.3.1.14: The value of this field shall be the value of the sync_interval of the Default Data Set (may or may not be the actual sync interval)

The specification states that all clocks in a subdomain shall have the same sync interval. However, once one clock is modified this is no longer the case. How is a change in the sync interval supposed to work? Does the value in the Sync message get ignored? Compared?

RESPONSE:
The specification is correct. The value for sync interval that a clock uses is the one in the default data set as noted in 7.4.2.12 and 8.3.1.14. The question relates to what happens when this value in the default data set is updated. There are two mechanisms provided for this update: 7.12.23 PTP_MM_SET_SYNC_INTERVAL and 7.12.10 PTP_MM_UPDATE_DEFAULT_DATA_SET. Both these management messages allow for an update to the value of sync interval in the default data set.

HOWEVER as noted in these specifications the receipt of these management messages alone DO NOT CAUSE the value to be immediately updated. The new value (in the management message) takes effect only after a reboot/power cycle or the receipt of a PTP_MM_INITIALIZE_CLOCK message. NOTE that this requirement implies that update values of sync_interval and subdomain_name be retained in non-volatile storage for instantiation after a possible power cycle.

The purpose of this is to allow graceful transition of a system of clocks to a new sync interval. How to do this transition, or a **similar one for the domain name** (see 7.12.5 and 7.12.10) is not specified. This is perhaps something that should be addressed in future revisions of the standard perhaps in a clause devoted to life cycle management of 1588 clock systems. One possible scenario is:

- Send a PTP_MM_SET_SYNC_INTERVAL to each clock in the system
- Send a PTP_MM_INITIALIZE_CLOCK to ALL clocks in the system.

To prevent this misinterpretation in the future it is recommended that a future revision of the standard be more explicit on this point. In particular clauses 7.12.5, 7.12.10, and 7.12.23 the words 'shall not take effect' should be clarified to explicitly state that the update values of either sync_interval or subdomain_name just provided to the clock via the referenced management message shall have no externally visible effect on the clock until one of the designated events occurs. This implies that these values are held in storage and not copied into the operational version of the default data set until the designated event (power cycle or initialize management message).

Future revision committees may wish to consider adding a new management message to view such pending but as yet inoperative update values prior to their instantiation.

One further note: In 7.5.18 it is recommended that an inconsistency between values of sync interval in received event messages and the value in the default data set be regarded as a FAULT_DETECTED event with the consequences specified by the protocol state machine of 7.3.1 Figure 9. The meaning of the term 'inconsistency' is not sufficiently well defined. For example a simple inequality of two sync-intervals may reflect a transient condition occurring during the processing of clause 7.12.4 initialization messages and should not result in a fault. On the other hand persistence of such a condition may indicate abnormal behavior. It is also possible that a low performance slave only node simply cannot process Sync messages at the designated rate but is able to process to needed accuracy a subset of these messages, implying that it operates at a longer sync_interval. Clearly this is a designed in feature and should not result in a fault. For the present it is recommended that only persistent disagreement not specifically provided for and allowed in the detecting clock be declared a fault.

Future revision committees should make explicit exactly what circumstances surrounding this issue are considered a fault.

QUESTION 16: (submitted May 3, 2004)
With reference to clause 7.12.4 it appears that provision should be made to either retain or clear updated values when switching to the specification initialization set.

RESPONSE:
This is a pertinent observation in view of the issues raised with respect to sync_interval and subdomain_name. Note that there are other values that are designated as 'modifiable' in the specification. Some of these may be modified internally others via specific update messages on one of the data sets. It is recommended that all such modifiable values be stored in non-volatile storage so that they can persist over power cycles.

Currently option '0' of 7.12.4 indicates that the specification initialization set be used but is silent on whether any updates in non-volatile storage are to be cleared (or more likely

set to the specification values). It is recommended that option '0' leave any updated values in non-volatile storage unchanged.

Currently option '1' of 7.12.4 indicates that the set stored in nonvolatile storage be used. It is recommended that this interpretation be followed. It is unclear what happens if this option is selected and no updates have been previously processed. It is recommended that implementations always initialize the data stored in nonvolatile storage to the appropriate values from the specification initialization set to prevent undefined behavior in this circumstance.

Other recommendations have noted that implementation specific option numbers start at 513. Future revision committees may wish to add one or more of the following options to 7.12.4
-Option value 2: The specification initialization set shall be used and any update values in non-volatile storage shall be returned to their specification values.
-Option value 3: Return any pending update values in nonvolatile storage to their specification values but do not cause the clock to execute reboot or power cycle behavior.

QUESTION 17: (submitted May 3, 2004)
Variance within the 1588 protocol is defined as being represented as 2^8 log2(PTP Variance). Figure 22 shows "2^8 log2" being applied to "Variance". However, when running the BMC the ordering is done on the applicable clock_variance unmodified because this is already represented as such...correct? In other words, when ordering the variances use the data set clock variance and do not apply 2^8 log2 as depicted in the figure. My guess is the figure is referring to the Variance as the actual 'PTP Variance' of the clock....not what is sent on the wire and stored in the data set(s).

RESPONSE:
The questioner is correct, there is an ambiguity in the Figure 22 specification. This should be corrected at the next revision of the standard. Until that time it is recommended that Figure 22 be interpreted as follows:

- The term "Variance of A" or "Variance of B" shall refer to the relevant variance as computed according to 7.7.1.
- There is no change in the data sets or messages. These variances are as stated in the specification as being represented as defined in 7.7.2.

QUESTION 18: (submitted May 3, 2004)
Definition of portState enumeration
The management message PTP_MM_PORT_DATA_SET contains a field "portState". The content of this field is supposed to be the port state for the clock. The problem is that the port state is defined as an enumeration with a defined set of symbolic values, and there is no guarantee that different compilers enumerate in the same way. Likewise, table 7 explains the purpose of each state, but does not state the enumeration order. Therefore, a table containing the official integer equivalent of each state must be included in the standard.

RESPONSE:
The PTP_MM_PORT_DATA_SET management message is defined in section 8.6.1.10.16. Table 34 states that its portState field is the port_state variable, and that variable is defined in section 7.4.6.1. The enumeration of port states is given in section 7.3.1, Table 7.

The questioner is correct -- the enumeration does not assign numeric values to the states. This should be corrected in the next revision of the standard. Until then it is recommended that implementations number the port states in the order given in table 7, starting with PTP_INITIALIZING = 1 and increasing in steps of 1.

QUESTION 19: (submitted May 3, 2004)
Definition and representation of variance
In chapter 7.7.2 the variance representation is defined but there is a big problem in the definition. The PTP variance has dimension (seconds squared) and it is therefore impossible to take the logarithm. As the representation of time and time differences are numbers, it is of course possible to take the logarithm of that number, but doing so implies a division by the time measurement unit, and that unit is not specified. Measuring the residual measurements in seconds gives very different results from measuring them in nanoseconds!

RESPONSE:
The questioner is correct. The next revision of the standard should make the units explicit. Until then it is recommended that implementations compute an estimate of the variance (the first bullet of section 7.7.2) in units of seconds^2.

The net result of 7.7.2 interpreted as above will allow for a range of variances roughly from 2^128 to 2^-128 with 8 bits of resolution in the mantissa of the logarithm. Note that 7.6.6, 7.7.2 and 7.0 together define threshold and hysteresis properties used in comparing two variance representations. With the above interpretation and the values in table 24 two variance representations must differ by a factor of 2 before they exceed the thresholds in figure 22. The hysteresis is a multiplicative factor of 0.5 based on the values in table 24.

QUESTION 20: (submitted May 3, 2004)
It would be nice to be able to learn more about a clock, perhaps in the PTP_CLOCK_IDENTITY message. For example it is useful to know whether a clock is full, slave only, or management only, product revision information, etc.

RESPONSE:
This should be considered in future revisions. This and a number of other items will require some extension mechanism to existing messages. In order to allow this to happen it is recommended that implementations of the current standard when receiving messages from a clock of some future revision standard in which there are additional message fields appended to those defined in the existing standard, or for which reserved fields

have been defined, ignore such additional and by necessity uninterpretable information in the extensions or reserved fields. This at least allows for some measure of backward compatibility.