Contribution to the Precise Networked Clock Synchronization Working Group for the Revision of IEEE 1588-2008

Working item: Upkeep proposal 89

Title: Field for Internal Implementation

Document Revision: 1.1

Subcommittee name: Upkeep

Author(s): Rodney Cummings

Date: March 5, 2015

Approved on: <date> By: <plenary %of quorum OR e-ballot %of voting members> (record yea, nay, abstain as well as %)

Incorporated into 1588 draft by P1588 editor: <date>

Summary

This proposal specifies a new field in the common header for use by internal implementations, meaning that it is transmitted as zero on the wire, but it can use a non-zero value within a single clock.

Discussion

The issue driving this proposal was described in the following presentation:

https://ieee-sa.centraldesktop.com/1588/file/36141473

and discussed in the following comment thread:

https://ieee-sa.centraldesktop.com/1588/discussion/28522166

To summarize the background for this proposal:

- There are implementations of 1588-2008 BC/TC in which each port is not immediately available to the centralized PTP clock. In these implementations, acquisition of each port’s transmit/receive timestamp from the centralized PTP clock would result in significant problems with accuracy/precision.
- It is extremely difficult for many of these implementations to share a physical timing signal between clock and ports.
- It is extremely difficult for many of these implementations to pass information between clock and ports using sideband techniques (i.e. outside the PTP message). One issue is that sideband techniques make it difficult to associate the extra info with the corresponding PTP message.
- It is extremely difficult for many of these implementations to append octets to the PTP message for timestamps between clock and ports (e.g. TLV or after end-of-frame). For example, if an external PHY timestamps, that PHY cannot add payload to the PTP message prior to passing it to the MAC (per IEEE 802 conformance).
- Many of these implementations today use fields of the PTP message marked as “reserved” for transfer of timestamps between port and clock. There is no clear consistency in which fields are used, or in the number of reserved bits used.
- In 1588-2008 common header (subclause 13.3), there are 3 fields marked explicitly as “reserved”, for a total of 44 bits.
- 1588-2008 is clear that reserved fields are dedicated exclusively to future editions of the 1588 standard (e.g. subclause 4.2.8), and that they shall be transmitted as zero and ignored on receive (e.g. subclause 13.2).
- Of the implementations that use reserved fields for port/clock timestamps, most seem to conform to 1588-2008 with respect to on-the-wire behaviour (transmit field as zero on the wire, but use a non-zero value between clock and port). Nevertheless, these implementations do not conform to the requirements of subclause 4.2.8, and that lack of conformance brings risk of incompatibility to the future 1588 revision.
- If the 1588 revision were to use most of the bits currently marked as reserved for on-the-wire purposes, those non-reserved bits must be transferred between clock and port. This would effectively break implementations that rely on bits in the common header for internal clock/port timestamps.

Despite the 1588-2008 conformance issues, many 1588 products that use reserved fields for internal implementation have contributed greatly to the success of 1588 technology. The goal of this proposal is to find a solution that is sufficient to allow these products to exchange internal timestamps between clock and ports in a manner that conforms to all future 1588 revisions.

This proposal changes bits in the common header currently marked as “reserved” into a new “internalImplementation” field, to be used for internal information between clock and ports.

This proposal works under the following assumptions:

1. The 1588 standard requires reserved fields in the common header. It is obviously reasonable for any standard to dedicate reserved fields for future enhancements to the standard. Reserved fields are critical to serving the needs of the broader community for decades to come, which must be weighed against short-term implementation needs.
2. The 1588 revision cannot guarantee that existing implementations using reserved fields for internal clock/port transfer will conform to the 1588 revision without change. Since there is no specification for the fields used, or the number of bits, this would require changing **all** reserved fields into internal fields. Due to the constraints of assumption #1, this cannot be done.
3. The proposal assumes that 32 bits is sufficient for all internal clock/port exchange. 32 bits allocates over 72% (32/44) of the existing reserved bits to internal use.
4. Due to the importance of assumption #1, the 1588 revision will add text to clarify that the remaining 12 reserved bits of the common header **shall not** be used for internal implementation.
5. This proposal has an obvious risk. If a future 1588 revision requires use of more than 12 new bits, all PTP messages may need to be lengthened by that 1588 revision. This could be a mandatory TLV, or expanding the common header. Although this doesn’t necessarily break backward compatibility, it will be a large hardware/software change. This proposal assumes that such a large change is best deferred to the future if/when needed, rather than doing it pre-emptively in the pending 1588 revision.
6. This proposal does not attempt to specify consistent use of the internalImplementation field, since those issues are outside the scope of 1588. For example, for timestamping in an external PHY, a PHY specification for the use of the internalImplementation field may enable greater interoperability among multiple PHY vendors.
7. This proposal assumes that the internalImplementation field is needed for some non-event messages (e.g. DelayResp), and that some internal implementations do not distinguish event messages from non-event messages. In addition, many 1588 proposals for use of reserved octets are required for event messages as well as non-event (e.g. profile isolation). Therefore, the proposal assumes that there is no significant value in distinguishing message type or class for the internalImplementation field. The internalImplementation field resides in the PTP common header, and applies to all messages.

Proposal

<original text in normal font; deleted text in ~~strikeout~~; new text in *italics* with yellow highlight; comments not part of the proposed modifications to the standard text are red text within braces>

13.3.1 General header specifications

The common header for all PTP messages shall be as specified in Table 18.

Table 18 –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
flagField								2	6
correctionField								8	8
~~reserved~~ *internalImplementation*								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

13.3.2.8 internalImplementation

<Insert this subclause between the existing 13.3.2.7 and 13.3.2.8. Increment the last subclause number for the existing 13.3.2.8 through 13.3.2.11.>

The four octets of the internalImplementation field may be used for internal implementation of a PTP clock and its ports. For example, if the clock consists of multiple hardware components that are not synchronized, internalImplementation can be used to transfer an internal timestamp between components (e.g. a physical layer chip and the clock's processor).

The internalImplementation field is not used for features of the IEEE-1588 standard, and it has no meaning from one clock to another. At the media of each PTP port, the internalImplementation field shall be transmitted with all bits of the field 0, and ignored on receive.

NOTE – As specified in subclause 4.2.8, fields marked as "reserved" are dedicated for use by future editions of this standard, and shall not be used for any other purpose. Therefore, reserved fields shall not be used for internal implementation purposes. If a reserved field is used for internal implementation, that implementation will demonstrate its non-conformance when a future edition of the standard uses the field for exchange of information between clocks.

Alternatives

This clause contains alternatives to the preceding proposal for discussion.

1588 Specifies Use of the Field

1. As proposed for the distributed chassis use case https://ieee-sa.centraldesktop.com/1588/file/36141473

   Rename internalImplementation to fracTime, and specify fracTime's internal usage in 1588.

   Designate an additional octet of reserved bits (offset 5) as a new usageField. The usageField is a bitfield for which each bit specifies the interpretation of the fracTime (still 4 octets). Bit 0(0x01) specifies a timestamp of nanosecond resolution (0 to 999999999). Bit 1 (0x02) specifies 16 bits for nanoseconds, and 16 bits for fractional nanoseconds (like correctionField). Additional specifications in 1588 apply use of fracTime to the existing TC and BC specifications for the clock.

   NOTE: This proposal collides with the Upkeep proposal on Profile Isolation, which uses the octet at offset 5 as its otwProfileDesignator.

2. Same alternative as #1, but use bits of offsets 16-19 as the usageField. For example, 4 bits for usageField, and 28 bits for fracTime.

3. Same alternative as #1, but use the 4 reserved bits of offset 1 as the usageField (fracTime still 4 octets).

Specific to Message Class / Type

1. The internalImplementation field is contained within messages of Event class only. For messages of General class (i.e. type of Follow_Up, Delay_Resp, Pdelay_Resp_Follow_Up, Announce, Signaling, Management), the field remains reserved.
2. The internalImplementation field is contained within timing messages only (i.e. type of Sync, Delay_Req, Pdelay_Req, Pdelay_Resp, Follow_Up, Delay_Resp, Pdelay_Resp_Follow_Up). For non-timing messages (i.e. Announce, Signaling, Management), the field remains reserved.

Name of Field

1. unused
2. staticBits