P802.1AS-2025-Rev: Refactoring MD clauses discussion

Alon Regev [Keysight]

Introduction

- This slide set discusses options and makes recommendations for refactoring MD clauses to avoid MD clauses that refer to other MD clauses with special cases in the refrences clauses. Biggest focus is on the current HDE support where clause 19 mostly contains pointers into clause 11, but clause 11 also has conditionals based on HDE vs full-duplex ethernet. The topic has been brought to discussion in comment #4 against draft 0.3 of P802.1AS-2025-Rev
- Underlying documents:
  - [AS]: 802.1AS-2025
  - [ASRev]: P802.1AS-2025 Rev/D0.3
- Previous relevant presentations:
  - https://www.ieee802.org/1/files/private/asds-drafts/d1/ds-specht-regev-cl-7-3-and-11-0725-v01.pdf
  - https://www.ieee802.org/1/files/private/asds-drafts/d1/ds-specht-regev-fde-0725-v01.pdf
  - https://www.ieee802.org/1/files/public/docs2025/ds-regev-comparing-document-structure-options-0925-v00.pdf
  - https://www.ieee802.org/1/files/public/docs2026/as-2025-rev-regev-refactor-discussion-v00.pdf

802.1AS structure observations

- 802.1AS divided into two layers: media independent and media dependent
  - At the time 802.1AS-2011 was released, the MD layer was mostly the only “option” with layering mostly clean
    - 802.1AS-2011 had CSN defined in Annex E, mostly referring to portions of clauses 10 and 11
  - 802.1AS-2020 added several features that affected structure
    - Example: multiple domain support and CMLDS was added, changed some variables to be per-domain and others to be across domains; but not completely propagated to clauses 12, 13, and 16
  - 802.1ASdm added several features
    - Example: Drift Tracking was specified to only work with Clause 11 MD, but the feature (potentially in the future) may be needed for other MD types as we may have the same issue with temperature-driven frequency drift occurring on other MD types.
      - Drift Tracking involved changes to clauses 9 and 10 to support it (in addition to clause 11), but it was decided to only support on Clause 11
- We are moving towards having more options (i.e. external port configuration, optional announce messages, peer delay measurement, etc.)
- Over time, the majority of “optional” features were added to clause 11 and often not to other MD clauses, and often in a way that also affects common clauses

This slide is a repeat from
ds-regev-comparing-document-structure-options-0925-v00.pdf

HDE: added in 802.1ASds

- 802.1ASds added a new clause 19 which mostly points to clause 11
- Also changed clause 11 rather that duplicating state machines / variables with small changes
- While a couple of state machine have changes to support HDE, the biggest changes are
  - Functions that behave differently in the two modes
  - Many variables are per-domain in HDE and shared across domains in FDE
- The changes to functions and variables make this hard to resolve in a “clean” way
- CSN also has its own clause with pointers to clause 11, but does not modify clause 11 (rather it describes changes to behavior of clause 11 in clause 16) and does not completely address the multi-domain issues or the fact that referenced variables from clause 10 are not shared across domains

This slide is a repeat from
ds-regev-comparing-document-structure-options-0925-v00.pdf

A little more on clause 11 / 19 split in 802.1ASds

- Even though clauses 11 (MD for full-duplex point-to-point ethernet) and clause 19 (MD for HDE) are closely coupled and both sides reference the other
- Clause 19 is written almost entirely as references to clause 11
  - In ~15 instances it follows the template “as specified in 11.x.x.x… However, for HDE…”
- Also, HDE still appears ~8 times in clause 11, embedded in variable definitions and notes
- State machines in clause 11 don’t reference HDE, but variables used by the state machines were specifically added to enable different behavior (they are effectively parameters to the state machine that depend on whether HDE or full-duplex ethernet is used).
- For details, see https://www.ieee802.org/1/files/private/asds-drafts/d1/ds-specht-regev-cl-7-3-and-11-0725-v01.pdf

This slide is a repeat from as-2025-rev-regev-refactor-discussion-v00.pdf

The role of CMLDS / multi-domain support

- I believe that the choices made during the integration of multi-domain support and CMLDS into 802.1AS-2020 cause a lot of the issues today
  - Multi-domain support assumes that some variables (those associated with link delay measurement or compensation) are only on domain 0 and are effectively shared with other domains
    - This is true for full-duplex peer-to-peer Ethernet, but not necessarily for other networks
  - CMLDS behavior is defined in BOTH clauses 10 & 11 even though it is effectively only applicable to full-duplex peer-to-peer Ethernet
  - A good example is table 10-1 which defines which clause 10 variables are shared across within a Link Ports for CMLDS (i.e. shared across domains sharing the same physical port)
- This could have been done in a cleaner way
  - One example: In clause 10, leave all variables as per PTP Instance or per PTP Port per PTP Instance and in clause 11 have the CMLDS and transport-specific delay peer-delay state machines or functions set the values across all PTP instances, leaving clause 10 as generic to all MDs.

This slide is a repeat from
ds-regev-comparing-document-structure-options-0925-v00.pdf

Another example: drift tracking clause 10/11 split

- Drift tracking, in theory, could be supported on any media, but today it is tightly coupled with full-duplex point-to-point ethernet (Cl. 10)
  - This is a big limitation for any potential network that contains both Ethernet and another media (for example, a single WiFi hop in the middle). Drift tracking relies on all nodes in the paths to update and forward the drift information.
- Drift tracking in the current 802.1AS violates the division between media-independent and media-dependent clauses
  - Instead of just being implemented in the MD clause, both clause 10 and 11 are modified to support the Drift_Tracking TLV (which arguably is a specific implementation of drift tracking) and variables are defined in clause 10 (for example syncStepsRemoved) that are only used in clause 11.
  - This is shown (for example) in the conformance clause 5.4.2 which states “Implement the Drift_Tracking TLV as specified in 10.2.4.26, 10.2.4.27, 10.2.4.28, 11.2.14, 11.2.15, and 11.4.4”
  - A cleaner approach would be to have the global variables which would be needed for a generic drift tracking accumaltion (such as rateRatioDrift) defined in clause 10 and less generic global variables as well as the Drift_Tracking TLV specification at the MD level.
  - Even better would be to define the Drift_Tracking TLV and related calculations as a feature that when implemented in an MD clause can augment the clause 10 state machine behavior without being in clause 10 (see slides below).

This slide is a repeat from [as-2025-rev-regev-refactor-discussion-v00.pdf]

Regrouping and evaluating options

- Consensus in Aug 2026 was to create a new clause 21 containing “reusable objects” and to modify MD clauses 11 & 19 to use these
- Offline discussions recommend re-evaluating the solution proposed in ds-specht-regev-cl-7-3-and-11-0725-v01.pdf
  - combines clause 11 & 19 into the original clause 19
  - Cleanest solution (at least without major refactoring)
- Possible variation
  - Creating a new “Ethernet common” clause containing the text proposed in ds-specht-regev-cl-7-3-and-11-0725-v01.pdf and changing clauses 11 & 19 to only provide a short overview and a pointer to the new clause
    - allows maintaining references to Cl. 11 as the FDE clause and Cl. 9 as the HDE clause

Backup slides from past discussions

Proposal: Reusable features

- Reusable features would be defined in a “Features” clause and can be use by an MD clause or called out by a TSN profile.
  - This is similar to the 1588 “General optional features” (IEEE Std 1588 clause 16)
- A reusable feature may define “objects” that can be instantiated in the MD clauses.
  - A feature object may be a function, a state machine, a grouping of such items, etc. that is instantiated together (in an MD clause or another object)
  - Example: The MDPdelayReq, MDPdelayResp state macheines and related variables and functions can be grouped together as a Pdelay feature object
- A reusable feature may augment functions or structures used in Clause 10 (media independent clause)
  - For example, if the feature needs to transfer additional data from the timeReceiver port to timeTransmitter ports, the feature may add feature-specific data to the PortSyncSync structure allowing the data to be passed from the TR port through the SIteSync state machine to the TT port.
  - Drift tracking, as an example, could be implemented in this way, without adding feature-specific steps in the SiteSync state machine (as was done previously)

Feature objects

- An feature object may be a function, a state machine, a grouping of such items, etc. that is instantiated together (in an MD clause or another object)
- There is a separate definition (in a feature clause) and instantiation (in the MD clause)
- The definition and instantiation of such a feature object include both
  - “Parameters” define instantiation specific constants or limitations
  - “interfaces” connect variables in the object to global variables
- This approach makes it easier to have hierarchical instantiations
- Precedent for parameterized reuse: In 802.1AS parameterized, shared content is used in Annex F (PTP Profile) which lists the per-PTP-profile parameters of 802.1AS in a table. A parameterized object reuse approach is also used in 802.11

Where to put the reusable blocks

- New clause between clause 10 and 11
  - Pros: most readable, with common blocks ahead of specializations
  - Cons: forces a renumber of clause 11-20, which are referenced by EVERY TSN profile
- New subclause of clause 10
  - Pros: good from reading order with common blocks ahead of specializations
  - Cons: clause 10 is no longer purely media-independent, as it contains features that are used by a subset of MD clauses
- New subclause at the end (e.g. new Clause 21)
  - Pros: no renumbering of clauses
  - Cons: less readable
- New normative Annex at the end
  - Pros: no renumbering of clauses
  - Cons: less readable; putting state machines in an annex is atypical
- Recommendation: New subclause 21

How to plumb and document the plumbing of the MD sections

- Explicitly define variable interconnects where they don't follow the per PTP Instance or per PTP-port shared variables.
  - In clause 10, define interconnects just for clause 10, if needed (not needed at the current time)
  - Interconnects within MD and between MD & MI are defined in each MD clause, as needed (for example, the connection of the CMLDS / transport-specific delay mechanism)
  - The connectivity of objects is defined where the are instantiated in the spec (not where they are defined)
  - While objects can access non-object variables, the intention is for most connections to objects to be explicit using the object interface
- In the MD sections, show a diagram showing the interconnects (see below slides for examples)
- tables 10-1, 10-3 and 11-2 clarifications
  - The tables can be simplified to just specify per-PTP Port or per-PTP Instance
  - On a side note, the BEGIN variable is defined as per-PTP Instance and shared across all instances
    - It really should be treated as per-PTP Instance, except for the CMLDS where it should be per-Link Port
    - as it's used, there are cases where it should split to have a per-PTP Port version – for example a physical port linking up should be treated as BEGIN (or equivalent) for the PTP portion of the code, but should not restart the per-PTP Instance state machines

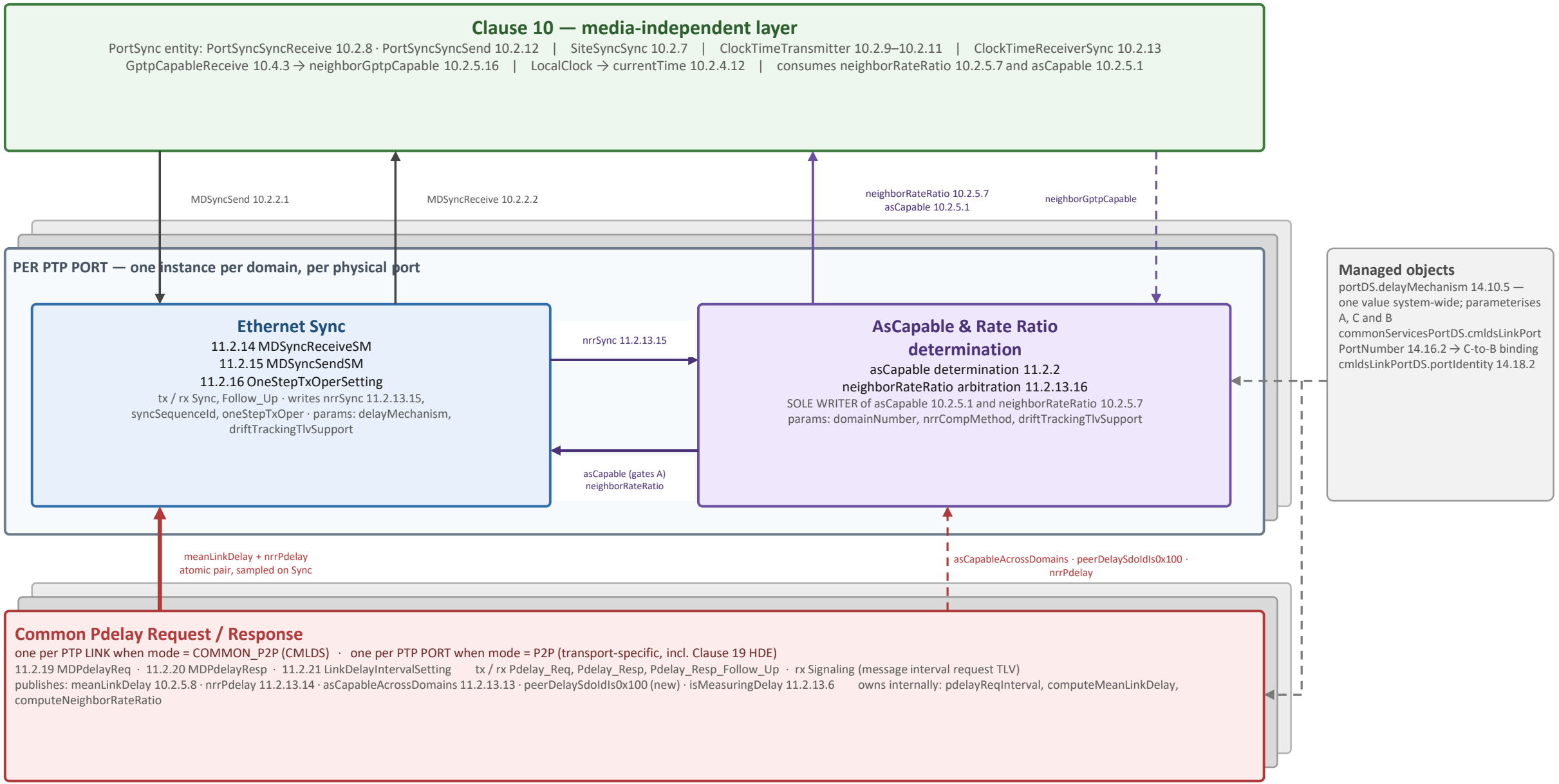
Implementing clauses 11 & 19 using features

- Create “Ethernet Interface” feature including
  - clauses 11.2.3 to 11.2.11, 11.4.1, 11.4.2, non-message specific parts of 11.3
  - Interface to 802.3 via service interfaces defined in 802.1AC and 802.3
  - Related limitations on 802.3 operation
- Create “Ethernet Sync” feature including
  - Clauses 11.2.14 to 11.2.16, 11.4.3, 11.4.4, sync/follow-up specific parts of 11.3, 11.5.2.1,
  - Provides the MDSyncSendSM and MDSyncReceiveSM state machines and related functions / variables and information on how to transmit over Ethernet
- Create “common Pdelay request/response” feature including
  - Clauses 11.2.19, 11.2.20, 11.5.2.2, 11.5.3, and 11.5.4, and parts of 11.2.17 rewritten to be generic
  - The MDPdelayReqSM and MDPdelayRespSM state machines and related functions / variables as well as timing requirements
- Create “Ethernet Pdelay” feature
  - Clauses 11.4.5 to 11.4.7, Pdelay specific parts of 11.3
  - Ethernet specific Pdelay_req, Pdelay_resp, and Pdelay_resp_follow_up message requirements

Clause 11 example new structure

- Retain clauses 11.1, 11.2.1, 11.2.2, 11.2.10, 11.2.11
- Rename 11.2 to “MD entity specific to full-duplex point-to-point Ethernet links”
- Modify (and renumber) 11.2.12, 11.2.13, 11.2.17, and 11.2.18 to reference only what’s not internal to the features integrated
- Add to new subclause into 11.2 titled “objects integrated into the MD entity” specifying the use of the 4 new objects
  - Instantiate the feature objects (for the “Ethernet Sync” and “common Pdelay request/response” features)
  - Specify the parameters features (currently I only envision the “common Pdelay request/response” feature having parameters)
- Specify the interconnect between the feature objects as per the following slides
  - asCapable / asCapableAcrossDomains is done outside the objects (due to differences with 19)
  - neighborRateRatio arbitration between nrrSync and nrrPdelay is done outside the objects (due to differences with 19)
  - Add a wiring diagram showing the interconnects

Object Interconnects



Interconnect of clause 11 to media independent clause

