An Idealistic Model for P802.1DU

Johannes Specht
(Self; Analog Devices, Inc.; Mitsubishi Electric Corporation; Phoenix Contact GmbH & Co. KG; PROFIBUS Nutzerorganisation e.V.; Siemens AG; Texas Instruments, Inc.)

DCN 1-22-0015-01-ICne

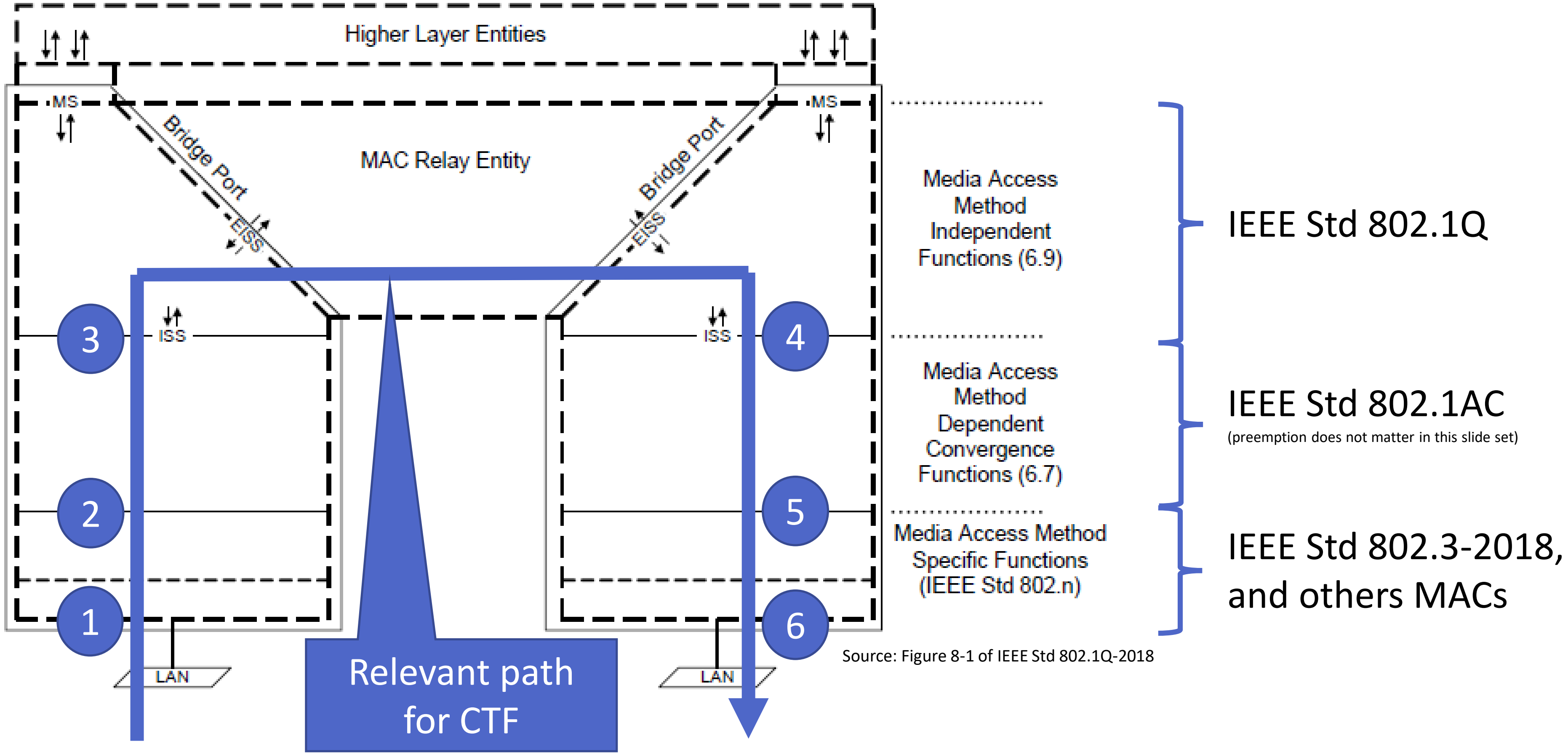
Introduction

- This slide set is a result of the inspiring discussions during the IEEE WG 802.3 PAR&CSD review ad-hoc on February 24, 2022 and subsequent meetings until the IEEE WG 802.3 closing plenary meeting in March 17, 2022.
- The impression of the author is, that at there may be the concern that P802.1DU would break compatibility with the IEEE Std 802.3-2018 MAC **model:**
  - The following properties appear to be of primary interest to be retained: (https://www.ieee802.org/3/email_dialog/msg01286.html):
    - Leave MA_UNITDATA.request as an **atomic** (and **instantaneous!**) event
    - Leave MA_UNITDATA.indiciation as an **atomic** (and **instantaneous!**) event

    The interpretation of **atomic** and **instantaneous** appears to be an idealized one – it is an *internal* model in an IEEE 802 Standard; only the *external* visible behavior of Bridge **implementations** matters for conformance.
  - In addition, there may be concerns that octet-by-octet transfers above an 802.3 MAC are inevitable.
- In contrast, the **model** demonstrated in https://www.ieee802.org/1/files/public/docs2021/new-specht-ctf-802-1-1121-v01.pdf is closer to Bridge implementations.
- <u>However:</u>
  - There is no issue in considering an idealized **model** for P802.1DU that satisfying the aforesaid properties very explicitly.
  - The subsequent slides outline how such a **model** could look like.
  - **Both models in combination** demonstrate a spectrum of options from which we can choose during Stds developments.

The Basics Explained

Layering/Baggy-Pants Diagram

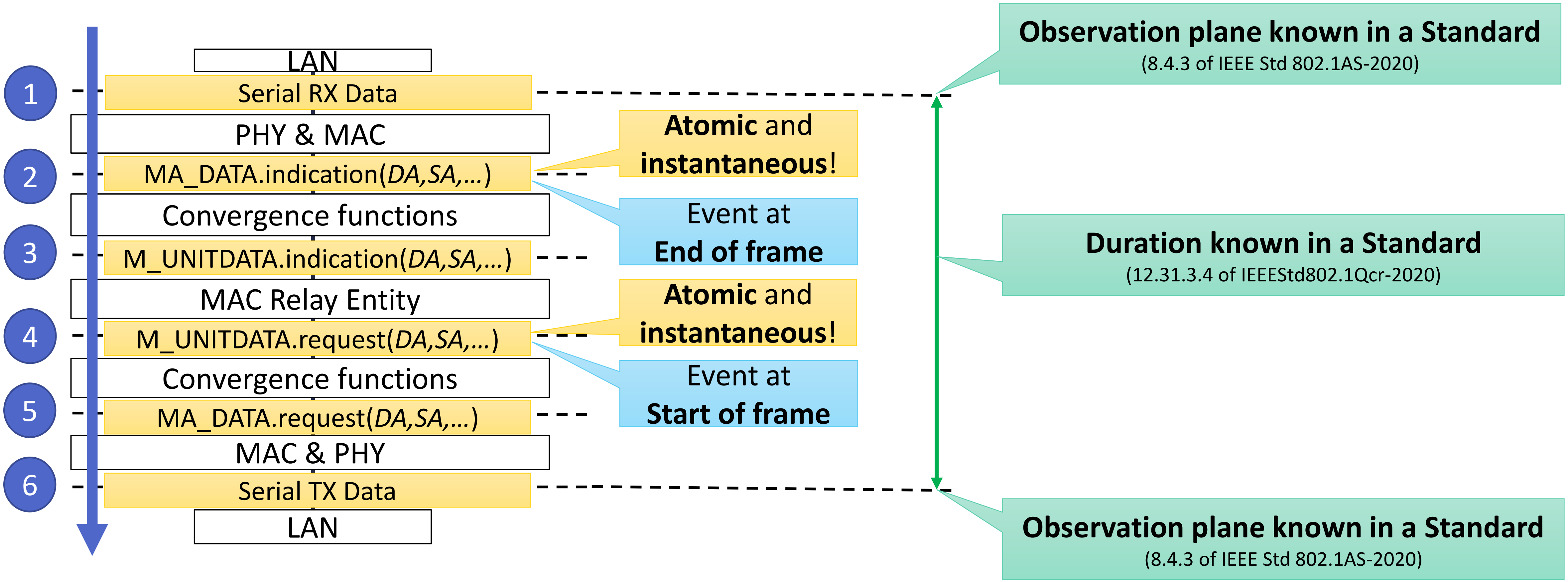


Source: Figure 8-1 of IEEE Std 802.1Q-2018

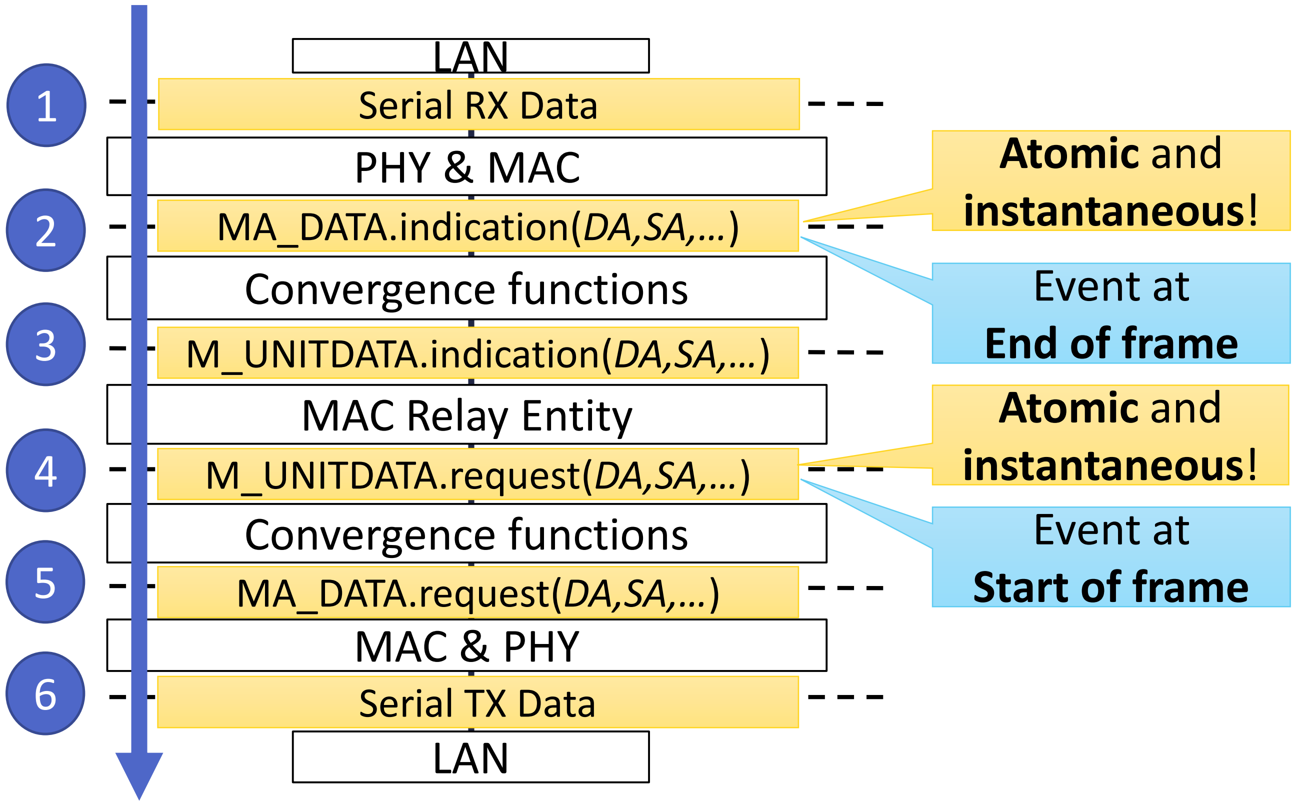
Notes:

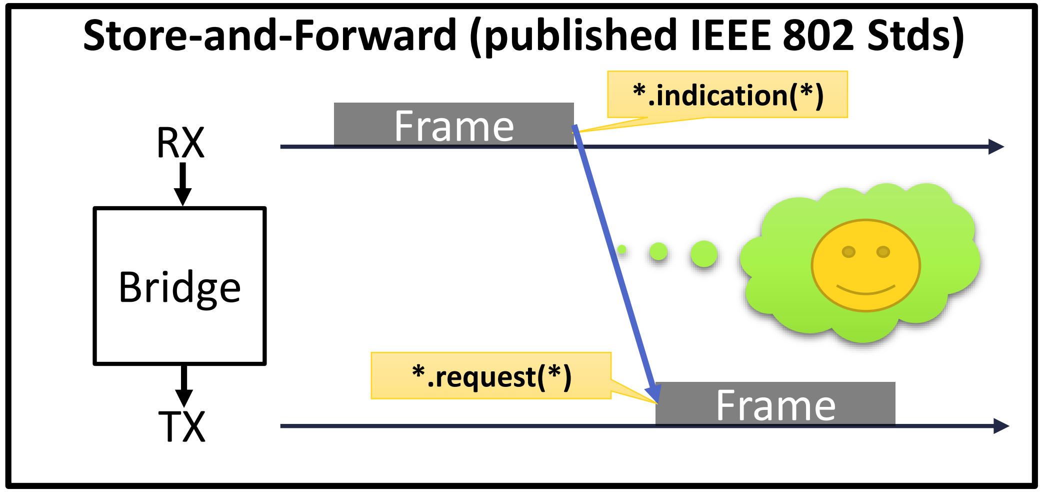
- The subsequent slides omit ISS⇔EISS translations for simplicity.
- While this slide set refers to the layering in existing IEEE 802.1 base standards (IEEE Std 802.1Q, IEEE 802.1AC, etc.), reasons for a new base Standard project instead of amendment projects are found in https://mentor.ieee.org/802.1/dcn/21/1-21-0037-00-ICne-ieee-802-tutorial-cut-through-forwarding-ctf-among-ethernet-networks.pdf.

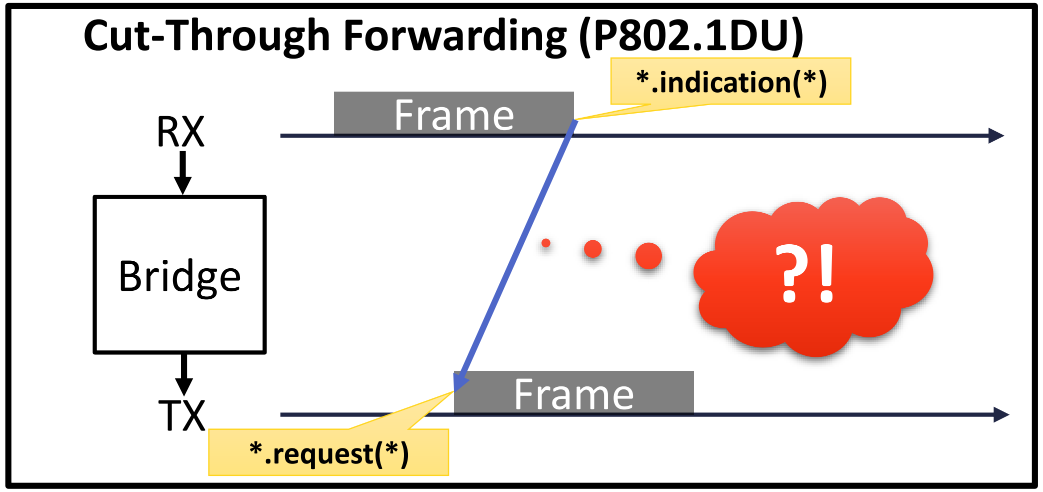
Linearized View



Store-and-Forward vs. Cut-Through Forwarding

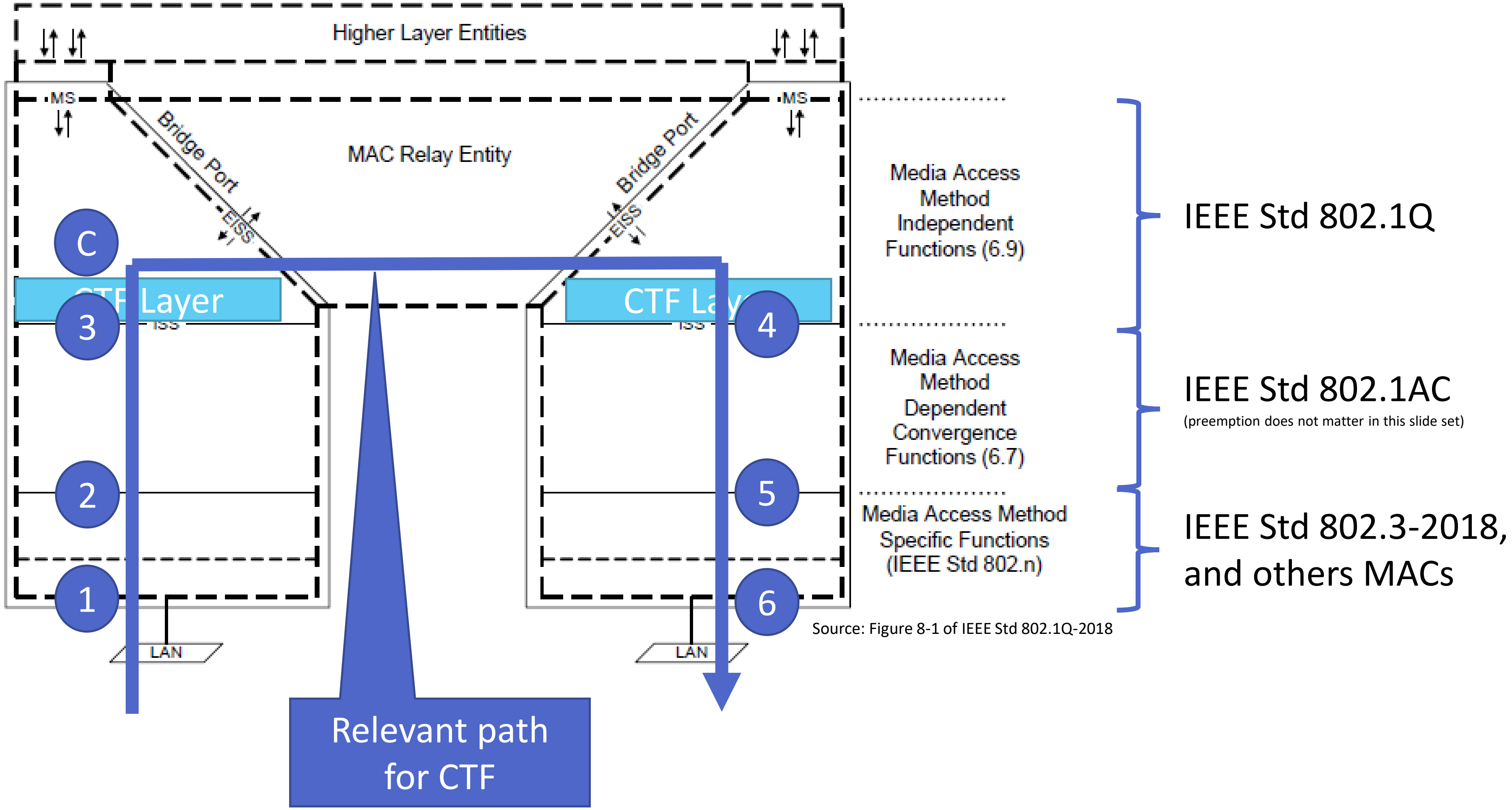






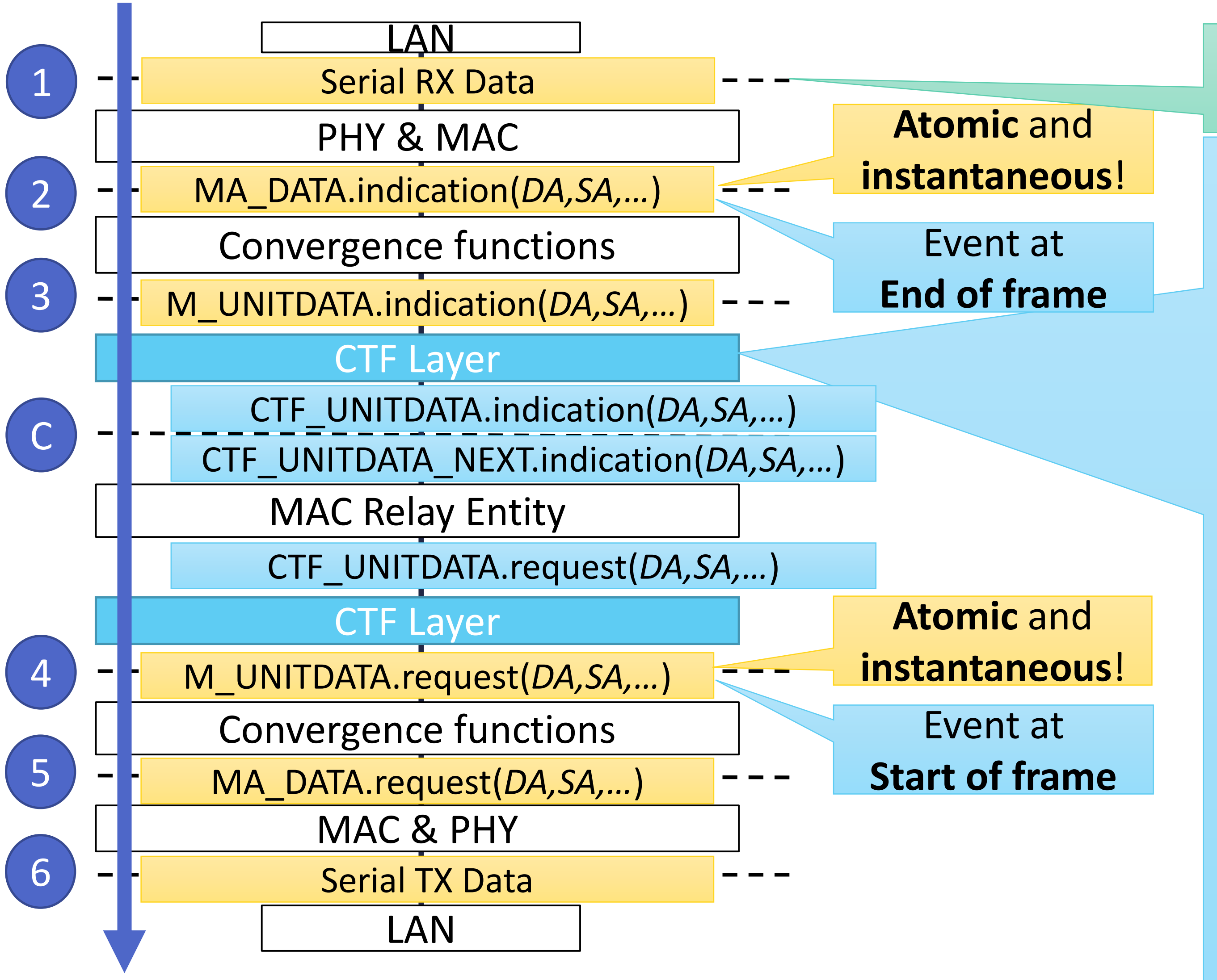
Layering

Layering/Baggy-Pants Diagram



Source: Figure 8-1 of IEEE Std 802.1Q-2018

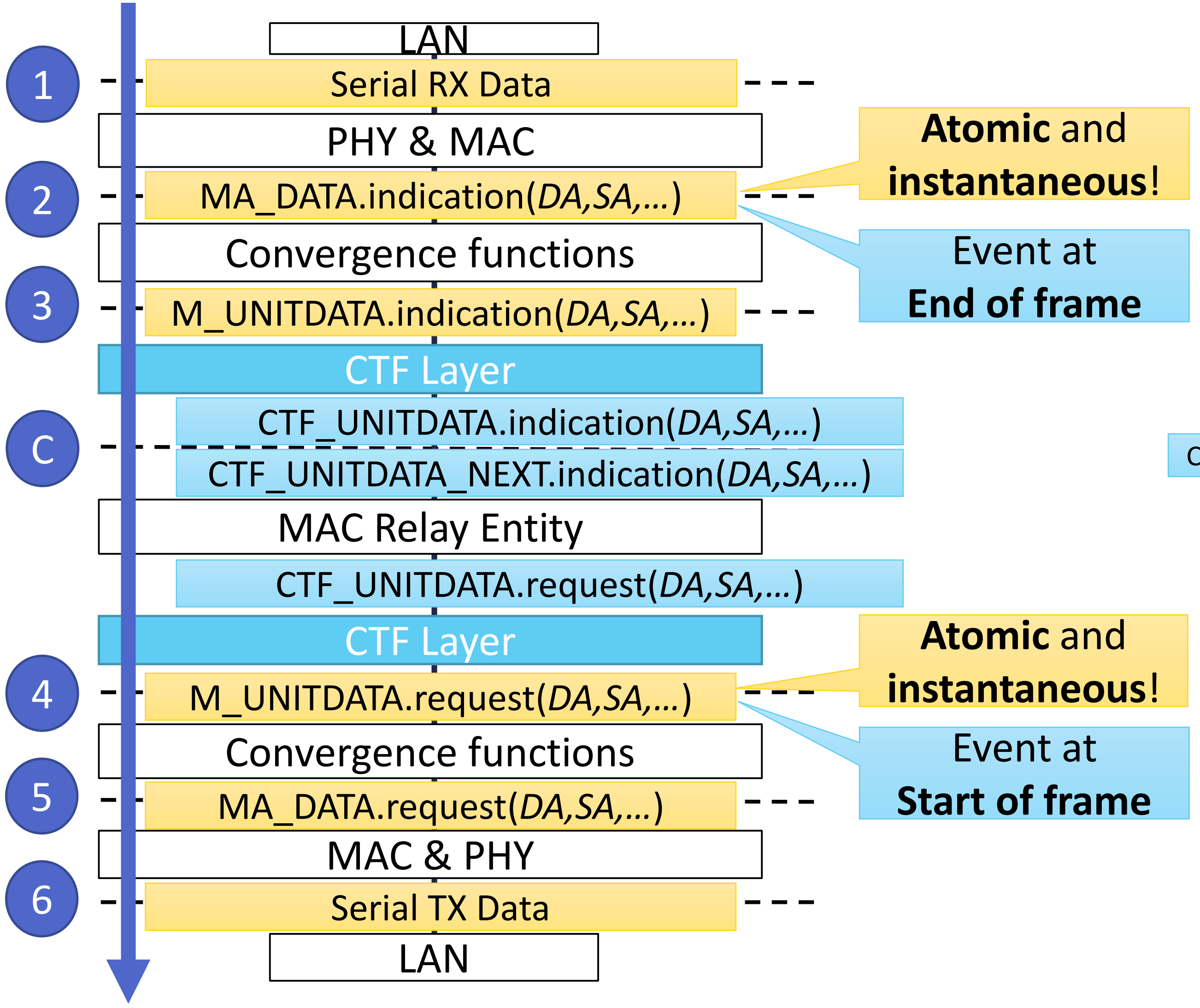
Linearized View - Description

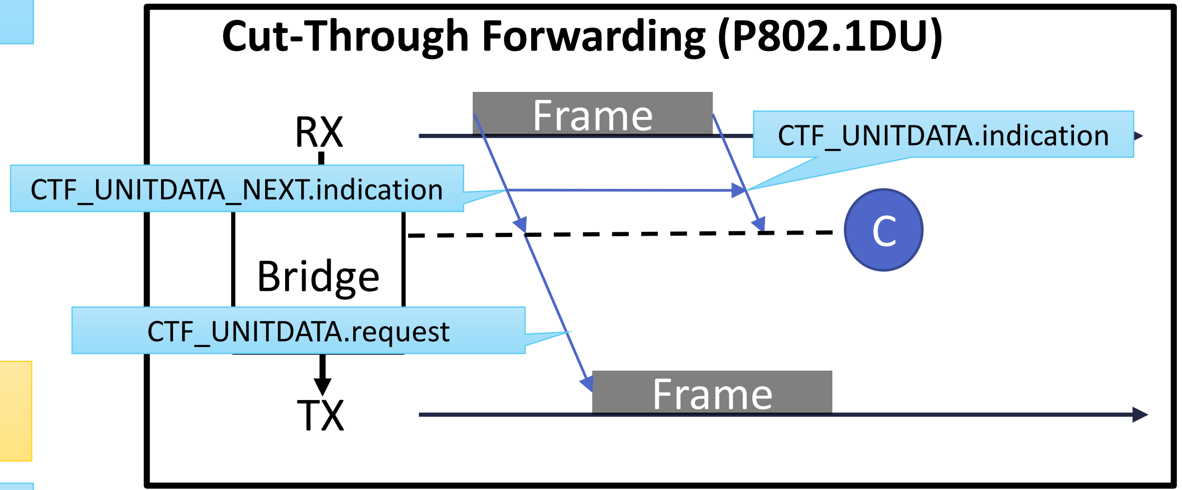


Summarized Description

- Operates on the sequence of M_UNITDATA.indication primitive invocations on the ISS
- Issues CTF_UNITDATA.indication and CTF_UNITDATA_NEXT.indication invocations for each M_UNITDATA.invocation
- CTF_UNITDATA.indication is issues instantaneously upon reception of M_UNITDATA.invocation
- **Look-ahead** of CTF_UNITDATA.indication invocations*:*
  - Invocation of CTF_UNITDATA_NEXT.indication ***before*** the associated CTF_UNITDATA.indication invocation
  - Time difference between CTF_UNITDATA.indication and CTF_UNITDATA_NEXT.indication pair is basically the frame length, measured at the observation plane
- Having both (CTF_UNITDATA.indication and associated CTF_UNITDATA_NEXT.indication) appears reasonable:
  - Fallback to S&F in the relay MAC Relay Entity
  - Separation of Frames not intended for CTF (e.g., to higher layer entities in the baggy-pants diagram)
- Stylistic word-smithing (e.g., "… 64 octet times after the invocation of CTF_UNITDATA_NEXT.indication …")

Linearized View - Simplified Illustration





Doesn’t this violate Physics?

- Probably in the world of **implementations**, but this is an idealistic **model**. There is no need for models to follow such physical rules.
- Other aspects in 802 models sometimes narrow physical realities, for example, ***instantaneous*** events:
  - Instantaneous ≈ no perceivable progress in time
    → Depends on the resolution
  - Example resolutions:
    1. During a **0.0 seconds** time interval
       (idealized, but impossible in implementations)
    2. During a **single clock cycle** in an RTL model
       (depends on the clock frequency)
    3. During a **single assignment** statement in C/C++ code
       (depends on number of CPU instructions, **clock cycles** per CPU instruction, …)
    4. During a **single octet time** on the wire
       (depends on the link speed)
    5. During a **single frame** on the wire
       (depends on the frame length)

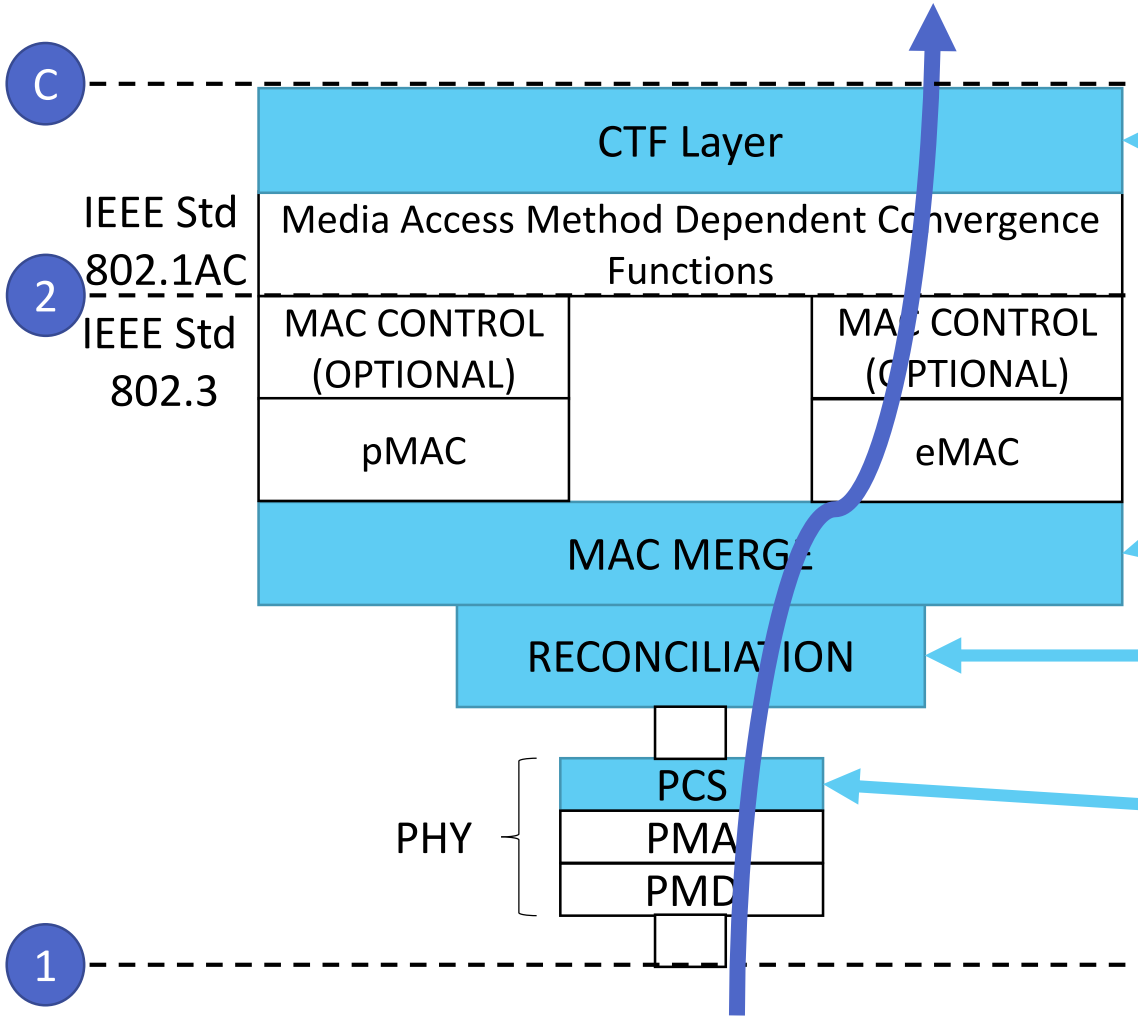
Look-aheads in IEEE 802 Standards

- ➢ **It may be uncommon for an IEEE 802.1 Standard …**
- ➢ **… but it exists in other IEEE 802 Standards**

IEEE Std 802.3-2018 Reference	Name	Description
36.2.5.1.4	check_end	**Prescient** End_of_Packet and Carrier_Extend **function** used by the PCS Receive process to set RX_ER and RXD<7:0> signals. The check_end function returns the current and next two codegroups in rx_code-group<9:0>.
48.2.6.1.4	check_end	**Prescient** Terminate **function** used by the PCS Receive process to set the RXD<31:0> and RXC<3:0> signals to indicate Error if a running disparity error was propagated to any Idle code-groups in ‖T‖, or to the column following ‖T‖. The XGMII Error control character is returned in all lanes less than n in ‖T‖, where n identifies the specific Terminate ordered-set ‖Tn‖, for which a running disparity error or any code-groups other than /A/ or /K/ are recognized in the column following ‖T‖. The XGMII Error control character is also returned in all lanes greater than n in the column prior to ‖T‖, where n identifies the specific Terminate ordered-set ‖Tn‖, for which a running disparity error or any code group other than /K/ is recognized in the corresponding lane of ‖T‖. For all other lanes the value set previously is retained.
49.2.13.2.3	R_TYPE_NEXT	**Prescient** end of packet check **function**. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.
55.3.6.2.4	R_TYPE_NEXT	**Prescient** end of packet check **function**. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.
55.3.6.2.4	T_TYPE_NEXT	**Prescient** end of packet check **function**. It returns the FRAME_TYPE of the tx_raw vector immediately following the current tx_raw vector.
65.2.3.4.5	check_ahead_tx	**Prescient function** used by the FEC Transmit process to find the Start_of_Packet in order to replace the Start_of_Packet and its two preceding IDLE ordered sets with /S_FEC/.
65.2.3.4.5	check_ahead_rx	**Prescient function** used by the FEC Receive process to find the /S_FEC/ and /T_FEC/, with fewer than d/2 errors.
82.2.19.2.3	R_TYPE_NEXT	This **function** classifies the 66-bit rx_coded vector that immediately follows the current rx_coded<65:0> vector as belonging to one of the five types defined in R_TYPE, depending on its contents. It is intended to perform a **prescient** end of packet check. The classification results are returned via the r_block_type_next variable.
99.4.7.4	MIN_REMAIN	**Prescient** function to check if enough octets of the current pMAC packet remain meet the minimum fragment requirement after preemption. Produces a Boolean value as follows: TRUE >= minFrag octets are left to transmit FALSE Otherwise
99.4.7.4	RX_MCRC_CK	**Prescient function** returning a Boolean value. The value is TRUE if rPLS_DATA_VALID.indication with a value of DATA_NOT_VALID will be received after the next 32 rPLS_DATA.indication primitives and the next 32 rPLS_DATA.indications equal the computed mCRC result for the preemptable packet being received. It is FALSE otherwise.
99.4.7.4	SFD_DET	**Prescient function** returning a Boolean value. The value is TRUE if an 8-bit vector produced from the next eight pPLS_DATA.request primitives contains an SFD.
113.3.6.2.4	R_TYPE_NEXT	**Prescient** end of packet check **function**. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.
113.3.6.2.4	T_TYPE_NEXT	**Prescient** end of packet check **function**. It returns the FRAME_TYPE of the tx_raw vector immediately following the current tx_raw vector.
119.2.6.2.3	R_TYPE_NEXT	This **function** classifies the 66-bit rx_coded vector that immediately follows the current rx_coded<65:0> vector as belonging to one of the five types defined in R_TYPE, depending on its contents. It is intended to perform a **prescient** end of packet check. The classification results are returned via the r_block_type_next variable.
126.3.6.2.4	R_TYPE_NEXT	**Prescient** end of packet check **function**. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.

Note: One stylistic method for describing look-ahead in an IEEE 802 Standard is via “prescient functions”, as found in IEEE Std 802.3-2018.

A Different View of the Stack (and Upside Down)



Note: Transmit path omitted.

IEEE Std 802.1DU-20xx Reference	Name	Direction	Layer	Media Type
<<TBS>>	**CTF_UNITDATA_NEXT. indication**	**Receive**	**CTF Layer**	*<<TBS>>*

IEEE Std 802.3-2018 Reference	Name	Direction	Layer	Media Type
99.4.7.4	RX_MCRC_CK	Receive	MAC MERGE	Generic

IEEE Std 802.3-2018 Reference	Name	Direction	Layer	Media Type
65.2.3.4.5	check_ahead_rx	Receive	RECONCILIATION	1000BASE-X

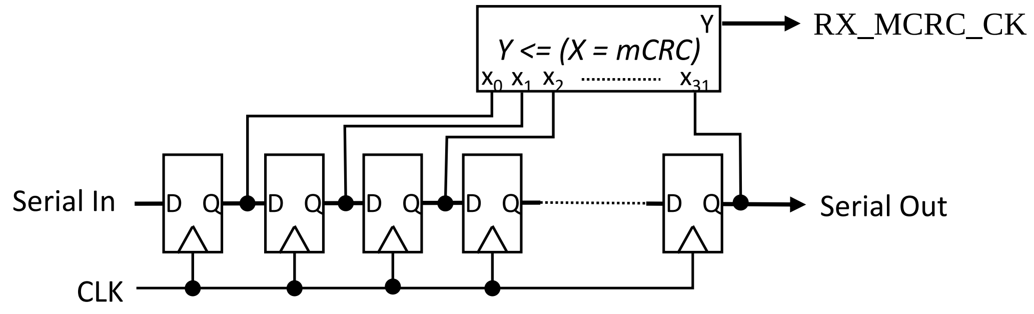
IEEE Std 802.3-2018 Reference	Name	Direction	Layer	Media Type
36.2.5.1.4	check_end	Receive	PCS	1000BASE-X
48.2.6.1.4	check_end	Receive	PCS	10GBASE-X
49.2.13.2.3	R_TYPE_NEXT	Receive	PCS	10GBASE-R
55.3.6.2.4	R_TYPE_NEXT	Receive	PCS	10GBASE-T
82.2.19.2.3	R_TYPE_NEXT	Receive	PCS	40GBASE-R and 100GBASE-R
113.3.6.2.4	R_TYPE_NEXT	Receive	PCS	25GBASE-T
119.2.6.2.3	R_TYPE_NEXT	Receive	PCS	200GBASE-R and 400GBASE-R
126.3.6.2.4	R_TYPE_NEXT	Receive	PCS	2.5GBASE-T and 5GBASE-T

Examples on different layers

Example from the MAC Merge Layer

Name	Description
RX_MCRC_CK	**Prescient function** returning a Boolean value. The value is TRUE if rPLS_DATA_VALID.indication with a value of DATA_NOT_VALID will be received after the next 32 rPLS_DATA.indication primitives and the next 32 rPLS_DATA.indications equal the computed mCRC result for the preemptable packet being received. It is FALSE otherwise.

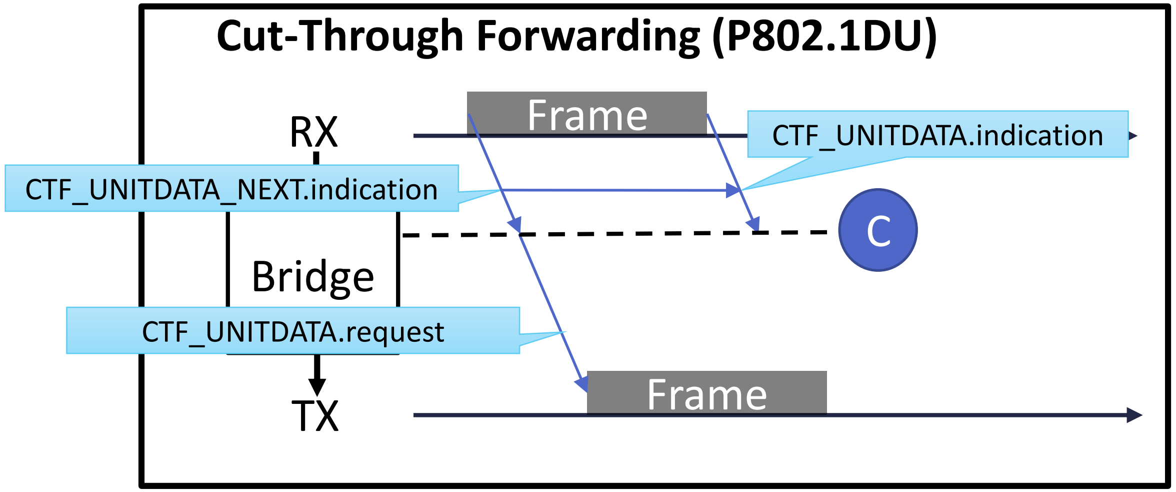
→ Implementers should know what to do:



Example for Cut-Through Forwarding

In absence of interfering transmissions, a CTF_UNITDATA_NEXT.indication results in a CTF_UNITDATA.request invocation at the transmission port after a duration of 64 octet times at the observation plane (8.4.3 of IEEE Std 802.1AS-2020) of the associated reception port.

→ Implementers should know what to do:



Notes:

- Examples are simplified for easier illustration.
- Decisions on technical and editorial of an IEEE 802 Standard developed in P802.1DU are subject to the regular Stds development process.

Summary

- The previous slides illustrated a modelling approach CTF, which assumes an idealized modelling world, further away from implementation realities. The modelling approach in https://www.ieee802.org/1/files/public/docs2021/new-specht-ctf-802-1-1121-v01.pdf is closer to implementation realities.
- Either of both modelling approaches can be used by WG 802.1 to specify the identical external visible behavior of CTF bridges.
- The two modelling approaches do not stand into competition. Instead, they demonstrate a spectrum of options for Stds development.

Thank You for Your Attention!

Questions,
Comments,
Opinions,
Ideas?