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.)
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.indication 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.

- However:
  - 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

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 [link to document].
Linearized View

1. LAN
   - Serial RX Data

2. PHY & MAC
   - MA_DATA.indication(DA, SA, …)
   - Convergence functions

3. MAC Relay Entity
   - M_UNITDATA.indication(DA, SA, …)
   - Convergence functions

4. MAC Relay Entity
   - M_UNITDATA.request(DA, SA, …)
   - Convergence functions

5. MAC & PHY
   - MA_DATA.request(DA, SA, …)

6. Serial TX Data
   - LAN

Observation plane known in a Standard
(8.4.3 of IEEE Std 802.1AS-2020)

Atomic and instantaneous!
Event at End of frame
Atomic and instantaneous!
Event at Start of frame

Duration known in a Standard
(12.31.3.4 of IEEE Std 802.1Qcr-2020)
Store-and-Forward vs. Cut-Through Forwarding

**Store-and-Forward (published IEEE 802 Stds)**

1. **LAN**
   - Serial RX Data
2. **PHY & MAC**
   - MA_DATA.indication(DA,SA,...)
3. Convergence functions
4. **MAC Relay Entity**
   - M_UNITDATA.indication(DA,SA,...)
5. Convergence functions
6. **MAC & PHY**
   - M_UNITDATA.request(DA,SA,...)
7. **Serial TX Data**

**Event at End of frame**
- Atomic and instantaneous!

**Event at Start of frame**
- Atomic and instantaneous!

**Cut-Through Forwarding (P802.1DU)**

1. **LAN**
2. **PHY & MAC**
3. **MAC & PHY**
4. **Serial RX Data**
5. **MAC Relay Entity**
6. **Serial TX Data**

**Event at End of frame**
- Atomic and instantaneous!

**Event at Start of frame**
- Atomic and instantaneous!

---

An Idealistic Model for P802.1DU

Johannes Specht

13-Apr-22
Layering
Layering/Baggy-Pants Diagram

IEEE Std 802.1Q
IEEE Std 802.1AC
(Preemption does not matter in this slide set)
IEEE Std 802.3-2018, and others MACs
An Idealistic Model for P802.1DU

Observation plane known by Standard
(8.4.3 of IEEE Std 802.1AS-2020)

Summarized Description

• Operates on the sequence of M_UNITDATA.indication primitive invocations on the ISS
• Issues CTF_UNITDATA.indication and CTF_UNITDATA_NEX.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_NEX.indication before the associated CTF_UNITDATA.indication invocation
  • Time difference between CTF_UNITDATA.indication and CTF_UNITDATA_NEX.indication pair is basically the frame length, measured at the observation plane
• Having both (CTF_UNITDATA.indication and associated CTF_UNITDATA_NEX.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_NEX.indication …”)

1. Serial RX Data
2. MA_DATA.indication(DA,SA,...)
3. M_UNITDATA.indication(DA,SA,...)
4. M_UNITDATA.request(DA,SA,...)
5. MA_DATA.request(DA,SA,...)
6. Serial TX Data

13-Apr-22
Johannes Specht
An Idealistic Model for P802.1DU

Linearized View - Simplified Illustration

1. LAN
   - Serial RX Data
2. PHY & MAC
   - MA_DATA.indication(DA,SA,...)
3. Convergence functions
   - M_UNITDATA.indication(DA,SA,...)
4. CTF Layer
   - CTF_UNITDATA.indication(DA,SA,...)
   - CTF_UNITDATA_NEXT.indication(DA,SA,...)
5. MAC Relay Entity
   - CTF_UNITDATA.request(DA,SA,...)
6. LAN

Event at Start of frame:
- Atomic and instantaneous!

Event at End of frame:
- Atomic and instantaneous!
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 $\approx$ no perceivable progress in time
    $\rightarrow$ 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

<table>
<thead>
<tr>
<th>IEEE Std 802.3-2018 Reference</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.2.5.1.4</td>
<td>check_end</td>
<td>Prescient End_of_Packet and Carrier_Extend function used by the PCS Receive process to set RX_ER and RXD&lt;7:0&gt; signals. The check_end function returns the current and next two codegroups in rx_code_group&lt;9:0&gt;.</td>
</tr>
<tr>
<td>48.2.6.1.4</td>
<td>check_end</td>
<td>Prescient Terminate function used by the PCS Receive process to set the RXD&lt;31:0&gt; and RXC&lt;3:0&gt; signals to indicate Error if a running disparity error was propagated to any Idle codegroups 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 [[T]], 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 [[T]], 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.</td>
</tr>
<tr>
<td>49.2.13.2.3</td>
<td>R_TYPE_NEXT</td>
<td>Prescient end of packet check function. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.</td>
</tr>
<tr>
<td>55.3.6.2.4</td>
<td>R_TYPE_NEXT</td>
<td>Prescient end of packet check function. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.</td>
</tr>
<tr>
<td>55.3.6.2.4</td>
<td>T_TYPE_NEXT</td>
<td>Prescient end of packet check function. It returns the FRAME_TYPE of the tx_raw vector immediately following the current tx_raw vector.</td>
</tr>
<tr>
<td>65.2.3.4.5</td>
<td>check_ahead_rx</td>
<td>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/.</td>
</tr>
<tr>
<td>99.4.7.4</td>
<td>MIN_REMAIN</td>
<td>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 =&gt; minFrag octets are left to transmit FALSE Otherwise</td>
</tr>
<tr>
<td>99.4.7.4</td>
<td>RX_MCRC_CS</td>
<td>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 PLS_DATA.indication primitives and the next 32 PLS_DATA.indications equal the computed mCRC result for the preemptable packet being received. It is FALSE otherwise.</td>
</tr>
<tr>
<td>99.4.7.4</td>
<td>SPD_DET</td>
<td>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 SPD.</td>
</tr>
<tr>
<td>113.3.6.2.4</td>
<td>R_TYPE_NEXT</td>
<td>Prescient end of packet check function. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.</td>
</tr>
<tr>
<td>113.3.6.2.4</td>
<td>T_TYPE_NEXT</td>
<td>Prescient end of packet check function. It returns the FRAME_TYPE of the tx_raw vector immediately following the current tx_raw vector.</td>
</tr>
<tr>
<td>119.2.6.2.3</td>
<td>R_TYPE_NEXT</td>
<td>This function classifies the 66-bit rx_coded vector that immediately follows the current rx_coded&lt;65:0&gt; 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.</td>
</tr>
<tr>
<td>125.3.6.2.4</td>
<td>R_TYPE_NEXT</td>
<td>Prescient end of packet check function. It returns the R_BLOCK_TYPE of the rx_coded vector immediately following the current rx_coded vector.</td>
</tr>
</tbody>
</table>

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)

#### IEEE Std 802.1DU-20xx

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Layer</th>
<th>Media Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTF_UNITDATA_NEXT</td>
<td>Receive</td>
<td>CTF Layer</td>
<td>&lt;&lt;TBS&gt;&gt;</td>
</tr>
</tbody>
</table>

#### IEEE Std 802.3-2018

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Layer</th>
<th>Media Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX_MCRC_CK</td>
<td>Receive</td>
<td>MAC MERGE</td>
<td>Generic</td>
</tr>
<tr>
<td>check_ahead_rx</td>
<td>Receive</td>
<td>RECONCILIATION</td>
<td>1000BASE-X</td>
</tr>
<tr>
<td>check_end</td>
<td>Receive</td>
<td>PCS</td>
<td>1000BASE-X</td>
</tr>
<tr>
<td>check_end</td>
<td>Receive</td>
<td>PCS</td>
<td>10GBASE-X</td>
</tr>
<tr>
<td>R_TYPE_NEXT</td>
<td>Receive</td>
<td>PCS</td>
<td>10GBASE-T</td>
</tr>
<tr>
<td>R_TYPE_NEXT</td>
<td>Receive</td>
<td>PCS</td>
<td>40GBASE-R and 100GBASE-R</td>
</tr>
<tr>
<td>R_TYPE_NEXT</td>
<td>Receive</td>
<td>PCS</td>
<td>25GBASE-T</td>
</tr>
<tr>
<td>R_TYPE_NEXT</td>
<td>Receive</td>
<td>PCS</td>
<td>200GBASE-R and 400GBASE-R</td>
</tr>
<tr>
<td>R_TYPE_NEXT</td>
<td>Receive</td>
<td>PCS</td>
<td>2.5GBASE-T and 5GBASE-T</td>
</tr>
</tbody>
</table>

**Note:** Transmit path omitted.
### Example from the MAC Merge Layer

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX_MCRC_CK</td>
<td><strong>Prescient function</strong> 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.</td>
</tr>
</tbody>
</table>

→ **Implementers should know what to do:**

![Diagram](image.png)

**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.

### 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:**

![Diagram](image2.png)

**Notes:**
- 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?