PTP Timestamping on Ethernet Interfaces: Accounting for Transmitter Skew

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

SMART | CONNECTED | SECURE

Andras de Koos

September, 2020

Problem (1) – No Mention of Transmitter Skew

There is no explicit mention of the effect of transmitter skew for PTP timestamping.

802.3-2018 Clause 90.7 does mention skew at the receiver:

The receive path data delay for a multi-lane PHY is reported as if the beginning of the SFD arrived at the MDI input on the lane with the smallest buffer delay

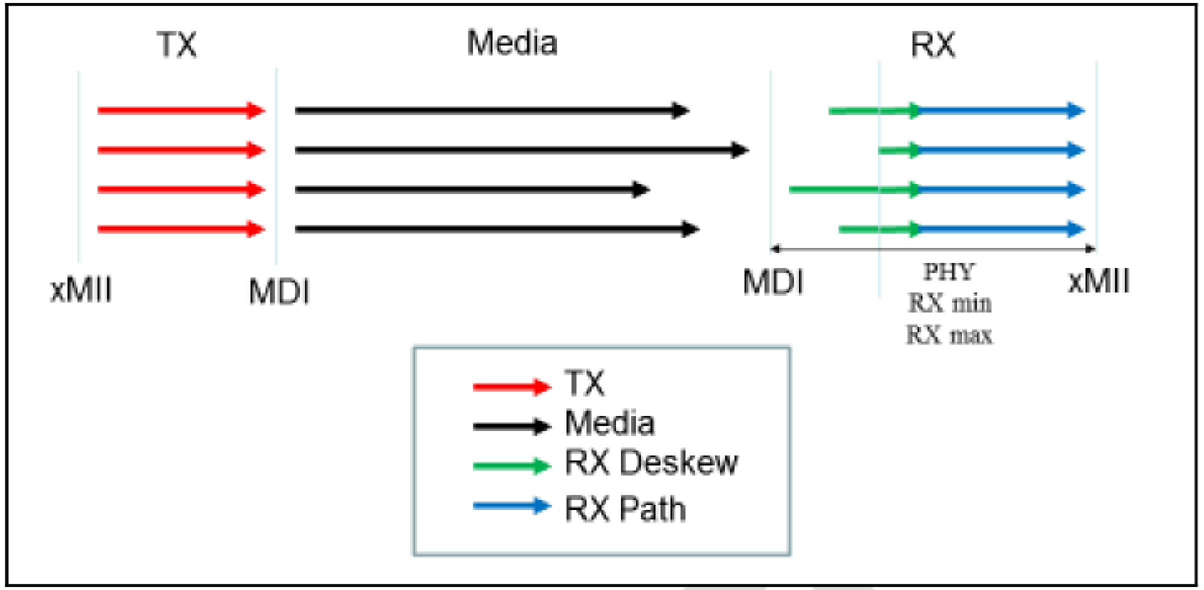


Figure source : IEEE 802.3 Ethernet Working Group Liaison Communication: Liaison response to ITU-T Study Group 15 on timestamping point for multilane Ethernet interfaces

I.e. The Rx timestamp happens as if all lanes were aligned to the *last arriving lane*.

This has the effect of "pushing" the skew into the medium – i.e. timestamp as if all lanes of the medium have latency equal to that of the lane with the largest latency.

But the scheme seems to assume *zero* skew at the transmitter.

Problem (2) – Tx skew is perfectly acceptable

802.3 Does allow for lane skew to be introduced by the transmitter.

For example, a 100GE can have up to 29ns (!) of skew at the transmitter (See 802.3-2018 Clause 80):

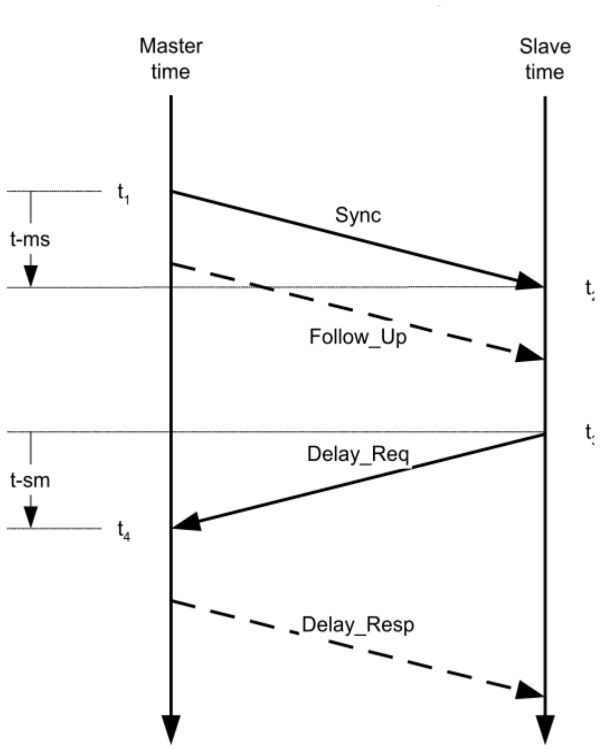
Table 80–6—Summary of Skew constraints

Skew points	Maximum Skew (ns)[a]	Maximum Skew for 40GBASE-R PCS lane (UI)[b]	Maximum Skew for 100GBASE-R PCS lane (UI)[c]	Notes[d]
SP0	29	N/A	≈ 150	See 83.5.3.1
SP1	29	≈ 299	≈ 150	See 83.5.3.2

MICROCHIP

Problem (3) – Tx skew affects time sync accuracy

How the transmitter accounts for the skew affects the accuracy of the time synchronization



If the Transmitter timestamps as if the frame start occurred on the *first departing lane*, then the transmitter skew is "pushed" into the medium.

If the Transmitter timestamps as if the frame start occurred on the *last departing lane*, then the transmitter skew is accounted for in the transmit timestamp.

Which one is correct? If the departure times t_1 and t_3 are interpreted different ways, or if the Tx skew at the master is different from the Tx skew at the slave, then t-ms and t-sm will have an error.

<u>Without a proper way to incorporate Tx skew into the departure timestamps, the result is a time synchronization error.</u>

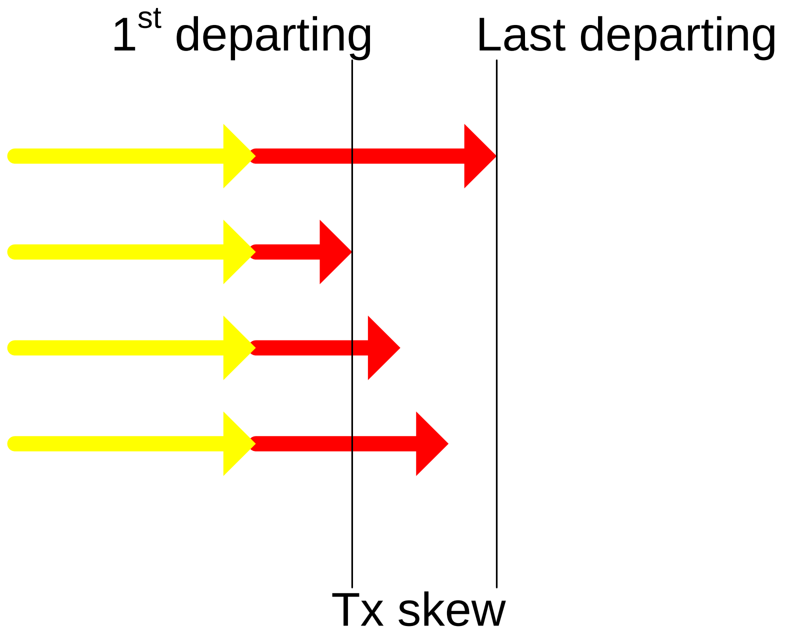
MICROCHIP

Problem (4) – Tx Skew may not be deterministic

If transmitter skew is known a priori, then it can be compensated for. It would cancel out if it is the same on either end, or otherwise show up as a known asymmetry.

But transmitter skew may have a non-deterministic component due to implementation. Lanes can have differing latency due to:

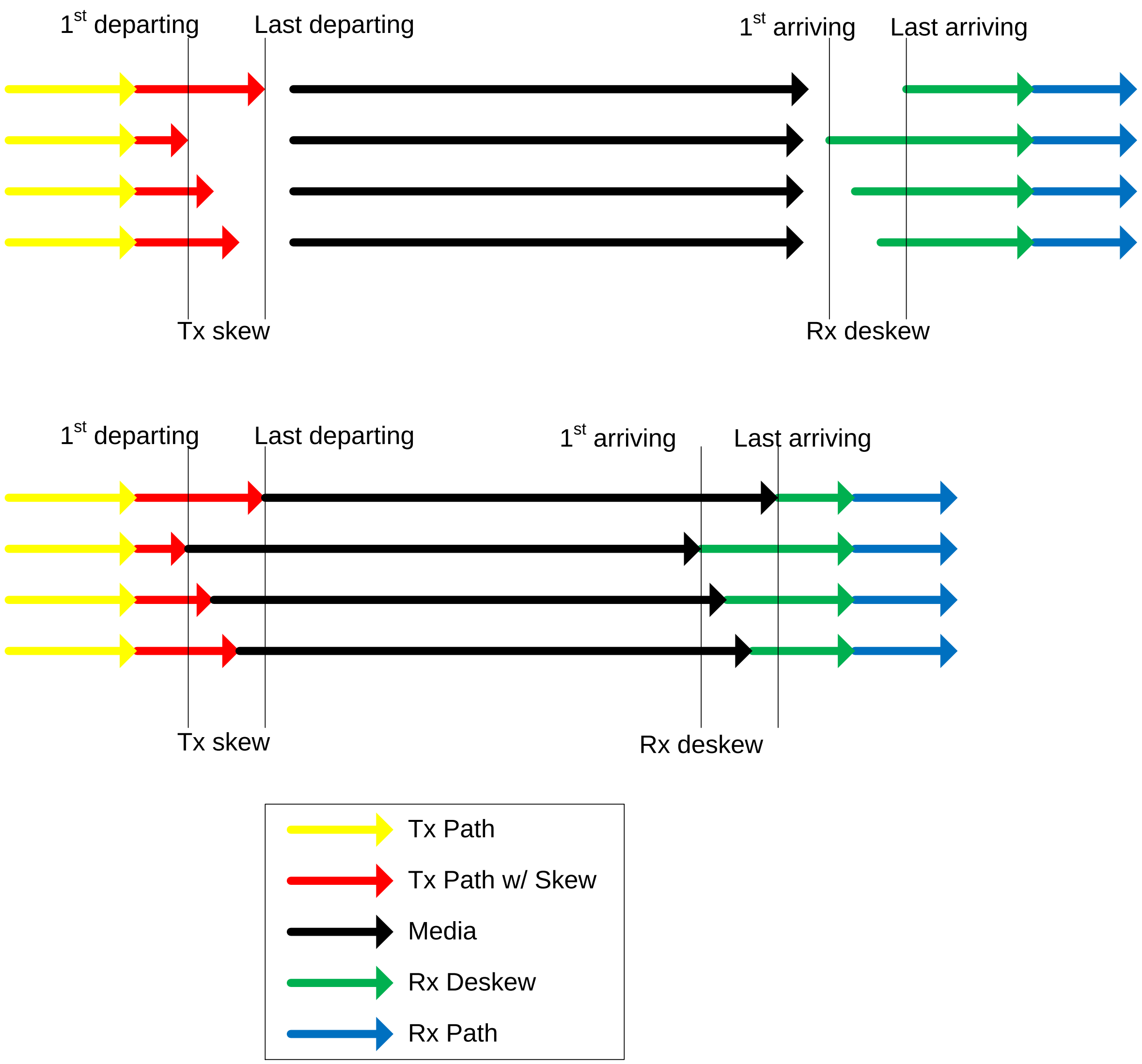
- FIFOs that have different startup conditions
- Individual Clock domain crossings
- Arbitrary SerDes clock phases between lanes



Or even more basically, different equipment might have different Tx skew (deterministic and non-deterministic).

Should the apparent latency of the medium
[(t2-t1)+(t4-t3)]/2
change based on the equipment performing the timestamp?

Illustration (1) – Zero skew on the medium



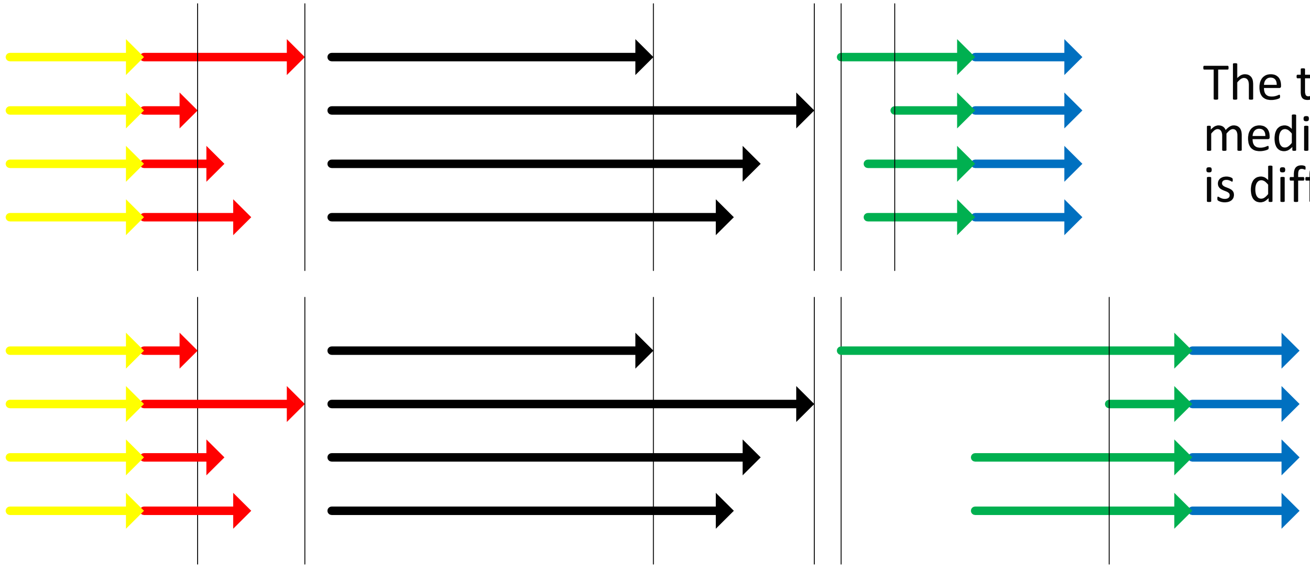
**The two figures are equivalent - just whitespace removed

In this case, the transmitter skew appears as-is at the receiver.

It is clear that if the receiver timestamps the last arriving lane, then the transmitter should timestamp the ***last departing lane.***

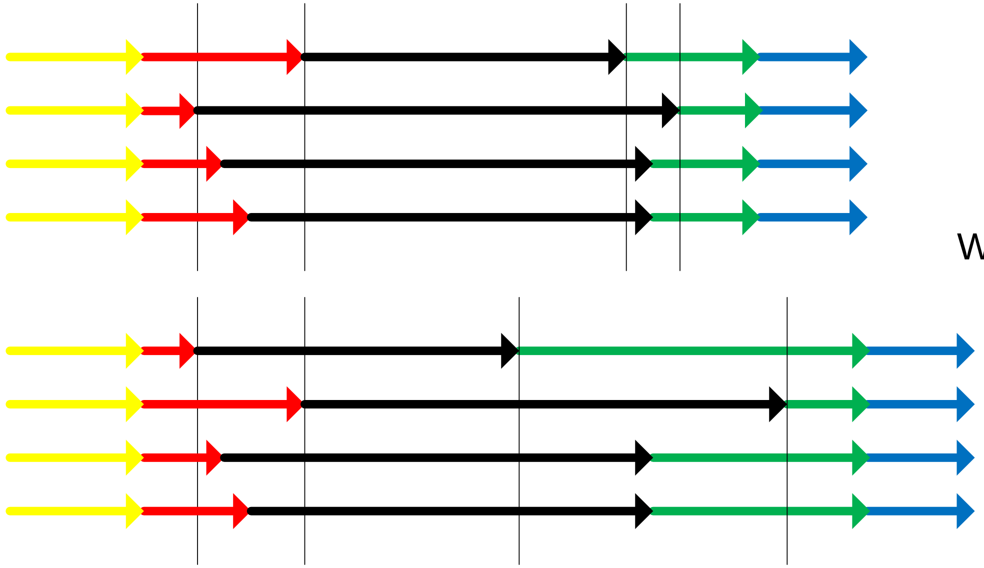
This way, the apparent latency of the medium is equal to the actual latency of the medium. No asymmetry or incorrect link delay is introduced by the Tx skew.

Illustration (2) – Random media skew



The two figures show the same Tx skew, and the same media skew, but with the Tx lanes re-arranged. The result is different final skew and different total lane latencies.

Transmitter skew in series with media skew may be additive, or may cancel out. In figures with the whitespace removed, note how the times of the last arriving lanes are very different.



WHITESPACE REMOVED

For example, if the transmitter has skew of 20ns, while the medium has skew of 100ns, then the maximum latency (and the skew) are not known. Final skew could be anything from 80ns to 120ns!

NO SIMPLE SOLUTION

Ideas Toward a Solution

The lazy solution might be to specify that for any node performing timestamping, its transmitter skew must be zero. Any deviation from zero may show up as a timestamp error.

- The lazy solution is not all bad.
- Altering long-established skew limits in 802.3 would not be simple.

A slightly improved solution would be to timestamp both at the receiver and the transmitter according to the *midpoint* of the lane latencies.

- Works in some contrived cases, but not generally.
- Would mean altering the current recommendation in 802.3 Clause 90.7, however.

A complete solution may require lane-by-lane accounting of the latencies, and the contributions of the transmitter and medium.

- Would make the standards much less straightforward
- Would make any implementation much more complex - “last-arriving” is simple.

Thank You