Clock Reference Stream Improvements Proposal Part 2: Next Level of Detail

Frank Bähren, Intel

June 2020

Issues with Current CRF Definition (1)

- PTP-related time stamps are absolute, but media clock is incremental
  - Have to “count” CRF PDUs and the number of contained time stamps, then multiply with timestamp_interval field to determine events elapsed
  - As opposed to the statement in 10.1

CRF is tolerant of lost packets since clocks free-wheel between defined clock points

    a lost packet means I don’t know how many events have elapsed, I am thus unable to maintain relation between media clock and PTP clock

  - I could interpolate from the latest time stamp and/or guess from seq_num how many PDUs/events I have missed, but that’s more of a heuristic, and it is not defined anywhere.

Issues with Current CRF Definition (2)

- Media Clock is represented in events that, in turn, represent elapsed media time
  - Several specific media event types defined (audio samples, video lines, video pixels, etc.) but at the end of the day, they all represent **media clock time**.
- Why not introducing a generic “media clock” type that can be used for everything?

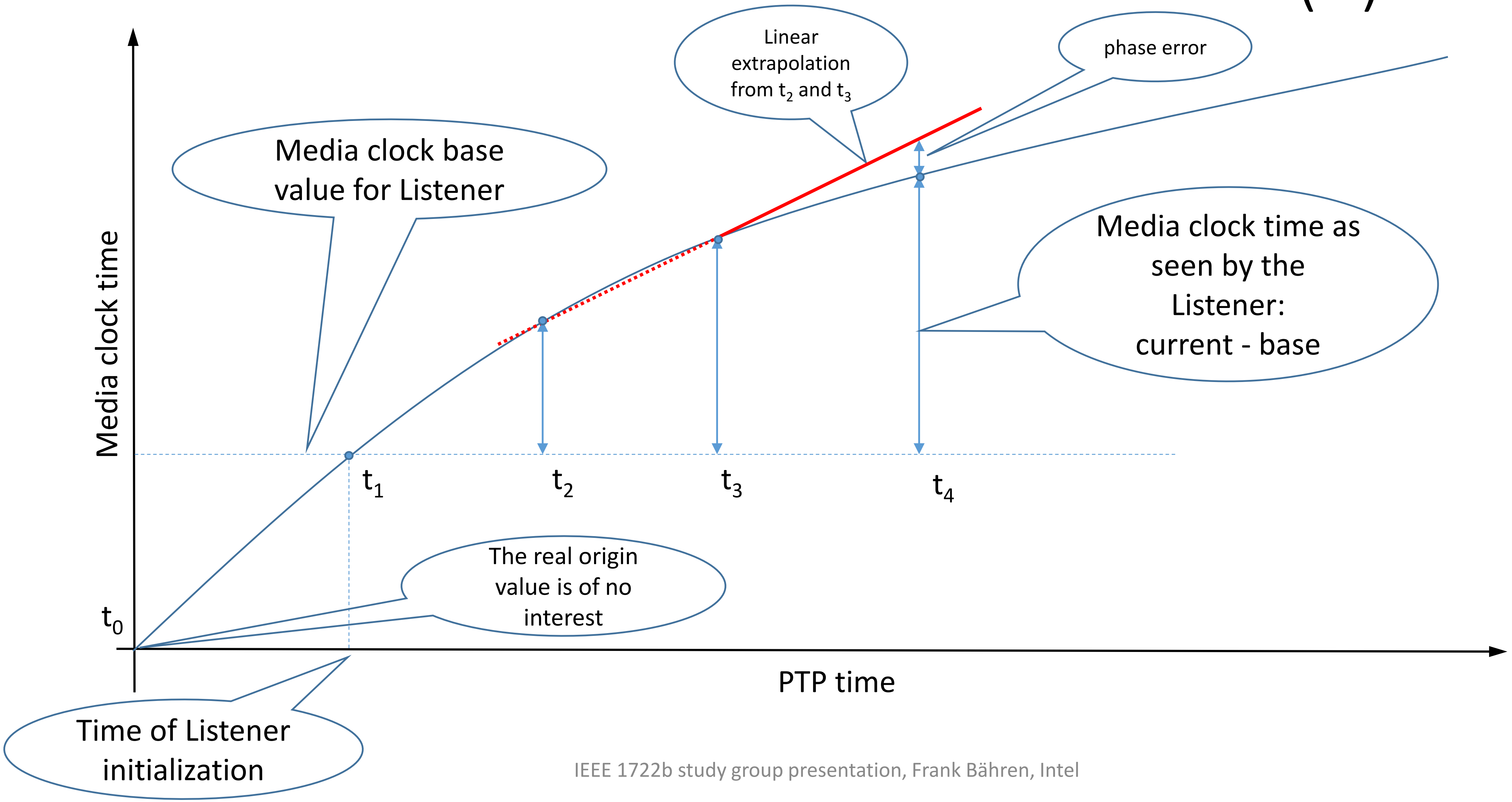
Introduction to Generic Media Time Type

- We still have “events”, but they are just time ticks
- Event durations that have an infinite fractional part can cause accumulating rounding errors, e.g. 48kHz → 20.8$\overline{3}$µs
  Need to express the time span of an event in a fractional way, i.e. numerator and denominator, ideally in the irreducible form.
- There is often still an underlying idea of what the “event” represents (e.g. an audio sample interval), but it only shows in the event granularity, and not in a denoted “type”
- Often, but not always, an event granularity can be found that allows the use of the same media clock for multiple purposes (e.g. audio and video synchronization)

Introduction to Absolute Media Time (1)

- Instead of having only PTP-related time stamps in a CRF PDU, have one or more tuples of (PTP time stamp, media time stamp), both in absolute time
- Upon initialization/reset, CRF Listeners take first received media time stamp and pin this as local reference, only looking at the subsequent increments → The origin (t_0 where media time value = 0) doesn’t really matter
- You can ignore/miss multiple intermediate PDUs and are still able to maintain PTP<->media clock relation
  - Maximum time span between PDUs that have to be received & processed is determined by phase error requirement
- Might require epoch version field rather than simple MR toggle

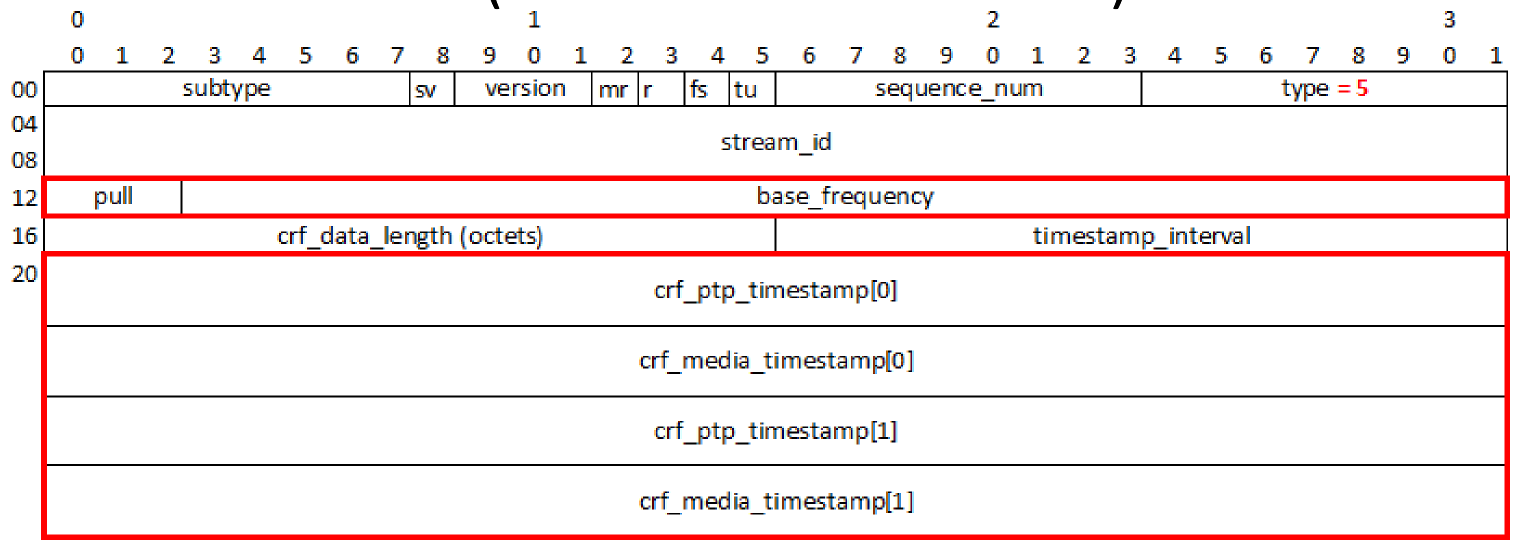
Introduction to Absolute Media Time (2)


Linear extrapolation from t_2 and t_3
phase error
Media clock base value for Listener
Media clock time as seen by the Listener: current - base
Media clock time
t_1
t_2
t_3
t_4
t_0
The real origin value is of no interest
PTP time
Time of Listener initialization

CRF Extension Proposal (First Shot)

- New CRF type: 5 = absolute media time
- crf_data contains pairs of PTP timestamps and media timestamps
- pull and base_frequency combined to one field (pull are the three MSB) that holds the divisor part of the event duration (to scale down to 1e-9)
- timestamp_interval holds the dividend part of the event duration

$$t_{event} = \frac{time\_interval}{(pull \cdot 2^{29}) + base\_frequency} \quad (s)$$



subtype | sv | version | mr | r | fs | tu | sequence_num | type = 5

stream_id

pull | base_frequency

crf_data_length (octets) | timestamp_interval

crf_ptp_timestamp[0]

crf_media_timestamp[0]

crf_ptp_timestamp[1]

crf_media_timestamp[1]

- The original meaning of timestamp_interval is meaningless for this kind of CRF, as there doesn't even have to be a fixed interval between timestamps