MPCP Timestamp Jitter with Idle Deletion / Insertion

Marek Hajduczenia (marek.hajduczenia@zte.com.cn)

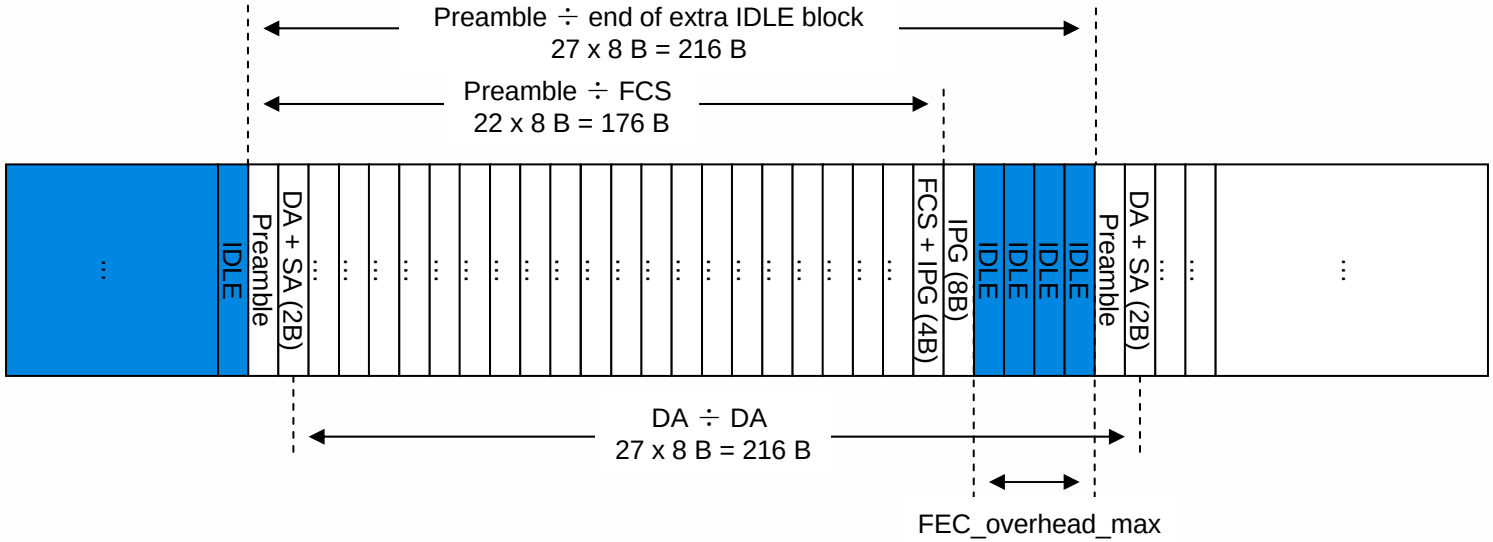
ZTE Corporation

Observed issues with MPCP timestamp jitter

- Status for D2.0
  - FEC overhead is accounted for using FEC_overhead_max function, which overestimates FEC overhead at PCS sublayer;
- Actual behaviour of MPCP timestamp jitter:
  - Despite detailed analysis of state diagrams / their operation, there is still significant MPCP timestamp jitter observed for certain sizes of frames;
  - MPCP timestamp jitter seems to become frame-size dependent (worst situation possible)
- Desired outcome from this meeting:
  - Examine the potential sources of MPCP timestamp jitter
  - Attempt to eliminate them or at least make MPCP timestamp jitter independent from frame size

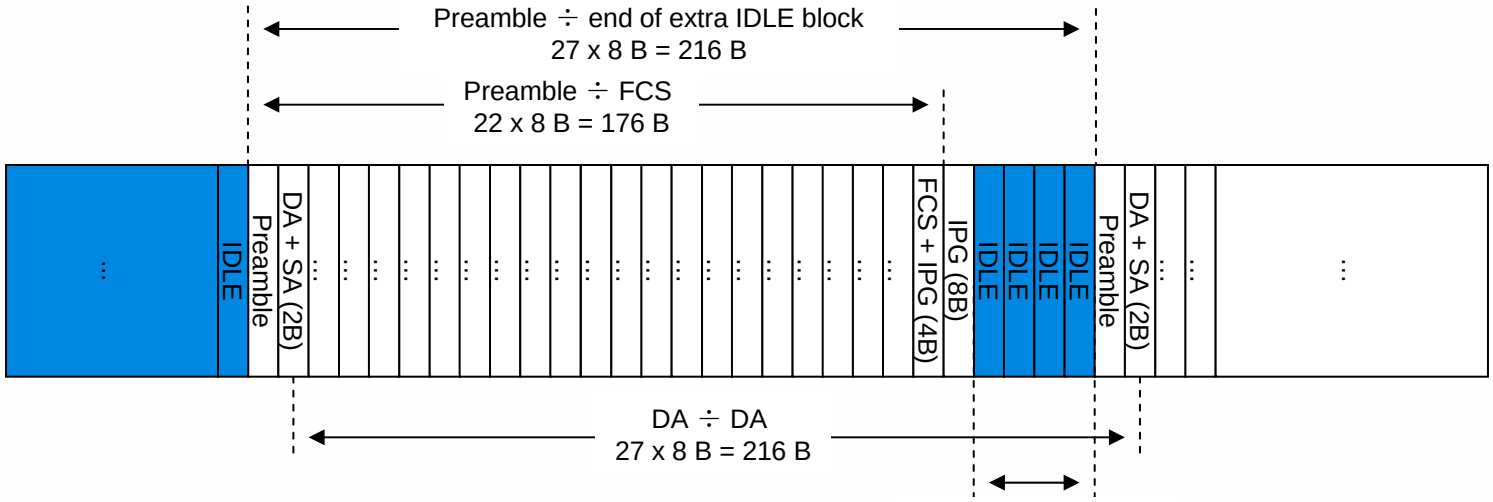
Hypothetical scenario [1]

- Imagine a frame of size 22 x 8 bytes (176 bytes with framing) followed by another frame of this size;
- 12 bytes of IPG (assuming balanced DIC) and 32 bytes of parity will be added (result of FEC_overhead_max function call on sizeof(data_tx) + tailGuard)
- Total size of a block with extra IDLEs = 27 x 8 B = 216 B
- DA ÷ DA distance is 27 x 8 B = 216 B



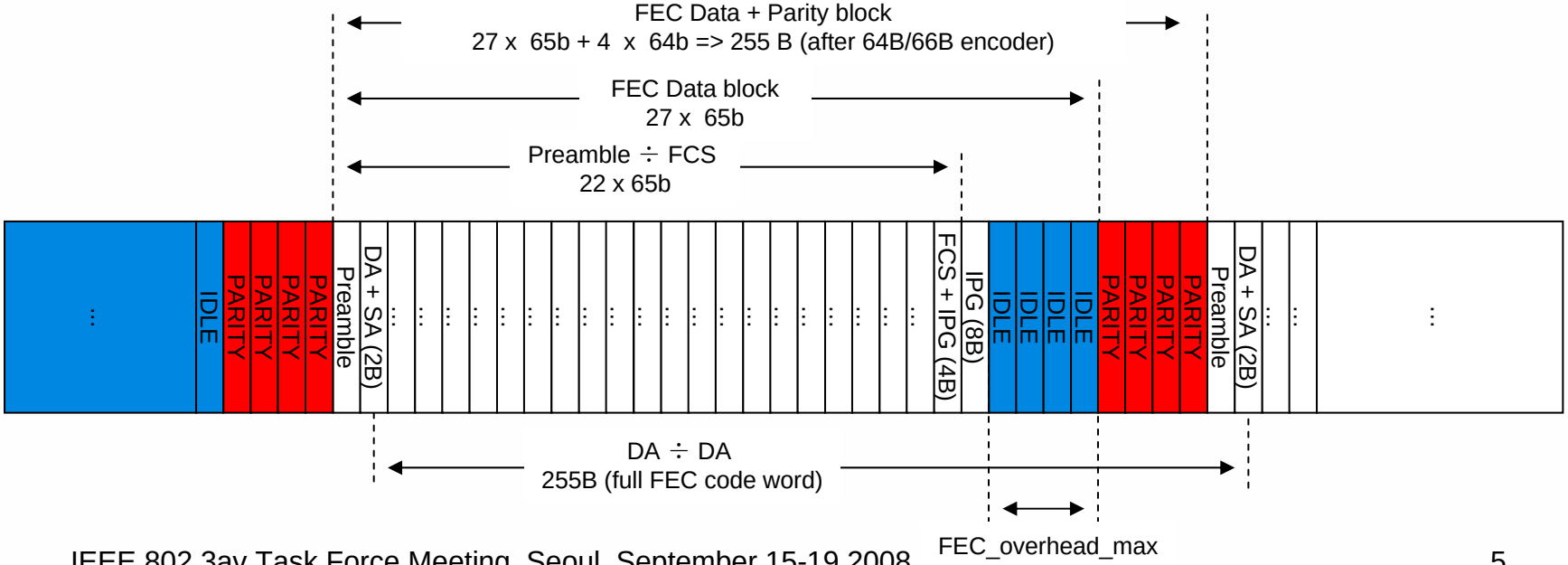
Hypothetical scenario [2]

- What happens at Idle Deletion function:
  - VectorCount and DelCount are 0 at the first preamble word (assume last packet ended clean and variables were reset after its transmission)
  - VectorCount = FEC_DSize at the word right before the 2nd preamble, DelCount is set to FEC_PSize (4) only after transmission of FEC_DSize vectors;
  - The next clock is the 2nd preamble word (not (C+E)), the DelCount cannot be deducted here.
  - Idle deletion is delayed to the end of the next packet.
  - After idle deletion, the distance of the two DAs is still 27 x 8 B



Hypothetical scenario [3]

- What happens at FEC Encoder function:
  - An FEC encoding boundary starts right at the first preamble word;
  - The 4 words reserved as FEC parity by FEC_overhead_max are not removed and are counted as the FEC data words;
  - Before the 2nd preamble word, 4 x 8 B blocks of FEC parity data need to be added - the distance between the two consecutive DAs becomes 31 words.



Final thoughts

- Data passing FEC decoder should always have fixed delay
- Delay should also be independent from packet size
- The current Idle Deletion state diagram does not perform correctly under for packets of size 22 XGMII transfer columns (and smaller)
  - Occurs when bit 1 of preamble is aligned with start of FEC word
  - Extra IDLEs for FEC parity are not deleted correctly, forcing a data shift / loss at the FEC encoder (data might be overwritten)
  - Needs more testing under longer runs of constant packet sizes (might be important for CBR applications, where packet sizes are constant e.g. VoIP, certain VOD with specific codecs etc.)
- Up to 4 * 6.4 ns = 25.6 ns can be introduced for RTT from one side of the link to the other. Is it considered ok for the RTT drift budget ?
- A fix seems to be simple – initialize DelCount with FEC_PSize (currently not initialized at all)

ZTE中兴

Thank You for Your attention