Encapsulation Baseline Proposal for EFM Copper

Barry O'Mahony

IEEE 802.3ah Interim Meeting

Vancouver, B.C.

6-9 January, 2003

Intel Research & Development

Current Status

- Why do we need this?
- Reason: polls at New Orleans meeting showed current HDLC baseline will generate lots of NO (TR) votes
  - HDLC variable overhead the killer
- But, discussion served to elicit group's requirements for encapsulation for encapsulation
- This proposal satisfies these requirements

Requirements

- Low, data-independent overhead (~<3%)
- Handle ethernet-sized frames: ~1500 octets
- Compatible with α/β-interface: see October 2002 ITU-T Liaison
- Minimal interframe gap: 'allow frames to be transmitted with a minimal gap between frames (IPG and preamble reconstructed at receiver)
- MTTFPA of ~10^9 to 10^{10} years: given specified BER and error distribution at the DSL error distribution at the DSL α/β-interface
- Quick recovery from errors: recovery from loss of sync lock should be quick, in order to minimize the number of lost frames

Proposal Highlights

- **Fixed-length codewords, similar to 64b/66b**
  - **Satisfies "quick recovery from errors"**
- **Length = 65 *bytes***
  - **Compatible with DSL α/β-interface**
- **Additional CRC added for each frame**
  - **Robustness compatible with error characteristics in DSL**
- **Small interframe gap**
  - **1 byte minimum**

Benefits

- Low, fixed overhead
  - 1.125% + 2 bytes per Ethernet frame + CRC
  - Compares to 0-100% + 3 bytes + CRC for current HDLC PTM-TC
    - Same range even with scrambler (intentional “malicious” high-overhead frames may still be generated)
- Synchronization survives a corrupted sync byte
  - In HDLC, single byte error can cause loss of sync;
  - In G.gfp, single corrupted header triggers complex sync hunt process
- Minimal, data-independent interpacket gap (1 byte)
  - No need for “slop” to allow for variable encapsulation overhead

Codeword Formats

- **Overhead of 1.125% - half that of 64b/66b**
- **C_i, , i=0 to 64, code values in control codewords**

Type	Frame Data	Sync Byte	Byte Fields 1-64									
all data	DDDD---DDDD	$0F_{16}$	D_0	D_1	D_2	D_3	D_4	D_5	$\rightarrow$	D_{61}	D_{62}	D_{63}
end of frame	k D's, k=0 to 63	$F0_{16}$	C_k	D_0	D_1	D_2	D_3	$\rightarrow$	D_{k-1}	Z	$\rightarrow$	Z
all idle	ZZZZ---ZZZZ	$F0_{16}$	C_{64}	Z	Z	Z	Z	Z	$\rightarrow$	Z	Z	Z

Encoding Start of Frame

- How?
- A frame may arrive from MAC at MII while an End of Frame codeword is being transmitted
  - At 100 Mbps, MII rate much faster than line rate
- Need ability to insert SOF into codeword once its transmission has started
- Otherwise, must wait for completion of codeword transmission, and beginning of new codeword,
  - May needlessly delay transmission of new frame; decreases encapsulation efficiency

Encoding Start of Frame (*cont'd*)

- **Here's how:**
- **Designate *S* byte value as Start of Frame Marker**
  - ***S* just needs to be distinct from *Z***
  - **Remember, *Z* is not MAC data, it's just idle codeword-fill transmitted when no MAC data is available**

Type	Frame Data	**Sync Byte**	Byte Fields 1-64									
all idle → start of frame	*k* D's, *k*=0 to 62	$F0_{16}$	C_{64}	Z	Z	S	D_0	D_1	→	$D_{k\text{-}3}$	$D_{k\text{-}2}$	$D_{k\text{-}1}$
end of frame → start of frame	1st frame: *k* D's, *k*=0 to 62 2nd frame: *j* D's, *j*=0 to 62-*k*	$F0_{16}$	C_k	D_0	→	$D_{k\text{-}1}$	Z	→	S	D_0	→	$D_{j\text{-}1}$

Error Analysis (1)

- **First, a word about 64b/66b:**
  - **Optical channels modeled as Binary Symmetric Channels (BSCs)**
    - **Bit errors independent**
    - ***N*+1 errors occur alot less frequently than *N* errors**
    - **64b/66b designed to detect 3 or fewer errors, regardless of frame content**
  - **However, DSL α/β-interface looks nothing like this**
    - **Bit errors bursty, e.g., R-S decode errors average a little more than 9 errored errored bytes (for *t*=8)**
    - **Four-bit errors are just as likely to occur in a frame as 1 bit errors**
  - **∴error analysis done for 64b/66b on BSCs not applicable to EFM-Cu**

Error Analysis (2)

- Robustness will depend on devising a scheme detects most errors, rather than immunity to <x errors
- Fortunately, EFM-Cu is not alone in this regard,
- EPON FEC robustness analysis is similar,
- So EFM So EFM-Cu group won’t be the only one proposing this to ‘dot-3.

Error Analysis (3)

- **Some numbers (see Backup)**
  - **Bit error ratio (at α/β-interface) $P_b = 1 \times 10^{-7}$**
  - **Byte error ratio (at α/β-interface) $P_B \cong 2 \times 10^{-7}$**
  - **Byte error ratio (at R-S decoder output) $P_{B'} \cong 1 \times 10^{-7}$**
  - **R-S decode error ratio (decoder error + decoder failure)**
    $P_M \cong 2.8 \times 10^{-6}$
  - **R-S undetectable error ratio (decoder error)**
    $P_E < 6.2 \times 10^{-11}$

Error Analysis (4)

- **Frame Error Ratio**
  - **R-S codewords per Ethernet Frame = 1500/239 = 6.2**
  - **Frame Error Ratio $P_F = 1-(1-P_M)^{6.2} = 1.7 \times 10^{-5}$**
- **Undetected Errored Frames**
  - **Ethernet FCS detects all but one in 2^{32} frame errors**
  - **$P_{FPA} = P_F \times 2^{-32} = 4 \times 10^{-15}$**
  - **10 Mbit/s = 833 frames/s = MTTFPA = 9500 years**
  - **∴Encapsulation must improve this by a factor of 10^6**

Improving Robustness

- Append another CRC to the frame, before encapsulation
  - i.e., the CRC is added per frame, not per codeword
  - Need to use a CRC that provides additional protection beyond that provided by Ethernet CRC
- Existing Ethernet CRC
  - $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$
  - Primitive (no factors)
  - Same as 32-bit HDLC CRC
  - d_{min} = 4 for Ethernet-sized frames (catches all errors of 3 bits or less)

Additional CRC

- **HDLC 16-bit CRC:**

  $$x^{16}+x^{12}+x^5+1 = (x+1)(x^{15}+x^{14}+x^{13}+x^{12}+x^4+x^3+x^2+x+1)$$

- **32-bit CRC32/4 (see ref. [9]):**

  $$x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^6+1 = (x+1)(x^{31}+x^{30}+x^{29}+x^{28}+x^{26}+x^{24}+x^{23}+x^{22}+x^{18}+x^{13}+x^{10}+x^8+x^5+x^4+x^3+x^2+x+1)$$

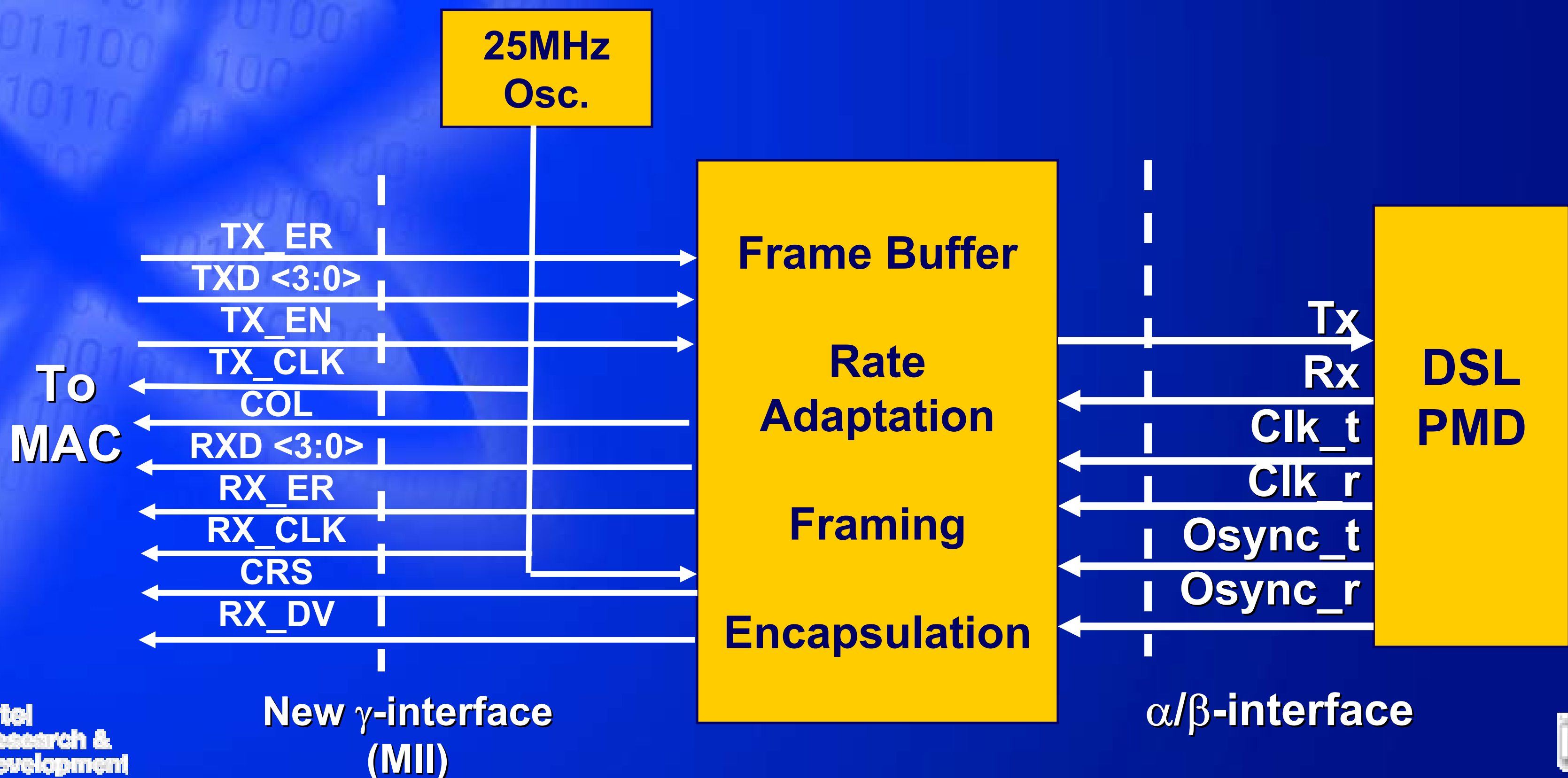
- **Both these detect all errors with odd number of bits**
  - **Since they contain x+1 as a factor**
  - **Neither of these have factors in common with Ethernet CRC**
  - **CRC32/4 has best dmin profile found by [9] for polynomials in the form (x+1)p(x)**
- **Propose that CRC32/4 be used as encapsulation CRC**

Additional CRC (2)

- The 6 dB margin:
  - 10^{-7} DSL BER specified at 6 dB noise margin;
  - i.e., DSL typically operated so that specified BER will be achieved, even if noise level is increased by 6 dB;
  - Ensures performance even in case of "the unexpected", the unmodeled, and transient conditions
- This presentation does not require any of this margin be used to achieve the computed MTTFPA
  - If operators wish to discount the margin, there are better things to "spend" it on rather than detecting bad packets
    - e.g., reach, rate, etc.

Layering and Interfaces

- We're defining a new TPS-TC, so we define a new γ-interface
    - ITU-T Q4/15 said "Go ahead"
- Since this is Ethernet-specific, this could be MII



What about aggregation?

- **Aggregation sublayer is below rate-matching sublayer**
- **Aggregation would generate multiple α/β-interfaces, rather than multiple γ-interfaces**
- **May need to define a second *S* code for aggregation**

Summary

- New framing and encapsulation method that meets all identified requirements is proposed
- This would be a new Ethernet-specific TPS-TC forwarded to ITU-T
- Optionally, γ-interface is simply MII as specified in current baselines

Backup

Error Computations

- P_b specified at 10^{-7}
- Estimate post-R-S decoder scrambler error multiplication at 2×; so BER at R-S output $P_{b'} = 0.5 \times 10^{-7}$
- $P_{B'}$, byte error ratio $\frac{P_{b'}}{P_{B'}} = \frac{2^{8-1}}{2^8 - 1} = \frac{128}{255} \Rightarrow P_{B'} \cong 2 \times P_{b'}$ $P_{B'} = 10^{-7}$ (see ref.[1])
- $P_{B'}$ as a function of pre-R-S byte error ratio (for (255,239) code):

$$P_{B'} \cong \sum_{j=9}^{255} \frac{j}{255} \binom{255}{j} p^j (1-p)^{255-j}$$

$p = 0.00445$; see G.975 and ref. [1]

Error Computations (2)

- P_M, probability of incorrectly decoded codeword:

$$P_M = \sum_{j=9}^{255} \binom{255}{j} p^j (1-p)^{255-j} = 2.8 \times 10^{-6}$$

- If "decoder failure" codewords (i.e., uncorrectable but detectable) are excluded:

$$P_E \le P_M \times 255^{-(255-239)} \sum_{s=0}^{8} \binom{255}{s} 255^s = 2.8 \times 10^{-6} \times 2.2 \times 10^{-5} = 6.2 \times 10^{-11}$$

(see ref. [2])

(this would require change to α/β-interface, however)

References

- [1] Sklar, B; *Digital Communications*, Prentice Hall 2001
- [2] McEliese, R.J., and Swanson, L; *On the Decoder Error Probability for Reed-Solomon Codes*, NASA TDA Progress Report 42-84, 1985
- [3] ITU-T G.975 (10/2000), *Forward Error Correction for Submarine Systems*
- [4] ITU-T G.gfp, *General Framing Procedures*
- [5] ITU-T contribution OJ-075, *Framing and encapsulation considerations for Framing and encapsulation considerations EFM*, Intel Corp., October 2002
- [6] ISO/IEC-3309, *Information technology - Telecommunications and Telecommunications and information exchange between systems - High-level data link control level data link control (HDLC) procedures - Frame structure*
- [7] IETF Internet Draft draft-sheinwald-iscsi-crc-02.txt, *iSCSI CRC Considerations* (2002)
- [8] Communications Statement from ITU-T Q4/15 to 802.3ah, October 2002
- [9] Castagnoli, et al; , et al; *Optimization of Cycle Redundancy-Check Codes with Check Codes with 24 and 32 Parity Bits*, IEEE Transactions on Comm., June 1993