Forward Error Correction (FEC) techniques for optical communications

IEEE 802.3 High-Speed Study Group
Plenary meeting, Montreal
July 1999

Kamran Azadet, Mark Yu
Lucent Technologies
Phone: 732-949-7280
Email: ka@lucent.com
or myu@lucent.com
Outline

- Why FEC?
- Cyclic codes - Overview
- Algorithms and implementation
  - Encoder
  - Decoder
- Performance analysis
- Latency issues
- Summary
Why FEC ?
Why FEC?

- FEC lowers BER by a great deal for low overhead
  - E.g. for RS(255,239) overhead is 6%: For input BER=10^{-4}, output BER=10^{-14}!

- Very low overhead codes exist (less than 0.1%) and have been widely used in other optical communication applications
  - Possible trade-off between performance and overhead (illustrated in this talk)

Overhead

- increased rate -> slightly reduces the overall coding gain
- electronics cost (MUX/DEMUX if line rate is increased)
- adds FEC circuitry (encoder is simpler than decoder)
How to incorporate FEC in 10GbE?

- WAN applications incorporate already FEC, solutions are currently proprietary. There is however an on-going effort to adopt an “in-band” FEC standard in 10G SONET (most likely based on a BCH-3 code)

- There may be ways to exploit already existing overhead in the Ethernet LAN (IPG?)
  - However available overhead may be limited
  - Ethernet packets are variable size -> decoding would be more complicated!

- For increased performance, line rate could be slightly increased
  - fixed length blocks are preferable

- FEC can be combined with constrained codes such as 8b/10b (or 16b/18b), or with zero-overhead scrambled codes.
Cyclic codes - overview
Some FEC definitions

- **Systematic block code**
  - An encoder accepts \( k \) information symbols and appends separately a set of \( r \) redundant symbols (parity bits) derived from the information symbols. Note the information word is not disturbed in any way in the encoder. This code is called \((n, k)\) code, where \( n = k + r \).
  - Since code space is enlarged and codewords are a constrained subset, error detection and correction is possible based on maximum likelihood decoding.

- Hamming distance of two codewords: number of non-zero elements in the codeword formed by their difference
- Hamming distance of a code = \( d \) (is the minimum distance between two codewords)
- Singleton bound: \( d \leq n - k + 1 \)
Cyclic code

- An (n, k) code is cyclic if a cyclic shift of a codeword is also a codeword.
- Associated with a generating polynomial (usually implemented as LFSR)
- Two representations of a codeword $w = (w_0, w_1, \ldots, w_{n-2}, w_{n-1})$ <-> $W(x) = w_0 + w_1 x + \ldots + w_{n-2} x^{n-2} + w_{n-1} x^{n-1}$

Examples of popular systematic cyclic block codes: binary BCH, Reed-Solomon

- Reed-Solomon is famous for its ability to correct “bursty” errors.
Galois fields

- Theory developed by French mathematician E. Galois
- Galois Fields: for a prime $p$, $GF(p)$ denote the integer ring (over $[+,\times]$) modulo $p$
- Example: $GF(2)$: addition is modulo-2 addition / XOR-gate
  multiplication is modulo-2 multiplication / AND-gate
  Simple codes such as Hamming codes or binary BCH codes are built over $GF(2)$.
- Galois Fields definition can be generalized to $GF(p^m)$, for instance $GF(2^8)$, where words are **Bytes** and not bits (this is used in Reed-Solomon codes)
Hamming code

- The well-known \((2^n-1, 2^n-n-1)\) code with Hamming distance of 3 is capable of correcting 1 error
- Error syndrome gives the error location
  - Easy to decode, but only correct one error

- Are there codes able to correct more than one error and easy to decode?
RS code

- RS(n,k) consists of all polynomials of degree at most n that are multiples of the generator polynomial 
  \( g(x) = (x-1)(x-\alpha_1)\ldots(x-\alpha_{2t-1}) \), where \( \alpha \) is a primitive element of GF(\(2^m\))

- Minimum distance between codewords is \( d=n-k=2t+1 \), thus RS(n,k) can correct t symbols

- 2t Erasures can be corrected (an Erasure is an error with known position)
BCH codes

- The cyclic code of length $n$ over $GF(q)$, $n$ coprime with $q$, whose generator polynomial is $g(x)$. RS code is a subclass of BCH code.
- Binary BCH code: a BCH code on $GF(2)$, i.e., $q=2$
- Binary BCH codes are most popular
- BCH codes usually correct few bits in a block (1-error correcting or Hamming, 2-error correcting, 3-error correcting, etc.)
- Encoding and decoding procedures are similar to those of RS code, except there is no need to compute error magnitude for binary BCH code.
Algorithms and implementation
- Encoder -
Send $W(x) = x^{n-k}D(x) - r(x)$, where $D(x)$ is the information symbols, and $r(x)$ is the remainder of $x^{n-k}D(x)$ divided by the generating polynomial $G(x)$.

- $W(x)$ is a codeword, because $G(x)$ divides $W(x)$.
- $W(x)$ is systematic, i.e., redundant symbols are added after information symbols.
RS is a cyclic code and often implemented as a LFSR with GF(2^m) operators.
Algorithms and implementation

- Decoder -
First some more definitions:

- Error word polynomial $e(x) = e_0 + e_1 x + \ldots + e_{n-1} x^{n-1}$
- The received word polynomial is given by $w(x) = c(x) + e(x)$
- Syndromes $S_j = w(\alpha^j) = \sum_{i=0}^{n-1} e_i \alpha^{ij}$ for $j = 0, \ldots, 2t-1$
- Syndromes can be computed using the Horner algorithm:
RS decoding - cont.

- Suppose that there are $r$ errors, $r \leq t$, occurred
  - at locations $i_1, \ldots, i_r$ (let $X_i = \alpha^{i_l}$)
  - with values $e_{i_1}, \ldots, e_{i_r}$ (let $Y_i = e_i \alpha^{i_l} = e_i X_i$)

- Reformulate $S_j = \sum_{i=1}^{r} e_i \alpha^{i_j}$ $\implies$

\[
\begin{bmatrix}
S_0 \\
S_1 \\
\vdots \\
S_{2t-1}
\end{bmatrix} =
\begin{bmatrix}
X_1^0 & X_2^0 & \ldots & X_r^0 \\
X_1^1 & X_2^1 & \ldots & X_r^1 \\
\vdots & \vdots & \ldots & \vdots \\
X_1^{2t-1} & X_2^{2t-1} & \ldots & X_r^{2t-1}
\end{bmatrix}
\begin{bmatrix}
Y_0 \\
Y_1 \\
\vdots \\
Y_r
\end{bmatrix}
\]

2t equations for 2r unknowns, $r \leq t$
RS decoding - cont.

- Define error locator polynomial
  \[ s(x) = (1-xX_1)(1-xX_2)\ldots(1-xX_r) = s_r x^r + \ldots + s_1 x + 1 \]
  - Inverse of zeros of \( s(x) \) gives error locations

- Multiplying both sides with \( Y_i X_i^{j+r} \) and letting \( x = X_i^{-1} \) \( \Rightarrow s(X_i^{-1}) = 0 \), for all \( i,j \), we get
  \[
  \begin{pmatrix}
  S_1 & S_2 & \ldots & S_r \\
  S_2 & S_3 & \ldots & S_{r+1} \\
  \vdots & \vdots & \ddots & \vdots \\
  S_r & S_{r+1} & \ldots & S_{2r-1}
  \end{pmatrix}
  \begin{pmatrix}
  s_r \\
  s_{r-1} \\
  \vdots \\
  s_1
  \end{pmatrix}
  =
  \begin{pmatrix}
  -S_{r+1} \\
  -S_{r+2} \\
  \vdots \\
  -S_{2r}
  \end{pmatrix}
  \]
  \[ M \equiv \]

- \( M \) is singular if \( m > r \) !
  - So we can determine how many errors have occurred
RS decoding steps

1. Compute Syndromes $S_i = w(\alpha^i)$, for $i=1,..,2t$

2. Determine the maximum number $r$ so that $M$ is nonsingular. $r$ is then the number of errors occurred.

3. Find the coefficients of the error-locator polynomial (solving the key equation) by computing

   \[
   \begin{bmatrix}
   S_{r+1} \\
   S_{r+2} \\
   \vdots \\
   S_{2r}
   \end{bmatrix}
   \]

   \[
   M^{-1} \times
   \]

   4. Solve $s(x)=0$

   5. Find error locations

6. Compute error magnitude at error locations and correct the errors
RS decoder architecture

Error Detection

Error Correction

compute the Syndromes
compute the error locator polynomial
compute the error polynomial

ENCODER
CHANNEL

DECODER
More on RS decoding algorithms

- Clearly finding $|M|$ and $M^{-1}$ is not easy for large $t$
- Fortunately, RS has been researched and widely used in many applications for a long time, therefore there are more efficient algorithms for each task.

- E.g.,
  - to solve the key equation ----> Euclidean algorithm, Berlekamp algorithm, Berlekamp-Massey algorithm
  - to find roots of $s(x)$ ----> Chien Search algorithm
  - to compute error magnitude ----> Forney algorithm
Performance analysis
Performance analysis

Bell Laboratories

Performance of various block codes

BCH-1 = BCH(8191,8178)
BCH-2 = BCH(8191,8165)
BCH-3 = BCH(8191,8152)
RS-4 = RS(255,247)
RS-8 = RS(255,239)
RS-16 = RS(255,223)

UNCODED

SNR (dB)

output BER

BCH-1 = 0.15%
BCH-2 = 0.32%
BCH-3 = 12%
RS-4 = 6%
RS-8 = 3%
RS-16 = 0.48%

Code Overhead

0 5 10 15

0 10^-8 10^-10 10^-12 10^-14 10^-16 10^-18 10^-20
RS codes: simulated results

RS256 without erasures corrections
dashed lines = theory

- RS(80,58)
- RS(69,65, RS(72,68))
- uncoded
- RS(255,239)

bit error rate (data)

$10^0 \quad 10^{-2} \quad 10^{-4} \quad 10^{-6} \quad 10^{-8} \quad 10^{-10} \quad 10^{-12} \quad 10^{-14} \quad 10^{-16}$

$10^0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 14 \quad 15$

$10^0 \quad 10^{-2} \quad 10^{-4} \quad 10^{-6} \quad 10^{-8} \quad 10^{-10} \quad 10^{-12} \quad 10^{-14} \quad 10^{-16}$
An alternative representation

- The above coding gains (in dB) do not map directly in optical gain (in dBm)
- Optical gain depends on channel and receiver response
- Usually optical power is proportional to detector current, and not to electrical power!
- Input BER versus output BER is a **unique representation** of the performance of a given code. It does not depend on optical response. It is very useful for comparing the performance of different codes with variable size / overhead.
Input BER versus Output BER

Performance of various block codes

- BCH-1 = BCH(8191, 8178)
- BCH-2 = BCH(8191, 8165)
- BCH-3 = BCH(8191, 8152)
- RS-4 = RS(255, 247)
- RS-8 = RS(255, 239)
- RS-16 = RS(255, 223)

UNCODED
Latency issues
Latency issues

- Latency is obviously implementation dependant
- In many cases, total latency of encoding + decoding can be on the order of $2n-3n$ (n block size)
- For instance for RS(255,239) total latency could potentially be $\sim 3 \times 255 \times 8 \times 100 \text{ps} < 1 \mu s$
- Some existing FEC chips used in WAN have $30 \mu s$ total latency (longer blocks, lower overhead)
- Propagation time in fiber:
  - $300m \rightarrow 1 \mu s$
  - $3km \rightarrow 10 \mu s$
  - $30km \rightarrow 100 \mu s$
- In LAN applications, smaller block sizes are preferable (to minimize buffer size).
Summary
Conventional FEC schemes can achieve 6dB coding gain or more without need for “soft decisions”

Error correction on the decode side is up to implementers

Parity check could be used to perform bit alignment (in a serial approach) or de-skewing (in a WDM approach)

Parity check could be used to indicate link quality

Clear trade-off between FEC performance on one hand and [complexity, latency, overhead] on the other hand

Simple FEC codes exist (Hamming, binary BCH)

BCH & RS codes can be (are) implemented in generic CMOS technology