8 FORWARD ERROR CORRECTION

8.1 Error Free Communication on Noisy Channels?

If we work with blocks of bits, rather than single bits, we can achieve remarkable gains. Here's a simple example:

- start with M-ary orthogonal signals, coherently detected (though coherence is not necessary to the argument), and loosen our error rate bound to gain simplicity:

$$P_{es} \le (M-1)\,Q(\sqrt{E_s/N_0}) < M\,Q(\sqrt{E_s/N_0} < M\,e^{-E_s/2N_0}$$

- now consider each pulse to be carrying b bits:

$$M = 2^b = e^{b \ln 2} \quad \text{and} \quad E_s = b E_b$$

- and substitute:

$$P_{es} < M e^{-E_s/2N_0} = e^{b\ln 2} e^{-bE_b/2N_0} = e^{b(\ln 2 - E_b/2N_0)}$$

- What happens when b (hence M) become very large?

$$P_{es} \le \begin{cases} 1, & E_b/N_0 < 2\ln 2 \\ 0, & E_b/N_0 > 2\ln 2 \end{cases}$$

As long as E_b/N_0 is over a threshold, we can force the error rate to zero with <u>no</u> extra energy per bit!

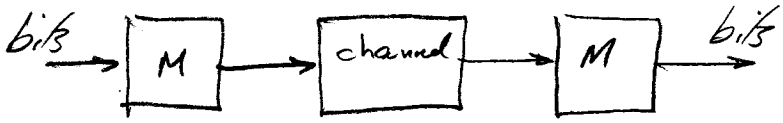
- Where's the catch? Bandwidth! Calculate bps/Hz:

$$R = b/T \qquad M = 2^b = 2WT$$

so $R/W = b/2^{b-1}$ very bad.

8.2 The Role of FEC

In our discussion of channels and modem techniques we focussed on bit by bit decisions:



and had a non zero BER even with optimized modems.

However, we can add redundant bits and process sequences as units, and correct some of the errors.

Consequences:

- for a given information rate, the channel rate increases (or we have to cutback on info rate, not attractive)
- so the bandwidth increases (unless we go to multilevel pulses)
- and the BER for received channel bits is worse
- but the error correcting properties of a well chosen code more than make up for the raw BER degradation.
- so the required SNR to achieve a certain net error rate is reduced — the coding gain.

Again trading bandwidth and complexity against improved performance.

The alternative is ARQ (retransmission error control) which is normally used when you have a return channel, since:

- it takes far fewer check bits to detect errors reliably than to correct them;
- ARQ uses extra capacity, in the form of repeats, only when needed. FEC always carries the load of redundant bits.
- ARQ is a lot easier to implement.

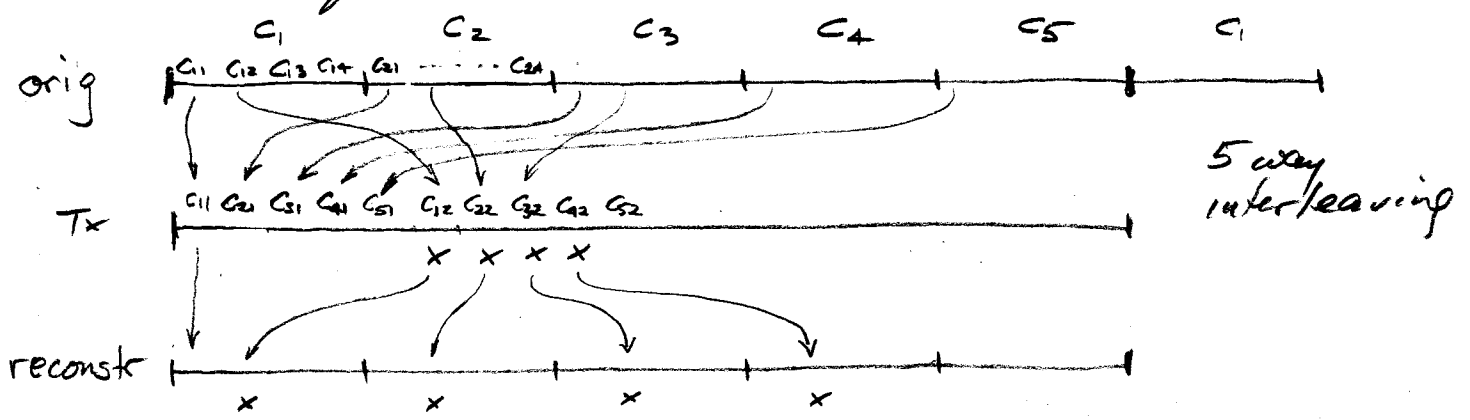
but you may not have that return channel.

FEC is much used in space applications (probes, comms satellites) since link power budget is tight, there's lots of bandwidth, and the return channel may not be useful because of the delay.

FEC is also used in computer memories (usually DRAM).

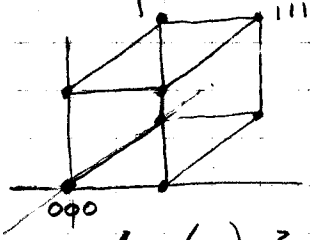
Radio paging systems and other one way systems rely on FEC. (broadcast protocols need FEC)

FEC is most effective with random (Bernoulli) errors. If errors occur in bursts, then interleave several code sequences to break up bursts



example repetition code – easy, but not very powerful

- send every bit 3 times, take majority vote at Rx



use only 2 of the 8 possible seq's

- new $P_e' = \binom{3}{2} P_e^2 (1-P_e) + \binom{3}{3} P_e^3 \approx 3P_e^2 < P_e$ if $P_e \ll 1$

  e.g. if $P_e = 10^{-2}$, we now have $P_e' \approx 3 \times 10^{-4}$

  but at the cost of three fold reduction in data rate!

- explore rate/power tradeoffs; keep it simple and assume incoherent detection of orthogonal pulses:

$$P_e = \tfrac{1}{2} e^{-E_p/2N_0} \quad \text{and} \quad P_{e0} = \tfrac{1}{2} e^{-E_b/2N_0} \text{ for original system.}$$

- scenario 1: no bandwidth restriction, keep info bit rate and transmit power constant, speed up channel bits by factor of 3 to compensate. Then each channel bit receives $\frac{1}{3}$ of original energy, and:

$$P_e' \approx 3P_e^2 = 3\left(\tfrac{1}{2} e^{-E_p/2N_0}\right)^2 = 3\left(\tfrac{1}{2} e^{-E_b/6N_0}\right)^2$$

$$= 3 \times \tfrac{1}{4} e^{-E_b/3N_0} = 3 \times \tfrac{1}{4} \times 2^{2/3} \left(\tfrac{1}{2} e^{-E_b/2N_0}\right)^{2/3} = 1.19\, P_{e0}^{2/3}$$

- since this is greater than P_{e0} we have a poorer system than the original.

- reason is that we split the energy and make several poor decisions, then try to recover by majority rules. Better to combine the signal energy while it's still analog.

- calculate the coding gain, ratio of original E_b to new E_b' to keep net BER unchanged:

$$P_e' \approx 3P_e^2 = P_{e0}$$

$$3\left(\tfrac{1}{2}e^{-E_b'/6N_0}\right)^2 = \tfrac{1}{2}e^{-E_b/2N_0}$$

$$E_b/2N_0 - E_b'/3N_0 = \ln(3/2) = 0.405$$

$$\left(1 - \tfrac{2}{3}\tfrac{E_b'}{E_b}\right)\tfrac{E_b}{2N_0} = 0.405$$

so for large $E_b/2N_0$, $E_b' \approx \frac{3}{2}E_b$. Means it takes more energy per bit with majority rules. Coding gain in dB is negative

- scenario 2: keep bandwidth, channel bit rate, power const, and reduce info bit rate.
  - as we saw originally, we improve BER at cost of data rate.
  - $P_e' \approx 3P_{e0}^2 = \frac{3}{4}e^{-E_p/N_0}$
  - and since data rate is 1/3 original, we are now spending $E_b = 3E_p$, and
    $$P_e' = \tfrac{3}{4}e^{-E_b/3N_0}$$
  - but why not just slow down without dividing into 3 bits with majority — just stretch pulses by factor of 3, tripling the energy per bit. Then:
    $$P_e'' = \tfrac{1}{2}e^{-E_b/2N_0} = \tfrac{1}{2}\left(\tfrac{4}{3}\right)^{3/2}\left(P_e'\right)^{3/2}$$

    better than majority rules because energy combined while analog.

example Hamming (7, 4) code — single error correcting
(4 ← info bits; 7 ← sequence length)

of the 128 possible 7-bit sequences, use only 16.

$p_1 = i_1 + i_2 + i_3$

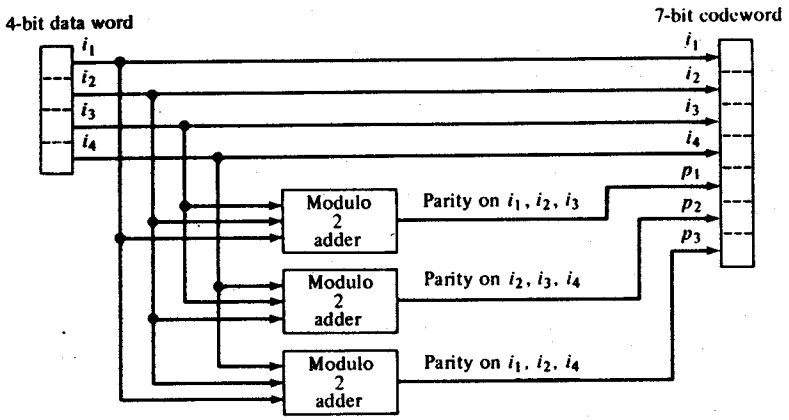
$p_2 = i_2 + i_3 + i_4$

$p_3 = i_1 + i_2 + i_4$.

i_1	i_2	i_3	i_4	p_1	p_2	p_3

Table 1.1 Hamming (7, 4) code.

0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	0	1	0	1	1	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	0	1	1	1	0
1	0	1	0	0	1	1
1	0	1	1	0	0	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	1	0	0
1	1	1	1	1	1	1



(a) Encoder

- send $\underline{c} = (i_1 \cdots i_4 p_1 \cdots p_3)$
- receive $\underline{r} = \underline{c} \oplus \underline{e}$

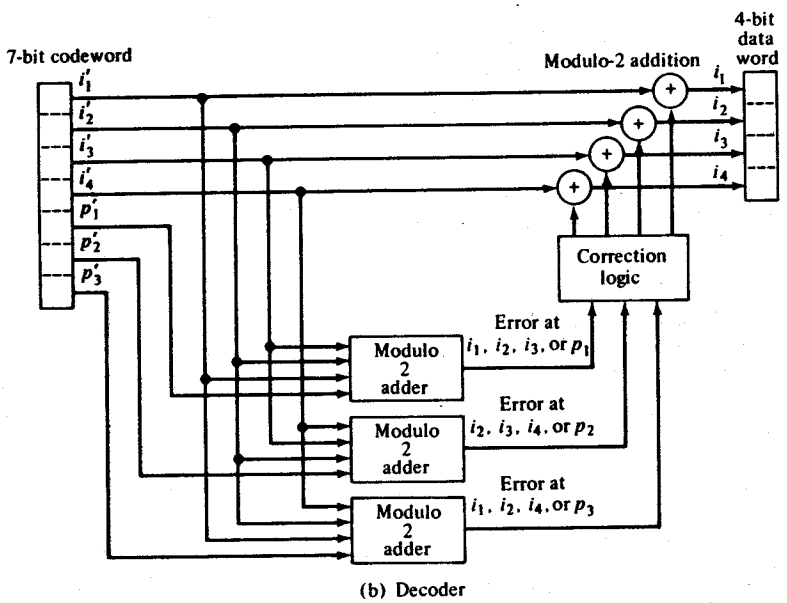
where $\underline{e}$ is error pattern

- at receiver, calculate parity bits $p_1' p_2' p_3'$ on basis of $r_1 r_2 r_3 r_4$, and compare with parity bits actually received

$$(p_1' \cdots p_3') \oplus (r_5 r_6 r_7) = (s_1 s_2 s_3)$$

where $(s_1 \cdots s_3)$ is the **syndrome**.

If $\underline{e} = \underline{0}$, then $p_1' p_2' p_3' = p_1 p_2 p_3$ and $\underline{s} = \underline{0}$. Hence non zero syndrome implies error. In fact, since it's linear, syndrome is sum of a (zero) contribution from codeword plus contribution from $\underline{e}$



(b) Decoder

Figure 1.4 A simple Hamming (7, 4) encoder/decoder.

- consider the seven 1-bit error patterns

e_1	e_2	e_3	e_4	e_5	e_6	e_7	s_1	s_2	s_3
1	0	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	1	1	1
		1					1	1	0
			1				0	1	1
				1			1	0	0
					1		0	1	0
						1	0	0	1

the syndrome patterns are unique, so we know which bit was in error, and can therefore correct it.

- since only 8 poss syndromes, all accounted for, multiple errors are uncorrectable

- is there a coding gain?

$$\text{new } P_e' = \sum_{k=2}^{7} \binom{7}{k} P_e^{\,k} (1-P_e)^{7-k} \approx 21\, P_e^{\,2}$$

$$= 21\left(\tfrac{1}{2} e^{-E_p/2N_0}\right)^2 = 21\left(\tfrac{1}{2} e^{-2E_b/7N_0}\right)^2 \quad \text{pulse energy } \tfrac{1}{7} E_b$$

$$\approx 21 \cdot \tfrac{1}{4}\left(e^{-E_b/2N_0}\right)^{8/7} = 21 \cdot \tfrac{1}{4} \cdot 2^{8/7} P_{e0}^{\,8/7}$$

$$= 11.6\, P_{e0}^{\,8/7}$$

Hence for small P_{e0}, there's an improvement, and we have a coding gain, not loss.

Note that uncorrectable errors occur as 4 bit blocks changed to another randomly selected 4 bit block (BER = 1/2 in the block), rather than singly, as in the original.