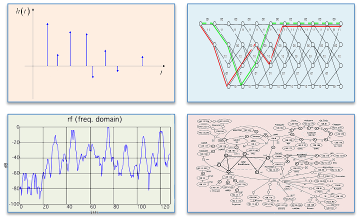


INTRODUCTION TO EECS II
DIGITAL COMMUNICATION SYSTEMS

6.02 Fall 2011
Lecture #6: Channel Coding - 2

- Linear (n,k) block codes
  - Rectangular parity codes
  - Hamming codes
- Combating burst errors: interleaving

(n,k) Systematic Linear Block Codes

- Split data into k-bit blocks
- Add $(n$-$k)$ parity bits to each block using $(n$-$k)$ linear equations, making each block n bits long

k	n-k
Message bits	Parity bits

n

The entire block is called the "code word in systematic form"

- Every linear code can be represented in systematic form

Example: Rectangular Parity Codes

Idea: start with rectangular array of data bits, add parity checks for each row and column. Single-bit error in data will show up as parity errors in a particular row and column, pinpointing the bit that has the error.

D_1	D_2	P_1
D_3	D_4	P_2
P_3	P_4	

P_1 is parity bit for row #1

P_4 is parity bit for column #2

(n,k,d)=?

0 1 1
1 1 0
1 0

Parity for each row and column is correct ⇒ no errors

0 1 1
1 0 0
1 0

Parity check fails for row #2 and column #2 ⇒ bit D_4 is incorrect

0 1 1
1 1 1
1 0

Parity check only fails for row #2 ⇒ bit P_2 is incorrect

Rectangular Code Corrects Single Errors

Claim: The HD of the rectangular code with r rows and c columns is 3. Hence, it is a single error correction (SEC) code.

Code rate = rc / $(rc + r + c)$.

If we add an overall parity bit P, we get a (rc+r+c+1, rc, 4) code

Improves error detection but not correction capability

D_1	D_2	D_3	D_4	P_1
D_5	D_6	D_7	D_8	P_2
D_9	D_{10}	D_{11}	D_{12}	P_3
P_4	P_5	P_6	P_7	P

Proof: Three cases.
(1) Msgs with HD 1 → differ in 1 row and 1 col parity
(2) Msgs with HD 2 → differ in either row OR col or both → HD >= 4 here.
(3) Msgs with HD 3 or more → systematic code so differ in that many bits

Decoding Rectangular Parity Codes

Receiver gets possibly corrupted word, *w*.

Calculates all the parity bits from the data bits.

If no parity errors, return *rc* bits of data.

Single row or column parity bit error → *rc* data bits are fine, return them

If parity of row *x* and parity of column *y* are in error, then the data bit in the (*x*,*y*) position is wrong; flip it and return the *rc* data bits

All other parity errors are *uncorrectable*. Return the data as-is, flag an "uncorrectable error"

How Many Parity Bits Do We Need?

- We have n-k parity bits, which collectively can represent 2^{n-k} possibilities
- For single-bit error correction, parity bits need to represent two sets of cases:
  - Case 1: No error has occurred (1 possibility)
  - Case 2: Exactly one of the code word bits has an error (n possibilities, not k)
- So we need $n+1 \le 2^{n-k}$

$$n \le 2^{n-k} - 1$$

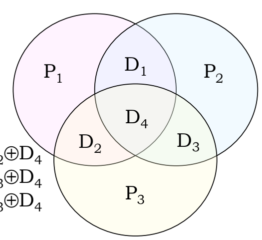
- Hamming codes correct single errors with this minimum number of parity bits (7,4,3), (15,11,3), …

Example: (7,4,3) Hamming Code

- Use multiple parity bits, each covering a subset of the data bits.
- No two message bits belong to exactly the same subsets, so a single-bit error will generate a unique set of parity check errors.

Modulo-2 addition, aka XOR

$P_1 = D_1 \oplus D_2 \oplus D_4$
$P_2 = D_1 \oplus D_3 \oplus D_4$
$P_3 = D_2 \oplus D_3 \oplus D_4$



Suppose we check the parity and discover that P2 and P3 indicate an error?
bit D3 must have flipped

What if only P3 indicates an error?
P3 itself had the error!

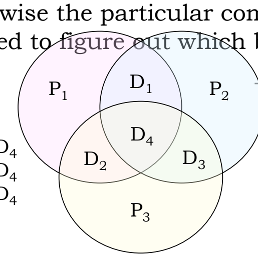
Syndrome Decoding

- After receiving the (possibly corrupted) message, compute a **syndrome** bit (E_i) for each parity bit

$$E_1 = D_1 \oplus D_2 \oplus D_4 \oplus P_1$$
$$E_2 = D_1 \oplus D_3 \oplus D_4 \oplus P_2$$
$$E_3 = D_2 \oplus D_3 \oplus D_4 \oplus P_3$$

- If all the E_i are zero: no errors
- Otherwise the particular combination of the E_i can be used to figure out which bit to correct

$P_1 = D_1 \oplus D_2 \oplus D_4$
$P_2 = D_1 \oplus D_3 \oplus D_4$
$P_3 = D_2 \oplus D_3 \oplus D_4$



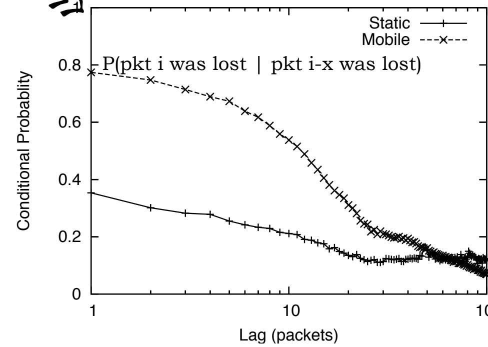
$E_3E_2E_1$	Corrective Action
000	no errors
001	p_1 has an error, flip to correct
010	p_2 has an error, flip to correct
011	d_1 has an error, flip to correct
100	p_3 has an error, flip to correct
101	d_2 has an error, flip to correct
110	d_3 has an error, flip to correct
111	d_4 has an error, flip to correct

Linear Block Codes: Wrap-Up

- (n,k,d) codes with rate k/n
- Code words are linear operations over message bits: sum of any two code words is a code word
- Message + 1 parity bit: (n+1,n,2) code
  - Good code rate, but only 1-bit error detection
- Replicating each bit c times is a (c,1,c) code
  - Simple way to get great error correction; poor code rate
- Hamming single-error correcting codes are (n, n-m, 3) where n = 2^m - 1 for m > 1
  - Adding an overall parity bit makes the code (n+1,n-p,4)
- Rectangular parity codes are (rc+r+c, rc, 3) codes
  - Rate not as good as Hamming codes

Burst Errors

Correcting single-bit errors is nice, but in many situations errors come in bursts many bits long (e.g., fading or burst of interference on wireless channel, damage to storage media etc.). How does single-bit error correction help with that?

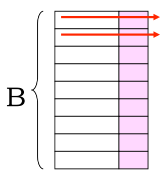


Experiment on 802.11g network (static and mobile)

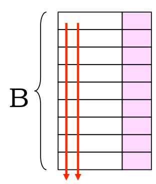
Note: P(pkt lost) ≈ s*P(bit error) where s = pkt size

Coping with Burst Errors by Interleaving

Well, can we think of a way to turn a B-bit error burst into B single-bit errors?



Row-by-row transmission order



Col-by-col transmission order

Problem: Bits from a particular codeword are transmitted sequentially, so a B-bit burst produces multi-bit errors.

Solution: interleave bits from B different codewords. Now a B-bit burst produces 1-bit errors in B different codewords.