Lecture 6 - Error Detection/Correction CRC

Block codes and d_{min}

Block code (n, k): k data bits into n-bit codewords

Key parameters:

- Code rate: $R_c = k/n$
- Hamming distance: number of bit positions where two words differ
- Minimum distance d_{min}: smallest hamming distance between any codeword pair

The d_{min} theorem: $e_c+e_d <= d_{min} -1$, $e_c<=e_d$

- To detect d errors: need $d_{min}>= d + 1$
- To correct d errors: need $d_{min}>= 2d+1$

Single bit error correction theorem: Single-bit error correction is a data integrity technique that detects and fixes a single flipped bit within a data unit (e.g., a byte or word) during storage or transmission. It uses redundant parity bits, such as Hamming codes, to identify the precise location of the error and reverse it, restoring the original data without needing retransmission.

Theorem: single bit error correction requires the codeword (actual bit and error bit) requires the codeword length n to satisfy: $n + 1 <= 2^{n-k}$

If n is less than the minimum value satisfying this inequality, the code cannot perform single bit error correction

Proof: sphere packing argument

For single-bit error correction ($e_c = 1$):

Each codeword needs $n + 1$ words reserved for itself:

- The codeword itself (0 errors)
- The n words that differ in exactly one bit position

Since there are 2^k codewords, we need at least $2^k(n + 1)$ distinct words.

But there are only 2^n possible words total. Therefore:

$$2^k(n + 1) \leq 2^n$$

Dividing both sides by 2^k:

$$n + 1 \leq 2^{n-k} \quad \blacksquare$$

Discussion of the theorem:

- This is a necessary condition (not sufficient)
- As increases, 2^{n-k} grows exponentially → inequality eventually holds
- Puts a limit on the maximum frame length for a given number of check bits

Intuition: the check bits must encode enough information to identify which bit (if any) was flipped. With n - k check bits we can distinguish 2^{n-k} cases. We need to identify n + 1 cases

BER comparison setup

Compare 3 error control methods:

1. Single bit error correction

2. Single bit error detection (with retransmission)
3. Rate ⅓ repetition coding (with error correction)

Parameters:

- Frame size: n = 1000 bits
- Channel BER = p = 10^{-6}
- Use the most efficient codes possible
- Retransmit if there is a detected error

Metrics:

- Utilization U: fraction of channel used for data
- Residual BER P_e: probability of bit error after decoding

Block error probabilities

Errors in each bit are independent → the number of errors in a block is binomial

$$f(w) = \binom{N}{w} p^w (1-p)^{N-w}, \qquad N = 1000, \quad p = 10^{-6}$$

Key values:

Errors w	$f(w)$
0	0.999
1	0.001
2	$\approx 10^{-6.3}$
3	$\approx 10^{-9.8}$

---> most blocks have zero errors → blocks with more than 2 errors are rare

Method 1: single bit error correction

From the theorem: $(1000 + 1) \leq 2^{n-k}$, so $n - k = 10$ check bits.

Utilization: No retransmissions needed:

$$U = \frac{k}{n} = \frac{990}{1000} = 0.990$$

Residual BER: Errors occur when ≥ 2 bits are wrong (can only correct 1):

$$P_e = \frac{1}{N} \sum_{q=2}^{N} q \cdot f(q) \approx \frac{2 \times 10^{-6.3} + 3 \times 10^{-9.8}}{1000} \approx 10^{-9}$$

Method 2: single bit error detection

Use $n - k = 1$ check bit (parity bit code).

Utilization: Retransmissions follow geometric distribution with success probability $\approx 1 - 10^{-3}$:

$$U = \frac{k}{n \cdot E[M]} = \frac{999}{1000 \times \frac{1}{0.999}} = \frac{998}{1000} = 0.998$$

Residual BER: Accepted frames may have ≥ 2 undetected errors (even number for parity):

$$P_e \approx 10^{-9}$$

Same order as error correction, but **higher utilization**!

Method 3: repetition coding

Rate 1/3 repetition: $n = 999$, $k = 333$. Each bit is transmitted 3 times.

Utilization:

$$U = R_c = \frac{1}{3} = 0.333$$

Residual BER: A bit error occurs when ≥ 2 of 3 copies are flipped:

$$P_e = \binom{3}{2} p^2(1-p) + p^3 = 3 \times 10^{-12} + 10^{-18} \approx 3 \times 10^{-12}$$

Better P_e but at **enormous throughput cost**.

BER comparison results

Method	**Utilization U**	**Residual P_e**
Single-bit error detection	0.998	10^{-9}
Single-bit error correction	0.990	10^{-9}
Rate 1/3 repetition coding	0.333	10^{-12}

n = 1000 bits
BER = 10^{-6}

Example

CRC introduction: cyclic redundancy check: most widely used error detection codes in networking
Treat bits strings as polynomials over GF(2)

Polynomial Notation:

Symbol	Meaning	Size
$M(x)$	Message (data) polynomial	k bits
$G(x)$	Generator polynomial	$r+1$ bits
$R(x)$	Remainder (CRC checksum)	r bits
$T(x)$	Transmitted frame (codeword)	$n = k + r$ bits

Key principle: Valid codewords $T(x)$ are divisible by $G(x)$:

$$T(x) \equiv 0 \pmod{G(x)}$$

Binary Polynomial Representation

Bits strings are represented as polynomials where each bit is a coefficient:
→ each x is a 1 and the exponent is the index

10011 → x^4+x+1
1101 → x^3+x^2+1
101000 → x^5+x^3

Conventions:

- Leftmost bit is the highest degree term
- Each bit position i corresponds to coefficient of x^i
- A k bit message has polynomial of degree <= k-1

GF(2) arithmetic

GF(2) is the smallest Galois field, consisting of elements with arithmetic performed modulo 2. Addition is equivalent to the logical XOR operation (), and multiplication is equivalent to the logical AND operation (). It is essential for cryptography (AES), coding theory, and digital logic, forming the

basis for extension fields like

All arithmetic is modulo-2 (XOR):

- Addition: $1+1=0, \quad 1+0=1, \quad 0+0=0$
- **No carries or borrows**
- Addition and subtraction are the **same operation** (XOR)

Example – Addition:

$$(x^3+x^2+1)+(x^3+x+1)=x^2+x$$

$$1101 \oplus 1011 = 0110$$

Example – Multiplication:

$$(x^2+1)(x+1)=x^3+x^2+x+1$$

$$101 \times 11 = 1111$$

CRC encoding steps

Given message $M(x)$ and generator $G(x)$ of degree r:

Step 1: Multiply $M(x)$ by x^r (append r zero bits to message)

Step 2: Divide $x^r M(x)$ by $G(x)$ to get remainder $R(x)$:

$$x^r M(x) = Q(x) \cdot G(x) + R(x)$$

Step 3: Transmitted frame:

$$T(x) = x^r M(x) + R(x)$$

Why this works:

$$T(x) = Q(x) \cdot G(x) + R(x) + R(x) = Q(x) \cdot G(x)$$

So $T(x)$ is divisible by $G(x)$ ✓ $\quad (R(x)+R(x)=0$ in GF(2))

CRC decoding

At the receiver: you divide the received frame T(x) by G(x)

If the remainder = 0 → accept since no detected error

If the remainder =/ 0 → reject since error detected → request retransmission

Verification of example:

Divide 1101001 by 1011 → reminder = 000 → valid codeword

If errors occurred:

Divide 1001001 by 1011 → reminder =/ 0 → error

CRC error detection capabilities

For a generator G(x) of degree rL

Error Type	Detection Condition
All single-bit errors	$G(x)$ has more than one term
All double-bit errors	$G(x)$ does not divide $x^k + 1$ for $k \leq n$
All odd numbers of errors	$G(x)$ has $(x + 1)$ as a factor
All burst errors $\leq r$ bits	Always (for any $G(x)$ of degree r)
Burst errors of length $r + 1$	Probability $1 - 2^{-(r-1)}$
Burst errors of length $> r + 1$	Probability $1 - 2^{-r}$

Burst error: contiguous sequence if I buts where the first and last buts are in error

Standard CRC polynomials

Name	Polynomial	r	Common Use
CRC-8	$x^8 + x^2 + x + 1$	8	ATM header
CRC-16	$x^{16} + x^{15} + x^2 + 1$	16	USB, Modbus
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$	16	HDLC, Bluetooth
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + \ldots$	32	Ethernet, WiFi

CRC-32 in practice

Used in ethernet and WiFi → the most common link layer protocol

CRC-32 detects:

- All single, double, and triple bit errors
- All odd numbers of errors
- All burst errors of length ≤ 32
- For random errors: undetected error probability ≈ 2 −32 ≈ 2.3 × 10−1

Implementation:

- Hardware: shift registers with XOR gates (computed at wire speed)
- Software: lookup tables (256-entry table, one byte at a time)
- Adds only 32 bits (= 4 bytes) of overhead per frame

Ethernet Frame Check Sequence (FCS): last 4 bytes of every frame are CRC-32