1. Fundamentals

Data transmission over noisy channels introduces errors. We add redundant bits to detect or correct them.

* **Code Word (n bits):** k data bits + r redundant/check bits ($n = k + r$). * **Hamming Distance (d_{min}):** Minimum number of bit positions in which two valid code words differ. * **Error Detection Capability:** To detect up to e errors, $d_{min} \geq e + 1$. * **Error Correction Capability:** To correct up to t errors, $d_{min} \geq 2t + 1$.

2. Block Coding Principles

Linear block codes map k-bit information blocks to n-bit code words using a generator matrix G.

* **Generator Matrix (G):** $k \times n$ matrix where $G = [I_k \mid P]$. * **Parity Check Matrix (H):** $(n - k) \times n$ matrix where $H = [P^T \mid I_{n-k}]$. * **Syndrome Detection:** $S = rH^T$, where r is the received code word. * If $S = 0$, code word is valid (or undetected error). * If $S \neq 0$, errors are present.

3. Linear Block Hamming Codes

Hamming codes are linear block codes designed with $d_{min} = 3$, capable of correcting single-bit errors.

* **Redundancy Condition:** $2^r \geq k + r + 1$ * **Encoding:** Positions that are powers of 2 $(1, 2, 4, 8, \dots)$ are parity bits; remaining are data bits. * **Decoding:** Calculate syndrome by checking parity bits. The binary value of the non-zero syndrome indicates the exact erroneous bit position.

4. Cyclic Redundancy Check (CRC)

CRC uses polynomial arithmetic (modulo-2 division) for burst error detection.

* **Polynomial Representation:** Bit string $110101 \rightarrow x^5 + x^4 + x^2 + 1$. * **Generator Polynomial ($G(x)$):** Degree r. Sender appends r zeros to data $M(x)$, divides by $G(x)$, and transmits $T(x) = x^r M(x) -$ Remainder. * **Receiver Check:** Divides received polynomial by $G(x)$. Remainder 0 implies success. * **Common Standards:** * CRC-12 (telecom) * CRC-16 (ANSI) * CRC-32 (Ethernet)

5. Checksum

Used primarily in the Internet Protocol (IP) suite (Transport and Network layers).

* **Mechanism:** Data is treated as a sequence of 16-bit integers. Added using one's complement arithmetic. * **Sender:** Sums all words, takes one's complement of the result, and places it in the checksum field. * **Receiver:** Sums all words including the checksum. If the result is all 1s (zero in one's complement), no error is detected.

6. Summary Comparison

Method	**Primary Use**	**Main Strength**
Hamming	Memory, RAM	Single-bit correction
CRC	Data links, Ethernet	Burst error detection
Checksum	TCP/IP, UDP	Fast software implementation