Forward Error Correction using Erasure Codes

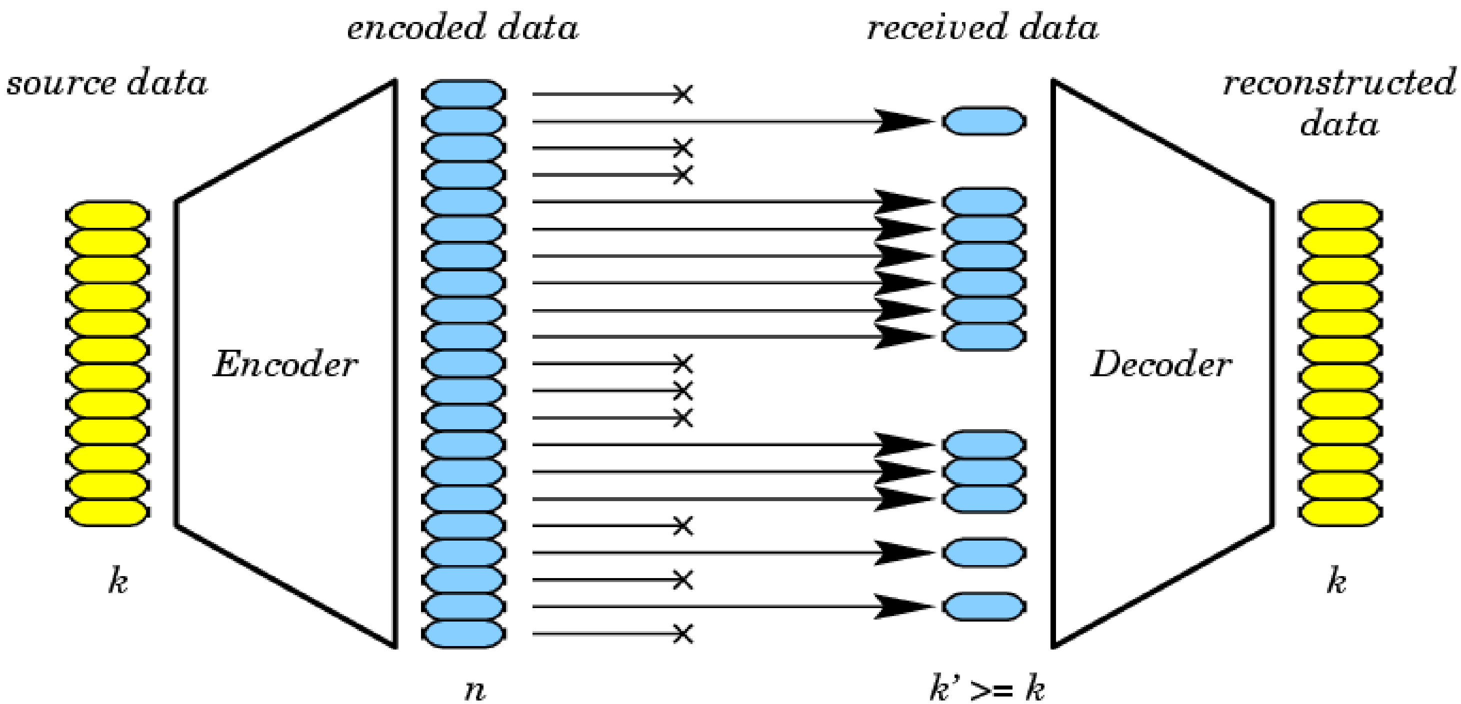
Reference:

L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," ACM SIGCOMM *Computer Communication Review*, April 1997

Erasure Codes

- Erasures are missing packets in a stream
  - Uncorrectable errors at the link layer
  - Losses at congested routers
- (n, k) code
  - k blocks of source data are encoded to n blocks of encoded data, such that the source data can be reconstructed from any subset of k encoded blocks
  - each block is a data item which can be operated on with arithmetic operations

Encoding/decoding process



- k fixed-length packets; each packet is partitioned into data items.
- The encoding/decoding process is applied to k data items from the k packets, one data item per packet

Applications of FEC

- Used to reduce the number of packets that require ARQ recovery

- Particularly good for large-scale multicast of long files (packet flows)
  - Different packets are missing at different receivers – the same redundant packet(s) can be used by (almost) all receivers with missing packets

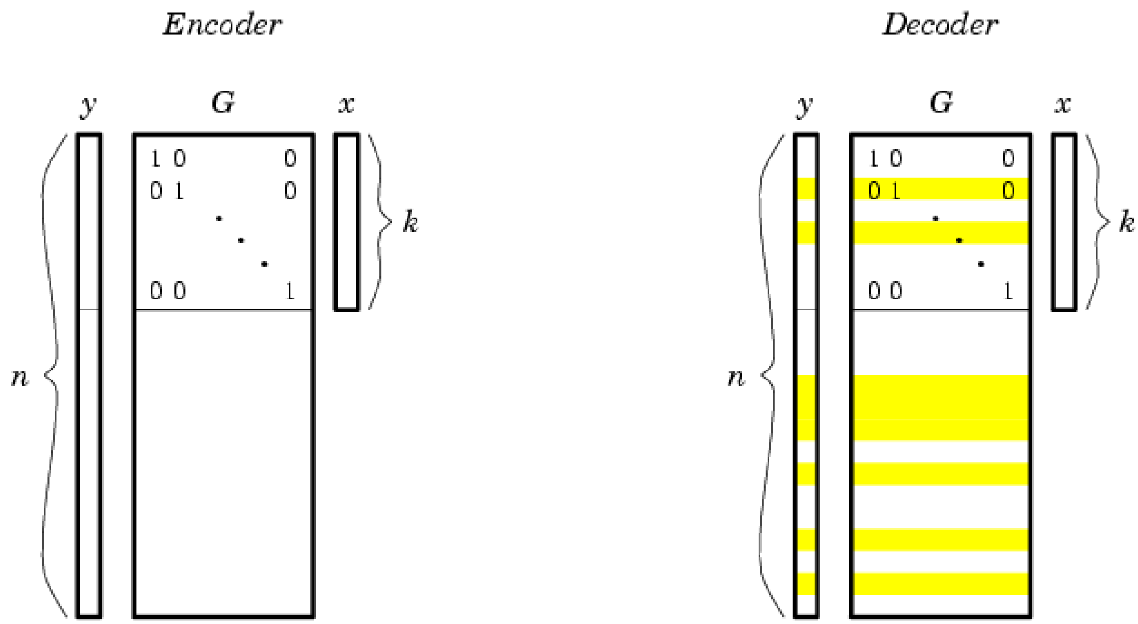
Linear codes

- Can be analyzed using the properties of linear algebra
- Let $\underline{x} = x_0 \ldots x_{k-1}$ be the source data items, G an n x k matrix, then an (n, k) linear code can be represented by

$$\underline{y} = G\, \underline{x}$$

for a properly defined G such that any subset of k equations are linearly independent, i.e., any k x k matrix extracted from G is invertible.

Encoding/decoding in matrix form



- For a **systematic** code, the top k rows of G constitute the identity matrix.
- With a systematic code, the number of equations to be solved is small (< k) when few losses are expected.

Encoding/decoding in matrix form (cont.)

- G is called the generator matrix of the code.
- For a systematic code, G contains the identity matrix

  => *the remaining rows of the matrix must all contain nonzero elements*
- Any subset of k encoded blocks should convey information on all k source blocks
  - G has rank k
  - each column of G has at most k-1 zero elements

Problem with using ordinary arithmetic

- Suppose each x_i is represented using b bits, each coefficient of G is represented using b' bits
- Then y_i needs b+b'+$\lceil \log_2 k \rceil$ bits to avoid loss of precision
  - Expansion of source data!
- Extra bits to represent y_i constitute a sizable communication overhead

Computations in finite fields

- A field is a set in which we can add, subtract, multiply, and divide
- A finite field has a finite number of elements. It is closed under addition and multiplication.
  - sums and products are field elements
  - exact computation without requiring more bits
- Map data items into field elements, operate on them according to field rules, then apply inverse mapping

Prime fields

- GF(p), with p prime, is the set of integers from 0 to p-1
  - GF stands for Galois field
- Field elements require $\lceil \log_2 p \rceil > \log_2 p$ bits each
  - Operand size may not align with word size
- Addition and multiplication require modulo p operations which are costly

Extension fields

- $GF(p^r)$, with p prime and $r > 1$
  - there are $q=p^r$ elements

- Each field element can be considered as a polynomial of degree r-1 with coefficients in GF(p)

- Addition of two elements (polynomials)
  - For each coefficient, sum modulo p

Polynomials

- Addition of two elements in $GF(p^r)$

$$c_0 + c_1 x^1 + \ldots + c_{r-2} x^{r-2} + c_{r-1} x^{r-1}$$

$$b_0 + b_1 x^1 + \ldots + b_{r-2} x^{r-2} + b_{r-1} x^{r-1}$$

$$d_0 + d_1 x^1 + \ldots + d_{r-2} x^{r-2} + d_{r-1} x^{r-1} \quad \text{sum}$$

$$\text{where} \quad d_i = (b_i + c_i) \bmod p$$

Extension fields (cont.)

- Multiplication
  - The product of two polynomials (elements) is computed modulo an irreducible polynomial (one without divisors in GF(p^r)) of degree r, and with coefficients reduced modulo p

- **The case of p=2**, GF(2^r)
  - each element requires exactly r bits to represent
  - addition and substraction are the same, implemented by bit-wise exclusive OR

Special element

- For both prime and extension fields, there exists at least one special element, denoted by α, whose powers generate all non-zero elements of the field
- Powers of α repeat with a period of length q-1, hence $\alpha^{q-1} = \alpha^0 = 1$
- Example: generator for GF(5) is 2

  whose powers are 1, 2, 4, 3, 1

  where 2^3 mod 5 = 3 and 2^4 mod 5 = 1

Special element for GF(2^3)

Let u be the root of $1 + x + x^3$ (u is the special element α)

Thus $1+u+u^3 = 0$

- $u^0 = 1$ 001
- $u^1 = u$ 010
- $u^2 = u^2$ 100
- $u^3 = u+1$ 011
- $u^4 = u^2+u$ 110
- $u^5 = u^2+u+1$ 111
- $u^6 = u^2+1$ 101
- $u^7 = 1$ 001

There are 7 nonzero elements ($q-1 = 7$)

Special element for $GF(2^8)$

u is root of the irreducible polynomial $1 + x^2 + x^3 + x^4 + x^8$

Thus, $1 + u^2 + u^3 + u^4 + u^8 = 0$

u generates a cyclic group of nonzero elements ($q-1 = 255$)

- $u^0 = 1$ 00000001
- $u^1 = u$ 00000010
- $u^2 = u^2$ 00000100
- $u^3 = u^3$ 00001000
- $u^4 = u^4$ 00010000
- $u^5 = u^5$ 00100000
- $u^6 = u^6$ 01000000
- $u^7 = u^7$ 10000000
- $u^8 = 1 + u^2 + u^3 + u^4$ 00011101
- $u^9 = u(1 + u^2 + u^3 + u^4)$
  $= u + u^3 + u^4 + u^5$ 00111010

...

$\boxed{u^{q-1} = u^0 = 1}$

Multiplication and division

- Any nonzero element x can be expressed as
  x = α^{k_x} where k_x is logarithm of x
- Multiplication and division can be computed using logarithms, as follows:

$$xy = \alpha^{\left|k_x + k_y\right|_{q-1}}$$

$$\frac{1}{x} = \alpha^{q-1-k_x}$$

- Division performed as multiplication by inverse element
- The logarithm, exponential, and multiplicative inverse of each non-zero element can be kept in tables

Multiplication example for GF(2^3)

- $u^5 \times u^6 = (u^2+u+1)\times(u^2+1) = u^4+u^3+u^2 + u^2+u+1$
  $= u^4 + u^3 + u + 1$
  $= u^4$ $\quad\quad (1+u+u^3=0)$

- Alternatively,
  $u^5 \times u^6 = u^{5+6-(q-1)} = u^{5+6-7} = u^4$

Data recovery

- Assume use of a systematic code
- Let $\underline{x}$ denote source data items, $\underline{y}'$ denote data items at receiver, and matrix G' the subset of rows from G
  - after y_i has been set equal to any x_i received
  - rank of G' is $\leq k$

$$\underline{y}' = G' \underline{x} \rightarrow \underline{x} = G'^{-1} \underline{y}'$$

- The cost of inverting G' is amortized over all data items contained in a packet

Data recovery (cont.)

- Cost of inverting G' is $O(kL^2)$,
  where $L \le \min\{k, n-k\}$ is the number of packets to be recovered
  - Cost counted in no. of multiplications
  - This cost is negligible because it is amortized over a large number of data items in a packet (e.g., number of bytes)
- Reconstructing the L missing packets has a total cost of $O(kL)$

Vandermonde matrix

- A kxk matrix with coefficients

$$v_{ij} = (x_i)^{j-1} \boxed{= (\alpha^i)^{j-1}}$$

where the x_i's are elements of GF(p^r)

for $q = p^r > k$

$$V = \begin{bmatrix} 1 & (\alpha)^1 & \ldots & (\alpha)^{k-1} \\ 1 & (\alpha^2)^1 & \ldots & (\alpha^2)^{k-1} \\ 1 & (\alpha^3)^1 & \ldots & (\alpha^3)^{k-1} \\ \ldots & \ldots & \ldots & \ldots \\ 1 & (\alpha^k)^1 & \ldots & (\alpha^k)^{k-1} \end{bmatrix}$$

- Such a matrix has the determinant

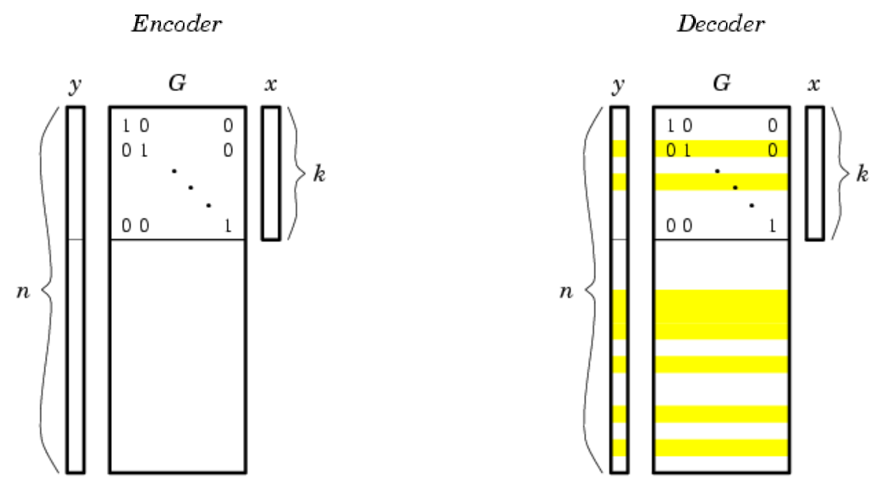
$$\prod_{i,j=1\ldots k, i<j} (x_j - x_i)$$

which is nonzero

Matrix G for a systematic code

- Use the top h rows of V as the bottom h rows of G under the identity matrix, for $1 \le h \le k$

$$V_{(n-k)\times k} = \begin{bmatrix} 1 & (\alpha)^1 & \ldots & \alpha^{k-1} \\ 1 & (\alpha^2)^1 & \ldots & (\alpha^2)^{k-1} \\ 1 & (\alpha^3)^1 & \ldots & (\alpha^3)^{k-1} \\ \ldots & \ldots & \ldots & \ldots \\ 1 & (\alpha^h)^1 & \ldots & (\alpha^h)^{k-1} \end{bmatrix}$$



RSE coder [Rizzo's implementation]

- Data items are elements of Galois field $GF(2^r)$, r ranges from 2 to 16
  - encoding time increases with r
- number of data items in each packet may be arbitrary (but must be same for all packets)
- 1-byte data items are most efficient in Rizzo's implementation
  - use table lookups
- (n, k) codes for $k \leq 2^r-1$ and $n \leq 2k$

Performance

- Encoding speed = $c_e/(n-k)$, where c_e is a constant
- Decoding speed = c_d/L , where c_d is a constant, L is the number of missing data items
  - c_d is slightly smaller than c_e due to matrix inversion overhead at receiver
  - matrix inversion has a cost of $O(kL^2)$, which is amortized over all data items in a packet and is negligible for packet size larger than 256 bytes

The end