**Forward Error Correction using Erasure Codes**

Reference:

---

**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 or 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 $\mathbf{x} = x_0 \ldots x_{k-1}$ be the source data items, $G$ an $n \times k$ matrix, then an $(n, k)$ linear code can be represented by
  \[ \mathbf{y} = G \mathbf{x} \]
  for a properly defined $G$ such that any subset of $k$ equations are linearly independent, i.e., any $k \times k$ 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.
- 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
- For a systematic code, $G$ contains the identity matrix => the remaining rows of the matrix must all contain nonzero 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 additions and multiplications.
  - 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 (except for $p=2$).
- 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 $p$

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 \( \alpha \), whose powers generate all non-zero elements of the field.
- Powers of \( \alpha \) 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 special element \( \alpha \)).
Thus \( 1+u+u^3 = 0 \).
- \( u^0 = 1 \) 001 1
- \( u^1 = u \) 010 2
- \( u^2 = u^2 \) 100 4
- \( u^3 = u+1 \) 011 3
- \( u^4 = u^2+u \) 110 6
- \( u^5 = u^2+u+1 \) 111 7
- \( u^6 = u^2+1 \) 101 5
- \( u^7 = 1 \) 001 1

There are 7 nonzero elements.
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 ($q-1 = 255$).

- $u^0 = 1$
- $u^1 = u$
- $u^2 = u^2$
- $u^3 = u^3$
- $u^4 = u^4$
- $u^5 = u^5$
- $u^6 = u^6$
- $u^7 = u^7$
- $u^8 = 1 + u^2 + u^3 + u^4$
- $u^9 = u(1 + u^2 + u^3 + u^4)$
  $= u + u^3 + u^4 + u^5$

... $u^{q-1} = u^0 = 1$

Multiplication and division

- Any nonzero element $x$ can be expressed as $x = \alpha^{k_x}$ where $k_x$ is the logarithm of $x$.
- Multiplication and division can be computed using logarithms, as follows:
  \[
  xy = \alpha^{k_x + k_y} \\
  \frac{1}{x} = \alpha^{q-1-k_x}
  \]

- The logarithm, exponential, and multiplicative inverse of a non-zero element can be kept in tables.
- Division performed as multiplication by inverse element.
### Multiplication example for GF(2^3)

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

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

### Data recovery

- Let \( x \) denote source data items, \( y' \) denote data items at receiver, and matrix \( G' \) the subset of rows from \( G \)

\[ y' = G' \times \rightarrow x = G'^{-1}y \]

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

Vandermonde matrix

- A $k \times k$ matrix with coefficients

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

where the $x_i$'s are elements of $GF(p^r)$ for $q = p^r > k$
- 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 $h=n-k$ rows of $V$ as the bottom $h$ rows of $G$ under the identity matrix, for $1 \leq h \leq k$

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

**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 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 $n \ll 2k$
Performance

- Encoding speed = \( \frac{c_e}{(n-k)} \), where \( c_e \) is a constant
- Decoding speed = \( \frac{c_d}{L} \), 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 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