In this document, I will try to sum up my current understanding of Random Linear Network Coding. It might be filled with errors ; use with care !

1 Theory

We want to send a set of packets $P_{i=0..N}$ from a sender to a receiver. In Linear Network Coding, the sender has to generate a set of coded packets C_j such that :

$$C_{j=0..M} = \sum_{i=0}^{N} P_i * a_{j,i}$$

Here, $a_{j,i}$ represents the coefficient of the corresponding original packet P_i in the current coded packet C_j. There are various ways of generating coefficients, to ensure the highest possible entropy in the set of C_j. However in practical cases, using random coefficients proved to be good enough.

The sender can generate any arbitrary number of packets, but if we want the receiver to retrieve the entire set of P_i, we need to have $M \geq N$ and to ensure sets of coefficients are linearly independents. The sender needs to send coded packets as well as coefficients used to encode them.

At this stage, matrix representation is more convenient :

$$\begin{pmatrix} C_0 \\ C_1 \\ \vdots \\ C_M \end{pmatrix} = \begin{pmatrix} a_{0,0} & \cdots & a_{0,N} \\ \vdots & \ddots & \vdots \\ a_{M,0} & \cdots & a_{M,N} \end{pmatrix} * \begin{pmatrix} P_0 \\ P_1 \\ \vdots \\ P_N \end{pmatrix}$$

or

$$C = A * P$$

On the receiver side, we got the coefficients matrix as well as the vector of encoded packets. To recover the vector of original packets, we need to solve the equation system. Basically, what we want to do is to find the invert of the coefficients matrix.

This can be achieved using Gauss-Jordan elimination, which I am not going to detail here.

Assuming every sets of coefficients were independent, we can compute the invert matrix A^{-1} on the receiver side. Then :

$$\text{We know that } C = A * P$$
$$\text{With the invert matrix, } A^{-1} * C = A^{-1} * A * P$$
$$\text{So, } P = A^{-1} * C$$

1.1 Numerical example

Using Octave, I applied the theory above on a simple example. Let's say we want to send a vector of three packets with a redundancy factor of $\frac{5}{3}$, thus sending 5 encoded packets. On the sender side :

$$P_s = \begin{pmatrix} 19 \\ 59 \\ 47 \end{pmatrix} \; A_s = \begin{pmatrix} 3 & 4 & 7 \\ 5 & 8 & 10 \\ 9 & 5 & 4 \\ 1 & 6 & 9 \\ 5 & 1 & 10 \end{pmatrix} \text{ Then we compute } C_s = A_s * P_s = \begin{pmatrix} 622 \\ 1037 \\ 654 \\ 796 \\ 624 \end{pmatrix}$$

On the receiver side, upon the reception of a packet, we add it to our matrix if the information is *innovative* ; i.e. if the new linear combination increases the rank of our A_r matrix. Let's go over the steps :

1. Reception of C_0 and associated coefficients.
   $A_r = \begin{pmatrix} 3 & 4 & 7 \end{pmatrix}$ $C_r = \begin{pmatrix} 622 \end{pmatrix}$
   Rank of A_r increased from 0 to 1.

2. Reception of C_1 and associated coefficients.
   $A_r = \begin{pmatrix} 3 & 4 & 7 \\ 5 & 8 & 10 \end{pmatrix}$ $C_r = \begin{pmatrix} 622 \\ 1037 \end{pmatrix}$
   Rank of A_r increased from 1 to 2.

3. Reception of C_4 and associated coefficients. **Note : Loss**
   $A_r = \begin{pmatrix} 3 & 4 & 7 \\ 5 & 8 & 10 \\ 5 & 1 & 10 \end{pmatrix}$ $C_r = \begin{pmatrix} 622 \\ 1037 \\ 624 \end{pmatrix}$
   Rank of A_r increased from 2 to 3.

We now have a square matrix of rank 3, we can invert it to recover P :

$$A_r^{-1} = \begin{pmatrix} -2 & \frac{33}{35} & \frac{16}{35} \\ 0 & \frac{1}{7} & \frac{-1}{7} \\ 1 & \frac{-17}{35} & \frac{-4}{35} \end{pmatrix} ; \text{and so } P_r = A_r^{-1} * C_r = \begin{pmatrix} 19 \\ 59 \\ 47 \end{pmatrix}$$

This method is a really efficient way of sending the correct amount of data, when you are confronted to frame losses in your channel.

2 Application over a finite set

Now that we got valid operations for the infinite setintegers, we need something for working with computers, i.e. use a finite set. We are going to use bytes as base symbol ; so we're working in the field $\mathbb{F}_{2^8}$.

2.1 Defining Operations

We need to define our standard set of operations :

- **Addition** : bitwise XOR
- **Subtraction** : bitwise XOR
- **Multiplication** : two possibilities
  - Use polynomials : Choose an irreductible polynomial for this field (for $\mathbb{F}_{2^8}$, we can use Rijndael's : $x^8 + x^4 + x^3 + x + 1$).
    Then, you can multiply the two terms, represented as polynomials and perform a modulo(Rijndael's polynomial) on the result
  - Use discrete logarithm : as a faster alternative, we can compute tables for logarithm and exponentiation. *In $\mathbb{F}_{2^8}$, we end up with two 255 bytes tables.*
    Let α be a generator in our field (for example α = 0x03). For each non-zero number x in the field, we have $x = \alpha^{l(x)}$, with $l()$ our logarithm function.
    Since $l(xz) = l(x) + l(y)$, we can compute xy very easily by simply looking up at our tables.
- **Division** : two possibilities
  - Euclidian algorithm
  - Discrete logarithms : as for the multiplication, we have $l(\frac{x}{y}) = l(x) - l(y)$. It's straightforward to use our pre-computed tables.

2.2 Example

I wrote a little piece of software to compute finite field operations using resources from `http://www.samiam.org/galois.html`. I added matrices functions to generate them and perform Gauss-Jordan elimination. All the following example is performed over the field $\mathbb{F}_{2^8}$; data is represented using one byte (`uint8_t`).
Using the same naming conventions as for the Integer examples, let's generate our first matrices randomly :

$$P_s = \begin{pmatrix} 67 \\ c6 \\ 69 \end{pmatrix} \; A_s = \begin{pmatrix} 73 & 51 & ff \\ 4a & ec & 29 \\ cd & ba & ab \\ f2 & fb & e3 \\ 46 & 7c & c2 \end{pmatrix}$$

$$\text{We can compute encoded packets : } C_s = \begin{pmatrix} 3b \\ 9b \\ 2d \\ b3 \\ 79 \end{pmatrix}$$

I implemented a function `loss()` that randomly remove rows from both C and A matrices. Hence, we can test various parameters for packet loss. Here, I set a loss of 10%, which led to the removal of 1 packet out of the 5 given.

On the receiver side, we now have 4 packets. We drop the last one, which does not provide any additional information. Here are the 3 encode packets we kept, and their coefficients.

$$C_r = \begin{pmatrix} 3b \\ 9b \\ b3 \end{pmatrix} \; A_r = \begin{pmatrix} 73 & 51 & ff \\ 4a & ec & 29 \\ f2 & fb & e3 \end{pmatrix}$$

$$\text{We can compute } A_r^{-1} = \begin{pmatrix} c & 6e & d7 \\ 22 & 50 & e8 \\ b4 & 66 & eb \end{pmatrix}$$

$$\text{And decode packets } P_r = A_r^{-1} * C_r = \begin{pmatrix} 67 \\ c6 \\ 69 \end{pmatrix}$$