# Codage canal

Avant d'être transmis par le canal, le message a de fortes chances d'être altéré de manière plus ou moins importante à cause de parasites d'origine électrique ou autres. Il est donc nécessaire de prévoir une certaine redondance dans le message ainsi que des procédures pour détecter les erreurs et éventuellement les corriger. C'est l'objectif du codage canal dont nous allons examiner quelques aspects dans ce chapitre.

Alors que le codage source tend à diminuer la place occupée par un message, le codage canal va augmenter celle-ci

## 1. Matrice canal pour un canal binaire symétrique

Un canal de transmission effectue le couplage entre deux alphabets, celui à l'entrée $\{x_i\}$ et celui à la sortie $\{y_i\}$. On suppose que cet alphabet est sans mémoire et on va s'intéresser à une transmission binaire où chaque alphabet est $\{0,1\}$.

Si la transmission n'est pas bruitée ou perturbée, il n'y a pas d'erreur de transmission. Si le canal est bruité, un 0 peut donner un 1 et inversement. On a le schéma de la figure 1. Pour caractériser les propriétés du canal on a besoin des probabilités conditionnelles $p(y_j|x_i)$, c'est-à-dire des probabilités d'obtenir $y_j$ sachant que $x_i$ est réalisé. Ceci conduit à introduire la **matrice canal** ou **matrice de transmission** suivante :

$$P(Y|X)=\begin{pmatrix} p(y_1|x_1) & p(y_1|x_2) \\ p(y_2|x_1) & p(y_2|x_2) \end{pmatrix}$$

Les éléments de la matrice de transition satisfont à :

$$p(y_1)=p(y_1|x_1)p(x_1)+p(y_1|x_2)p(x_2)$$
$$p(y_2)=p(y_2|x_1)p(x_1)+p(y_2|x_2)p(x_2)$$
et
$$p(y_1|x_i)+p(y_2|x_i)=1$$

Les probabilités $p(y_i)$ sont les **probabilités marginales**. La matrice de transition permet d'obtenir les $p(y_i)$ à partir des $p(x_i)$.

$$\begin{pmatrix} p(y_1) \\ p(y_2) \end{pmatrix}=\begin{pmatrix} p(y_1|x_1) & p(y_1|x_2) \\ p(y_2|x_1) & p(y_2|x_2) \end{pmatrix}\begin{pmatrix} p(x_1) \\ p(x_2) \end{pmatrix}$$

On va considérer un canal symétrique ce qui veut dire que la matrice de transition est symétrique. S'il n'y a pas de bruit sur le canal on a :

$p(y_j|x_i)=\delta_{ij}$ c'est-à-dire $p(y_j|x_i)=1$ si $i=j$ et $p(y_j|x_i)=0$ si $i\neq j$. La matrice de transmission est diagonale. Elle vaut :

$$P(Y|X)=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$

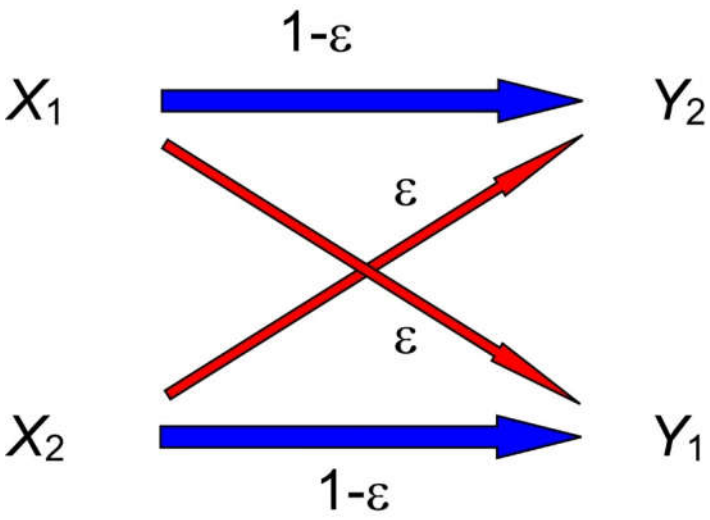


*Figure 1*

Si le canal est bruité, il y aura des éléments non diagonaux dans la matrice de transition. Si $p(y_{j\neq i}|x_i)=\varepsilon$ avec $0\leq\varepsilon\leq 1$, alors $p(y_i|x_i)=1-\varepsilon$. La matrice de transition devient :

$$P(Y|X)=\begin{pmatrix}1-\varepsilon & \varepsilon\\ \varepsilon & 1-\varepsilon\end{pmatrix}$$

Si $\varepsilon$ est petit, la transmission est bonne et il y a peu d'erreur. Si $\varepsilon=0{,}5$, on est dans le cas où y a tellement d'erreurs dans la transmission que les symboles 0 ou 1 apparaissent au hasard. Si $\varepsilon>0{,}5$ un bit à une plus grande chance d'être inversé que d'arriver correctement et pour $\varepsilon=1$ le bit 0 donne 1 et le bit 1 donne 0.

## 2. Capacité d'un canal

Considérons une source $S$ caractérisée par son entropie $H(S)$ et un débit de symboles par seconde $D_S$. Le taux d'émission moyen $T$ de la source est donné par :

$$T=H(S)D_S$$

Ce taux d'émission est la quantité d'information que la source émet en moyenne par seconde. Si maintenant on veut transmettre cette information via un canal, il faut que celui-ci ait la capacité d'accepter ce débit. On est exactement dans le cas d'un robinet d'eau (source) dont l'eau coule dans une baignoire (qui joue le rôle de buffer) dont l'évacuation (canal) est ouverte. Si le débit du robinet est faible, toute l'eau va s'évacuer immédiatement. S'il est trop fort, le débit d'évacuation n'est pas suffisant et la baignoire va se remplir puis déborder.

Pour le cas qui nous intéresse, il y a une source en entrée du canal, $X$, et une source en sortie, $Y$. Il faut comparer la similitude qui existe entre ces deux sources. Si elles sont identiques, la transmission est parfaite. Pour évaluer la qualité de la transmissions on peut utiliser la quantité d'information mutuelle $I(X,Y)$ que nous avons définie précédemment. Elle représente la quantité d'information partagée entre les deux sources. Rappelons qu'elle est définie par :

$$I(X,Y) = I(Y,X) = \sum_{i,j} p(x_i, y_j) \ln_2 \left( \frac{p(x_i, y_j)}{p(x_i) p(y_j)} \right)$$

Avec la loi de Bayles, la probabilité conjointe $p(x_i, y_j)$ peut s'écrire :

$$p(x_i, y_j) = p(y_j | x_i) p(x_i) = p(x_i | y_j) p(y_j)$$

Ce qui permet d'écrire $I(X,Y)$ comme :

$$I(X,Y) = \sum_{i,j} p(y_j | x_i) p(x_i) \ln_2 \left( \frac{p(x_i | y_j)}{p(x_i)} \right)$$

Faisant ainsi apparaître les probabilités marginales $p(x_i)$ qui ne dépendent que de la source et du système de codage utilisé. L'objectif est de rendre maximal l'information partagée entre l'entrée et la sortie du canal. La capacité du canal est définie comme :

$$C = \max I(X,Y)$$

Le maximum étant pris pour toutes les sources possibles $p(x_i)$.

Si l'on veut transmettre $N$ symboles, on doit avoir un capacité canal de :

$$C = T = H(N) D_C = \ln_2(N) D_C$$

- Si $T > C$ la transmission de l'intégralité de l'information est impossible (débit du robinet supérieur à l'évacuation). Dans ce cas il faut changer de type de canal (augmenter le diamètre du tuyau d'évacuation) ou réduire la quantité d'information à transmettre en utilisant un codage ayant une meilleure compression ou avec perte d'information (réduire le débit du robinet d'arrivée).
- Si $T \leq C$ la transmission de l'intégralité de l'information est possible.

La transmission par le canal dépend aussi fortement du codage choisi. Un codage utilisant des mots de codes plus longs que nécessaire peut saturer le canal.

## 3. Canal bruité

Supposons maintenant, de manière plus générale, que l'on ait en entrée une source $X$ caractérisée par un alphabet $\{x_1, x_2 \dots x_N\}$ et que l'on ait en sortie $Y$ caractérisée par un alphabet $\{y_1, y_2 \dots y_N\}$. Comme pour le cas binaire, le canal bruité est caractérisé par une **matrice de perturbations** $P$ dont les éléments de matrice sont les probabilités conditionnelles $p(y_i | x_i)$ :

$$[P]_{ij} = p(y_i | x_j)$$

Une bonne transmission nécessite que l'information mutuelle soit maximale. Les différentes entropies permettent de caractériser le canal :

- $H(X)$ est l'entropie qu'aurait le canal en l'absence de bruit.

- $H(X|Y)$ représente la perte d'information due au bruit. C'est la quantité d'information non transmise.
- $I(X,Y)$ mesure l'entropie réellement transmise par le canal bruité.

L'objectif est de maximiser $I(X,Y)$

## 4. Erreurs de transmission

Dans une transmission binaire de l'information, les messages sont une succession de 0 et de 1. Des erreurs peuvent se produire lors du processus de transmission et, en sortie de canal, le message peut ne pas être identique à celui qui était en entrée de canal. Lors d'une transmission, on peut avoir un remplacement d'un 0 par 1 ou inversement. Un taux d'erreur de $10^{-6}$ avec une connexion de 1 Mo/s donne environ 8 bits erronés par seconde ce qui est important.

On peut corriger les erreurs lors de la transmission comme c'est le cas pour le protocole TCP, par exemple. Il est souvent nécessaire de corriger rapidement les erreurs et donc d'opérer en temps réel dans le cas de support comme les CD et DVD qui peuvent avoir des rayures perturbant fortement leur lecture.

Pour éviter les erreurs, on pourrait penser agir sur le canal de transmission pour qu'il soit de meilleure qualité. C'est toutefois difficile et coûteux. De plus, il y aura toujours des erreurs donc le problème ne sera pas résolu. La stratégie adoptée est donc de détecter et corriger les erreurs en utilisant des codes correcteurs. Il faut néanmoins prendre en compte un certain nombre de contraintes pour leur utilisation pratique qui sont :

- Un faible coût
- Pas ou une très faible altération de la vitesse de transmission
- Une excellente fiabilité

Une fois l'erreur détectée, on peut :

- Retransmettre le message mais ce n'est pas toujours possible et peut générer de nouvelles erreurs
- Corriger l'erreur en utilisant la théorie des codes

Voilà comment peut se glisser une erreur dans la transmission de la séquence binaire suivante :

Message original ⇒ 00101100110000000101001001001001001001001011
Message reçu ⇒ 00101000110000010101001001101001001001001011

Le but de la théorie des codes est de trouver des méthodes de codage permettant de mieux détecter et corriger les erreurs de transmission. Ils appartiennent à deux grandes familles :

Les **codes détecteurs** signalent s'il y a eu une erreur de transmission. Le destinataire peut alors rejeter le symbole altéré et, si le canal est bidirectionnel, il peut envoyer un message à la source pour lui demander une retransmission.

Une altération d'un ou plusieurs symboles d'un message ne le rend pas nécessairement illisible. Voyons ceci sur un exemple.

Message initial : « On se donne rendez-vous demain à 15H40 au café de la place »

Message avec erreur : « On se don**X**e rend**X**z-vo**X**s demain à 15H40 au café **X**e la place »

Message avec erreur : « On se **X**onne rendez-vous demain à 1**X**H40 au café de la place »

Le premier message avec erreur est compréhensible et donne une information exploitable alors que le second, qui contient moins d'erreurs, ne donne pas l'information importante qui est l'heure du rendez-vous.

Les codes correcteurs permettent de détecter des erreurs mais sont aussi capable de les corriger avec une bonne probabilité. Il sont utilisés lors de transmission de données, de lecture de celles-ci (CD ou DVD) ou d'écriture sur un support (CD,DVD, mémoire, etc.)

Prenons le cas de la transmission d'un numéro de téléphone par sms (tableau 1). Si l'on tape les chiffres et qu'on se trompe sur l'un d'entre eux, l'information reçue sera complètement erronée et inutilisable. On a codé le numéro de téléphone au plus juste. Si maintenant on transmet ce numéro de téléphone en tapant les mots correspondants (dans un mail par exemple), comme on peut le voir dans le tableau 1, le numéro est toujours déchiffrable malgré la présence de plusieurs erreurs. Le texte a permis de mettre un excédent d'information.

| **Message transmis** | **Message reçu** |
|---|---|
| 0612708535 | 06**9**7085**2**5 |
| Zéro six douze sept zéro quatre-vingt-cinq trois cinq | Zéro slx doze soicante dix-huit cinq troi cinq |

*Tableau 1*

Donc :

- Si l'information est concise, elle sera difficile à corriger
- Si l'information est redondante, la détection des erreurs et leur correction est plus facile. Parfois le contexte aide à corriger les erreurs.

Pour introduire de la redondance au téléphone ou à la radio, on peut procéder de la manière suivante :

- On répète le numéro
- Pour un nom propre on épèle le mot en utilisant parfois « A comme Antoine, B comme Brigitte, etc. »
- En radio on peut utiliser l'alphabet universel (Alpha, Bravo, Charlie Delta, etc.)

## 5. L'opérateur XOR

Quand on s'intéresse au codage binaire, on part d'un message source de $n$ bits qui est en général la traduction brute d'un message - qui peut par exemple être alphanumérique - et on génère un message de $m$ bits ($m \geq n$) pour gérer les erreurs qui peuvent se produire lors de la transmission.

L'algèbre booléenne est très utilisée dans le traitement des données binaires, en particulier l'opérateur XOR ($\oplus$) pour le codage et la correction d'erreurs. Les opérations de base sont les suivantes :

$$0 \oplus 0 = 0 \quad 0 \oplus 1 = 1 \quad 1 \oplus 0 = 1 \quad 1 \oplus 1 = 0$$

XOR effectue une addition sans retenue ou **addition modulo 2**. Les propriétés générales de l'opérateur $\oplus$ sont les suivantes :

$$a \oplus b = b \oplus a$$
$$a \oplus (b \oplus c) = (a \oplus b) \oplus c$$
$$a \oplus 0 = a$$
$$a \oplus a = 0$$
$$a \oplus b = c \quad \Leftrightarrow \quad a \oplus c = b \quad \Leftrightarrow \quad b \oplus c = a$$

exemple : si $a = 1$ et $b = 0$
$1 \oplus 0 = 1$ $(c = 1)$ $1 \oplus 1 = 0$ $(b = 0)$ et $0 \oplus 1 = 1$ $(a)$

Considérons deux nombres binaires $a = a_1 a_2 \ldots a_n$ et $b = b_1 b_2 \ldots b_n$. La somme modulo 2 est le mot binaire dans lequel on fait pour le bit $i$ : $a_i \oplus b_i$.

$$a = 001101011001$$
$$b = 010100110011$$
$$\text{----------}$$
$$a \oplus b = 011001101010$$

Le **poids** $W$ d'un mot binaire est le nombre de bits égaux à 1. Par exemple

$$W(011001101010) = 6$$

$a \oplus b$ indique les endroits où $a$ et $b$ diffèrent. Le poids $W(a \oplus b)$ indique le nombre de différences entre $a$ et $b$ .

## 6. Redondance par itération

Une manière simple mais coûteuse de prévenir les erreurs est de transmettre plusieurs fois le même message. Par exemple si on veut envoyer 0100110, on peut envoyer 3 fois ce mot : 0100110 0100110 0100110. Si par exemple on reçoit deux mots identiques et que le troisième est différent, il y a une forte probabilité pour que le message exact soit celui donné par les deux mots identiques.

C'est cette méthode est utilisée quand on donne son numéro de téléphone. On le répète pour que l'interlocuteur vérifie qu'il a bien noté les bons chiffres.

## 7. Contrôle de parité

Le contrôle de parité consiste à ajouter à $n$ bits (mot de code) un bit supplémentaire indiquant la parité des $n$ bits. Dans le cas du choix d'un parité paire, le bit ajouté est choisi de telle manière que la somme totale des bits soit paire. Dans le cas du choix d'un parité impaire c'est le contraire.

Nous allons considérer ici un contrôle de parité pair. Une autre façon de présenter les choses est de dire que lorsque le nombre de bits 1 dans le mot de code est pair on met zéro dans le bit de parité et 1 lorsque le mot de code est impair.

Le contrôle de parité est aussi appelé VRC (Vertical Redundancy Check ou Vertical Redundancy Checking).

Voyons ceci sur un exemple. Nous allons considérer un mot de code de 7 bits auquel on ajoute un bit de parité pour obtenir un octet, qui est la structure de base utilisée dans les ordinateurs modernes[9]. Prenons deux mots de code différents (tableau 2) :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Mot A | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| Mot B | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

*Tableau 2*

Le bit de parité sera de 0 pour le mot A et de 1 pour le mot B ce qui donne le tableau 3.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Mot A | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| Mot B | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

*Tableau 3*

Le message envoyé par le codeur sera alors celui-ci, incluant le bit de parité. À la sortie du canal, le décodeur effectue un test de parité du mot reçu. Nous avons mis le bit de parité à droite des autres bits mais on peut tout aussi bien le mettre à gauche. Il suffit que l'émetteur et le destinataire aient fait un choix commun.

Si l'on reçoit (tableau 4) :

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Mot A | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| Mot B | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

*Tableau 4*

[9] Les ordinateurs font leurs opérations sur des octets. Les ordinateurs UNIVAC travaillaient par exemple avec des mots de 9 bits au lieu des 8 bits (octets) utilisés par IBM. Cela présentait parfois des avantages. En effet un nombre réel (REAL, simple précision) est codé avec 4 octets, soit 32 bits alors que chez UNIVAC il était codé avec 36 bits. Un calcul numérique effectué avec 36 bits est plus précis qu'avec 32 bits. Dans certains cas il était possible de faire les calculs en simple précision avec un UNIVAC alors qu'il fallait travailler en double précision avec IBM (64 bits, 8 octets).

On sait que le mot A est sans doute correct mais pas le mot B. Si le canal est bidirectionnel, on peut demander à l'émetteur de renouveler l'envoi sinon le message est rejeté.

Avec le contrôle de parité on peut détecter une erreur mais pas la corriger. De plus, s'il y a deux erreurs, comme c'est le cas pour la deuxième ligne du tableau 5, le message apparaîtra comme correct alors qu'il est faux.

| Mot envoyé | 0 | 1 | 1 | 0 | 1 | 0 | 0 | **1** |
|---|---|---|---|---|---|---|---|---|
| Mot B reçu | 0 | 1 | 1 | 0 | **0** | 0 | 0 | **0** |
| Mot B reçu | **1** | 1 | 1 | 0 | **0** | 1 | 1 | **1** |

*Tableau 5*

Le système de contrôle de parité ne permet de détecter qu'un nombre impair d'erreurs et il ne les corrige pas.

Le code ASCII, très utilisé en informatique, est un code de longueur fixe sur 7 bits auxquels on peut ajouter un bit de parité. Voici un exemple dans lequel les deux conventions de parité sont utilisées (tableau 6) :

| Lettre | Code ASCII | Parité paire | Parité impaire |
|---|---|---|---|
| O | 1001111 | 10011111 | 10011110 |
| U | 1010101 | 10101010 | 10101011 |
| I | 1001001 | 10010011 | 10010010 |

*Tableau 6*

Le contrôle de parité croisé ou LRC (Longitudinal Redundancy Check) consiste à ne pas seulement contrôler un caractère (plusieurs bits) mais un bloc de caractères dans deux dimensions. Voyons cela sur l'exemple ci-dessus dans le cas d'un choix de parité paire (tableau 7).

| Lettre | Code ASCII | LRC |
|---|---|---|
| O | 1 0 0 1 1 1 1 | 1 |
| U | 1 0 1 0 1 0 1 | 0 |
| I | 1 0 0 1 0 0 1 | 1 |
| VRC | 1 0 1 0 0 1 1 | |

*Tableau 7*

## 8. Polynômes modulo 2

À un nombre binaire $a_0 a_1 \ldots a_{k-1} a_k$ constitué de $k$ chiffres $a_i = 0$ ou $1$ ($a_0 \neq 0$), on peut associer un polynôme de la forme :

$$a_0 x^k + a_1 x^{k-1} + \ldots + a_{k-1} x + a_k$$

où $x$ est la variable et $a_i$ les coefficients du polynôme.

Ce type de polynôme, où les coefficients sont définis modulo 2, est utilisé dans le domaine du codage des erreurs comme nous le verrons dans la section suivante. On peut définir des opérations élémentaires (addition, soustraction, multiplication et division) entre deux polynômes en effectuant des opérations modulo 2. Soit par exemple :

$P_1 = x^3 + x + 1$ et $P_2 = x^2 + x$

$P_1 + P_2 = x^3 + x^2 + 1$

Le schéma de l'addition est indiqué dans la figure 2.

$$\begin{array}{r} x^3 \oplus x \oplus 1 \\ x^2 \oplus x \\ \hline x^3 \oplus x^2 \oplus 1 \end{array}$$

*Figure 2*

Si l'on n'avait pas effectué l'addition modulo 2, on aurait trouvé :

$$P_1 + P_2 = x^3 + x^2 + 2x + 1$$

Mais $(x^3 + x^2 + 2x + 1) \bmod 2 = x^3 + x^2 + 1$

Pour la multiplication on a :

$$P_1 \times P_2 = x^5 + x^4 + x^3 + x$$

Le principe est indiqué sur la figure 3.

$$\begin{array}{r} x^3 \oplus x \oplus 1 \\ x^2 \oplus x \\ \hline x^4 \oplus x^2 \oplus x \\ x^5 \oplus x^3 \oplus x^2 \\ \hline x^5 \oplus x^4 \oplus x^3 \oplus x \end{array}$$

*Figure 3*

Si l'on n'avait pas effectué la multiplication modulo 2 on aurait trouvé :

$$P_1 \times P_2 = x^5 + x^4 + x^3 + 2x^2 + x$$

Mais $(x^5 + x^4 + x^3 + 2x^2 + x) \bmod 2 = x^5 + x^4 + x^3 + x$

Pour la division, on procède comme la figure 4. On obtient :

$$\text{Reste}\left(\frac{P_1}{P_2}\right) = 1$$

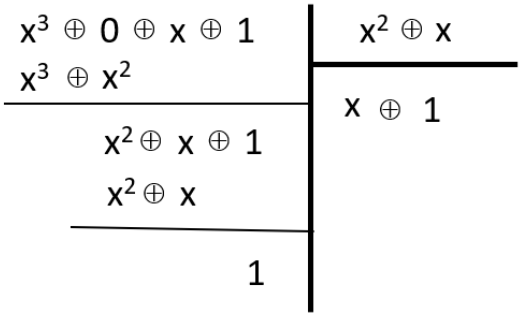


*Figure 4*

Si l'on avait effectué la division normale des deux polynômes on aurait trouvé :

$$\frac{P_1}{P_2} = (x^2 + x)(x-1) + 2x + 1 \text{ mais } 2x + 1 \bmod 2 = 1$$

## 9. Les codes cycliques (CRC)

Le code CRC (Code de redondance Cyclique ou Cyclic Redundancy Check) est la principale méthode utilisée pour contrôler l'intégrité des données dans les télécommunications. Ils travaillent sur des trames (frames en anglais) qui sont des ensembles de bits. Cette méthode permet de détecter des erreurs. Ils sont faciles à mettre en œuvre matériellement.

Le code CRC utilise un polynôme générateur $G(x)$ qui considère que toute information de $n$ bits (trame) peut être écrite sous forme polynomiale. Par exemple, à 10101 on peut associer le polynôme :

$$1 \times x^4 + 0 \times x^3 + 1 \times x^2 + 0 \times x^1 + 1 \times x^0 = x^4 + x^2 + x^0$$

L'algorithme CRC permet de déterminer le code qu'il faudra concaténer à la trame originale avant de la transmettre au canal . Pour cela on procède de la manière suivante :

- On détermine le polynôme $M(x)$ associé à la trame
- On multiplie $M(x)$ par $x^r$ où $r$ est le degré du polynôme générateur $G(x)$.
- On calcule le reste $R(x)$ de la division du polynôme $M(x) \times x^r$ par $G(x)$ modulo 2.
- Le mot de code est alors $M(x) \times x^r + R(x)$ (concaténation) que l'on transmet.

Pour le décodage on procède de la manière suivante :

- On détermine le polynôme $M'(x)$ associé à la trame reçue
- On calcule reste $R'(x)$ de la division du polynôme $M'(x)$ par $G(x)$.
- Si $R'(x)=0$ il n'y a pas d'erreur. Si $R'(x)\neq 0$ il y a erreur et on demande la retransmission du message.

### *Exemple 1*

Votons ceci sur un exemple très simple. On veut coder le message *M*=10110 avec le polynôme générateur :

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

Le polynôme associé au message est :

$$M(x)=x^4+x^2+x$$

On a :

$$x^2M(x)=x^6+x^4+x^2$$

La division de $x^2M(x)$ par $G(x)$. donne $-x$ c'est-à-dire $x$ modulo 2.

Le monôme $x$ correspond à 10. Le message à envoyer est donc

10110**10**

Le destinataire du message associe à celui-ci le polynôme :

$$x^6+x^4+x^3+x$$

Il vérifie que le reste de la division est 0 sinon cela veut dire qu'il y a une erreur et le contenu du message reçu doit être rejeté.

### *Exemple 2*

Prenons comme autre exemple le message de 10 bits :

$$M=1101011011$$

Avec le polynôme générateur $G(x)=x^4+x+1$, qui correspond à 1011, le polynôme associé au message est :

$$M(x)=x^9+x^8+x^6+x^4+x^3+x+1$$

Multiplier par $x^4$ revient à ajouter 4 bits à $M \rightarrow M'$ ce qui donne au niveau polynômial :

$$x^4M(x)=x^{13}+x^{12}+x^{10}+x^8+x^7+x^5+x^4$$

Le reste de la division de $x^4M(x)$ par $G(x)$ est :

$$-3x^2-5x^2-3x$$

Ce qui donne modulo 2

$$(-3x^2 - 5x^2 - 3x) \bmod 2 = x^3 + x^2 + x$$

Les 4 bits supplémentaires qui constituent le code CRC sont donc :

1110

Le message à transmettre est :

11010110111110

Le destinataire l'associera au polynôme :

$$x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + x$$

Il vérifiera que le reste de la division de ce polynôme par $G(x)$ est nul. Le reste de la division est en effet $-2x^3 - 4x^2 - 2x$ ce qui donne 0 modulo 2.

On peut aussi obtenir le reste de la division comme indiqué sur la figure 5. Cette méthode est plus simple à mettre en œuvre que la division explicite de polynômes.

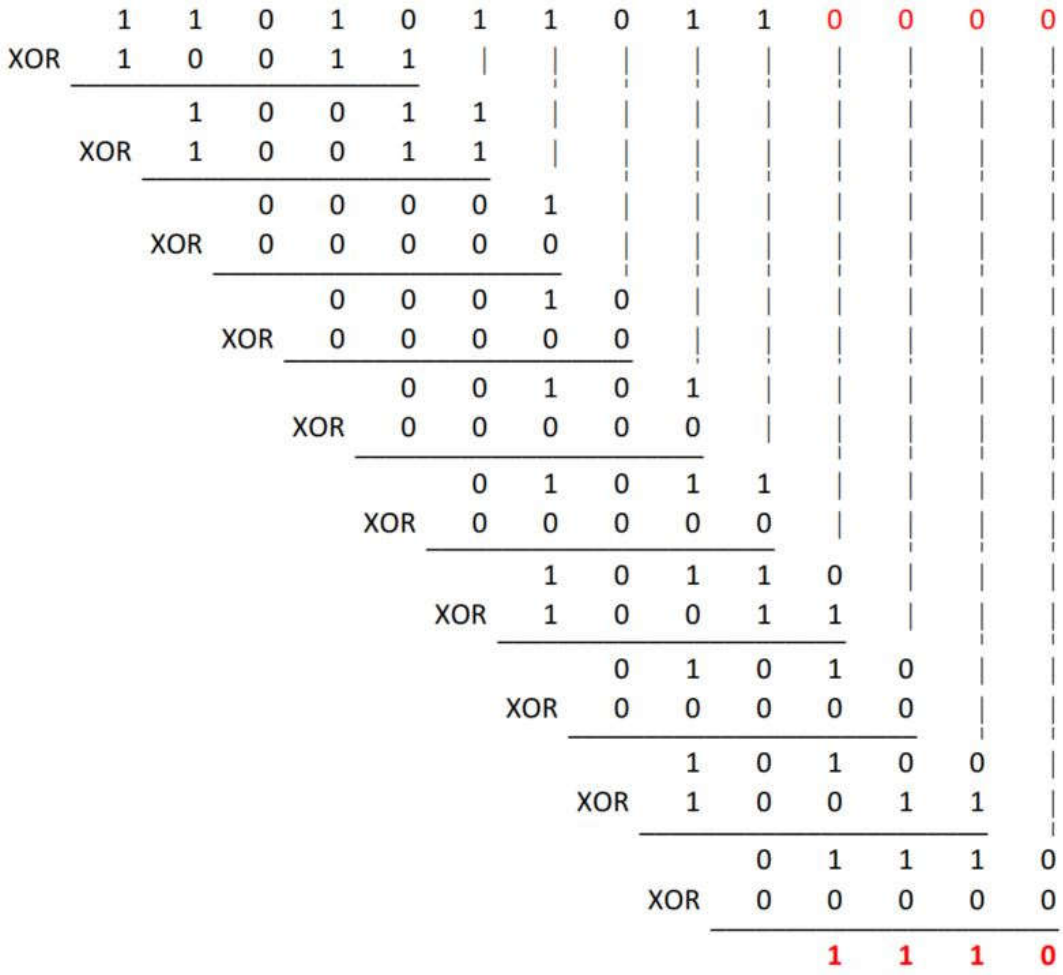


*Figure 5*

Le reste est 1110 et la trame à transmettre est donc :

$$11010110111110$$

Comme on l'a trouvé plus haut.

Parmi les polynômes employés citons les

CRC-16 : $x^{16}+x^{15}+x^2+1$

CRC-32 (Ethernet) : $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

## 10. Distance de Hamming

Le poids de Hamming $W_{\mathrm{H}}$ d'un nombre binaire est le nombre de bits non nuls. Par exemple :

$$W_{\mathrm{H}}(10001)=2 \;\; ; \;\; W_{\mathrm{H}}(00000)=0 \;\; ; \;\; W_{\mathrm{H}}(11111)=5$$

La distance de Hamming $d_{\mathrm{H}}$ entre deux nombres binaires de même longueur est égale au nombre de bits différents entre ces deux mots. Par exemple :

$d_{\mathrm{H}}(10110,10010)=1$ bit
$d_{\mathrm{H}}(1010,0101)=4$ bits
$d_{\mathrm{H}}(10,100)$ ne peut être définie (les mots n'ont pas la même longueur)

La distance de Hamming est une distance au sens mathématique du terme car, $\forall a,b$, elle satisfait :

$d_{\mathrm{H}}(a,b)=d_{\mathrm{H}}(b,a)$ (symétrie)
$d(a,b)\geq 0$ (positivité)
$d_{\mathrm{H}}(a,b)=0 \;\Leftrightarrow\; a=b$
$d_{\mathrm{H}}(a,c)\geq d_{\mathrm{H}}(a,b)+d_{\mathrm{H}}(b,c)$ (inégalité du triangle)

La distance de Hamming de deux nombres binaires est égale au poids de leur somme modulo 2 (ou exclusif, ou XOR) :

$$d_{\mathrm{H}}(u_1,u_2)=W_{\mathrm{H}}(u_1\oplus u_2)$$

Par exemple :

$$d_{\mathrm{H}}(10010,01010)=2$$
$$W_{\mathrm{H}}(10010\oplus 01010)=W_{\mathrm{H}}(11000)=2$$

Considérons les 4 mots de code suivants à transmettre par le canal :

$$c_1=00000$$
$$c_2=01110$$
$$c_3=10101$$
$$c_4=11011$$

La distance de Hamming entre ces mots est donnée pour chaque couple dans le tableau 8 :

| | $c_1$ | $c_2$ | $c_3$ | $c_4$ |
|---|---|---|---|---|
| $c_1$ | 0 | 3 | 3 | 4 |
| $c_2$ | 3 | 0 | 4 | 3 |
| $c_3$ | 3 | 4 | 0 | 3 |
| $c_4$ | 4 | 3 | 3 | 0 |

*Tableau 8*

On voit, à partir du tableau 8, que la distance minimale de Hamming est $d_{\mathrm{H}} = 3$ bits.

Supposons maintenant que le message reçu soit $01100$. Alors les distances de Hamming sont les suivantes :

$$d_{\mathrm{H}}(00000, 01100) = 2 \text{ bits}$$
$$d_{\mathrm{H}}(01110, 01100) = 1 \text{ bit}$$
$$d_{\mathrm{H}}(10101, 01100) = 3 \text{ bits}$$
$$d_{\mathrm{H}}(11011, 01100) = 4 \text{ bits}$$

On va prendre pour le mot exact $c_2$ qui est à une distance de Hamming de 1 bit. S'il y avait eu 2 bits d'erreur, et que $c_1$ soit le bon code, on n'aurait pas bien corrigé l'erreur et le résultat de la transmission aurait été faux.

De manière générale, un code binaire de distance minimale de Hamming $d_{\mathrm{H}}$ permet :

- Si $d_{\mathrm{H}} = 2n_e$ de détecter $n_e$ erreurs et d'en corriger $n_e - 1$
- Si $d_{\mathrm{H}} = 2n_e + 1$ de détecter et de corriger $n_e$ erreurs.

On peut donc en général corriger $\frac{d_{\mathrm{H}} - 1}{2}$ erreurs.

Dans l'exemple ci-dessus, $d_{\mathrm{H}} = 3$, donc on peut corriger 1 erreur.

## Code de Hamming : exemple

Le principe du code de Hamming est d'introduire des bits supplémentaires par rapport à ceux nécessaires à la transmission du message pour détecter une erreur sur un seul bit et corriger celle-ci. On peut expliciter ce mécanisme sur un exemple. Nous verrons, dans la prochaine section une représentation plus générale de ce codage. Nous avons choisi pour cela un exemple présenté dans la vidéo suivante :

https://www.youtube.com/watch?v=373FUw-2U2k

Nous conseillons vivement au lecteur de la regarder car elle est d'excellent qualité et explicite ce qui va être donné ci-dessous sous forme résumée.

Supposons que l'on veuille coder le mot 10011010 (figure 6). Cela correspond à 8 bits. Pour pouvoir détecter et corriger une erreur on ajoute des bits redondants qui vont jouer le rôle de bit de parité. Leur position sera une puissance de 2, donc 1, 2, 4, 8, 16…

On va coder le message sur 12 bits, 8 étant réservés au message et 4 aux bits de parité. Chacun de ces bits va calculer la parité d'un certain nombre de bits.

Le bit 1 ($p_1$) va tester la parité des bits impairs du mot que l'on veut transmettre, c'est-à-dire les bits 1, 3, 5, 7, 9, 11… c'est-à-dire les bits $d_3$, $d_5$, $d_7$, $d_9$ et $d_{11}$. Le bit p1 est pour l'exemple qui nous intéresse égal à 0 puisqu'il y a un nombre pair de 1.

Le bit 2 ($p_2$) contrôle la parité des bits de deux bits, en saute 2, en contrôle 2, en saute 2 etc. Donc 2,3,6,7,10,11… soit les bits $d_3$, $d_6$, $d_7$, $d_{10}$ et $d_{11}$. Il y a 3 bits à 1, donc on met 1 pour le bit $p_2$.

Le bit $p_4$ contrôle la parité de 4 bits, en saute 4, en contrôle 4, etc. Donc 4,5,6,7,12,13,14,15… soit d5, d6, d7 et d12. Il y a 1 bit à 1 donc le bit $p_4$ est à 1.

Le bit $p_8$ contrôle la parité de 8 bits, en saute 8, etc. Donc 8,9,10,11,12…. Soit $d_9$, $d_{10}$, $d_{11}$ et $d_{12}$. Il y a 2 bits à 1, donc le bit $p_8$ est à zéro.

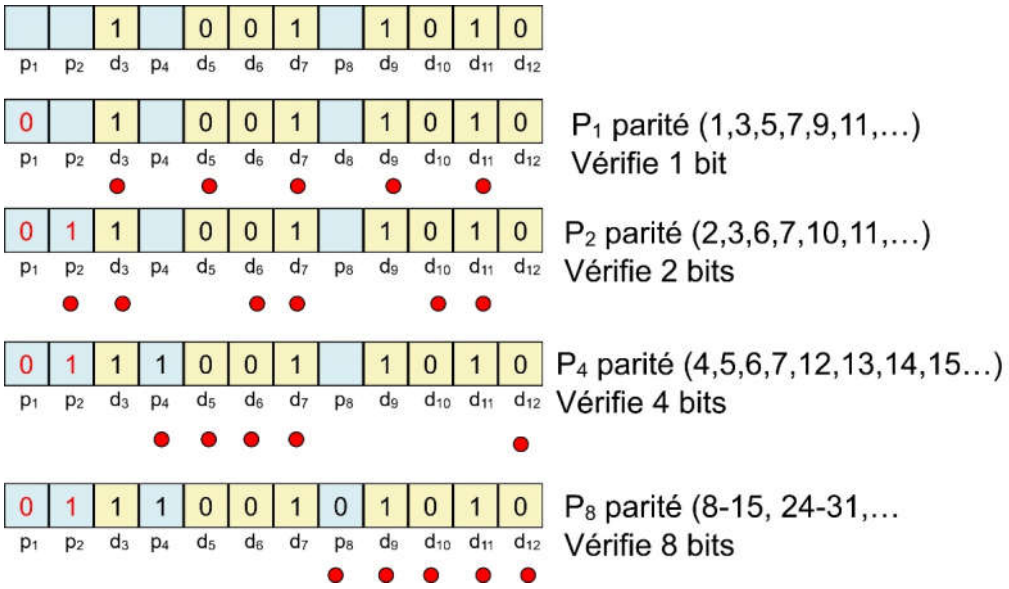


*Figure 6*

Supposons maintenant qu'il y ait une différence entre le code émis et le code reçu (figure 7). Le bit $d_{10}$ a basculé de 0 à 1. Les bits de parité p1 et p4 sont corrects mais pas p2 et p8. Le message reçu a donc une erreur. Le bit incriminé est le bit 2+8=10.

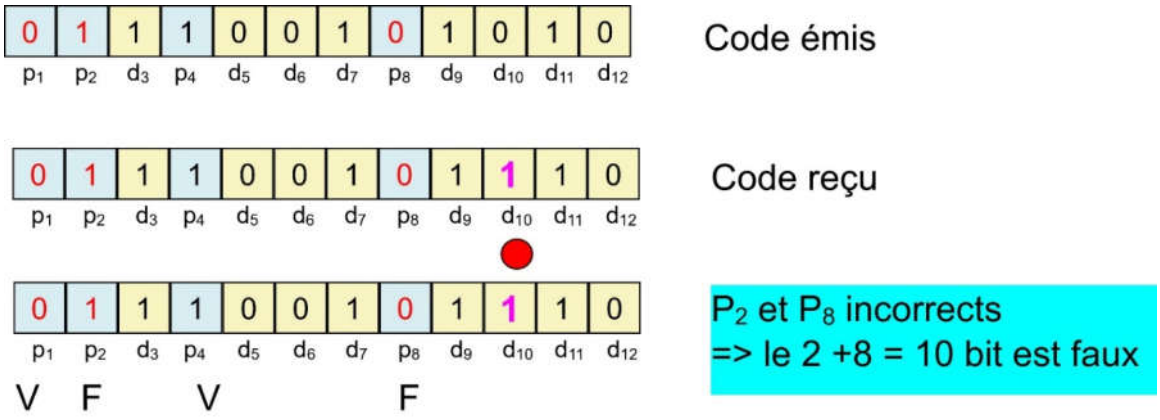


*Figure 7*

## 11. Code de Hamming et diagrammes de Venn

Après cet exemple particulier voyons maintenant une approche utilisant les diagrammes de Venn. Le codage de Hamming est basé sur une redondance des bits qui va permettre de repérer des erreurs et les corriger. Cette méthode permet la correction d'une erreur de transmission sur 1 bit tout en donnant une interprétation géométrique du codage de Hamming.

Prenons pour exemple un message de 4 bits $1101 = m_1 m_2 m_3 m_4$ que l'on veut transmettre en utilisant ma méthode de Hamming. Celle-ci consiste à ajouter 3 bits de parité $p_1 p_2 p_3$ que l'on va définir. Au total 7 bits seront envoyés dans l'ordre suivant :

$$p_1 p_2 m_1 p_3 m_2 m_3 m_4$$

On va utiliser un diagramme de Venn constitué de 3 cercles ayant les parties communes indiquées sue la figure 8. La partie (A) de la figure montre quelles sont les parties attribuées aux bits $m_1 m_2 m_3 m_4$. La partie (B) donne explicitement la valeur des bits. Les bits de parité $p_1 p_2 p_3$ sont positionnés comme indiqué en (C). Ils sont définis de telle manière que la somme dans chaque cercle soit un nombre pair. Cela donne $p_1 = 1, p_2 = 0, p_3 = 0.$

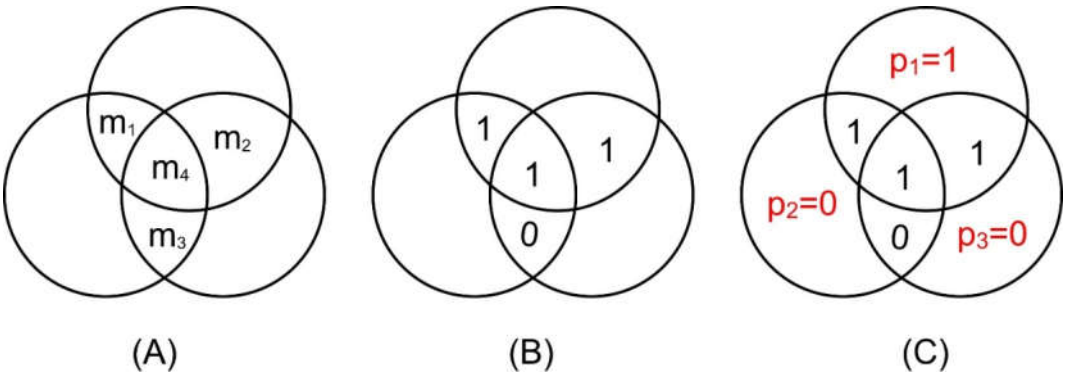


*Figure 8*

Le message envoyé sera donc $1010101$.

Supposons que le message reçu soit $1010001$. Il suffit de faire un contrôle de parité sur chacun des cercles (figure 9). Lors de l'analyse, on voit que le cercle vert donne un contrôle correct de la parité alors que pour les deux cercles rouges ce n'est pas correct. On voit aussi que l'erreur se localise sur $m_2$.

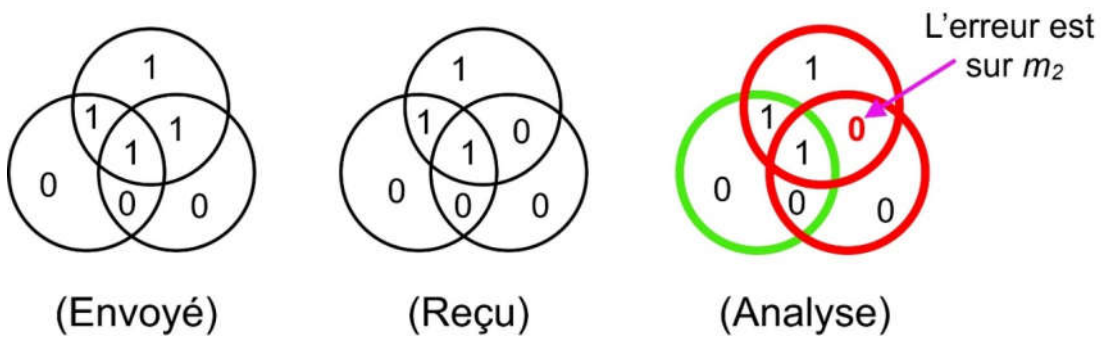


*Figure 9*