# ***Codes Correctifs & Détecteurs d'erreurs***

*Contexte*

On transmet des suites de bits sous forme de paquet sur les réseaux.

Or les sources d'erreur de transmission sont nombreuses et fréquentes (interférence éléctrique, bruit thérmique).

Cela même lors de transmission sur fibre optique.
Dans les réseaux sans fil, les erreurs sont très fréquentes

Les erreurs dans les systèmes informatiques ont tendance à se produire en rafale (bursts).

*But*

mettre en place des mécanismes qui récupèrent ces erreurs.

*Solutions*

- Codes détecteursd'erreur
- Codes correcteurs d'erreur

*Principe de base (commun à toutes les méthodes)*

- rajout d'information redondante (bits) au message à transmettre
- l'émetteur et le récepteur se mettent d'accord sur un algorithme de protection
- l'émetteur calcule l'info. redondante en fonction du message à envoyer et transmet ce message suivi de l'info. redondante
- le récepteur calcule ce que devrait être l'info. redondante en fonction du message qu'il a reçu et le compare à l'info redondante attaché au message.
  Si les 2 info. redondantes concordent => pas d'erreur

(avec une très forte probabilité)

*Enjeux*

- L'info. redondante doit comporter aussi peu de bits que possible tout en protégeant le plus possible.
- L'algo. pour calculer l'info. redondante doit être rapide

*Remarque*

Les techniques que nous allons voir ne s'appliquent pas seulement aux paquets transmis sur un réseau mais aussi aux données stockées en mémoire vive ou sur disque par ex.

## *Les Codes Détecteurs d'Erreurs:*

Cette première stratégie ne permet que de détecter les erreurs et éxige une retransmission si on veut corriger les erreurs

- Exemple le plus simple: le **contrôle de parité** qui consiste à avoir une somme du nombre de bit à 1 (y compris le bit de contrôle) qui soit paire ou impaire selon que l'on soit en parité paire ou impaire.

| | |
|---|---|
| 0101001 ***1*** | parité paire |
| 0101001 ***0*** | parité impaire |

- Le **contrôle de parité à 2 dimensions** est mieux armé pour parer à certaines erreurs en rafales («bursts»).
  Ici, l'idée est de contrôler aussi bien les bits sur les colonnes que sur les lignes.

0101001 ***1***
1101001 ***0***
1011110 1
0001110 *1*
0110100 *1*
1011111 *0*

*1111011 0*

Dans cet exemple, on utilise 14 bits de redondance pour 42 bits de message.

Exercice: montrez que cette technique permet de détecter toutes les erreurs simples, doubles, triples et bcp d'erreurs quadruples.

– Les **codes polynômiaux** sont les plus utilisés dans la pratique. Aussi appelés CRC (Cyclic Redundancy Code)

*L'idée générale:*

L'émetteur a un message à transmettre.

Il rajoute un certains nombre de bits (pour la redondance) à ce message.

Il fait en sorte que le message avec les bits rajoutés soit divisible par une valeur binaire sur laquelle le récepteur et lui-même s'étaient mis d'accord à l'avance. (Ce sont les valeurs des bits rajoutés qui sont à définir pour que le tout soit divisible).

Ainsi, une fois le message transmis, le récepteur n'a plus qu'à refaire la division. Si ce qu'il a reçu est divisible, le message transmis est correct, sinon, il y a erreur.

*Le détail:*

Une trame de bits est considérée comme un polynôme dont les coefficients des termes sont les valeurs des bits de la trame.

Une trame de r bits est considérée comme une suite de r termes (de $x^{r-1}$ à $x^0$).

Un tel polynôme est dit de degré r-1.

Le bit de poids fort est le coefficient du terme $x^{r-1}$.

Exemple: r = 6

Trame 110001

Polynôme $M(x) = 1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0$

$= x^5+x^4+x^0$

L'émetteur et le récepteur doivent se mettre d'accord à l'avance sur un **polynôme générateur G(x)** de degré ***k*** (on trouve des valeurs pour G(x) dans les livres).

L'idée:

Lorsqu'un émetteur veut émettre un message M(x),

- il concatène *k* bits à M(x) (*k* étant le degré de G(x)). Ces *k bits* sont l'information redondante. On appèlera P(x) le message transmis composé de M(x) suivi des *k* bits.
- toute l'astuce consiste à forcer le choix de bonnes valeurs pour les *k* bits, de telle sorte que **P(x) soit divisible par G(x)** (càd. que le reste de la division = 0).
- Si P(x) est transmis et qu'aucune erreur ne se produit lors de la transmission, le récepteur qui divisera P(x) par G(x) obtiendra un reste de 0, signe d'une transmission

correcte. Sinon le reste sera non nul, signe d'une erreur.

La question qui reste ici est de savoir quelles valeurs affecter aux *k* bits.
La solution:

- d'abord rajouter *k* bits à zéro à M(x),
  et appelons cela T(x)
- diviser T(x) par G(x) et trouver le reste qui est la valeur des *k* bits.
- soustraire le reste à T(x) et l'on obtient ce que l'on désire, P(x), les données protégées à émettre
- il doit être évident à ce stade que P(x) **est** divisible par G(x). (dans toutes divisions, ce qui résulte de la soustraction du reste au dividende donne qqch de divisible par le diviseur)

On travaille ici avec ce que l'on appelle **arithmétique polynômiale modulo 2,** dont voici certaines propriétés:

- tout polynôme B(x) peut être divisé par un diviseur polynômial G(x), si B(x) est de degré plus élevé.
- tout polynôme B(x) peut être divisé une fois par un diviseur polynômial G(x), si B(x) est de même degré que G(x). (peut importe si la valeur binaire de G(x) est plus grande que celle de B(x))
- quand on divise une fois, le reste de la division de B(x) par G(x) peut être obtenu par la soustraction de G(x) à B(x).
- pour soustraire G(x) à B(x), il suffit d'appliquer un ou-exclusif (XOR) sur les coéfficients de chaque terme.

*Exemple:*

Soit un polynôme générateur de degré 4 (=> $k = 4$)

$G(x) = 10011 = x^4+x^1+x^0 = x^4+x+1$

Et un message M(x) = 1101011011

On ajoute *k* bits à 0 à la suite des bits de poids faible de M(x) ce qui nous donne T(x).

La division polynomiale de ce nouveau polynôme par G(x) nous donne un reste qui sont les valeurs que l'on recherche pour les k bits (l'information redondante).

1 1 0 1 0 1 1 0 1 1 **0 0 0 0**<-- T(x)  |1 0 0 1 1 <-- G(x)

1 0 0 1 1  1100001010

0 1 0 0 1 1

1 0 0 1 1

0 0 0 0 0 1

0 0 0 0 0

0 0 0 0 1 0

etc ... etc...

reste 0 1 1 1 0

Donc, le reste soustrait à T(x) donne P(x) qui sera effectivement l'info. transmise. Càd.11010110111110.

Le Récepteur contrôle la trame P(x) reçue en divisant P(x) reçue par G(x). Et si le reste = 0 alors il n'y a pas eu d'erreur.

*Puissance de la méthode:*

Un code polynomial de *k* bits de contrôle peut détecter toutes les erreurs de «burst» de longueur <= *k*

Si le «burst» est de longueur *k*+1 alors le reste de la division polynomiale est égal à 0 ssi le «burst» = G(x). Sachant que la caractéristique d'un «burst» est d'avoir son premier et dernier bit à 1 et les autres indéterminés. Les bits à 1 sont les erreurs.

Les codes polynomiaux sont devenus des standards internationaux. Ainsi, parmi les plus utilisés on trouve :

- Le CRC-12 = $x^{12}+x^{11}+x^3+x^1+1$

  Utilisé pour des données de 6 bits

- Le CRC-16 = $X^{16}+x^{15}+x^2+1$

- Le CRC-CCITT = $x^{16}+x^{12}+x^5+1$
  - Les deux précédents pour des données de 8 bits
  - Détecte les «bursts» à 99.998%

Dans la pratique, c'est implanté par du Hardware dans la plus part des cas.

## *Les Codes Correctifs d'Erreurs:*

Cette stratégie prévoit d'introduire suffisamment d'informations redondantes avec les informations à transmettre afin que le récepteur puisse non seulement détecter les erreurs mais aussi déduire ce que les données initiales étaient.

On définira ici un **mot de code** comme une suite de bits composée ***m*** bits de données (le message) et ***k*** bits de contrôle

(l'info. redondante).
La longueur totale du mot de code est ***n*** = m + k.

Etant donné deux mots de code : 10001001 et 10110001, il est possible de déterminer de combien de bits ils diffèrent. Ici 3bits.

Un XOR des deux mots de code est utilisé:

10001001
XOR 10110001
00**111**000 => ici 3 bits à 1 donc **3**

Ce résultat est appelé **distance de Hamming** (1950)

La signification de ce résultat dit que si deux mots de code sont éloignés l'un de l'autre d'une distance de Hamming de d alors il faut **d** erreurs simples (1 bit) pour passer de l'un à l'autre.

Une technique qui permet de corriger les erreurs est l'idée suivante:

- On va faire en sorte que certains mots parmi tous les $2^n$ mots de code possibles ne soit pas valides.
- Si qqn reçoit un de ces mots non valides, c'est qu'une erreur de transmission s'est produite.
- Tout mot erroné se trouvera toujours "coincé" entre 2 mots valides.
- Pour corriger un mot erroné, il suffira alors de comparer sa distance de Hamming avec les 2 mots valides qui lui sont les plus proches.
- Le mot corrigé est alors, parmi ces 2 mots valides, le mot le plus proche du mot erroné.

- Remarque: on ne permet pas tous les $2^n$ mots possibles mais on permet quand même tous les $2^m$ mots.
- Avec cette technique, pour corriger jusqu'à d erreurs, il faut une distance de Hamming de **2d+1** entre chaque mot valide!

Exemple: supposons que les seuls mots valides soient les suivants.

0000000000, 0000011111, 1111100000, et 1111111111

Ces mots ont une distance de Hamming de 5 entre eux.
Les erreurs doubles peuvent donc être corrigées.

Si le mot 0000000111 arrive et si on suppose que 2 erreurs seulement se sont produites, on conclut que le message original est 0000011111.

Par contre on voit que si le message original était 0000000000,

il y a donc eu 3 erreurs et le mécanisme se tromperait en concluant que le message original était 0000011111.

Les propriétés d'auto-correction et de détection d'un ensemble de mot de code valide dépendent de sa distance de Hamming:

- Pour détecter d erreurs, il faut un ensemble de mot avec une distance de Hamming de d+1
- Pour corriger d erreurs, il faut un ensemble de mot avec une distance de Hamming de 2d+1

## ***Une autre technique appelée Code de Hamming**:*

- Permet de corriger une erreur (1 bit)
- Utilise le contrôle de parité comme moyen de correction.

– Les bits représentant un message et son info. redondante sont numérotés depuis la gauche de 1 à n

– Ici, l'info redondante ne suit pas le message mais se trouve mélée à celui-ci.
Càd. que tous les bits qui se trouvant à une position qui correspond à une puissance de 2 sont des bits de contrôle (qui font partie de l'info. redondante).
Ex.:

mot envoyé: ***10***1***1***100***1***001
tous les bits en gras et italique sont des bits de contrôle.

- Tous les autres bits sont des bits de données, càd. les bits qui forment le message.
- Chacun des bits du message est contrôlé par certains bits des bits de contrôle.

  Un bit de message à la position n est contrôlé par les bits de contrôle qui correspondent à la décomposition en puissances de 2 de n.
  Ex.:

  le bit de message à la position 3 est contrôlé par les bits de contrôle à la position 1 et 2, car la décomposition en puissance de 2 de 3 donne 1 + 2.

  Pareil pour le bit de message à la position 5, il est contrôlé par les bits 1 et 4.
- Lors de l'envoi du message, la question est savoir quelles valeurs affecter aux bits de contrôle.

Une fois que l’on connaît tous les bits du message que contrôle un bit de contrôle, la valeur de ce bit de contrôle est alors la parité des valeurs de tous les bits du message qu’il contrôle.

– Lors de la réception d‘un message, le contrôle de parité se fait de la même manière et la somme des numéros d‘ordre des bits de contrôle qui sont en erreur donne le numéro d‘ordre du bit en erreur.

– Exemple: en parité PAIRE

| | |
|---|---|
| Message | `1100001` |
| Trame envoyée | `10111001001` |
| Trame reçue | `10111001011` |
| No d‘ordre | **`1 2`** `3` **`4`** `5 6 7` **`8`** `9 10 11` |
| Trame | `1 0 1 1 1 0 0 1 0 1  1` |

*La vérification du côté récepteur:*

| Bits du message | | Sont contrôlés par (décomp en 2^) | | |
|---|---|---|---|---|
| 3 | : | 1 | 2 | |
| 5 | : | 1 | 4 | |
| 6 | : | 2 | 4 | |
| 7 | : | 1 | 2 | 4 |
| 9 | : | 1 | 8 | |
| 10 | : | 2 | 8 | |
| 11 | : | 1 | 2 | 8 |

| Bits de contrôle | Bits du message<br>Valeur du bit | | | | |
|---|---|---|---|---|---|
| 1 | 3 | 5 | 7 | 9 | 11 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 1 | 0 | 0 | 1 | 1 |
| 2 | 3 | 6 | 7 | 10 | 11 | |
| | 1 | 0 | 0 | 1 | 1 | ***1*** |
| 4 | 5 | 6 | 7 | | | |
| | 1 | 0 | 0 | | | 1 |
| 8 | 9 | 10 | 11 | | | |
| | 0 | 1 | 1 | | | ***0*** |

Bits 8 et 2 sont faux : 8 + 2 = 10 ème bit en erreur

En cas de plusieurs erreurs consécutives : «BURSTS» ?

On bufferise les données de façon à construire une matrice de plusieurs mots de code et l'envoi se fait par colonnes. De l'autre côté, la matrice est reconstituée puis contrôlée.

Cette technique ne permet de corriger que des cas de k bits consécutifs en erreurs (k est le nombre de ligne dans la matrice).

***Conclusion:***

Dans la pratique, il vaut souvent mieux utiliser un modèle pour détecter les erreurs et retransmettre qu‘un modèle auto-correctif. C‘est plus efficace. Mais dans le cas d‘une transmission Simplex, il est évident qu‘une stratégie auto-corrective s‘impose.