Protection contre les erreurs

Introduction, taux d'erreurs, détection d'erreurs, code auto-correcteur.

IV.1 Introduction

Les voies de communication sont imparfaites :

- A cause des bruits.
- Bruits aléatoire.
- Bruits en provenance des canaux voisins.
- Bruits dû aux équipements
- A cause de la qualité ligne.

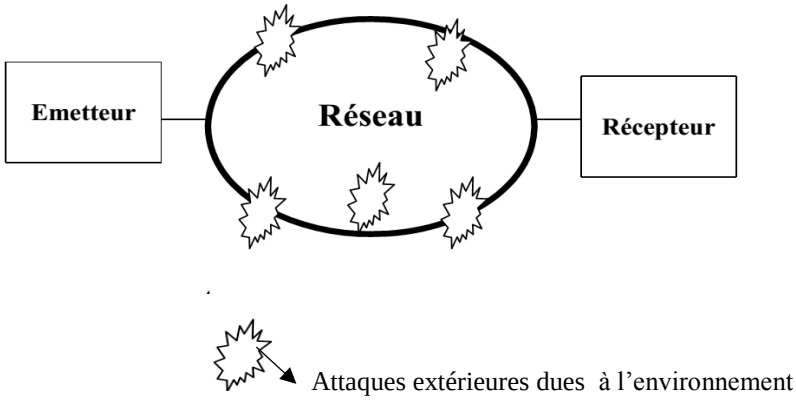


Figure IV.1 Réseau et de communication son environnement

- Conséquences des bruits :
  - Perte d'information durant la transmission.
  - Dégradation de l'information.

Nécessité d'un mécanisme de protection contre les erreurs.

Il est donc primordial de s'assurer que la perte d'informations soit minimale : c'est là qu'interviennent les codes correcteurs. L'objectif est de rajouter une information au message initial qui i permettront de détecter les erreurs et de les corriger. Bien évidemment l'efficacité d'un code sera donc déterminée par la quantité d'informations ajoutée et par sa capacité à conserver l'information d'un message.

IV.1.1 Problématique

Dans la Figure(IV.2) trois questions qui se posent :

1) Comment B peut détecter l'occurrence d'une erreur ?
2) Comment B peut localiser une erreur ?
3) Comment B peut corriger une erreur ?

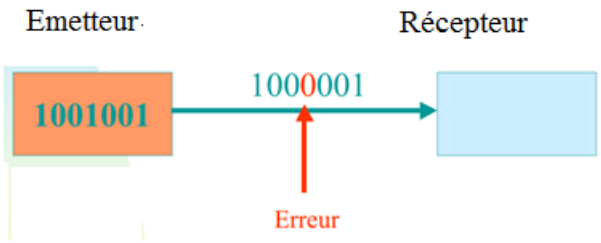
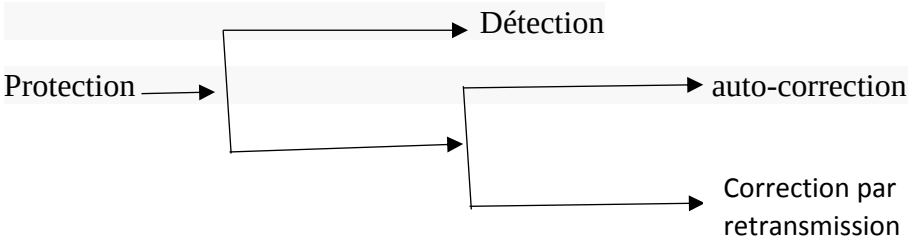


Figure IV.2

Pour répondre à ces questions nous prévoyons Stratégies de protection contre les erreurs de transmission :



IV.2 Taux D'erreurs

Qu'est-ce que l'erreur?

L'erreur c'est lorsque les informations de sortie ne correspondent pas aux informations d'entrée. Pendant transmission, les signaux numériques souffrent de bruit qui peut introduire des erreurs dans le déplacement des bits binaires d'un système à l'autre. Cela signifie qu'un bit 0 peut changer à 1 ou un bit 1 peut changer à 0. Le taux d'erreurs d'une liaison de donnée est le rapport du nombre de bits erronés reçus aux nombres total de bits transmis.

$$T_e = \frac{Nb(bits\ erronés)}{Nb(bits\ total\ transmis)} \qquad \text{(IV.1)}$$

- Exemple :

Soit la transmission de la suite: "0110 0100 1100 1001 0100 1010", qui est reçue "0110 0110 1100 1011 0100 0010". Quel est le taux d'erreurs ?

- Solution :

Le message reçu diffère de 3 bits du message émis. Le nombre de bits émis est de 24 bits. Le taux d'erreurs est:

$$T_e = \frac{3}{24} = 0.125$$

IV.3 Détection D'erreurs

Les signaux numériques se composent seulement des uns et de zéros. Par conséquent, toute erreur peut avoir deux conséquences: un est remplacé par un zéro ou un zéro par un. En fonction de la durée de l'interférence, cela peut changer un bit ou plusieurs. Le problème avec les erreurs de transmission de données est que le récepteur peut ne pas être de différencier les données incorrectes des données correctes, car les données incorrectes peuvent ne pas être différenciées des données correctes. Par conséquent, il est nécessaire qu'un système de détection d'erreur soit intégré au système de transmission. De cette façon, le récepteur sera en mesure de trouver une erreur et de demander à l'émetteur de retransmettre les données correctes. Ces erreurs n'ont rien à voir avec les erreurs de quantification, elles sont dues uniquement au support de transmission. Ces erreurs peuvent modifier la qualité de la liaison, voire même la couper si elles sont trop nombreuses. Il faut donc les détecter et les corriger

IV.3.1 Techniques De détection D'erreur

1) Les différents types d'erreurs sont:

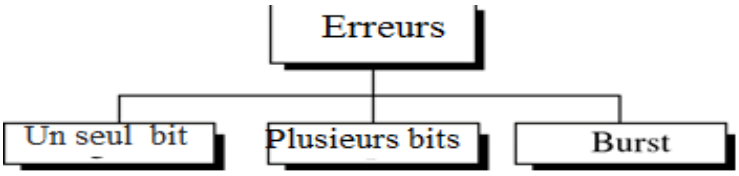


Figure IV.3 Types Erreurs

a) Erreur avec Un seul bit

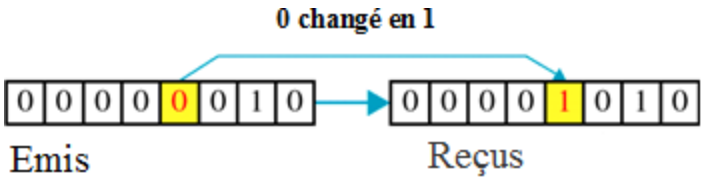


Figure IV.4 Erreur avec Un seul bit

b) Erreur de bits multiples

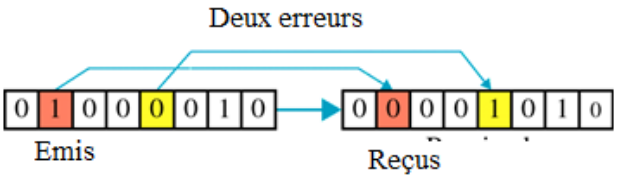


Figure IV.5 Erreur avec bits multiples

c) Erreur de Brust

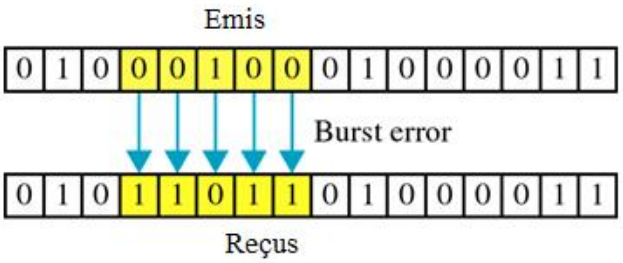


Figure IV.6 Erreur De Brust

2) technique de détection par répétition

Chaque message émis en double exemplaire. Si les deux messages sont différents, le récepteur demande une retransmission. Cette technique est très utilisée dans les milieux sécurisés et très perturbés.

Exemple :

Envoyer 10010011001001 au lieu de 1001001 Le récepteur détecte une erreur si les exemplaires ne sont pas identiques.

Remarque :

Certaines erreurs ne peuvent pas être détectées le cas où la même erreur sur les deux exemplaires.

3) **technique de détection par code:**

Une information supplémentaire au niveau du caractère (bit de parité) ou au niveau d'un groupe de caractères (clé) est ajoutée à l'information transmise. Le récepteur contrôle le bit de parité ou la clé, s'il détecte une erreur, il ignore les données reçues et en demande la retransmission

4) **Détection d'erreurs par bit de parité**

Cette technique, est appelé parfois VRC, pour(*Vertical Redundancy Check* ou *Vertical Redundancy Checking*) introduit une information complémentaire, un bit ou un caractère, dépendant du contenu binaire du message à protéger, tel que le nombre de bits, à 1 ou à 0, à transmettre soit pair (bit de parité) ou impair (bit d'imparité).

Exemple :

Pour protéger un caractère de 7 bits (code ASCII), on introduit un 8ème bit, dit bit de parité qui illustré dans le tableau **IV.1** suivant :

Tableau IV.1 contrôle de parité

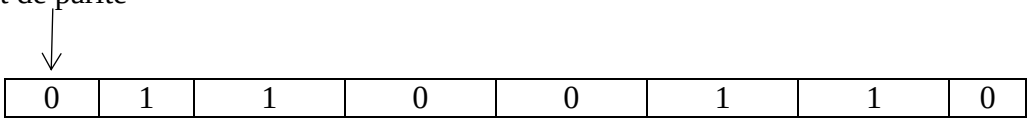
Caractère	O	S	I
Bit 0	1	1	1
Bit 1	0	0	0
Bit 2	0	1	0
Bit 3	1	1	1
Bit 4	1	1	0
Bit 5	1	1	0
Bit 6	1	1	1
Bit de parité	1	0	1
Bit d'imparité	0	1	0

Cette technique est utilisée essentiellement dans les transmissions asynchrones. Dans les transmissions synchrones, les caractères sont envoyés en blocs.

Remarque :

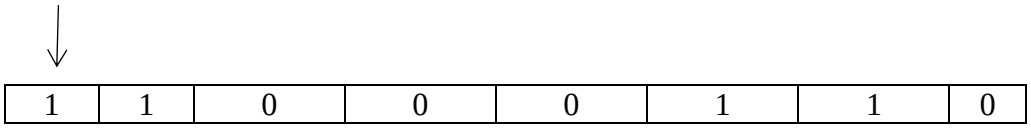
Soit l'exemple suivant : le nombre de bits de données à *1* est pair, le bit de parité est donc positionné à 0

Bit de parité



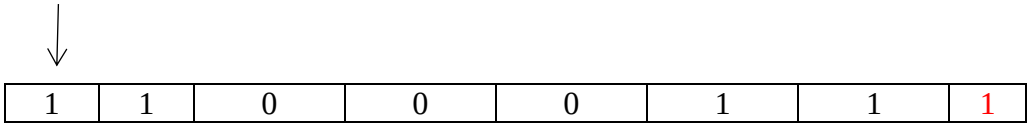
Dans ce deuxième 'exemple suivant, les bits de données étant en nombre impair, le bit de parité est à 1 :

Bit de parité



Imaginons qu'après transmission le bit de poids faible (le bit situé à droite) de l'octet précédent soit infecté d'une interférence :

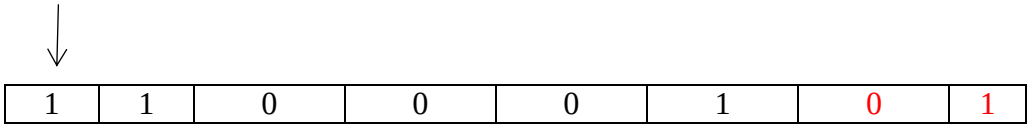
Bit de parité



Le bit de parité ne correspond alors plus à la parité de l'octet : **une erreur est détectée**.

Toutefois, si deux bits (ou un nombre pair de bits) venaient à se modifier simultanément lors du transport de données, aucune erreur ne serait alors détectée.

Bit de parité



Le système de contrôle de parité ne détectant que les erreurs en nombre impair, il ne permet donc de détecter que 50% des erreurs. L'inconvénient majeur de ne pas permettre de corriger les erreurs détectées (le seul moyen est d'exiger la retransmission de l'octet erroné...).

IV.4 code auto-correcteur

IV.4.1 Double parité

C'est le code obtenu en effectuant un double contrôle de parité. L'exemple d'un nombre codé en ASCII (7 bits) illustré dans le tableau IV.2

Pour le codage :

- Chaque caractère est codé sur une ligne du tableau
- Un code de parité impaire est effectué sur chaque ligne (contrôle transversal)
- Un code de parité impaire est effectué sur chaque colonne (contrôle longitudinal)

Pour le décodage :

- Le contrôle transversal permet de détecter une erreur sur la première ligne (le premier caractère)
- Le contrôle longitudinal permet de détecter une erreur sur la quatrième colonne (bit numéro 4).
- Le bit numéro 4 du premier caractère est faux, on peut le corriger

Tableau IV.2 Contrôle de parité impaire

Caractère	Numéro de bit							**Bit de Parité**	**Contrôle** Transversal
	1	**2**	**3**	**4**	**5**	**6**	**7**		
1. car.= 1	0	1	1	1	0	0	1	**0**	Faux
2. car.= 9	0	1	1	1	0	0	1	**1**	OK
3. car.= 6	0	1	1	0	1	1	0	**1**	OK
4. car.= 8	0	1	1	1	0	0	0	**0**	OK
Bit de Parité	**1**	**1**	**1**	**1**	**0**	**0**	**1**		
contrôle Longitudinal	**OK**	**OK**	**OK**	**Faux**	**OK**	**OK**	**OK**		

IV.4.2 Code de Hamming

IV.4.2.1 Définitions générales

Un code (k, n) transforme (code) tout bloc initial de k bits d'information en un bloc codé de n bits. Le code introduit une redondance puisque n >= k. Soit $r = n - k$ derniers bits forment un champ de contrôle d'erreur. Le rendement d'un code (k, n) est : $R = \frac{k}{n}$

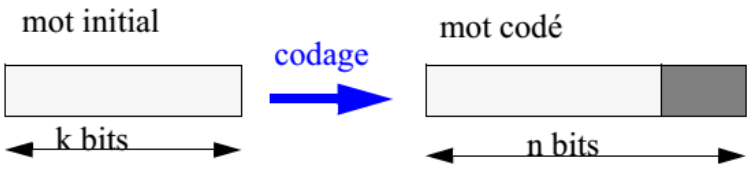


Figure IV.7 code (k, n)

On appelle **mot du code**, la suite de n bits obtenue après un codage (k, n). Le nombre n de bits qui composent un mot du code est appelé la longueur du code. La dimension k étant la longueur initiale des mots.

IV.4.2.2 Les codes linéaires

Les codes linéaires sont des codes dont chaque mot du code (noté c) est obtenu après transformation linéaire des bits du mot initial (noté i).

Ces codes sont caractérisés par leur matrice $G(k,n)$(appelée **matrice génératrice**) telle que :

$$i.G = c \qquad \textbf{(IV.2)}$$

$$G = [I_k \, , r] \qquad \textbf{(IV.3)}$$

I_k : Matrice identité de taille k(message initial)

r : on appelle (bits de parité, ou de contrôle, ou redondants) de taille $(k, n-k)$

Exemple :

On considère des mots de 3 bits et un codage linéaire de matrice G. Déterminer les mots codés.

$$G = \begin{Vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{Vmatrix}$$

Solution

Ici le message initial = 3bit, donc $I_k = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$, $r = n - k = 4 - 3 = 1$, n=4 est donné d'après la matrice génératrice car $G(k,n)$, **r**=$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ est valable pour cet exemple

Dans le cas général il y a des possibilités de calculer **r** d'après l'énoncé d'exercice.

On applique l'équation (**IV.2)** on obtient le tableau suivant :

Message Initial (k=3bit donc 2^3 combinaison)	Mot codé (n de taille 4)
0 0 0	0 0 0 0
0 0 1	0 0 1 1
0 1 0	0 1 0 1
0 1 1	0 1 1 0
1 0 0	1 0 0 1
1 0 1	1 0 1 0
1 1 0	1 1 0 0
1 1 1	1 1 1 1

IV.4.2.3 Procédure Codage de Hamming

Permet de détecter et corrigé les erreurs sur 1 seul bit. Consiste à ajouter de la redondance qui se traduit par l'ajout d'un nombre spécifiques de bit de parité. On pose la question suivante

Combien de bits de parité on doit ajouter au message initial pour détecter et corrigé l'erreur sur 1 seul bit.

Soit k le message initial et r bits de parité donc il y a (k+ r) cas ou l'erreur se produit sur 1 seul bit, et 1 état non erreur.

(k+ r) taille du message à envoyer et r bit de parité, donc il y a 2^r bits valeurs possible d'où

La formule qui détermine le nombre de bit de parité

$$2^r \geq k + r + 1 \qquad \textbf{(IV.4)}$$

Exemple :

$n = 7, k = 4$ Si on applique l'équation (**IV.4)** on trouve $\boldsymbol{r} = 3$.ce code hamming(7,4)

Soit k le message initial =**1110** et on veut coder suivant hamming comment trouver les 3

Bit de parité on le note (r_1, r_2, r_3) comme $n = 7$ alors on écrit

7	6	5	4	3	2	1
1	1	1	r_3	0	r_2	r_1

On place r_1, r_2, r_3 suivant puissance de 2 on commençant a droite c'est-à-dire le poids plus faible

r_3	r_2	r_1	
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

On remarque r_1 vaut 1 dans les bits 1, 3,5,7 et r_2 vaut 1 dans les bits 2,3,6,7 et r_3 vaut1 dans les bits 4,5,6,7

Remarque : l'émetteur et le récepteur se met d'accord sur le même protocole de parité paire ou impaire. Dans notre cas on choisit parité pair.

$r_1 \rightarrow$ 1,3,5,7 Correspond 110r_1 car le bit1r_1, le bit 3 correspond 0, le bit 5 correspond 1, le bit 7 correspond 1

110r_1 on applique la parité pair donc $r_1 = 0$ parce que on a deux 1(pair)

$r_2 \rightarrow$ 2, 3, 6,7 et on fait même chose comme r_1

110$r_2$on applique la parité pair donc $r_2 = 0$

$r_3 \rightarrow$ 4,5,6,7 on fait même chose comme r_1

111r_3 ici le nombre de 1 est impaire pour rendre pair il faut r_3=1

Finalement le message à transmettre suivant le code hamming est

7	6	5	4	3	2	1
1	1	1	1	0	0	0

Ce code **1 1 1 1 0 0 0** qu'on doit envoyer