Chapitre **IV :**

Protection contre les erreurs

IV.1. Introduction

Les rayonnements électromagnétiques, les perturbations propres au système (distorsions, bruit...) peuvent modifier les informations transmises (bits erronés). Compte tenu de l'extension des réseaux et de l'utilisation massive de la fibre optique, la perte de la synchronisation des horloges est, aujourd'hui, la principale source d'erreurs.

IV.2. Taux d'erreur binaire

On appelle **taux d'erreur binaire** ou **BER** (Bit Error Rate) le rapport entre le nombre d'informations (bits) erronées reçues et le nombre d'informations (bits) transmises.

$$\boldsymbol{TEB} = \frac{\boldsymbol{Nb\ de\ bits\ erron\acute{e}s}}{\boldsymbol{Nb\ de\ bits\ transmis}}$$

Soit, par exemple, la transmission de la suite « 011001001100100101001010 » qui est reçue « 011001101100101101000010 ».

Le message reçu diffère de 3 bits du message émis. Le nombre de bits émis est de 24 bits.

Le taux d'erreur binaire (**TEB**) est de :

$$\mathbf{TEB = 3/24 = 0{,}125}$$

IV.3. La détection d'erreur

On appelle détection d'erreur les mécanismes mis en œuvre pour que le système destinataire puisse vérifier la validité des données reçues. La détection d'erreur repose sur l'introduction d'une certaine redondance dans l'information transmise. Quatre techniques peuvent être mises en œuvre pour détecter et éventuellement corriger les erreurs :

– **La détection par écho** : Le récepteur renvoie en écho le message reçu à l'émetteur, Si le message est différent de celui émis, l'émetteur retransmet le message. Cette technique est utilisée dans les terminaux asynchrones (Telnet,...).

– **La détection par répétition** : chaque message émis est suivi de sa réplique. Si les deux messages sont différents, le récepteur demande une retransmission. Cette technique est utilisée dans les milieux sécurisés très perturbés et dans certaines applications dites temps réel.

– **La détection d'erreur par clé calculée :** une information supplémentaire (clé) déduite des informations transmises est ajoutée à celles-ci. En réception, le récepteur recalcule la clé, si le résultat obtenu correspond à la clé reçue les données sont réputées exactes, sinon le récepteur ignore les données reçues et éventuellement en demande la retransmission (reprise sur erreur).

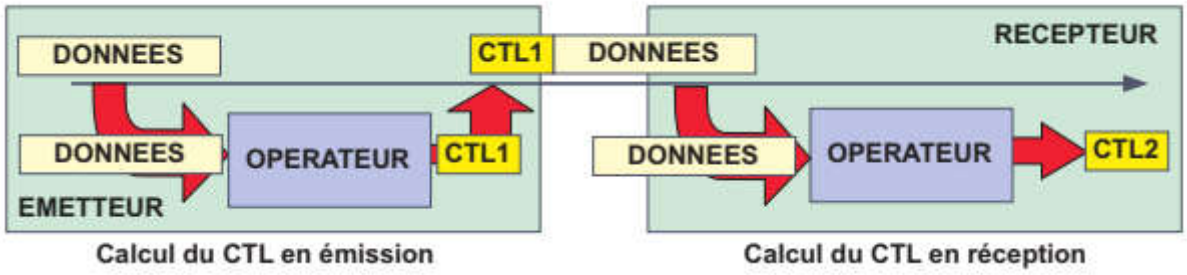
Auteur : Dr. H. ZERROIKI

– **La détection et correction d'erreur par code** : cette technique consiste à substituer aux caractères à transmettre, une combinaison binaire différente du codage de base (code auto-correcteur).

IV.4 Détection d'erreur par clé calculée

IV.4 .1 Principe

Dans les systèmes à clé calculée, une séquence de contrôle (CTL1) déduite d'une opération mathématique appliquée au message à émettre est envoyée avec le message. Le récepteur effectue la même opération. Si le résultat trouvé (CTL2) est identique à la clé calculée par la source (CTL1) le bloc est réputé exact, dans le cas contraire le bloc est rejeté (figure 1).

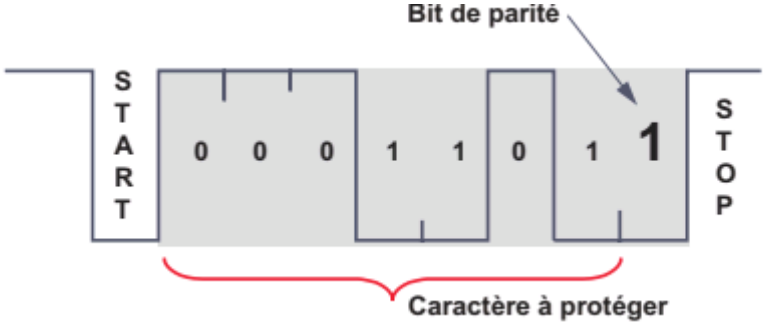


IV.4.2 Technique dit du "bit de parité"

La technique du **bit de parité** consiste à ajouter, à la séquence binaire à protéger, un bit, telle que la somme des bits à 1 transmis soit paire (bit de parité) ou impaire (bit d'imparité). Cette arithmétique modulo 2 est simple, mais elle n'introduit qu'une faible redondance. La protection apportée est limitée au caractère.

Lettre	Code ASCII (7bits)	Mot codé (parité paire)	Mot codé (parité impaire)
B	1000010	10000100	10000101
D	1000100	10001000	10001001
F	1000110	10001101	10001100
R	1010010	10100101	10100100

Cette technique, connue sous le nom de **VRC (*Vertical Redundancy Check*),** vérification par redondance verticale ne permet de détecter que les erreurs portant sur un nombre impair de bits. Elle est, essentiellement, utilisée dans les transmissions asynchrones.



Auteur : Dr. H. ZERROIKI

Dans les transmissions synchrones, les caractères sont envoyés en blocs. La technique du bit de parité est insuffisante, elle est complétée d'une autre information : le **LRC** (***Longitudinal Redundancy Check*)**.

Caractère à transmettre	**bit de parité**	Caractère à transmettre	**bit de parité**	...	Caractère LRC	**bit de parité**

Dans ce mode de contrôle dit de parité à deux dimensions, un caractère LRC est ajouté au bloc transmis. Chaque bit du caractère LRC correspond à la parité des bits de chaque caractère de même rang : le premier bit du LRC est la parité de tous les 1er bits de chaque caractère, le second de tous les 2e bits... etc. Le caractère ainsi constitué est ajouté au message. Le LRC est lui-même protégé par un bit de parité (VRC).

Lettre	Code ASCII (7bits)	VRC
H	**1001000**	**0**
E	**1000101**	**1**
L	**1001100**	**1**
L	**1001100**	**1**
O	**1001111**	**1**
LRC	**1000010**	**0**

1001000	0	1000101	1	1001100	1	1001100	1	1001111	1	1000010	0
H		E		L		L		O		LRC	

IV.4.3 Les codes cycliques (CRC, *Cyclic Redundancy Check*)

Dans la détection par clé calculée, l'information redondante, la clé (**CRC**, *Cyclic Redundancy Check*), est déterminée par une opération mathématique complexe appliquée au bloc de données à transmettre et transmise avec celui-ci

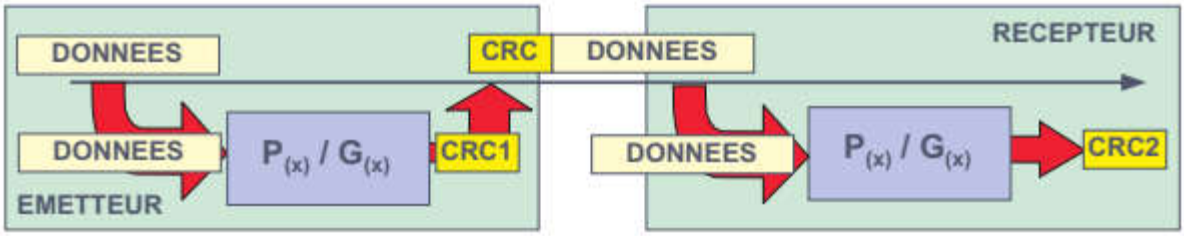
Données : suite de bits quelconque.	Clé ou CRC ou FCS
Bloc ou TRAME à Transmettre	

Structure d'un bloc de bits protégé par clé calculée.

La méthode de contrôle par clé calculée considère le bloc de *N* bits à transmettre comme un polynôme P(x) de degré *N–1*. Ce polynôme est divisé par un autre, dit polynôme générateur G(x) selon les règles de l'arithmétique booléenne ou arithmétique modulo 2. Le reste de cette division R(x) constitue le CRC parfois appelé aussi **FCS** (*Frame Check Sequence*). Le CRC calculé est transmis à la suite du bloc de données.

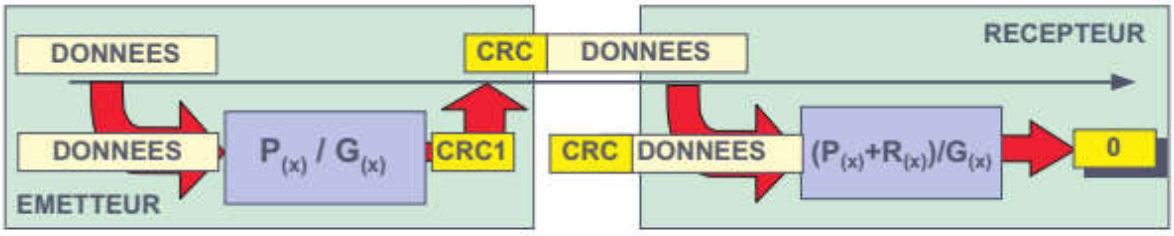
En réception, le destinataire effectue la même opération sur le bloc reçu. Le CRC transmis et celui calculé par le récepteur sont comparés, si les valeurs diffèrent une erreur est signalée.

Auteur : Dr. H. ZERROIKI



Principe de la détection d'erreur par clé calculée.

En réalité la méthode utilisée est quelque peu différente. En effet, si D est le dividende, d le diviseur et R le reste, la division $(D - R)/d$ donne un reste nul. Dans ces conditions, la division par le polynôme générateur G(x) de l'ensemble bloc de données et du CRC soit P(x) + R(x) donne un reste égal à zéro. En réception, l'automate effectue la division sur l'ensemble du bloc de données y compris la clé calculée, lorsque le calcul du reste donne zéro et que le caractère suivant est le fanion, le bloc est réputé exact.



Détection d'erreur par CRC.

L'arithmétique modulo 2 est une arithmétique sans retenue, l'affirmation précédente n'est donc exacte que si le reste est ajouté à une séquence binaire nulle. Pour réaliser cette condition, avant d'effectuer la division, on multiplie le polynôme **P(x)** par $\mathbf{x^m}$ où **m** est le degré du polynôme générateur, ce qui correspond à une translation de **m** positions.

Rappelons que le reste de la division par un diviseur de degré ***m*** est de degré ***m – 1***, il comporte donc ***m*** termes. Cette opération a pour effet d'insérer ***m*** bits à zéro, pour y ajouter les termes du reste.

L'exemple développé ci-dessous devrait éclairer le lecteur.

Exemple :

On désire protéger le message « **110111** » par une clé calculée à l'aide du polynôme générateur $\mathbf{x^2 + x + 1}$.

Au message **110111**, on fait correspondre le polynôme : $x^5 + x^4 + 0x^3 + x^2 + x^1 + x^0$

Pour permettre l'addition de la clé au message, on multiplie le polynôme représentatif du message par $\mathbf{x^m}$ où m est le degré du polynôme générateur. Le dividende devient :

$$(x^5 + x^4 + 0\,x^3 + x^2 + x^1 + 1) \times x^2 = x^7 + x^6 + 0x^5 + x^4 + x^3 + x^2 + 0 + 0$$

La division est réalisée par des systèmes « hardware » qui effectuent des « ou exclusif ». Aussi, appliquons la division par « ou exclusif » au polynôme **1010010111**. Si le polynôme générateur est $\mathbf{x^4 + x^2 + x + 1}$, il lui correspond la séquence binaire **10111**.

Le degré du polynôme générateur étant de 4, on ajoute 4 zéros à la trame de données (initialisation à zéro d'un registre à 4 positions).

Appliquons la division par « ou exclusif » au polynôme **1010010111**, On obtient la division ci-dessous :

Auteur : Dr. H. ZERROIKI

```
1 0 1 0 0 1 0 1 1 1 0 0 0 0 | 1 0 1 1 1
1 0 1 1 1                   | 100 110 0100
0 0 0 1 1 1 0 1             |
      1 0 1 1 1             |
      0 1 0 1 0 1           |
        1 0 1 1 1           |
        0 0 0 1 0 1 0 0     |
            1 0 1 1 1       |
            0 0 0 1 1 0 0   |
```

Le reste (clé) comporte 4 termes, il est de degré 1 par rapport au polynôme générateur. Le reste ou CRC4 est donc **1100**. Le message à transmettre est P(x) + R(x) : 10100101111**1100**.

En réception, l'ensemble de message, données et clé, subit la même opération ; si le reste de la division est égal à zéro, on estime que le message n'a pas été affecté par une erreur de c transmission. Vérifions cette affirmation sur l'exemple précédent :

```
message                reste
1 0 1 0 0 1 0 1 1 1  1 1 0 0 | 1 0 1 1 1
1 0 1 1 1                    |
0 0 0 1 1 1 0 1              |
      1 0 1 1 1              |
      0 1 0 1 1 1            |
        0 0 0 1 0 1  1 1     |
              1 0 1  1 1     |
              0 0 0  0 0 0 0 |
```

Le message est réputé correctement transmis, le reste de la division (message + reste) est nul.

IV.5 Les codes auto-correcteurs (code Hamming)

Le principe des codes correcteurs et le même que celui des codes détecteurs : On substitue au mot à transmettre (mot naturel) un nouveau mot (mot code) :

- Lors de l'émission rajouter des bits de contrôle supplémentaires.
- A la réception, détecter les erreurs grâce aux bits de contrôle et possibilité de corriger ces erreurs.

Le code de Hamming est un code correcteur linéaire. Il permet la détection et la correction automatique d'une erreur de transmission. Cette correction est permise grâce à l'ajout d'informations redondantes. Il est basé sur les tests de parité. La version la plus simple permet de corriger un bit en erreur.

Aux **m** bits d'information, on ajoute **k** bits de contrôle de parité. On a donc **m + k = n** bits. Puisque les **k** bits de contrôle doivent indiquer les **n + 1** possibilités d'erreurs (dont l'absence d'erreur, ce qui explique le +1), il faut que $\mathbf{2^k \leq n + 1}$. Les $\mathbf{2^k}$ possibilités de codage sur les **k** bits servent à coder la position de l'erreur. Dès que cette position est calculée, on peut corriger le bit en erreur.

Auteur : Dr. H. ZERROIKI

On a intérêt à choisir les codes à redondance minimale qui donnent un maximum de positions d'information, c'est-à-dire à prendre $\mathbf{n = 2^k - 1}$ au lieu de $\mathbf{n < 2^k - 1}$.

Si l'on numérote les bits de droite à gauche à partir de 1, les bits de contrôle (ou de parité) sont placés sur les puissances de 2 (bits n° 1, 2, 4, 8, 16...) .Chaque bit de contrôle effectue un contrôle de parité (paire ou impaire) sur un certain nombre de bits de données. On détermine ainsi les n bits à transmettre ou à stocker. Il faut noter que le premier bit à droite porte le numéro 1 et non pas 0.

Si le nombre de bit d'information est 4 (m=4), on peut construire un code de Hamming sur 7 bits (n=7) en ajoutant 3 bits de contrôle (k=3).

7	6	5	4	3	2	1
m4	m3	m2	k3	m1	k2	k1

Les 3 bits de contrôle k3, k2 et k1 sont placés sur les puissances de 2 : k1 en position 1, k2 en position 2 et k3 en position 4.

Nous allons voir maintenant, pour chaque bit du message, quels sont les bits de contrôle qui permettent de vérifier sa parité :

7	(0111)	= 4 + 2 + 1	→ 7	est contrôlé par k3, k2, k1
6	(0110)	= 4 + 2	→ 6	est contrôlé par k3, k2
5	(0101)	= 4 + 1	→ 5	est contrôlé par k3, k1
4	(0100)		est le bit contrôle k3	
3	(0011)	= 2 + 1	→ 3	est contrôlé par k2, k1
2	(0010)		est le bit contrôle k2	
1	(0001)		est le bit contrôle k1	

Ou inversement, qui contrôle qui ?

- k1 contrôle les bits 1, 3, 5, 7
- k2 contrôle les bits 2, 3, 6, 7
- k3 contrôle les bits 4, 5, 6, 7

Avec une parité paire, par exemple, k1 doit être tel que le nombre de bits à 1, compté sur les bits 1, 3, 5 et 7 soit pair.

Dans le cas d'une parité paire, on fait simplement une addition modulo 2. Pour une parité impaire, il faut faire une addition modulo 2 inversée :

- nombre impair de 1 → 0
- nombre pair de 1 → 1

Quand on reçoit ou que l'on relit l'information, on effectue à nouveau le contrôle de la parité. Pour chacun des bits de contrôle, on compare la valeur reçue, ou relue, à celle recalculée :

- Si elles sont identiques, on assigne la valeur 0 à la variable binaire $\mathbf{A_i}$ associée au bit de contrôle $\mathbf{k_i}$.
- Sinon, on lui assigne la valeur 1.

Auteur : Dr. H. ZERROIKI

Reprenons l'exemple précédent :

- Supposons que la valeur associée à k_1 soit A_1=1;
- Supposons que la valeur associée à k_2 soit A_2=1;
- Supposons que la valeur associée à k_3 soit A_3=0;

On sait alors qu'une erreur se trouve dans la transmission suivante : $A_3A_2A_1$=011

Une erreur détectée en faisant intervenir k_1 ne peut venir que des bits dont l'adresse binaire se termine par 1, c'est-à-dire 1, 3, 5 ou 7. De même, le test k_2 porte sur les bits 2, 3, 6, 7. Enfin, le contrôle k3 porte sur les bits 4, 5, 6, 7.

Vous comprenez maintenant que la valeur $A_3A_2A_1$=000 indique une absence d'erreur.

la valeur $A_3A_2A_1$=001 indique une erreur sur le bit numéro 1.

la valeur $A_3A_2A_1$=110 indique une erreur sur le bit numéro 6.

-

Exemple :

Coder **1010 1011 001** avec une parité paire : m=11, donc k=4. Le message à transmettre contient n=15 bits.

numéro	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
type	m11	m10	m9	m8	m7	m6	m5	k4	m4	m3	m2	k3	m1	k2	k1
valeur	1	0	1	0	1	0	1	?	1	0	0	?	1	?	?

Dans le message à transmettre, on a des bits à 1 dans les positions suivantes : 15, 13, 11, 9, 7, 3.

On transforme ces positions en leur valeur binaire et on les additionne modulo 2 : On met 1 lorsque l'on a un nombre impair de 1 et 0 pour un nombre pair de 1.

15	**=**	**1**	**1**	**1**	**1**
13	**=**	**1**	**1**	**0**	**1**
11	**=**	**1**	**0**	**1**	**1**
9	**=**	**1**	**0**	**0**	**1**
7	**=**	**0**	**1**	**1**	**1**
3	**=**	**0**	**0**	**1**	**1**

		0	**1**	**0**	**0** → bits de parité

k_4	k_3	k_2	k_1

Le message codé est donc **1010 1010 1001 100**.

Supposons qu'on a reçu le mot Hamming suivant : **1010 0010 1001 100**. On a des bits à 1 dans les positions :

15	=	**1**	**1**	**1**	**1**	
13	=	**1**	**1**	**0**	**1**	
9	=	**1**	**0**	**0**	**1**	
7	=	**0**	**1**	**1**	**1**	
4	=	**0**	**1**	**0**	**0**	
3	=	**0**	**0**	**1**	**1**	
			----------			Addition modulo 2
		1	**0**	**1**	**1**	
		A_4	A_3	A_2	A_1	→ erreur à la position **11**

Après la correction du bit en position 11, on a le message suivant : 1010 1010 1001 100