DETECTION ET CORRECTION DES ERREURS

Le support matériel utilisé par la couche physique n'est pas fiable à 100%. Il est par conséquent nécessaire de pouvoir détecter des erreurs parmi la suite de bits reçue, et éventuellement les corriger. Pour cela, la couche liaison de données de l'émetteur ajoute des bits au message à transmettre, qui permettent à la couche liaison de données de l'entité réceptrice du message de vérifier la cohérence de ce qu'elle a reçu.

La couche liaison de données construit ainsi des LPDU, encore appelées trames, qui comportent en particulier un FCS (Frame Check Sequence).

La problématique des erreurs comporte 3 aspects :

– la détection d'une erreur ;
– la localisation de l'erreur détectée ;
– la correction de l'erreur trouvée.

Pour répondre à ces problèmes, on utilise des codes qui sont appliqués au message à transmettre. Ils permettent de détecter certaines erreurs, mais pas nécessairement toutes, et peu permettent la correction. Ces techniques ne sont donc pas complètement fiables, d'autant que le FCS, utilisé pour vérifier et corriger le message, peut lui aussi être erroné.

1. Un code simple : la répétition

Une approche naïve consiste à dupliquer (c'est-à-dire répéter) le message à transmettre. Supposons que le message effectivement transmis soit le double du message réel. Par exemple, pour envoyer 11100010, on transmet 1110001011100010. La détection et la localisation des erreurs sont alors simples : on cherche les différences entre la première et la seconde moitiés du message. Par contre, il est impossible de corriger une erreur détectée : le bit erroné est différent dans les deux copies, et rien ne permet de dire lequel est le bon.

Pour remédier à ce problème, on peut envoyer le message en 3 exemplaires au lieu de 2. Dans ce cas, un bit a soit la même valeur dans toutes les copies, ou la même valeur dans deux d'entre elles et l'autre valeur dans la troisième copie. Le bit correct est probablement celui qui apparaît en deux exemplaires : on peut cette fois corriger l'erreur.

2. Codes à contrôle de parité

Les codes à contrôle de parité sont de parité soit paire, soit impaire. Dans le premier cas, on va protéger une séquence de bits en ajoutant un nouveau bit de telle sorte que le nombre de bits ayant la valeur 1 (dans la séquence protégée plus le bit introduit) soit pair. Dans le second cas, ce nombre doit être impair.

2.1. VRC (Vertical Redundancy Check)

C'est la technique la plus simple. Un code ASCII étant défini sur 7 bits, on utilise le 8ème bit de l'octet pour introduire le code vérificateur.

Exemple : Pour transmettre la chaîne de caractères IUT, on code chaque lettre en ASCII, puis on ajoute le code de parité.

Lettre	ASCII	VRC pair	VRC impair
I	1001001	11001001	01001001
U	1010101	01010101	11010101
T	1010100	11010100	01010100

Pour envoyer le message avec un code de parité pair, on transmet (avec l'ordre d'envoi des bits de gauche à droite) : 11001001 01010101 11010100

Ce code permet de détecter les erreurs en nombre impair sans pouvoir corriger. Il est peu efficace.

2.2. LRC (Longitudinal Redundancy Check)

Le principe est similaire à celui du VRC, mais au lieu de protéger les caractères un par un, on protège l'ensemble des bits de même rang de tous les caractères. On obtient alors un code de protection sur 7 bits.

Exemple : Pour protéger IUT, on calcule le code :

I	1001001
U	1010101
T	1010100
LRC pair	1001000
LRC impair	0110111

Pour envoyer le message avec un code de parité pair, on transmet :
1001001 1010101 1010100 1001000

2.3. LRC et VRC

On peut également combiner les deux techniques précédentes. On protège alors chaque caractère par un code VRC et l'ensemble des bits par un code LRC. On obtient donc un LRC sur 8bits. La parité des LRC et VRC utilisés est la même (tous les deux pairs ou tous les deux impairs).

Exemple : Pour transmettre la chaîne de caractères IUT, on code chaque lettre en VRC puis en LRC :

Lettre	ASCII	VRC pair	VRC impair
I	1001001	11001001	01001001
U	1010101	01010101	11010101
T	1010100	11010100	01010100
LRC		01001000	00110111

Pour envoyer le message avec un code de parité pair, on transmet :
11001001 01010101 11010100 01001000

3. Codes polynomiaux

les codes VRC et LRC s'appliquent sur des blocs de données de tailles équivalentes, si les blocs de données sont de tailles variables, ces codes deviennent inutiles.
Les codes polynomiaux possèdent un avantage : ils peuvent opérer sur des blocs de taille variable. C'est un cas qui se présente souvent en transmission, puisque la taille des données peut varier.

Un code polynômial est basé sur l'utilisation d'un polynôme générateur G(x). Les polynômes manipulés sont binaires : tous les coefficients sont 0 ou 1. Par conséquent, un polynôme générateur de degré k s'écrit sous la forme :

$$G(x) = a_0 + a_1x + a_2x^2+...+a_kx^k$$

Le polynôme G(x) est associé à une valeur binaire.

Exemple : La valeur binaire associée au polynôme $G(x) = x^3 + x + 1$ est 1011.

Soit M le message (séquence de bits) à protéger. Un polynôme M(x) lui est associé :

$$M = m_n \ldots m_2m_1m_0$$
$$M(x) = m_nx^n + \cdots + m_2x^2 + m_1x + m_0$$

Exemple : Au message M = 1101 est associé le polynôme $M(x) = x^3 + x^2 + 1$.

Codage

Le calcul du CRC (Cyclic Redundancy Code)s'effectue dans le corps Z/2Z, c'est-à-dire que :

- $1 + 1 = 0$
- $x + x = 0$
- $x = -x$

Soient :
– G(x) un polynôme générateur de degré k ;
– M(x) le polynôme associé au message M à transmettre.

La procédure de codage consiste à :

- calculer: $P(x) = M(x)*x^k$.

  Ceci correspond à un décalage de k bits (vers la gauche) du message M. La longueur du CRC calculé sera aussi de k bits. Cette opération de décalage revient à préparer la place nécessaire pour ces *k* bits de CRC.

- diviser le polynôme P(x) par G(x). Soient Q(x) et R(x) les polynômes quotient et reste ainsi obtenus :

  $$P(x) = Q(x)*G(x) + R(x)$$

- le CRC est le reste R(x) calculé. On remarque que le reste est forcément au maximum de degré $k-1$.

- le message effectivement transmis est associé au polynôme:

$$M'(x) = P(x) + R(x).$$

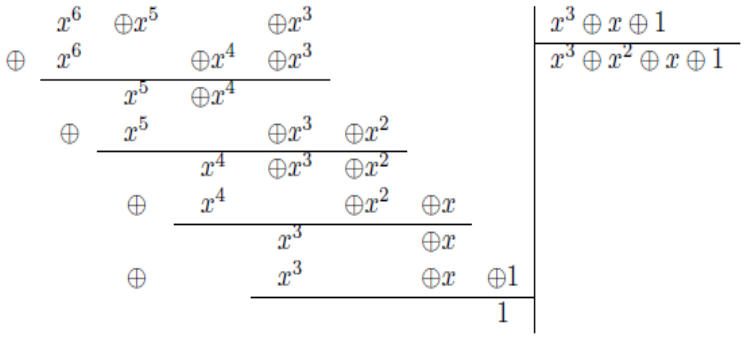
Il est par conséquent composé du message initial M suivi de la séquence de *k* bits correspondant à R(x).

Exemple : Soient le polynôme générateur $G(x) = x^3 + x + 1$ et le message à envoyer M = 1101.

Le polynôme correspondant au message est $M(x) = x^3 + x^2 + 1$. Le degré de G(x) est 3. Donc,

$P(x) = M(x) * x^3 = x^6 + x^5 + x^3$.

Effectuons la division de P(x) par G(x) :



Le quotient est donc $Q(x) = x^3 + x^2 + x + 1$,
et le reste $R(x) = 1$.

Le message transmis a alors pour polynôme $M'(x) = x^6 + x^5 + x^3 + 1$,

d'où M' = 1101001.