La solution de TD N° 04 : Couche liaison de données (traitement des erreurs de transmission

Objectif du TD : Ce TD est consacré à l'acquisition des mécanismes de traitement des erreurs employes au niveau de la couche liaison de données.

Solution de l'exercice 01 :

1) **Quelles sont les deux fonctionnalités mises en oeuvre par tous les protocoles de la couche liaison ?**
Le découpage en trame et le contrôle d'erreurs.

Pourquoi le contrôle d'erreurs ?

Lors de la transmission d'un train de bits, des erreurs peuvent se produire au niveau de la couche physique (c'est à dire qu'un "1" peut être transformé en un "0" ou réciproquement). C'est pour ça on a besoin des mécanisme de contrôle d'erreurs au niveau de la couche liaison de données.

Le contrôle d'erreurs, Comment ça se fait?

Le contrôle d'erreurs se fait par la collaboration de l'émetteur et le récepteur. Au niveau de l'émetteur, l'information utile (mot utile) est encodée de manière à lui ajouter de l'information de contrôle ; le récepteur examine l'information de contrôle et vérifie si elle est convenable à l'information utile. Ou avec une autre façon, le récepteur vérifie si l'information codée (mot codé) appartient à l'ensemble des mots codés valides. Si la vérification est positive, il n'y a pas une détection erreurs et le récepteur considère que l'information est correcte. Sinon, le récepteur détecte qu'il y a des erreurs ; cette détection sera suivie par une correction, si le mécanisme de codage lui permet .Sinon elle sera suivie par une demande de retransmission du mot codé erroné reçu.

2) **Quelle est la distance de Hamming entre deux mots valides dans le code ASCII ?**

Pour passer d'un mot du code valide à un autre mot du code valide, il suffit de modifier au minimum un seul bit. Donc la distance de Hamming $\mathbf{d_{min}} = 1$.
En d'autres termes, la distance de Hamming est 1 parce que tous les mots codés possibles forment une séquence; il suffit de modifier un seul bit pour passer d'un mot du code valide à un autre mot du code valide.

Un bref rappel :

La **distance** « **d** »entre deux mots d'un code est le nombre de différences entre ces deux mots.
Par exemple, m1 = 10110010 m2 = 10000110 **d**(m1,m2) = 3
Pour déterminer **la distance de Hamming** « $\mathbf{d_{min}}$ » du code entier, il faut comparer entre tous les mots **codés** deux à deux en calculant la distance et à la fin, il doit retenir la distance la plus petite.

La distance de Hamming est un concept théorique important pour déterminer la puissance de tels codes.

Remarques :

1/La distance de Hamming est le nombre minimum de bits modifiés pour passer d'un mot du code valide à un autre mot du code valide.

2/Pour unifier les termes:

«Un mot du code valide » = « mot du code »= « un mot du code correct ».
« Information utile » = « mot utile »= « partie utile ».

« Information codée » = « mot codé »= « mot du code ».
« Encodage » = « codage ».

3) **Quelle est la distance de Hamming du code {000 ; 111}?**

La distance de Hamming du code est 3.

4) **Quelle est la relation entre la distance de Hamming du code d_{min}, le nombre possible d'erreurs détectées « p » et d'erreurs corrigées « q »?**

$$p < d_{min}$$

$$q < \frac{d_{min}}{2}$$

5) **Quand est ce que on dit le code en bloc, systématique ou linéaire ?**

On dit le code **en bloc** lorsque l'information de contrôle et l'information utile forment un tout consistent. Si le bloc est composé de deux parties distinctes (information utile et information de contrôle) le code est dit **systématique**.

Un code **linéaire** (n, m) : c'est un code en bloc systématique dans lequel les r = n - m bits de contrôle dépendent linéairement des m bits d'information.

Tel que « r » est la taille de la partie de contrôle, « m » est la taille de la partie utile et « n » est la taille d'un mot du code

Solution de l'exercice 02 :

1) **Donner une méthode simple qui, si exactement une erreur s'est produite pendant la transmission de n bits, permet au récepteur de la détecter.**

La simple méthode consiste à l'ajout d'un bit de parité après n-1 bits de manière à ce que le nombre de bits y compris celui de parité, qui sont égaux à 1, doit être pair.
Cette méthode nous permet de détecter une erreur s'il y a une érreur parce que d_{min} =2 donc p=1

Un bref rappel :

Code de parité paire : consiste à ajouter d'un bit de parité à un nombre de bits de manière à ce que le nombre de bits y compris celui de parité, qui sont égaux à 1, doit être pair.
Code de parité impaire : consiste à ajouter d'un bit de parité à un nombre de bits de manière à ce que le nombre de bits y compris celui de parité, qui sont égaux à 1, doit être impair.
Ce genre de ce code s'appelle aussi **code VRC** (Vertical Redundancy Check).

2) **Quelle est la distance de Hamming de votre code ?**

La distance de Hamming de ce code est **2** car si l'on change un bit quelconque d'un mot il faut aussi changer le bit de parité pour que cela soit un mot du code.

3) **Combien y a-t-il de mots dans votre code ?**

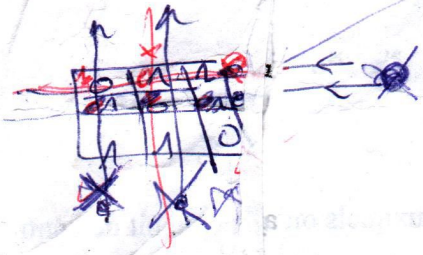
Il y en a 2^{n-1}

4) **Que se passe-t-il s'il y a un nombre pair d'erreurs ?**

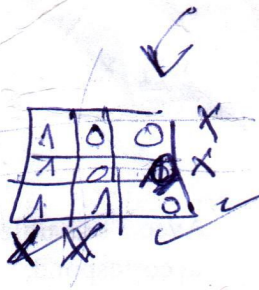
Le récepteur ne détecte pas les mots erronés avec un nombre pair d'erreurs parc que avec ce type d'erreurs, la parité est toujours conservée (le mot erroné corresponde à un autre mot du code).

Solution de l'exercice 03 :

1) **« b=3 » ça veut dire que le mot de code de parité croisée se compose de 3 caractères utiles (O, S, I)(chaque de ces caractères utiles est suivi d'un bit de contrôle VRC) et un caractère de contrôle LRC**



		← VRC
O	1001111	1
S	1010011	0
I	1000011	1
LRC →	1011111	0



Quelle sera la transmission série des bits du terme «OSI »?

10 1111 0 1000011 1 1010011 0 1001111 1

LRC I VRC S VRC O VRC

Un bref rappel :

Le code LRC (Longitudinal Redundancy Check) consiste à effectue un calcul de parité sur les bits de même rang pour « b » blocs

Le code VRC s'utilise avec le code LRC pour renforcer le code. Et dans ce cas le code engendré s'appelle **code de parité croisée**.

2) **Quelle est la distance de Hamming de ce code ?**

On constate que sur un bit quelconque de données il y a trois bits de contrôle qui interviennent. Il faut donc changer 4 bits pour retomber sur un autre mot du code. Donc la distance est égale à **4**.

3) **Combien d'erreurs ce code peut-il détecter ? Combien peut-il en corriger ?**

D'après les formules données ci-dessus, on peut détecter « p » erreurs tel que $p < d_{min}$ et corriger « q » erreur tel que $q < \frac{d_{min}}{2}$. Pour nous, on trouve p=3 et q=1 (arrondis a l'entier inférieur).

Solution de l'exercice 04 :

1) **Déterminer les mots codés des mots utiles de 3 bits et un codage linéaire de la matrice G suivante :**

$$G = \overbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{x_1, x_2, x_3} \overbrace{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}^{a_1}$$

Il y a 3 bits utiles (x_1, x_2, x_3) et selon la matrice on déduit qu'il y a 1 bit de contrôle a_1 , soit 4 bits pour un mot du code (mot codé). La relation matricielle $\tilde{y} = \tilde{X} \cdot G$ permet de déterminer la valeur de ce bit de contrôle :

$$(y_1\ y_2\ y_3\ a_1) = (x_1\ x_2\ x_3) \begin{Vmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{Vmatrix} = (x_1\ x_2\ x_3\ \underbrace{x_1 + x_2 + x_3}_{a_1})$$

Avec 3 bits utiles, on peut former 8 mots utiles (mots non codés) auxquels on ajoute le bit de contrôle a_1 correspond.

mot non codé	mot codé
000	0000
001	0011
010	0101
011	0110
100	1001
101	1010
110	1100
111	1111

Solution de l'exercice 05 :

1) **Avec un code linéaire (6,3) dont la matrice G, Combien d'erreurs ce code peut-il détecter ? Combien peut-il en corriger ?**

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

- La détermination du nombre d'erreurs détectées et d'erreurs corrigées revient à déterminer la distance de Hamming du code.
- Pour calculer cette distance de Hamming, il doit comparer tous les mots codés deux à deux en calculant la distance correspond et retenir la distance la plus petite.

Un code linéaire (6,3), ça veut dire que la taille d'un mot du code (mots codés) est 6 bits et la taille d'un mot utile est 3 bits (x_1, x_2, x_3). Et selon la matrice, cela indique que la taille de la partie de contrôle est 3 bits (a_1, a_2, a_3)

Avec 3 bits utiles, on peut former 8 mots utiles (mots non codés) auxquels on ajoute 3 bits de contrôle (a_1, a_2, a_3) corresponds.

La relation matricielle $\tilde{y} = \tilde{X} \cdot G$ conduit à

$$a_1 = x_2 + x_3$$
$$a_2 = x_1 + x_3$$

information utile : $\tilde{X} = (x1\ x2\ x3)$ donc 8 mots possibles

information codée : $\tilde{Y} = (x1\ x2\ x3\ a1\ a2\ a3)$

La relation $\tilde{Y} = \tilde{X} \cdot G$ conduit à

le calcul conduit à l'ordre des valeurs :

$$\begin{cases} a_1 = x_2 + x_3 \\ a_2 = x_1 + x_3 \\ a_3 = x_1 + x_2 \end{cases}$$

c'est l'ordre en modulo 2

Les mots du code sont :

000000	001110	010101	011011
100011	101101	110110	111000

On constate que $d_{min} = 3$ ce qui permet la correction des erreurs simples et la détection des erreurs doubles