Protocoles réseaux
TD n° 2 : Codes détecteurs

Un **code** est une fonction qui a tout mot de m bits associe un mot de n bits, $n > m$. On note $r = n - m$ la **redondance** du code.

Le code est une fonction *injective*, c'est-à-dire qu'il n'existe pas deux mots de m bits qui se codent pareil. Le **dictionnaire** du code est l'ensemble des 2^m mots de n bits codés.

Lors du décodage,
- Si le mot y de n bits reçu est dans le dictionnaire, alors on décode avec l'unique x à m bits tel que $C(x) = y$
- Sinon, le code a **détecté** une erreur

Si le mot envoyé est y et le mot reçu est y', le nombre d'**erreurs sur un bit** qui ont lieu est le nombre de 1 de $y \oplus y'$ ($\oplus$ est le XOR).

Un code *détecte k erreurs* si, pour tout y' reçu avec au moins une erreur et au plus k par-rapport au y envoyé, le code signale une erreur.

Exercice 1 :

1. Construire un code qui détecte 1 erreur. L'étendre pour détecter k erreurs (k étant un entier donné du protocole).
2. Puisque l'on peut faire facilement, pour tout k, un code qui détecte k erreurs, pourquoi celui-ci n'est-il pas davantage utilisé ?

Exercice 2 : bit de parité

Les anciens modems RTC utilisaient un système de contrôle d'erreurs basé sur le bit de parité. À la fin de chaque octet, on rajoute un 9ème bit de sorte que le nombre de 1 dans l'octet soit pair.

1. L'utiliser pour coder 0110010 puis 011101000111101
2. Donner l'algorithme de détection d'erreur.
3. Combien d'erreur sur un bit ce code détecte-t-il ? Donner un exemple d'erreur non détectée.

Exercice 3 : matrice de parité

Une variante du bit de parité est la matrice de parité. On écrit 8 octets de données sur une matrice 8x8, puis on rajoute une 9eme ligne et une 9eme colonne. Le bit de la i*eme* colonne (resp. ligne) est choisi pour que le nombre de 1 dans cette colonne (resp. ligne) soit pair.

1. Coder la matrice ci-contre
2. Que vaut le bit en position (9,9) ?
3. Combien d'erreur sur un bit ce code détecte-t-il ?
4. Donner un exemple d'erreur non détectée

1	0	0	0	1	1	0	1	
0	1	1	1	0	0	1	0	
1	0	0	0	0	0	0	1	
0	1	1	0	1	1	0	1	
0	1	0	0	0	0	0	0	
1	0	0	0	1	1	0	1	
0	1	0	1	0	1	0	0	
1	1	0	0	0	0	0	1	

Codes CRC

Soit A un message de m bits à coder. A est vu comme un polynôme à coefficients binaires. Par exemple 10011010 devient $X^7 + X^4 + X^3 + X$.

On pose $m + r = n$. Soit B un polynôme binaire de degré r.

Le codage de A par le CRC B consiste à calculer le reste R de la division euclidienne de $X^r A$ par B et à envoyer $X^r A - R$ (multiple de B ayant les même m premiers bits que A).
L'algorithme utilise une variable P, qui est un polynôme de degré maximal r. Initialement il vaut les $r + 1$ premiers monômes de $X^{r-m}A$. On note a_i le coefficient du $i$ème monôme de A, c'est-à-dire que $A = \sum_{i=0}^{m} a_i X^i$.

Données : le polynôme A de degré m (B de degré r étant connu)
début

Polynôme $P = \sum_{i=0}^{r} a_{(i+m-r)} X^i$;
pour j *de* m *à* 0 *par pas de -1* **faire**
  si P *est de degré* r *(c'est-à-dire* $P = X^r + ...$*)* **alors**
    $P = P - B$ // on soustrait le dénominateur si $P \not< B$
  si $j \neq 0$ **alors**
    $P = X * P$ // on décale
  si $j > r$ **alors**
    $P = P + a_{(j-r-1)}$ // on "descend" le coef du monôme de degré $j - 1$ de $X^r A$
  retourner P;

Le décodage consiste à vérifier si le polynôme reçu A' est multiple de B et
- si non : erreur détectée,
- si oui : supposé correct. Transmettre A' divisé par X^r.

Exercice 4 : codes CRC

1. Coder 0110010 puis 011101000111101 avec le polynôme $X^4 + X + 1$

2. Décoder 11010111 puis 10010010 avec le polynôme $X^4 + X + 1$

3. Implémenter l'algorithme en C. Un polynôme de degré $r < 64$ peut se stocker dans un `uint64_t`. Chaque ligne de pseudo-code est ensuite une ligne de C ! Cet algorithme, nommé parfois *shift-or*, est très facile à implémenter logiciellement ou matériellement, ce qui explique le succès des CRC à une époque où le *hardware* était cher et peu puissant.

4. Montrer que cet algorithme peut être utilisé à la fois pour calculer les CRC et pour les vérifier.

5. Dans une trame Ethernet, le dernier champ est un CRC-32. Pourquoi le champ est-il en dernier ?

6. À quel code correspond le code du CRC dont le polynôme est $X + 1$?

7. Voici le code CRC du MPEG 2 - aussi utilisé en 802.3 :

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

Est-il capable de détecter 20 erreurs ?