Détection et correction d'erreurs

1. Introduction

Quelle que soit la qualité des supports de communication et les performances des techniques de transmission utilisées, des perturbations vont se produire entraînant des erreurs sur les données transmises (cela peut atteindre une probabilité de 10^{-4} sur une ligne téléphonique par exemple).
Dans ces conditions, la suite binaire reçue ne sera pas identique à la suite émise et il faut utiliser des méthodes de détection des erreurs et éventuellement de correction des erreurs.

Les stratégies de protection contre les erreurs de transmission sont :

- La détection
- Et la correction (auto-correction ou correction par retransmission)

Dans ce cadre, l'émetteur inclut dans le bloc de données :

1. Suffisamment de redondance pour que le récepteur puisse reconstituer les données originales. On utilise des **codes correcteurs d'erreur** plutôt pour des canaux non fiables comme le sans-fil)
2. Juste assez de redondance pour que le récepteur puisse détecter les erreurs et demander une retransmission. On utilise des **codes détecteurs d'erreur** (plutôt pour des canaux fiables comme la fibre optique)

Le principe

- Un émetteur veut transmettre un message (suite binaire quelconque) à un récepteur.
- L'émetteur transforme le message initial à l'aide d'un procédé de calcul spécifique qui génère une certaine redondance des informations au sein du message codé.
- Le récepteur vérifie à l'aide du même procédé de calcul que le message reçu est bien le message envoyé grâce à ces redondances.

Exemple : la technique de détection par répétition.
Le message codé est un double exemplaire du message initial, le récepteur sait qu'il y a eu erreur si les exemplaires ne sont pas identiques.
Dans ce cas, certaines erreurs sont indétectables par exemple une même erreur sur les deux exemplaires simultanément.

Le surcoût typique d'un code auto-correcteur est de quelques octets par message

Définitions générales

Un code **(k, n)** transforme tout bloc initial de **k** bits d'information en un bloc codé de **n** bits.
Les r (r=n-k) derniers bits forment un champ de contrôle d'erreur.
Le rendement d'un code (k, n) est : $R = k/n$

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

La distance de Hamming

Entre deux mots de même longueur est définie par le nombre de positions binaires qui diffèrent entre ces deux mots.
La **distance de Hamming** d'un code est la distance minimum entre tous les mots du code.

Exemple

- distance_de_Hamming(01001100, 01010101) = 3
- distance_de_Hamming({01001100, 01010101, 00000000}) = 3

La capacité de **détection** (respectivement de ***correction***) d'un code est définie par les configurations erronées qu'il est capable de **détecter** (resp. ***corriger***).

Pour qu'un code ait une capacité de détection de **n** bits avec une distance de Hamming **dist,** il suffit de vérifier l'inégalité suivante : **n+1 ≤ dist**

Pour qu'un code ait une capacité de correction de **n** bits avec une distance de Hamming **dist,** il suffit de vérifier l'inégalité suivante : **2n+1 ≤ dist**

Exercices

1. Quelle est la distance de Hamming entre deux mots valides dans le code ASCII ?
Quels sont les types d'erreurs que permet de déceler un tel codage ?

2. On choisit un code à trois bits, les mots de code valides sont ceux dont les trois bits sont identiques.
Quelle est la distance de Hamming ?
Si une erreur se produit sur un seul bit, quels sont les mots que l'on peut obtenir, et par quel mot valide chacun peut-il être remplacé sans ambiguïté ?

2. Codes par bloc

Le contrôle de parité

Parité paire (impaire) : le poids de Hamming des mots du code est pair (impair).
C'est un code systématique **(k, k+1)** dans lequel un bit (le bit de parité) est ajouté au mot initial pour assurer la parité. Son rendement est faible lorsque k est petit.

Exemple :
Transmission de caractères utilisant un code de représentation (le code ASCII sur 7 bits).

Lettre	**Code ASCII**	**Mot codé (parité paire)**	**Mode codé (parité impaire)**
E	1 0 1 0 0 0 1	1 0 1 0 0 0 1 1	1 0 1 0 0 0 1 0
V	0 1 1 0 1 0 1	0 1 1 0 1 0 1 0	0 1 1 0 1 0 1 1
A	1 0 0 0 0 0 1	1 0 0 0 0 0 1 0	1 0 0 0 0 0 1 1

Ce code est capable de détecter toutes les erreurs en nombre impair.
Il ne détecte pas les erreurs en nombre pair !

Parité longitudinale et transversale

Association d'un double codage de la parité :
- **LRC** : « *Longitudinal Redundancy Check* » et **VRC** : « *Vertical Redundancy Check* »

Le bloc de données est disposé sous une forme matricielle (k=a*b). On applique la parité (uniquement paire) sur chaque ligne et chaque colonne.
On obtient alors une matrice [a+1, b+1]. Le rendement est faible : a*b/ (a+1)*(b+1).

Le délai de codage et décodage important : il faut recevoir tout le bloc.
Concernant la capacité de détection et d'autocorrection, une erreur simple modifie simultanément la parité d'une ligne et d'une colonne.
Pour la correction : inverser le bit situé à l'intersection de la ligne et de la colonne ayant une parité incorrecte.

Exercices

Dans l'alphabet ASCII le mot « OSI » se code par les 3 caractères de 7 bits suivants :
'O' = 1001111, 'S' = 1010011 et 'I' = 1000011

1. La LRC (Longitudinal Redundancy Check) consiste à rajouter un bit de parité à la fin d'un bloc de données (octet, caractère, suite de bits, …). La VRC (Vertical Redundancy Check) consiste à calculer les bits de parité entre plusieurs blocs de données en vertical :1 bit de parité pour les bits qui sont à la même position dans les différents blocs considérés.
Donnez la VRC du mot « OSI » en utilisant une parité paire pour calculer le LRC de chaque caractère

2. Combien d'erreurs ce code peut-il détecter ? Combien peut-il en corriger ?

3. Code de Hamming

Le code de Hamming est utilisé dans les transmissions de données car il permet de détecter et de corriger une erreur survenue dans un bloc transmis.

Principe du codage

On envoie en sus des **m** bits du message à transmettre, **k** kits de contrôle de parité dits de correction a **certaines positions** du bloc de données transmises.

La longueur totale des messages et de $\mathbf{2^k-1}$ et donc **la longueur des messages m = $(2^k-1)-k$**

Le tableau suivant indique quelques nombres de bits de correction pour différentes valeurs de k.

k=3	m=4	**n=7**	
k=4	m=11	**n=15**	
k=5	m=26	**n=31**	

Nota

- Un mot de code 7 (3 bits de contrôle) a un coefficient d'efficacité de 4/7 = 57%
- Un mot de code 31 (5 bits de contrôle) a un coefficient d'efficacité de 26/31 = 83%

Position des k bits de correction

Les **k** bits de correction sont places dans le bloc envoyé aux positions d'indice **d'une puissance de 2.**

Par exemple, si le message à envoyer est constitué de 4 bits, en notant k_1 k_2 k_3 les bits de correction et $m_1 m_2$ m_3 m_4 les bits de données, le bloc envoyé sera :

	m_4	m_3	m_2	$\mathbf{k_3}$	m_1	$\mathbf{k_2}$	$\mathbf{k_1}$
Position	***7***	***6***	***5***	***4***	***3***	***2***	***1***
Codage binaire de la position	111	110	101	**1**00	011	0**1**0	00**1**

Ainsi, le bit de contrôle k_1 (position 1 = 00**1**) contrôle les positions 3 (01**1**) 5 (10**1**) et 7 (11**1**)

Le bit de contrôle k_2 (position 2 = 0**1**0) contrôle les positions 3 (0**1**1), 6 (1**1**0) et 7 (1**1**1)

Le bit de contrôle k_3 (position 4 = **1**00) contrôle les positions 5 (**1**01), 6 (**1**10) et 7 (**1**11)

On positionne sur les bits de contrôle k le calcul de la parité paire des bits contrôlés. Chaque bit k contrôle de manière redondante plusieurs bits de message

Principe du décodage

A réception, on recalcule les valeurs des bits de contrôle k. Si ceux-ci diffèrent de la valeur reçue, on affecte la valeur 1 au bit de contrôle défaillant. La valeur décimale correspondant au code binaire défini par les bits de contrôle indique la position de l'erreur (que l'on peut ainsi corriger)

Exemple :

Valeurs k_2 et k_1 non conformes → $(k_3k_2k_1)_2 = (011)_2 = 3$ → le bit de la position 3 est erroné

Exercices

1. On souhaite utiliser le codage de Hamming pour émettre la suite de bits (1011). Quelle est la séquence à transmettre ?
2. Y a-t-il une erreur dans le mot suivant : 1101101 ?
3. Soit le mot de Hamming suivant : 101101111011011
    a. Quels sont les bits de contrôle de parité ?
    b. Quelles positions contrôle chacun de ces bits ?
    c. Quel est le message reçu ?
    d. Y a-t-il une erreur dans le message ?

4. Codes CRC (Cyclic Redundancy Check)

Définition : Un code polynômial est un code linéaire dont chacun des mots du code est un multiple d'un polynôme générateur (noté **g(x)**).

Nous allons utiliser ici une représentation sous forme polynomiale des suites de bits à transmettre où chaque valeur de bit est un coefficient (0 ou 1 donc) du polynôme. La valeur 10011 code par exemple le polynôme x^4+x+1.

Exemples de codes polynômiaux :

- L'avis V41 du CCITT conseille l'utilisation du CRC-CCITT avec le polynôme générateur :
  $g(x) = x^{16} + x^{12} + x^5 + 1$
- Le polynôme CRC-16 est utilisé par le protocole HDLC :
  $g(x) = x^{16} + x^{15} + x^2 + 1$
- Le polynôme suivant est utilisé par Ethernet :
  $g(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+1$

Principe du codage

Le mot de code **m(x)** d'un code polynômial **(k, n)** de polynôme générateur **g(x)** associé au mot initial **i(x)** est défini par :

$\mathbf{m(x)} = i(x)*x^{n-k} + r(x)$, où **r(x)** est le reste de la division de $i(x)*x^{n-k}$ par le polynôme générateur **g(x)**

- On choisit un polynôme générateur puis on le transforme en un mot binaire.
- On ajoute m zéros au mot binaire à transmettre où m est le degré du polynôme générateur.
- On va ajouter itérativement à ce mot, le mot correspondant au polynôme générateur jusqu'à ce que le mot obtenu soit inférieur au polynôme générateur. Ce mot obtenu correspond au CRC à ajouter au mot avant de l'émettre.
- On effectue donc une division euclidienne dans laquelle on ne tient pas compte du quotient.

Remarques :

1. Les **r** = **n-k** bits de r(x) (de degré ≤ n-k-1) forment les bits du champ de contrôle.
2. Les bits de poids fort (de degré ≤ n-k-1) forment le mot initial
3. L'opération de codage effectuée à l'émission est ramenée à une division polynômiale en base 2 sans tenir compte du quotient, qui peut être réalisée électroniquement simplement.

Nota : la division binaire correspond à une série de soustractions

Exemple

Soit le code 1011 à envoyer avec le code générateur `g(x)= x+1`.

1011 → `i(x) = x3 + x + 1`

Degré d du polynôme générateur : 1

On multiplie donc i(x). $x^d = x^4 + x^2 + x^1$

On effectue la division polynomiale et on calcule le reste $r(x) = (x^4 + x^2 + x) / (x+1) = 1$

Le message envoyé sera donc : $m(x) = x^4 + x^2 + x + \mathbf{1}$ soit 1011**1**

Principe du décodage

A la réception, chaque mot reçu **m(x)** est divisé par le polynôme générateur **g(x)** (Le message reçu doit être divisible par le polynôme générateur donc avec un reste nul).

- Un reste non-nul indique qu'il y a eu erreur lors de la transmission.
- Un reste nul indique (hors erreurs résiduelles) qu'il n'y a pas eu d'erreurs de transmission

Exercices

1. Calculez le CRC du mot « OSI » en utilisant le polynôme générateur $x^8 + 1$ et en supposant que le 8ème bit de chaque caractère est un bit de parité paire et que le mot d'information est composé des bits des 3 caractères à la suite.

2. Calculez avec la méthode de la division polynomiale le bloc de contrôle (CRC) correspondant à la suite de bits 1100101010101011 en utilisant le polynôme générateur $G(x) = x^4 + x^3 + x + 1$

3. On utilisera le polynôme générateur x^4+x^2+x
   - On souhaite transmettre le message suivant 1111011101, quel sera le CRC à ajouter ?
   - On reçoit les messages : 1111000101010 et 11000101010110, sont-ils corrects ?

4. La détection d'erreurs utilise le CRC $x^6 + x^4 + x + 1$.
Le récepteur reçoit la séquence binaire suivante 101011000110
Le message est-il correct (tester avec la méthode de division polynomiale binaire) ?