Chapitre 3 : Codes détecteurs et correcteurs d'erreurs

Capacité mémoire des ordinateurs

- Une RAM fait couramment 8 GigaOctets
- Un disque dur plusieurs Téra-octets
- Le volume de données utilisé par un ordinateur augmente très très vite !
- Aucun mécanisme de stockage n'est fiable à 100 %
- Il y donc des erreurs dans les données stockées sur les disques durs et dans les barrettes de RAM.

Besoin de fiabilité

- Une donnée erronée peut avoir des conséquences catastrophiques.
- On va donc essayer de détecter les données erronées ou encore tenter de les corriger automatiquement afin d'éviter de les utiliser.

Principe de base

- On rajoute de l'information afin de fiabiliser les données.
- La donnée est plus volumineuse mais on peut espérer savoir si elle est erronée ou non, ou même tenter de la corriger.

Principe de base

- On rajoute de l'information afin de fiabiliser les données.
- La donnée est plus volumineuse mais on peut espérer savoir si elle est erronée ou non, ou même tenter de la corriger.

Bit de parité sur 8 bits

- On rajoute 1 bit
- On choisit ce bit pour obtenir un nombre de bits à 1 total pair

Exemple

- 1100 1110 ==> 1100 1110 1
- 1011 1011==>1011 1011 0
- 1000 0000 ==> 1000 0000 1
- On obtient une donnée plus volumineuse sur 9 bits

Détection des données erronées

- 1100 1111 0 ==> non erronée
- 1111 1110 0 ==> ERRONEE
- 0000 0000 0 ==> non erronnée
  1111 1111 1 ==> ERRONEE

Bit de parité

- Pas fiable à 100%
- Permet de détecter les erreurs sans les corriger
- Améliore très fortement la fiabilité
- Utilisation : des barrettes de RAM comportent un bit de parité

Codage de Hamming

- On a une donnée sur N bits on va lui rajouter t bits
- t dépend de N
- T est le plus petit entier vériifiant $2^t-1>=N+t$

Exemple

- Voici une donnée dont on veut calculer son codage de Hamming
  1100 1101

- On veut trouver t :

- N=8
  donc t=4 car
  $2^4-1 >= 8+4$

Codage de Hamming

- On obtient une donnée sur N+t bits
- Si N=8, la donnée obtenue est sur 12 bits

  f12 f11 f10 f9 f8 f7 f6 f5 f4 f3 f2 f1

- Les bits de controles seront les bits dont l'indice est une puissance de 2

Codage de Hamming

- f12 f11 f10 f9 **f8** f7 f6 f5 **f4** f3 **f2 f1**
- On reporte la donnée initiale sous les bits qui ne sont pas des bits de contrôle
- **f12 f11 f10 f9 f8 f7 f6 f5 f4 f3 f2 f1**
  1 1 0 0 1 1 0 1
- **Respe à trouver les valeurs des bits de contrôle f1, f2, f4 et f8**

Construction d'un tableau de bits

- On écrit les entiers de 1 = N+t en base 2 sur t bits
  1 ==> 0001
  2 ==> 0010
  3 ==> 0011
  4 ==> 0100
  5 ==> 0101
  6 ==> 0110
  7 ==> 0111
  8 ==> 1000
  9 ==> 1001
  10==>1010
  11==>1011
  12==>1100

Construction de t ensembles de bits

- On construit les ensembles de bits E1, E2, E3 et E5 en regardant les colonnes du tableau et en repérant les 1

1 ==> 0001
2 ==> 0010
3 ==> 0011
4 ==> 0100
5 ==> 0101
6 ==> 0110
7 ==> 0111
8 ==> 1000
9 ==> 1001
10==>1010
11==>1011
12==>1100

- E1={f1,f3,f5,f7,f9,f11}
- E2={f2,f3,f6,f7,f10,f11}
- E3={f4,f5,f6,f7,f12}
- E4={f8,f9,f10,f11,f12}

- E1={f1,f3,f5,f7,f9,f11}
- E2={f2,f3,f6,f7,f10,f11}
- E3={f4,f5,f6,f7,f12}
- E4={f8,f9,f10,f11,f12}
- On va choisir le bit de controle pour que dans chaque ensemble le nombre de 1 soit pair

f12	f11	f10	f9	f8	f7	f6	f5	f4	f3	f2	f1
1	**1**	**0**	**0**		**1**	**1**	**0**		**1**		

- E1={f1,f3,f5,f7,f9,f11}={f1,1,0,1,0,1}==>f1=1
- E2={f2,f3,f6,f7,f10,f11}={f2,1,1,1,0,1}==>f2=0
- E3={f4,f5,f6,f7,f12}={f4,0,1,1,1}==>f4=1
- E4={f8,f9,f10,f11,f12}={f8,0,0,1,1}==> f8=0

RESULTAT FINAL

f12	f11	f10	f9	f8	f7	f6	f5	f4	f3	f2	f1
1	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**

EXERCICE

- Calculez le codage de Hamming de

  1010 1110

  1100 1101 10

  0011 1100 1001

Vérification

- Voici une donnée codée avec le codage de Hamming. Est-elle corrompue ? Donnez la donnée initiale après une éventuelle correction.

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

On calcule t bits e1, e2, e3 ... e_t

- On construit les ensembles de bits E1, E2 ....
- ei vaut 0 si le nombre de bits à 1 dans Ei est pair 1 sinon
- Normalement tous les ei valent 0

Ensemble de bits

- **f12 f11 f10 f9 f8 f7 f6 f5 f4 f3 f2 f1**
  1 0 0 0 0 1 1 0 1 1 0 1

- E1={f1,f3,f5,f7,f9,f11}={1,1,0,1,0,0}==>e1=1
- E2={f2,f3,f6,f7,f10,f11}={0,1,1,1,0,0}==>e2=1
- E3={f4,f5,f6,f7,f12}={1,0,1,1,1}==>e3=0
- E4={f8,f9,f10,f11,f12}={0,0,0,0,1}==> e4=1

Vérification

- E est l'entier qui s'écrit en base 2 (e4 e3 e2 e1)

- Si E=0, la donnée n'est pas corrompue
- Sinon elle est corrompue et la correction la plus probable est d'inverser f_E

- E=(1011)=11

- La donnée est corrompue et on inverse la valeur de f11

- E=(1011)=11
- La donnée est corrompue et on inverse la valeur de f11
- **f12 f11 f10 f9 f8 f7 f6 f5 f4 f3 f2 f1**
  1 **1** 0 0 0 1 1 0 1 1 0 1
- Donnée initiale après correction
  1100 1101

Utilisation

- La RAM ECC utilise le codage de Hamming
  - Les bits de contrôles sont rajoutés automatiquement
  - Les données sont corrigées par la barrette de RAM

Cyclic Redundant code CRC

- Le CRC est un code détecteur d'erreur : il ne permet pas de corriger une erreur
- il est redoutablement efficace en détection d'erreurs.

Polynôme générateur

- On se donne une fois pour toute un polynôme appelé polynôme générateur.
- Nous choisirons X^4+X+1
- En général on utilise des polynômes de plus haut degré :
- Exemple CRC-32 (Ethernet) : = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

Degré du polynôme générateur

- On appelle d le degré du polynôme générateur
- Ici d=4
- Il y a aura d bits dans le CRC

Calcul du CRC

- On veut calculer le CRC de la suite de bits 1001 0011

- Étape 1 : on forme un polynômeà partit des bits
$1X^7+0X^6+0X^5+1X^4+0X^3+0X^2+1X^1+1X^0$
$=X^7+X^4+X+1$

- Étape 2
  - On multiplie le polynôme par X^d donc ici par X^4
  - On obtient $X^{11}+X^8+X^5+X^4$
- Etape 3
  - On divise par le polynome générateur X^4+X+1 en faisant la division dans Z/2Z
    Règle de calcul : on remplace les -1 par des +1
  - On s'arrête dans le degré du reste est strictement inférieur à d

- $X^{11}+X^8+X^5+X^4$ | X^4+X+1

  $-X^{11}-X^8-X^7$ X^7+X^3+X

  $\Longrightarrow X^7+X^5+X^4$

  $-X^7-X^4-X^3$

  $\Longrightarrow X^5+X^3$

  $-X^5-X^2-X$

  $\Longrightarrow X^3+X^2+1$

- Étape 4 : on écrit le reste avec tous ses coefficients de X^{d-1} à X^0

  $1X^3+1X^2+0X^1+1X^0$

  CRC = 1101

- Etape 5 : on écrit le CRC à la fin
  1001 0011 **1101**

Exercice

- Calculez le CRC des suites de bits suivantes avec le polynôme générateur X^4+X+1

  1100 1100
  1010 1101 1100
  1110 1110 1111

Vérifier un CRC

- Voici une donnée avec son CRC à la fin et utilisant le polynôme générateur X^4+X+1 est-elle corrompue ?
  1001 0111 1001
- Étape 1 : on construit le polynôme
  $X^{11}+X^8+X^6+X^5+X^4+X^3+1$
- Etape 2 : on divise par le polynôme générateur
  Attention on ne multiplie pas par X^d quand on vérifie un CRC
- **Si le reste est nul, la donnée est non corrumpue sinon elle est corrompue**

- $X^{11}+X^{8}+X^{6}+X^{5}+X^{4}+X^{3}+1 \quad | \; X^{4}+X+1$

  $-X^{11}-X^{8}-X^{7} \qquad X^{7}+X^{3}+X^{2}+X$

  $==>X^{7}+X^{6}+X^{5}+X^{4}+X^{3}+1$

  $-X^{7}-X^{4}-X^{3}$

  $==>X^{6}+X^{5}+1$

  $-X^{6}-X^{3}-X^{2}$

  $==>X^{5}+X^{3}+X^{2}+1$

  $-X^{5}-X^{2}-X$

  $==>X^{3}+X+1$

  CORROMPUE

EXERCICES

- Ces données sont-elles corrompues ?
  1100 1001 1100
  1000 0011 0111
  1010 1000 1101

Utilisation du CRC

- Sur des disques dur
- Dans les archives zip
- Dans les trames éthernet
- Dans le protocole TCP

CONCLUSION

- Nous avons vu un code détecteur d'erreurs simple : le bit de parité.
- Nous avons un code détecteur et correcteur d'erreurs : le code de Hamming.
- Nous avons finalement vu un code détecteur d'erreurs très performant : le CRC.