Le contrôle d'erreur

- **Les données peuvent être modifiées (ou perdues) pendant le transport**
  - Un service primordial pour de nombreuses applications
  - Exemple : le transfert de fichier
- **Modification au niveau physique :**
  - déformation de l'onde
- **Perte complète de paquet :**
  - congestion du réseau, saturation des routeurs ou des switch

Taux d'erreur au niveau physique

- Réseaux étendus:
  - » ADSL sur ligne téléphonique: 10^{-3} à 10^{-9}
  - » Fibre optique : 10^{-9}
  - » Satellite: 10^{-6} à 10^{-11}
- Réseaux locaux: de 10^{-12}
- CD: 10^{-5}: 7ko erronés sur un CD de 700 Mo

- **La détection d'erreur**
  - Comment se rendre compte de la modification des données à l'arrivée des trames (problème au niveau physique) ?
- **Suppression des erreurs, deux techniques :**
  - Comment corriger à l'arrivée les données erronées ? :
    - » **la *correction d'erreur*** à l'arrivée du paquet
  - Faire en sorte que l'émetteur, renvoie les trames erronées/perdues:
    - » **la *récupération d'erreurs par re-émission***

Applications

- Transfert de donnée sur un réseaux très peu fiable:
  - Exemple: correction d'erreur dans l'ADSL : FEC (Forward Error Correction) affichées sur les Box
- Transfert de donnée sur un réseaux peu fiable:
  - Code correcteur coûte cher (en terme quantité d'information rajoutée)
  - On préfère faire de la récupération d'erreur par re-émission (si il n'y a pas de contrainte de temps)
- Méthodes de détection et correction sont aussi utilisées pour le stockage de donnée
  - Lecture de fichier sur un disque dur
    - » Système de correction intégré aux disques (RAID **Redundant Arrays of Inexpensive Disks)**
  - Lecture de CD/DVD (problème de rayure)
    - » Problème légèrement différent : l'information peut aussi ne pas exister (et on le sait)

La détection/correction d'erreur

- **Idée:** rajouter de l'information aux données permettant de détecter/ corriger les erreurs à l'arrivée
- **Exemple de détection: Code de répétition**
  - On duplique l'information
  - Par exemple on rajoute un bit identique pour chacun des bits à transmettre
  - Exemple:
    - » Données: 1 0 0 0 1 1
    - » Code: 1 1 0 0 0 0 0 0 1 1 1 1
- Coût en taille : élevé
- Coût en calcul faible
- **Qualité de la détection d'erreur ?**
  - Est-on capable de détecter 1 seul bit erroné ? 2 bits ?...

La correction d'erreur

- **Exemple de correction: Code de répétition**
  - On triple l'information
  - On rajoute deux bits identiques pour chacun des bits à transmettre
  - Exemple:
    - » Données: 1 0 0 1 1
    - » Code: 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1
- Coût en taille: très élevé
- **Qualité de la détection d'erreur ?**
  - Si il y a une seule erreur on peut la corriger ?
  - Si il y a deux erreurs ?

La détection d'erreur

- **Exemple de détection: Le code de parité**
  - On rajoute un bit à 1 ou 0 suivant la parité du nombre de bits à 1 dans les données Le récepteur vérifie la valeur de ce bit de parité.
  - Exemple:
    - » Données: 1 0 0 0 1 1 Bit de parité: 1
    - » Données: 1 0 0 1 1 1 0 1 1 Bit de parité : 0
- Très peu couteux en taille
- Très peu couteux en calcul
- **Qualité de la détection d'erreur ?**
  - Est-on capable de détecter 1 seul bit erroné ? 2 bits ?...

Contrôle d'erreur : un modèle d'étude

- **Mot de code**

  Si une trame contient m bits de données et r bits de contrôle,
  on appelle mot du code le mot formé par les m + r bits.
  On pose n = m + r

- **Distance de Hamming**
  - Étant donné deux mots de n bits m1 et m2, le nombre de bits dont ils diffèrent est appelé leur distance de Hamming (notée Disth)
- **Distance de Hamming du code complet (ou distance *minimale*)**

  h = { Min Disth(x1, x2) ; x1 et x2 ∈ M et x1≠x2}

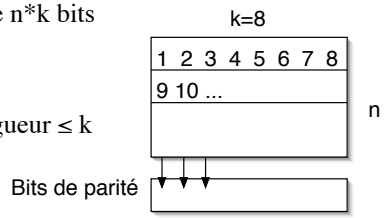
  M est l'ensemble des 2^m mots de codes possibles si on admet que les r bits de contrôle sont calculés en fonction des m bits de données.

Détection d'erreur

- **Propriété:**
  - Pour détecter (à coup sûr) x erreurs il suffit que la distance ***minimale*** $h \geq x + 1$
  - En effet ainsi s'il y a x erreurs on ne pourra pas "retomber" sur un code existant (différent forcément de x+1 bits), on saura donc qu'il est faux.
- **Exemple:**
  - On suppose que le récepteur possède en mémoire l'ensemble des mots justes
  - Il compare le mot reçu à chacun de ces mots justes
  - m = 2, r =1 : M = {000, 011, 101, 110}
  - h=2
  - Supposons 011 envoyé et 111 reçu (donc une erreur)
  - 111 forcément différent des mots justes: encore au moins un bit différent puisque h=2. Le récepteur sait donc qu'il y a une erreur

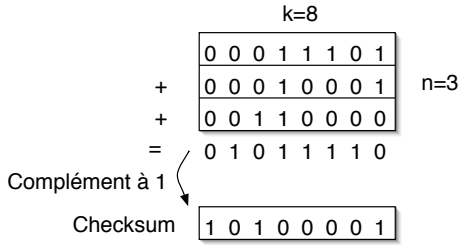
Exemple de code détecteur d'erreur

- **Les bits de contrôle sont calculés par l'émetteur, le récepteur fait un calcul analogue après réception**
  - Bit de parité
    - m = 2, r =1 : M = {000, 011, 101, 110 }
    - h = 2 aucune erreur double mais détecte aussi tous les erreurs dont le nombre est impair
  - **Bit de parité par colonne:**
    - Trame considérée comme une matrice n*k bits
    - 1 bit de parité par colonne
    - h=2
    - détection des rafales d'erreurs de longueur ≤ k



Détection d'erreur par "checksum"

- Données considérées comme n mots de k bits
- Bits de contrôle = complément à 1 de la somme des n mots
- A la réception la somme des n mots de données plus le checksum ne doit pas contenir de 0
- **h=2** mais aussi détection de certaines erreurs doubles (dans les colonnes) et des rafales d'erreur de longueur ≤ k
- Utilisé dans UDP, TCP



Détection d'erreur par CRC (Cyclic redundancy Code)

- Plus performant que les simples Checksums, surtout pour les paquets/rafales d'erreurs
- Ne dépend pas de la taille des données
- Peu coûteux en taille
- Calcul coûteux mais souvent fait par hard : ou exclusif successifs au fur et à mesure que la trame arrive
  - Circuit simple et rapide à base de registre à décalage et de portes XOR

Principe de calcul d'un CRC

- Basée sur des calculs de division de polynôme à coefficient dans [0, 1]
- Exemple: les bits de données 1 0 1 0 1 représentent x^4+x^2+1
- Arithmétique polynomiale modulo 2 (sans retenue): soustraction et addition sont équivalentes à un ou-exclusif bit à bit
- Ainsi $x^2 + x^2 = 0$, x + x = 0, x- x= 0
- Division d'un polynôme A par B : A= B*Q + R où Q est le quotient et R le reste avec R < B
- Exemple x^3+x^2+1 divisé par $x^2 + 1$
- Reste =x, Quotient = x+1
- en effet (x+1) $(x^2 + 1)= x^3 + x + x^2 +1$ le reste R vaut x car (x+x)= 0

Principe de calcul d'un CRC

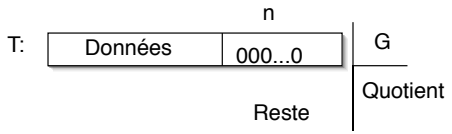
- On se donne un polynôme générateur G de degré n qui détermine le nombre de bits de contrôle (connu par l'émetteur et le récepteur)
- On rajoute n bits à 0 en poids faible à D (polynôme représentant les données) qui donne T
- Ainsi si l'on effectue la division de T par G:

  T= G*Q + R alors (T- R)= G * Q

  donc T - R divisé par G donne un reste nul

comme les n bits de poids faible de T' valent 0 il suffit de mettre R comme bits de poids faible de T pour avoir T- R

- Autrement dit : si l'on envoie E= T- R, il suffit à l'arrivée de calculer la division de E par G. Si le reste est non nul il y a une erreur.



Exemple calcul de CRC

- 6 bits de données: 110101 , Polynôme générateur 101 : $x^2 +1$

```
1 1 0 1 0 1 0 0 | 1 0 1
1 0 1           | -------
0 1 1 1         | 1 1 1 0 1 1
  1 0 1
  -----
  0 1 0 0
    1 0 1
    -----
    0 0 1 1 0
      1 0 1
      -----
      0 1 1 0
        1 0 1
        -----
Reste:  0 1 1
```

- On envoie E= T - R = 110101 11
- Si on refait la divison :

```
1 1 0 1 0 1 1 1 | 1 0 1
1 0 1           | -------
0 1 1 1         | 1 1 1 0 1 1
  1 0 1
  -----
  0 1 0 0
    1 0 1
    -----
    0 0 1 1 1
      1 0 1
      -----
      0 1 0 1
        1 0 1
        -----
        0 0 0
```


Qualité de détection du CRC

- On peut avec n=16 en choisissant bien le polynôme générateur:
  - détecter toutes les erreurs simples et doubles,
  - toutes les erreurs comportant un nombre impair de bits
  - toutes les rafales d'erreur de longueur ≤ 16
  - et, avec une très bonne probabilité, les rafales d'erreurs de longueur supérieure.
- Exemple: Ethernet utilise un champs CRC à 32 bits,
- Compression ZIP utilise un CRC à 16 ou 32 bits

Correction d'erreur

- **Propriété:**
  - Pour corriger x erreurs il suffit que la distance ***minimale*** $h \geq 2x + 1$
  - En effet s' il y a x erreurs, le code erroné reste ainsi «le plus proche» du code juste, on peut donc le retrouver. Les autres ont encore au moins x+1 différences.
- **Exemple de code correcteur**

  m = 2, r = 3

  M = {00111, 01100, 10000, 11011}, h = 3, on corrige une erreur

  On envoie 01100, il arrive 11100 : une erreur

  Le récepteur compare le mot arrivé aux 4 mots du code possible:

  00111 et 11100 : 4 différences

  01100 et 11100 : 1 différence

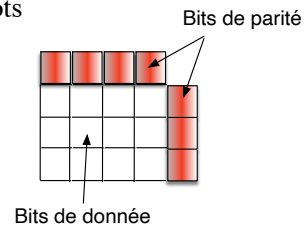
  10000 et 11100 : 2 différences

  11011 et 11100 : 3 différences

  Seul 01100 possède une seule différence, c'est le mot envoyé, le récepteur peut corriger

Exemple simple de code correcteur

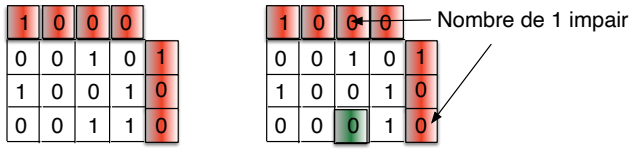
- On organise les données en x mots de m bits
- On calcul un bit de parité pour chaque mot et un bit de parité pour l'ensemble des ième bits des x mots



- Si on change un bit de donnée, le bit de parité de la colonne et de la ligne correspondantes changent donc toujours au moins 3 différences en deux mots du code donc h=3

On est donc capable de corriger n'importe quelle erreur simple

Exemple



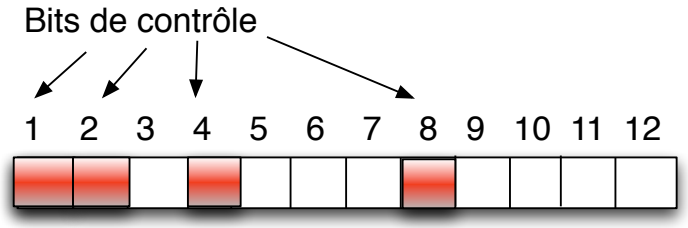
- Remarque:
  - Si c'est un bit de parité qui change, on a un seul bit de parité faux
  - Si il y a deux erreurs, on le détecte mais on ne sait pas les reconnaitre et donc les corriger
  - Le nombre de bits de contrôle est grand

- **Problème :**
  - Quelle est la valeur minimale de r permettant de corriger les erreurs simples dans des trames de m bits de données et r de contrôle ?
    - Pour être capable de corriger chaque erreur il faut qu'un mot erroné soit forcément différent de chacun des mots justes et de chacun des autres mot erronés; sinon on ne peut pas retrouver le mot initial et donc le corriger
    - Il faut donc avoir assez de mots de code ($2^{(m+r)}$) pour coder à la fois les mots justes (2^m) et les mots erronés ($(m+r)*2^m$)
    - Il faut donc que $2^{(m+r)} \geq 2^m + (m+r)*2^m$
    - soit $2^{(m+r)} \geq (m+r+1)*2^m$; soit $\mathbf{2^r \geq m + r + 1}$
  - Les codes correcteurs d'erreur "coûtent" chers

    Par exemple: 5 bits sont nécessaires pour corriger une erreur sur 12 bits de données ($\mathbf{2^5 \geq 12 + 5 + 1}$)

Le code correcteur de Hamming

- Propriété: nécessite le nombre minimal de bit de contrôle pour corriger une erreur
  - m=1 , r=2
  - jusqu'à m=4 , r= 3
  - jusqu'à m=11 r=4 ....
- Utilisable pour n'importe quelle taille de donnée
- Les bits de contrôle sont les bits de numéro égal à une puissance de 2

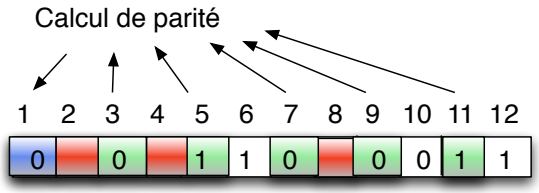


Le code correcteur de Hamming

- Les bits de données qui servent au calcul d'un bit de contrôle de numéro X sont ceux tel que X apparaît dans la décomposition en puissance de 2 de le numéro de position du bit de contrôle.
- Exemple: 7 = 1 + 2+ 4 donc 7 apparaît dans le calcul de 1, de 2 et de 4

1 calculé de telle façon que (1, 3, 5, 7, 9, 11, …) parité paire
2 calculé de telle façon que (2, 3, 6, 7, 10, 11,…) parité paire
4 calculé de telle façon que (4, 5, 6, 7, 12, …) parité paire
...

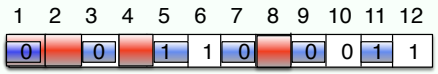
- Exemple :



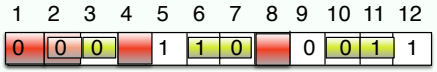
Le code correcteur de Hamming

- Exemple:
  - Trame envoyée:

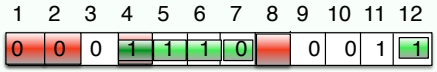
Bit de contrôle de position 1



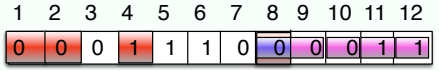
Bit de contrôle de position 2



Bit de contrôle de position 4



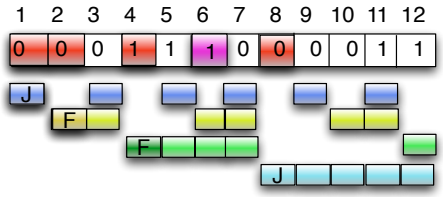
Bit de contrôle de position 8



- A destination on recalcule les bits de contrôle...

A la réception

Trame reçue



- Calcul de 1 (1, 3, 5, 7, 9 , 11): 2 bits à 1: pair donc juste
- Calcul de 2 (2, 3, 6, 7, 10, 11): 1 bit à 1 : impair donc faux
- Calcul de 4 (4, 5, 6, 7, 12) : 3 bits à 1 : impair donc faux
- Calcul de 8 (9,10, 11, 12) : 2 bits à 1: pair donc juste
- La somme des numéros des bits de contrôle erronés donne le numéro du bit qui porte l'erreur
  - 2 + 4 = 6 , le 6ème bit est faux, on peut le corriger

Le code correcteur de Hamming

- Quelle est la distance de Hamming de ce code correcteur ?
  - Si on change un bit de donnée, forcément au moins deux bits de contrôle change aussi, donc h=3
- Que se passe-t il si c'est un bit de contrôle qui est erroné ?
  - Seul une erreur apparait pour ce bit de contrôle, cela ne sert à rien mais on peut le corriger

Le code correcteur de Hamming

- Que se passe-t il si deux bits sont erronés ?
- Exemple

Trame envoyée

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	1	1	1	0	0	0	0	1	1

Trame reçue

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	1	0	0	0	0	0	1	1

- On trouve impairs pour un 1et 4,
  - On corrige le bit 5, on rajoute une troisième erreur !!!

Le code correcteur de Hamming généralisé

- On rajoute un bit de contrôle supplémentaire
- Bit de parité sur l'ensemble de la trame

Trame envoyée

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	1	1	1	0	0	0	0	1	1	1

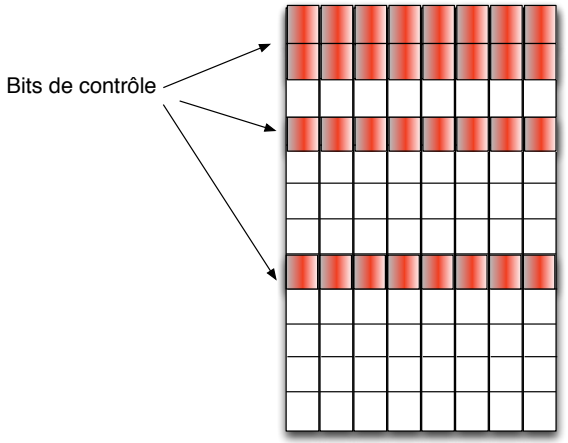
Trame reçue

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	1	1	0	0	0	0	0	1	1	1

- On trouve impairs pour un 1et 4,
  - Le bit de parité supplémentaire est juste, il devrait être faux si le bit 5 seulement était faux
  - On ne peut pas corriger mais on a détecté au moins 2 erreurs

Correction de rafale d'erreur

- On peut utiliser la correction d'erreur de Hamming en calculant les bits de contrôle sur les colonnes des bits de données



Autres exemple de codes correcteur utilisés

- Code de Reed-Salomon: utilisé pour l'ADSL et les DVDs/CDs
  - Permet de corriger de grosses rafales d'erreur (rayure sur un CD)
  - Il est lié à des techniques d'entrelacement (dispersion des erreurs consécutives)