Codes détecteurs et correcteurs d'erreurs

Malika More
(malika.more@u-clermont1.fr)

IREM Clermont-Ferrand

Formation ISN

02 décembre 2013

Nobody's perfect

Canaux de transmission et supports de stockage

Ils sont forcément imparfaits !

- Lors du transfert et/ou du stockage d'information, des erreurs surviennent.
- On souhaite détecter les erreurs de transmission et/ou de stockage et les corriger.
- C'est le rôle des codes détecteurs et correcteurs d'erreurs.
- On utilise une stratégie de surcodage
  - Plus efficace que de recommencer trop fréquemment la transmission ou de stocker l'information en la dupliquant.

Canal Binaire symétrique

- Nous utilisons le modèle idéalisé et quelque peu simpliste du canal binaire symétrique paramétré par une probabilité $0 \leq p \leq 1$ fixée.
- Chaque bit b envoyé est mal transmis avec probabilité p (c'est-à-dire 0 changé en 1 ou vice versa).
- Les erreurs sont indépendantes au cours du temps.

Exemples

- probabilité de mal transmettre 2 bits de suite : $p \times p$

  (car les deux événements sont indépendants)
- probabilité de mal transmettre 1 bit, puis transmettre correctement le suivant, puis un autre à nouveau mal : $p \times (1 - p) \times p$

Codage et décodage

Un peu de vocabulaire

Avant émission

- L'information est surcodée par ajout de bits de contrôle
- Le mot binaire obtenu est le message émis

Codage et décodage

Un peu de vocabulaire

Avant émission

- L'information est surcodée par ajout de bits de contrôle
- Le mot binaire obtenu est le message émis

C'est le codage

Codage et décodage

Un peu de vocabulaire

Avant émission

- L'information est surcodée par ajout de bits de contrôle
- Le mot binaire obtenu est le message émis

C'est le codage

Après réception

- À partir du message reçu, en utilisant les bits de contrôle
- On détecte les erreurs
- On corrige le message reçu pour retrouver l'information initiale.

Codage et décodage

Un peu de vocabulaire

Avant émission

- L'information est surcodée par ajout de bits de contrôle
- Le mot binaire obtenu est le message émis

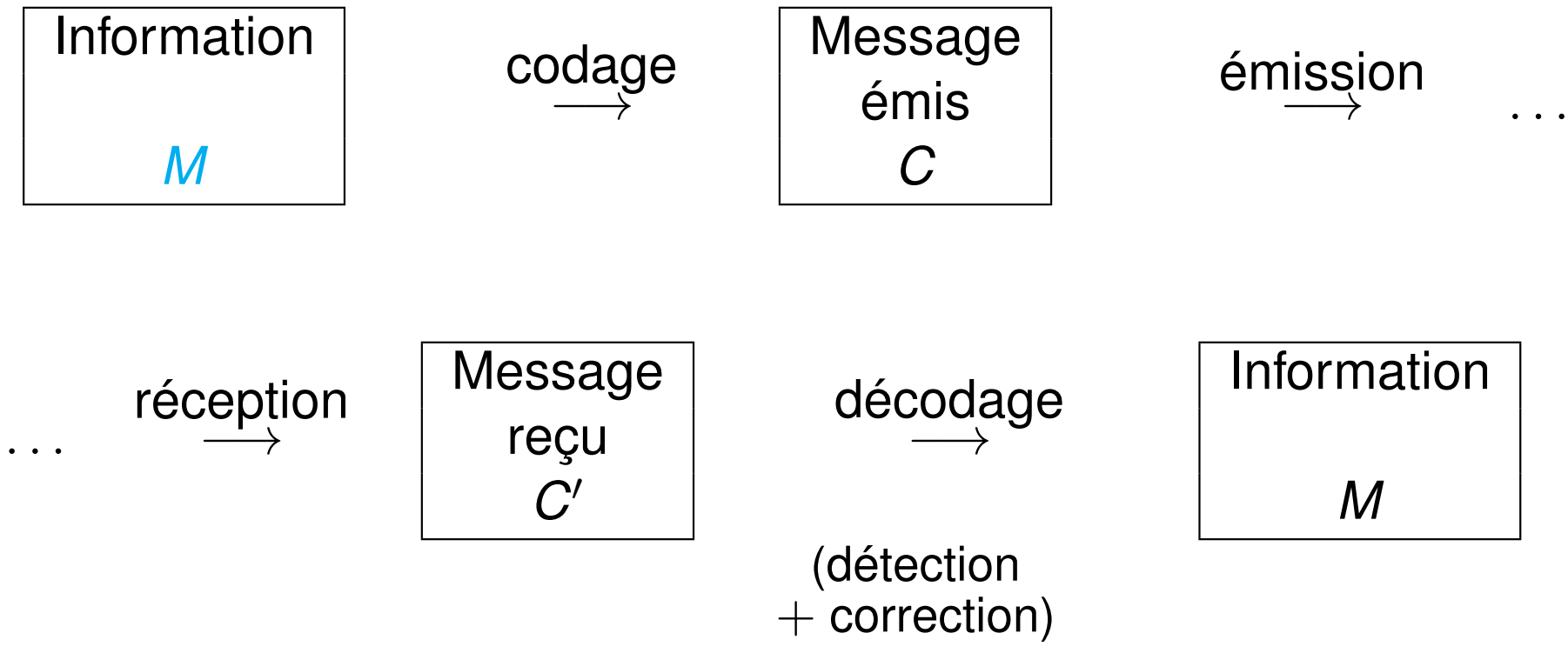
C'est le codage

Après réception

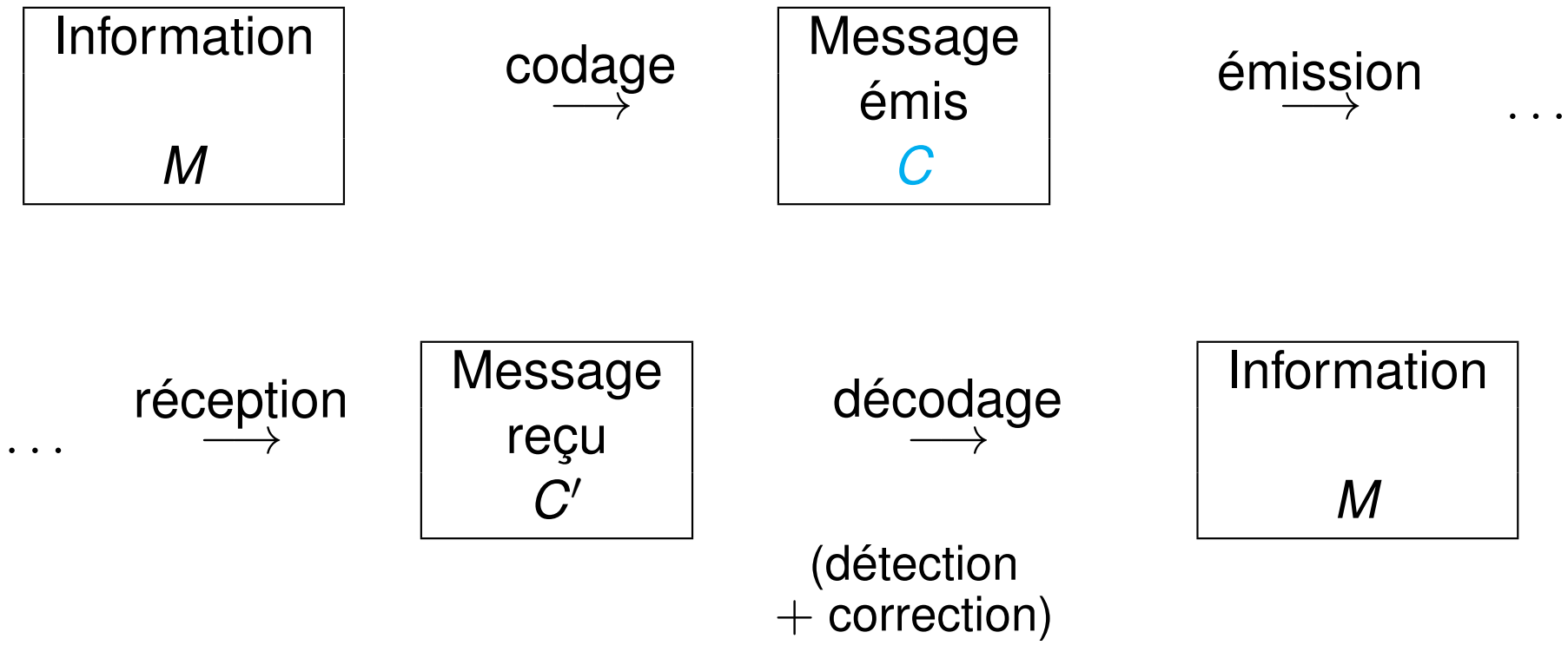
- À partir du message reçu, en utilisant les bits de contrôle
- On détecte les erreurs
- On corrige le message reçu pour retrouver l'information initiale.

C'est le décodage

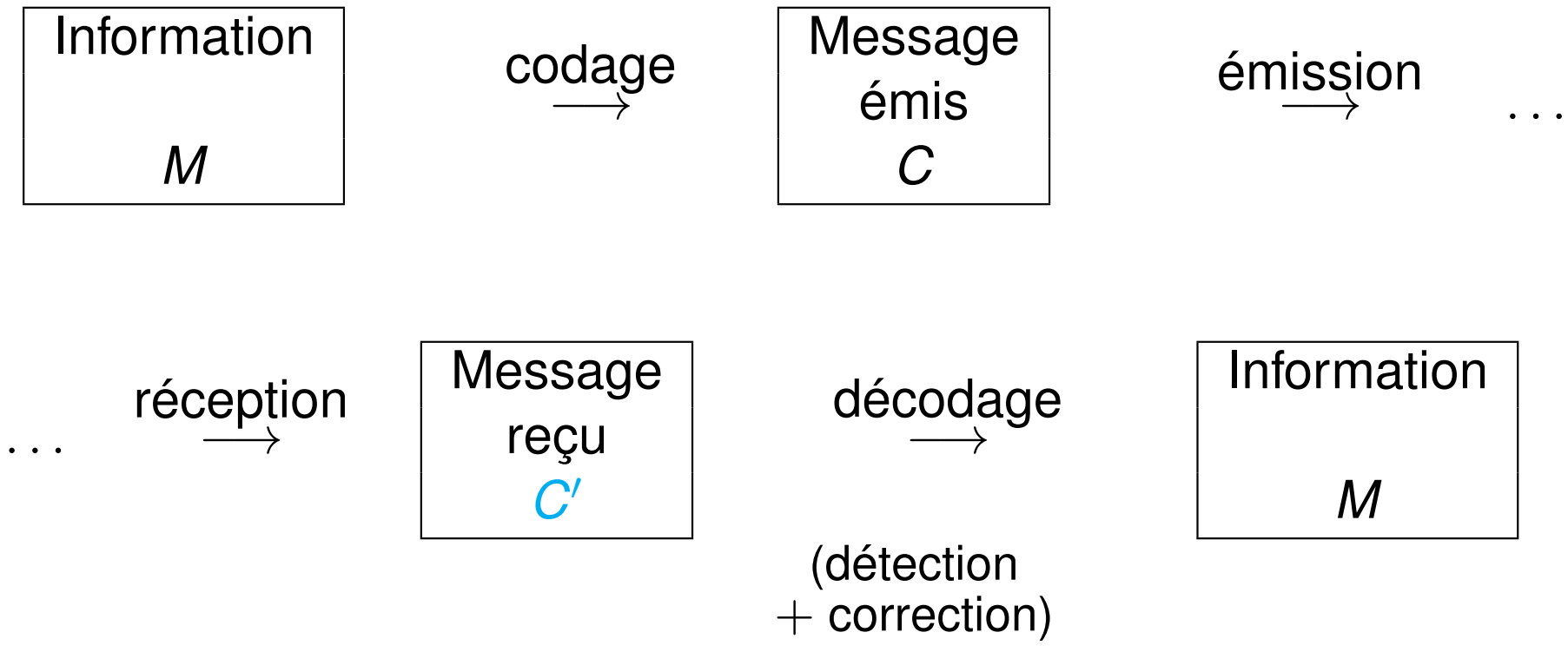
Résumé


Information
M
codage
Message
émis
C
émission
...
...
réception
Message
reçu
C′
décodage
(détection
+ correction)
Information
M

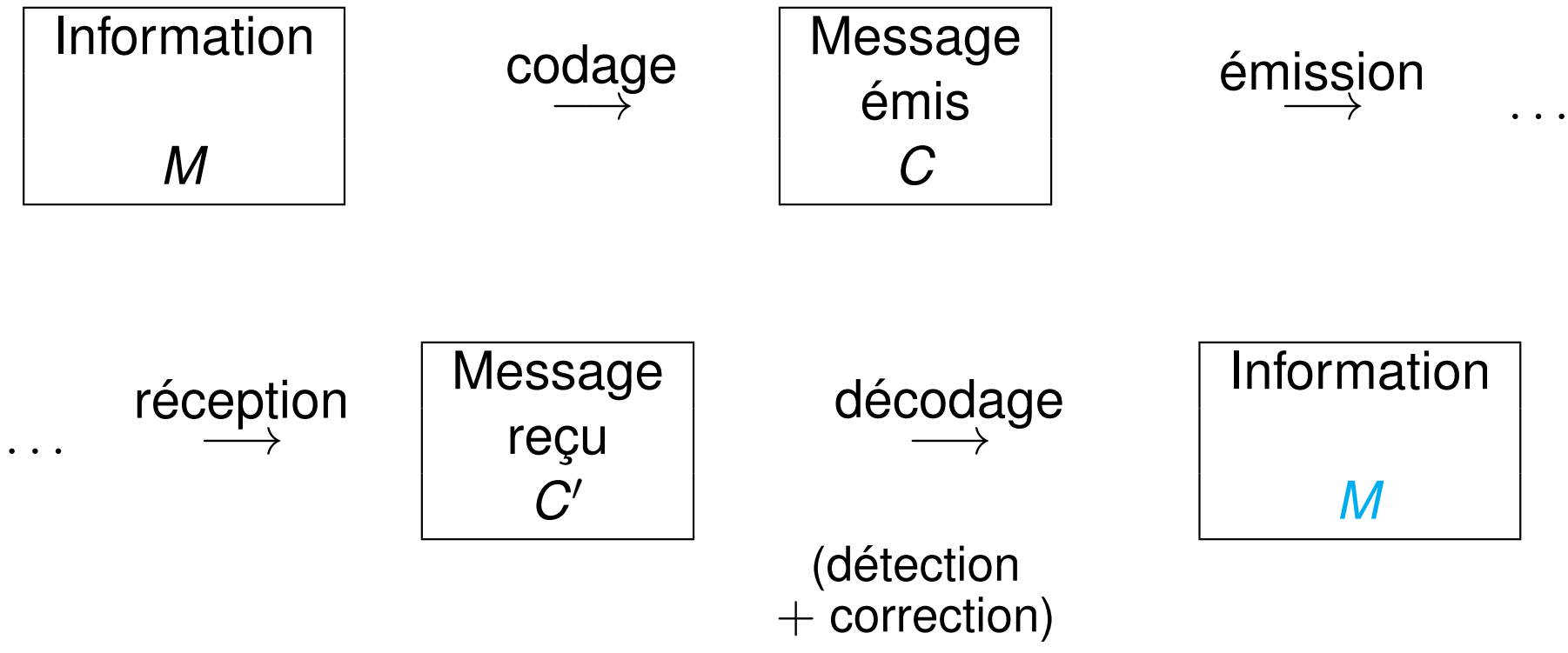
Résumé


Information
M
codage
Message
émis
C
émission
...
...
réception
Message
reçu
C′
décodage
(détection
+ correction)
Information
M

Résumé


Information
M
codage
Message
émis
C
émission
...
...
réception
Message
reçu
C′
décodage
(détection
+ correction)
Information
M

Résumé


Information
M
codage
Message
émis
C
émission
...
...
réception
Message
reçu
C′
décodage
(détection
+ correction)
Information
M

Code de type $\mathcal{C}_{n,m}$

- L'information est découpée en blocs de longueur fixe :

codage

mot d'information : m bits $\longrightarrow$ mot de code : n bits = m bits d'information + r bits de contrôle

- Redondance : $r = n - m =$ nombre de bits de contrôle
- Rendement : $\rho = \frac{m}{n} = \frac{m}{m+r}$

Code de type $\mathcal{C}_{n,m}$

- L'information est découpée en blocs de longueur fixe :

codage

mot d'information
m bits
$\longrightarrow$
mot de code
n bits
=
m bits d'information
+
r bits de contrôle

- Redondance : $r = n - m =$ nombre de bits de contrôle
- Rendement : $\rho = \frac{m}{n} = \frac{m}{m+r}$

Un code pour détecter mais pas corriger

Le code de parité

- On ajoute 1 seul bit de contrôle de telle sorte que le nombre de 1 soit pair
- Si il y a 1 erreur (et plus généralement un nombre impair d'erreurs) alors on peut le détecter au décodage
- Il suffit de calculer la parité du nombre de 1 : si c'est impair, alors il y a nécessairement une erreur.

Dans ce cas, $r = 1$

Par exemple pour $m = 4$.

Un code pour détecter mais pas corriger

Le code de parité

- On ajoute 1 seul bit de contrôle de telle sorte que le nombre de 1 soit pair
- Si il y a 1 erreur (et plus généralement un nombre impair d'erreurs) alors on peut le détecter au décodage
- Il suffit de calculer la parité du nombre de 1 : si c'est impair, alors il y a nécessairement une erreur.

Dans ce cas, $r = 1$

Par exemple pour $m = 4$.
Info. 1001
Code. 10010
Reçu. 10110

Un code pour détecter mais pas corriger

Le code de parité

- On ajoute 1 seul bit de contrôle de telle sorte que le nombre de 1 soit pair
- Si il y a 1 erreur (et plus généralement un nombre impair d'erreurs) alors on peut le détecter au décodage
- Il suffit de calculer la parité du nombre de 1 : si c'est impair, alors il y a nécessairement une erreur.

Dans ce cas, $r = 1$

Par exemple pour $m = 4$.
Info. 1001
Code. 10010
Reçu. 10110
Nombre impair de 1 : erreur détectée.

Un code pour détecter mais pas corriger

Le code de parité

- On ajoute 1 seul bit de contrôle de telle sorte que le nombre de 1 soit pair
- Si il y a 1 erreur (et plus généralement un nombre impair d'erreurs) alors on peut le détecter au décodage
- Il suffit de calculer la parité du nombre de 1 : si c'est impair, alors il y a nécessairement une erreur.

Dans ce cas, $r = 1$

Par exemple pour $m = 4$.
Info. 1001
Code. 10010
Reçu. 10111

Un code pour détecter mais pas corriger

Le code de parité

- On ajoute 1 seul bit de contrôle de telle sorte que le nombre de 1 soit pair
- Si il y a 1 erreur (et plus généralement un nombre impair d'erreurs) alors on peut le détecter au décodage
- Il suffit de calculer la parité du nombre de 1 : si c'est impair, alors il y a nécessairement une erreur.

Dans ce cas, $r = 1$

Par exemple pour $m = 4$.
Info. 1001
Code. 10010
Reçu. 10111
Nombre pair de 1 : 2 erreurs mais pas détectées.

Un code pour détecter et corriger

Le code par répétition

- On répète chaque bit du message k fois (k impair).
- Si il y a strictement moins de k erreurs alors on peut le détecter puisque tous les bits reçus devraient être les mêmes et que ce n'est pas le cas.
- Pour la correction, on corrige au bit le plus fréquent (toujours possible puisque k est impair)

Dans ce cas, $r = (k-1)m$

Pour $m = 1$ et $k = 3$ répétitions, soit $n = 3$ et $r = 2$.
Info. 1
Code. 111
Reçu. 110

Un code pour détecter et corriger

Le code par répétition

- On répète chaque bit du message k fois (k impair).
- Si il y a strictement moins de k erreurs alors on peut le détecter puisque tous les bits reçus devraient être les mêmes et que ce n'est pas le cas.
- Pour la correction, on corrige au bit le plus fréquent (toujours possible puisque k est impair)

Dans ce cas, $r = (k-1)m$

Pour $m = 1$ et $k = 3$ répétitions, soit $n = 3$ et $r = 2$.
Info. 1
Code. 111
Reçu. 110
Pas tous égaux : erreur détectée.
On corrige à 1 (bit le plus fréquent).

Un code pour détecter et corriger

Le code par répétition

- On répète chaque bit du message k fois (k impair).
- Si il y a strictement moins de k erreurs alors on peut le détecter puisque tous les bits reçus devraient être les mêmes et que ce n'est pas le cas.
- Pour la correction, on corrige au bit le plus fréquent (toujours possible puisque k est impair)

Dans ce cas, $r = (k-1)m$

Pour $m = 1$ et $k = 3$ répétitions, soit $n = 3$ et $r = 2$.
Info. 1
Code. 111
Reçu. 010

Un code pour détecter et corriger

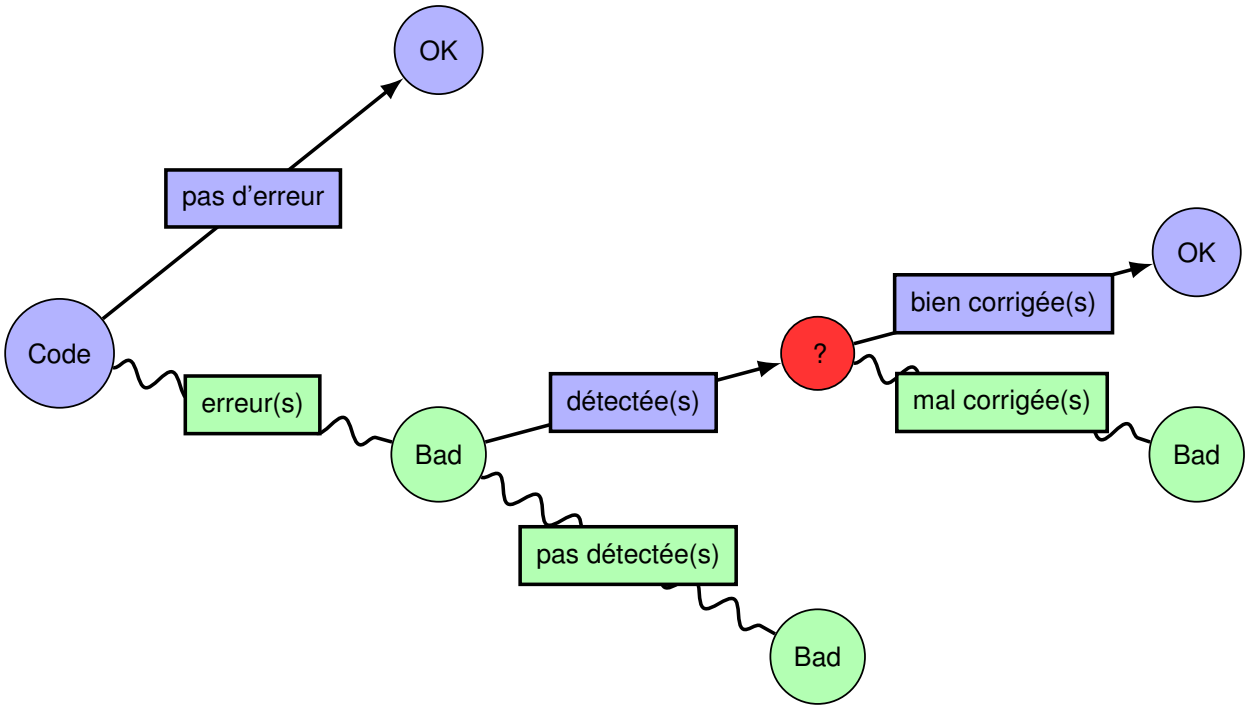
Le code par répétition

- On répète chaque bit du message k fois (k impair).
- Si il y a strictement moins de k erreurs alors on peut le détecter puisque tous les bits reçus devraient être les mêmes et que ce n'est pas le cas.
- Pour la correction, on corrige au bit le plus fréquent (toujours possible puisque k est impair)

Dans ce cas, $r = (k-1)m$

Pour $m = 1$ et $k = 3$ répétitions, soit $n = 3$ et $r = 2$.
Info. 1
Code. 111
Reçu. 010
Pas tous égaux : erreur détectée.
On corrige à 0 (bit le plus fréquent).

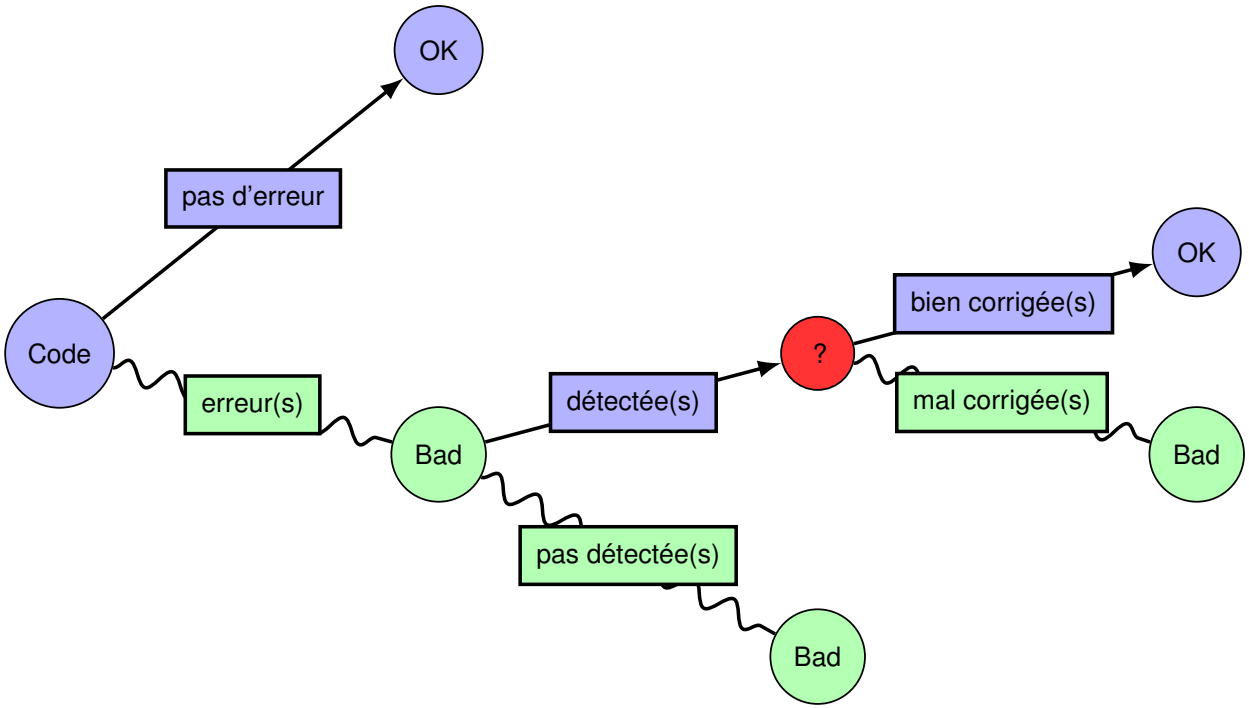
La perfection n'est pas possible



Quel est l'intérêt ?

On fait le pari que les bons cas seront beaucoup plus fréquents que les mauvais, et que en moyenne utiliser un code détecteurs et correcteurs d'erreurs sera mieux que de ne rien faire.

La perfection n'est pas possible



Quel est l'intérêt ?

On fait le pari que les bons cas seront beaucoup plus fréquents que les mauvais, et que en moyenne utiliser un code détecteurs et correcteurs d'erreurs sera mieux que de ne rien faire.

Le pari de la détection des erreurs

Message sans erreur détectée

- Il est possible qu'il soit quand même erroné
  - "sans erreur détectée" signifie que le message reçu est un mot de code
  - Le message envoyé était aussi un mot de code, mais on ne peut pas savoir si c'est le même
  - Peut-être que les erreurs qui se sont produites ont transformé le message envoyé en un autre mot de code
- On fait le pari qu'il n'y a effectivement pas eu d'erreurs
  - Autrement dit que le message reçu est identique au message envoyé
  - Est-ce un pari risqué ?
  - $\rightsquigarrow$ taux de messages erronés détectés

Le pari de la correction des erreurs

Message reçu reconnu erroné puis corrigé

- Il est possible qu'il soit en fait mal corrigé
  - "corrigé" signifie que le message reçu erroné est remplacé par le mot de code le plus proche
  - Le message envoyé était aussi un mot de code, mais on ne peut pas savoir si c'est le même
  - Peut-être que les erreurs puis la correction ont transformé le message envoyé en un autre mot de code
- On fait le pari que la correction a atteint son but
  - Autrement dit que le message corrigé est identique au message envoyé
  - Est-ce un pari risqué ?
  - $\rightsquigarrow$ taux de messages reconnus erronés bien corrigés

Pourquoi ça marche ?

Hypothèse de travail

- Pour $0 \leq k < k' \leq n$, il est beaucoup plus probable de recevoir un message avec k erreurs qu'avec k' erreurs
- Quand on ne détecte aucune erreur dans le message reçu, il est très probable qu'il ne contient effectivement aucune erreur
- Quand on corrige un message erroné par le mot de code le plus proche, il est très probable qu'il soit corrigé correctement

Remarque

- Pour tous les exemples réalistes, ceci est vrai.
- On verra que c'est presque toujours le cas dans notre cadre de travail théorique (le canal binaire symétrique).

Messages possibles vs. mots de code

- Bon sens :

  mots d'information $\neq$ $\longrightarrow$ mots de code $\neq$

- Mots d'information de m bits :

  2^m mots d'information $\longrightarrow$ 2^m mots de code

- Mots de code de n bits :

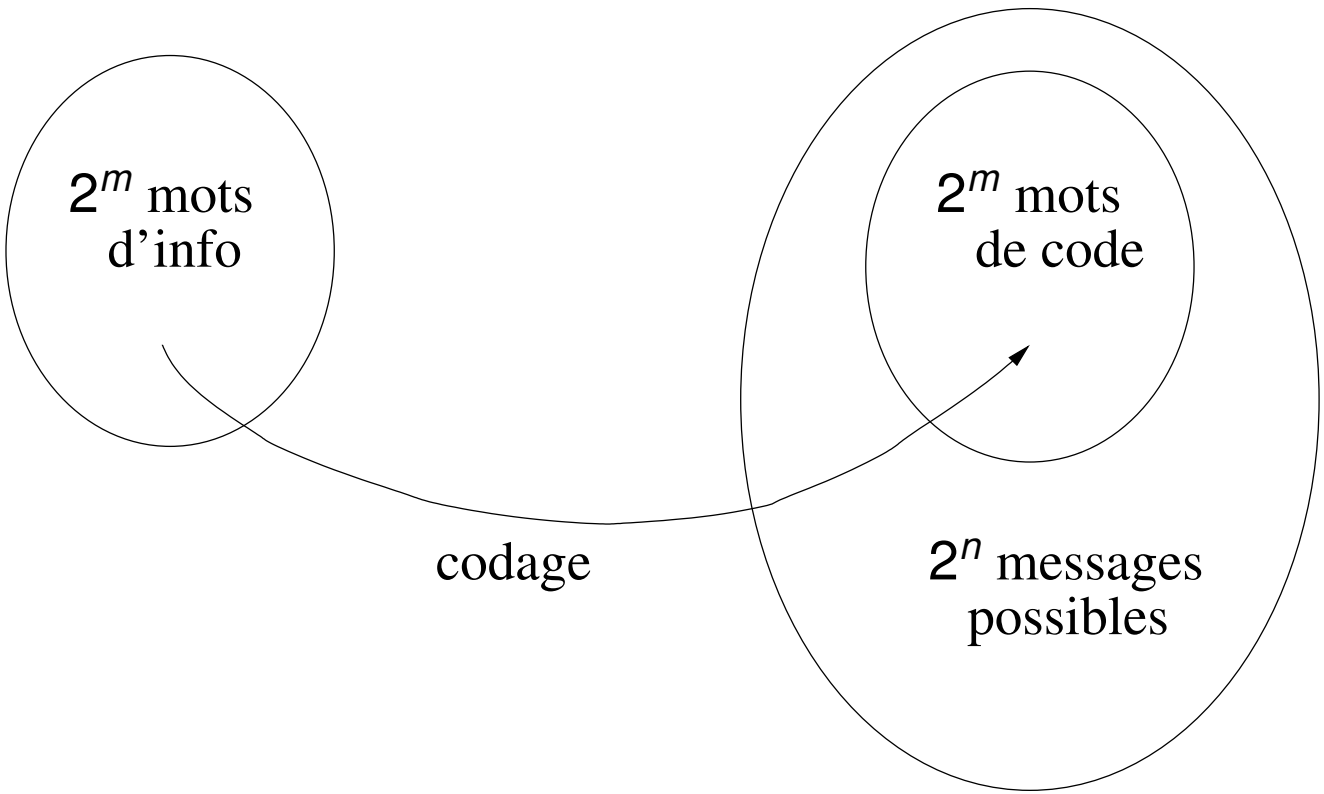
  $\boxed{2^n \text{ messages possibles}}$ parmi lesquels

  seulement

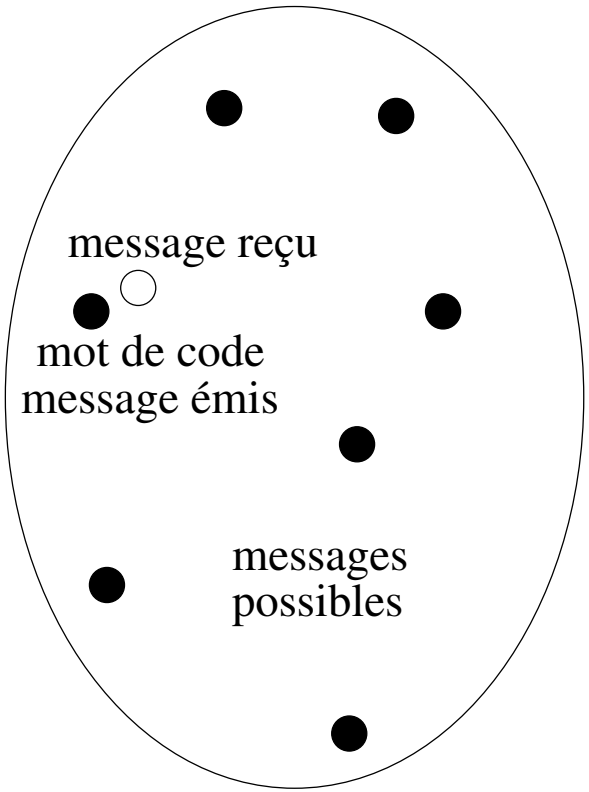
  $\boxed{2^m \text{ mots de code}}$

Messages possibles vs. mots de code

- C'est ce qui va rendre possible la détection et la correction d'erreurs



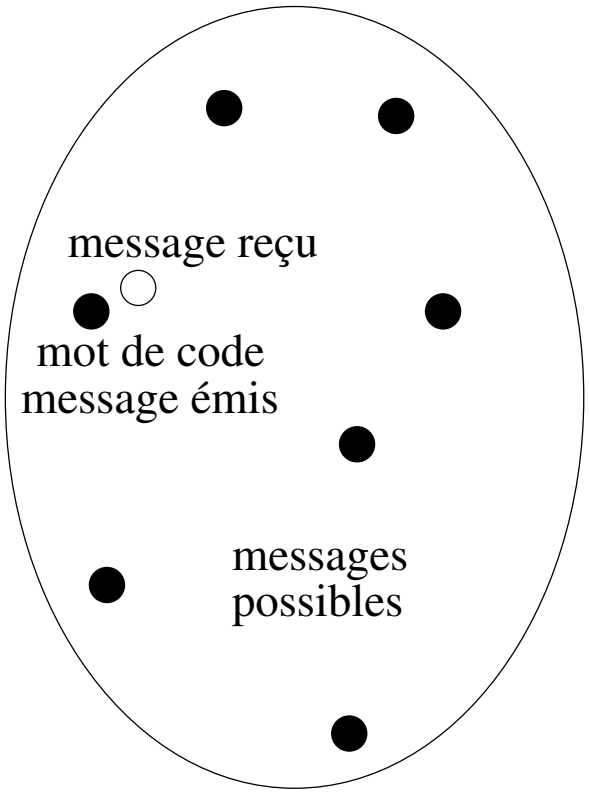
Détection d'erreurs



- S'il y a des erreurs alors le message reçu n'est pas un mot de code
- Conséquence :

Détection

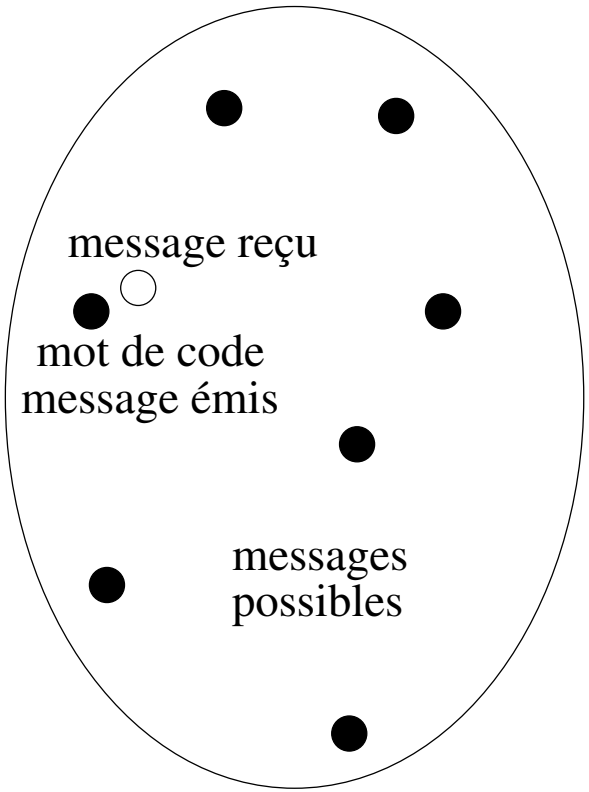
Détection d'erreurs



- S'il y a des erreurs - *jusqu'à un certain point* - alors le message reçu n'est pas un mot de code
- Conséquence :

  Détection*...imparfaite*

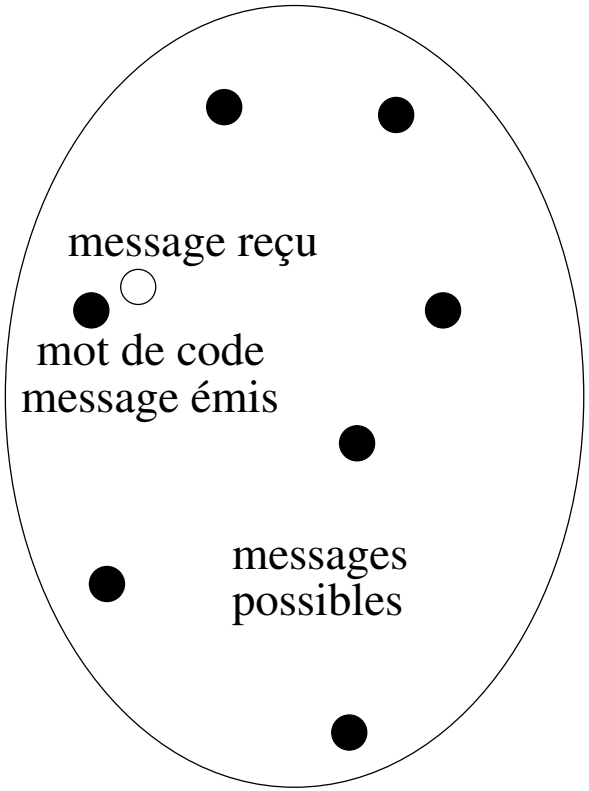
Correction d'erreurs



- Remplacer le message reçu par le mot de code *le plus proche*
- Conséquence :

  Correction

Correction d'erreurs



- Remplacer le message reçu par le mot de code *le plus proche*
- Conséquence :

  Correction. . . imparfaite

Efficacité

Pour le choix d'un code :

- On fixe la longueur m des mots d'information.
- On veut un code qui détecte/corrige bien les erreurs :
  - Redondance élevée $\longrightarrow$ r grand
- On veut un code qui est rapide/court :
  - Rendement élevé $\longrightarrow$ ρ grand
- On doit faire un compromis entre :
  - Efficacité de la détection/correction vs. efficacité de la transmission
  - Redondance r vs. rendement ρ

Matrice génératrice du code ou matrice de codage

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

Utilisation de la matrice génératrice $GI = M$

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \quad = \quad \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

matrice génératrice mot d'info mot de code

$$G \qquad I \qquad = \qquad M$$

Structure de la matrice génératrice G

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

- Code de type $\mathcal{C}_{7,4}$
  - Longueur des mots de code $n = 7$ bits
  - Longueur des mots d'info $m = 4$ bits
- Matrice génératrice G
  - Nombre de lignes $n = 7$
  - Nombre de colonnes $m = 4$

$$I_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Matrice identité I_m
  - Nombre de lignes $m = 4$
  - Nombre de colonnes $m = 4$

$$Q = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

- Matrice de surcodage Q
  - Nombre de lignes $r = n - m = 3$
  - Nombre de colonnes $m = 4$

Rôle des matrices I_m et Q

- La matrice identité I_m sert à recopier les bits d'information

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

- La matrice de surcodage Q sert à calculer les bits de contrôle

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Matrice de contrôle H

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Matrice de contrôle H

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matrice de contrôle H

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

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

$$I_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Q = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Utilisation de la matrice de contrôle

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \\ \\ \end{pmatrix}$$

matrice de contrôle message reçu *syndrome*

H M $= S$

Utilisation de la matrice de contrôle

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

matrice de contrôle — message reçu — *syndrome*

$$H \qquad M = S$$

Syndrome d'un message $HM = S$

- Le message reçu est un mot de code : syndrome nul

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- Le message reçu n'est pas un mot de code : syndrome non nul

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Structure de la matrice de contrôle H

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- Code de type $\mathcal{C}_{7,4}$
  - Longueur des mots de code $n = 7$ bits
  - Longueur des mots d'info $m = 4$ bits
- Matrice de contrôle H
  - Nombre de lignes $r = n - m = 3$
  - Nombre de colonnes $n = 7$

$$Q = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

- Matrice de surcodage Q
  - Nombre de lignes $r = 3$
  - Nombre de colonnes $m = 4$

$$I_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Matrice identité I_r
  - Nombre de lignes $r = 3$
  - Nombre de colonnes $r = 3$

Rôle des matrices Q et I_r

- La matrice de surcodage Q agit sur les bits d'info reçus et sert à recalculer les bits de contrôle

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

- La matrice identité I_r agit sur les bits de contrôle reçus et sert à les recopier

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

- La somme $\oplus$ permet de tester si les deux colonnes obtenues sont identiques

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Rôle des matrices Q et I_r

- La matrice de surcodage Q agit sur les bits d'info reçus et sert à recalculer les bits de contrôle

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

- La matrice identité I_r agit sur les bits de contrôle reçus et sert à les recopier

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

- La somme $\oplus$ permet de tester si les deux colonnes obtenues sont identiques

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Détection des erreurs

Propriété

M est un mot de code ssi son syndrome $S = HM = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$

Message reçu	Syndrome	Contrôle
M	$HM = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	OK
M'	$HM' \neq \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$	Anomalie

Un seul bit erroné

Le message reçu n'est pas un mot de code : syndrome non nul

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Le syndrome donne le numéro du bit erroné (écrit et binaire, poids faibles en haut)

Deux bits erronés ou plus

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

Même si l'erreur est détectée, la correction est incorrecte !

Données stockées ou transmises

Il y a parfois des erreurs

- Perturbations (par exemple électromagnétiques)
- Reconnaître si les données ont été altérées (détection)
- Reconstituer les données d'origine (correction)

Données stockées ou transmises

Il y a parfois des erreurs

- Perturbations (par exemple électromagnétiques)
- Reconnaître si les données ont été altérées (détection)
- Reconstituer les données d'origine (correction)

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

$\rightsquigarrow$

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

Bob transmet un bloc de données à Alice

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

Bob ajoute des données redondantes

- Tout est écrit en binaire
- Il crée une ligne et une colonne supplémentaires

Bob transmet un bloc de données à Alice

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

Règle de calcul des valeurs supplémentaires

- Le nombre de 1 sur chaque ligne doit être pair
- Le nombre de 1 sur chaque colonne doit être pair

Bob transmet un bloc de données à Alice

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

Règle de calcul des valeurs supplémentaires

- Le nombre de 1 sur chaque ligne doit être pair
- Le nombre de 1 sur chaque colonne doit être pair

Bob transmet un bloc de données à Alice

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

Pendant la transmission

- Une perturbation se produit : une valeur est modifiée
- Une ligne et une colonne ne respectent plus la règle

Bob transmet un bloc de données à Alice

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

Pendant la transmission

- Une perturbation se produit : une valeur est modifiée
- Une ligne et une colonne ne respectent plus la règle

Bob transmet un bloc de données à Alice

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

Pendant la transmission

- Une perturbation se produit : une valeur est modifiée
- Une ligne et une colonne ne respectent plus la règle

Bob transmet un bloc de données à Alice

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

À la réception

- Alice ne sait pas si les données ont été altérées
- Elle ne sait pas non plus ce que Bob lui a envoyé

Bob transmet un bloc de données à Alice

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

Mais Alice constate

- Certaines lignes et colonnes ne respectent pas la règle
- C'est une preuve qu'une perturbation s'est produite
- Elle détecte l'altération dans les données transmises

Bob transmet un bloc de données à Alice

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

Alice peut faire mieux

- Il lui suffit de changer la valeur à l'intersection pour que la règle soit de nouveau respectée
- Elle corrige l'altération dans les données transmises

Bob transmet un bloc de données à Alice

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

Alice peut faire mieux

- Il lui suffit de changer la valeur à l'intersection pour que la règle soit de nouveau respectée
- Elle corrige l'altération dans les données transmises

Bob transmet un bloc de données à Alice

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

Alice peut faire mieux

- Il lui suffit de changer la valeur à l'intersection pour que la règle soit de nouveau respectée
- Elle corrige l'altération dans les données transmises

Bob transmet un bloc de données à Alice

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

Et maintenant

- Toutes les altérations peuvent-elles être décelées ?
- La correction effectuée est-elle toujours la bonne ?

Bob transmet un bloc de données à Alice

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

Toutes les altérations peuvent-elles être décelées ?

- Supposons que Bob envoie ceci

Bob transmet un bloc de données à Alice

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

Toutes les altérations peuvent-elles être décelées ?

- Et supposons qu'Alice reçoive ceci
- Toutes les lignes et les colonnes respectent la règle

Bob transmet un bloc de données à Alice

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

Toutes les altérations peuvent-elles être décelées ?

- Alice pense que la transmission a été correcte
- Ce n'est pas le cas
- Mais elle n'a aucun moyen de s'en apercevoir

Bob transmet un bloc de données à Alice

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

L'altération passe inaperçue

- Ce problème est-il fréquent ?
- Existe-t-il des méthodes permettant de déceler toutes les altérations ?

Bob transmet un bloc de données à Alice

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

La correction proposée est-elle toujours la bonne ?

- Supposons que Bob envoie ceci

Bob transmet un bloc de données à Alice

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

La correction proposée est-elle toujours la bonne ?

- Et supposons qu'Alice reçoive ceci
- Une ligne et une colonne ne respectent pas la règle
- Alice s'aperçoit que les données ont été altérées

Bob transmet un bloc de données à Alice

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

La correction proposée est-elle toujours la bonne ?

- Et supposons qu'Alice reçoive ceci
- Une ligne et une colonne ne respectent pas la règle
- Alice s'aperçoit que les données ont été altérées

Bob transmet un bloc de données à Alice

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

La correction proposée est-elle toujours la bonne ?

- Alice corrige l'intersection de la ligne et de la colonne
- L'erreur n'est pas à cet endroit
- Mais elle n'a aucun moyen de s'en apercevoir

Bob transmet un bloc de données à Alice

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

La correction proposée est-elle toujours la bonne ?

- Alice corrige l'intersection de la ligne et de la colonne
- L'erreur n'est pas à cet endroit
- Mais elle n'a aucun moyen de s'en apercevoir

Bob transmet un bloc de données à Alice

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

La correction n'est pas la bonne

- Ce problème est-il fréquent ?
- Existe-t-il des méthodes permettant de toujours corriger correctement ?