

Chapitre 3 : Protection contre les erreurs

/udd/bcousin/Pages-web-Armor/Enseignement/Reseaux-generalites/Cours/3.auto.fm - 1 juin 2001 18:28

Plan

- 1. Introduction p68
- 2. Les codes de protection contre les erreurs p72
- 3. Exemples simples de codes par bloc p75
- 4. Les codes linéaires p78
- 5. Les codes polynômiaux p83
- 6. Les codes cycliques p88
- 7. Autres techniques de correction d'erreurs p90
- 8. Techniques de contrôle d'erreur par retransmission p91
- 9. Conclusion p95

Bibliographie

- K. Lahèche, Les codes en informatique : codes détecteurs et correcteurs, Hermès, 1995.
- H. Nussbaumer, Téléinformatique - tome 1, Presses polytechniques romandes, 1983.
- C. Macchi, J-F. Guibert, Téléinformatique, Dunod, 1987.
- A.Tanenbaum, Réseaux, InterEditions, 1997.

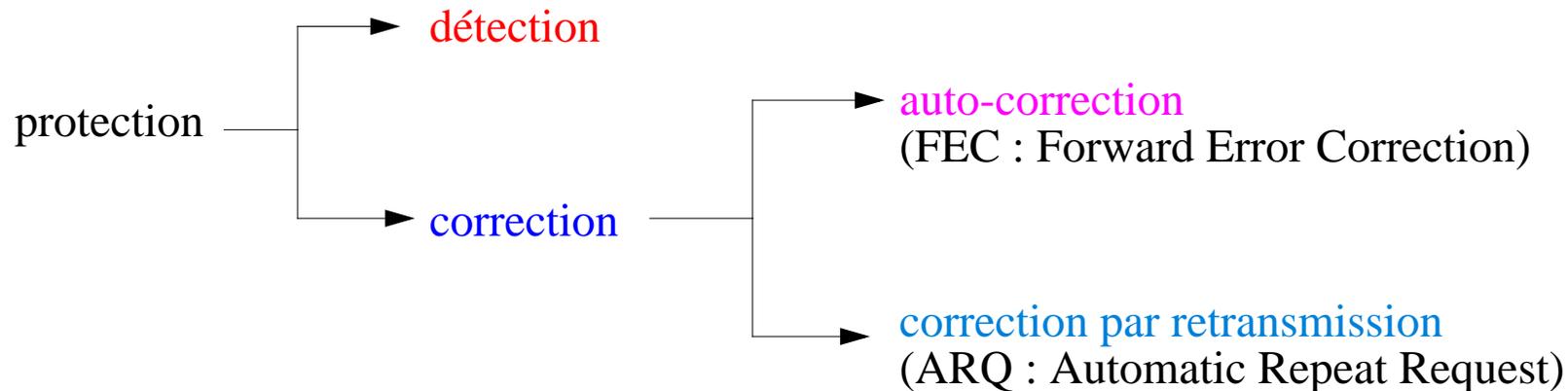
1. Introduction

Quelque 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.

Dans ces conditions, la suite binaire reçue ne sera pas identique à la suite émise.

Mise en oeuvre de techniques de protection contre les erreurs de transmission

Stratégies de protection contre les erreurs de transmission :



1.1. Principe général pour la **détection** des erreurs de transmission

- 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.
- **Note** : certaines erreurs sont indétectables !
 - . ex. : une même erreur sur les deux exemplaires simultanément

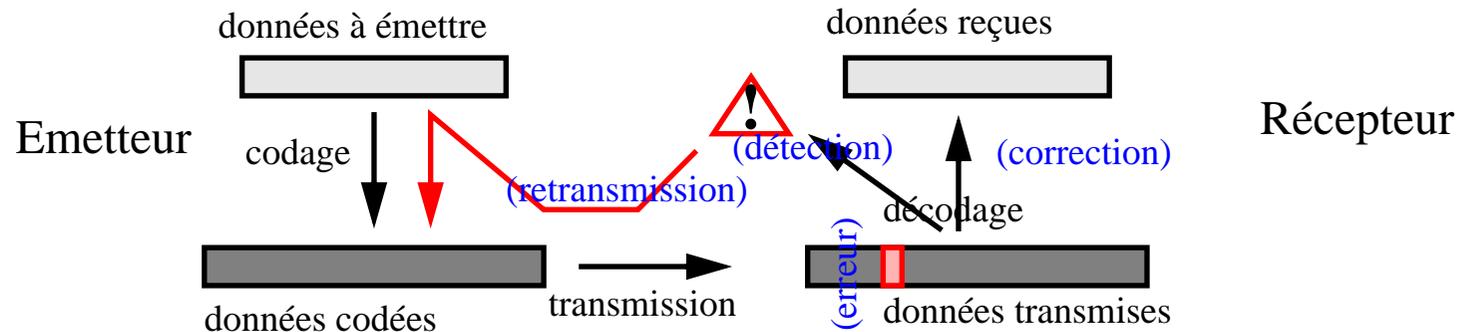
1.2. Principe général pour la **correction** des erreurs de transmission

- Après détection d'une erreur, la redondance est suffisante pour permettre de retrouver le message initial.

- **Exemple** : la technique de correction par répétition
 - . le message codé est un triple exemplaire du message initial, le récepteur suppose que le message initial correspond aux deux exemplaires qui sont identiques.
- **Note** : certaines erreurs détectées ne sont pas corrigibles !
 - . ex. : une erreur différente sur au moins deux exemplaires
- **Note** : certaines erreurs sont détectées et mal corrigées !
 - . ex. : une même erreur sur deux exemplaires simultanément

1.3. Principe général pour la correction par retransmission des erreurs de transmission

- Après détection d'une erreur, le récepteur demande à l'émetteur, implicitement (temporisateur) ou explicitement (nack), de retransmettre une nouvelle fois le message (codé).
- Exemple : de très nombreux protocoles de télécommunication : HDLC, X25, TCP, TP.



La correction par retransmission est préférée dans les réseaux où le taux de perte est faible et le délai de retransmission tolérable, car son surcoût est généralement plus faible que celui induit par les codes auto-correcteurs.

Estimation du surcoût (message de longueur moyenne = 1000 bits) :

- taux d'erreur typique = 10^{-9} , taux de retransmission $\Rightarrow 1000 \cdot 10^{-9} = 10^{-6}$
- surcoût typique d'un code auto-correcteur : qq octets par message $\Rightarrow 8/1000 = 10^{-2}$

2. Les codes de protection contre les erreurs

2.1. Classification des codes

Deux grandes familles de codes :

Les **codes en bloc** (linéaires, cycliques ou non) : le codage/décodage d'un bloc dépend uniquement des informations de ce bloc.

Les **codes en treillis** (convolutifs, récurcifs ou non) : le codage/décodage d'un bloc dépend des informations d'autres blocs (généralement de blocs précédemment transmis).

- Un code convolutif s'applique sur une suite infinie de symboles et produit une suite infinie.

On préfère généralement le codage par bloc dans les applications téléinformatiques classiques :

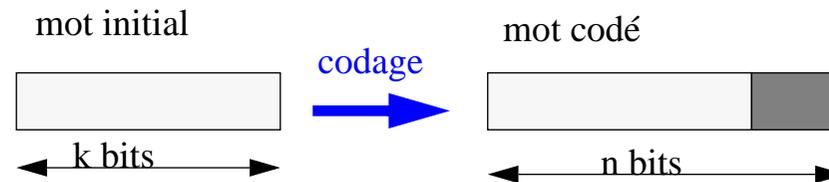
- le codage/décodage est plus simple et il y a moins de délai

Par la suite, on ne va présenter que les codes par bloc :

- codes simples
- codes linéaires, de Hamming
- codes polynômiaux
- codes cycliques

2.2. Définitions générales

Un **code (k, n)** transforme (code) tout bloc initial de k bits d'information en un bloc codé de n bits. Le code introduit une redondance puisque $n \geq k$. Le code est **systematique** si les k premiers bits du bloc codé sont égaux aux bits du bloc initial. Alors 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.

Le **poids de Hamming** d'un mot est le nombre de bits à 1 qu'il contient. La **distance de Hamming** entre deux mots de même longueur est définie par le nombre de positions binaires qui diffèrent entre ces deux mots. On l'obtient par le poids de Hamming de la somme binaire des 2 mots. La distance de Hamming d'un code est la distance minimum entre tous les mots du code.

- Exemple :

. $\text{poids_de_Hamming}(01001100) = 3$

. $\text{distance_de_Haming}(01001100, 01010101) = 3$

La **capacité de détection** (de correction) d'un code est définie par les configurations erronées qu'il est capable de détecter (corriger). Une **erreur simple** (resp. double, ou d'ordre p) affecte une seule (resp. 2, ou p) position(s) binaire(s) d'un mot. Pour qu'un code ait une capacité de détection (resp. correction) des erreurs d'ordre e , il faut que sa distance de Hamming soit supérieure à $1+e$ (resp. $1 + 2e$).

- Exemple :

. $\text{distance} = 3 \implies \text{capacité de détection} = < 2, \text{ capacité de correction} = < 1.$

3. Exemples simples de codes par bloc

3.1. 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).

<u>Lettre</u>	<u>Code ASCII</u>	:	<u>Mot codé (parité paire)</u> :	<u>Mode codé (parité impaire)</u>
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 !

3.2. Parité longitudinale et transversale

Association d'un double codage de la parité :

- LRC : “Longitudinal Redundancy Check” et VRC : “Vertical ...”

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 une matrice ($a+1, b+1$).

VRC (parité paire)

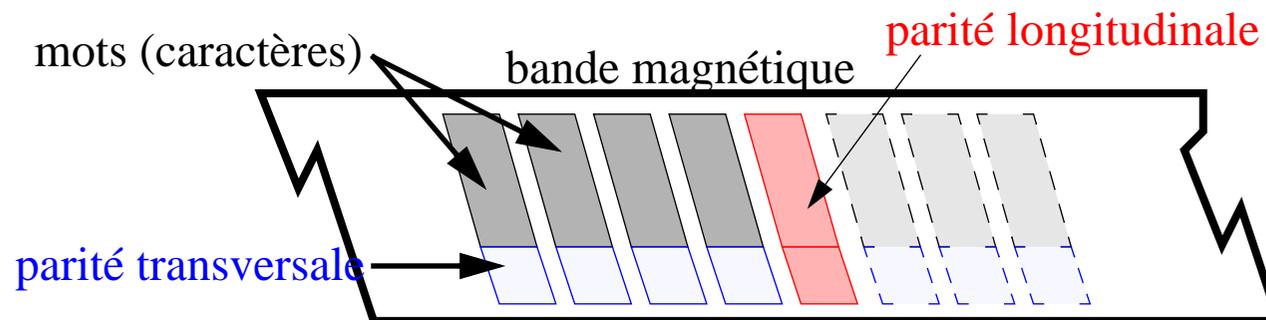
1 0 1 0 0 0 1 1

0 1 1 0 1 0 1 0

1 0 0 0 0 0 1 0

LRC = 0 1 0 0 1 0 1 1

Historique de la dénomination :



Le rendement est faible : $a.b / (a+1).(b+1)$.

Le délai de codage et décodage important : il faut recevoir tout le bloc

Capacité de détection et d'autocorrection :

- Principe : Une erreur simple modifie simultanément la parité d'un ligne et d'une colonne.
- Correction : inverser le bit situé à l'intersection de la ligne et de la colonne ayant une parité incorrecte.

Exemple :

```

1 0 1 0 0 0 1 1
0 1 1 0 1 0 1 0
1 0 0 0 1 0 1 0
0 1 0 0 1 0 1 1
    
```

Attention : une erreur triple peut faire croire à une erreur simple et sa correction sera inadaptée !

Exemple :

```

1 0 1 0 0 0 1 1
0 1 1 0 1 0 1 0
1 0 0 0 1 0 1 0
0 1 0 0 1 0 1 1
    
```

4. Les codes linéaires

4.1. Définitions

Les **codes linéaires** sont des codes dont chaque mot du code (noté c) est obtenu après transformation linéaire des bits du mot initial (noté i).

Ces codes sont caractérisés par leur matrice $G_{(k, n)}$ (appelée **matrice génératrice**) telle que :

$$i \cdot G = c$$

Exemple : $[1 \ 0 \ 1] \cdot \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} = [0 \ 1 \ 1 \ 0]$

La matrice $H_{(n-k, n)}$ (appelée **matrice de contrôle**) permet de savoir si un mot reçu est un mot du code, en calculant son **syndrome**. Si le syndrome du mot est nul, ce mot appartient au code.

$$\text{Syndrome du mot reçu } c' : c' \cdot H^T = 0_{(n-k)} \iff c' \in C_{(k, n)}$$

L'équation $G \cdot H^T = 0$ définit la relation entre les deux matrices.

$$\text{Preuve : } c' \cdot H^T = i \cdot G \cdot H^T = i \cdot 0$$

4.2. Propriétés

La distance de Hamming d'un code linéaire est égale au plus petit poids de Hamming non nul des mots du code.

Si un code linéaire est systématique, sa matrice génératrice s'écrit :

$$- G_{(k, n)} = [\text{Id}_{(k)}, P_{(k, n-k)}]$$

alors sa matrice de contrôle s'écrit : $H_{(n-k, n)} = [P_{(n-k, k)}^T, \text{Id}_{(n-k)}]$

Exemples :

. Soit le code $C_{1(4, 6)}$ de matrice $G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$.

- Le mot de code c associé au mot initial $i = [1 \ 0 \ 1 \ 1]$

est le mot $c = i \cdot G_1 = [1 \ 0 \ 1 \ 0 \ 0 \ 1]$

. Le code $C_{2(7, 8)}$ associé à la parité paire a pour matrice de contrôle $H_2^T = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$.

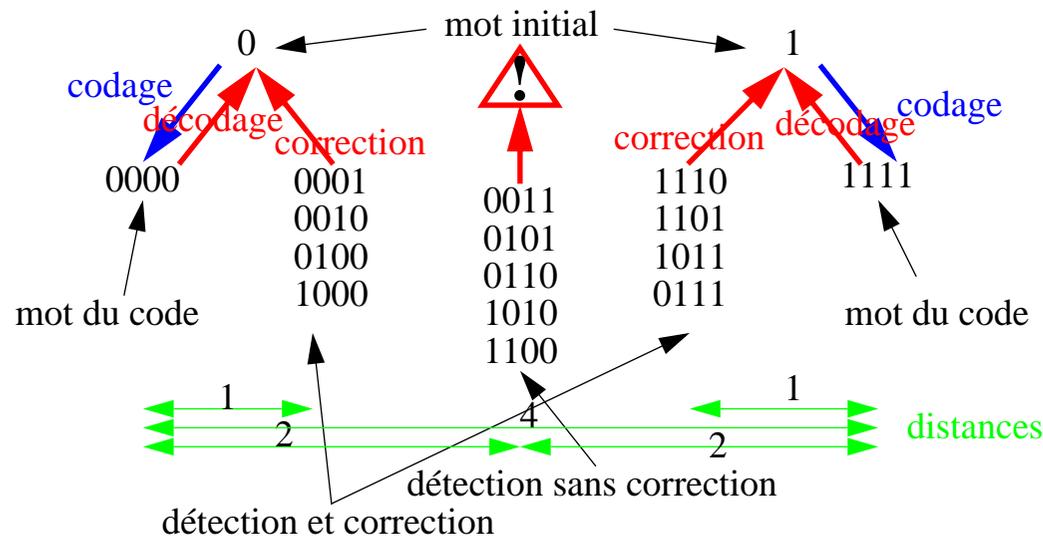
4.3. Correction

Correction par proximité : le procédé de correction transforme un mot reçu et détecté comme erroné dans le mot de code le plus proche au sens de la distance de Hamming.

- Il peut exister plusieurs mots équidistants. On choisit un représentant
- procédé de correction lourd

Exemple :

Le code à répétition $C_{3(1, 4)}$, distance de Hamming de $C_3 = 4$.



4.4. Codes de Hamming

Famille de codes linéaires auto-correcteurs faciles à corriger.

Principe de construction d'un code de **Hamming dense** $C_{(2^m - m - 1, 2^m - 1)}$:

- Chaque colonne de sa matrice de contrôle $H_{(m, 2^m - 1)}$ est **non-nulle** et **différente**.
- Propriétés :
 - . leur distance minimale est égale à 3 (au moins)
 - . correction des erreurs simples et détection des erreurs doubles, au moins.

Auto-correction :

- le syndrome du mot reçu est identique à la colonne de la matrice de contrôle correspondant au bit à corriger.
- si l'on trie les colonnes de H suivant leur poids binaire croissant et que les poids de ses colonnes couvrent l'intervalle $[1, 2^m - 1]$ alors la valeur binaire du syndrome est égale au numéro de bit erroné.

Un code de Hamming est compact si les 2^{n-k} syndromes différents sont utilisés pour identifier les $2^{n-k} - 1$ erreurs simples (+ syndrome nul !).

Exemple :

- Code de Hamming $C_{4(4, 7)}$ tel que sa matrice de contrôle $H_{4(3, 7)} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$.

Calculez la matrice génératrice G_4 puis les syndromes de $c_1 = [0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1]$, de $c_2 = [0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0]$ et de $c_3 = [0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1]$.

Le code est systématique, la transposée de H_4 : $H_4^T = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ donc $G_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$

$c_1 \cdot H_4^T = [0 \ 0 \ 0] \quad \rightarrow i_1 = [0 \ 0 \ 1 \ 0]$

$c_2 \cdot H_4^T = [0 \ 0 \ 1] \quad \rightarrow$ erreur sur le dernier bit $\rightarrow i_2 = [0 \ 0 \ 1 \ 0]$

$c_3 \cdot H_4^T = [1 \ 0 \ 0] \quad \rightarrow$ erreur sur le 5ème bit $\rightarrow i_3 = [0 \ 0 \ 0 \ 1]$

4.5. Conclusion

La méthode matricielle impose de travailler sur des mots de taille fixe, ce qui est un inconvénient majeur pour les réseaux informatiques où les messages sont de taille variable.

5. Les codes polynômiaux

5.1. Présentation

Notation : tout vecteur peut être présenté sous une forme polynômiale :

$$\bullet U = \langle u_0, u_1, u_2, \dots, u_n \rangle \iff U(x) = u_0 + u_1 \cdot x + u_2 \cdot x^2 + \dots + u_n \cdot x^n$$

Attention : les opérations sont binaires (construits sur le corps $\mathbb{Z}/2\mathbb{Z}$) : $1 \cdot x + 1 \cdot x = 0 \cdot x$!

- Somme polynomiale :

. somme deux à deux de chaque coefficient (somme vectorielle)

. exemple : $\langle x^3 + 1 \rangle + \langle x^4 + x^2 + 1 \rangle = \langle x^4 + x^3 + x^2 + 1 \rangle$

- Produit polynomiale :

. produit vectoriel

. exemple : $\langle x^3 + 1 \rangle \cdot \langle x + 1 \rangle = \langle x^4 + x^3 + x + 1 \rangle$

- Division polynomiale

. idem

. Exemple : $\langle x^4 + x^3 + x^2 + x + 1 \rangle / \langle x + 1 \rangle = \langle x^3 + x^2 + 1 \rangle$

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

\Leftrightarrow les lignes de la matrice génératrice sont engendrées par le polynôme générateur.

Le degré du polynôme définit la longueur du champ de contrôle d'erreur.

Exemple :

- . soit le polynôme générateur : $g(x) = x+1$
- . on s'intéresse à un code $C_5(2,3)$ donc les calculs se feront modulo x^3+1
- . première ligne : $g(x) \cdot 1 = x+1$
- . deuxième : $g(x) \cdot (x^2+1) \bmod (x^3+1) = (x^3+x^2+x+1) \bmod (x^3+1) = x^2+x$
- . ce qui donne la matrice génératrice suivante : $G_5(2,3) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$

Exemples de codes polynômiaux :

- L'avis V41 du CCITT conseille l'utilisation de codes polynômiaux (de longueurs $n = 260, 500, 980$ ou 3860 bits) 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 :
 - . $g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$.

5.2. Capacité de détection

Capacité de détection des erreurs :

Un code polynômial (k, n) dont le polynôme générateur a plus d'un coefficient non-nul (donc il ne divise pas $x^i, i < n$) permet de détecter toutes les **erreurs simples**.

- idée de preuve : $(C(x) + x^i) \bmod g(x) \neq 0$

Si le polynôme générateur d'un code polynômial (k, n) a un facteur irréductible de trois termes (il ne divise ni x^i ni $1 + x^{j-i}, i < j < n$), le code permet de détecter les **erreurs doubles**.

. par exemple : $g(x) = x^3 + x + 1$

Pour qu'un code polynômial détecte toutes les **erreurs d'ordre impair**, il suffit que son polynôme générateur ait $(x+1)$ comme facteur.

. Par exemple : le code de polynôme générateur $(x+1)$ qui est équivalent à la parité.

Capacité de détection des paquets d'erreurs:

Un code polynômial (k, n) permet de détecter toutes les **erreurs d'ordre $l \leq n-k$** (c'est-à-dire inférieur au degré du polynôme générateur).

Et la probabilité de ne pas détecter les **erreurs d'ordre $l > n-k$** est très faible et égale à : $2^{-(n-k)}$

5.3. 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 :

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

Remarques :

- (i) Les $r = n-k$ bits de $r(x)$ (de degré $\leq n-k-1$) forment les bits du champ de contrôle.
- (ii) Les bits de poids fort (de degré $> n-k-1$) forment le mot initial (--> code systématique)
- (iii) L'opération de codage effectuée à l'émission est ramenée à une division polynômiale, qui peut être réalisée simplement (électroniquement).

5.4. Principe du décodage

A la réception, chaque mot reçu $m'(x)$ est divisé par le polynôme générateur $g(x)$.

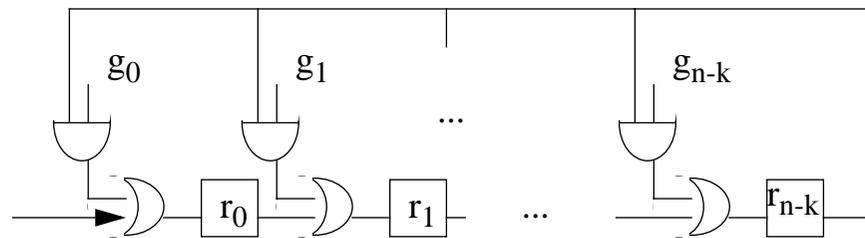
- Un reste **non-nul** indique qu'il y a eu **erreur** lors de la transmission.
- Syndrome de $m'(x)$: $m'(x) \bmod g(x) \neq 0 \implies$ Erreur !

Attention : la réciproque est fautive ! Si le reste est nul cela ne veut pas dire qu'il n'y a pas eu d'erreurs --> subsistance d'erreurs dites **résiduelles**.

5.5. Schéma électronique d'un diviseur polynomial

La multiplication est réalisée par un *ET logique*, l'addition par un *OU exclusif*, plus des registres à décalage.

Le diviseur : $g(x) = g_0 + g_1 \cdot x + g_2 \cdot x^2 + \dots + g_{n-k} \cdot x^{n-k}$



Procédé :

- (i) les registres r_i sont mis à zéros
- (ii) les bits du mot à diviser sont insérés en entrée (k étapes), bits de poids fort en tête.
- (iii) les registres r_i contiennent alors le reste, qu'on extrait ($n-k$ étapes).

De nombreuses optimisations sont possibles :

- Lorsque $g_i=0$ on supprime simplement la connexion et la porte ET !
- phase spécifique d'initialisation, etc.

6. Les codes cycliques

6.1. Définitions

Un **code cyclique** (k, n) est un code linéaire (k, n) tel que toute permutation circulaire d'un mot du code est encore un mot du code.

Exemple :

- Un code cyclique $(1, 2)$ possède les mots de code suivants : $\{01, 10\}$ ou $\{00, 11\}$, mais pas $\{01, 11\}$.
- Un code cyclique $(1, 3)$ possède les mots de code suivants : $\{000, 111\}$.

On appelle **période** du polynôme $H(x)$ le plus petit entier u tel que $H(x)$ divise $x^u + 1$.

Un polynôme est **irréductible** s'il ne possède aucun diviseur de degré supérieur à zéro.

Si la période d'un polynôme irréductible est égale à $n-k$ alors le polynôme est dit **primitif**.

6.2. Propriétés/autocorrection

Un code cyclique (k, n) est un code polynômial dont le polynôme générateur divise $x^n + 1$.

Les dispositifs de codage et de décodage sont identiques à ceux des codes polynômiaux.

Les codes cycliques sont principalement utilisés pour leur capacité de correction.

Les codes cycliques (k, n) dont le polynôme générateur est **primitif** et tel que $n=2^{n-k} + 1$, sont des codes de Hamming.

6.3. Codes correcteurs performants

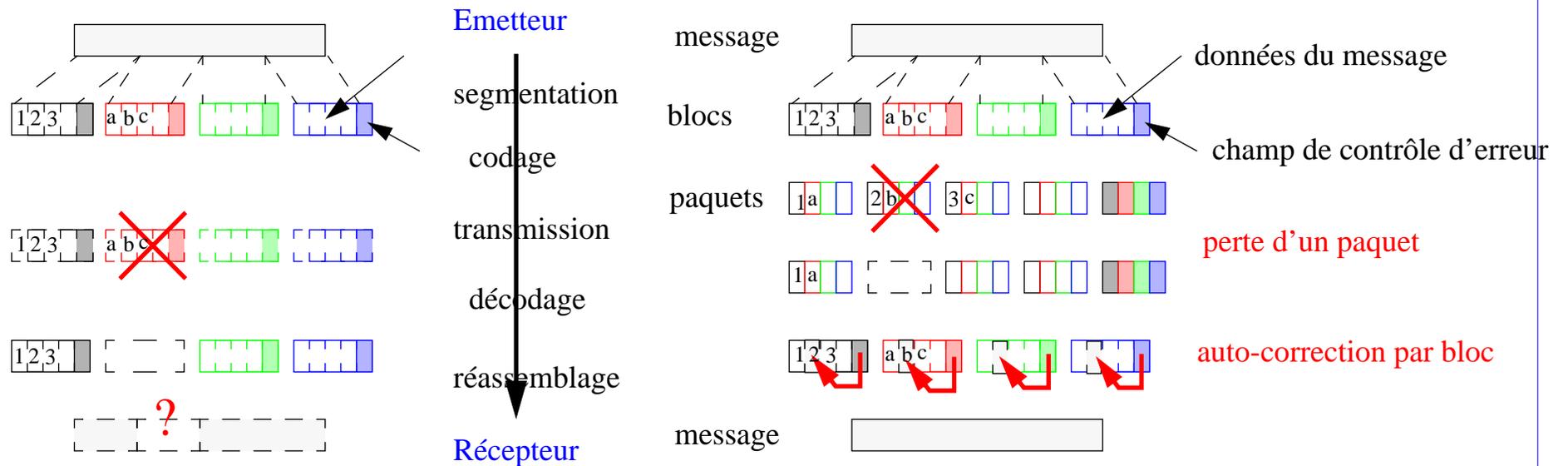
Les codes **BCH** (Bose-Chaudhuri-Hocquenghem) sont ceux qui ont la plus grande capacité de correction d'erreurs indépendantes pour une redondance et une longueur données. Leur rendement n 'est pas très élevé. Ce sont une extension des codes cycliques, ils sont non pas construit sur un alphabet binaire mais un alphabet composé d'un grand ensemble de symboles.

Les codes **RS** (Reed-Solomon) sont des codes correcteurs très puissants. Ils peuvent être présentés comme des codes BCH dans lequel chaque bit des mots du code est remplacé par un entier défini modulo 2^v (avec $n=2^v-1$). La distance d'un code RS(m, n) est égale à $n-m+1$.

7. Autres techniques de correction d'erreurs

7.1. Entrelacement

On peut illustrer le principe qui sous-tend l'entrelacement par une technique simple. L'idée est de répartir les bits consécutifs des blocs formant le message à transmettre dans des paquets différents. Ainsi grâce à cet entrelacement, un groupe d'erreurs (affectant un paquet) affecte uniquement quelques bits dans plusieurs blocs. La technique de protection contre les erreurs appliquée aux blocs est à même de corriger indépendamment ces quelques bits erronés dans chacun de ces blocs.

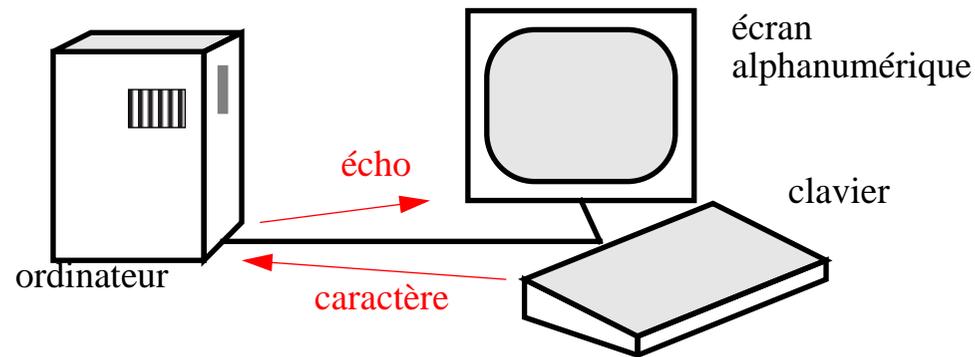


8. Techniques de contrôle d'erreur par retransmission

8.1. Techniques de détection des erreurs

L'écho permet de s'assurer que le récepteur a reçu **correctement** les informations émises.

Exemple : technique utilisée sur les liaisons (asynchrones) entre les ordinateurs et les périphériques par caractères !



Actuellement, peu utilisé car très mauvais rendement et délai important.

8.2. Technique d'autocorrection par répétition

Les informations à transmettre sont répétées plusieurs fois. Au moins un des exemplaires (redundants) sera correctement reçu.

Très mauvais rendement.

8.3. Technique de correction par retransmission

8.3.1 Principe de fonctionnement

- L'émetteur conserve une copie des données qu'il envoie.
- Le récepteur détecte les erreurs grâce à la présence d'un champ de contrôle d'erreur (code polynômial) dans les paquets de données.
- Le récepteur informe l'émetteur de la bonne (resp. mauvaise) réception en lui retournant un paquet spécifique :
 - . **acquiescement** positif (resp. négatif) souvent appelé ACK (resp. NACK)
- Dans le cas d'un acquiescement négatif, l'émetteur doit réémettre le paquet erroné.
- Sinon il peut émettre le prochain paquet.

--> Protocole "Send and wait"

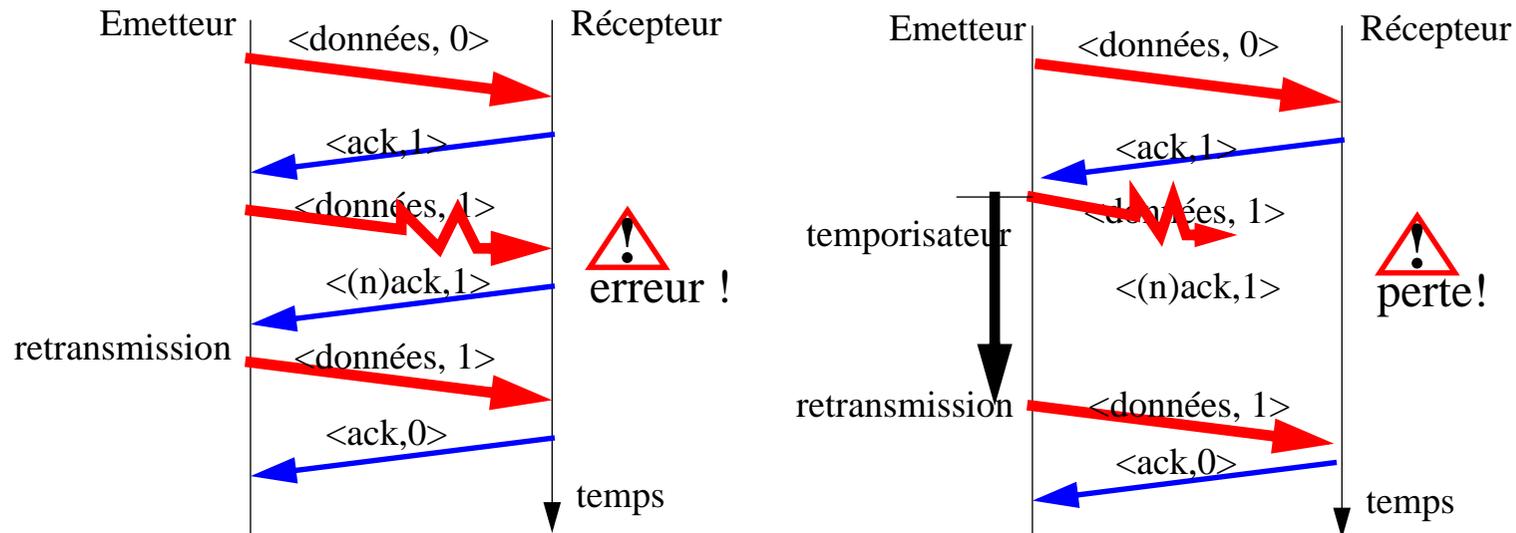
Un **temporisateur** bornant la durée d'attente des acquiescements est nécessaire pour assurer la correction du mécanisme lors des pertes de paquets de données.

L'**identification** des paquets (de données et d'acquiescement) est nécessaire pour assurer la correction du mécanisme lors des pertes d'acquiescement : au moins numérotation modulo 2.

--> Protocole "du bit alterné"

8.3.2 Exemple de fonctionnement

Fonctionne à l'alternat : Emetteur --> Récepteur



Communication bidirectionnelle : x 2 !

Son rendement très faible :

- si le temps de transmission T_t est faible vis-à-vis du temps de propagation T_p .
- car la liaison est inutilisée lorsque l'émetteur attend l'acquittement.
- $R = T_t / (2T_p + T_t)$

--> Optimisation : mécanisme de la fenêtre coulissante (“sliding window”) !

9. Conclusion

Mécanismes de base de protection contre les erreurs :

- détection,
- autocorrection,
- retransmission

--> Amélioration de la fiabilité de la communication, mais (bien qu'avec une faible probabilité) :

- certaines configurations d'erreur ne sont pas détectées !
- certaines configurations d'erreur ne peuvent pas être corrigées !
- certaines configurations d'erreur sont mal corrigées !!!

Ces mécanismes sont présents au sein de nombreuses couches (protocoles), parmi lesquelles on peut citer :

- Liaison de données (ex. Ethernet, Token Ring, FDDI, HDLC, etc)
- Réseau (ex. IP, etc)
- Transport (ex. TP4, TCP, UDP, etc)