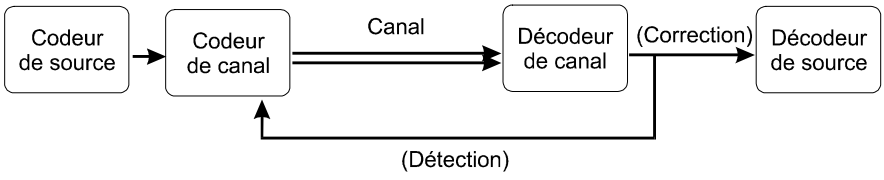
Chapitre 3 :

Codage canal / Gestion des erreurs

Objectifs

- **Détecter** et/ou **corriger** les erreurs de transmission



Codeur de canal à **introduire** une **redondance** utilisable

En pratique

- **Sources d'erreurs :**
  - **le support ;**
  - **le débit ;**
  - **la modulation ;**
  - **le type de codage ;**
  - **le rapport S/N.**

- Taux d'erreur = 10^{-4} à 10^{-7} avec en plus des phénomènes de groupement d'erreurs par paquets.
- Erreur tolérée = 10^{-10} à 10^{-12} dans les réseaux locaux industriels

à détection et correction

Théorème des canaux à perturbation

" Pour une source à débit d'information de R bit/s et un canal de capacité C bit/s, si R<C, il existe un code ayant des mots de longueur n, de sorte que la probabilité d'erreur de décodage p_E soit minimale"

Rq1 : un résultat inatendu !

Rq2 : à p_E constant, n augmente si R tend vers C.

Rq3 : en pratique, si R<0.5 C, des codes existent avec p_E faible.

Quelques Définitions

- **Taux d'erreur** $T_e = \frac{Nombre\ de\ bits\ erronés}{Nombre\ de\ bits\ transmis}$

01100100100110010010100 1010 => 01100110110010110100 0010
~

$T_e = 3/24 = 0.125$

- **Taux de codage / rendement**

$$R = \frac{k}{n}$$

- k taille du mot d 'information (avant codage)
- n taille du mot-code (après codage)

Quelques Définitions

- **Efficacité de la détection :**

$$E = \frac{Nombre\ messages\ reconnus\ erronés}{Nombre\ messages\ erronés}$$

- **Taux d 'erreurs brut :** $t = 1 - (1-p)^n$
- **Taux d 'erreurs résiduel :** $q = t.(1-E)$

Méthodes de détection d'erreurs

- **Détection par écho / répétitions**
- **Détection par codes linéaires**
- **Détection par bit de parité**
- **Détection par codes cycliques**
- **Détection par codes convolutifs**

Méthodes de détection d'erreurs

- **Echo :**
  - **Tout message émis est comparé à son écho et réémis si différent. Problème : il peut y avoir des erreurs dans l'écho, des compensations d'erreurs**, …
- **Codage par répétition**

Codes détecteur et/ou correcteur

3 **Codes linéaires**

- **Parité**
- **Codes groupes**
  Code de Hamming
- **Codes cycliques**
  CRC/FCS, code BCH, RS

3 **Codes convolutifs**

Algorithme de Viterbi

Détection des Erreurs

- **Bit de parité (*checksum*)**
  - **Pour une trame donnée, t = a_1 … a_k ($a_i \in \{0,1\}$), on rajoute un bit a_{k+1} tel que le nombre de 1 dans la trame soit toujours pair.**
  - **On peut aussi transmettre la somme de tous les a_i. Une technique du *checksum* est utilisée dans les entêtes des trames IP et TCP.**

Détection d'erreurs par bit de parité (caractère)

- **VRC (**Vertical Redundancy Check)
  - Asynchrone
- **LRC (**Longitudinal Redundancy Check)
  - Synchrone

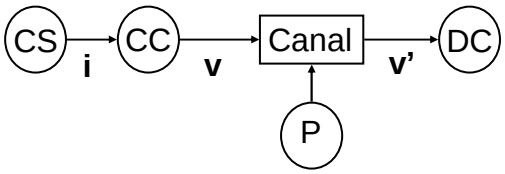
Exercice **à**

	H	E	L	L	O	LRC →
bit 1	0	1	0	0	1	0
bit 2	0	0	0	0	1	1
bit 3	0	1	1	1	1	0
bit 4	1	0	1	1	1	0
bit 5	0	0	0	0	0	0
bit 6	0	0	0	0	0	0
bit 7	1	1	1	1	1	1
VRC ↓	0	1	1	1	1	0

Transmiss

0001001	0	1010001	1	0011001	1	0011001	1	1111100	1	0100001	0
H		E		L		L		O		LRC	

Codes linéaires



- **Notations :**
  - **Mot-code : v**

$$v=[a_1\ a_2\ \ldots.\ a_k\ a_{k+1}\ a_{k+2}\ \ldots.\ a_n]=[i\quad c]$$

[c] : *m* symboles de contrôle
[i] : *k* =*n*-*m* symboles d'information

  - **Mot-erreur : ε**

$$\varepsilon=[\varepsilon_1\ \ \varepsilon_2\ \ldots\ldots\ \varepsilon_n] \qquad v_i=v_i'+\varepsilon \quad\Leftrightarrow\quad v_i'=v_i+\varepsilon$$

$$\varepsilon_i=\begin{bmatrix}1 \text{ si erreur à la ième position}\\ 0 \text{ sinon}\end{bmatrix}$$

Propriétés des codes linéaires

Les symboles de contrôle sont obtenus par une combinaison linéaire des symboles d 'information

- **Code bloc linéaire**

  Symboles binaires et addition modulo 2.

- **Code systématique**

  Les symboles d'information et de contrôle sont séparés.

Codage et matrice génératrice

Soit $G_{(k,n)}$ la matrice génératrice, $[G] = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & & g_{2n} \\ \cdots & & & \cdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}$

$i = [i_1 \quad i_2 \quad \ldots \quad i_k]$

$$v = i.G$$

Décodage et matrice de contrôle

$$v = [a_1 \quad a_2 \quad \ldots \quad a_n]$$

Soit $H_{(m,n)}$ la matrice de contrôle, $[H] = \begin{bmatrix} h_{11} & h_{12} & \ldots & h_{1n} \\ h_{21} & h_{22} & & h_{2n} \\ \ldots & & & \ldots \\ h_{m1} & h_{m2} & \ldots & h_{mn} \end{bmatrix}$

Soit z le syndrome (ou correcteur), $z = H.v'^{T} = \begin{bmatrix} z_1 \\ : \\ z_m \end{bmatrix}$

Si z=[0] pas d 'erreur, sinon erreur et + / - correction

Codage et matrice génératrice

Les matrices H et G sont liées par : $G.H^{t} = 0$

et peuvent se mettrent sous la forme systématique :

$$G = \begin{bmatrix} & : & \\ I_k & : & A_{k,m} \\ & : & \end{bmatrix} \quad H = \begin{bmatrix} & : & \\ A^{t}{}_{k,m} & : & I_m \\ & : & \end{bmatrix}$$

Tableau standard

Bloc données utiles	00	10	01	11
Mots de code	0000	1011	0101	1110
Classe 1	1000	0011	1101	0110
Classe 2	**0100**	1111	**0001**	1010
Classe 3	0010	1001	0111	1100

Classe : ensemble C(Z) = {Z + X, $\forall$ X$\in$ Code}

Représentant de classe : mot de poids le plus faible

Codage et matrice

•*k* vecteurs constituant une base du code *C* sont utilisés pour former les lignes d'une matrice **G** de taille $k \times n$.

•Tout mot de code est une combinaison linéaire des lignes de **G**.

•La correspondance entre G et H n'est pas unique.

•La correspondance entre mots d'information et mots de code n'est pas unique.

•**G** et **G'**, les matrices génératrices de deux codes équivalents sont reliées par :

- des permutation des colonnes
- des combinaisons linéaires sur les lignes.

Exemple k=2, m=1, n=3

$$[G_1]=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix} \quad [H]=\begin{bmatrix}1 & 1 & 1\end{bmatrix} \quad [G_2]=\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}$$

$$\begin{bmatrix}0 & 0\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 0\end{bmatrix} \qquad \begin{bmatrix}0 & 0\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}0 & 0 & 0\end{bmatrix}$$

$$\begin{bmatrix}0 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 1\end{bmatrix} \qquad \begin{bmatrix}0 & 1\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 0\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}0 & 1 & 1\end{bmatrix} \qquad \begin{bmatrix}1 & 0\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 1\end{bmatrix}$$

$$\begin{bmatrix}1 & 1\end{bmatrix}\times\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 1 & 0\end{bmatrix} \qquad \begin{bmatrix}1 & 1\end{bmatrix}\times\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}=\begin{bmatrix}1 & 1 & 0\end{bmatrix}$$

Hamming et correction des erreurs

- **Distance de Hamming**

$$D(v_i,v_j)=(a_{i1}\oplus a_{j1})+(a_{i2}\oplus a_{j2})+....+(a_{in}\oplus a_{jn})$$

˜ Le nombre de coordonnées par lesquels les 2 mots diffèrent

Correction directe :

- C'est le récepteur qui corrige à Il faut donc un code très redondant.
- Ce mécanisme est très coûteux mais aussi très efficace. On parle alors de code auto-correcteur.
- Utilisation : Retrouver le code exact à partir d'un code erroné consiste à retrouver le plus proche voisin au sens de d_H.

Hamming et correction des erreurs

- Exemple : code $d_H = 2$, sur des mots de 4 bits
- Les codes autorisés sont :

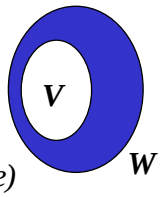
0 0 0 0	**(1)**
0 0 1 1	**(3)**
0 1 0 1	**(1)**
0 1 1 0	**(2)**
1 0 0 1	**(3)**
1 0 1 0	**(3)**
1 1 0 0	**(1)**
1 1 1 1	**(3)**

- Par exemple, soit le message suivant : 0 1 0 0. Ce n'est pas un code valide.
- Pour pouvoir corriger en plus de détecter, il faut un code de d_H > ou égale à 3 pour lequel les seuls mots autorisés sont :

0 0 0 0	**(1)**
1 1 1 1	**(3)**

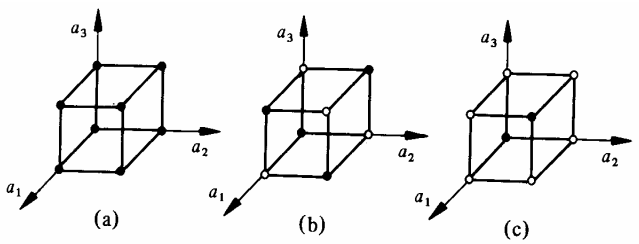
Illustration spatiale

• Un mot = un vecteur dans un espace à n dimensions !
w=[a_1 a_2 ... a_n]



W = ensemble des N = 2^n mots

V = ensemble des S = 2^k mots ayant un sens (mot-code)



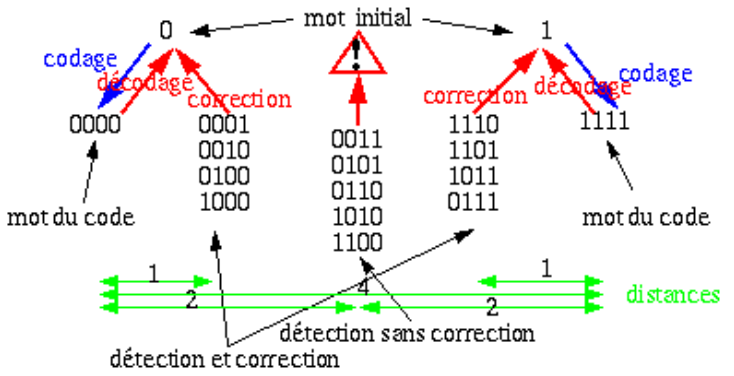
Hamming et correction

$v_i \rightarrow$ Région W_i dépendantes de d_H et disjointes

=> Détection et correction ä si W_i grand

- Théorème de Hamming :
- Si on veut :
  - **détecter p erreurs isolées alors il faut $d_H \geq p+1$.**
  - **corriger q erreurs isolées alors il faut $d_H \geq 2q+1$.**

Hamming et correction des erreurs



Poids de Hamming et distance min.

Le poids de Hamming $w(c)$ d'un mot de code c est égal au nombre de composantes non nulles de c.

Le poids minimal w^* d'un code C est le minimum des poids $w(c)$ des mots de code c non nuls.

La distance minimale d_{min} d'un code linéaire est égale à son poids minimal w^*

Code de Hamming : Principe

- Chaque bit est vérifié par un sous-ensemble distinct des bits de validation
- Une erreur sur un bit provoque une erreur de parité pour chaque bit de validation du sous-ensemble correspondant
- H doit vérifier :
  - Chaque colonne est la représentation binaire des nombres 1 à n.
  - $n = 2^m - 1$ et $k = n - m = 2^m - 1 - m$

Code de Hamming : Principe

- 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.

Hamming et correction des erreurs

- On veut détecter et corriger toutes les erreurs de 1 bits
  - *Dans un mot code de n = m + k bits*
    - *Avec k bits de données*
    - *Avec m bits de validation*
- *Chaque 2^k mot code est à distance d = 1 de n mots invalides*
- *Pour chaque mot valide, on a donc besoin de n+1 mots (n inutilisés + 1 mot code)*
  - *On veut trouver m tel que $(n+1)2^k <= 2^n$, sachant que n = m + k*
  - *Limite théorique : le plus petit m tel que $(m+k+1) <= 2^m$*

Explications

Pour la correction d'une erreur

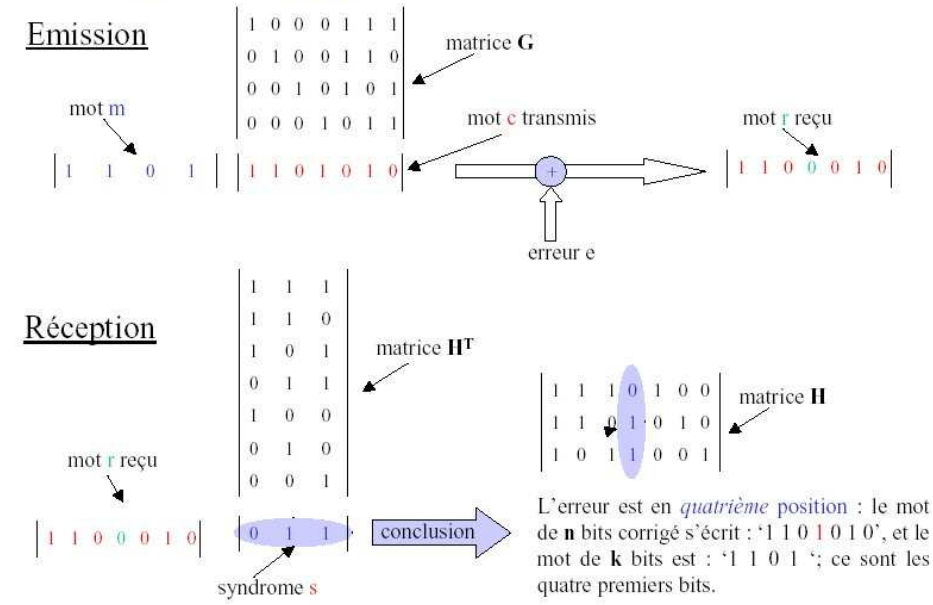
Si on a :

- $[H] = [h_1 \quad h_2 \quad \ldots \quad h_n] = \begin{bmatrix} 0 & 0 & \ldots & \\ : & : & : & \ldots \\ 0 & 1 & 1 & \ldots \\ 1 & 0 & 1 & \ldots \end{bmatrix}$ avec $h_i = bin(i)$
- Mot-erreur : $\varepsilon = [\ldots. \ \alpha_i \ \ldots.]$

$$v'_j = v_j + \varepsilon \Leftrightarrow z = H.v'_j = H.\varepsilon^T \Leftrightarrow z = h_i$$

~ L'erreur est à la position $dec(h_i)$

Exemple : mot m à transmettre '1 1 0 1'



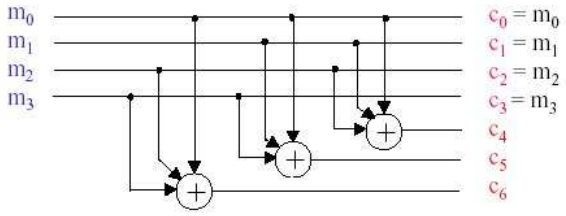
L'erreur est en *quatrième* position : le mot de **n** bits corrigé s'écrit : '1 1 0 1 0 1 0', et le mot de **k** bits est : '1 1 0 1 '; ce sont les quatre premiers bits.

$$\mathbf{H} = \begin{vmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{vmatrix}$$

Valeurs décimales : *7 6 5 3 4 2 1*

La matrice génératrice **G** s'écrit ($\mathbf{GH^T = 0}$) : $\mathbf{G} = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{vmatrix}$

Entrée du codeur — Sortie du codeur



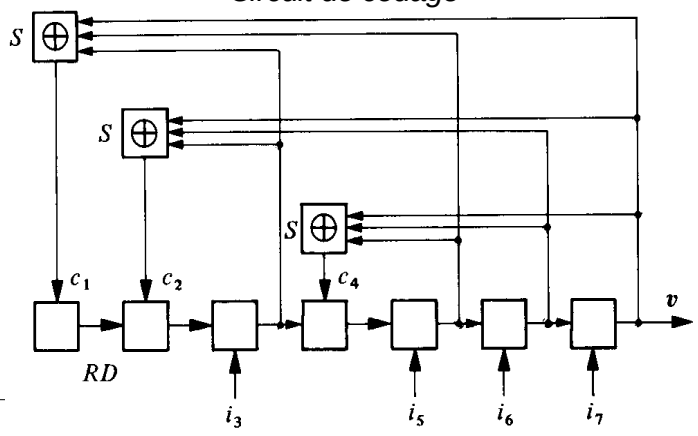
$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \qquad v = [c_1 \quad c_2 \quad i_3 \quad c_4 \quad i_5 \quad i_6 \quad i_7]$$

$$H.v^T = 0$$

ò

$$\begin{cases} c_1 = i_3 + i_5 + i_7 \\ c_2 = i_3 + i_6 + i_7 \\ c_4 = i_5 + i_6 + i_7 \end{cases}$$

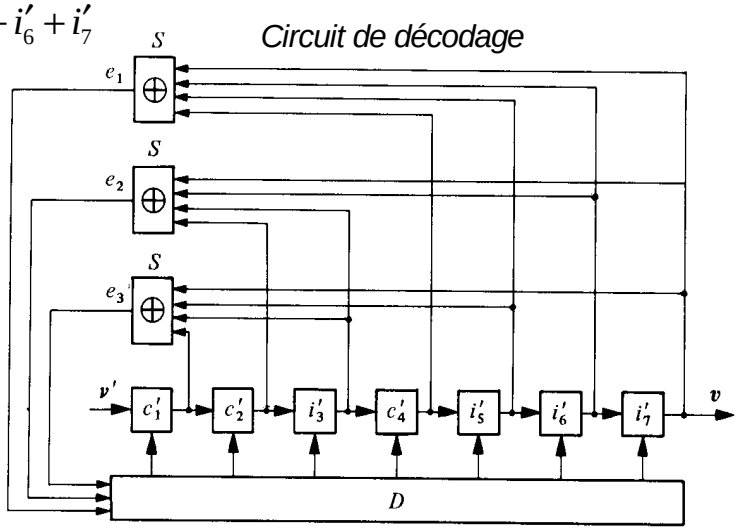
Circuit de codage



$$\begin{cases} e_3 = c'_1 + i'_3 + i'_5 + i'_7 \\ e_2 = c'_2 + i'_3 + i'_6 + i'_7 \\ e_1 = c'_4 + i'_5 + i'_6 + i'_7 \end{cases}$$

$$\varepsilon_i = 1 \text{ pour } i = e_3.2^0 + e_2.2^1 + e_1.2^2$$

Circuit de décodage



Exercice sur le codage de Hamming

Codes polynomiaux

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

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

Codes polynomiaux

• Tout vecteur peut être présenté sous une forme polynômiale

• Les opérations (+, X, /) sont binaires : 1.x + 1.x = 0.x !

- Soit t= $a_0\, a_{n-1} \ldots a_0$. On peut lui associer un polynôme de degré n-1
- par : $P(x) = a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$
- Exemple : le mot 1001101 est associé à $P(x) = x^6 + x^3 + x^2 + 1$

Codage / Décodage

- **Division / Multiplication (en binaires)**

$a(x) = x^3 + x^2 + x$ et $b(x) = x^3 + x + 1 \rightarrow \quad c(x) = x^6 + x^5 + x$

$c(x) = a(x) \times b(x)$ $\quad [1110] \times [1011] = [1100010]$

- **Codage par division**

$v(x) = c(x) + x^m.i(x)$

Systématique !

$$c(x) = Reste\left(\frac{x^m.i(x)}{g(x)}\right)$$

- **Décodage par division**

Si *z(x)=0* => Transmission OK
Sinon => Détection ou correction

$$z(x) = Reste\left(\frac{v'(x)}{g(x)}\right)$$

Codage / décodage

- Soit une clef G(x), polynôme de degré v. On pose :
- $x^v P(x) = Q(x) G(x) + R(x)$
- où Q(x) et R(x) sont deux polynômes de degré au plus égal à, respectivement, n-1 pour Q et v pour R.
- On travaille en binaire **donc** $x^v P(x) + R(x) = Q(x) G(x) = Y(x)$

Codage / décodage

1. l'émetteur transmet les mots associés aux polynômes P(x) et R(x) (généralement par simple concaténation).
2. Le décodeur peut fabriquer Y(x) et la division puisque G est normalisé (donc v et G sont connus).
3. Si le reste est non nul (*i.e.* Y(x) ≠ G(x) Q(x)) alors il y a au moins une erreur.

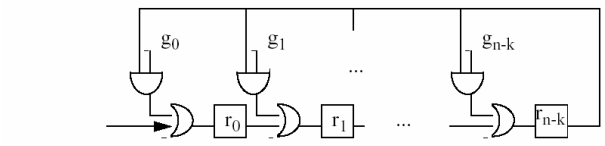
- On utilise pas n'importe quel polynôme générateur. Il est le plus souvent normalisé pour un protocole donné.
- Pour v = 16, le CCITT préconise dans l'avis V24, le polynôme :

$$G(x) = x^{16} + x^{12} + x^5 + 1$$

- L'implémentation Hardware de cet algorithme est facile.

Diviseur électronique

$$g(x) = g_0+g_1.x+g_2.x^2+...+g_{n-k}.x^{n-k}$$



- **La multiplication est réalisée par un *ET logique*, l'addition par un *OU exclusif*, plus des registres à décalage.**
- **Procédé :**
  - **(i) les registres ri 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 ri contiennent alors le reste, qu'on extrait (n-k étapes).**
- **De nombreuses optimisations sont possibles :**
  - **- Lorsque gi=0 on supprime simplement la connexion et la porte ET !**
  - **- phase spécifique d'initialisation, etc.**

Codes cycliques

Code cyclique = linéaire + propriété de permutation circulaire

Définition :

- toute permutation de tout mot code donne un autre mot code

- les polynômes associés aux mots codes sont tous multiples d'un générateur g(x) diviseur de $(1+x^n)$

- 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}.

Codes cycliques

- g(x) définit le codeur (n,k)
- g(x) est de degré m=n-k
- Il vérifie : $1+x^n = g(x)\times p(x)$

$$g(x) = 1 + g_1.x + g_2.x^2 + \ldots + g_{n-k}.x^{n-k}$$

Exemple : code cyclique (n=7, k=4) :

$$1+x^7 = (1+x)\times(1+x^2+x^3)\times(1+x+x^3)$$

g(x) est de degré 3 soit :

$$g(x) = (1+x^2+x^3) \quad \text{ou} \quad g(x) = (1+x+x^3)$$

Bilan sur les codes cycliques

- La théorie des codes cycliques permet de montrer que ce type de code permet de détecter jusqu 'à k erreurs pour G(x) de degré k
- Pour CRC 16 :

  1) toutes les erreurs simples, doubles et triples

  2) toutes les salves d'erreurs de longueur impaire ou de moins de 17 bits

  3) 99,998% des salves d erreurs de plus de 16 bits.
- On peut ainsi obtenir un taux d'erreur résiduel de 10^{-10}.

Code cycliques BCH et RS

- 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 BCH (Bose-Chaudhuri-Hocquenghem) sont ceux qui ont la plus grande capacité de correction d'erreurs indépendantes pour une redondance et une longueur données.
- 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.

Exemple de polynômes générateurs

ATM

$x^8 + x^2 + x + 1$ => Cellule ATM

$x^{10} + x^9 + x^5 + x^4 + x + 1$ => Couche AAL type 3/4

CCITT N°41 => X25 (HDLC)

$x^{16} + x^{12} + x^5 + 1$

IEEE 802 => Réseaux locaux

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$

k peut varier de 200 à 3000 bits

Codes continus / convolutifs

- **Généralités**

=> Les symboles d'information sont traités en flux continu

Rque :Blocs de n_0 symboles, mais dont les m_0 contrôleurs ne dépendent pas que des k_0 symboles d'information !

Taux d'émission ou rendement : $R = \frac{k_0}{n_0}$

Codes convolutifs systématiques

Mot-code : $V = \lfloor X_1 Y_1 X_2 Y_2 ..... X_j Y_j ..... _1 \rfloor$

avec $X_j = \lfloor X_j^1 ..... X_j^{k_0} \rfloor$ Information

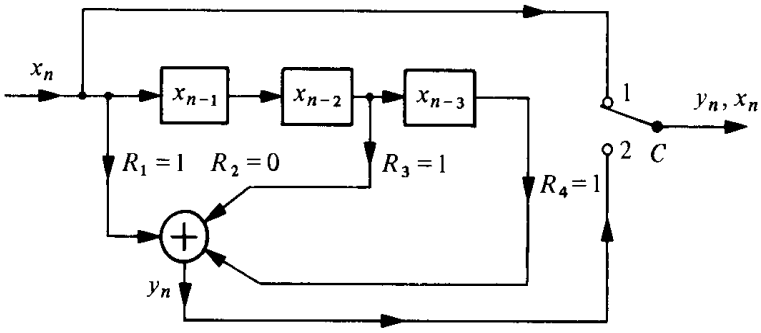
$Y_j = \lfloor Y_j^1 ..... Y_j^{m_0} \rfloor$ Contrôle

- **Codes convolutifs non systématiques**
  - Contrôle et information sont mélangés

Mot-code : $V = \lfloor U_1 U_2 ..... U_j ..... \rfloor$

Exemple : m=4, k_0=1, m_0=1, n_0=2

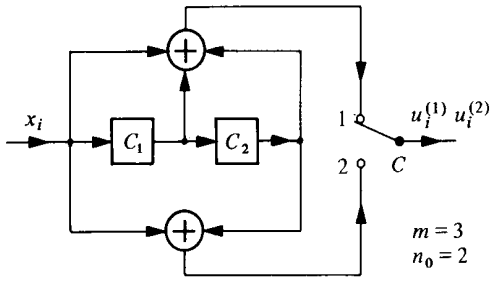
$$y_n = R_4.x_{n-3} + R_3.x_{n-2} + R_2.x_{n-1} + R_1.x_n$$



- R=[1011]

Exemple : n_0=2, R=0.5 , m=3

$$U_n^{(1)} = x_n + x_{n-1} + x_{n-2}$$
$$U_n^{(2)} = x_n + x_{n-2}$$

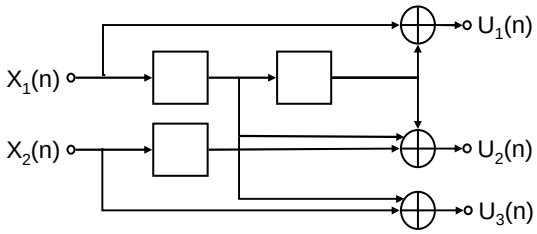


t_i	1	2	3	4	5	6	7	8
x_i	1	1	1	0	1	0	0	0
$C_1 C_2$	00	10	11	11	01	10	01	00
$u_i^{(1)} u_i^{(2)}$	11	01	10	01	00	10	11	00

49

Représentation des codes convolutifs

- Par le codeur

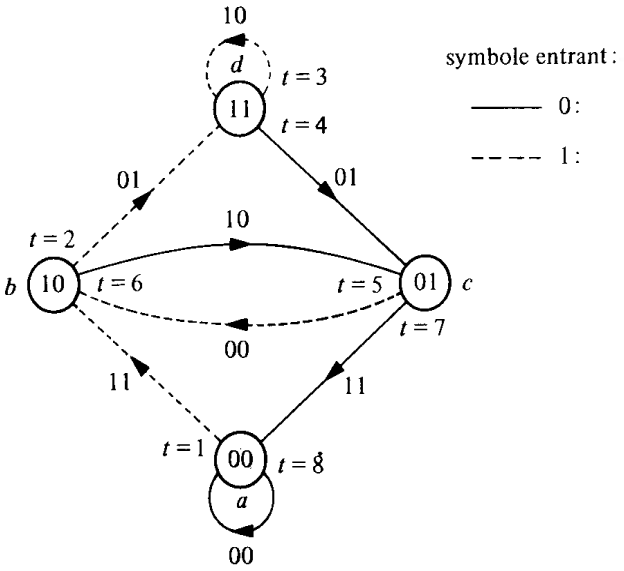


- Par une matrice de transfert

$$G_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \qquad G_2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \qquad G_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$G = \begin{bmatrix} 5 & 3 & 2 \\ 0 & 2 & 4 \end{bmatrix}$$

- Un diagramme d'état
- Un treillis à chemin à décodage par chemin le + probable

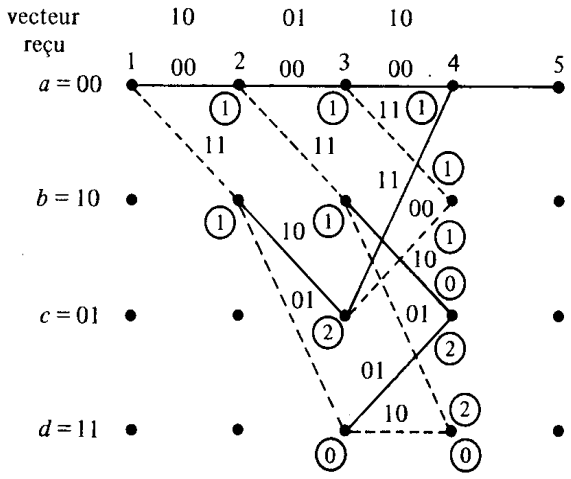


Décodage : algorithme de Viterbi

Stratégie de recherche de D_{min}

10 01 10 => $Min\left(\sum_{i=1}^{3} d_i\right) = ?$

=> 11 01 10



Conclusion sur le codage de canal

Indispensable

Théories mathématiques complexes $\prod$ des solutions concrètes

- Reed-Salomon (1984) : BCH
- Turbo-Codes (1993) : Code convolutif

Recherche de codeurs conjoint source / canal

- complexité --
- robustesse ++
- flexibilité ++

Conclusion sur le codage de canal

- **Bilan sur la correction des Erreurs** :
- Pour les codes cycliques (de taille de clef = v) on peut détecter les paquets d'erreurs de taille ≤ v ou corriger les paquets de taille ≤ .
- La performance de ces codes est cependant contrebalancée par un coût important en longueur de code ce qui explique qu'ils sont très peu utilisés dans les réseaux locaux industriels.

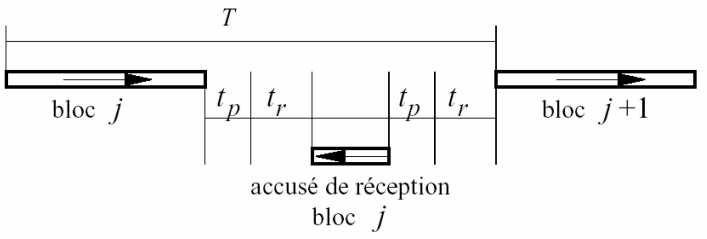
==> Correction par retransmission

Correction par retransmission

- Quand ?
  - Correction de l ’erreurs impossible
- Types de retransmission :
  - send and wait
  - envoi avec arrêt et attente d ’ACK ou NACK ou echo ou time-out
  - Nécessité d ’identification des messages

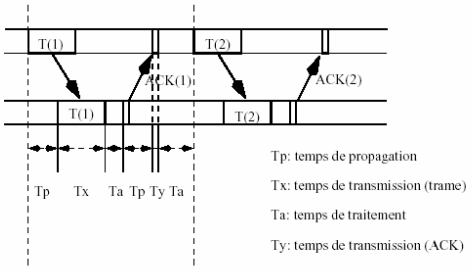
Correction par retransmission

$T_{trans} = (N_k + N_m + N_{ack})/D + 2.(t_p + t_r)$



- Rendement très faible

Correction par retransmission



Utilisation: $U = \frac{Tx}{Tt}$ $Tt = Tx + Ty + 2Tp + 2Ta$

Exemple: Ligne satellite à 56kbps et trames de 1000 bits. Le satellite est stationné à 33.000km sur la terre et la vitesse de propagation $3 \cdot 10^8$ m/sec. L'utilisation $U = \frac{T_x}{T_t}$ est de 3.4% seulement en utilisant un protocole 'envoyer et attendre' pour envoyer la trame et renvoyer l'acquittement.

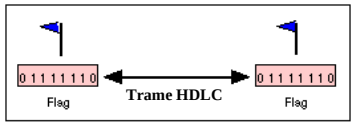
Correction par retransmission

- Transmission continue : pas d 'attente de l 'ACK
  - renvoi total a partir de l 'erreur
  - fenêtre d 'anticipation
  - Rendement moyen
- Retransmission sélective :
  - Mise en œuvre complexe
  - bon rendement
  - Satellites, TCP

==> Voir cours de Réseaux

Gestion de la communication : Norme HDLC

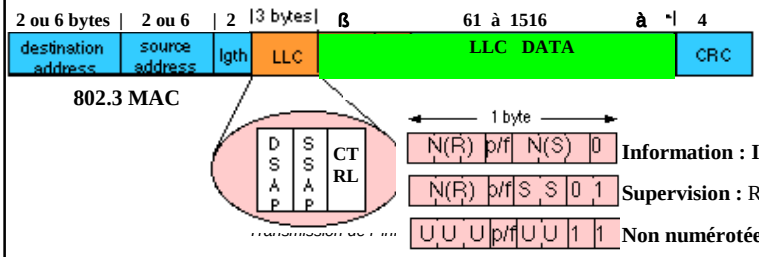
- La norme HDLC (High level Data Link Control) fournit un service de transmission synchrone transparent de niveau 2.
- Service Sans ou Avec connexion :
- **Délimiteurs de trames**



- **La trame HDLC :**

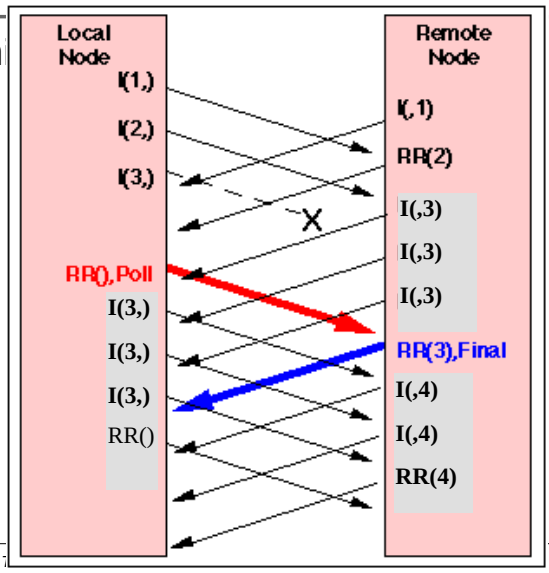
N(S) : Numéro de trame émise

N(R) : Numéro de trame attendue



Norme HDLC

- A an



60

Norme HDLC

- **Exemples de fctnt (rejet, …)**
- **Fenêtre d'anticipation:** Nombre maximum de trames pouvant être envoyées sans attendre d'ACK.
- **Notion de Timer :** délais préétabli au bout duquel on réémet la trame