REPUBLIQUE DU CAMEROUN
Paix - Travail – Patrie

UNIVERSITE DE YAOUNDE I

ECOLE NATIONALE SUPERIEURE POLYTECHNIQUE

REPUBLIC OF CAMEROUN
Peace - Work – Fatherland

UNIVERSITY OF YAOUNDE I

NATIONAL ADVANCED SCHOOL OF ENGENEERING

MASTER PRO 2 EN TELECOMMUNICATIONS

TELEVISION NUMERIQUE

Séquence 2 : CODAGE DE SOURCE ET DE CANAL

Equipe des concepteurs :

- Janvier FOTSING
- Pierre TSAFACK

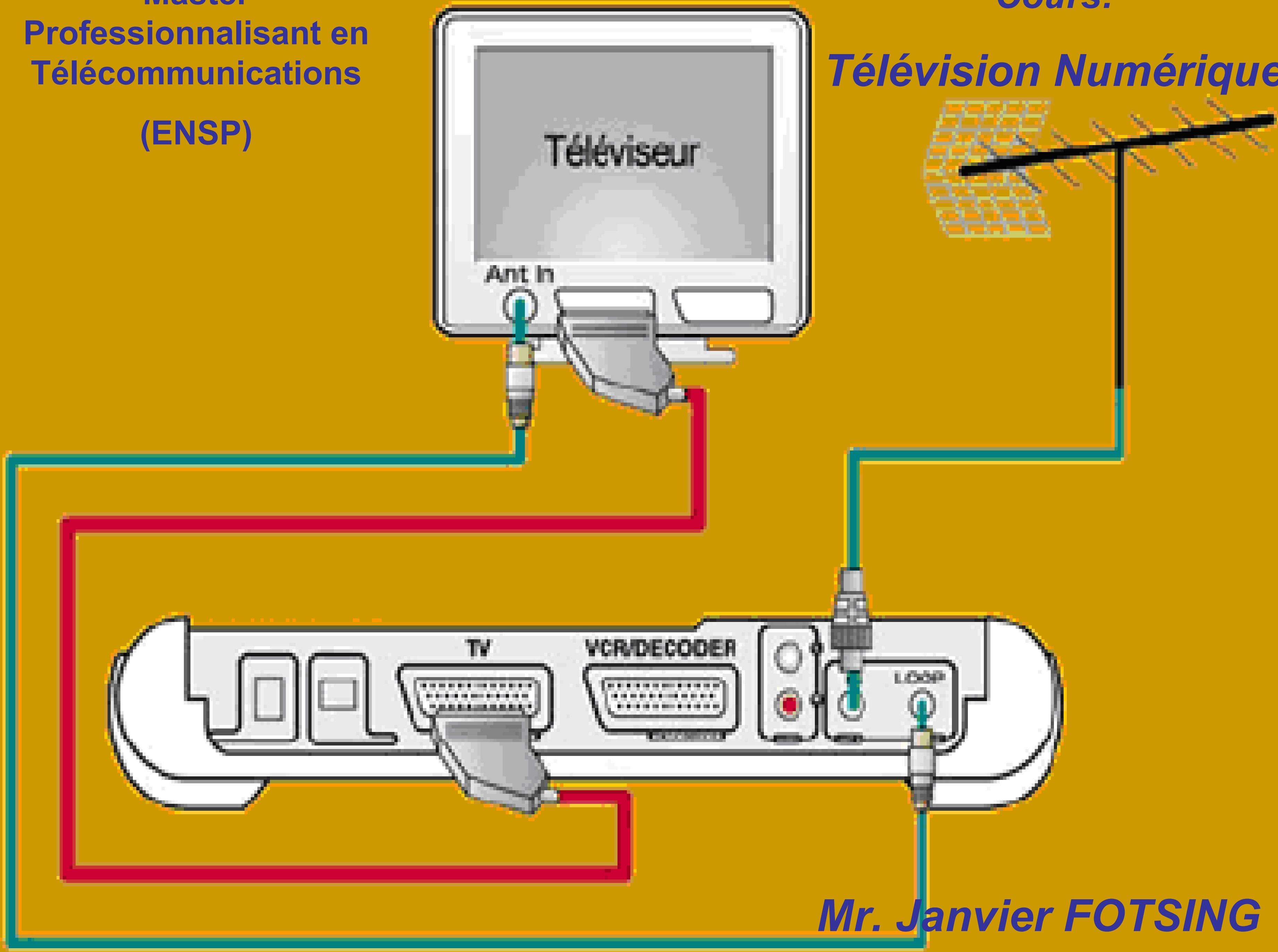
Le contenu est placé sous licence /creative commons/ de niveau 5 (Paternité, Pas d'utilisation commerciale, Partage des conditions initiales à l'identique)..

Master Professionnalisant en Télécommunications

(ENSP)

Cours:

Télévision Numérique



Mr. Janvier FOTSING

Cours: TVN 224

Codage de source et de canal

Master Professionnalisant en Télécommunication

Plan du cours

- Rappel,entropies et schéma général de communication
- Introduction au codage source/canal
- Types de codages
- Codage canal
- Codage source
- Détection et correction des erreurs

Rappel

•*Entropie et information*

- Information liée à un évenement de probabilité p(k)

$$\log\frac{1}{p(k)}$$

  - d'autant plus d'information que l'évenement est inattendu
  - croissance exponentielle du nombre de combinaisons avec la longueur du message

- Entropie = espérance de l'information, c.à.d la moyenne au sens probabiliste.

$$H=\sum_k p(k)\log\frac{1}{p(k)}=-\sum_k p(k)\log p(k)$$

Rappel

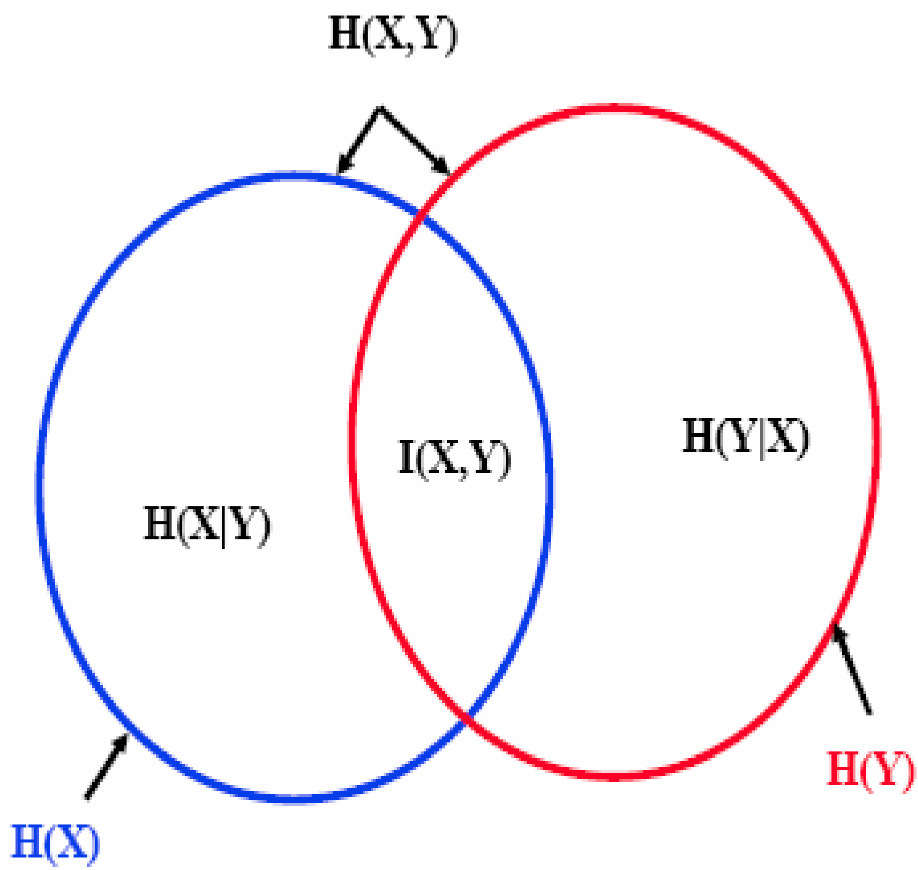
• ***Entropie et information***

$$H(X) = -\sum_x p(x) \log p(x)$$

$$H(X,Y) = -\sum_x \sum_y p(x,y) \log p(x,y)$$

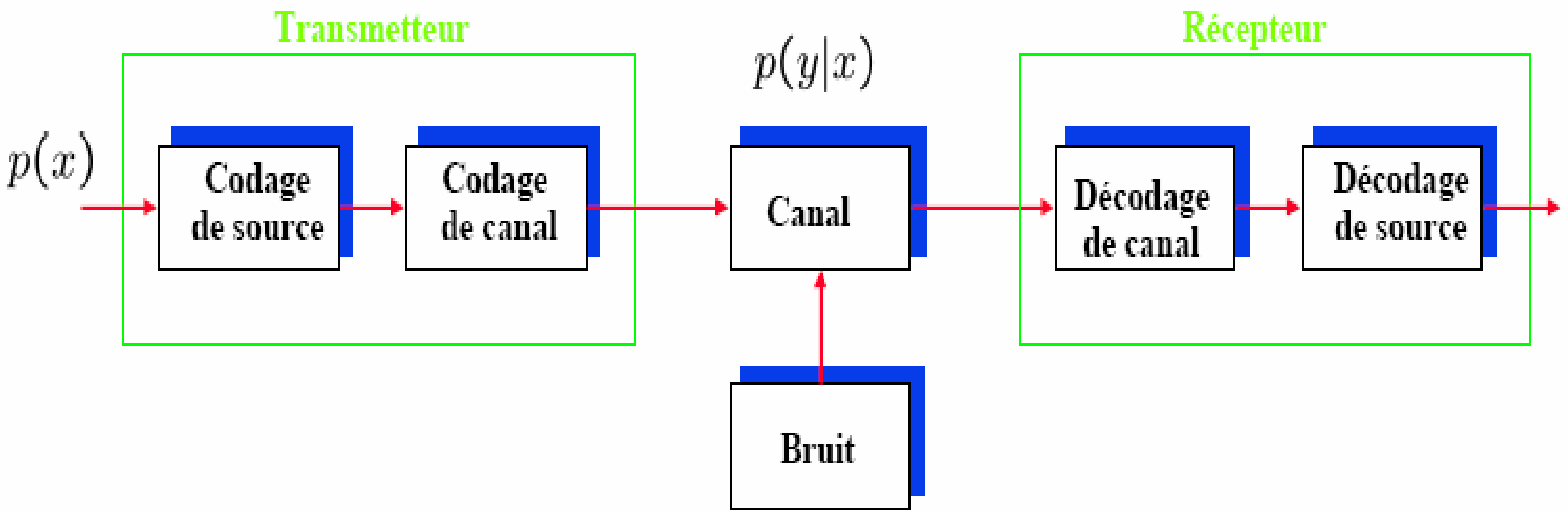
$$H(Y|X) = -\sum_x \sum_y p(x,y) \log p(y|x)$$

$$\begin{aligned} I(X,Y) &= \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= H(Y) - H(Y|X) \\ &= H(X) - H(Y|X) \end{aligned}$$



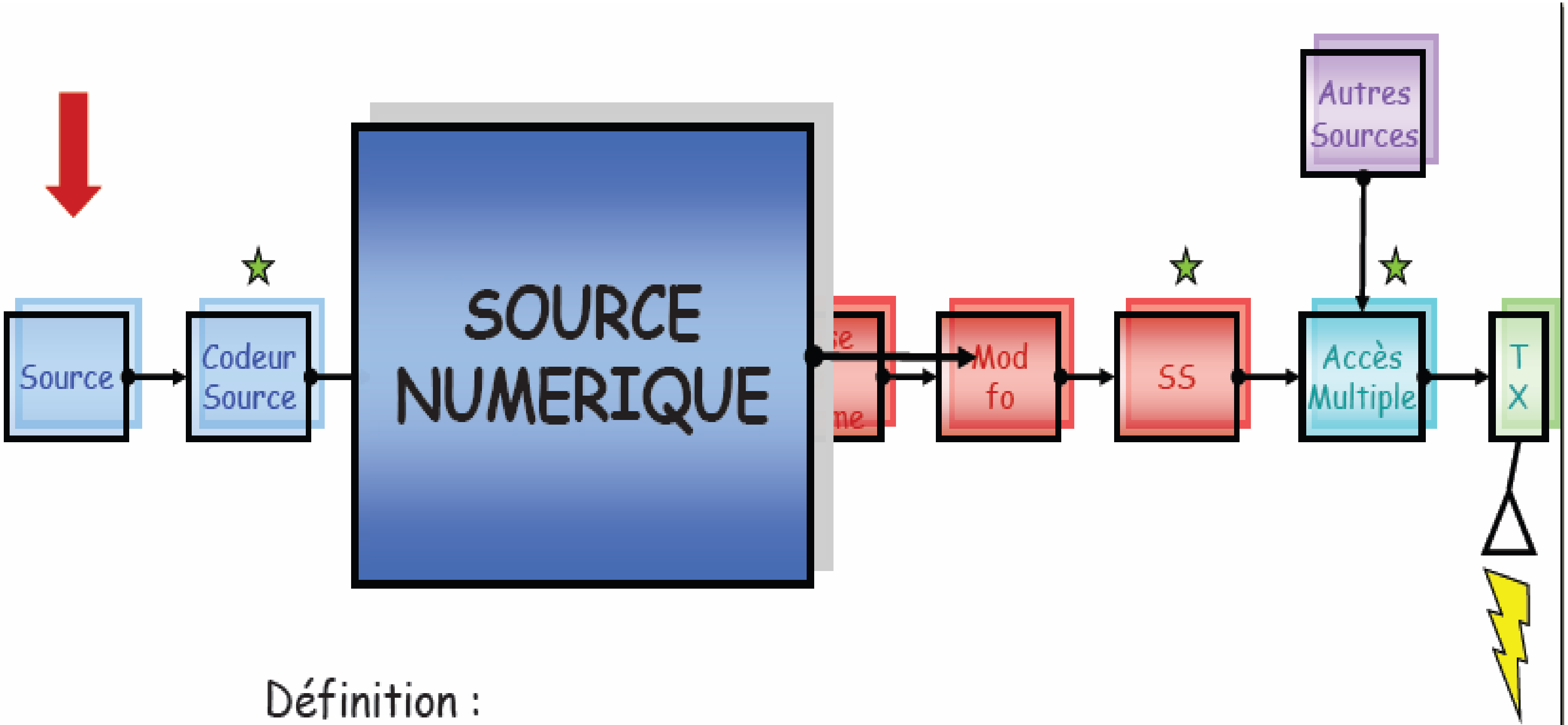
Rappel

Schéma de communication



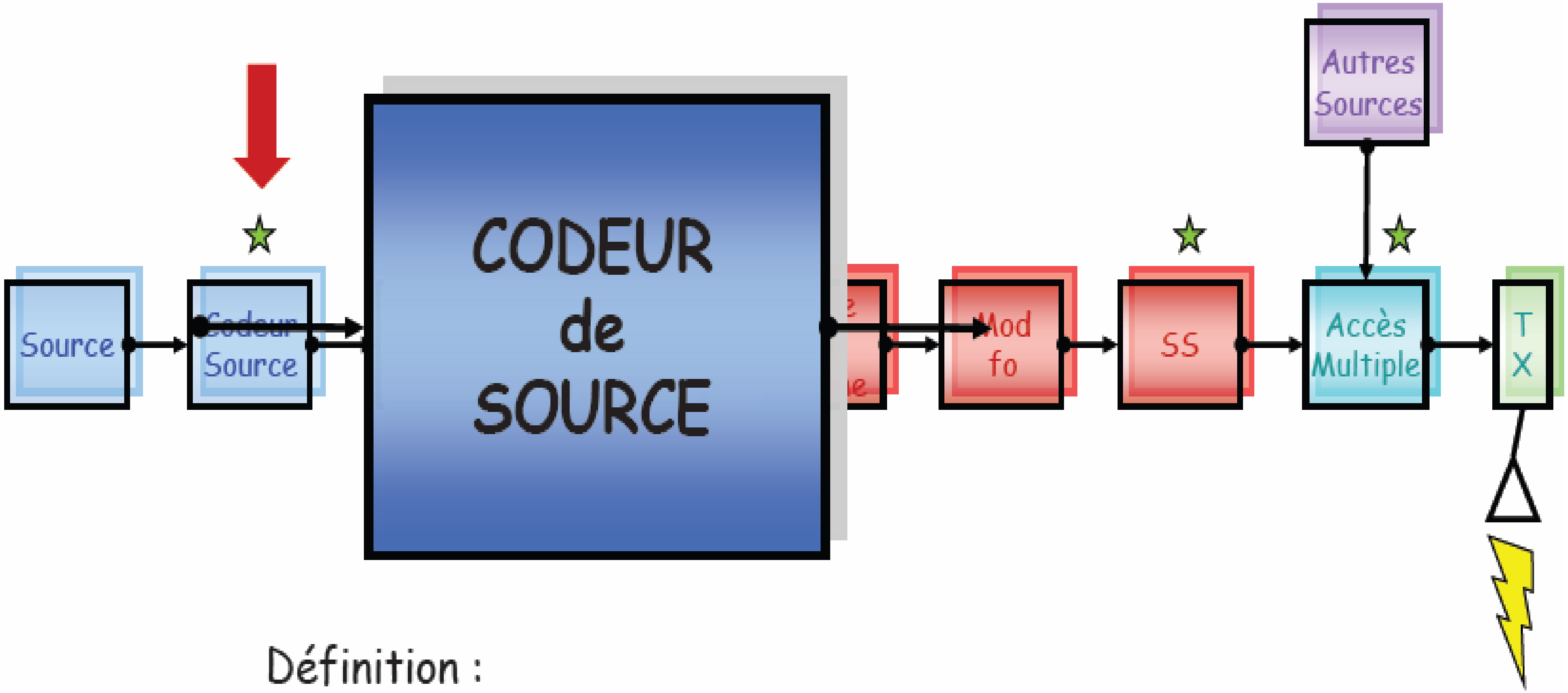
- Codage de source pour profiter de la redondance de la source et réduire la longueur du message à transmettre

- Codage de canal pour détecter et corriger les erreurs lors de la transmission sur le canal



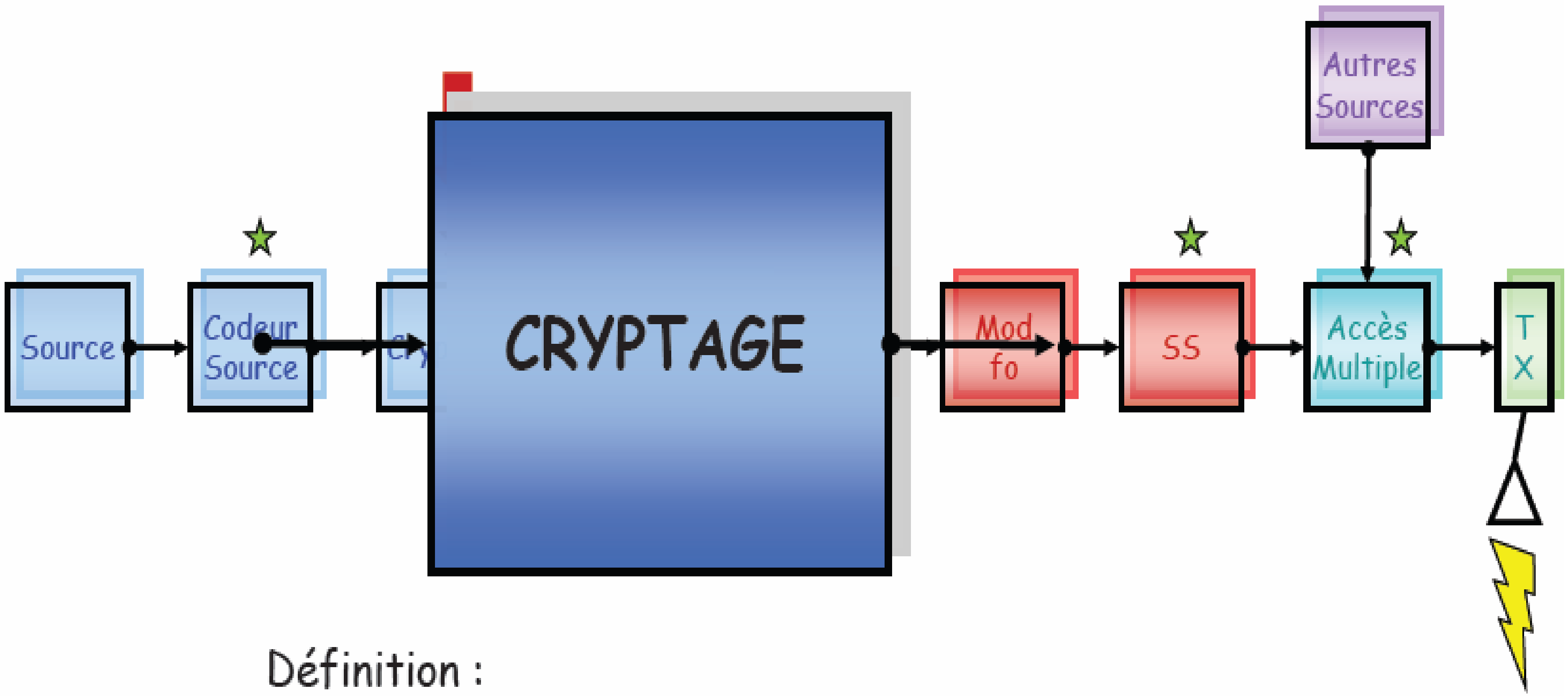
Définition :
Tout dispositif capable de générer des messages appartenant à un alphabet discret et fini

Exemples :
- un ordinateur ;
- un convertisseur A/N ;
- une base de données.



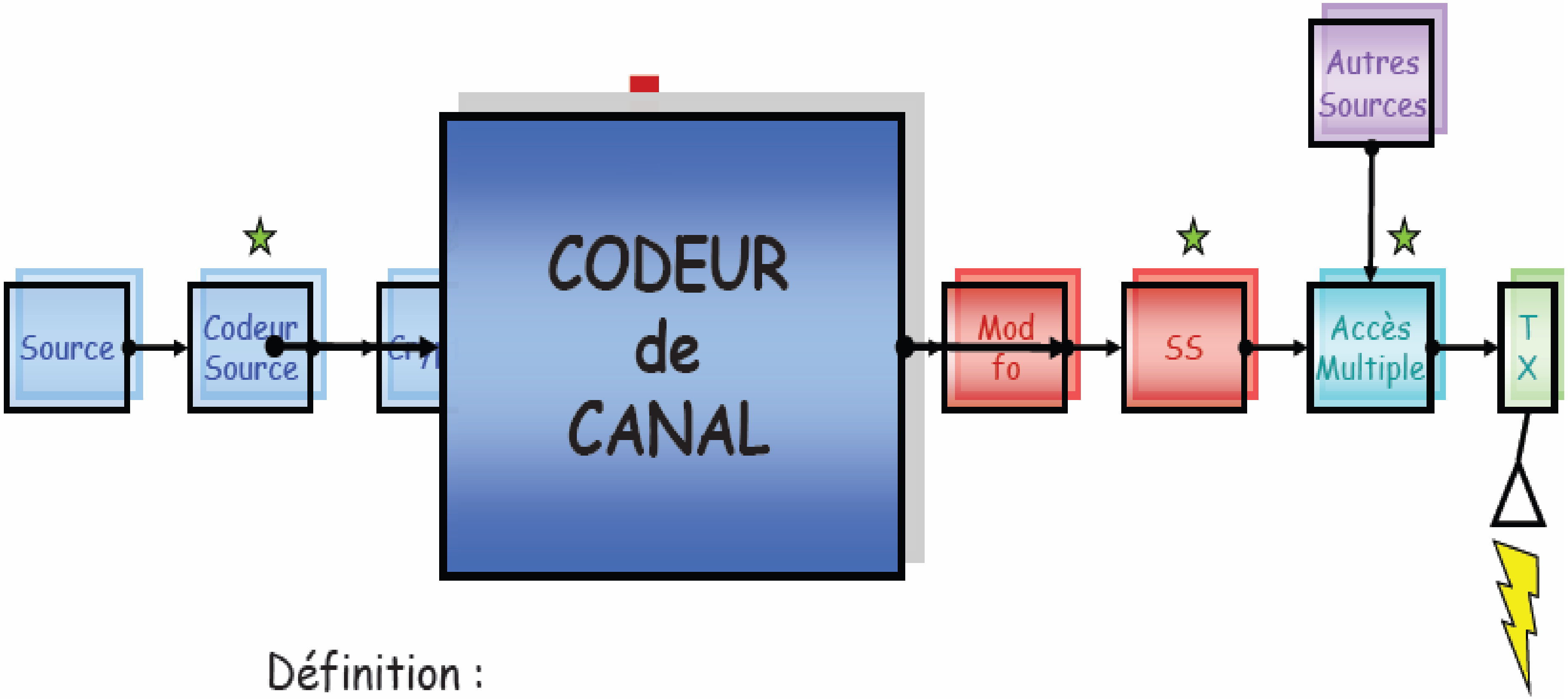
Définition :
dispositif capable de générer une représentation plus « économique » de l'information générée par la source

Exemples :
- codeur de parole ;
- codeur d'images ;
- PKZIP, gzip.



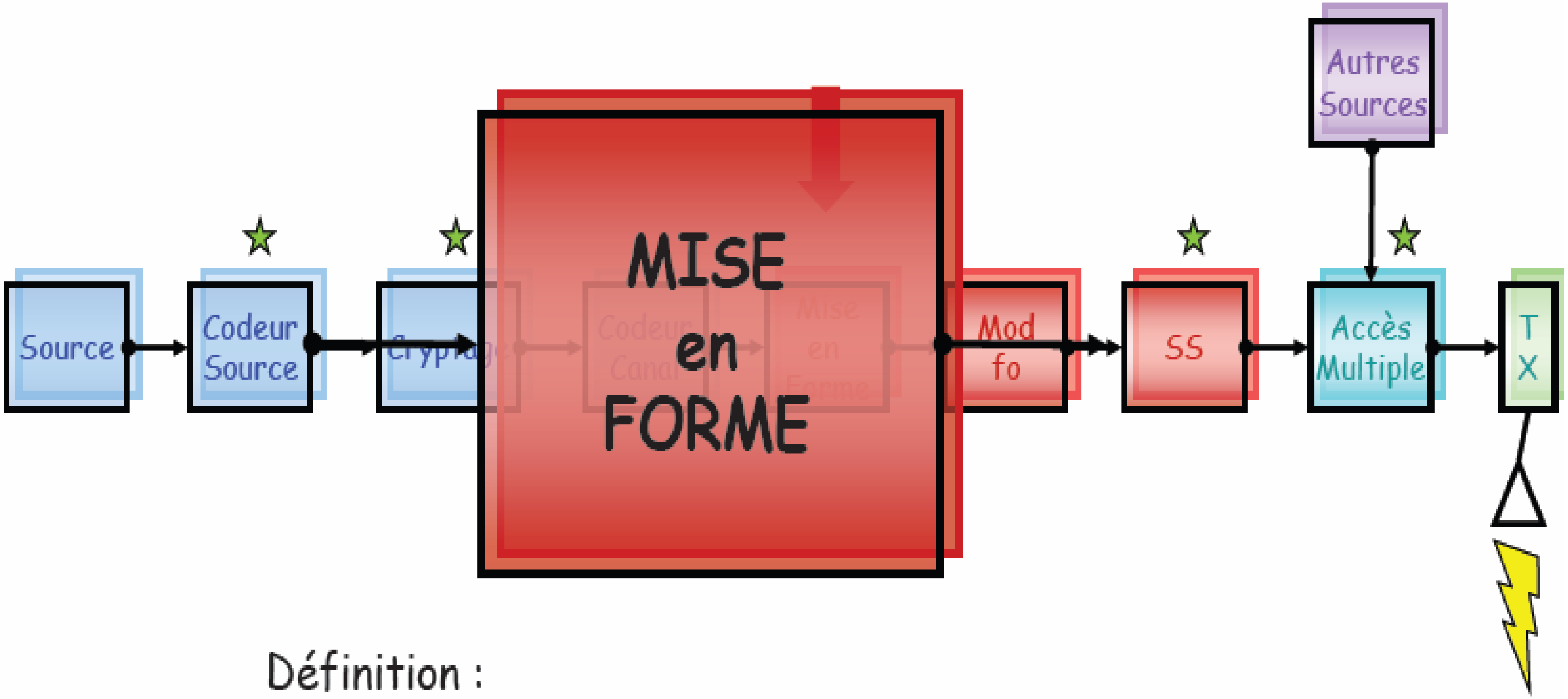
Définition :
dispositif capable « protéger » l'information.

Exemples :
- Clés de chiffrement ;
- RSA ;
- Watermarking.



Définition :
dispositif capable de protéger l'information des perturbations du canal

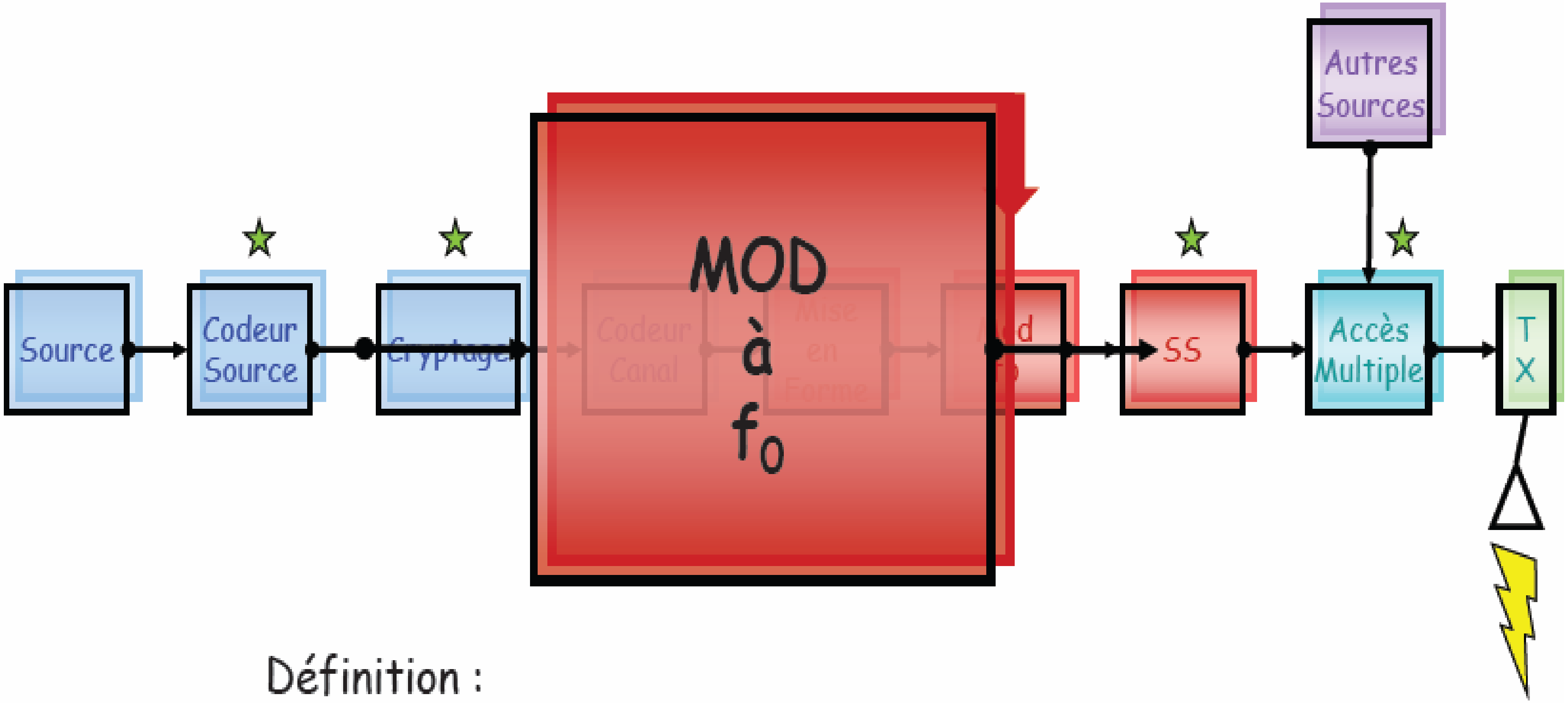
Exemples :
- codes détecteurs d'erreurs ;
- codes correcteurs d'erreurs ;
- codes en bloc;
- codes convolutifs.



Définition :
dispositif qui représente l'information sous forme d'un signal apte à être transmis sur un support physique (signal à temps continu)

Exemples :

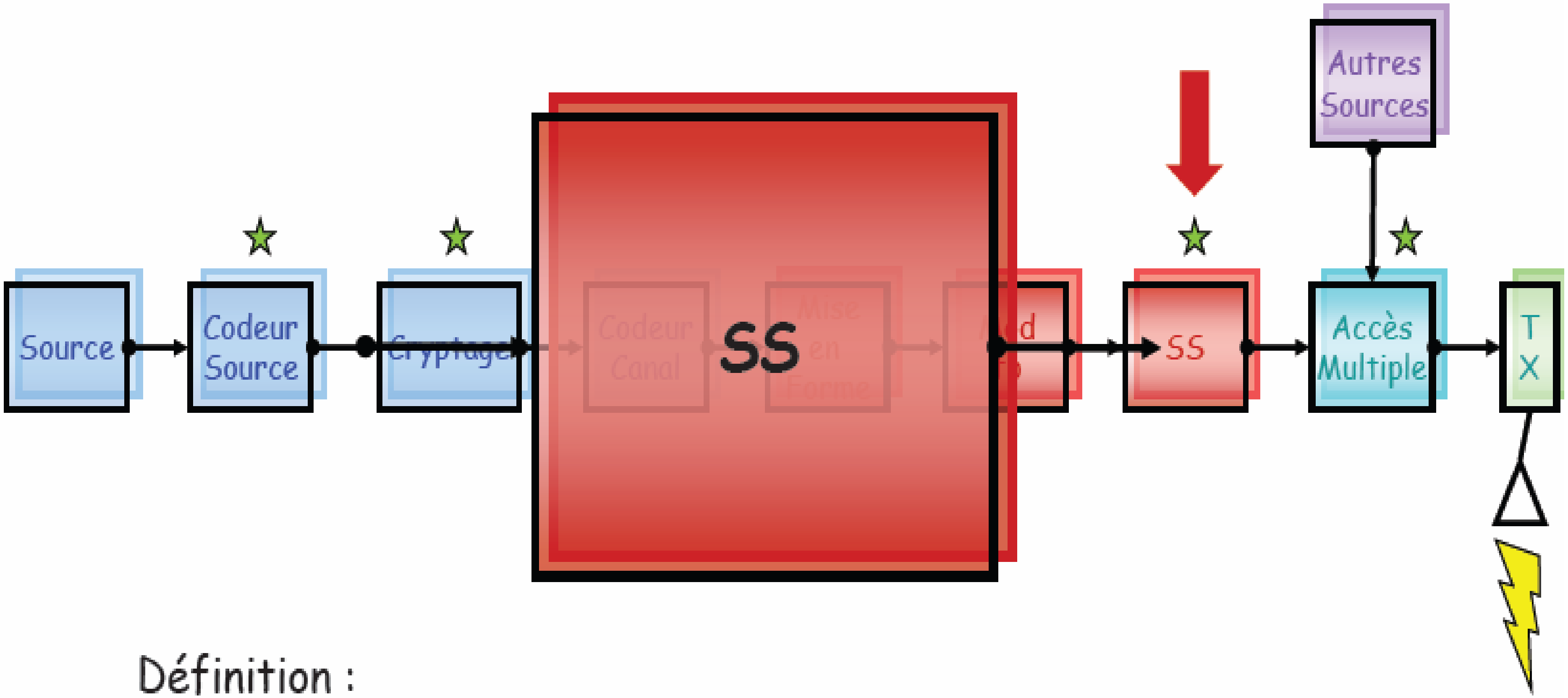
- code en ligne ;
- filtre de Nyquist ;
- filtre en racine de Nyquist;
- filtre en cosinus surélevé.



Définition :
Convertisseur de fréquence ; transpose le spectre du signal autour de la fréquence f_0.

Exemples :
- modulateur numérique ;
- modulations linéaires ;
- modulations à phase continue ;

- BPSK, QPSK, QAM, MSK.

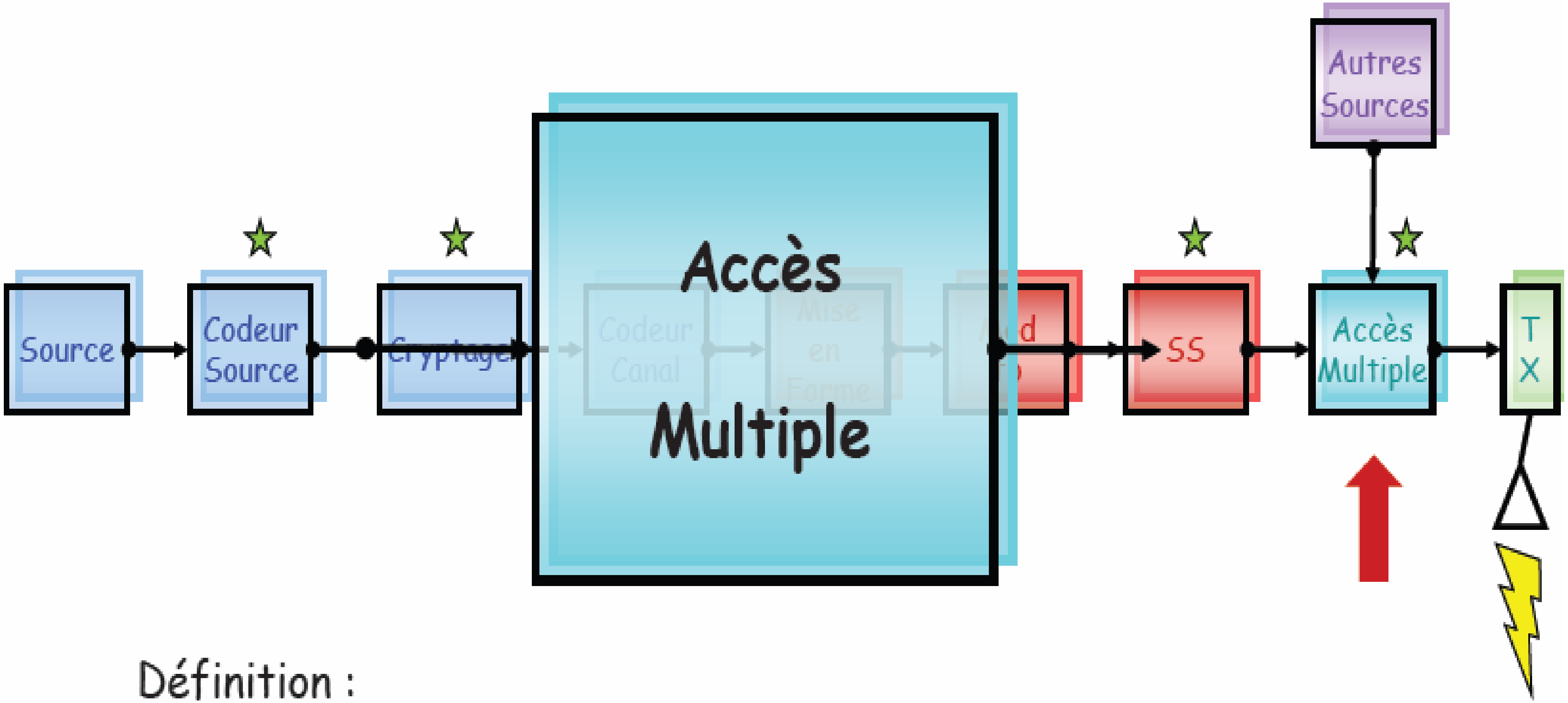


Définition :

Dispositif à étalement du spectre(*spread spectrum*), transforme un signal de largeur de bande B en signal de largeur de bande B_{SS} avec $B_{SS} \gg B$.

Exemples :

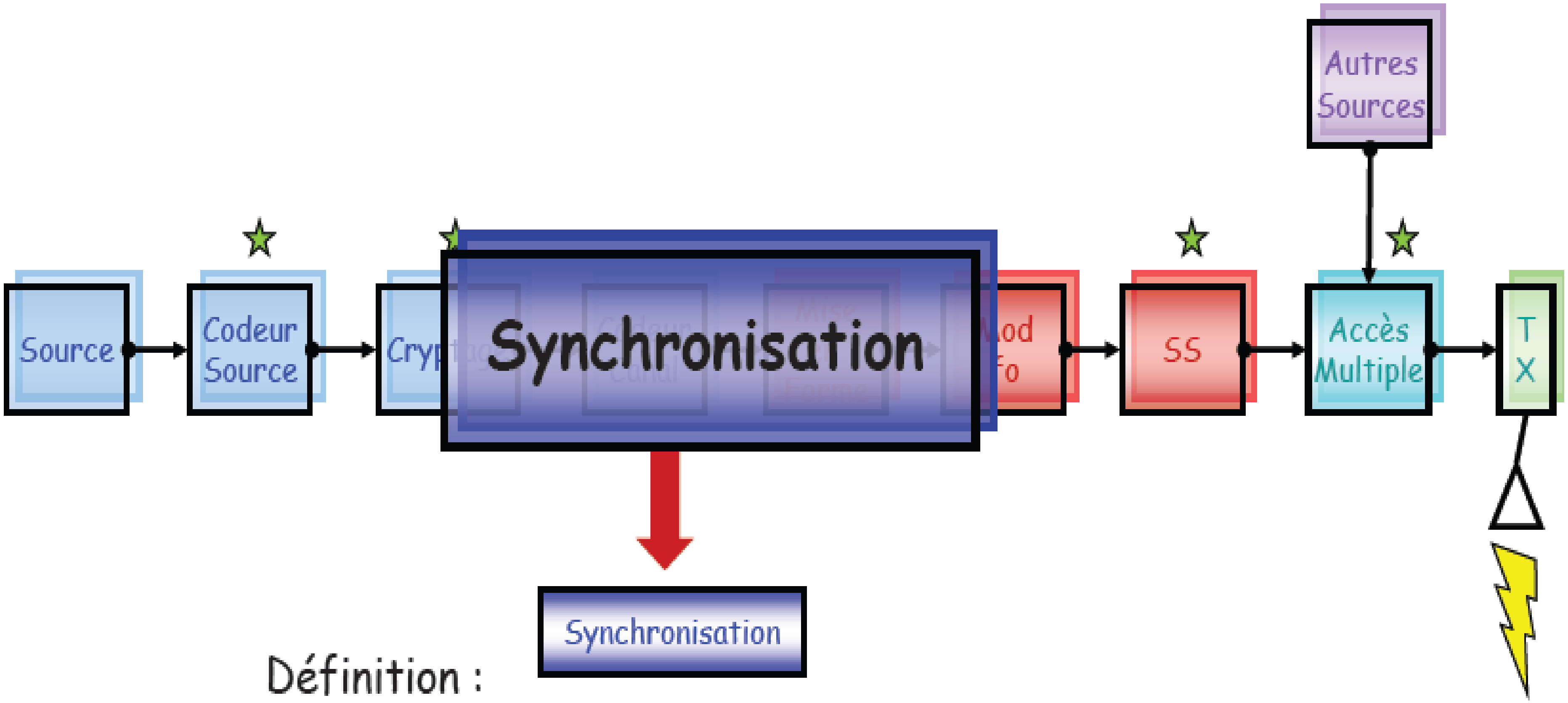
- étalement par séquence directe ;
- étalement par saut de fréquence ;
- système anti-brouillage ;
- CDMA.



Définition :
Gestionnaire d'accès à la ressource de communication ; partage de La ressource avec d'autres éventuelles source d'information.

Exemples :

- FDMA, TDMA ;
- ALOHA, p-ALOHA, S-ALOHA ;
- CSMA, CSMA-CD, CSMA-CA ;
- CDMA.



Définition :

Dispositif capable de générer des références temporelles et fréquentielles compatibles avec la nature « temps discret » du signal numérique.

Exemples :

- synchro bit ;
- synchro fréquence ;
- synchro trame.

Types de codage

- *Rappels*

Code binaire= correspondance entre un ensemble d'informations élémentaires (l'alphabet) et un ensemble de configurations binaires (les mots de code, de longueur fixe pour la plupart des codes employés).

Le CCITT (Comité Consultatif International de Télégraphie et Téléphonie, actuel UIT-T) a normalisé plusieurs codes. Parmi lesquels de code CCITT n°2 et le code CCITT n°5.

Code CCITT n° 5 :

$b_6b_5b_4$ → $b_3b_2b_1b_0$ ↓	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	à (1)	P	\ (1)	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	# (1)	3	C	S	c	s
0100	EOT	DC4	$ (1)	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	° (1)	k	é (1)
1100	FF	FS	,	<	L	ç (1)	l	ù (1)
1101	CR	GS	–	=	M	§ (1)	m	è (1)
1110	SO	RS	.	>	N	^ (1)	n	~ (1)
1111	SI	US	/	?	O	_ (1)	o	DEL

(1) Caractères à usage national, ici français.

La signification des caractères spéciaux est donnée dans le tableau suivant :

Fonctions de mise en page	
BS	*Backspace* (retour en arrière)
HT	*Horizontal Tabulation*
LF	*Line Feed* (nouvelle ligne)
VT	*Vertical Tabulation*
FF	*Form Feed* (nouvelle page)
CR	*Carriage Return* (retour chariot)
Fonctions de contrôle de la transmission	
SOH	*Start Of Heading* (début d'en-tête)
STX	*Start of Text* (début de texte)
ETX	*End of Text*
EOT	*End Of Transmission*
ENQ	*Enquiry* (demande)
ACK	*Acknowledge* (Acquittement)
DLE	*Data Link Escape* (echappement liaison de données)
NAK	*Negative Acknowledge* (acquittement négatif)
SYN	*Synchronous Idle* (caractère de synchronisation)
ETB	*End of Transmission Block* (fin de bloc)

Fonctions de commande des périphériques	
DC1	*X-on* (mise en route)
DC2	
DC3	*X-off* (arrêt)
DC4	
Fonctions de séparation dans les fichiers	
US	*Unit Separator* (séparateur de sous-articles)
RS	*Record Separator* (séparateur d'article)
GS	*Group Separator* (séparateur de groupes)
FS	*File Separator* (séparateur de fichiers)
Autres caractères	
NUL	caractère vide (sans aucun effet)
BEL	*Bell* (sonnerie)
SO	*Shift Out* (hors code)
SI	*Shift In* (retour en code)
CAN	*Cancel* (annulation)
EM	*End of Medium* (fin de support)
SUB	*Substitution*
ESC	*Escape* (echappement)
DEL	*Delete*

Codage Canal

- *Canal de transmission idéalisé*

Un canal est discret si les deux alphabets X et Y sont discrets. On considèrera que le canal peut transmettre tous les K symboles de l'alphabet X de la source. On supposera que l'alphabet du canal, noté Z, est composé de D symboles, avec D<K.

Soit $\overline{n}$ la longueur moyenne des séquences

$$\overline{n} = \sum_{k=1}^{K} n_k . p(x_k)$$

Codage Canal

- *Canal de transmission idéalisé (suite)*

Dans le codeur idéal, nous désirons trouver un codeur qui assure une valeur minimale pour $\overline{n}$.

L'entropie de la source H(X), qui sera aussi l'entropie du codeur ***par messages***; l'entropie du codeur par symboles sera alors $\frac{H(X)}{\overline{n}}$. L'entropie maximale du codeur est *log*$_2$*D,* dont l'efficacité du codeur est

$$E = \frac{\frac{H(X)}{\overline{n}}}{\log_2 D}$$

E<1 dans tous les cas utiles, donc

$$\overline{n} \geq \frac{H(X)}{\log_2 D}$$

Codage canal

- *Canal de transmission réel*

Contrairement au canal idéalisé, un canal est affecté par le bruit. Le cana est caractérisé par une matrice de transition P (appelé aussi matrice stochastique)

$$P = \begin{bmatrix} p(y_1|x_1) & \cdots & p(y_N|x_1) \\ \vdots & \ddots & \vdots \\ p(y_1|x_M) & \cdots & p(y_N|x_M) \end{bmatrix}$$

(rappel : $p(y|x) = \dfrac{p(x|y) \cdot p(y)}{p(x)}$).

Codage canal

- *Canal de transmission réel (suite)*

Le canal est constant (par rapport à un canal à mémoire) si sa matrice de transition est constante dans le temps (ne change pas de message). Nous considérons ici uniquement des canaux constants.

Les cas extrêmes:

1. Le canal n'est pas perturbé par le bruit. Dans ce cas, M=N et PΞ I_N modulo une renumérotation des y ou des x (matrice identité d'ordre N)
2. Il n'y a aucune corrélation entre les messages d'entrée et les messages de sortie, c'est-à-dire

$$p(y_i / x_k) = p(y_i)$$

Dans ce cas, tous les lignes de la matrice sont identiques

Codage de source

- Un code source C pour une variable aléatoire X est une fonction de *X*, l'ensemble des valeurs possibles de X, vers Ø^k, l'ensemble des chaînes de symboles d'un alphabet D-aire.
- C(x) est le mot code correspondant au symbole x de la source
- n_x est la longueur du mot code C(x)
- La longueur moyenne attendue du code $\overline{n}$ du code C pour la variable X caractérisée par une distribution de probabilités p(x) est:

$$\overline{n} = \sum_{x \in X} p(x) n_x$$

Codage de source

- **Exemple 1:**
  - **X = { rouge, bleu }**
  - **Alphabet 3-aire: { A,B,C }**
  - **C(rouge)=AA, C(bleu)=ABC**
  - $\overline{n} = p(rouge) \times 2 + p(bleu) \times 3$

- **Exemple 2:**
  - **X = { 1,2,3,4 } , alphabet binaire: { 0,1 }**
  - **P(x=1) = ½,   C(1) = 0**
  - **P(x=2) = ¼,   C(2) = 10**
  - **P(x=3) = ? ,   C(3) = 110**
  - **P(x=4) = ? ,   C(4) = 111**
  - $\overline{n} = \sum_{x \in X} p(x).n_x = 1.75 = H(X)$

Codage de source

- *Exemple3*

Soit une source X qui produit deux symboles A et B avec les probabilités respectives de 4/5 et 1/5 à une cadence de 80 symboles par minutes.

L'entropie de cette source est

$$\begin{aligned} H(X) &= -\sum_{x \in X} p(x).\log p(x) \\ &= -0.2\log(0.2) - 0.8\log(0.8) \\ &= 0.72\ bit/symbole \end{aligned}$$

soit 0.72x80/60=0,99 bit/sec

Soit un canal binaire sans perte qui transmet 2 symboles 0 et 1 à une cadence de 1 symbole par seconde. La capacité du canal est évidemment de 1bit/sec

Codage de source

Exemple (suite)

- Le codage le plus simple qu'on puisse imaginer est:

Symboles x de la source	*Probabilités p(x)*	*Symboles C(x) du code*	*Produits $p(x).n_x$*
A	0,8	0	0,8
B	0,2	1	0,2

- La longueur moyenne des mots code, exprimée en nombre de symboles {0,1} du code C par symbole x **2** {A,B} de la source X, est

$$\begin{aligned}\overline{n} &= \sum_{x\in\{A,B\}} p(x).n_x \\ &= 0.8\times 1+0.2\times 1=1\end{aligned}$$

- Une suite typique de 30 symboles de la source est codé ainsi :

A	B	A	A	A	A	B	A	A	A	A	A	B	A	B	A	A	A	A	A	A	A	A	B	A	B	A	A	A	A
0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0

- Avec ce code, on génère donc 80 symboles/min, ce que le canal ne peut pas absorber

Codage de source

Exemple (suite)

- Un meilleur code peut être obtenu en groupant les symboles de la source par paires:

Symboles x de la source	*Probabilités p(x)*	*Symboles C(x) du code*	*Produits $p(x).n_x$*
AA	0,64	0	0,64
AB	0,16	10	0,32
BA	0,16	110	0,48
BB	0,04	111	0,12

- La longueur moyenne des mots code, en nombre de symboles par paire de lettres de la source, est

$$\overline{n} = \sum_{x \in \{AA, AB, BA, BB\}} p(x).n_x = 0.64 \times 1 + 0.16 \times 2 + 0.16 \times 3 + 0.04 \times 3 = 1.56$$

soit 1.56 / 2 = 0,78 par lettre de la source

- La même suite de 30 symboles de la source est maintenant codée avec 24 bits:

AB	AA	AA	BA	AA	AA	BA	BA	AA	AA	AA	AB	AB	AA	AA
10	0	0	110	0	0	110	110	0	0	0	10	10	0	0

- L'encodeur produit 0,78 x 80 = 62,4 symboles par minute, encore trop haut pour le canal

Codage de source

Exemple (suite)

- On peut grouper les symboles de la source 3 par 3 pour arriver au code suivant:

Symboles x de la source	*Probabilités p(x)*	*Symboles C(x) du code*	*Produits $p(x).n_x$*
AAA	0,512	0	0,512
AAB	0,128	100	0,384
ABA	0,128	101	0,384
BAA	0,128	110	0,384
ABB	0,032	11100	0,160
BAB	0,032	11101	0,160
BBA	0,032	11110	0,160
BBB	0,008	11111	0,040

- La longueur moyenne des mots code est 2,184 symboles par 3 lettres de la source, donc 0,728 par lettre
- L'encodeur produit donc 0,728 x 80 = 58,24 symboles par minute, suffisant pour le canal

Exemple (suite)

Codage de source

- Une suite typique de 30 symboles de la source est codée ainsi :

ABA	AAA	BAA	AAA	BAB	AAA	AAA	AAB	ABA	AAA
101	0	110	0	11101	0	0	100	101	0

- La suite de 30 symboles de la source est maintenant codée avec 22 symboles de code, 11 «0 » et 11 « 1 ».
- L'équiprobabilité des 0 et des 1 n'est pas un hasard, la capacité d'un canal à 2 symboles correspond au maximum de l'entropie de la source. Cette entropie est maximum quand les symboles sont équiprobables
- Cet encodeur doit disposer d'une mémoire et introduire un retard. Trois lettres de la source doivent être mémorisées avant qu'un mot code soit émis. Un mot code à 5 symboles est transmis plus lentement qu'un mot à un seul symbole et aussi plus lentement que l'arrivée d'un groupe de 3 symboles de la source primaire.

Codes préfixes-inégalité de Kraft

- ***Décodabilité d'un code***

✓ Non singulier : $x_i \neq x_j \Rightarrow C(x_i) \neq C(x_j)$

- Permet de décoder un symbole unique, mais pas une suite de symboles
- Pour décoder une suite, il faut introduire un symbole de ponctuation
- Exemple: code Morse

✓ Uniquement décodable:

- C^* est le code étendu tel que $C^*(x_1x_2\ldots x_n)=C(x_1)C(x_2)\ldots C(X_n)$
- C est uniquement décodable si C^* est non singulier

✓ Instantanément décodable:

- Chaque symbole C peut être décodé sans référence aux symboles suivants
- Condition fixe: aucun mot code n'est le préfixe d'un autre mot code

Codes préfixes-inégalité de Kraft

Condition de préfixe

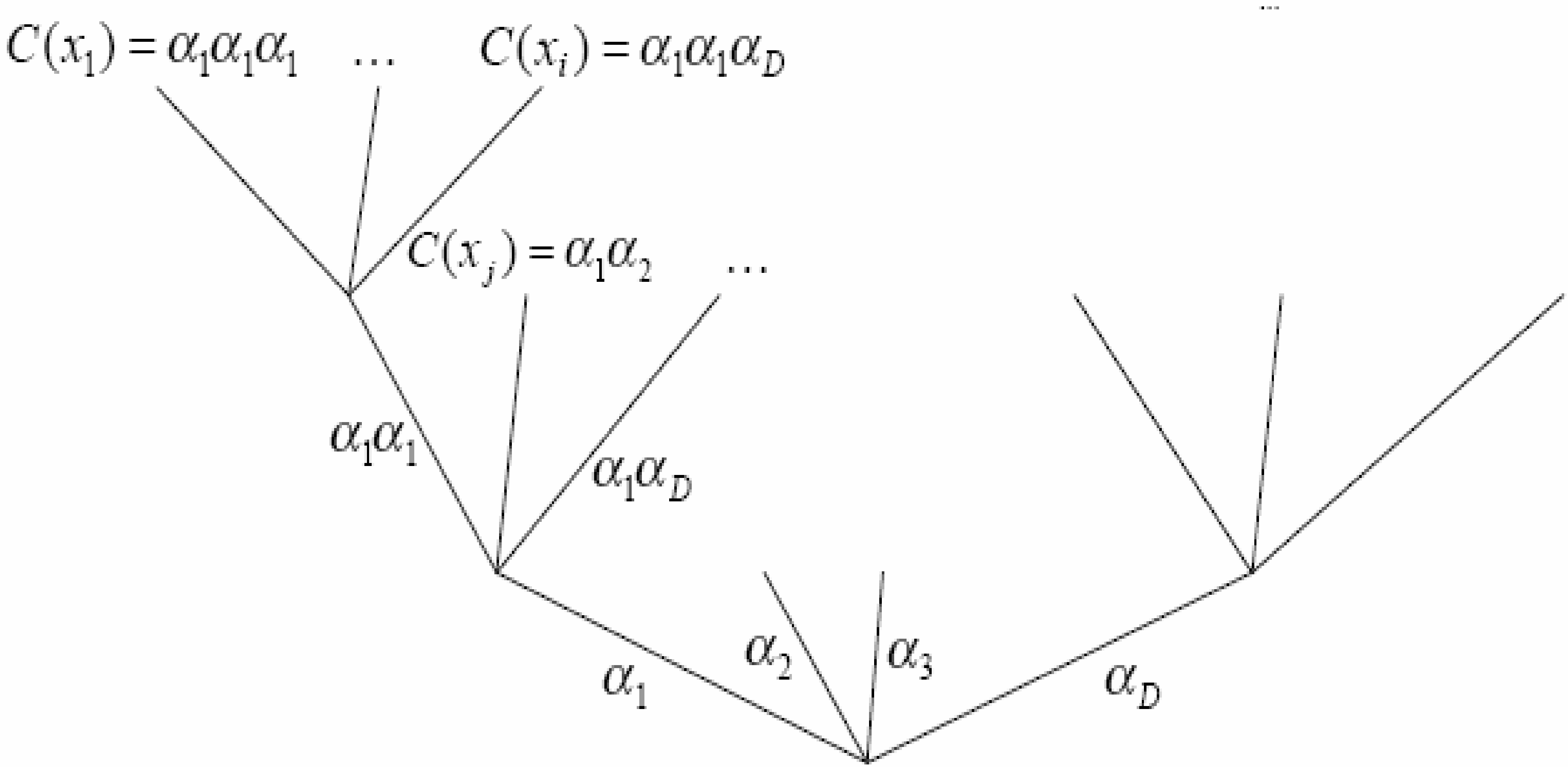
- Soit une source X générant des symboles x_k d'un alphabet K-aire (k **2** {1,…,K})
- Soit $C(x_k)$ le mot-code corresponsant au symbole x_k de la source
- $C(x_k)$, de longueur n_k, peut s'écrire $C(x_k) = (c_{k,1}, c_{k,2} \ldots, c_{k,n_k})$, où $c_{k,i}$ représente une lettre α_j de l'alphabet D-aire du code (*j* **2** {1,…,*D*}).
- Toute séquence de lettre construite par la partie initiale de $C(x_k)$ est appelée préfixe
- La condition de préfixe stipule que dans un code aucun mot-code n'est le préfixe d'un autre mot-code

- Les codes à condition de préfixe sont à décodage unique
- Tous les codes à décodage unique ne satisfont pas nécessairement la condition de préfixe
- La condition de préfixe permet de reconnaître la fin d'un mot-code, donc un décodage sans retard (*codes instantanés*)

Codes préfixes-inégalité de Kraft

Construction d'un code à préfixe

- **Un code préfixe D-aire peut etre représenté par un arbre D-aire dont les feuilles sont les mots du code**



Codes préfixes-inégalité de Kraft

- *Propriétés des codes préfixes*

Pour tout code préfixe K mots sur un alphabet de D symboles dont les mots codes ont pour longueur $n_1, n_2, \ldots, n_k$, ces entiers satisfont

$$\sum_{k=1}^{K} D^{-n_k} \leq 1$$

Inversement, étant donnés des entiers $n_1, n_2, \ldots, n_k$ qui satisfont cette inégalité, on peut construire un code préfixe dont les mots code ont ces longueurs.

Codes préfixes-inégalité de Kraft

Inégalité de Kraft: preuve

- Soit n_{max} la longueur du mot le plus long du code
- On considère l'arbre D-aire de profondeur n_{max}, les feuilles au niveau n_{max} sont soit
  - des mots code (1)
  - des descendants de mots code (2)
  - aucun des deux (3)
- Un mot code de longueur n_k a $D^{n_{\max}-n_k}$ descendants.
- La condition de préfixe implique que l'ensemble de ces descendants doit être disjoint pour tous les mots codes.
- Si l'on considère l'ensemble des feuilles de niveau n_{max} qui sont ou descendent d'un mot code, soit (1)+(2), il y en a donc

$$\sum_{k=1}^{K} D^{n_{\max}-n_i}$$

Codes préfixes-inégalité de Kraft

Inégalité de Kraft: preuve

Cet ensemble (1)+(2) de feuilles étant descendant de mots-codes est évidemment inclus dans l'ensemble (1)+(2)+(3) de toutes les feuilles de niveau n_{max}. On a donc

$$\sum_{k=1}^{K} D^{n_{\max}-n_k} \leq D^{n_{\max}}$$

Et en divisant par $D^{n_{\max}}$, on obtient bien l'inégalité de Kraft

$$\sum_{k=1}^{K} D^{-n_k} \leq 1$$

Codage et optimisation

- La recherche d'un bon code pour une source caractérisée par une probabilité *p(x)* peut être vue comme un problème d'**optimisation sous contrainte**: trouver les entiers $n_1, n_2, \ldots n_K$
  - Minimiser $\bar{n} = \sum_{x \in X} p(x) n_x$
  - Sous contraite de $\sum_i D^{-n_i} \leq 1$
- En **négligeant la contrainte de longueurs entières** et en assumant l' égalité pour la contrainte, on applique la méthode de Lagrange:

$$J = \sum_x p(x).n_x + \lambda.\left(\sum_x D^{-n_x} \right)$$

$$\frac{\partial J}{\partial n_x} = p(x) - \lambda.D^{-n_x}.\log D = 0$$

$$D^{-n_x} = \frac{p(x)}{\lambda.\log D}$$

$$\sum_x D^{-n_x} = \sum_x \frac{p(x)}{\lambda.\log D} = \frac{1}{\lambda.\log D} = 1$$

$$\lambda = 1/\log D$$

$$p(x) = D^{-n_x}$$

$$n_x = -\log_D p(x)$$

Codage et optimisation

Codes optimaux

- **Pour le code optimaux, la longueur moyenne est égale à l'entropie de la source:**

$$n_x = -\log_D p(x)$$

$$\bar{n} = \sum_x p(x) n_x = -\sum_x p(x) \log_D p(x) = H_D(X) = \frac{H(X)}{\log D}$$

- **Exemple 2 (rappel):**
  - **X = { 1,2,3,4 } , alphabet binaire: { 0,1 }**
  - **P(x=1) = ½,** **n_1 = -$\log_2$(½) = 1** **C(1) = 0**
  - **P(x=2) = ¼,** **n_2 = -$\log_2$(¼) = 2** **C(2) = 10**
  - **P(x=3) = ? ,** **n_3 = -$\log_2$(?) = 3** **C(3) = 110**
  - **P(x=4) = ? ,** **n_4 = -$\log_2$(?) = 3** **C(4) = 111**

$$\bar{n} = \sum_x p(x) n_x = 1.75 = H(X)$$

Théorème de Shannon

En pratique, on peut négliger la contrainte n_x entier, et il est donc souvent impossible d'atteindre l'optimum. Cependant, on peut prouver qu'il est possible de s'en approcher:

Si *H(X)* est l'entropie de la source, alors

$$\frac{H(X)}{\log D} \le \bar{n} \le \frac{H(X)}{\log D} + 1$$

Si les symboles de la source sont groupées *L* par *L* avant le codage, alors

$$\frac{H(X)}{\log D} \le \bar{n} \le \frac{H(X)}{\log D} + \frac{1}{L}$$

Théorème de Shannon

Théorème Shannon: Preuve

On a vu précédemment que pour un code optimal, $\bar{n} = H_D(X)$

Pour un code non optimal, on a donc $\bar{n} \geq H_D(X)$

Pour prouver l'autre inégalité, on construit un code avec des longueurs $n_x = \left\lceil \log_D \frac{1}{p(x)} \right\rceil$

Ces longueurs permettent de construire un code préfixe, vu qu'elles satisfont l'inégalité de Kraft

$$\sum_x D^{-\left\lceil \log_D \frac{1}{p(x)} \right\rceil} \leq \sum_x D^{-\log_D \frac{1}{p(x)}} = \sum_x p(x) = 1$$

Par ailleurs, on a $\log_D \frac{1}{p(x)} \leq n_x < \log_D \frac{1}{p(x)} + 1$

ce qui donne, en moyenne pondérée par les p(x), $H_D(X) \leq \bar{n} < H_D(X) + 1$

c.q.f.d.

Théorème de Shannon

Théorème Shannon: Preuve (suite)

- Si cette borne supérieure n'est pas satisfaisante, on peut l'améliorer en groupant les symboles L par L. On définit un nouveau code C^* pour coder les séquences $(X_1,X_2,\ldots X_L)$ avec des mots code de longueur n^*_x qui satisfont

$$H_D(X_1,X_2,\ldots X_L) \leq \bar{n}^* < H_D(X_1,X_2,\ldots,X_L)+1$$

$$L.H_D(X) \leq L.\bar{n} \leq L.H_D(X)+1$$

$$H_D(X) \leq \bar{n} \leq H_D(X)+\frac{1}{L}$$

- **Exemple 3 (rappel):**
  - X = { A,B } , p(A)=0.2 , p(B)=0.8 , H(X)=0.72
  - 1 symbole à la fois: H(X) = 0.72 = n = 1 < 1.72 = H(X) + 1
  - 2 symboles à la fois: H(X) = 0.72 = n = 0.78 < 1.22 = H(X) + 1/2
  - 3 symboles à la fois: H(X) = 0.72 = n = 0.728 < 1.05 = H(X) + 1/3

Codes de Fano et Huffman

- **Code binaire respectant** $$n_x = \left\lceil \log_2 \frac{1}{p(x)} \right\rceil$$

- **Algorithme:**
  - **Ordonner les symboles de la source dans l'ordre des probabilités**
  - **Grouper en deux ensembles d'égale probabilité, séparés avec 0 et 1**
  - **Diviser chaque ensemble en deux sous-ensemble, séparés avec 0 et 1**
  - **Continuer jusqu'à ce que chaque sous-ensemble ne contienne qu'un seul symbole de la source**

- **Exemple:**
  - **source avec 8 symboles A, B, …, H**
  - $P(A) = 0{,}1$, $P(B) = 0{,}18$, $P(C) = 0{,}4$, $P(D) = 0{,}05$, $P(E) = 0{,}06$, $P(F) = 0{,}1$, $P(G) = 0{,}07$ **et** $P(H) = 0{,}04$.
  - **L'entropie de cette source sans mémoire est** $H = 2{,}55$ **bits par symbole**

Codes de Fano et Huffman

Code Fano

• $A = 0{,}1$	$C = 0{,}4$	0			00
• $B = 0{,}18$	$B = 0{,}18$				01
• $C = 0{,}4$	$A = 0{,}1$	1	0		100
• $D = 0{,}05$	$F = 0{,}1$				101
• $E = 0{,}06$	$G = 0{,}07$		1	0	1100
• $F = 0{,}1$	$E = 0{,}06$				1101
• $G = 0{,}07$	$D = 0{,}05$			1	1110
• $H = 0{,}04$	$H = 0{,}04$				1111

$$\overline{n} = \sum_x p(x) n_x = 2{,}64$$

L'efficacité de codage est 2,55/2,64, soit 96,6%

Codes de Fano et Huffman

Code Hoffman:

Code plus efficace que le code Fano, décrit ici pour le cas binaire mais généralisable à d'autres alphabets.

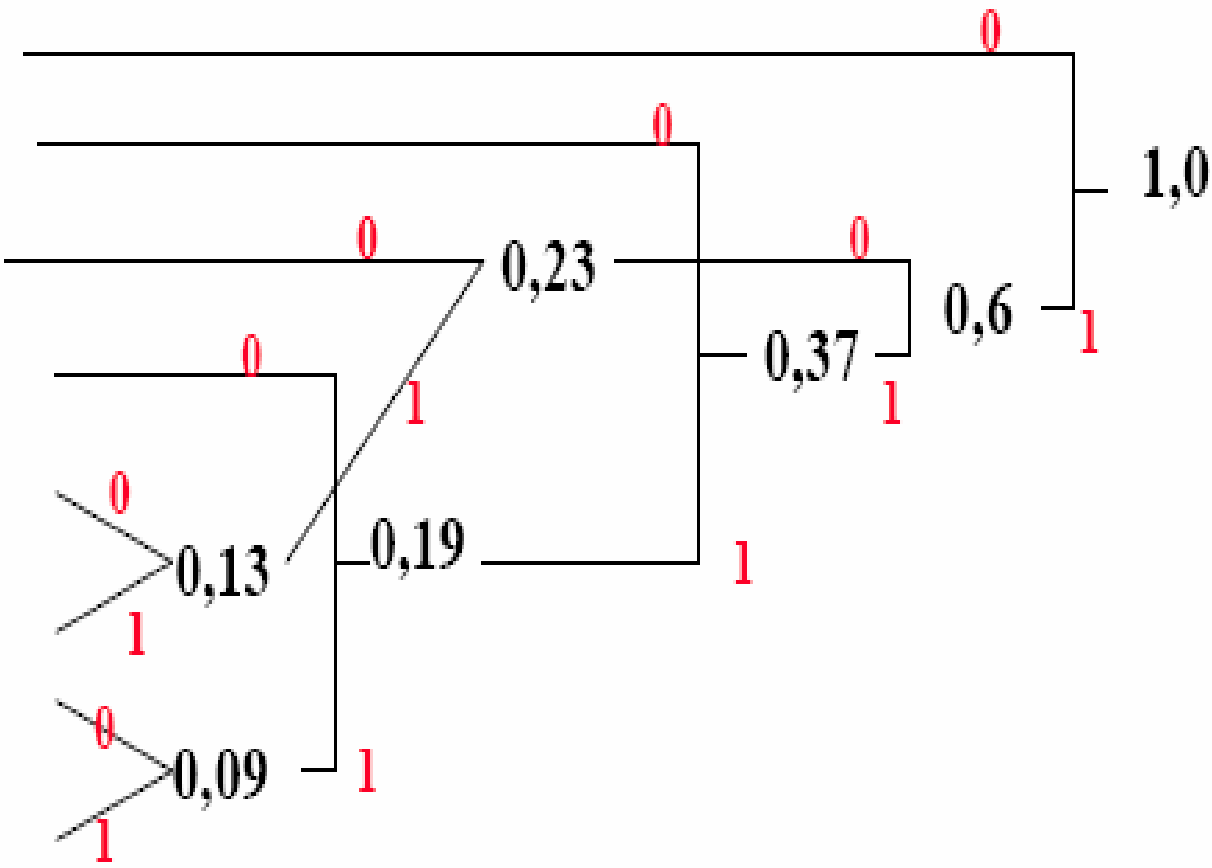
Algorithme:

- Ordonner les symboles de la source dans l'ordre des probabilités
- Isoler les deux symboles les moins probables et les distinguer avec 0 et 1
- Regrouper ces deux symboles en un seul nouveau en additionnant leurs probabilités
- Recommencer jusqu'à ce que toutes les symboles soient regroupés

Codes de Fano et Huffman

Exemple comme précédemment:

- $C = 0,4$
- $B = 0,18$
- $A = 0,1$
- $F = 0,1$
- $G = 0.07$
- $E = 0,06$
- $D = 0,05$
- $H = 0,04$



Code:

C:	0
B:	110
A:	100
F:	1110
G:	1010
E:	1011
D:	11110
H:	11111

$$\overline{n} = \sum_x p(x) n_x = 2,61$$

L'efficacité de codage est 2,55/2,61, soit 97,8%

Codes de Fano et Huffman

Code Huffman

- On peut également construire des code de Huffman D-aires.
  - regrouper D a chaque étape
  - compléter l'ensemble des symboles de source par des symboles de probabilité nulle de sorte que le nombre de symboles de X soit 1+k.(D-1) avec k entier.

- Propriété: le code d'Huffman est optimal
  - On peut prouver qu'il existe un code optimal qui satisfait
    - » Si $p(x) > p(y)$, alors $n_x = n_y$
    - » Les deux plus longs mots code ont la même longueur
    - » Les deux plus longs mots code ne diffèrent que par leur dernier bit et correspondent au deux symboles les moins probables

  - On obteint le code d'Huffman en groupant ces deux symboles les moins probables et en itérant

Codes de Fano et Huffman

Erreur sur l'estimation de p(x):

- **Dans de nombreux cas pratiques, p(x) n'est connue que par estimation. Quel est l'effet de coder un source p(x) avec un code optimisé pour q(x) ?**

$$n_x = \left\lceil \log \frac{1}{q(x)} \right\rceil$$

$$\bar{n} = \sum_x p(x).\left\lceil \log \frac{1}{q(x)} \right\rceil$$

$$\sum_x p(x).\log \frac{1}{q(x)} \leq \bar{n} < \sum_x p(x).\left(\log \frac{1}{q(x)} + 1 \right)$$

$$\sum_x p(x).\log \frac{p(x)}{q(x)} \frac{1}{p(x)} \leq \bar{n} < \sum_x p(x).\left(\log \frac{p(x)}{q(x)} \frac{1}{p(x)} + 1 \right)$$

$$\sum_x p(x).\log \frac{p(x)}{q(x)} + \sum_x p(x).\log \frac{1}{p(x)} \leq \bar{n} < \sum_x p(x).\log \frac{p(x)}{q(x)} + \sum_x p(x).\log \frac{1}{p(x)} + 1$$

$$D(p\|q) + H(p) \leq \bar{n} < D(p\|q) + H(p) + 1$$

Détection et correction des erreurs

Quelle que soit la qualité d'une ligne de transmission, la probabilité d'apparition d'erreurs est non nulle.

Pour certains types de transmissions des erreurs groupées peuvent apparaître par exemple à cause des:

1. Parasites électromagnétiques pour les transmissions sur paires torsadées non blindées
2. Conditions atmosphériques pour les transmissions par satellite

La détection et la correction des erreurs est fondée sur l'utilisation d'une information redondante transmise avec l'information utile. L'ajout de cette information redondante est obtenu par recodage

Détection et correction des erreurs

Remarque: On ne peut pas être sûr qu'un message reçu est correct. Quel que soit le code employé, il ne peut pas détecter toutes les erreurs possibles. Par conséquence, les codes choisis dans une situation particulière sont ceux qui détectent le mieux et éventuellement corrigent les erreurs les plus fréquentes dans la situation en question.

Types de codes:

1. Codes en blocs. Si les informations utiles à transmettre sont découpées en blocs et à chaque bloc on associe un mot de code qui ne dépend que du bloc en question, le code est appelé code en blocs

Détection et correction des erreurs

2. Codes continus. Si l’information à expédier n’est pas divisible en blocs et la redondance est introduite de façon continue dans l’information utile, le code est appelé code continu.
3. Codes cycliques

NB: Le développement de ces codes peut être consulté dans le support numérique PDF déposé dans la rubrique ressource complémentaire de la présentation de la séquence 2 de l’UV TVN 224 ou l’apprenant pourra procéder au téléchargement.

Détection et correction des erreurs

- Quelques définitions

❖ Pour un code détecteur d'erreurs, l'**efficacité** de détection **e** (à ne pas confondre avec l'efficacité définie au chapitre précédent) est le rapport moyen entre le nombre de messages erronés reconnus comme tels et le nombre total de messages erronés. La proportion 1 – **e** de messages sont donc erronés et reconnus comme corrects.

❖ Le **taux d'erreurs brut** ζ est la proportion moyenne de messages erronés reçus (détectés ou non comme tels). Le taux d'erreurs brut dépend du taux d'erreurs par bit ; si la longueur moyenne des messages est de n bits et les erreurs sont indépendantes, avec une probabilité p par bit, alors $\zeta = 1 - (1 - p)^n$.

Détection et correction des erreurs

- Quelques définitions (suite)

❖ Le **taux d'erreurs résiduel *q*** est la proportion des messages qui restent erronés (après détection et correction des erreurs, par codes ou par retransmission). Pour qu'un message erroné soit reconnu comme correct, il est nécessaire : 1° qu'il soit erroné (probabilité ζ) et 2° qu'il n'ait pas été détecté comme tel par le code (probabilité 1 – **e**) ou qu'erroné une première fois et détecté comme tel (probabilité $\zeta \times$ **e**) il soit erroné une seconde fois (erreur différente) et non détecté comme tel (probabilité ζ^2.**e .**(1- **e**)), etc. Par conséquence $q = \zeta.(1-e) + \zeta^2.e.(1-e) + \ldots$

Détection et correction des erreurs

- *Correction par retransmission*

Dans le cas où les erreurs peuvent être détectées mais ne peuvent pas être corrigées par la technique de codage, la retransmission devient impérative.

Les types de retransmissions sont:

1. Retransmission avec arrêt et attente
2. Retransmission continue
3. Retransmission sélective

Compression des informations

- *Compression des images*

Une image est représentée par une collection (matrice 2D) de pixels, chaque pixel étant décrit par un entier unique (plages de gris ou adresse dans une table de sélection de couleurs) ou par plusieurs entiers (composantes RVB, ou luminance et chrominances)

➢ Méthode JPEG

La norme JPEG (Joint Photographics Expert Group) a été adopté en 1992 et utilise deux étapes dans la compression:

1° Une transformation cosinus discrète appliquée sur des blocs de 8×8 pixels sur chaque composante de l'image (luminance Y, chrominance U, chrominance V) et donne 64 coefficients de fréquence par composante.

2° Les coefficients TCD sont divisés par des valeurs présentes dans une table de quantification et les valeurs sont arrondies à des entiers. C'est dans cette étape que certaines informations sont éliminées.

3° Les coefficients obtenus à l'étape antérieure sont comprimés par un algorithme de Huffman dynamique (adaptatif) à partir de tables de codes connues (afin d'éviter le stockage d'un en-tête pour chaque bloc).

Compression des informations

- *Compression des sons*

Les sons sont représentés par une suite d'échantillons obtenus à une certaine cadence (par exemple, pour les CD audio la fréquence d'échantillonnage est de 44 KHz et les échantillons sont codés sur 16 bits).

➢ Méthode PASC

Cette méthode (*Precision Adaptive Sub-band Coding*), employée pour les DCC, utilise les observations que nous avons faites.

La compression se déroule selon les étapes suivantes :

1° Le signal subit une division spectrale sur 32 bandes de 750 Hz de largeur et sur chacune on calcule le niveau moyen.

2° En utilisant les connaissances sur les courbes dynamiques d'audibilité, on détermine dans quelles bandes le signal peut être éliminé.

3° Dans chaque bande retenue, le signal est codé en utilisant un nombre variable de bits répartis entre les sous-bandes selon la différence entre le niveau de la sous-bande et le niveau moyen dans la bande.

Référence bibliographique

1. TRANSMISSION DE L'INFORMATION
SUPPORT DE COURS
E3i, 2001-2002 Université de Tours : Michel Crucianu
2. Signaux et systèmes 20 avril 2004
Codage de source: **Signal Processing Institute Swiss Federal Institute of Technology, Lausanne**
3. Ressources Internet