Perturbations sonores

Cet article est le résultat d'un travail mené dans le cadre d'un atelier MATh.en.JEANS par des élèves des Lycées Montaigne de Bordeaux et Sud Médoc du Taillan-Médoc durant l'année scolaire **2011-2012**.
Élèves :
Gabriel DETRAZ (TS), Gabriel SIMONET-DAVIN (TS), Hugo LALLEMAND (2nd), Cédric MERILLOU(TS), Hugo DODELIN(TS), Paul ISART(TS), Valentin DUBOIS(TS), Adrien HAILNAUT(TS), Héloïse BOGAS-DROY(2nd), Marjorie PHILIPPOT(2nd).
Professeurs :
Olivier CARCONE, Dominique GRIHON, Pierre GRIHON, Sébastien MAIMARAN
Chercheur :
Christine BACHOC, Université BORDEAUX 1

Énoncé du sujet :

« Tic et Tac communiquent avec des mots binaires (ex. 010011) par téléphone. Toutefois, la ligne n'est pas très bonne et il arrive que Tic ou Tac entende « BIIIIIIIIP » au lieu de 0 ou 1.

-Tout d'abord, supposons que cela n'arrive pas plus d'une fois par mot. Quelle stratégie Tic et Tac peuvent-ils adopter pour poursuivre leurs conversations sans perte d'information ?

-Décidément le téléphone, c'est ringard ; ils décident de correspondre par mail. Mais alors, à cause de certaines interférences, il arrive qu'un chiffre soit modifié, c'est-à-dire un 0 transformé en 1 ou bien un 1 en 0. Pas plus de un par mot, par contre Tic et Tac n'ont aucun moyen de savoir lequel. Que peuvent-ils faire ? »

Démarche :

Le message transmis est une chaîne de 0 et de 1 appelés bits d'une longueur infinie. (1) Nous décidons de limiter le nombre de bits étudiés à des « mots » constitués de 4 bits afin de simplifier le problème. La démarche que nous allons aborder dans tout ce dossier va consister à rajouter des bits à l'information initiale. Ces bits supplémentaires vont servir à détecter les éventuelles erreurs et nous permettre de les corriger.

I) Correction d'une erreur localisée

On a cherché à corriger une erreur localisée. On a aléatoirement choisi un mot de 4 bits. Nous avons essayé le principe du bit de parité : lorsque le nombre de 1 est pair, on rajoute un 0 et lorsque le nombre de 1 est impair on met un 1.

Pour le mot **0100** on compte un seul bit prenant la valeur 1 ; on rajoute donc un bit de parité de valeur 1 : **0100 1**

Voici un mot où le bit coloré en vert est illisible : **0?10 1**

Cependant, comme le bit de parité possède la valeur 1, il y a un nombre impair de 1 dans le message originel. Il s'y trouve déjà un bit de valeur 1 et deux bits de valeur 0, on en déduit donc que le bit vert a pour valeur 0 donc le mot est **0010 1**

Cependant dans la réalité une erreur dans une trame (2) ne se solde pas par un « trou » dans cette dernière. Un bit est une donnée qui ne peut prendre que deux valeurs : 0 ou 1. De ce fait, on ne connaît pas la position de l'erreur et le problème est de la localiser.

II) Correction d'une erreur non localisée

Deux hypothèses sont appliquées dans la suite du dossier :

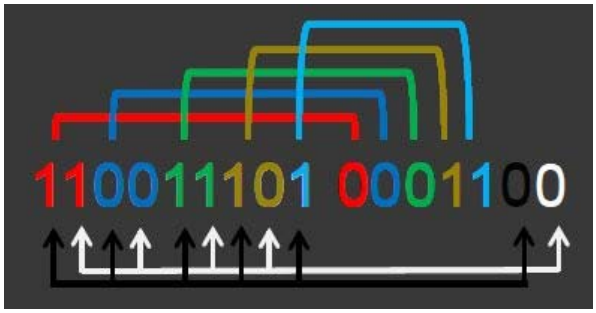
- Les bits de parité peuvent aussi être altérés
- Les messages ne contiennent qu'une **ou** aucune erreur.

Nous avons tenté deux approches :

A) Première approche graphique

Afin de pouvoir localiser ET corriger une erreur, nous avons trouvé qu'il fallait croiser les bits de parité. En d'autres termes, il faut que chaque bit intervienne dans au moins deux bits de parité. On a ainsi groupé les bits par deux et associé un bit de parité à chaque groupe. (3)

Puis afin de croiser les bits, on a ajouté un bit de parité pour l'ensemble des premiers bits de chaque groupe, et un deuxième sur tous les seconds bits de chaque groupe, comme sur l'exemple ci-dessous. (4)



Remarque: si le nombre de bits du message originel n'est pas pair, le dernier bit esseulé forme un groupe et est donc codé par un bit de parité.

Jusqu'ici, nous faisions des groupes de deux bits, auxquels on attribuait un unique bit de parité. Cependant, cette méthode présente l'inconvénient de multiplier le nombre de bits de parité et donc d'alourdir le message. En effet, on avait 7 bits de parité pour 9 bits utiles...

Nous cherchons à **optimiser** le nombre de bits de parité afin de garantir la transmission correcte du message originel sans multiplier la taille du message. (5)

En suivant le raisonnement précédent, nous avons groupé les bits par trois, puis par quatre en tentant de réduire le nombre de bits de parité.

En posant :

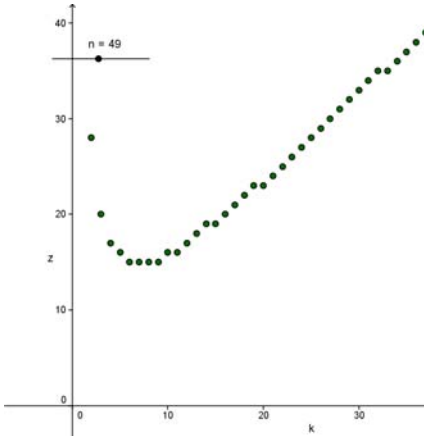
- n le nombre de bits du message originel
- z le nombre de bits de parité
- k le nombre de bits par groupes

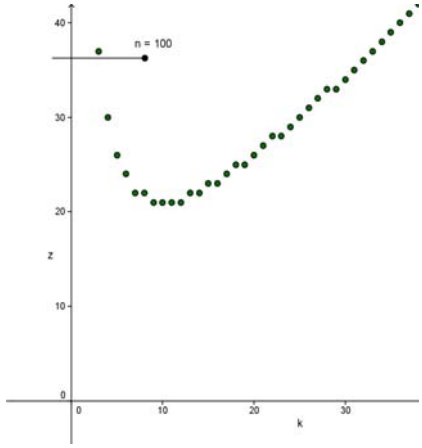
Nous avons traduit le problème avec la formule suivante où $[\ ]$ désigne l'arrondi entier supérieur : $z = \left[\frac{n}{k}\right] + k$

soit le nombre *z* de bits de parité est égal à l'arrondi supérieur du nombre de bits originels *n* divisé par le nombre *k* de bits par groupes, additionné du nombre de bits par groupes.

En plaçant en abscisse le nombre *k* de bits par groupes, en ordonnée le nombre *z* de bits de parité, et en traçant la courbe représentative de la fonction ci-dessus pour différentes valeurs de *n*, nous avons conjecturé que ces fonctions présentaient un minimum pour $k = \left[\sqrt{n}\right]$.

Voici les graphiques pour $n = 49$ et $n = 100$.





On peut ainsi calculer le nombre de bits de parité nécessaires pour coder le message.
Mais nous avons cherché à prouver notre conjecture.

B) <u>Deuxième approche : analogie du carré</u>

Nous pouvons traiter le problème de manière analogue en plaçant nos bits dans un tableau, comme ceci :
Message = 110011101

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

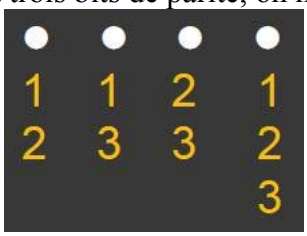
Le message est placé dans les colonnes du tableau, et on peut mettre les bits de parité correspondants pour chaque ligne et chaque colonne du tableau.
Ici, *n* représente l'aire en gris, *k* le nombre de colonnes et *z* le demi périmètre. Afin d'optimiser *z*, on conclut qu'il faut un périmètre minimal pour une aire fixée. La figure qui correspond à cette idée dans notre cas est le carré, d'où le périmètre doit être égal à quatre fois la racine de l'aire.

z étant le demi-périmètre et *n* l'aire et *k* le côté, cela donne pour des entiers $k = \left[\sqrt{n}\right]$.

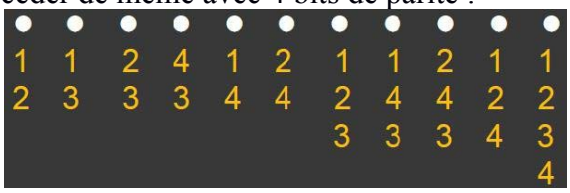
Théorème 1
Pour un mot de *n* bits, le nombre minimal de bits de parité associés à des groupes de bits de taille identique permettant de détecter une erreur est obtenu avec des groupes de $\left[\sqrt{n}\right]$ éléments.

C) Tentative d'amélioration : numérotation

La solution que nous venons de trouver est–elle pour autant la plus efficace ?
Afin de trouver une solution plus optimisée, (6) on va se placer non plus du coté des bits originels (en cherchant à les grouper), mais du coté des bits de parité.
On commence par numéroter les bits de parité. On place alors le numéro des bits de parité vérifiant un bit utile sous ce dernier, de sorte à ce que chaque bit ait sous lui **un couple de nombres unique.**
Par exemple, on a 4 bits originaux et trois bits de parité, on note un tel message ainsi :



Les bits originaux sont représentés par les points. On remarque que l'on a 4 combinaisons de bits de parité. On peut procéder de même avec 4 bits de parité :



Si une erreur se glisse dans les bits lors de la transmission, deux, trois ou quatre bits de parité vont indiquer une erreur : le bit faux est celui qui est vérifié par tous ces bits.

D) Recherche d'une relation entre *n* et *z*

- *n* le nombre de bits du message originel
- *z* le nombre de bits de parité
- *k* le nombre de bits par groupes

On a remarqué que :

- pour **z = 2**, on a **n = 1**, soit $\mathbf{2^2 - 3}$
- pour **z = 3**, on a **n = 4**, soit $\mathbf{2^3 - 4}$
- pour **z = 4**, on a **n = 11**, soit $\mathbf{2^4 - 5}$
- pour **z = 5**, on a **n = 26**, soit $\mathbf{2^5 - 6}$
- pour **z = 6**, on a **n = 57**, soit $\mathbf{2^6 - 7}$

On conjecture donc la relation suivante :

$$n = 2^z - z - 1.$$

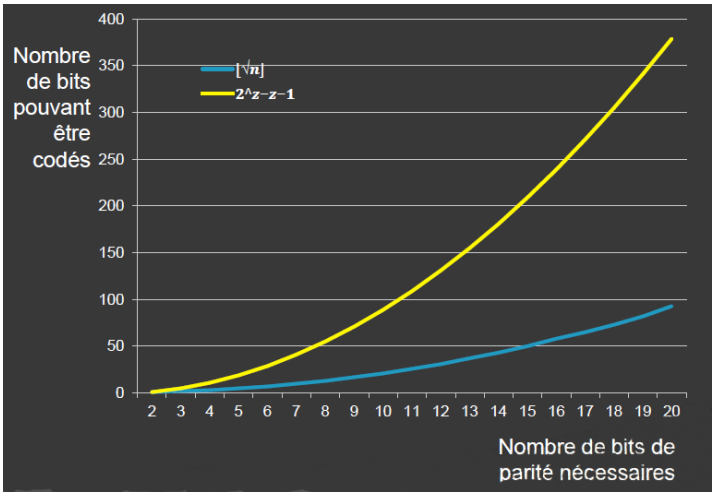
Cette relation peut être démontrée en utilisant le triangle de Pascal :

z\σ	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1

Les colonnes représentent le nombre **σ** de bits de parité vérifiant un bit originel donné ; les lignes représentent le nombre **z** de bits de parité.
Il faut que chaque bit soit associé à **un couple unique de deux bits** de parité (7)
(il peut néanmoins être associé à plus de bits de parité).
Les valeurs 0 et 1 de σ ne sont pas utilisées donc la somme d'une ligne sans les deux premiers termes représente le nombre de bits originels vérifiables. Ainsi pour un message de 4 bits originels comme vu précédemment, on a 3 fois 2 bits de parité et 1 fois 3 bits de parité.
La somme des termes de la $\mathbf{z}^{\text{ième}}$ ligne du triangle de Pascal vaut $\mathbf{2^z}$. Les groupes de 0 ou 1 bits sont inutiles. Ces groupes sont au nombre de **z+1** ; il faut donc retrancher **z+1** à $\mathbf{2^z}$**.** On trouve bien la formule précédente : $n = 2^z - z - 1$.

Théorème 2
On peut détecter une erreur avec *z* bits de parité et en utilisant toutes les tailles de combinaison de 2 à *z* pour un mot contenant $n = 2^z - z - 1$ bits.

En comparant cette méthode avec la précédente, on obtient le graphique ci-dessous. On remarque que pour un même nombre de bits de parité, on peut coder beaucoup plus de bits avec la deuxième méthode qu'avec la première.

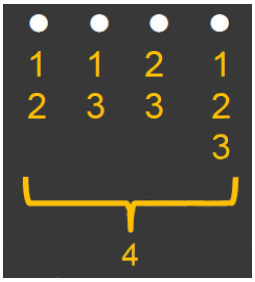


(8)

III. Correction de deux erreurs

Dans les cas précédents si deux erreurs avaient lieu dans une trame il devenait impossible de décoder celle-ci convenablement. En effet, la deuxième erreur compensait la première, et la trame était donc identifiée comme correcte alors qu'elle était doublement fausse.
Comme nous n'avons pu trouver de solution totale, nous avons cherché à contourner ce problème, de manière à permettre de détecter la présence d'erreurs multiples.
Notre solution consiste à rajouter un bit de parité sur l'ensemble de la trame codée. Ainsi :

- S'il n'y a aucune erreur : les bits de parité et le bit d'ensemble ne vont indiquer aucune erreur.
- S'il y a une erreur : les bits de parité et le bit d'ensemble vont indiquer une erreur.
- S'il y a deux erreurs : les bits de parité vont indiquer une erreur mais pas le bit d'ensemble.



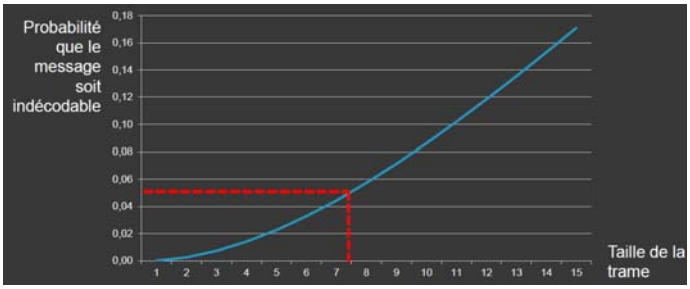
On peut donc détecter la présence de deux erreurs ou plus, mais pas les identifier ni les corriger. La trame doit être renvoyée à l'expéditeur, en espérant qu'elle sera plus correcte la prochaine fois !

IV. Equilibre erreur / longueur

On a remarqué que plus **z** est grand, plus le rapport bits de parité/bits utiles est intéressant. Nous avons donc voulu savoir s'il est plus intéressant de faire des chaînes longues ou des chaînes courtes.

Un problème qui survient lorsque l'on transmet une chaîne longue est que la probabilité qu'elle contienne une erreur est plus importante que pour une chaîne courte. Supposons par exemple que la probabilité qu'un bit soit altéré vaut 0,05 (5%). Si on suppose que les bits sont indépendants les uns de autres, on sait que le nombre aléatoire X de bits altérés sur une chaîne de x bits suit une loi binomiale de paramètres x et 0,05. Alors la probabilité qu'une chaîne de x bits puisse être lue (avec moins de deux erreurs) est égale à $\mathrm{P}(X<2)=0{,}95^{x}+x.0{,}95^{x-1}.0{,}05.$

Le graphique ci-dessous nous permet de bien identifier la limite au-delà de laquelle la probabilité que la trame soit indécodable est trop grande pour être acceptable.



Si l'on veut que cette probabilité ne dépasse pas 5%, alors il faut d'après le graphique fabriquer des chaînes de 8 bits maximum

Notes d'édition

(1) Plutôt que de "longueur infinie", il vaudrait mieux parler de longueur inconnue, non bornée, arbitrairement grande, etc.

(2) trame : bloc d'information, ici série de chiffres d'une certaine taille.

(3) les couleurs au-dessus dans le schéma qui suit

(4) en noir et en blanc en dessous

(5) optimiser signifie : faire au mieux, ici réduire au maximum le nombre de bits de parité

(6) La formulation est maladroite : on vient de démontrer que c'était l'optimal !

(7) Le mot « couple » est inadapté : à un bit du message peut correspondre plus que deux bits de parité. On peut d'ailleurs se demander pourquoi ne pas associer à certains bits un seul bit de parité ?

(8) Les paramètres des deux fonctions ne sont pas les mêmes : « n » et « z ». De plus, étant donné qu'il s'agit de valeurs entières, un tableau, ou un graphe « en escalier » éclairerait peut-être mieux.