# Comment ça marche ?

# Les transmissions radio-numériques

# 13 – Détection des erreurs

Par le radio-club F6KRK

***Après avoir vu le formatage des données et les modes de transmission, nous allons maintenant aborder quelques méthodes employées pour détecter les erreurs.***

**Présentation du problème**

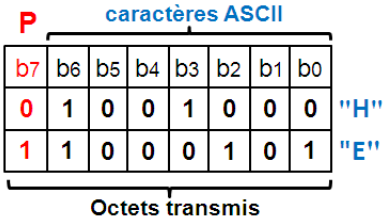
Nous avons vu que les données numériques transmises étaient organisées par blocs. Après traitement, une seule erreur restante dans un bloc entraîne le rejet de celui-ci. Le problème posé est de déterminer la taille optimale du bloc, compte tenu de la nature des données à transmettre. Par exemple pour du texte, la taille du bloc sera le caractère. Pour de la téléphonie et pour du visuel, la taille du bloc sera telle que son absence ne devra quasiment pas être perçue par nos sens. Pour un fichier informatique, le bloc devra impérativement être retransmis après un non-acquittement (d'où sa numérotation).
Ainsi à la réception on devra avoir la possibilité de détecter les erreurs et éventuellement de les corriger. Ces opérations ne pourront se faire qu'en augmentant la quantité de données transmises. Le concepteur du système devra trouver le bon compromis entre le risque d'erreurs et l'augmentation des données.

**Détection des erreurs de transmission**

Le principe général consiste à l'émission à ajouter au bloc de données une somme de contrôle (checksum) obtenue avec un calcul particulier. A la réception, on effectue le même calcul et on compare le résultat au checksum transmis. Il y a au moins une erreur s'ils ne sont pas identiques. Le problème réside dans la compensation des erreurs qui fausse le résultat. Nous allons voir trois systèmes ayant une fiabilité croissante.

***Contrôle de parité***

Celui-ci agit au niveau d'un seul mot, soit 7 bits pour un caractère ASCII. Il consiste à faire la somme bit à bit (sans retenue) de tous les bits du mots, puis d'ajouter un bit dit de "parité"pour avoir une somme totale, soit égale à "0" pour une parité paire, soit égale à "1" pour une parité impaire. Rappel de la table de vérité de l'addition bit à bit :
0 + 0 = 0, 0 + 1 = 1
1 + 0 = 1, 1 + 1 = 0
C'est la table de vérité d'une fonction "XOR" (OU-EXclusif).
Exemple (parité paire) :




### *Contrôle de parité croisé*

Cette méthode est une extension du principe de parité appliqué non seulement sur chaque mot, mais aussi sur l'ensemble des mots, comme montré sur la figure 1.

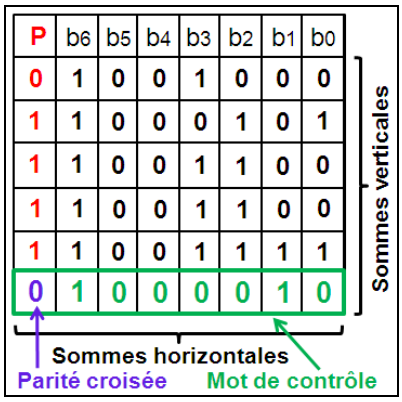



Figure 1 : **Contrôle de parité croisé**

La somme des parités des mots doit être la parité du mot de contrôle qui est la somme bit à bit de tous les mots. Le calcul du bit de parité est standard pour chaque mot. Ensuite on obtient le mot de contrôle en appliquant la fonction XOR sur tout le fichier : ***a*** = XOR(A,B), ***b*** = XOR(***a***,C) … mot de contrôle = XOR(***n***-1,***n***). A la fin, le bit de parité croisée (en bleu) doit correspondre à la parité des parités individuelles (rouge) et à la parité du mot de contrôle (vert).

### *Checksum amélioré*

Une deuxième méthode consiste à additionner non plus les bits, mais les mots en entier, modulo la valeur maxi des mots. On parle de "hachage". Comme le mot résultat ne détecte pas une interversion de mots dans le fichier, on utilise un deuxième hachage pour réduire la probalité de ne pas détecter d'erreur de transmission. Dans ce deuxième cas, on additionne, non plus les mots, mais les mots multipliés par le rang qu'ils occupent. Alors le résultat (checksum) comprendra deux mots. Exemple figure 2 avec 4 octets ASCII pour "Allo".

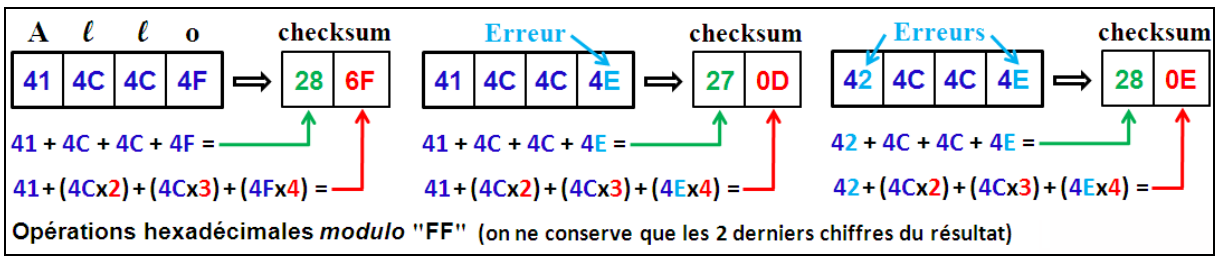



Figure 2 : **Checksum amélioré avec deux exemples d'erreurs**

On voit qu'avec deux erreurs qui se compensent, le premier mot du checksum est identique, mais le deuxième est différent, d'où l'intérêt du système.
Avec des nombres hexadécimaux à deux chiffres, on peut travailler sur des paquets contenant jusqu'à 255 octets.

### *CRC (Contrôle de Redondance Cyclique)*

Le principe général consiste à diviser l'ensemble des bits de données du paquet (le dividende) par un mot de contrôle particulier (le diviseur) appelé "polynôme générateur". Par exemple celui du standard CRC CCITT V41 : "$X^{16} + X^{12} + X^5 + 1$", soit "1000100000010001" en binaire et "8811" en hexadécimal. Le quotient est ignoré et le reste de la division constitue le mot CRC de contrôle qui sera transmis après la fin du fichier. A la réception on effectue la même opération et on compare les CRC qui doivent être identiques si aucune erreur n'est survenue.
Voir sur la figure 3 le principe du processus de la division binaire à l'aide de l'opérateur XOR.

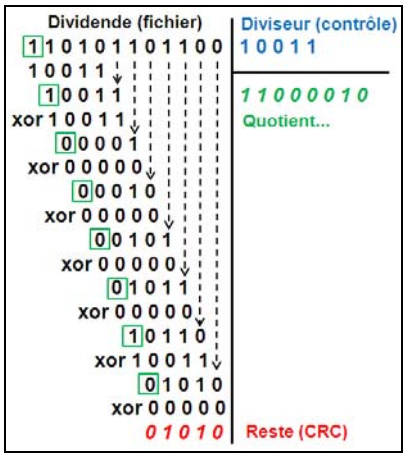



Figure 3 : **Division entre 2 nombres binaires**

Cet algorithme est très facile à implanter dans un logiciel selon l'organigramme de la figure 4.

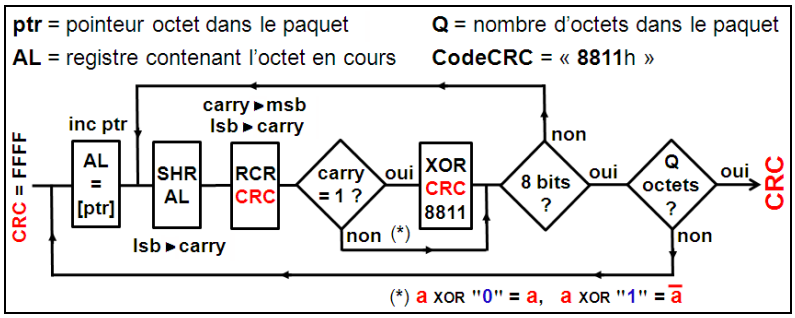



Figure 4 : **Organigramme logiciel pour calculer un CRC**

Il est important de savoir avec certitude si le paquet transmis n'est pas corrompu pour pouvoir, soit l'ignorer, soit demander sa répétition. Mais en cas de canal de propagation perturbé, il sera sans doute plus pertinent de sacrifier quelques octets pour corriger certaines erreurs. Ce sera l'objet du prochain "Comment ça marche".

**La Rubrique "Comment ça marche" est une activité collective du radio-club F6KRK (http://www.f6krk.org). Pour une correspondance technique concernant cette rubrique : "f5nb@orange.fr".**