Fabien DONIUS, Nicolas GRILL, Chérine KAMEL, Selim MILED - Ing1 Gr4

DOSSIER D'ARCHITECTURE

Les codes correcteurs d'erreur

Dossier d'architecture

Sommaire

Appropriation du sujet :

Les codes correcteurs sont un élément indispensable au développement des TIC, mais pourtant méconnu. Ils sont aujourd'hui parfaitement intégrés à ces technologies, par conséquent peu se soucient de savoir ce qu'ils sont, comment ils fonctionnent et dans quel but. Mais face à ce sujet, nous avons dû nous y intéresser, et pour cela plonger profondément dans les théories de l'information.

I. L'utilité des codes correcteurs – But du projet

La première information au sujet des codes correcteurs est contenue dans leur nom. Il s'agit de codes (ou algorithmes) qui, sur une base mathématique, permettent de vérifier l'intégrité des messages transmis, et si besoin est de corriger les éventuelles erreurs qui se seraient glissées dans ce message. En effet tous les canaux de transmission possèdent un BER (Bit Error Ratio) non nul : comprenez par là qu'il y a forcément des pertes ou des altérations du signal électrique/optique/électromagnétique lors de l'envoi d'un terminal à un autre. Ces pertes dépendent du canal : une connexion filaire sur petite distance présentera beaucoup plus de chances de préserver le signal d'origine qu'une transmission sans fil établie sur de très grandes distances. Ainsi la connaissance de l'erreur est nécessaire pour la compréhension et l'établissement de la correction : il est inutile de demander beaucoup de ressources (bande passante ou puissance de calcul) au service d'un algorithme puissant lorsque les probabilités d'erreurs sont faibles, et inversement. C'est cette optique d'optimisation qui régit notre sujet. Ainsi plutôt que de répondre la question « Qu'est-ce qu'un code correcteur », nous cherchons à répondre à la question « Quel code correcteur dans quelle situation ? », tout en pensant secrètement « Quoi de mieux que ces codes correcteurs ? ».

II. Les différents types de codes correcteurs – aspect mathématique du projet

Il existe plusieurs types de codes, dont les fonctionnements sont totalement différents. Des codes détecteurs CRC aux turbo-codes, en passant par les codes de correction à base matricielle, on peut discerner plusieurs familles d'algorithmes reposant sur des principes différents. On peut donc analyser et anticiper grossièrement les performances et défauts de ces grands groupes dans la théorie par une étude purement mathématique d'un ou deux codes de chaque type. Cela évite d'avoir à analyser trop de codes similaires (donc d'avoir à réitérer des analyses relativement identiques, ce qui est dénué d'intérêt), mais également de nous retrouver face à des connaissances mathématiques nous dépassant de très loin. Notre stratégie lors de l'étude mathématique consiste donc en l'analyse des groupes plutôt que des codes individuellement, via des cas particuliers caractéristiques.

III. Les différents codes correcteurs – l'intérêt de notre logiciel

Le stade suivant de notre projet consistera en l'étude d'une plus large variété de codes, individuellement. Comme dit précédemment, l'analyse théorique d'un grand nombre de codes ne présente pas grand intérêt, et surtout risque de très vite dépasser nos compétences actuelles. C'est là qu'intervient notre logiciel. En y intégrant directement des algorithmes poussés existant (Hamming, Reed-Solomon, Xing, Goppa …) puis en les testant dans les mêmes conditions, on pourra avoir directement des résultats de leurs performances, et ainsi déduire lequel d'entre eux est adapté à tel ou tel canal de transmission. Pour cela, il nous faudra étendre notre projet aux erreurs elles-mêmes, et définir quelles altérations interviennent sur différents canaux. Au final, cela se présentera comme un simple choix à l'utilisateur entre plusieurs cas de transmission prédéfinis.

Le tout sera donc au final très facile d'usage : saisie d'un message, choix du type de transmission, des codes à soumettre au test, puis après le test un bilan des performances de chacun.

Ressources

I. Ressources humaines au sein de l'équipe

Si nous sommes tous impliqués dans la totalité du projet, il est évident que la spécialisation de chacun dans un domaine permet de mieux gérer le travail et d'avoir un rendu plus précis. Notre équipe s'organise donc comme suit :

- **Fabien DONIUS : Responsable de la programmation**
  Toute la réalisation du programme se fait sous sa tutelle, et il se chargera de la programmation du corps du logiciel. Ce choix s'est naturellement imposé de par son passé de développeur indépendant et par la qualité de ses travaux précédents dans ce domaine.
  Autres tâches principales : recherche des codes, analyse des CRC, mise en page des documents supports, centralisation des documents sur un serveur

- **Chérine KAMEL : Responsable de l'analyse mathématique**
  Il se charge des études les plus complexes, et choisit parmi ce qu'il aura dû étudier en mathématiques ce qu'il nous faut étudier à notre tour pour l'analyse des codes qu'il nous aura désigné. La raison de ce choix est qu'il est le seul parmi nous à avoir fait une spécialité « mathématiques » en Terminale, et qu'il est celui qui a le plus grand goût pour ce domaine.
  Autres tâches principales : programmation des dll, aide au design du logiciel

- **Selim MILED : Responsable de l'étude des erreurs**
  Il s'occupe de toute la partie concernant les altérations du message selon le canal de transmission. Cela est dû en premier lieu au fait qu'il ait été depuis le début celui qui s'est intéressé à ce sujet, et également à son intérêt pour tout ce qui se rattache au hardware.
  Autres tâches principales : programmation des dll, synchronisation au sein de l'équipe, finalisation de la rédaction des documents supports, analyse mathématique, recherche de contacts intéressants

- **Nicolas GRILL : Responsable de la communication - Débuggeur**
  Il se charge du contact avec les intervenants extérieurs (professeurs, chercheurs, entreprises), que nous avons voulu très présents dans ce projet. Ce choix est dû à son aisance à communiquer et ses qualités de persuasion. De plus il se chargera de tester successivement les différentes versions du logiciel afin d'en corriger les bugs. Cette tâche lui revient de par son goût pour la perfection.
  Autres tâches principales : analyse mathématique, recherche d'informations

II. Ressources humaines extérieures

Afin de ne pas risquer de mauvaises interprétations, informations ou résultats, et également rester les plus professionnels possibles, il nous a semblé essentiel de faire appel à des spécialistes pour nous supporter dans notre projet.

Si la recherche de soutien d'entreprise nous a demandé beaucoup de temps, cela s'est achevé par des échecs dans la plupart des cas (Iliade, France Télécom). Cependant nous gardons encore espoir avec Sagem, dont nous attendons actuellement d'obtenir un rendez-vous avec le service Recherche et Développement en télécommunications, le début de nos relations ayant été prometteur. En revanche, nous avons eu beaucoup plus de chance avec les professeurs et chercheurs, et continuons de privilégier cette source d'informations et de soutien.

Tout d'abord, nous avons contacté le Professeur Nicolas SENDRIER, dont le sujet de thèse était les codes correcteurs, lequel nous a redirigé vers Thomas CAMARA, actuellement en train de finaliser sa thèse sur les codes correcteurs quantiques. Nous avons déjà eu

l'occasion d'échanger plusieurs e-mails, de lui poser de nombreuses questions, et prévoyons de le rencontrer pour les points les plus complexes du sujet sur lesquels nous butterions.

Nous avons également commencé à contacter des chercheurs de l'INRIA dans le domaine des télécommunications et des performances réseaux, et tentons de prendre rendez-vous avec Mr Antoine MERCIER (enseignant en Télécom et Réseaux à l'ECE) pour toute la partie concernant les erreurs selon le canal de transmission. Tout cela a été interrompu par les vacances, et nous comptons reprendre activement dès la rentrée.

Pour la partie mathématique, nous avons demandé l'aide de Mme Fabienne COUDRAY, et passons beaucoup de temps avec Mr Houari MECHKOUR afin d'acquérir les connaissances nécessaires à l'analyse des codes, qui dépassent notre niveau actuel.

Enfin, si nous venons à rencontrer des problèmes lors de la réalisation du logiciel, nous nous tournerons vers nos enseignants en informatique, largement aptes à résoudre les problèmes que nous pourrions rencontrer.

III. Ressources matérielles

Notre projet ne requiert pas d'équipement spécifique : outre une connexion Internet et une ligne téléphonique pour nos contacts, nous n'avons besoin que de bons ordinateurs avec l'environnement logiciel et matériel suffisant pour la réalisation du logiciel.

Par environnement logiciel nous entendons Windows XP en tant qu'OS, et Visual Studio 2005 pour la programmation tant en C qu'en C#, langage que nous utiliserons pour sa capacité à permettre la création d'interfaces graphiques et sa proximité au C étudié en cours.

En ce qui concerne les besoins matériels, nous utiliserons nos propres ordinateurs, largement suffisants pour ce genre de tâches. Il s'agira donc de :

- Intel Core 2 Duo 2,4Ghz ; 2Go de RAM DDR2 ; 740Go de disque dur ; Sparkle Geforce 7950 GX2 1GO de RAM DDR3.

- AMD Athlon X2 64bits 2,2Ghz ; 2Go de RAM DDR2; 250Go de disque dur ; Sapphire Radeon X1800GTO 256Mo de RAM DDR3.

- Athlon 64bits 3500+ 2,2 GHz, 1 Go de DDR, 100 Go de disque dur, Gigabyte GeForce 7600GT 256 Mo

De plus pour pouvoir travailler ensemble sur le même projet, nous mettrons en place un serveur SVN sous Windows Serveur 2003, sur lequel seront stockées toutes nos informations collectées, nos ressources ainsi que les différentes versions de nos codes sources.

- Serveur : Intel Pentium 4 1.7Ghz, 256Mo de RAM, 270Go de disque dur, Connexion 1Mo/s en réception, 128Ko/s en émission, Bande passante illimitée.

IV. Ressources temporelles

Jusqu'en Décembre, notre temps était consacré à la recherche d'informations, aux prises de contacts avec les spécialistes, à l'étude des codes simples, à l'apprentissage mathématique dont nous avions besoin, et aux réflexions générales sur notre projet.

Dès Janvier commenceront les analyses mathématiques les plus poussées d'une part, et le développement du logiciel d'autre part (avec tout d'abord une phase de pré développement indispensable pour définir en détails l'architecture du logiciel). C'est également à ce moment là que nous finirons d'obtenir toutes les informations nécessaires de nos contacts.

Mi-février, les analyses mathématiques devront être achevées, puis commencera leur vérification. La collecte d'information concernant les erreurs sera également achevée. C'est alors que nous débuterons la phase de programmation pure, qui occupera tout le reste de notre temps.

Trois semaines avant la présentation finale du projet, la programmation devra être achevée. Nous vérifierons alors le logiciel dans son intégralité, puis préparerons la présentation finale.

Analyses, traitements, et premiers résultats

I. Introduction à la correction d'erreurs

Tout d'abord, il faut s'intéresser aux théories de l'information. L'idée de codes correcteurs découle directement du deuxième théorème de Shanon : si un canal de transmission génère des erreurs de transmission, il est tout de même possible de trouver une manière de coder les messages émis en leur rajoutant suffisamment de redondance de sorte qu'on puisse retrouver le message émis sans erreur.

Une autre notion importante est celle de distance de Hamming : Si on pose $a = a_0...a_{n-1}$ et $b = b_0...b_{n-1}$, où les a_i et b_i sont des symboles appartenant à un ensemble A, la distance de Hamming se définit formellement par : $d(a,b) = \#\{i : a_i \neq b_i\}$ la notation $\#E$ désignant le cardinal de l'ensemble E. (définition de la Wikipédia.) En effet, la correction d'erreurs consistera parfois à choisir un message correct plutôt qu'un autre parce que sa distance de Hamming est la plus petite des deux.

Les familles de codes évoquées précédemment sont les suivantes :

- Codes à bit de parité
- Codes CRC (Cyclic Redundancy Code)
- Codes de correction à base matricielle ou linéaire
- Codes convolutifs ou codes en blocs
- Turbo-Codes (entrelacement de codes convolutifs)

II. Analyse mathématique

La première analyse, la plus simple à faire, est celle des codes détecteurs d'erreurs, proches cousins des codes correcteurs à proprement parler.

Tout d'abord, les codes à bit de parité : ils consistent à ajouter un bit au message, qui informe de la parité de la somme des autres bits. Leur simplicité entraîne directement leur fragilité. Deux exemples simples pour comprendre leur grande faiblesse :

- Si le bit de parité, et lui seul, est altéré, le message sera considéré comme faux alors que le reste de l'information était en réalité correcte.
- Si un nombre pair de bits sont altérés, alors la parité de la somme sera conservée, et le message faux sera considéré comme correct.

Des versions plus évoluées consistent à faire des sommes « croisées » de plusieurs messages, puis ajouter le bit de parité. Là encore l'altération des bits de parités et la

redondance de la parité toutes les x erreurs sont un problème important : les probabilités sont simplement réduites.

Ensuite, les codes CRC (Cyclic Redundancy Code), reposant sur les polynômes. Chaque bit est considéré comme terme d'un polynôme de degré n, avec n le nombre total de bits considérés (en général, un octet). Par exemple pour un octet 10101010, on aura P(x) = x^8+x^6+x^4+x^2. Il sera ensuite multiplié par un polynôme de même degré, appelé polynôme générateur. Le message est ensuite transmis, puis à la réception divisé par le polynôme générateur. Si le reste de la division est nul, alors il n'y a pas eu d'erreur. Dans le cas contraire, il y a eu erreur et l'octet doit être retransmis. Si ces codes sont très performants à la détection, ils restent limités à cette action et l'information doit être retransmise. La conséquence, nous la connaissons tous via le message d'erreur Windows : « CRC error ». La machine détecte que le message lu est faux, mais elle est incapable de la corriger, et le CD/DVD étant par exemple rayé, il est impossible de récupérer l'information originelle.

Actuellement, nous n'avons pu étudier entièrement qu'un seul code correcteur d'erreur. Ceci est dû principalement à leur complexité mathématique. En effet, ils utilisent des notions que nous n'avons pas encore étudiées en cours.

Le code étudié est le code de Hamming (7,4) qui utilise les matrices et les sous-espaces vectoriels. Le document principal qui nous a servi de source et l'article sur la Wikipédia. L'analyse de ce code a été faite avec l'assistance de notre professeur de mathématique M. MECHKOUR.

Soit X les données à transmettre sous la forme d'une matrice à 4 colonnes et 1 ligne : F = (a,b,c,d)

Le contenu de la transmission correspondra aux données à transmettre X auxquelles ont ajoutera un complément noté T de la forme T = (x,y,z).

Nous obtenons ainsi la transmission M = (a,b,c,d,x,y,z), M étant une matrice de 7 colonnes et 1 ligne.

Nous posons ensuite :

x=b+c+d

y=a+c+d

z=a+b+d

En remplaçant dans la matrice M, nous obtenons :

M=(a,b,c,d,b+c+d,a+c+d,a+b+d)

Pour que cette matrice soit exploitable, nous devons la transposer. Nous obtenons alors :

M = $\begin{pmatrix} a \\ b \\ c \\ d \\ b+c+d \\ a+c+d \\ a+b+d \end{pmatrix}$

Or, M étant les données effectivement transmises, cette matrice correspond également aux données reçues lors de la transmission sans erreurs. Il suffira alors de faire un calcul inverse pour retrouver les données à transmettre.

Ce calcul est :

M = T x C avec M, T, C matrices, T étant une matrice de dimensions (7x4) et C une matrice de (4,1)

Nous obtenons alors grâce aux règles de produit de matrices :

$$\begin{pmatrix} a \\ b \\ c \\ d \\ b+c+d \\ a+c+d \\ a+b+d \end{pmatrix} = \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 0&1&1&1 \\ 0&1&0&1 \\ 1&1&0&1 \end{pmatrix} \times \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

Or, ceci n'est valable que dans le cas idéal d'une transmission sans erreurs.

Dans le cas contraire, c'est-à-dire dans le cas d'une transmission avec erreurs, nous obtenons avec M' matrice des données effectivement reçues :

M' = M + G

Avec M matrice des données envoyées et G une matrice représentant les erreurs de transmission

Or, nous avons M = T x F, d'où M' = T x F + G

Nous définissons une matrice T' telle que T' x T = 0 [2] (les données sont en binaire).

Il est évident que T' n'est pas unique, c'est-à-dire qu'il existe plusieurs matrices T' telles que T' x T = 0 [2].

Nous obtenons ainsi M'= T x F + G, et nous cherchons la matrice G représentant les erreurs

Pour ce faire, nous multiplions l'expression précédente par la matrice T' (connue) :

T' x M' = T' (T x F + G)

Ainsi, nous obtenons en développant l'expression :

T' x M' = T' x T x F + T' x G = T' x G car T' x T = 0.

Nous obtenons : T' x M' = T' x G, d'où M' = G, nous pouvons ainsi déterminer l'emplacement de l'erreur, et la corriger, les données étant transmises en binaire. Nous retrouvons ainsi la matrice de transmission, et donc les données à transmettre.

Si nous avons pu jusque là déterminer et comprendre son fonctionnement, nous n'avons cependant pas encore achevé d'étudier ses limites théoriques.

III. Structure du software

Le programme sera composé de cinq phases :

➔ Message initial : image ou texte au choix. S'il s'agit d'une image un bitmap sera alors sélectionné et les informations d'en-tête du fichier conservées pour éviter les problèmes de lecture de fichier.

➔ Codage du signal avec le code sélectionné. Plugin identique à celui de la correction.

➔ Perturbation du signal : nous utiliserons ici, si possible, un système de plugins qui nous permettra de définir plusieurs types de perturbations possibles (pourcentage, BER, …). Le plugin récupérera les données en entrée et après les avoir perturbées, il les renverra au programme principal. L'avantage d'un système de plugins est que l'on pourra rajouter sans modifier le programme principal des nouvelles perturbations.

➔ Transmission du signal : ici, il ne s'agira uniquement d'une phase d'affichage du message perturbé.

➔ Correction du signal : Tout comme la perturbation, cette partie sera sous forme de plugin pour les mêmes avantages. Ils auront la tache de corriger le message perturbé et de renvoyer le résultat au programme principal.

➔ Affichage du résultat et si possible affichage du taux de correction.

Le résultat de chaque expérience : perturbation x avec correction y, nous permettra alors de dresser un tableau ou graphe des meilleures solutions de correction. Ainsi nous devons mesurer les temps d'exécution de la correction mais aussi la charge en mémoire, l'occupation processeur et la quantité de donnée transmise en plus du signal original. Ainsi nous pourrons voir si la meilleure correction est adaptable à des solutions portables où les ressources matérielles sont bien moins puissantes.

Annexe : notre journal de bord

09/11/2006

➔ Bon alors on a fait une mise en place des choses :

➔ Début d'après-midi : mise au courant de Chérine sur le projet du PSTE. Réflexion sur l'idée générale du projet sur l'année.

➔ Apres chez Selim :

- Création de l'emploi du temps (Planning).

- Réflexion sur les divers contacts qu'on peut avoir pour le projet (relation & obtention de nouveau contact).

- Réflexion sur comment aborder le projet : les algorithmes and comparaison des codes entre eux.

--- Objectif pour demain soir : faire son propre cahier des charges version beta afin d'aborder au mieux le cahier des charges (CDC) qu'on va rendre. ---

10/11/2006

➔ Réunion courte pour récupérer le travail de chacun sur les CDC. Deux seulement en version écrite.

➔ Chérine et Selim se sont retrouvés dans l'après midi afin de mettre ce qui a été fait en commun. Première version du cahier des charges avec les grands traits détaillés.

➔ Réorganisation du CDC par Fabien et adaptation du planning selon les voies vu dans le CDC.

WE du 11&12/11/2006

➔ Travail via MSN pour finir tous les petits détails du CDC.

➔ Mise en page du CDC et du planning.

➔ Version finale ➔ ultime vérification lundi 13/11/2006 avant envoi des deux documents au professeur référent.

Mercredi du 15/11/2006

➔ Envoi d'un mail à Nicolas SENDRIER lui demandant un rendez-vous.

WE du 18&19/11/2006

➔ Préparation du Powerpoint pour la soutenance orale sur le CDC et le PP.

Lundi du 20/11/2006

➔ Préparation de la soutenance avec toute l'équipe.

➔ IDEE : Superposition de deux codes correcteurs.

➔ Récupération des adresses de Sagem et Free (Iliad)

➔ Iliad (24 rue Emile Meunier – 75016 Paris)

➔ Sagem (27 rue Leblanc – 75015 Paris)

➔ Soutenance + Question au professeur référent.

Mardi du 21/11/2006

➔ Travail à l'appart de Selim.

➔ Réception d'une réponse de Nicolas SENDRIER, on nous à redirigé vers une autre personne en fin de thèse sur les codes correcteurs quantiques.

➔ Réflexion sur tout les contacts possibles et organisation de comment les contacté.

➔ Tentative de contact avec SAGEM, premier appel assez prometteur mais aucune nouvelle dans l'après midi.

➔ Recherche du siège Social de Iliad (Free) sans succès. Ils ont déménagé. Mais on a une piste sur une certaine entreprise d'édition appelé Iliade avec un gros chiffre d'affaire faut chercher de ce côté.

➔ Soir : Nico contact ses oncles afin d'obtenir de nouveau contacts.

➔ Soir : Chérine à vérifié notre hypothèse sur la société dans le 8eme on ira faire un saut dès que possible. ☺

Mercredi du 22/11/2006

➔ Rappel de SAGEM on obtient le numéro de tel de la secrétaire du service de recherche et développement de l'ADSL ainsi que son nom.

➔ On laisse tombé le porte à porte avec les entreprise car c'est impossible…

➔ Contacte par mail de M. MECHKOUR (avec des problèmes).

➔ Contacte de l'oncle à Nico (en attente d'une réponse).

Jeudi du 23/11/2006

➔ On recontacte par mail M MECHKOUR, on obtient une réponse des plus satisfaisante en effet, il nous à envoyé beaucoup de doc afin que nous puissions avancer sur la théorie des codes correcteur d'erreurs.

➔ Obtention d'un cours sur les matrices auprès de notre professeur qui nous suit.

➔ Tentative de contacte avec la secrétaire, sans succès ça répond pas…

Vendredi du 24/11/2006

➔ Rassemblement des sources.

WE du 25&26/11/2006

➔ Rédaction du Bilan des informations collectées (BIC).

Pendant la semaine du 27/11/2006 au 01/12/2006

➔ Poursuite de tentative de prise de contact.

➔ Vendredi 01 : Mme BERENGER nous demande de lui envoyer un mail avec un descriptif de notre projet (PPS joint).

Semaine du 04/12/2006 au 08/12/2006

➔ Réponse de Thomas CAMARA nous demandant plus d'information sur le projet ➔ Mail résumant notre projet avec notre cahier des charges en pièce jointe.

Semaines du 11/12/2006 au 22/12/2006

➔ Echange avec Thomas CAMARA.

➔ Réponse de SAGEM.

➔ Tentative de prise de contact avec M MERCIER

➔ Préparation du travail avec M. MECHKOUR.

➔ Compréhension du Code de Hamming linéaire.

➔ Travail personnel de chaque membre sur l'aspect mathématique.

Semaines du 25/12/2006 au 05/01/2007

➔ Rendez-vous avec M. MECHKOUR à Polytechnique pour une explication des matrices, espaces vectoriels appliqués au codes correcteur d'erreur (Code traité : Hamming matriciel). Prise de rendez-vous pour la rentrée, pour traiter d'autres codes.

➔ Rédaction du DA.

➔ Recherche d'information sur les pertes de transmission.

Weekend du 06/01/2007 au 07/01/2007

➔ Avancée de Chérine sur les différents codes autre que Hamming.

➔ Relecture du DA.