

---

# Le RFC 5170 en pratique : conception et évaluation d'un codec AL-FEC LDPC-staircase hautes performances<sup>1</sup>

**Mathieu Cunche et Vincent Roca**

*INRIA, Equipe-Projet Planète, France*  
{mathieu.cunche|vincent.roca}@inria.fr

---

*RÉSUMÉ.* Ce travail s'intéresse à une brique capitale pour de nombreuses applications réseaux : les codes correcteurs pour des canaux à effacement de paquets, aussi appelés AL-FEC ("Application Level Forward Erasure Correction" codes) puisqu'ils opèrent au niveau applicatif. Nous nous focalisons plus précisément sur les codes LDPC-staircase tels que spécifiés dans le RFC 5170. Nous en expliquons les principes (relativement simples) et nous analysons leurs performances, très élevées, que ce soit en terme de capacité de corrections d'effacements ou de vitesse d'encodage/décodage. Nous expliquons en particulier où se trouvent les points durs avant de pouvoir obtenir de tels résultats. Enfin nous positionnons ces codes face à d'autres solutions, en particuliers les codes propriétaires Raptor<sup>2</sup>.

*ABSTRACT.* This work focusses on a building block that is critical for many networking applications: correction codes for the packet erasure channel, also called AL-FEC ("Application Level Forward Erasure Correction" codes) since they work at application level. More specifically, we focuss on LDPC-staircase codes as specified in RFC 5170. We explain their principles (relatively simple) et we analyze their performances, very high, both in terms of erasure recovery capabilities and encoding/decoding speed. We explain in particular what difficulties we faced before achieving such results. Finally, we compare these codes to other solutions, in particular the proprietary Raptor codes.

*MOTS-CLÉS :* Canal à effacement de paquets, AL-FEC, codes LDPC, codes LDPC-staircase

*KEY WORDS:* Packet erasure channel, AL-FEC, LDPC codes, LDPC-staircase codes

---

---

1. Ce travail est réalisé dans le cadre du projet ANR RNRT2006 CAPRI-FEC.

2. Copyright © CNES (« logiciel de codage/décodage Raptor ») : Tous droits réservés

## 1. Introduction : pourquoi la brique AL-FEC est-elle si utile aux applications réseaux ?

Ce travail s'intéresse à une brique capitale pour de nombreuses applications réseaux : les codes correcteurs pour des **canaux à effacement de paquets**, qui opèrent au niveau applicatif (ou AL-FEC, "Application Level Forward Erasure Correction"). En effet, dès que l'on s'élève dans les couches protocolaires, les données reçues sont garanties intègres car vérifiées par les différents checksum et CRC des couches inférieures. En revanche, avec des flux UDP (ou tout protocole de transport non fiable), certains datagrammes peuvent manquer, que ce soit le résultat de congestions, ou d'altérations supérieures aux capacités de correction des codes FEC de la couche physique. Nous avons donc l'équivalent d'un canal à "effacement de paquets".

Ajouter de la redondance au flux transmis, ce que fait le code AL-FEC, est dès lors une approche naturelle pour améliorer la fiabilité globale, sans pour autant avoir besoin de recourir à des techniques de retransmission. Cela minimise le délai de bout en bout (utile pour des flux temps réel) et permet un passage à l'échelle vis à vis du nombre de destinataires, lorsque ces derniers observent des pertes indépendantes (utile pour les applications de diffusion multipoints). L'usage de codes AL-FEC est donc utile pour *robustifier* des flux et, le cas échéant, *passer à l'échelle*.

A cela s'ajoute une "optimisation du service". Considérons un système diffusant des contenus sur de longues périodes à un nombre élevé de clients au moyen d'une diffusion multipoints, sans retransmission (par ex. pour une mise à jour logicielle). Les contenus sont alors placés dans un carrousel qui les transmet en boucle. Un client rejoint le flux à n'importe quel moment, reconstruit les contenus, puis part. Si un client dispose d'une connectivité intermittente et si aucun encodage AL-FEC n'a été utilisé, se pose alors le problème de récupérer le dernier paquet manquant et il faudra potentiellement attendre plusieurs cycles. L'ajout de redondance est donc essentielle afin d'augmenter le choix des paquets dans lesquels puiser pour pouvoir décoder, tout en minimisant la probabilité de recevoir un paquet déjà reçu. L'usage des codes AL-FEC est donc utile pour *optimiser le service de diffusion*, indépendamment de la présence de pertes durant les échanges.

Un autre champ d'application est constitué par les techniques de codage réseau ("network coding") qui ont montré que si des noeuds de transit sont autorisés à combiner différents paquets reçus, alors il est possible d'augmenter la fiabilité globale tout en limitant le trafic échangé. Nous avons là une application des AL-FEC dont la finalité est d'*optimiser le service d'acheminement* au sein d'un réseau fortement maillé.

Enfin, les systèmes de stockage distribué répartissent différentes tranches d'un fichier sur plusieurs serveurs (ou disques avec les systèmes RAID), en ajoutant des tranches de redondance. C'est par exemple la technique utilisée par le système Parhise [PAR]. Récupérer un sous-ensemble de tranches peut être vu comme la transmission de paquets (les tranches) sur un canal à effacements (les tranches qui n'interviennent pas). L'usage des codes AL-FEC permet ici d'*optimiser et robustifier le service de stockage distribué* en apportant de la flexibilité dans le choix des tranches.

L'importance des codes AL-FEC pour des "canaux à effacement de paquets" est donc avérée et ils sont devenus une brique essentielle des systèmes commerciaux tels que ceux du DVB-H/SH/IPI pour la TV mobile personnel et la TV par ADSL<sup>1</sup>. De plus, et de façon générale, puisque les codes AL-

1. Il est à noter que l'on parle aussi, dans le contexte DVB, de codes UL-FEC, pour "Upper Layer FEC".

FEC sont au plus près de l'application, ils peuvent intégrer les contraintes de l'application en faisant varier dynamiquement la taille des blocs codés, ou en jouant sur le niveau de redondance ajouté, ou enfin en proposant des niveaux de protection différenciés en fonction des données manipulées. Ces codes AL-FEC sont largement complémentaires aux codes FEC opérant sur la couche physique qui sont conçus avec d'autres hypothèses, d'autres modèles de canaux, et qui n'ont pas la connaissance à priori des flux d'informations.

**Mais qu'est-ce qu'un bon code AL-FEC ?** Les qualités recherchées sont essentiellement :

**des capacités de correction d'effacement optimales :** si l'objet encodé est constitué de  $k$  symboles sources, alors idéalement la réception de  $n$  importe quel sous ensemble de  $k$  symboles parmi les  $n$  transmis devrait suffire ;

**des vitesses d'encodage et décodage élevées,** ou, ce qui est équivalent, une charge CPU faible, ce qui rend ces codes plus adaptés à des environnements embarqués ;

**des besoins en mémoire de travail limités,** ce qui est intéressant pour les systèmes embarqués ;

Pour certaines applications, en particulier lorsqu'il s'agit de diffuser de gros contenus, les propriétés suivantes peuvent être recherchées :

**codes "grands blocs" :** ces codes peuvent encoder en une seule passe de très gros objets. Ceci est bénéfique puisqu'un symbole de parité pourra récupérer (idéalement) n'importe quel symbole manquant, là où avec un code "petit bloc", le pouvoir correcteur est limité au bloc d'origine.

**codes "small rate" :** ces codes peuvent produire un nombre de symboles de parité très élevé, ce qui est très utile pour les systèmes de diffusion sur de longues périodes, au sein d'un carrousel (voir ci-dessus). Nous retrouvons ici la notion de "fontaine numériques", et dans le cas extrême où le nombre de symboles de parité est infini, on parlera de codes "rateless".

Ainsi Les codes Reed-Solomon sont des codes "idéaux" en terme de correction, en contre partie de quoi ils ont des vitesses d'encodage/décodage plus faibles, et des limites sur la taille des blocs lorsqu'ils opèrent sur  $GF(2^8)$ , puisque  $k < n \leq 255$  symboles. En revanche, les codes LDPC-staircase sont des codes grands blocs, caractérisés par des vitesses d'encodage/décodage très élevées, au prix de capacités de correction légèrement plus faibles. Il en est de même pour les codes Raptor, qui de surcroît sont quasi "rateless", bien qu'en pratique  $n \leq 65536$  [LUB 07].

Il faut être conscient que l'obtention de performances élevées nécessite des codes AL-FEC au potentiel élevé, mais aussi un codec (implantation de ces codes) performant. En effet, les meilleurs codes qui soient ne donneront jamais leur maximum s'ils ne sont pas associés à un codec optimisé, doté de techniques d'encodage et surtout de décodage performantes. Aucun des deux aspects ne doit donc être négligé. A ce sujet, si des standards tels que ceux de l'IETF (par exemple le RFC 5170 qui nous intéresse et dont nous sommes à l'origine) spécifient les codes afin de pouvoir concevoir des systèmes interopérables, rien n'est dit de leur implantation efficace. La moitié du travail reste encore à faire avant de disposer d'une solution optimisée.

Dans ce travail, nous nous focalisons plus précisément sur les codes LDPC-staircase, tels que spécifiés dans le RFC 5170 [ROC 08]. Nous en expliquons les principes et nous montrons leurs potentialités très élevées, que ce soit en terme de capacité de corrections d'effacements ou de vitesse d'encodage et décodage. Nous expliquons en particulier où se trouvent les points durs avant de pouvoir obtenir de tels résultats. Enfin nous les positionnons face aux codes propriétaires Raptor qui sont en position dominante, et nous expliquons les enjeux associés.

## 2. Conception d'un codec LDPC-staircase hautes performances

Nous détaillons maintenant les principes, relativement simples, qui sont à la base des codes LDPC-staircase, puis nous discutons des techniques d'implantation qui nous ont permis d'atteindre des niveaux de performances très élevés.

Dans ce document, un symbole correspondra par exemple à la charge utile d'un datagramme UDP, à savoir l'unité de données transmise qui peut soit arriver de façon intégrée au(x) destinataire(s), soit être perdue. Par abus, nous parlerons également de "paquet".  $k$  est le nombre de symboles source en entrée de l'encodeur,  $n - k$  est le nombre de symboles de parité produits par l'encodeur, et  $n$  est donc le nombre total de symboles, source et parité. Le ratio  $k/n$  est appelé "code rate" et représente la proportion de symboles source dans le flux de données total. Une valeur proche de 1 signifie qu'un faible niveau de redondance est ajouté, et inversement pour les valeurs proches de 0.

### 2.1. Introduction aux codes LDPC-staircase

Les codes LDPC-staircase ont été décrits initialement dans [MAC 03] et de façon plus éloignée dans [DIV 98] sous l'appellation "Repeat Accumulate". Ce sont des codes linéaires qui appartiennent à la classe des codes LDPC ("Low Density Parity Check"), ainsi appelés parce que leur matrice de parité  $H$  est creuse. Cette matrice  $H$  définit en fait un système d'équations linéaires entre les symboles source (ou données) et les symboles de parité (ou redondance), une entrée à 1 dans  $H$  signifiant que le symbole correspondant de la colonne est pris en compte dans l'équation. Plus précisément la matrice  $H$  est constituée de deux sous matrices,  $H_1$  (qui ne fait intervenir que les  $k$  symboles sources) et  $H_2$  (qui ne fait intervenir que les  $n - k$  symboles de parité).  $H_1$  est remplie avec  $N1$  entrées à 1 par colonne (dans ce travail  $N1 = 5$ ), dont la position est choisie aléatoirement.  $H_2$  est dotée pour sa part d'une structure escalier (ou double diagonale). Les codes LDPC-staircase sont des *codes réguliers*, puisque  $H_1$  et  $H_2$  le sont. La principale motivation pour ce choix est de rester le plus éloigné possible de certains brevets connus (voir la section 5 pour une justification).

### 2.2. Un encodage naturellement rapide

Du fait de la structure de  $H_2$ , l'encodage est trivial (vrai pour toute structure triangulaire inférieure). Il suffit de produire le premier symbole de parité, somme XOR de deux symboles sources (au minimum) dont l'identité est définie par les entrées à 1 dans  $H_1$ . Puis on continue avec le deuxième symbole de parité, somme du précédent symbole de parité et d'un certain nombre de symboles source, et ainsi de suite. L'encodage a donc une complexité linéaire.

La principale limite pratique est liée aux accès aléatoires en mémoire, puisque l'on doit ajouter au symbole de parité en cours de construction un symbole source qui peut se trouver n'importe où dans l'objet d'origine. Par défaut, il est souhaitable de disposer d'une quantité de mémoire (RAM) permettant de stocker l'objet dans sa globalité. Si ce n'est pas le cas, il peut être nécessaire d'encoder en plusieurs fois, après avoir découpé l'objet source en  $B$  blocs indépendants, la taille de ces blocs étant choisie compatible avec la mémoire disponible. On y perd alors en capacité de correction mais on y gagne en vitesse d'encodage (et probablement décodage).

### 2.3. Décodage hybride : une clef pour atteindre des performances élevées

Considérons maintenant le décodeur. Il s'agit de résoudre un système d'équations linéaires dont les variables sont les symboles source et parité (même si au final seules les variables sources sont pertinentes). Deux grandes familles de décodage existent :

- les algorithmes de type ITératif, ou **IT**, et
- les algorithmes de type **ML** ("Maximum Likelihood"), par ex. au moyen d'un pivot de Gauss.

L'approche IT est une technique triviale de résolution. L'idée est de chercher une équation faisant intervenir une unique variable inconnue. Cette variable est alors égale au terme constant, somme XOR de toutes les variables connues. On ré-injecte alors cette valeur dans toutes les équations où la variable apparaît et on ré-itére. Cette approche est très rapide mais certains systèmes, pourtant de rang plein, ne peuvent être résolus. D'où l'intérêt de l'approche ML pour achever le décodage. Ici la complexité est significative mais en revanche la capacité de correction est élevée puisque tout système de rang plein est résolu.

Notre approche [CUN 08] consiste à pousser le plus loin possible l'usage du décodage IT, tant que de nouveaux symboles sont disponibles. Si l'approche IT n'a pu aboutir, nous passons alors à un décodage ML, en utilisant le système (généralement simplifié) résultant de la phase IT. Ce décodage hybride offre une *flexibilité idéale*, avec la possibilité d'optimiser soit les capacités de correction, soit la charge CPU et/ou l'autonomie énergétique. Le curseur peut en effet être poussé plus ou moins loin en fonction des capacités de traitement ou de l'autonomie restante de l'hôte. Ainsi un terminal mobile peut décider de ne décoder en ML que les systèmes simplifiés de taille inférieure à un certain seuil, seuil qui peut être dynamique et dépendre également du type d'alimentation (secteur ou batterie).

### 2.4. Techniques d'implantation hautes performances du décodage IT/ML

Intéressons nous à la phase de décodage ML la complexité algorithmique pose question. Pour une matrice aléatoire  $N \times N$ , cette complexité est en  $(N^3)$ . Ce résultat n'est pas applicable ici car la matrice n'est pas quelconque mais largement creuse. De plus le paramètre  $N$  est potentiellement significativement plus faible que  $k$  puisque le système a été simplifié par la phase IT. Enfin les *détails de l'implantation du décodage ML deviennent vite prépondérants* vis à vis des complexités théoriques, et changent du tout au tout les performances observées. Cette section détaille ces aspects pratiques et montre comment atteindre des vitesses de plusieurs centaines de Mbps dans le pire cas.

#### 2.4.1. Décodage ML hautes performances : cas d'un code rate élevé

Considérons tout d'abord le cas usuel d'une utilisation des LDPC-staircase avec un code rate relativement "élevé" (approx.  $\text{code rate} > 2/5$ ). Tout d'abord, la matrice H est simplifiée en supprimant les colonnes correspondants aux symboles source et parité connus et en accumulant leurs valeurs dans le terme constant de chaque équation. Nous utilisons alors plusieurs techniques :

1) la matrice est stockée sous forme dense, chaque ligne étant représentée par un champ de bit, afin de pouvoir XOR'er facilement deux équations lors des opérations d'élimination<sup>2</sup>. Pour accélérer le processus, on peut remarquer que lors de l'étape "i" de l'élimination par la méthode de Gauss, les opérations concerneront des lignes dont les  $i - 1$  premières entrées sont nulles. On ne XOR'e donc que la deuxième partie des lignes, ce qui est d'autant plus intéressant que le processus avance.

2) la représentation en mémoire de la matrice est inversée et débute par la sous-matrice  $H_2$  et se poursuit par  $H_1$ . Ceci permet au processus d'élimination de débiter naturellement par la partie la plus creuse de la matrice, repoussant de la sorte la densification du sous-système restant à inverser.

3) le choix du pivot se fait en identifiant la ligne de plus faible poids parmi les lignes candidates. Ceci permet de minimiser efficacement le remplissage de la matrice lors de l'élimination, et retarde la densification inexorable du système restant. Cependant il faut maintenir le poids de Hamming de chaque ligne, une opération coûteuse malgré l'existence d'algorithmes optimisés. Nous avons donc mis en place des heuristiques qui ne recourent à cette solution que si la densité de matrice est inférieure à un certain seuil. Au delà la recherche du pivot de poids de Hamming minimum devient trop coûteuse et la densité trop importante pour que l'impact soit significatif.

#### 2.4.2. Décodage ML hautes performances : cas d'un petit code rate

Dans le cas particulier d'une utilisation des LDPC-staircase avec un petit code rate (grand nombre de symboles de parité), le décodage à partir de la matrice  $H$  est confronté à deux problèmes :

- 1) la taille de  $H$ , à savoir  $(n - k) \times n$ , avec  $n \gg k$ , devient importante, même après l'étape IT ;
- 2) on reconstruit tous les symboles de parité, alors qu'ils sont en grand nombre ;

Une technique alternative consiste alors à effectuer le décodage ML en partant d'un *sous-ensemble de la matrice génératrice*  $G$ . Cette matrice  $G$  indique tous les symboles sources dont dépend un symbole de parité. Elle est facilement construite grâce à la structure de  $H_2$ , en partant du premier symbole de parité, en remplaçant à chaque fois le précédent symbole de parité d'une ligne de  $H$  par tous les symboles source dont il dépend. De cette matrice  $G$  on ne garde que les équations correspondant aux symboles de parité connus (dont la valeur constitue le terme constant) et les colonnes correspondants aux symboles source manquant (les variables dont on cherche la valeur). Les autres symboles sources (connus) d'une équation sont alors XOR'és dans le terme constant de l'équation.

Procéder de la sorte permet d'avoir un système à résoudre dont la taille ne dépend plus du code rate mais seulement de  $k$  puisqu'il y a au plus  $k$  variables. Le pivot de Gauss se fait alors comme expliqué plus haut, en utilisant une représentation dense ( $G$  est de toutes façons plus dense que  $H$ ).

### 3. Positionnement par rapport aux codes Raptor

Intéressons nous aux codes Raptor, de la société Digital Fountain (rachetée par Qualcomm début 2009). Depuis 2005, ces codes se sont imposés dans différents standards, en particulier : le 3GPP, au sein de l'application FLUTE/ALC du service MBMS, puis le DVB-H, au sein de FLUTE/ALC du

2. Il est intéressant de noter que la phase de décodage IT nécessite en revanche d'utiliser la représentation éparsée de  $H$  afin d'identifier rapidement où se trouve la prochaine entrée à 1 d'une colonne ou d'une ligne. Une conversion est donc faite lorsque lors du passage du décodage IT au décodage ML.

service IP Datacasting, le DVB-IPI pour la fiabilisation des flux TV streamés (en compléments de codes XOR entrelacés), et enfin dans la couche MPE-IFEC du DVB-SH. Ces succès sont en grande partie le résultats de performances très élevées en terme de capacités de correction d'effacements, et dans le cas de l'usage dans FLUTE/ALC, de la possibilité de générer potentiellement un nombre élevé de symboles de parité ("code small rate", section 1). Nous verrons en section 4 que les codes LDPC-staircase dotés d'un décodage IT/ML ont souvent des performances en capacités de correction quasi équivalentes (par ex. quand  $code\ rate > 0.5$  ce qui recouvre nombre de champs d'application).

Considérons les performances en vitesse. S'il est affirmé que les codes Raptor bénéficient de vitesses d'encodage/décodage très élevées, il n'y a pas de document public détaillant ces aspects. Seules des allusions imprécises existent à notre connaissance, telles que [SHO 06] qui mentionne une vitesse de "plusieurs Gbps", sans aucune précision sur la notion de "plusieurs" (1, 10 ou 100 Gbps ?), ni sur des paramètres essentiels que sont la taille de l'objet, des symboles, sachant que chacun peut à lui seul modifier de plusieurs ordres de grandeur cette vitesse.

Après avoir implanté les codes Raptor, nous avons conclu que leur complexité est du même ordre de grandeur que celle des codes LDPC-staircase, voir légèrement supérieure du fait de leur construction. Par contre il est possible que l'usage de la technique décrite dans le brevet [SHO 05] apporte un gain significatif. Cette technique, dans la lignée du décodage hybride IT/ML, consiste à repousser le plus loin possible l'utilisation d'un décodage IT en invalidant des symboles inconnus bloquant le décodeur IT (notons que cette technique est globalement générique, hormis le choix intelligent des symboles à invalider qui dépend des codes Raptor). Cette approche étant propriétaire, nous ne l'avons pas utilisée et nous ne disposons pas d'informations sur son efficacité.

Enfin les codes et codecs Raptor sont *couverts par plusieurs brevets*, là où les codes LDPC-staircase ont été conçus pour être libres. Ces aspects seront discutés plus en détail dans la section 5.

#### 4. Performances

Nous étudions maintenant les performances des codes LDPC-staircase. Nous montrons que ces codes sont proches des codes idéaux, à peine en retrait par rapport aux codes Raptor, tout en ayant des vitesses un ordre de grandeur plus élevées qu'avec des codes Reed-Solomon classiques.

##### 4.1. Conditions expérimentales

Les tests ont été réalisés en utilisant le codec C++ LDPC, version 2.1 [CUN 09], auquel nous avons ajouté le support pour le décodage hybride IT/ML. Le paramètre  $N_1$  est fixé à 5, et nous considérons systématiquement un décodage hybride, IT/ML (section 2.3).

Le principe suivi pour les tests est le suivant : une fois l'encodage réalisé, les symboles source et redondance sont transmis dans un ordre totalement aléatoire. Une probabilité de pertes est alors appliquée, conduisant à l'effacement d'un certain nombre de symboles. L'application réceptrice soumet chaque symbole reçu au décodeur. Cette application s'arrête soit lorsque le décodage est achevé, soit lorsque tous les symboles source ont été reçus (fin de transmission). Le statut (décodage réussi ou échec) est alors déterminé. Il est à noter qu'il n'y a pas de modèle de pertes explicite (par ex.,

aléatoires ou en rafales) du fait du caractère aléatoire des transmissions. La probabilité de pertes (pour un code rate donné) est volontairement le seul paramètre à considérer.

Les comparaisons avec les codes Reed-Solomon sur  $GF(2^8)$  utilisent le codec de L. Rizzo [CUN 09] [RIZ 97] [LAC 09]. Les comparaisons avec les codes Raptor utilisent un codec que nous avons réalisé conjointement avec l'ISAE, conforme aux spécifications du RFC 5053 [LUB 07], et reposant sur un décodage ML (performances optimales en terme de correction d'effacements).

#### 4.2. Performances en termes de capacité de correction

Nous commençons par les tests destinés à apprécier les capacités de correction d'effacements (il est à noter que la phase initiale de décodage IT n'impacte pas ces performances qui sont à mettre entièrement sur le compte du décodage ML).

##### 4.2.1. Ratio d'inefficacité en fonction du code rate et de la taille de l'objet

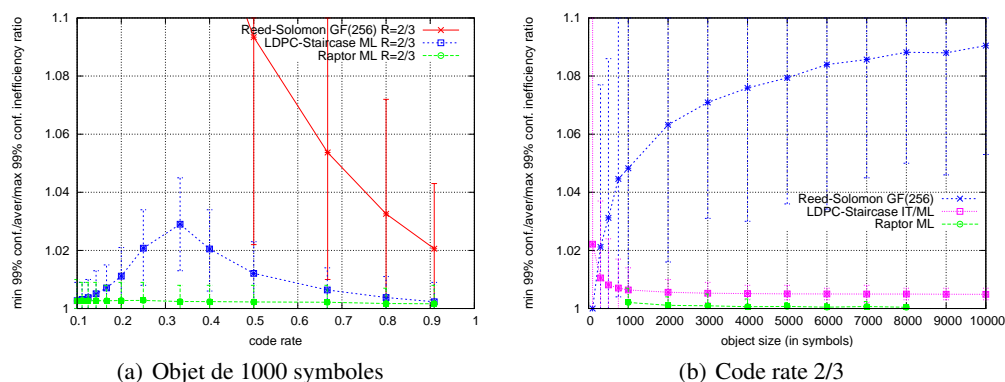


Figure 1 – Ratio d'inefficacité en fonction du code rate et de la taille de l'objet.

Ces performances sont mesurées au moyen de la métrique "**ratio d'inefficacité**" qui représente le ratio du nombre de symboles nécessaires pour que le décodage soit réussi sur  $k$ . Nous considérons également la métrique "**overhead**", égale au ratio d'inefficacité moins 1 (en pourcentage).

La figure 1(a) montre que les codes Reed-Solomon sont rapidement inadaptés lorsque le code rate diminue. À l'inverse, les codes Raptor montrent tous leurs intérêts avec des performances stables quel que soit le code rate effectif (ce n'est pas surprenant vu que rien dans leur construction n'est spécifique au code rate) et un overhead de 0,21%. De même, avec les codes LDPC-staircase, hormis pour des code rates entre 0.2 et 0.5 où l'overhead est moins bon (il monte presque à 3%), il reste sinon nettement sous la barre des 1%. L'utilisateur final ne verra que peu de différences en pratique par rapport aux codes Raptor<sup>3</sup>.

3. Il est à noter qu'utiliser un code rate plus faible que nécessaire pour ensuite n'utiliser qu'un sous ensemble des symboles de parité potentiels (technique de "poinçonnage", ou "puncturing", est une solution qui permet de parer éventuellement aux moindres performances des codes LDPC-staircase dans l'intervalle [0.2; 0.5].

La figure 1(b) souligne également les faibles performances des codes Reed-Solomon dès lors que la taille de l'objet augmente, phénomène lié à la nécessité de découper l'objet en plusieurs petits blocs au lieu d'effectuer un encodage global<sup>4</sup>. Logiquement, les codes "grand blocs" tels LDPC-staircase et Raptor ont d'excellentes performances. Là encore, si les codes Raptor sont meilleurs, le gain ne sera pas décisif pour l'utilisateur final puisque les codes LDPC-staircase convergent très rapidement, sous la barre des 0.6% d'overhead.

#### 4.2.2. Probabilité d'échec du décodage en fonction du taux de perte

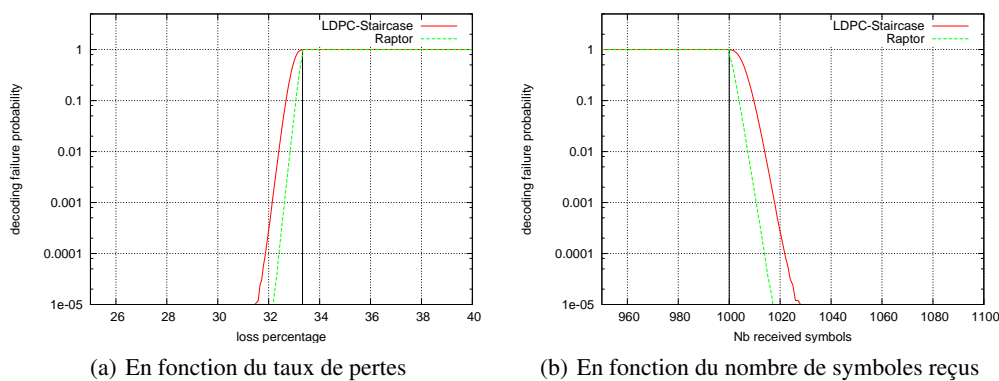


Figure 2 – Probabilité d'échec de décodage ( $k = 1.000$  symboles, code rate  $2/3$ ).

Une autre façon d'analyser les performances de correction consiste à regarder la probabilité d'échec du décodeur lorsque le taux de pertes approche la limite théorique liée au code rate choisi. Ainsi pour un code idéal, un code rate de  $2/3$  permet de décoder parfaitement tant que le taux de pertes est inférieur à 33.3% (probabilité d'échec nulle), et jamais au delà (probabilité à 1). La courbe est donc verticale au niveau du seuil théorique de 33.3%. Une autre façon de présenter ces performances est de donner la probabilité d'échec du décodeur en fonction du nombre de symboles reçus. Elle est à 1 tant que l'on est en deçà de  $k$ , et elle tombe plus ou moins vite au delà. Nous ne montrons pas ici les résultats obtenus avec Reed-Solomon, les performances étant trop mauvaises.

La figure 2(a) montre que les codes LDPC-staircase sont proches de l'idéal, avec une pente raide à l'approche du seuil (plus ou moins verticale en fonction de l'échelle!). Aucun pied de courbe d'inclinaison plus faible n'est visible jusqu'à une probabilité d'échec  $10^{-5}$  ce qui est excellent. La figure 2(b) montre que la réception de quelques symboles supplémentaires seulement suffisent à faire tomber la probabilité d'échec en deçà de  $10^{-5}$ . Là encore les résultats sont très bons.

La table 1 synthétise ces résultats. Afin d'obtenir une probabilité d'échec de décodage inférieure à  $10^{-4}$ , il suffit de recevoir 22 symboles en plus des  $k = 1.000$  avec les codes LDPC-staircase, contre 14 symboles avec Raptor, soit une différence minimale de 8 symboles supplémentaires. Pour le code rate considéré, les différences entre ces codes seront en pratique très limitées, voir invisibles.

4. L'utilisation de codes sur  $GF(2^{16})$  n'est pas réaliste en raison de la complexité déjà élevée sur  $GF(2^8)$ .

	overhead moyen	nombre de symboles et overhead pour une proba d'échec $\leq 10^{-4}$	taux de pertes pour une proba d'échec $\leq 10^{-4}$	proba d'échec effective
Raptor	0,21	1014 symboles, soit overhead 1,4%	32,39%	$0,91 \times 10^{-4}$
LDPC-staircase	0,64	1022 symboles, soit overhead 2,2%	31,86%	$0,90 \times 10^{-4}$

Tableau 1 – Overhead et taux de perte nécessaires à l'obtention d'une probabilité d'échec inférieure à  $10^{-4}$  ( $k = 1.000$  symboles, code rate  $2/3$ ).

### 4.3. Performances en termes de vitesse de décodage

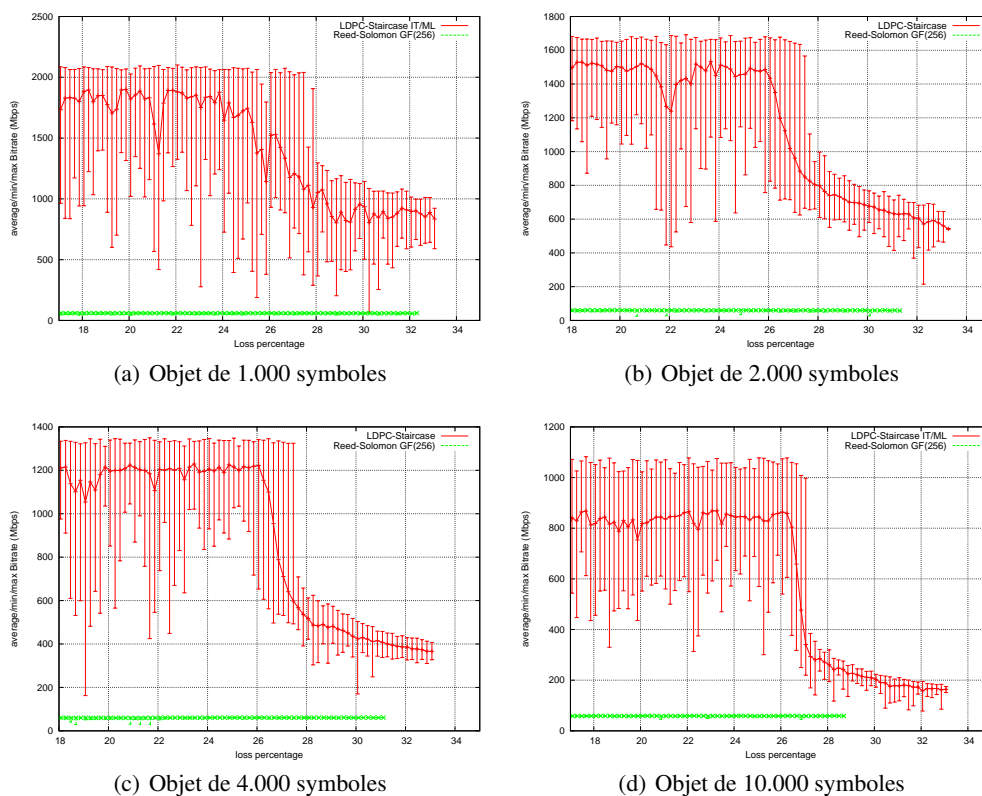


Figure 3 – Vitesse de décodage en fonction de la taille de l'objet (code rate  $2/3$ ).

Nous avons vu que la vitesse d'encodage était par construction même des codes LDPC-staircase très élevée, linéaire avec la taille de l'objet source. Nous nous focalisons donc seulement sur la vitesse de décodage d'un codec disposant d'une approche hybride IT/ML. Aucune comparaison n'est faite avec Raptor puisque nous n'avons pas de codec optimisé en terme de vitesse de décodage. Les tests sont effectués sur un PC sous Linux/Ubuntu 8.10, utilisant un noyau 2.6.27-11/64 bits et un processeur Intel Dual Core Xeon 5120/1.86 GHz(1066 MHz).

Considérons le cas d'un objet de 1 MO, constitué de 1.000 symboles de 1024 octets chaque (figure 3(a)). Nous voyons que pour les taux de perte faibles, le décodage IT suffit et garantit une vitesse de décodage de l'ordre de 1,8 Gbps (30 fois plus rapide que les Reed-Solomon qui restent à 60 Mbps). Ensuite, au fur et à mesure que le taux de perte s'approche de la limite théorique de 33,3%, cette vitesse diminue pour se stabiliser à environ 850 Mbps. Ceci est dû à l'usage de plus en plus fréquent du décodage ML. C'est encore 14 fois plus rapide que les Reed-Solomon. A noter également que la courbe pour les LDPC-staircase s'approche davantage de la limite théorique que celle des Reed-Solomon, attestant là encore de meilleures capacités de correction. Les vitesses de décodage restent confortables avec des objets volumineux. Ainsi avec un objet de 10 MO ( $k = 10.000$ ) la vitesse sera comprise entre 840 Mbps et 162 Mbps, contre 58 Mbps pour Reed-Solomon qui n'est du coup plus que 14,5 à 2,8 fois plus lent. En revanche les capacités de correction des Reed-Solomon sur  $GF(2^8)$  ne sont plus pertinentes pour cette taille d'objet !

## 5. Discussion et conclusions

Concrètement, un développeur ayant besoin de codes correcteur pour un canal à effacement a plusieurs alternatives. Utiliser des codes idéaux de type Reed-Solomon est une solution éprouvée mais qui a ses limites : d'une part du fait de la contrainte "petits blocs", qui induit souvent des capacités de correction sous optimales, d'autre part du fait de la complexité du décodeur, même si des approches alternatives peuvent exister (ainsi les Reed-Solomon/FFT mais qui ne sont cependant pas encore très développés à notre connaissance).

Il peut aussi utiliser des codes triviaux, du type XOR avec un entrelacement plus ou moins long. Ceci reste viable avec certaines applications, en particulier dans le domaine du streaming temps réel (c'est la protection de base du standard DVB-IP). Cependant les cas d'échec sont trop nombreux pour que ce soit une alternative crédible lorsque les taux de pertes sont élevés.

Les codes Raptor sont probablement la solution la plus performante et flexible. Cependant ils sont protégés par plusieurs brevets et une licence d'exploitation est nécessaire. La déclaration d'IPR faite à l'IETF [WAT 06] par Digital Fountain montre qu'en dehors des standards 3GPP/DVB, la société se réserve le droit de ne pas octroyer de licence, et en tous cas pas dans des conditions RAND ("Reasonable and Non Discriminatory"). Le nouveau titulaire, Qualcomm, a priori garantit l'octroi de licences dans des conditions RAND, mais cela peut être remis en cause à tout moment.

Il est de ce fait essentiel de pouvoir disposer de codes AL-FEC hautes performances et libres, ayant fait l'objet d'un processus de standardisation reconnu qui spécifie non seulement les détails du code mais aussi tous les paramètres permettant leur usage dans des applications de type transfert de fichier ou streaming : informations de synchronisation entre encodeur et décodeur, format binaire, et intégration dans les piles protocolaires associées.

Un principe qui a prévalu durant la conception des codes LDPC-staircase est d'être resté le plus éloigné possible des brevets connus afin de garantir autant que possible le caractère libre de la solution. Nos travaux montrent que malgré ces contraintes et leur simplicité, ces codes sont capables, lorsqu'ils sont associés à un codec optimisé, d'atteindre des capacités de corrections proches de l'idéal (moins de 1% d'overhead moyen), et des vitesses d'encodage et décodage très élevées (entre 1,8 Gbps et 850 Mbps pour des objets composés de 1.000 symboles et un code rate 2/3). De ce fait,

dans un grand nombre d'applications où ce type de code rate est suffisant, l'utilisateur final ne verra que peu de différences avec les codes Raptor.

D'autres pistes sont en cours d'étude pour améliorer la situation, en particulier au moyen de codes LDPC dont la structure même est conçue pour diminuer la complexité du décodage ML. Ces travaux pourraient élargir encore plus le champ d'application de ces codes.

## 6. Remerciements

Les auteurs remercient Alexandre Soro et Jérôme Lacan de L'ISAE pour leur participation au développement du logiciel de codage/décodage Raptor.

## 7. Bibliographie

- [CUN 08] CUNCHE M., ROCA V., « Optimizing the Error Recovery Capabilities of LDPC-staircase Codes Featuring a Gaussian Elimination Decoding Scheme », *10th IEEE International Workshop on Signal Processing for Space Communications (SPSC'08), Rhodes Island, Greece*, octobre 2008.
- [CUN 09] CUNCHE M., ROCA V., NEUMANN C., LABOURÉ J., « An Open-Source LDPC Large Block FEC Codec », 2009, URL : <http://planete-bcast.inrialpes.fr/>.
- [DIV 98] DIVSALAR D., JIN H., MCELIECE R. J., « Coding theorems for "turbo-like" codes », *Proceedings Thirty-Sixth Annual Allerton Conference on Communication, Control, and Computing*, 1998.
- [LAC 09] LACAN J., ROCA V., PELTOTALO J., PELTOTALO S., « Reed-Solomon Forward Error Correction (FEC) Schemes », avril 2009, IETF Request for Comments, RFC 5510.
- [LUB 07] LUBY M., SHOKROLLAHI A., WATSON M., STOCKHAMMER T., « Raptor Forward Error Correction Scheme for Object Delivery », octobre 2007, IETF Request for Comments, RFC 5053.
- [MAC 03] MACKAY D., *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, ISBN : 0521642981, 2003.
- [PAR] « Parchive : Parity Archive Volume Set », <http://parchive.sourceforge.net/>.
- [RIZ 97] RIZZO L., « Effective erasure codes for reliable computer communication protocols », *ACM Computer Communication Review*, vol. 27, n° 2, 1997.
- [ROC 08] ROCA V., NEUMANN C., FURODET D., « Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes », juin 2008, IETF Request for Comments, RFC 5170.
- [SHO 05] SHOKROLLAHI A., LASSEN S., KARP R., « Systems and Processes for Decoding Chain Reaction Codes Through Inactivation », février 2005, U.S. Patent Number 6,856,263.
- [SHO 06] SHOKROLLAHI A., « Raptor Codes », *IEEE Trans. Inform. Theory*, vol. 52, n° 6, 2006.
- [WAT 06] WATSON M., « Digital Fountain, Inc.'s statement about IPR claimed in draft-ietf-rmt-bb-fec-raptor-object-04.txt », juin 2006, <https://datatracker.ietf.org/ipr/734/>.