Codes AL-FEC hautes performances pour les canaux à effacements : variations autour des codes LDPC

Mathieu Cunche

UNIVERSITE DE GRENOBLE

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR de l'Université de Grenoble

Spécialité : **Informatique**

préparée à
l'**INRIA Rhône-Alpes, Equipe-Projet PLANETE**

dans le cadre de l'École Doctorale
Mathématiques, Sciences et Technologies de l'Information

présentée et soutenue publiquement par

Mathieu Cunche

le 1er Février 2010

Titre:

Codes AL-FEC hautes performances pour les canaux à effacements : variations autour des codes LDPC

Directeur de thèse: **Dr Vincent Roca**

Jury

M. Pr Pierre Duhamel,	Rapporteur
M. Pr Kave Salamatian,	Rapporteur
M. Pr Ernst W. Biersack,	Membre du jury
M. Pr Jérôme Lacan ,	Membre du jury
M. Dr Vincent Roca,	Directeur de thèse

Résumé

Nous assistons au développement rapide des solutions de diffusion de contenus sur des systèmes, où en plus des traditionnelles corruptions de l'information dans les couches basses, se pose le problème des pertes de paquets d'informations. Le besoin de fiabiliser ces systèmes de transmission a conduit à l'émergence de codes correcteurs d'effacements, qui grâce à l'ajout d'informations redondantes, permettent de reconstruire l'information perdue.

Dans cette thèse nous abordons le problème de la conception de codes à effacements ayant de bonnes capacités de correction et dont les algorithmes de décodage possèdent une complexité permettant d'atteindre des débits élevés. Pour cela, nous avons choisi de travailler conjointement sur les codes et sur leur implémentation au sein d'un codec logiciel, et plus particulièrement sur les algorithmes de décodage.

La première partie de nos travaux montre que des solutions basées sur les codes "Low-Density Parity-Check" (LDPC) permettent d'obtenir d'excellents résultats. En particulier lorsque ces codes sont décodés avec un décodeur hybride Itératif (IT)/"Maximum Likelihood" ou maximum de vraisemblance (ML) qui permet d'obtenir des capacités de corrections proches de l'optimal, tout en conservant une complexité acceptable. De plus, nous montrons que grâce à l'utilisation de codes LDPC structurés la complexité du décodage ML peut être largement réduite.

Nous étudions ensuite le développement de systèmes combinant un code à effacements et des fonctionnalités cryptographiques. Les systèmes résultants permettent de réduire la complexité globale du système tout en garantissant un niveau de sécurité élevé.

Finalement, nous présentons une technique de tolérance aux fautes basée sur des codes correcteurs pour des applications de multiplications matricielles. Cette technique nous permet de construire un système de calcul distribué sur plateforme P2P tolérant efficacement aussi bien les pannes franches que les erreurs malicieuses.

Abstract

Abstract (english version).

Remerciements

Merci aux (re)lecteurs. Merci également à l'ANR qui a financé ma thèse à travers le projet CAPRIFEC.

Liste des contributions

Publications

Mathieu Cunche, Vincent Roca, “Adding Integrity Verification Capabilities to the LDPC-Staircase Erasure Correction Codes”, *IEEE Global Communications Conference (GLOBECOM’09)* Honolulu US, 2009.

Alexandre Soro, **Mathieu Cunche**, Jerome Lacan, Vincent Roca, “Erasure Codes with a Banded Structure for Hybrid Iterative-ML Decoding”, *IEEE Global Communications Conference (GLOBECOM’09)* Honolulu US, 2009.

Mathieu Cunche, Vincent Roca, “Le RFC 5170 en pratique : conception et évaluation d’un codec AL-FEC LDPC-staircase hautes performances”, **Mention spéciale jeune chercheur**, *Colloque Francophone sur l’Ingénierie des Protocoles (CFIP’09)* Strasbourg FR, 2009.

Thomas Roche, **Mathieu Cunche**, Jean-Louis Roch, “Algorithm-based Fault Tolerance Applied to P2P Computing Networks”, *the First International Conference on Advances in P2P Systems (AP2PS’09)* Malta, 2009.

Mathieu Cunche, Valentin Savin, Vincent Roca, Ghassan Kraidy, Alexandre Soro, Jerome Lacan, “Low-rate coding using incremental redundancy for GLDPC codes”, *IEEE International Workshop on Satellite and Space Communications (IWSSC’08)* Toulouse FR, 2008.

Mathieu Cunche, Vincent Roca, “Optimizing the Error Recovery Capabilities of LDPC-staircase Codes Featuring a Gaussian Elimination Decoding Scheme”,*IEEE International Workshop on Signal Processing for Space Communications (SPSC’2008)* Rhodes GR, 2008.

Documents de standardisation IETF

V. Roca, **M. Cunche**, J. Lacan, “LDPC-Staircase Forward Error Correction (FEC) Schemes for FECFRAME”, *IETF RMT Working Group*, draft-roca-fecframe-ldpc-00.txt (Work in Progress), July 2009.

V. Roca, **M. Cunche**, J. Lacan, A. Bouabdallah, K. Matsuzono, “Reed-Solomon Forward Error Correction (FEC) Schemes for FECFRAME”, *IETF RMT Working Group*, draft-roca-fecframe-rs-01.txt (Work in Progress), July 2009.

Rapports de recherche

M. Cunche, V. Roca, “Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme”, Rapport de recherche (2008) 19 [inria-00263682 - version 2]

M. Cunche, V. Roca, “Adding Integrity Verification Capabilities to the LDPC-Staircase Erasure Correction Codes”, Rapport de recherche (2007) 22 [inria-00131002 - version 2]

Logiciels

Codec LDPC hautes performances *http ://planete-bcast.inrialpes.fr* principal contributeur au cours de la période 2007-2009. Plus précisément j'ai développé :

- le décodeur hybride IT/ML, contenant entre autres les optimisations suivantes : décodage ML sur matrice de parité et matrice génératrice, résolution de système avec élimination de Gauss, techniques de réordonnancement de pivots, construction optimisée du système linéaire ;
- le support des codes LDPC-Band et les méthodes de constructions basées sur la représentation polynomiale des matrices (contribution à la hauteur de 50 %) ;
- le support des codes LDPC quasi-cycliques ;
- l'extension du codec pour les codes LDPC-Generalisés basées sur les Reed-Solomon ;
- l'optimisation des procédures de tests d'évaluation de performances pour le décodage ML et le support du multi-threading ;
- ajout des fonctionnalités cryptographiques (VeriFEC) ;

Codec Raptor Développement d'un codec conforme au RFC5053 dans le cadre d'un contrat avec le CNES[1] (contribution à la hauteur de 50 %).

Projet openfec.org Contributeur à la librairie de codecs et de codes à effacements libres de droits (à paraître prochainement).

- participation à la définition de l'architecture de la librairie.

Autres

- encadrement du développement d'un prototype FECFrame, stage de Master Ensimag, A. Albano, 2009.
- évaluation des performances des codes LDPC-staircase pour la brique MPE-IFEC du standard DVB-SH (2006-2007).

Présentations à des groupes

"Maximum Likelihood decoding over the Binary Erasure Channel", **Mathieu Cunche**, *Séminaire PI2C*, Limoges, Février 2009 .

"Coding for loss tolerant systems", **Mathieu Cunche**, *Workshop APRETAF'09*, Grenoble, Janvier 2009.

"Ontologie et codes correcteurs : Des langages tolérants aux pertes", **Mathieu Cunche**, *Séminaire Alanotr II*, Grenoble, Avril 2008.

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

Table des matières

Liste des acronymes

AES ”Advanced Encryption Standard“

ABFT tolérance aux fautes basée sur l’algorithme ou ”Algorithm-based Fault Tolerance“

AL-FEC correction d’erreurs de niveau applicatif ou “Application-Level Forward Error Correction”

ANR Agence Nationale de la Recherche

ARQ requête automatique de répétition ou “Automatic Repeat reQuest“

AWGN canal à bruit blanc gaussien additif ou “Additive White Gaussian Noise”

BCH Bose, Ray-Chaudhuri, Hocquenghem nom des inventeurs du code.

BEC canal binaire à effacements ou “Binary Erasure Channel”

BOINC ”Berkeley Open Infrastructure for Network Computing“

BSC canal binaire symétrique ou “Binary Symmetric Channel”

CAPRIFEC “Design and Analysis of Application-Level Forward Error Correction (AL-FEC) Codes and Application to Mobile Communications”

CEA Commissariat à l’Énergie Atomique

CNES Centre National d’Etudes Spatiales

CRC Contrôle de Redondance Cyclique

DE évolution de densité ou “Density Evolution“

DoS déni de service ou ”Denial of Service“

DVB-H “Digital Video Broadcasting - Handheld”, en francais Diffusion Vidéo Numérique - Portable

DVB-SH “Digital Video Broadcasting - Satellite Handheld”, en français Diffusion Vidéo Numérique - Satellite Portable

FEC correction d’erreurs ou “Forward Error Correction”

FFT transformée de Fourier rapide ou ”Fast Fourier Transform“

FLUTE “File Delivery over Unidirectional Transport”

GE élimination de Gauss ou “Gaussian Elimination”

GLDPC “Generalized-LDPC”

GNU ”Gnu’s Not Unix“

GPGPU ”General-purpose computing on Graphics Processing Units“

GPL ”General Public License“

HARQ requête automatique de répétition hybride ou ”Hybrid Automatic Repeat reQuest“

HPC calcul hautes performances ou ”High Performance Computing“

IETF ”Internet Engineering Task Force“

INRIA Institut National de Recherche en Informatique et Automatique

ISAE Institut Supérieur de l’Aéronautique et de l’Espace

IT Itératif

LDGM “Low-Density Generator Matrix”

LDPC “Low-Density Parity-Check”

LETI Laboratoire d’Électronique et de Technologies de l’Information

LT “Luby Transform”

LWC mot de code de poids faible ou “Low Weight Codeword”

MDS Maximum Distance Séparable

ML “Maximum Likelihood” ou maximum de vraisemblance

P2P pair-à-pair ou ”Peer-to-Peer“

PAR Parchive

PGCD plus grand commun diviseur

PLANETE Protocoles et applications pour l’Internet

PND probabilité de non détection

PRNG générateur de nombres pseudo-aléatoires “PseudoRandom Number Generator”

OSI “Open Systems Interconnection”

QC Quasi-Cyclique

RA “Repeat-Accumulate”

RAID matrice redondante de disques bon marché ou ”Redundant Array of Inexpensive Disks“

RFC “Requests For Comments”

RS Reed-Solomon

SUMMA ”Scalable Universal Matrix Multiplication Algorithm“

TCP “Transmission Control Protocol”

UDP “User Datagram Protocol”

XOR *ou* exclusif ou “eXclusive OR”

Glossaire

Symbole : un symbole est un élement d'un alphabet, le plus souvent un corps fini. Il correspond à unité de données sur laquelle opère le code. Dans la couche physique un symbole correspond souvent à un bit, alors que sur la couche application un symbole correspond à un ensemble de bits (par exemple un datagramme IP, un datagramme UDP, un morceau de fichier) ;

Code bloc : un code bloc est un code correcteur prenant en entrée un bloc de données composé de k symboles, et produisant un ensemble de n symboles codés. k et n sont appelés respectivement la dimension et la longueur du code ;

Mot de code : un mot de code est un élément du code, i.e. c'est un élément du sous espace vectoriel de dimension k de l'epace vectoriel $\mathbb{F}_q^n$. Il correspond à une chaîne de n symboles ;

Matrice génératrice : la matrice génératrice est la matrice de l'application d'encodage ;

Matrice de parité : une matrice de parité est la matrice de l'application dont le noyaux est formé par les éléments du code ;

Code FEC : les codes "Forward Error Correction" (FEC) ajoutent de la redondance aux données sources afin de corriger à posteriori les erreurs apparues durant la transmission ;

Code AL-FEC : les codes "Application-Level FEC" sont des codes FEC qui opèrent au sein de la couche transport ou application. Ils sont également appelés codes UL-FEC (UL pour "Upper-Layer").

Symbole source : un symbole source est un symbole appartenant aux données sources ;

Symbole de redondance : un symbole de redondance (parfois appelé symbole de parité dans le cas des codes binaires) est un symbole généré par le code, contenant de l'information permettant de corriger les erreurs ;

Symbole codé : un symbole codé est un symbole résultant de l'encodage. Il peut être soit un symbole source (uniquement dans le cas d'un code systèmatique), soit un symbole de redondance ;

Bloc source : un bloc source est un ensemble contiguës de symboles sources considérés globalement lors d'un encodage ;

Codec : un codec est une implémentation logicielle ou matérielle de l'encodeur et du décodeur ;

Codes MDS : les codes MDS sont des codes "Maximum Distance Separable". Ce sont des "codes idéaux", i.e. ils sont capables de décoder à partir de k symboles quelconques reçus parmi les n symboles codés, où k et n sont respectivement la dimension et la longueur du code ;

Codes sans rendement : les codes sans rendement, ou code "rateless", sont des codes capables de produire un nombre de symboles de redondance potentiellement infini ;

Objet : un objet est une séquence de bits devant être transmis ;

Décodage IT : le décodage itératif (IT) est un type de décodage sous optimal mais à complexité réduite ;

Décodage ML : le décodage par maximum de vraisemblance (ML) est un décodage optimal dans le sens qu'il exploite au mieux les capacités de correction du code correcteur ;

Codes LDPC : les codes LDPC sont une famille de codes basées sur des matrices de parité creuses ;

Codes Raptor : les codes Raptor sont une famille de code sans rendement ;

Codes Reed-Solomon : les codes Reed-Solomon sont famille de codes MDS ;

Fonction de hachage : une fonction de hachage est une fonction calculant, à partir d'une donnée de taille arbitraire, une empreinte le longueur finie permettant d'identifier les données d'entrée ;

Fonction de chiffrement : une fonction de chiffrement est une fonction effectuant une transformation sur des données, à partir d'une clef, rendant ces données inintelligibles à toute personne ne possédant pas la clef ;

Chapitre 1

Introduction

Le développement de la théorie de l'information par Claude Shannon en 1948 a donné naissance à la théorie des codes. Poussés par les avancées scientifiques et technologiques d'un côté, et les besoins industriels de l'autre, les codes n'ont eu de cesse de progresser. Lorsqu'ils sont utilisés pour corriger des erreurs intervenant sur des informations codées sous forme numérique, ces codes sont appelés codes correcteurs d'erreurs.

Les codes correcteurs sont largement utilisés dans le domaine des télécommunications, où ils permettent des transmissions fiables sur des médiums de communication bruités, comme les canaux sans fil. On les retrouve également dans le domaine du stockage, pour protéger l'information enregistrée face à la détérioration du support.

Si la correction des erreurs est généralement effectuée dans les couches basses des systèmes de télécommunications, afin de corriger la corruption de l'information, les codes correcteurs sont également capables de corriger les effacements. Ces effacements, qui correspondent à des pertes d'informations, sont des phénomènes de plus en plus courants qui se retrouveront généralement dans les "couches hautes" des systèmes de télécommunications. Lorsque les codes sont appliqués à la correction d'effacements, on les appelle codes correcteurs d'effacements ou encore codes à effacements. C'est autour des codes à effacements que cette thèse s'articule.

Motivations et objectifs des travaux

Choix relatifs aux codes et principes directeurs de nos travaux : Parmi l'ensemble des codes à effacements, deux familles se détachent :

- Les codes du type Reed-Solomon qui constituent un ensemble de codes optimaux en termes de capacité de correction, mais limités par leur complexité d'encodage/décodage cubique.
- Les codes LDPC , dont les performances exceptionnelles ont été mises en évidence par des travaux récents. L'avantage majeur de ces codes est qu'ils sont décodables efficacement par un algorithme à complexité linéaire appelé décodage itératif.

Au vu du potentiel des codes LDPC, nous avons choisi de focaliser nos travaux sur cette famille de codes.

S'il est possible en théorie de construire des codes parfaits, c'est à dire des codes possédant une capacité de correction optimale, leur utilisation est souvent impossible en pratique. En effet les algorithmes de décodage associés possèdent une complexité prohibitive pour la plupart des applications. Ainsi, lors de la conception d'un système de

correction d'erreurs, la construction du code n'est pas suffisante pour garantir une solution globalement performante et il est souvent nécessaire de prendre en compte l'efficacité des algorithmes d'encodage et de décodage, c'est à dire son implantation. L'implantation des fonctions d'encodage et de décodage sera appelée codec. Lorsque l'on conçoit une solution de fiabilisation basée sur les codes correcteurs, il est donc impératif de prendre en compte à la fois le code et le codec associé. **Cette approche de conception conjointe du code et du codec a été systématiquement adoptée dans cette thèse et en est un aspect discriminant par rapport à d'autres travaux du domaine de la théorie des codes.**

Une grande majorité des travaux concernant les codes LDPC sur le canal à effacements présente des résultats théoriques sur les capacités de corrections et en particulier sur la complexité théorique des algorithmes d'encodage et de décodage. Dans cette thèse nous avons mis l'accent sur l'aspect pratique de l'utilisation des codes à effacements. **Nous insisterons tout particulièrement sur l'évaluation de la complexité et des débits atteignables en pratique par ces codes et codecs.**

Les travaux concernant les codes LDPC présentent souvent des résultats sur leurs performances asymptotiques (des codes de longueur infinie). Les cas pratiques d'utilisation, telle que la protection de flux de paquets, nécessitent des codes de longueur moyenne et courte (quelques milliers de symboles) pour lesquels les résultats asymptotiques ne sont plus applicables. **Ainsi les travaux présentés dans cette thèse se focaliseront sur les performances des codes delo ngueur courte et moyenne.**

Si le décodage itératif des codes LDPC permet d'atteindre asymptotiquement la capacité du canal, ses performances sur des codes de longueur finie en sont loin.En effet ce décodage n'exploite pas totalement la capacité de correction du code, ce qui n'est pas le cas du décodage par ML. Le décodage ML permet d'exploiter au maximum les capacités de correction du code au prix d'une complexité accrue. Il a été montré que l'utilisation d'un décodeur ML permet une amélioration significative des capacités de corrections des codes LDPC. L'augmentation de complexité peut être justifiée par l'amélioration des capacités de correction et n'être que d'importance limitée, en particulier pour des codes de longueur modérée. Le décodage ML des codes LDPC de longueur courte, et en particulier ses aspects pratiques, n'ayant pas encore été sujet d'une étude, il nous a paru important d'identifier les gains et les limites apportés par cette technique.

Champs d'applications : Les terminaux mobiles ont tendance à se démocratiser et à embarquer de plus en plus d'applications multimédia. Parmi celles-ci on pourra citer la diffusion de contenus multimédia à grande échelle par voie hertzienne, comme par exemple le système DVB-H/DVB-SH. Ces terminaux apportent deux types de contraintes :

- Premièrement leur mobilité et la faible taille de leurs antennes font que la qualité du signal reçu varie vite et peut devenir très mauvaise. Les codes à effacements constituent alors une technique efficace pour améliorer la qualité de service.
- Deuxièmement ils possèdent des ressources limitées en calcul, mémoire et énergie. Il faudra être d'autant plus vigilant quant à la complexité des algorithmes utilisés afin d'économiser ces ressources.

Ce type d'application constituent une cible privilégiée pour les codes à effacements à complexité modérée.

Étant donné que les codes correcteurs à effacements travaillent souvent au niveau applicatif, il est naturel de considérer une implémentation logicielle plutôt que matérielle de ceux-ci. Contrairement à une implémentation matérielle, une implémentation logicielle

permet de modifier à la volée les paramètres du code. Ceci apporte une grande flexibilité au système qui pourra facilement s'adapter aux conditions de transmission. En utilisant les ressources du système d'exploitation, l'implémentation logicielle permet d'utiliser la mémoire du terminal qui est souvent moins onéreuse et plus grande que celle disponible dans les chipsets. La taille de la mémoire est un critère d'autant plus important que les objets encodés par les codes de niveau applicatif ont une taille conséquente. Ainsi, même si une implémentation matérielle des solutions présentées dans cette thèse est possible, nous nous sommes concentrés sur l'évaluation de l'implémentation logicielle.

Solutions libres : Le domaine des codes à effacements a connu une activité particulièrement intense depuis quelques années, et certaines productions scientifiques et technologiques ont été largement brevetées. Il en est ainsi des codes Raptor qui sont souvent considérés comme la solution la plus performante pour transmettre sur le canal à effacements. Beaucoup de détails relatifs à l'usage des codes à effacements (ce qui va bien au delà des codes Raptor) sont également couverts par ces brevets. Nous pensons qu'il est nécessaire de développer des alternatives libres de droit à ces solutions. **Ainsi, la conception de solutions libres de droit a été une des motivations des travaux présentés dans cette thèse et a guidé certains choix.**

Contexte de la thèse

Cette thèse s'est déroulée au sein de l'équipe PLANETE de l'INRIA Rhône-Alpes. Elle a été financée par le projet ANR CAPRIFEC dont l'objectif est l'analyse et la conception de codes correcteurs d'erreurs de niveau applicatif pour les applications mobiles. Ce projet implique trois partenaires académiques : Institut National de Recherche en Informatique et Automatique (INRIA), ISAE et le CEA-LETI, ainsi que deux partenaires industriels : ST-MicroElectronics et Eutelsat.

Plan du manuscrit

En plus des chapitres d'introduction et conclusions, ce manuscrit est composé de dix chapitres organisés en quatre parties.

Une première partie présente les pré-requis à la compréhension de cette thèse. Elle est composée de deux chapitres :

- Dans un premier temps nous introduisons les codes correcteurs à effacements sous la forme d'un état de l'art. Une présentation du concept de canal à effacements, puis un passage en revue des principaux codes à effacements et de leurs algorithmes d'encodage et de décodage sont effectués (chapitre 2).
- Puis, la méthodologie utilisée dans cette thèse afin d'évaluer les performances des solutions considérées est présentée sous la forme de métriques de performances et d'outils associés (chapitre 3).

La seconde partie est constituée de propositions de systèmes de codage pour le canal à effacements :

- Une technique de décodage des codes LDPC, appelée décodage hybride IT/ML, est présentée avec les détails d'implémentation et un ensemble d'optimisations possibles (chapitre 4).
- Nous montrons ensuite l'adaptation des codes LDPC-staircase pour le décodage hybride IT/ML (chapitre 5).
- Puis, dans l'objectif de réduire la complexité du décodage ML, nous présentons un nouveau type de code LDPC à matrice structurée, les "LDPC-Band" (chapitre 6).
- En conservant l'idée de matrice structurée pour réduire la complexité du décodage ML, nous présentons des résultats préliminaires obtenus avec les codes LDPC Quasi-Cycliques dont les matrices possèdent une structure par bloc (chapitre 7).
- Finalement une construction de codes LDPC généralisés, permettant d'atteindre des rendements très faibles de façon incrémentale est présentée (chapitre 8). .

La troisième partie présente des résultats relatifs à l'utilisation conjointe des codes à effacements et des briques cryptographiques :

- Premièrement nous présentons un système intégrant codage à effacements et vérification d'intégrité, capable de détecter la majorité des corruptions avec un surcoût très faible (chapitre 9).
- Deuxièmement, une utilisation conjointe des codes Reed-Solomon avec une fonction de chiffrement, permettant de réduire la quantité de données devant être chiffrée, est proposée (chapitre 10).

La quatrième partie présente un exemple d'application des codes correcteurs : nous montrons comment les codes correcteurs permettent de construire un système de calcul distribué pair-à-pair tolérant aux fautes (chapitre 11).

Nous terminons cette thèse par un résumé des contributions, les conclusions que l'on peut en tirer, ainsi que les perspectives pour la suite des travaux (chapitre 12).

Chapitre 2

État de l'art des codes correcteurs pour le canal à effacements

Contents

Les codes correcteurs d'erreurs sont des outils mathématiques permettant la correction de différents types d'erreurs pouvant affecter des données numériques. Ces codes sont en particulier capable de corriger des pertes d'informations aussi appelées effacements. Dans ce cas on parlera de codes correcteurs pour le canal à effacements ou encore de codes à effacements.

Dans ce chapitre nous présentons les notions nécessaires à la compréhension de cette thèse. En particulier nous introduisons les concepts relatifs au canal à effacements et aux codes correcteurs. Nous fournissons également une présentation de l'ensemble de codes correcteurs auxquels il sera fait référence dans la suite du manuscrit.

2.1 Quelques notions de théorie de l'information et concept de canal de transmission

2.1.1 Théorie de l'information

En 1948 Claude E. Shannon publia un article [102] qui donna naissance au domaine de la théorie de l'information. Shannon démontra que pour n'importe quel canal, il était possible de transmettre sur celui-ci avec un taux d'erreur arbitrairement proche de zéro tant que le taux de codage était inférieur à un seuil appelé capacité du canal. Malheureusement cette preuve n'est pas constructive, et la conception de codes performants en termes de correction d'erreurs et de complexité est toujours un problème d'actualité. Même si les travaux de Shannon étaient motivés par des problématiques venant du domaine des télécommunications, ses travaux eurent des répercussions dans de nombreux autres domaines tels que l'économie, les statistiques ou la physique.

Canal de communication : Un canal de communication peut être modélisé comme une entrée à valeurs dans un alphabet A, une sortie à valeurs dans un alphabet B, ainsi qu'une loi de transition qui détermine la probabilité d'obtenir Y à la sortie du canal sachant que l'entrée était X. Cette probabilité est notée : $P(Y|X)$. La connaissance du canal nous donne ainsi la loi de transition définie par la probabilité de chaque couple de valeurs d'entrée et de sortie, Y et X :

$$\forall (X,Y) \in A \times B, P(Y|X) = P_{Y,X}$$

Fonction d'entropie : La fonction d'entropie de X, notée $H(X)$, mesure l'incertitude sur la variable X à valeur dans $\{x_i\}_{i=0..n}$ et elle s'exprime de la manière suivante :

$$H(X) = -\sum_{i=0}^{n} P(X = x_i) \log_2 P(X = x_i) \tag{2.1}$$

L'entropie d'une source est en fait égale au nombre minimal de bits nécessaires pour la coder.

Information mutuelle : Le but du récepteur est de déduire quel X a été transmis connaissant la valeur Y qu'il a reçue. On considère alors l'information mutuelle des variables X et Y, noté $I(X,Y)$ qui représente l'information que l'on peut obtenir sur X a partir de Y. Cette quantité peut être exprimée sous la forme suivante :

$$I(X,Y) = H(X) - H(X|Y) \tag{2.2}$$

$H(X|Y)$ étant le niveau d'incertitude sur la variable X sachant Y.

Dans le cas d'un canal parfait qui transmet l'information sans erreur, on a $H(X|Y) = 0$. En effet Y est totalement déterminée par la valeur de X, il n'y a donc aucune incertitude

sur cette première. Ainsi l'information mutuelle des deux variables est maximum et $I(X,Y) = H(X)$. Dans le pire cas, le canal est tellement mauvais que les variables X et Y sont totalement indépendantes et on a $H(X|Y) = H(X)$ et donc l'information mutuelle est minimale $I(X,Y) = 0$. Dans ce dernier cas l'entrée du canal n'a aucune influence sur la sortie du canal, c'est le pire cas possible, et il est impossible de transmettre une information sur ce canal.

Capacité d'un canal On peut maintenant définir la capacité C du canal qui vaut :

$$C = max_X\{I(X,Y)\} \tag{2.3}$$

max_X étant le maximum sur toute les lois de probabilité possibles, i.e. sur toutes les loi de probabilité définies sur A. Ainsi, dans le cas du canal parfait la capacité est maximum, $C = max_X\{H(X)\}$, alors que dans le cas du pire canal la capacité est nulle, $C = max_X\{0\} = 0$.

2.1.2 Exemple du canal binaire symétrique

Appliquons les notions précédemment introduites au modèle de canal le plus simple : le canal binaire symétrique ou "Binary Symmetric Channel" (BSC). Comme son nom l'indique ce canal considère des entrées et des sorties à valeur dans $\{0,1\}$. Il est caractérisé par un paramètre p qui correspond à la probabilité d'erreur lors de la transmission d'un bit sur ce canal (voir figure 2.1). Sa loi de transition est symétrique ce qui signifie que les probabilités d'erreurs sur les bits 0 et 1 sont identiques :

$$\begin{aligned} P(Y=0|X=0) &= 1-p \\ P(Y=0|X=1) &= p \\ P(Y=1|X=1) &= 1-p \\ P(Y=1|X=0) &= p \end{aligned}$$

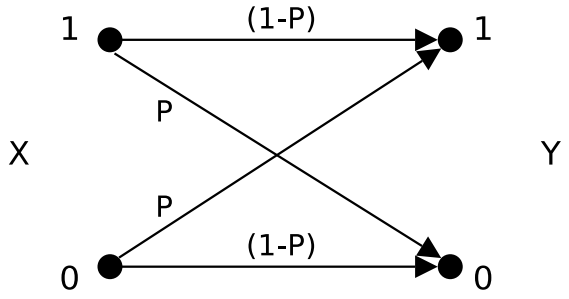


FIG. 2.1 – Représentation du canal binaire symétrique

Le maximum de la fonction d'information mutuelle est atteint pour une variable aléatoire X tel que $H(X) = 1$. Ce maximum est atteint lorsque X suit une loi de probabilité uniforme. On obtient alors :

$$C_{\text{BSC}} = max_X\{I(X,Y)\} = 1 - H(p) \tag{2.4}$$

La capacité du canal binaire symétrique est donc égale à 1 si la probabilité d'erreur est nulle. Comme la fonction d'entropie $H(.)$ atteint en 0.5 son maximum, 1 , la capacité d'un canal binaire symétrique de paramètre 0.5 est nulle. C'est le plus mauvais canal binaire symétrique possible.

2.2 Le cas des canaux à effacements

2.2.1 Le canal binaire à effacements

C'est en 1955 que Peter Elias [39] présenta le modèle du canal binaire à effacements ou "Binary Erasure Channel" (BEC). Les erreurs qui interviennent sur ce type de canal sont des effacements d'information. Contrairement au canal binaire symétrique, l'information transmise sur le canal n'est pas altérée, mais une partie de celle-ci est tout simplement perdue.

Sur un canal à effacements de paramètre p, la probabilité qu'un symbole transmis soit effacé est égale à p. On modélise souvent ce canal en ajoutant à l'ensemble des valeurs que peut prendre la sortie du canal, un symbole E représentant l'effacement (voir figure 2.2). La loi de transition du canal binaire à effacements est :

$$\begin{aligned} P(Y=0|X=0) &= 1-p \\ P(Y=1|X=0) &= 0 \\ P(Y=1|X=1) &= 1-p \\ P(Y=0|X=1) &= 0 \\ P(Y=E|X=0) &= p \\ P(Y=E|X=1) &= p \end{aligned}$$

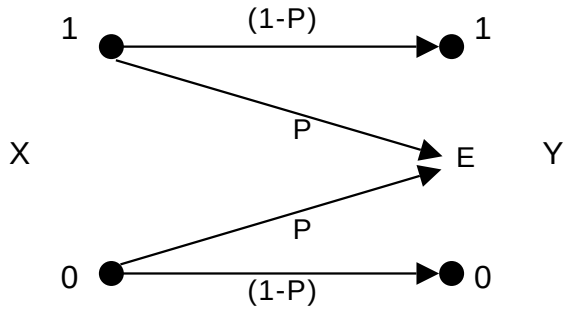


FIG. 2.2 – Représentation du canal binaire à effacements

La capacité du canal à effacements peut s'exprimer comme :

$$\begin{aligned} C_{\text{BEC}} &= max_X((1-p)H(X)) \\ &= (1-p) \end{aligned}$$

On remarque que pour une probabilité d'erreur donnée la capacité du canal binaire à effacements est supérieure à celle du canal binaire symétrique égale à $(1-H(p))$, puisque $\forall\, p \in [0,1], p < H(p)$. En effet sur le BEC l'emplacement des effacements est connu, et il ne reste plus qu'à les corriger, alors que sur le canal binaire symétrique la position des erreurs est inconnue, ce qui rend la tâche de correction plus difficile.

2.2.2 Le canal à effacements de paquets

Dans la plupart des applications, les effacements ne concernent pas des bits isolés, mais plutôt des groupes de bits, pouvant aller de quelques octets à plusieurs Giga-octets. On ne peut donc plus parler de canal binaire à effacement, et il convient alors d'introduire un modèle de canal plus général, le canal à effacements de paquets dans lequel les symboles sont des mots de plusieurs bits. Sur ce canal, soit les paquets sont transmis sans altération, soit ils sont totalement effacés. La plupart du temps l'intégrité du paquet est garantie par un système de détection d'erreur sur l'ensemble de celui-ci (CRC, checksum TCP ou UDP, fonction de hachage).

Ce modèle permet de décrire le comportement d'un large nombre de systèmes dans lesquels de l'information peut être sujette à effacements. Les effacements de paquets ont deux sources principales : soit l'information a tout simplement disparu, soit l'information est présente mais elle est trop altérée pour être utilisable et on préfère l'ignorer. La section suivante présente des exemples d'applications dans lesquelles on rencontre des canaux à effacements de paquets.

2.2.3 Exemples pratiques de canaux à effacements

Les canaux à effacements de paquets se retrouvent dans un grand nombre d'applications pratiques. Cette section présente les exemples les plus courants d'applications qui ont motivé l'usage des codes à effacements.

Exemple 1 : Réseaux de communication

Les systèmes de communications modernes suivent une organisation en couches (modèle OSI). Chaque couche possède une interface permettant de délivrer un service aux couches supérieures en utilisant un protocole déterminé. Au sein de ces couches, les données sont regroupées, traitées et transmises par paquets de bits. Lors de leur transmission ces paquets peuvent être perdus ou altérés :

- on dit d'un paquet qu'il est *perdu* lorsqu'il n'arrive pas à destination, soit à cause d'erreurs de routage sur le réseau, soit le plus souvent à cause d'un encombrement d'un noeud du système ("drop" de paquets dans un routeur avec une file d'attente pleine),
- on parle d'*altération* de paquet lorsqu'un ou plusieurs bits subissent une modification inattendue. Les altérations proviennent le plus souvent de perturbations du signal transmis sur le médium de communication. Un code correcteur de la couche physique peut tenter de corriger ces altérations. En cas d'échec du décodage, le paquet est considéré en erreur [52].

Certaines couches possèdent une fonctionnalité de vérification d'intégrité du paquet. La vérification d'intégrité consiste à utiliser de la redondance ajoutée aux données utiles, afin de détecter les éventuelles altérations du paquet. On citera l'exemple du "Checksum"

(somme de contrôle) dans les protocoles TCP [89] et UDP [88] et du CRC dans le protocole Ethernet [2]. Grâce à ces systèmes, les erreurs peuvent être détectées et traitées en conséquence. Dans la plupart des protocoles, un paquet identifié comme erroné n'est pas transmis à la couche supérieure. Les systèmes de contrôle d'intégrité ne sont pas infaillibles, mais la probabilité qu'une erreur ne soit pas détectée est le plus souvent faible. Par exemple pour le Contrôle de Redondance Cyclique (CRC)-32 d'Ethernet, la probabilité de non-détection est de 2^{-32}. On peut donc faire l'hypothèse que tout paquet renvoyé directement ou indirectement par une couche incluant un système de détection d'erreur est 100% intègre. Ces considérations théoriques sont à mettre au regard des travaux [109] qui ont mis en évidence que ces probabilités de détection d'erreur peuvent être diminuées lorsqu'ils travaillent sur des flux réels. En effet les données sont rarement aléatoires ce qui introduit un biais dans les probabilités de détections. Néanmoins ces probabilités de non-détection sont toujours assez faible pour que l'hypothèse des paquets sans erreurs reste raisonnable. Ainsi les couches supérieures de la pile de communications vont être amenées à ne pas recevoir certains paquets soit parce qu'ils ont été perdus dans le réseau, soit parce qu'ils ont été ignorés après avoir été détectés comme erronés. Dans la suite nous appellerons ce phénomène la perte de paquets.

Le problème des pertes de paquets peut être résolu par l'utilisation d'un système de type requête automatique de répétition ou "Automatic Repeat reQuest" (ARQ) (Automatic Repeat reQuest) consistant à demander via un canal de retour, la retransmission des paquets manquants (exemple : le protocole TCP). Cependant cette solution peut ne pas être utilisable dans plusieurs situations :

- le canal de retour peut ne pas exister (liaison satellite unidirectionnelle),
- problème de passage à l'échelle en termes de nombre de destinataires dans le cas de distribution de type multicast ou broadcast [17]. Chaque récepteur subit un schéma de pertes différent et retransmettre ce qui n'a pas été reçu reviendrait à une retransmission totale de l'objet ;
- la retransmission implique un délai inacceptable, comme dans la distribution de flux temps réels (par exemple le streaming vidéo), et les communications longues distances (par exemple l'Internet interplanétaire [19]).

Apport des codes à effacements : Les codes correcteurs pour le canal à effacements s'imposent comme une solution aux pertes de paquets dans ces systèmes de communications où les approches de type ARQ sont insuffisantes ou inutilisables [16]. Une approche hybride est également possible : les solutions de requête automatique de répétition hybride ou "Hybrid Automatic Repeat reQuest" (HARQ) [31] combinent un système de retransmission avec un codage de correction d'erreurs ou "Forward Error Correction" (FEC).

Exemple 2 : Distribution de données via un carrousel

Le *Carrousel de données* ou disque de diffusion à grande échelle (broadcast disk) est un concept qui a été introduit par Acharya et. Al [5] en 1995. L'objectif est d'améliorer l'efficacité des systèmes de distribution de contenus à grande échelle, comportant une asymétrie au niveau des canaux de communications (canal montant à faible bande passante ou bien inexistant), ou ayant des récepteurs à connectivité intermittente. Le principe est de diffuser en boucle le contenu d'un disque virtuel sur un canal. Les récepteurs ayant une connexion asynchrone ou faisant l'objet de pertes massives pourront ainsi

recevoir l'intégralité du contenu diffusé au bout d'un certain temps d'écoute. Ce type d'approche implique un surcoût en réception non négligeable, qui correspond à la quantité d'information inutile reçue. Par exemple un récepteur auquel il manque un seul morceau du fichier sera obligé d'attendre le prochain tour du carrousel en espérant que la connectivité sera favorable à ce moment là.

Apport des codes à effacements : ce surcoût peut être considérablement réduit grâce à l'emploi de codes correcteurs pour le canal à effacements. En effet, l'ajout de redondance permet à un récepteur d'exploiter le contenu diffusé dès qu'il aura reçu une quantité d'information égale ou légèrement supérieure à celle du contenu original. Cela permet d'optimiser le service indépendamment du problème des pertes, puisque l'utilisateur pourra récupérer le contenu de manière efficace quel que soit l'instant auquel il commence le téléchargement. Dans ce type de situation les codes à effacements sans rendement utilisés sous la forme d'une "fontaine numérique" [22] sont particulièrement appropriés.

Exemple 3 : Stockage distribué

Dans le cas du stockage distribué, les données sont réparties sur plusieurs unités de stockage. Cette distribution a plusieurs objectifs :

- garantir la disponibilité des données en cas de panne d'un sous-ensemble d'unités de stockage,
- augmenter la vitesse de récupération à distance des données, en exploitant la bande passante de plusieurs unités de stockage à la fois.

La réplication des données constitue la solution la plus basique et elle induit une inefficacité au niveau de l'utilisation d'espace de stockage tout en apportant une garantie médiocre de disponibilité des données.

Apport des codes à effacements : l'usage de codes à effacements permet d'obtenir une disponibilité optimale des données tout en réduisant considérablement l'espace de stockage nécessaire [121]. Il permet également d'augmenter le nombre d'unités de stockage proposant une information pertinente pour la récupération d'un ensemble de données. Lorsqu'ils sont utilisés dans des applications de type P2P le coût en transmission nécessaire à la reconstruction des données perdues [37] devra être considéré.

On peut citer l'exemple très particulier des systèmes RAID, dans lesquels les données sont distribuées sur les disques d'une même machine. Dans ce type de système, chaque disque constitue une unité de stockage, et la panne d'un disque correspond à la perte de plusieurs giga-octets de données. L'utilisation d'un ou de plusieurs disques supplémentaires permet alors de tolérer efficacement la panne de disques.

S'inspirant de l'architecture en couche des systèmes de communication, le système Parchive [25] inclut une couche de détection d'erreur au niveau des unités de données distribuées : un contrôle d'intégrité avec la fonction de hachage MD5 [96] est effectué afin de détecter d'éventuelles erreurs. L'algorithme de récupération ne prendra en compte que les unités de données ayant un hash MD5 valide. La version 2 [25] du système Parchive utilise des codes de Reed-Solomon [91] basés sur les matrices de Vandermonde tels que Plank les a décrits dans [87] (voir section 2.4.1). La troisième version de Parchive est en cours de spécification et elle devrait incorporer, en plus des codes RS, des codes LDPC [42] de type LDPC-staircase [99] (voir section 2.5.4).

2.3 Codes correcteurs d'erreurs

D'après les travaux de Shannon, on sait que la transmission fiable sur des canaux comportant des erreurs est possible grâce à un codage approprié. Ces codes sont appelés codes correcteurs d'erreurs. Dans un premier temps nous entendrons par *erreurs* des altérations et/ou des effacements (section 2.3.1). Par la suite (section 2.3.2 et suivantes) nous nous limiterons aux effacements.

2.3.1 Notions relatives aux codes

Commençons par présenter quelques concepts et outils indispensables à la compréhension des codes dont nous allons parler dans cette thèse. Ces codes travaillent sur des corps finis, qui sont des ensembles d'éléments sur lesquels des opérations sont définies.

Définition On appelle corps fini (ou corps de Galois) un ensemble d'éléments sur lequel sont définies l'addition, la multiplication, la soustraction et la division. Pour chaque entier premier p et entier m, il existe un unique corps fini (à un isomorphisme près) à $q = p^m$ éléments noté GF(q) ou $\mathbb{F}_q$.

On peut prendre l'exemple de l'ensemble des entiers modulo p, $\mathbb{Z}/p\mathbb{Z}$, qui est un corps fini si et seulement si p est premier. On s'intéressera au cas où q est une puissance de 2, GF(2^k). En effet l'addition et la soustraction sur ce corps fini correspondent au XOR bit à bit des éléments, ce qui permet une implémentation efficace sur les processeurs.

Au lieu de décrire le corps fini dans son intégralité, on peut utiliser un élément générateur de celui-ci. On obtient alors par des puissances successives de cet élément générateur, l'ensemble des éléments du corps fini.

Définition On appelle élément générateur d'un corps fini GF(q) un élément α tel que GF$(q) = \{1, \alpha, \alpha^2, \ldots, \alpha^{q-2}\}$

Les travaux présentés par la suite se concentrent sur des codes correcteurs protégeant une quantité finie de données à la fois qu'on appelle code bloc. Les codes bloc linéaires bénéficient d'une structure d'espace vectoriel qui en font des codes particulièrement pratiques à utiliser.

Définition On appelle code bloc linéaire $\mathscr{C}$ de longueur n et de dimension k un sous espace vectoriel de dimension k de l'espace vectoriel $\mathbb{F}_q^n$, où $\mathbb{F}_q$ désigne le corps fini à q éléments. On appelle ce code un $[n,k]$-code. Les éléments de $\mathscr{C}$ sont appelés mots de code.

Définition On appelle symbole les éléments du corps fini sur lequel travaille le code. Ces symboles composent les mots du code.

A partir de sa longueur et de sa dimension, on peut définir le rendement d'un code. Le rendement, également appelé taux de codage, représente la quantité de redondance ajoutée par le code.

Définition On appelle rendement d'un code de dimension k et de longueur n le rapport $R = k/n$.

Définition On appelle matrice génératrice, la matrice de l'application linéaire de $\mathbb{F}_q^k$ dans $\mathbb{F}_q^n$ dont l'image est le code. Cette application est aussi appelée encodage. Les lignes de la matrice génératrice $G \in \mathscr{M}(\mathbb{F}_q)_{k,n}$ d'un $[n,k]$-code linéaire forment une base de l'espace vectoriel du code.

Soit X un élément de $\mathbb{F}_q^k$ et G la matrice génératrice d'un $[n,k]$-code, X est encodé en un mot de code $Y \in \mathbb{F}_q^n$ par une multiplication à gauche par la matrice génératrice :

$$Y = X\,G \tag{2.5}$$

Lorsque le vecteur source se retrouve dans le vecteur encodé, on parle d'un code systématique. La matrice génératrice d'un tel code contient alors la matrice identité (à une permutation des colonnes près) :

$$G_{\text{systematique}} = [Id_k|C] \tag{2.6}$$

On peut définir à partir de là une autre matrice, appelé matrice de parité, dont l'application correspondante a comme noyaux les éléments du code.

Définition Soit $\mathscr{C}$ un $[n,k]$-code. On dit que $H \in \mathscr{M}(\mathbb{F}_q)_{(n-k),n}$ est une matrice de parité de ce code ssi $\forall X \in \mathscr{C}, HX = 0$ et H est de rang plein.

Si on possède la matrice génératrice du code sous forme systématique $G_{\text{systématique}} = [Id_k|C]$, alors on peut construire une matrice de parité pour ce code de la façon suivante :

$$H = [-C^T|Id_{n-k}] \tag{2.7}$$

En effet, soit $Y \in \mathscr{C}$:

$$\begin{aligned} HY^T &= H(XG)^T = HG^TX^T && (2.8)\\ &= [-C^T|Id_{n-k}][Id_k|C]^TX^T = (-C^T + C^T)X^T = 0 && (2.9)\end{aligned}$$

Le décodage d'un code correcteur ayant comme objectif de retrouver la séquence ayant été la plus probablement envoyée, et donc la plus proche de celle reçue, il est nécessaire d'introduire une notion de distance. La distance utilisée est la distance de Hamming qui découle de la fonction de poids de Hamming.

Définition Le poids de Hamming d'une chaîne de symboles sur un alphabet donné est le nombre de symboles différents du symbole nul.

Quand cet alphabet est le corps fini à 2 éléments $\mathbb{F}_2$, le poids de Hamming est le nombre de bits égaux à 1.

Définition La distance de Hamming entre 2 chaînes de symboles a et b est le nombre de positions pour lesquelles les symboles correspondant sont différents. Dit autrement, la distance de Hamming est le poids de Hamming de la différence $a - b$.

L'application de la notion de la distance de Hamming à un code, amène à s'intéresser à la distance séparant les différents vecteurs de cet ensemble, et en particulier le minimum de ces distances : la distance minimale du code.

Définition La distance minimale d'un code linéaire, notée d_{min}, est la plus petite distance de Hamming qui existe entre deux mots de code différents. A cause de la structure d'espace vectoriel des codes linéaires, c'est aussi le minimum des poids de Hamming de tous les mots de code.

Un $[n,k]$-code de distance minimale d sera noté $[n,k,d]$-code. Les capacités de correction d'un code linéaire sont fortement dépendantes de sa distance minimale. En effet celle-ci nous donne une borne sur le nombre de corrections garanties par le code. Un code correcteur de distance minimale d_{min} sera toujours capable de corriger si $d_{min} \geq 2t + e + 1$, où e est le nombre d'effacements et t est le nombre d'erreurs. Dans certains cas, il est possible que le code puisse corriger même si l'inégalité précédente n'est pas respectée.

Cette distance minimale possède une borne supérieure qui dépend de la longueur et de la dimension du code. Les codes qui atteignent cette borne supérieure sont appelés codes maximum distance séparable et possèdent une capacité de correction optimale.

Définition Un code linéaire de longueur n et de dimension k est dit MDS si : $n - k = d_{min} - 1$. On dit alors que le code atteint la borne de Singleton [105].

2.3.2 Codage pour le canal à effacements

Cette thèse s'intéresse principalement au codage pour la correction des erreurs du canal à effacements. Nous allons donc présenter les principes généraux du fonctionnement et de l'utilisation de ces codes.

Considérons un canal à effacements ayant comme probabilité d'effacement p et supposons que nous souhaitons transmettre sur celui ci un message composé de k symboles. On choisit un code correcteur MDS dont le rendement R est inférieur à la capacité du canal $1 - p$, afin de permettre une transmission fiable (voir figure 2.3).

L'étape d'encodage consiste à utiliser le code pour transformer les k symboles du message en n symboles encodés. Les n symboles encodés sont alors transmis sur le canal à effacements. En moyenne, pn symboles seront perdus. Le récepteur n'aura alors à sa disposition en moyenne que $(1 - p)n$ symboles. Si le nombre de symboles reçus est supérieur ou égal à k, alors le décodage est possible, et le récepteur peut reconstruire le message original. Dans le cas inverse, le décodage échoue et le récepteur n'aura pas à sa disposition la totalité des symboles source.

2.3.3 Codage par paquets

Contrairement aux codes corrigeant des altérations de données, les codes à effacements travaillent généralement sur les couches hautes de la pile protocolaire (voir section 2.2). Alors que les premiers travaillent sur des petites quantités de données (quelques milliers de bits), les codes à effacements travaillent le plus souvent sur des quantités de données pouvant aller jusqu'à plusieurs méga octets. On les trouve la plupart du temps, juste en dessous de la couche applicative, là où les unités de données transmises sont grandes (paquets IP). C'est pourquoi on les appelle codes correcteurs de niveau applicatif ou codes AL-FEC pour "Application-Level Forward Error Correction". Ces codes correcteurs de niveau applicatif doivent protéger des ensembles de données divisées en paquets. Le code correcteur et la façon dont le code encode et organise les données est appelé brique AL-FEC. En effet la plupart du temps, le code à effacements ne peut pas être directement utilisé

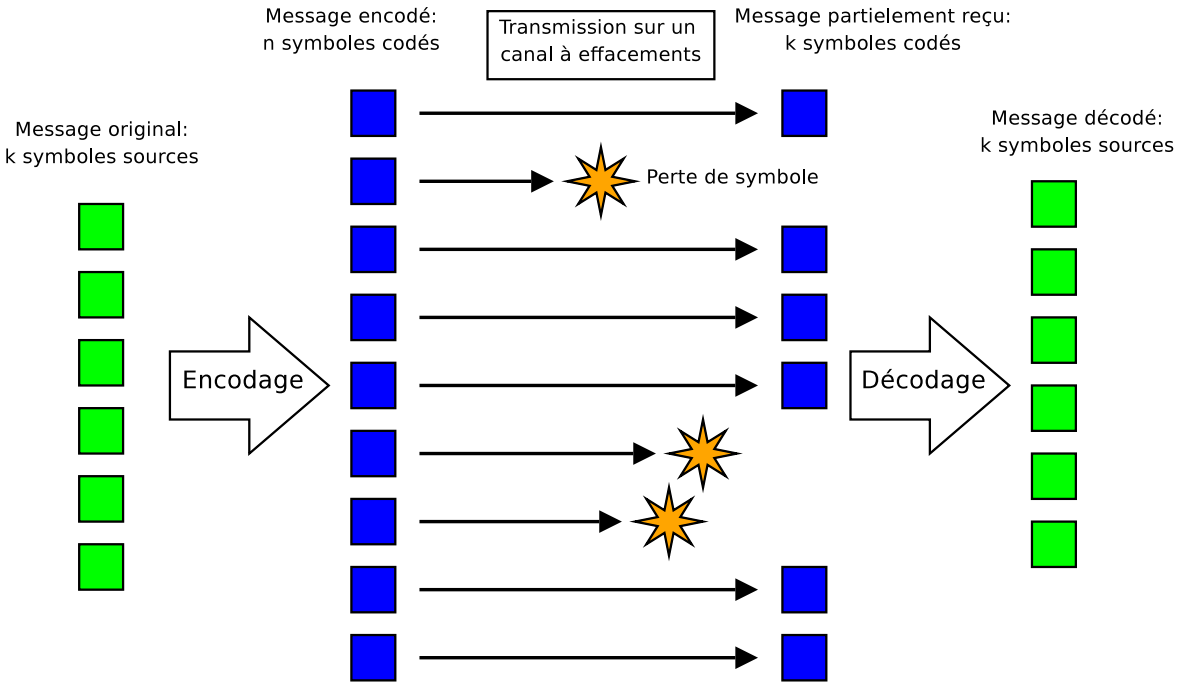


FIG. 2.3 – Utilisation d'un code à effacements

sur le flux de paquets. Comme nous allons le voir, certaines opérations de préparation et d'organisation des données sont nécessaires avant d'utiliser le code à effacements.

Comme nous l'avons dit plus tôt, ces paquets ont des tailles qui peuvent atteindre plusieurs milliers d'octets, et rien ne garantit qu'ils soient tous de la même taille. Le cas de paquets de taille variable peut être ramené au cas de paquets de taille constante par l'utilisation de padding (bourrage avec des zéros pour atteindre la taille cible). Nous considérerons donc à partir de maintenant que les paquets d'un flux à protéger ont tous la même taille.

Correspondance symboles/paquets

Les codes de niveau applicatif protégeant ces flux de données doivent être capables de récupérer les effacements de paquets entiers. Nous avons vu précédemment que les codes à effacements sont capables de récupérer des effacements de symboles. Il suffit donc de trouver une correspondance adéquate entre symboles et paquets. Plusieurs solutions sont alors possibles :

1. on considère chaque paquet comme un seul et unique symbole. Ceci nécessite d'avoir des symboles de taille arbitrairement longue, et donc de travailler sur un corps fini très grand. Avec certains codes, l'utilisation de symboles de grande taille peut s'avérer difficile en pratique car les opérations deviennent trop coûteuses (voir section 2.4).
2. on choisit des symboles de petites tailles et on divise chaque paquet en symboles. Il faudra utiliser un code de grande longueur $n = N_{paquet}\, Taille_{paquets}/Taille_{symbole}$. La longueur du code dépend, pour certain codes, de la taille du corps fini, et l'utilisation de grand corps fini n'est pas toujours possible en pratique (voir section 2.4).

Cependant, cette technique a l'avantage de limiter la quantité de padding éventuelle en cas de taille variable, puisqu'il ne faut compléter les paquets que jusqu'à obtenir une longueur multiple de la taille des symboles.

3. on utilise un tableau d'encodage pour faire un encodage par ligne des paquets rangés par colonnes. On range un paquet dans chaque colonne, et les lignes du tableau sont telles que chaque élément du tableau a la taille d'un symbole. On effectue alors l'encodage ligne par ligne, ce qui revient à avoir plusieurs instances du même code qui travaillent en parallèle (voir figure 2.4). Ainsi la dimension du code utilisé est égale au nombre de paquets contenant les données sources à protéger. Il est important de noter que lorsque cette solution est utilisée, chaque instance du code est soumise au même schéma de pertes. Ainsi une part importante du décodage est commune aux différentes instances du code.

Le codage avec des codes travaillant sur le corps fini $\mathbb{F}_2^s$, tels que les codes LDPC binaires, peut être vu de différentes manières. Puisque les opérations sur les corps fini binaires sont des XOR, on peut considérer soit qu'il s'agit d'un seul code sur $\mathbb{F}_2^s$ (première solution), soit qu'il s'agit de s instances différentes d'un même code, travaillant sur $\mathbb{F}_2$, utilisées en parallèle (troisième solution).

Lorsque l'on utilise un code avec une longueur et une taille de symbole limitée (comme les codes Reed-Solomon ,voir section 2.4), on utilisera la troisième solution. Par exemple un code Reed-Solomon travaillant sur GF(256) (un symbole correspond à un octet) permettra d'encoder des paquets par bloc de 255. Pour cela on rangera les paquets dans un tableau dont les colonnes ont une largeur égale à un octet, et on encodera ligne par ligne avec un code Reed-Solomon. Ainsi si les paquets sont de taille t bits, le tableau d'encodage possédera $t/8$ lignes et autant d'instances de code Reed-Solomon travailleront en parallèle.

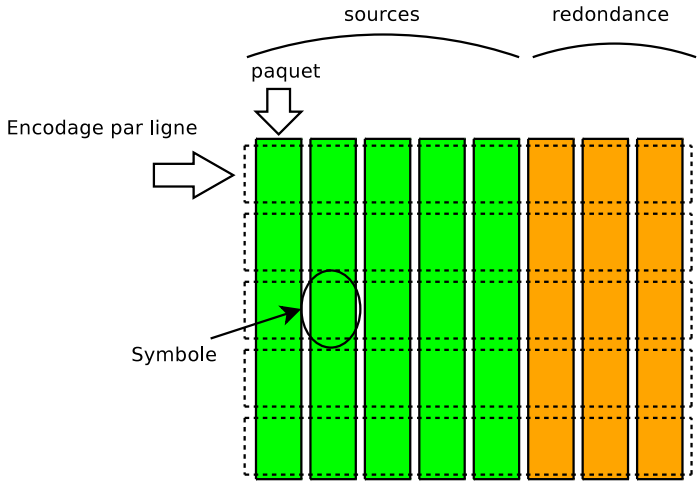


FIG. 2.4 – Utilisation d'un tableau pour l'encodage de paquets

Transmission des paquets

Indépendamment du découpage des paquets en symboles, il est possible, après l'étape d'encodage, d'effectuer une permutation des symboles. En effet certains codes peuvent être sensibles à la perte de plusieurs symboles consécutifs (pertes en rafale). La permutation

des symboles permet donc du point de vue du code, de transformer ces effacements consécutifs, en effacements de symboles dispersés dans le mot de code, améliorant ainsi les performances du système [79].

Du point de vue de l'utilisateur de la brique AL-FEC, le padding, le découpage en symbole des paquets, ainsi que la permutation des symboles à l'intérieur des paquets, sont totalement transparents. En effet la brique AL-FEC prend en entrée un ensemble de paquets à protéger et renvoie un ensemble de paquets protégés contre les effacements.

2.4 Codes Reed-Solomon

Les codes Reed-Solomon sont une classe de codes correcteurs introduits par Irvine Reed et Gustave Solomon en 1960 [91]. Ils constituent un cas particulier des codes BCH [14]. Ils ont été rendus célèbres par leur utilisation dans les "Compact Disk" et dans les communications avec les sondes Voyager. Travaillant sur des corps finis, les codes Reed-Solomon sont basés sur le suréchantillonnage des polynômes et sur le fait qu'un polynôme de degré t est determiné par $t+1$ valeurs. Les codes Reed-Solomon ont comme avantage d'être des codes parfaits (MDS).

En 1969 Massey présenta un algorithme [66] pour les décoder efficacement basé sur un algorithme introduit par Berlekamp [8][9]. L'algorithme de décodage, baptisé algorithme de Berlekamp-Massey, permet de décoder avec une complexité de $O(n\log^2 n)$ sur les canaux à erreurs par valeur (tel que le canal binaire symétrique, voir section 2.1.2).

La longueur des codes Reed-Solomon est limitée par la taille du corps fini utilisé. Un corps fini de taille q permet de construire des codes Reed-Solomon (RS) de taille maximum $q-1$. Ainsi pour un code RS construit avec le corps fini GF(2^p), la longueur maximum est $n_{max} = 2^p - 1$. En théorie ce n'est pas une limitation car quelle que soit la longueur désirée on peut toujours choisir un corps fini assez grand pour construire un tel code. Cependant d'un point de vue *pratique* il est préférable de garder le corps fini le plus petit possible, puisque le coût des opérations sur le corps fini augmente rapidement avec sa taille. C'est pour cela qu'en pratique on considère des corps finis à 2^4, 2^8 et parfois 2^{16} éléments si l'on dispose d'une capacité de calculs suffisante (par exemple PAR2 [25]), mais très rarement plus.

Codes de Reed-Solomon pour le canal à effacements : Les codes Reed-Solomon peuvent aussi être utilisés pour corriger les effacements. Dans ce cas l'algorithme de Berlekamp Massey [66] n'est plus approprié, et il faut utiliser d'autres constructions basées sur les matrices de Vandermonde par exemple.

Nous insistons sur le fait que l'utilisation des codes de Reed-Solomon sur le canal à effacements est différente de celle qui est utilisée sur les canaux à erreurs par valeur. En effet, pour transmettre sur ces derniers canaux, on multiplie le polynôme représentant nos données source par un polynôme générateur du code. Alors que sur le canal à effacements, on va effectuer une évaluation en n points différents, du polynôme représentant les données source. Ces deux approches sont totalement équivalentes [122], et on notera que celle que nous utilisons sur le canal à effacements est en fait la méthode qui avait été présentée par Reed et Solomon dans leur papier original [91]. La seconde approche, avec polynôme générateur, qui a vu le jour plus tard grâce aux travaux de Berlekamp [8][9] et de Massey [66], est devenue la plus populaire.

De récents travaux [108] ont mis en avant la possibilité de décoder les codes Reed-Solomon sur le canal à effacements, avec des transformées de Fourier. Cette technique permet de réduire la complexité à $O(n \log n)$ et ainsi d'atteindre un débit de décodage de 10Mbps sur des codes de longueur 10000. Même si le gain est important, le débit atteignable reste toujours de l'ordre de la dizaine de Mbps, ce qui peut ne pas être suffisant pour un certain nombre d'applications.

2.4.1 Codes Reed-Solomon sur matrice de Vandermonde

Ces ensembles de codes utilisent comme matrice génératrice une matrice de Vandermonde, et une approche détaillée a été présentée par Plank dans [87]. Luigi Rizzo a proposé une implémentation de ces codes [97] qui sert aujourd'hui de référence de comparaison pour l'évaluation des codes à effacements. Ces codes ont été standardisés à l'IETF [55].

Une matrice de Vandermonde V de taille $m \times n$ est construite à partir de α, où α est un élément générateur du corps fini. Une telle matrice de Vandermonde a la forme suivante : $V = \{v_{i,j}\}_{0 \leq i \leq m-1; 0 \leq j \leq n-1}$ avec $v_{i,j} = \alpha^{i \times j}$.

$$V = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & & \alpha^{2(n-1)} \\ & & & & & \\ 1 & \alpha^{m-1} & \alpha^{2(m-1)} & \alpha^{3(m-1)} & \dots & \alpha^{(m-1)(n-1)} \end{pmatrix}$$

Rappelons que toute sous-matrice carrée d'une matrice de Vandermonde est inversible. De cette propriété provient le caractère MDS du code ainsi construit. Pour construire un code Reed-Solomon de longueur n et de dimension k, on sélectionne un corps fini de taille supérieure ou égal à n, puis on construit une matrice de Vandermonde G de taille $n \times k$ à partir d'un élément générateur. La matrice G est alors la matrice génératrice du code. On pourra remarquer que cette matrice ne contient pas nécessairement l'identité, le code n'est donc pas forcément systématique. Néanmoins, on peut créer à partir de cette matrice un nouveau code systématique. Cette transformation consiste à appliquer un algorithme d'élimination de Gauss sur les colonnes de G de manière à faire apparaître une matrice identité sur les k premières lignes : $G' = (Id_k | G'_1)^T$.

Encodage Pour encoder un vecteur de taille k dont chaque élément appartient au corps fini, on effectue simplement la multiplication par la matrice génératrice :

$$Y = GX \tag{2.10}$$

On obtient un vecteur de longueur n représentant le mot de code que l'on peut transmettre sur le canal. Par exemple sur GF(2^8), le vecteur X composé de k octets sera encodé en un vecteur Y de n octets, avec $k \leq n \leq 2^8 - 1$.

Sur un canal à effacements de paquets, nous sélectionnons en guise de vecteur X, les octets de position i, $0 \leq i \leq l-1$, de chaque paquet source. Encoder un ensemble de k paquets revient donc à faire l encodage successifs [55].

Décodage Après réception, le mot de code contient potentiellement des effacements qu'il va falloir corriger. Il est à noter que le plus souvent il est inutile de corriger les effacements concernant les symboles de redondance, puisque la récupération des données source est suffisante.

Soit $Y' = \{y'_0, y'_1, \ldots, y'_{n-1}\}$ où $y'_i \in \{\mathrm{GF}(q) \cup E\}$ le mot de code reçu. L'objectif du décodage est de retrouver X à partir de Y' grâce à la relation 2.10. Supposons que t symboles de Y aient été reçus correctement. Le décodage revient alors à résoudre un système linéaire de t équations (les symboles reçus) à k inconnues (les symboles source). Si le nombre de symboles reçu est suffisant, c'est à dire $t \geq k$, alors on sélectionne k éléments de Y' non effacés ,$\overline{Y'}$, ainsi que les lignes de G correspondantes, $\overline{G}$. On appelle $\overline{G}$, la matrice de décodage. On obtient la relation suivante :

$$\overline{Y'} = \overline{G}X \tag{2.11}$$

Comme $\overline{G}$ est une sous matrice carré d'une matrice de Vandermonde, elle est inversible. Le décodage peut alors être effectué en inversant la matrice $\overline{G}$ puis en la multipliant à $\overline{Y'}$:

$$X = \overline{G}^{-1}\overline{Y'} \tag{2.12}$$

En revanche, si le nombre de symboles reçus est insuffisant, c'est à dire $t < k$, alors le système n'admet pas de solution unique et le décodage complet est impossible.

Interprétation polynomiale Si l'on considère que le vecteur X représente les coefficients d'un polynôme P de degré $k-1$ ($P(z) = \sum_{s=0}^{k-1} z^s x_s$), le mot de code représente alors les évaluations de ce polynôme en les points $\{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$. En effet :

$$\forall i \in [0 \,..\, n-1]\ ,\ y_i = \sum_{s=0}^{k-1} v_{i,s} x_s = \sum_{s=0}^{k-1} \alpha^{i \times s} x_s = \sum_{s=0}^{k-1} (\alpha^i)^s x_s = P(\alpha^i)$$

Le décodage revient alors à retrouver le polynôme original à partir d'un ensemble de t évaluations différentes de ce polynôme. Il est alors évident que k évaluations sont suffisantes pour retrouver le polynôme de degré $k-1$. Cette approche correspond donc à l'utilisation originale présentée par Reed et Solomon, puisque la multiplication par la matrice de Vandermonde revient à faire une évaluation du polynôme en n points différents.

2.4.2 Codes MDS sur matrice de Cauchy

Sur le modèle des codes Reed-Solomon sur matrice de Vandermonde, il est possible de construire un code MDS avec une matrice de Cauchy. En effet ces matrices ont également l'intéressante propriété d'avoir toutes leurs sous-matrices carrées inversibles. Une matrice de Cauchy de taille $k \times n$ sur un corps fini se construit à partir de 2 vecteurs composés d'éléments distincts $A = \{a_i\}_{0 \leq i \leq k}$ et $B = \{b_i\}_{0 \leq i \leq n}$ et a la forme suivante :

$$C = \{c_{i,j}\}_{0 \leq i \leq m-1; 0 \leq j \leq n-1} \text{ avec } c_{i,j} = \frac{1}{a_i - b_j}.$$

Comme pour les codes Reed-Solomon sur matrice de Vandermonde, l'encodage est une multiplication matrice-vecteur, et le décodage revient à résoudre un système linéaire.

Ces codes ont été introduits pour la première fois par McWilliam et Sloane[65] et on été utilisés par la suite dans le cadre de la fiabilisation de transmissions [13]. Ces codes présentent des performances similaires à celles des codes Reed-Solomon basés sur des matrices de Vandermonde [108].

2.5 Codes à matrice creuse : LDPC

En 1963 Robert G. Gallager présenta dans sa thèse [42] un nouveau type de codes correcteurs, les codes LDPC, pour "Low Density Parity Check". Ces codes sont appelés ainsi car ils ont la particularité d'avoir une matrice de parité possédant une faible densité. Dans le cas binaire, une matrice de parité d'un code LDPC possède une majorité d'entrées nulles et une minorité d'entrées égales à '1'. Grâce à cette faible densité de la matrice de parité, il est possible d'utiliser une classe particulière d'algorithmes de décodage reposant sur le principe de propagation de croyance ("Belief propagation"). Ces algorithmes ont l'avantage de posséder une complexité linéaire en la longueur du code. Ces algorithmes seront détaillés dans la section 2.5.3. Ainsi les codes LDPC peuvent protéger efficacement des messages de grande longueur, ce qui peut se révéler essentiel en pratique.

L'état des technologies dans les années 1960 ne permettait pas une utilisation pratique des codes LDPC, ils ont donc été "oubliés" pendant 30 ans, jusqu'à ce qu'ils soient "redécouverts" par MacKay [64]. Depuis les codes LDPC constituent un domaine de recherche très actif, et ils sont aujourd'hui incontournables dans le domaine de la théorie des codes et des télécommunications. Les codes LDPC ont également servi de base à plusieurs codes dérivés, tels que les codes Tornado [22] , les codes LT [58] et les codes Raptor [103], ainsi que les innombrables variantes des codes LDPC : tels que les LDPC généralisés, les LDPC basés sur protographe et les LDPC Quasi-cycliques.

Les codes LDPC ont d'abord été utilisés pour la transmission fiable sur des canaux à erreurs par valeur tels que le BSC ou le canal à bruit blanc gaussien additif ou "Additive White Gaussian Noise" (AWGN). Les efforts se sont alors concentrés sur l'amélioration des codes LDPC et des décodeurs pour ces types de canaux. De nouvelles applications telles que l'Internet et le stockage distribué ont fait apparaître le besoin de corriger efficacement les pertes d'information. Les codes LDPC et leurs décodeurs ont alors été rapidement adaptés pour transmettre sur les canaux à effacements [61]. On remarquera qu'en 1974, Zyablov et Pinsker parlaient dans leur article [123] de l'utilisation des codes à faible densité pour le canal à effacements et ils y décrivaient l'adaptation (triviale) de l'algorithme de "Belief propagation" au cas du canal à effacements.

Tout comme pour les autres codes linéaires, la capacité de correction des codes LDPC dépend du poids des mots de codes et en particulier de la distance minimale. Si en moyenne le poids des mots de code LDPC est grand, il existe des mots de poids faible qui font que les codes LDPC possèdent une petite distance minimale. Lorsqu'ils sont décodés avec un décodeur itératif, les capacités de correction des codes LDPC seront également limitées par des caractéristiques telles que la longueur des cycles dans le graphe associé et la taille des "stopping sets" (voir section 2.5.5).

On peut utiliser les codes LDPC comme des codes non-systématiques si on se contente d'envoyer les symboles de redondance ou alors comme des codes systématiques si on transmet également les symboles source.

Nous allons maintenant faire une introduction rapide aux codes LDPC en présentant dans un premier temps les moyens de représentation de ces codes, puis les algorithmes

d'encodage et de décodage supportés par ces codes, et enfin quelques outils d'évaluation et de construction de ces codes.

2.5.1 Représentation des codes LDPC

Il existe deux représentations usuelles des codes LDPC : le graphe biparti et la matrice de parité. Ces deux représentations sont totalement équivalentes, et l'on choisira l'une ou l'autre en fonction du niveau de simplification qu'elle apporte pour la résolution du problème donné.

Graphe de Tanner

La première représentation des codes LDPC est un graphe biparti (voir figure 2.5), aussi appelé graphe de Tanner [113]. Un graphe est dit biparti s'il existe deux ensembles de noeuds (sommets) U et V et un ensemble d'arêtes, tels que chaque arête relie un noeud de U à un noeud de V . La figure 2.6 représente le graphe de Tanner d'un code LDPC.

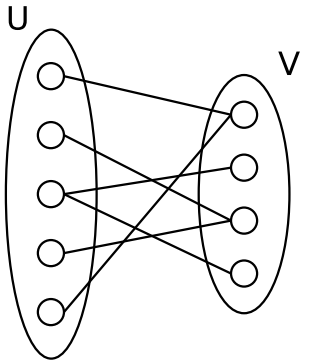


FIG. 2.5 – Graphe biparti

Un code LDPC peut être défini à partir d'un graphe biparti dont l'ensemble des noeuds de gauche, noté V (noeuds variable), représente les symboles du mot de code, et l'ensemble des noeuds de droite, noté C (noeuds de contrainte), représente les contraintes.

Une suite de symboles constitue alors un mot de code valide si et seulement si pour chaque noeud de contrainte, la somme des symboles correspondants aux noeuds variables adjacents est nulle.

La représentation d'un code LDPC sous la forme d'un graphe de Tanner est particulièrement utile lorsque l'on veut se représenter le mécanisme de passage de messages ("message passing") qui intervient dans les décodeurs itératifs. De plus, cette représentation permet également de faire appel aux puissants outils de la théorie des graphes pour étudier et concevoir les codes LDPC. En effet nous verrons que les cycles, et en particulier leur taille, ont une influence directe sur les performances de correction de ces codes.

Matrice de parité

Un graphe biparti peut être représenté sous la forme d'une matrice binaire, aussi appelée matrice d'adjacence. Dans une telle matrice, chaque colonne correspond à un

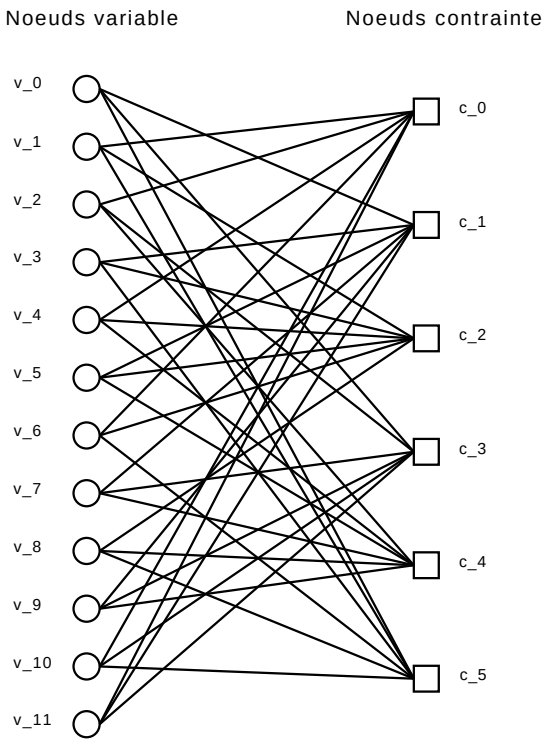


FIG. 2.6 – Graphe de Tanner d'un code LDPC

noeud de U et chaque ligne à un noeud de V. L'élément (i, j) de la matrice est non nul si et seulement si il existe une arête reliant le noeud d'indice j de U au noeud d'indice i de V.

La matrice d'adjacence du graphe de Tanner d'un code LDPC est en fait sa matrice de parité. Comme pour tout code linéaire, la matrice de parité définit totalement le code. Plus précisément la matrice de parité définit des relations entre les symboles des mots du code, sous la forme d'un système linéaire. Lorsqu'il a introduit les codes LDPC, Gallager a spécifié que cette matrice devait avoir un faible nombre d'éléments non-nuls. La matrice étant creuse, chaque relation va concerner un petit nombre de symboles.

Rappelons que pour un code linéaire donné, il peut exister plusieurs matrices de parité. Par matrice de parité d'un code LDPC, on désignera la matrice, à priori la plus creuse, qui a été utilisée pour définir le code.

Une matrice de parité d'un code linéaire de longueur n et de dimension k sur $\mathbb{F}_q$ est une matrice de rang plein à n colonnes et $n-k$ lignes à éléments dans $\mathbb{F}_q$. Dans le cas d'un code systématique, chaque colonne correspond à un symbole, les k premières colonnes correspondant aux symboles sources et les $n-k$ dernières colonnes correspondant aux symboles de redondance (symboles de parité dans le cas binaire). La partie gauche de la matrice correspondant aux symboles sources est appelé H_1 et la partie droite correspondant aux symboles de redondance est appelée H_2.

Si X est un élément de $\mathbb{F}_q^k$ représentant les symboles sources et si Y est un élément de $\mathbb{F}_q^{(n-k)}$ représentant les symboles de redondance, alors on a la relation :

$$\left(\begin{array}{c|c} H_1 & H_2 \end{array} \right) \left(\begin{array}{c} X \\ Y \end{array} \right) = 0 \tag{2.13}$$

Dans le cas d'un code binaire, si on considère ce système équation par équation on

obtient la propriété suivante : pour un mot de code donné, pour chaque ligne de la matrice de parité, la somme des symboles (source et parités) présents dans celle-ci est nulle.

La matrice de parité peut avoir une structure particulière. Par exemple pour les codes de types “Repeat-Accumulate” (RA) (cf. section 2.5.4), la partie droite de la matrice de parité a une structure triangulaire inférieure (dans le cas des LDPC-staircase elle a une structure double diagonale inférieure). Elle peut également posséder une structure par bloc, comme pour les codes LDPC quasi-cycliques (cf. chapitre 7) ou une structure bande (cf. chapitre 6).

Au delà de faible densité de sa matrice de parité, les performances obtenues seront largement tributaires du nombre d’entrées non nulles par ligne et par colonne. On appelle cette quantité, le degré d’une ligne (resp. colonne). Quand cette quantité est constante, on appelle le code ainsi défini un code LDPC régulier.

La figure 2.7 présente la matrice de parité du code LDPC régulier défini par le graphe de Tanner de la figure 2.6. La figure 2.8 montre le système linéaire correspondant où les $\{x_i\}_{1\leq i\leq 12}$ sont les variables représentant les symboles.

$$H_a = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

FIG. 2.7 – Matrice de parité d’un code LDPC régulier ($k = 6$, $n = 12$, $d_c = 3$, $d_r = 6$)

$$\left\{ \begin{array}{cccccccccccc} & x_2\oplus & x_3\oplus & & x_5\oplus & & x_7\oplus & & & & x_{11}\oplus & x_{12} & = 0 \\ x_1\oplus & & & x_4\oplus & & x_6\oplus & & x_8\oplus & & x_{10}\oplus & & x_{12} & = 0 \\ & x_2\oplus & & x_4\oplus & x_5\oplus & x_6\oplus & x_7\oplus & & x_9\oplus & & & & = 0 \\ x_1\oplus & & x_3\oplus & & & & & x_8\oplus & & x_{10}\oplus & x_{11}\oplus & x_{12} & = 0 \\ & & x_3\oplus & & x_5\oplus & x_6\oplus & & x_8\oplus & x_9\oplus & x_{10} & & & = 0 \\ x_1\oplus & x_2\oplus & & x_4\oplus & & & x_7\oplus & & x_9\oplus & & x_{11} & & = 0 \end{array} \right\}$$

FIG. 2.8 – Système linéaire défini par la matrice de parité H_a

Le degré des colonnes est noté d_c et le degré des lignes est noté d_r. Par conservation du nombre total d’entrées non-nulles dans la matrice on obtient la relation suivante :

$$d_c\, n = d_r\, (n-k) \tag{2.14}$$

Codes réguliers et irréguliers : terminologie On parlera de code LDPC irrégulier [62] quand les degrés des noeuds ne sont pas constants et suivent des distributions particulières. Une distribution définit la fraction des lignes (resp. colonnes) ayant un degré donné. Ces distributions sont habituellement représentées par des polynômes.

Les distributions de degrés d’un point de vue noeud sont caractérisées par les polynômes

(Λ,P) et ont la forme suivante :

$$\Lambda(x) = \sum_{n \geq 1} \Lambda_n x^n$$
$$P(x) = \sum_{n \geq 1} P_n x^n$$

où Λ_n et P_n sont les fractions respectivement des noeuds variable et noeuds de contrôle de degré n, ce qui correspond respectivement aux fractions des colonnes et des lignes de la matrice de parité qui ont un degré égal à n.

On parle aussi de distribution de degré d'un point de vu arêtes. Les polynômes associés sont (λ,ρ) et ont la forme suivante :

$$\lambda(x) = \sum_{n \geq 1} \lambda_n x^n$$
$$\rho(x) = \sum_{n \geq 1} \rho_n x^n$$

où λ_n et ρ_n sont les fractions des arêtes connectées respectivement aux noeuds symboles et aux noeuds de contrôle de degré n.

Ces deux couples de polynômes sont liés par les relations suivantes :

$$\lambda(x) = \frac{\Lambda'(x)}{\Lambda'(1)}$$
$$\rho(x) = \frac{R'(x)'}{R(1)'}$$

Ces polynômes déterminent également le rendement R du code :

$$R = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \tag{2.15}$$

Lorsque la matrice possède une structure par bloc on pourra appliquer ces critères aux différents blocs afin de déterminer le caractère régulier ou irrégulier du code.

Prenons l'exemple des codes "Repeat-Accumulate" (voir section 2.5.4) dont les colonnes de la matrice de parité peuvent être divisées en deux blocs. La partie droite est une matrice escalier alors que la partie gauche est une matrice creuse sans structure particulière. La partie droite est une matrice régulière puisqu'elle possède deux entrées par lignes et par colonnes à l'exception de la première ligne et de la dernière colonne. La partie gauche quant à elle, peut être soit régulière soit irrégulière. Dans le cas ou elle est régulière, le code est régulier et on parlera de code "Regular Repeat Accumulate", alors que si elle est irrégulière, le code sera irrégulier et on parlera de code "Irregular Repeat Accumulate".

2.5.2 Encodage des codes LDPC

Pour effectuer l'encodage avec un code LDPC, on peut utiliser la matrice génératrice comme pour tout code linéaire : une multiplication du vecteur source par la matrice génératrice nous donne le vecteur correspondant au mot de code. Cette approche présente cependant plusieurs inconvénients :

– la matrice génératrice doit être connue. Si ce n'est pas le cas il faut la calculer par une élimination de Gauss sur la matrice de parité, opération coûteuse. Cette opération peut parfois être effectuée en avance et n'a donc pas d'impact sur la complexité de l'étape d'encodage. Cependant lorsque le code est généré à la volée, il est impossible d'effectuer ce pré-calcul, et la matrice génératrice devra être calculée avant de pouvoir faire l'encodage.
– la multiplication par la matrice génératrice (qui est dense à l'inverse de la matrice de parité) est coûteuse en nombre d'opérations (complexité en $O((n-k)n)$).

Pour pallier à ce problème, deux approches ont été proposées. La première consiste à utiliser une méthode d'encodage [93] qui transforme la matrice de parité en faisant apparaître une structure presque triangulaire inférieure sur la partie droite de la matrice. La résolution d'un système linéaire de taille réduite, suivie d'une étape de substitution à rebours ("backward substitution"), permet un encodage avec un coût moindre.

La seconde approche considère des codes LDPC dont la matrice possède une structure triangulaire inférieure dans H_2, de manière à supporter un encodage en temps linéaire (cf. section 2.5.4). Il s'agit de l'approche que nous avons systématiquement suivie dans nos travaux.

2.5.3 Décodage des codes LDPC

Sur le canal à effacements, le décodage revient à résoudre un système linéaire, dans lequel les variables sont les symboles non reçus et le membre constant contient l'information obtenue à partir des symboles reçus. Pour ce décodage, plusieurs solutions sont possibles. En particulier on peut utiliser un décodeur à faible complexité appelé décodeur itératif ou bien un décodeur par maximum de vraisemblance.

Décodage itératif

Le décodage itératif est sous-optimal en termes de capacité de correction, mais il a une complexité linéaire. Appliqué aux codes LDPC il permet d'atteindre une très bonne capacité de correction. Cette classe d'algorithmes de décodage a été introduite par Gallager [42]. De nombreuses variantes ont depuis vu le jour, et on notera celle de Zyablov [123] pour le décodage itératif sur le canal à effacements qui nous intéresse en premier lieu.

Le principe général est le suivant : dans un graphe où des valeurs sont associées aux noeuds, l'algorithme fait passer l'information le long des arêtes du graphe et modifie la valeurs des noeuds en fonction de l'information qu'il a reçu via les arêtes. L'algorithme converge vers un état stable après un certain nombre d'itérations.

Dans le cas d'un code LDPC binaire travaillant sur le canal binaire à effacement, les noeuds variables peuvent prendre 3 valeurs : $\{0,1,E\}$. Si un noeud de contrôle reçoit plus d'un message E (effacement), il ne peut rien faire, et il retourne aux noeuds variable la valeur E. Si un noeud contrôle reçoit un seul message E, il peut en déduire une nouvelle valeur pour ce noeud, cette valeur étant la somme de tous les autres messages reçus (en effet la somme des valeurs des noeuds variable adjacents à un noeud de contrôle doit être égale à zéro). L'algorithme s'arrête quand il ne reste plus de noeuds variable ayant comme valeur E. La figure 2.9 présente un décodage itératif sur le graphe d'un code LDPC.

Le décodage itératif sur le canal à effacements peut aussi s'interpréter sur la matrice de parité : si une équation du système n'a qu'une variable dont la valeur est inconnue, alors on lui attribue comme valeur la coordonnée correspondante du membre constant. On réinjecte

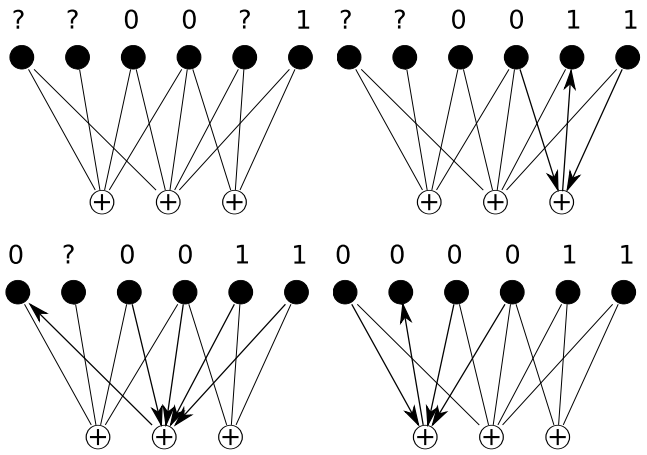


FIG. 2.9 – Exemple de décodage itératif sur un graphe

cette valeur dans toutes les équations où cette variable intervient. L'algorithme s'arrête quand il ne reste plus de variable indéterminée.

Cet algorithme permet de reconstruire au plus $n-k$ symboles (1 par équation). Chaque reconstruction nécessite de sommer $d_r - 1$ symboles. Le décodage nécessite donc au plus $(n-k)*(d_r-1)$ opérations sur les symboles. Pour un rendement donné, la complexité de cet algorithme est donc linéaire en la dimension du code.

Même si le décodage itératif aboutit dans un grand nombre de situations, il existe des cas où il ne pourra pas résoudre le système alors qu'une solution existe. Dit autrement le décodage itératif ne permet pas d'exploiter totalement les capacités de correction du code. Cette sous-optimalité est le prix à payer pour avoir un algorithme à complexité linéaire.

Décodage par Maximum de Vraisemblance

Les techniques de décodage par maximum de vraisemblance ou décodage ML ("Maximum Likelihood"), constituent une deuxième classe de solutions. Le décodage ML renvoie le mot de code le plus proche (dans le cas du canal BSC la distance s'exprime en termes de distance de Hamming) d'une suite de symboles donnée. Le mot de code retourné est en fait celui qui a été le plus vraisemblablement envoyé sur le canal. Lorsqu'on effectue un décodage ML, on exploite au mieux les capacités de correction du code. Cependant cette optimalité a un coût en complexité, qui peut devenir prohibitif pour des codes de grande longueur.

Pour le canal à effacements le décodage ML revient à résoudre un système linéaire avec une technique du type élimination de Gauss. Contrairement au décodage itératif, cet algorithme trouvera toujours la solution du système si elle existe. Nous verrons dans le chapitre 4) un moyen efficace d'implémenter le décodage ML des codes LDPC.

2.5.4 Codes LDPC définis par une matrice de parité avec structure triangulaire inférieure

Pour pallier au problème de complexité d'encodage des codes LDPC produits à la volée (cf. 2.5.2), des codes dont la structure permet un encodage en un temps linéaire ont été proposés. Ces codes sont tels que H_2 (la partie correspondant aux symboles de

redondance), possède une structure triangulaire inférieure. L'encodage peut être effectué en temps linéaire, à la condition de générer ces symboles dans leur ordre naturel.

On désigne souvent ces codes par "Repeat-Accumulate" [33] qui désigne également un type particulier de codes turbo [10] équivalents. En effet chaque symbole de redondance créé dépend du précédent, il y a ainsi un effet d'"accumulation". De plus comme les symboles sources sont présents dans plusieurs équations, on dit qu'ils sont "répétés". Les codes "Repeat-Accumulate" correspondent seulement aux codes de type staircase, c'est à dire des codes dont H_2 est une matrice escalier inférieure. Cependant on peut généraliser le principe des codes "Repeat-Accumulate". On appelle alors codes "Generalized Repeat-Accumulate" les codes dont la partie droite de leur matrice de parité est une matrice identité à laquelle on a ajouté une ou plusieurs sous-diagonales.

Un algorithme simple permet l'encodage de ce type de code. Prenons l'exemple de la matrice binaire de la figure 2.10. Le premier symbole de parité (7-ième colonne) ne dépendant que des symboles source, on le construit en lui attribuant comme valeur la somme des symboles présents sur la première ligne (symboles 2, 3 et 5). Le second symbole peut alors être construit, puisqu'il est le seul symbole inconnu dans la seconde ligne. Le reste des symboles de redondance est produit de la même manière en continuant à "descendre" l'escalier.

Les sections suivantes présentent des exemples particuliers de codes LDPC avec une structure triangulaire inférieure. Ces codes ont servi de base aux travaux présentés dans cette thèse.

LDPC-staircase

Les codes LDPC-staircase tiennent leur nom de la structure en escalier inférieur H_2 [63]. On en dérive les codes LDPC-triangle en ajoutant des entrées sous la diagonale d'un code LDPC-staircase suivant un certain algorithme. Ces codes ont été standardisés à l'IETF au sein du RFC5170 [99]. Un codec sous licence GNU/GPL auquel j'ai contribué est également disponible en ligne [118].

La sous matrice H_1 est une matrice ayant un degré colonne constant et égal à N_1, et un degré ligne également constant et égal à $N_1 \frac{k}{n-k}$. De plus, H_2 possède une distribution régulière, à l'exception de la première ligne et de la dernière colonne. Les codes LDPC-staircase sont donc par définition des codes LDPC réguliers. En plus de leur longueur, de leur dimension, et de la graine utilisée pour initialiser le générateur de nombres pseudo-aléatoires "PseudoRandom Number Generator" (PRNG), les codes LDPC-staircase sont déterminés par le paramètre N_1 que l'on appelle aussi *degré gauche*.

$$H_{LDPC-Staircase} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

FIG. 2.10 – Matrice de parité d'un code LDPC-staircase ($k = 6$, $n = 12$, $N_1 = 3$)

Lors de l'encodage, chaque symbole de parité construit nécessite $N_1 \frac{k}{n-k}$ opérations. Le nombre total d'opérations nécessaires à l'encodage est donc $N_1 k$. La complexité de

l'encodeur est donc linéaire en la dimension du code.

La matrice des codes LDPC-triangle, tels qu'ils sont définis dans le RFC5170 [99], forment un cas particulier des codes LDPC ayant une matrice avec une structure triangulaire inférieure. On les construit à partir d'une matrice LDPC-staircase à laquelle on ajoute des entrées sous la diagonale de la partie droite en suivant un algorithme spécifié dans le RFC5170 [99].

Par la suite nous verrons que malgré leur simplicité, ces codes permettent d'obtenir d'excellents résultats.

Codes à matrice génératrice creuse (LDGM)

Les codes à matrice génératrice creuse, ou codes "Low-Density Generator Matrix" (LDGM), peuvent être vus comme des codes LDPC possédant une partie droite égale à l'identité (et donc triangulaire inférieure). Comme indiqué dans la section 2.3, la matrice de parité d'un code linéaire s'obtient en concaténant la transposée de la partie non-systématique de la matrice génératrice à la matrice identité :

$$H_{\text{LDGM}} = (C^T | Id)$$

La matrice de parité hérite donc du caractère faible densité de la matrice génératrice, on peut alors la considérer comme un code LDPC et en particulier utiliser un décodage itératif.

Le grand nombre de colonnes de degré 1 dans la matrice de parité (matrice identité) fait que les codes LDGM ayant une matrice génératrice régulière présentent de médiocres capacités de corrections. Cependant une distribution irrégulière permet de créer des codes LDGM performants (cf. section 2.6.1).

2.5.5 Quelques outils associés aux codes LDPC

De nombreuses techniques ont été élaborées afin d'optimiser les codes LDPC et d'évaluer leurs performances. Cette section se limitera à une présentation de deux approches incontournables dans le domaine des codes LDPC.

Evolution de densité

Les performances des codes LDPC irréguliers dotés d'un algorithme de décodage itératif ont été mises en évidence dans [62]. Richardson et. al ont alors présenté une méthode [94] pour calculer les meilleures distributions de degrés. Cette méthode porte le nom d'évolution de densité ou "Density Evolution" (DE).Elle consiste en la recherche de distributions des degrés lignes et colonnes optimales pour le décodage itératif. Pour effectuer cette étude, il faut faire l'hypothèse que le graphe associé ne contient aucun cycle et donc qu'il est de taille infinie (le code est donc de longueur infinie), ceci afin que l'information reçue par un noeud soit indépendante de sa propre information.

Ces distributions sont déterminées par le couple de polynômes (λ,ρ) (cf section 2.5.1) :

$$\lambda(x) = \sum_{n\geq 1} \lambda_n x^n$$
$$\rho(x) = \sum_{n\geq 1} \rho_n x^n$$

où λ_n et ρ_n sont les fractions des arêtes connectées respectivement aux noeuds symboles et aux noeuds de contrôle de degré n.

Soit x_l la probabilité qu'à l'étape l du décodage itératif, un noeud variable envoie un effacement à un noeud de contrôle. Il a été montré [61] que x_l vérifie la relation de récurrence :

$$x_{l+1} = p\lambda(1-\rho(1-x_l)) \tag{2.16}$$

où p est la probabilité d'effacement du canal.

On définit le seuil ("threshold") p^*, qui est la probabilité d'effacement maximum du canal, pour laquelle la probabilité d'effacement des symboles peut être rendue arbitrairement petite en augmentant le nombre d'itérations autant que nécessaire. Ainsi p^* est le p maximum pour lequel $x_l \to 0$ quand $l \to \infty$. De manière équivalente on peut exprimer p^* comme le p maximum, pour lequel :

$$\rho(1-p\lambda(x)) > 1-x, \quad \forall x \in]0,1] \tag{2.17}$$

Soit f la fonction qui associe aux polynômes des distributions, le seuil p^* :

$$\begin{array}{rccl} f: & P[x]\times P[x] & \to & [0,1] \\ & (\lambda,\rho) & \to & f(\lambda,\rho) = p^* \end{array}$$

Puisque nous cherchons à construire un code permettant de transmettre sur le canal le plus mauvais possible, pour un rendement donné R, il nous faut trouver un couple de distribution pour lequel p^* soit maximum. Le problème se réduit donc à maximiser une fonction ayant pour variables les coefficients des polynômes λ et ρ sous la contrainte

$$R = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}$$

Cette optimisation peut être effectuée à l'aide d'un algorithme d'évolution différentielle [110].

L'évolution de densité est donc en théorie un outil permettant de construire des codes LDPC optimaux pour le décodage itératif. Cependant l'hypothèse forte d'un code de longueur infinie est une limite à cette approche. En effet lorsqu'une distribution optimale, au sens de la DE, est utilisée pour construire un code LDPC de longueur finie, la capacité de correction observée n'est pas toujours celle donnée par la DE. De plus en longueur finie, il est parfois possible d'obtenir de meilleures performances en choisissant une distribution différente de celle donnée par la DE.

Stopping Sets

Contrairement à l'évolution de densité, l'approche qui va suivre concerne les codes LDPC de longueur finie. Commençons par donner la définition d'un Stopping Set.

Définition Soit un code LDPC et son graphe de Tanner. Soit $\mathscr{V}$ un ensemble de noeuds variables et $\mathscr{S}$ l'ensemble de noeuds de contrôle qui sont adjacents à $\mathscr{V}$. $\mathscr{S}$ est appelé le voisinage de $\mathscr{V}$. Un ensemble de noeuds variables $\mathscr{V}$ est un *stopping set* du graphe de Tanner si tous les noeuds de contrôle qui sont dans son voisinage $\mathscr{S}$ sont connectés à au moins deux noeuds variable dans $\mathscr{V}$.

Un "stopping set" correspond à un schéma d'effacements qui met le décodeur itératif dans un état de blocage. Supposons que tous les symboles d'un stopping set sont effacés. Les symboles du stopping set ne pourront pas être reconstruits, car les noeuds de contrainte auxquels ils sont adjacents seront dans l'impossibilité de déduire leur valeur. En effet au moins deux symboles sont effacés dans leur voisinage, alors qu'une reconstruction n'est possible que si un seul symbole est effacé.

La figure 2.11 montre un "stopping set" de taille 3. En effet les noeuds symboles $V = \{v_2, v_3, v_5\}$ sont connectés uniquement aux noeuds de contrôle c_1, c_2, c_3, et chacun de ces noeuds contrôle est connecté à 2 symboles de V.

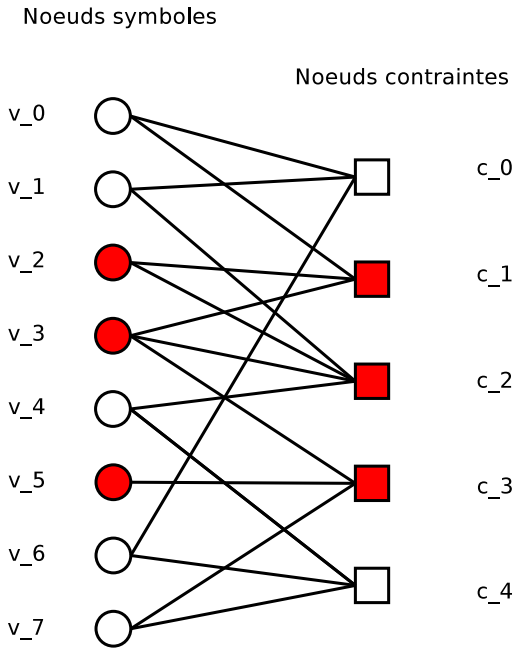


FIG. 2.11 – "Stopping set" de taille 3 dans un graphe de Tanner.

L'ensemble de ces stopping sets, et en particulier la taille du plus petit stopping set [32], va influencer la capacité de correction du décodage itératif sur le canal à effacements. Plus particulièrement, le plancher d'erreur, ou "error-floor" (voir section 3.2.2), dépendra de la taille des "stopping set". La longueur du plus petit cycle dans un graphe d'un code LDPC est habituellement appelée "girth".

A partir des caractéristiques d'un ensemble de codes LDPC, il est parfois possible de calculer la distribution de la taille des stopping set, appelé énumérateur. Cet énumérateur permet alors de calculer la probabilité que le décodage échoue en fonction de la probabilité d'effacement du canal [32]. Cette approche a l'avantage de pouvoir déterminer une borne inférieure sur la probabilité d'erreur des codes de longueur finie. Il faut noter que les énumérateurs s'obtiennent à partir d'ensembles de code LDPC et non pas à partir d'un code en particulier. Ainsi, seule une valeur moyenne pourra être obtenue pour un ensemble de codes construits à partir de la même méthode.

2.6 Codes sans rendement

Contrairement aux codes présentés précédemment, les codes sans rendement (“rateless”), également appelés codes fontaine, sont capables de produire à partir d’un ensemble fini de symboles source, une quantité *potentiellement infinie* de symboles de redondance.

Sur le sens du terme *infini* : en théorie ces codes sont capables de produire un nombre *potentiellement infini* de symboles. Cependant il existe des limites à cette quantité :

- Premièrement, les symboles produits par le code sont des combinaisons linéaires des symboles source. Ces symboles sources étant en nombre fini, le nombre de combinaisons de ces symboles sources est lui aussi fini. Pour un code binaire de dimension k, le nombre de combinaisons linéaires différentes est 2^k.
- Deuxièmement, en pratique les identifiants des symboles sont stockés dans des entiers de taille limitée. Et c’est cela qui contraint le plus le nombre de symboles. Par exemple, dans le RFC5053 [60] spécifiant une implémentation des codes Raptor [103], l’identifiant des symboles est stocké sur 16 bits, ce qui limite le nombre de symboles à $2^{16} = 65536$.

Même si il est vrai que le nombre de symboles que peuvent produire ces codes peut être extrêmement grand, il serait plus juste de parler de codes à rendement très faible (“very small rate”), plutôt que de codes sans rendement. Une autre façon de les définir serait de les considérer comme des codes sans rendement prédéfini.

2.6.1 Codes LT

C’est en 2002 que M. Luby présenta [58] une classe de codes non-systématiques sans rendement, supportant la création à la volée des symboles de redondance, qu’il baptisa codes “Luby Transform” (LT).

Chaque symbole de redondance est obtenu en sommant d symboles sources choisis aléatoirement, où d, le degré du symbole de redondance, dépend d’une distribution particulière (cf. figure 2.12). Ainsi les symboles de parité peuvent être produits de manière indépendante. En pratique, on utilise un PRNG (générateur pseudo-aléatoire de nombre) et l’identifiant du symbole de redondance pour calculer le degré du symbole et l’identité des symboles sources utilisés pour construire le symbole de redondance.

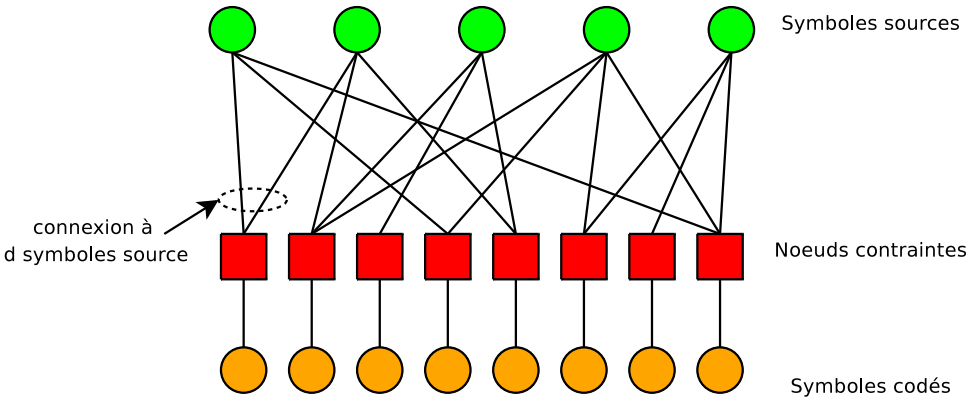


FIG. 2.12 – Exemple de graphe d’encodage d’un code LT

Lorsqu'ils sont décodés avec un décodeur itératif, les codes LT sont des codes asymptotiquement optimaux quel que soit le canal à effacements considéré. Cette propriété est vraie à condition que la distribution des degrés soit correctement choisie. Dans [58], Luby propose l'utilisation de la distribution robuste Soliton ou RSD ("Robust Soliton Distribution").

Le degré moyen est alors de $\log(k)$, la complexité d'encodage est donc de $O(\log(k))$ opérations par symbole. Grâce à cette distribution les codes LT peuvent décoder en moyenne à partir de $k+\log^2(k)/\sqrt{k}$ symboles en utilisant un décodeur itératif qui a alors un coût de l'ordre de $O(k\log(k))$. Le nombre de symboles nécessaires pour décoder tend vers k quand la dimension du code tend vers l'infini. En d'autres termes les codes LT sont asymptotiquement parfaits.

On peut remarquer que les codes LT sont en fait des *codes LDGM irréguliers non-systématiques dont les lignes de la matrice génératrice sont produites à la volée et possèdent un degré moyen proportionnel à* $\log(k)$.

2.6.2 Codes Raptor

Afin de pallier à certains inconvénients des codes LT, tels que le caractère non systématique, les complexités d'encodage et de décodage super-linéaire ($O(\log k)$ opérations par symbole, soit $O(n\log k)$ pour l'encodage de n symboles) et les performances médiocres pour les petites dimensions, A. Shokrollahi a proposé [103] de concaténer un précode avec un code LT, créant ainsi les codes Raptor. Les codes Raptor sont considérés par beaucoup comme les meilleurs codes à effacements connus à ce jour. A l'instar des codes LT, ils sont capables de produire à la volée les symboles de redondance, mais contrairement à leurs prédécesseurs, ils peuvent être rendus systématiques. Ces codes font l'objet de plusieurs brevets et ont été standardisés au sein de l'IETF [60], du 3GPP [4] et du DVB [3].

Cette section provient de l'étude que nous avons effectuée sur les codes Raptor, lors du développement d'un codec Raptor dans le cadre d'un contrat avec le CNES[1].

Structure des codes Raptor

Comme indiqué plus haut, les codes Raptor sont composés de la mise en série de deux codes, un code bloc linéaire à rendement élevé appelé *précode*, et un code LT dont la distribution des degrés est tronquée à un degré maximal D.

Nous allons décrire plus en détail les codes Raptor tels qu'ils sont spécifiés dans le RFC 5053 [60]. Le précode va encoder les K symboles source, notés C', en L symboles intermédiaires, notés C. Le code LT encodera alors en prenant comme source les L symboles intermédiaires, C. La figure 2.13 présente la structure de l'encodeur du code Raptor tel qu'il est spécifié dans le RFC 5053. Nous allons maintenant détailler les deux codes composant le code Raptor.

Précode : Le précode est un code par bloc à rendement élevé, qui doit être capable de reconstruire les symboles sources C à partir d'un sous-ensemble des symboles intermédiaires. De plus afin de rendre le code Raptor systématique ce code doit effectuer un encodage LT inverse. Dans [103], plusieurs types de précodes sont considérés : des codes LDPC, des codes de Hamming. Dans le cas du RFC5053, le précode est composé de :

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

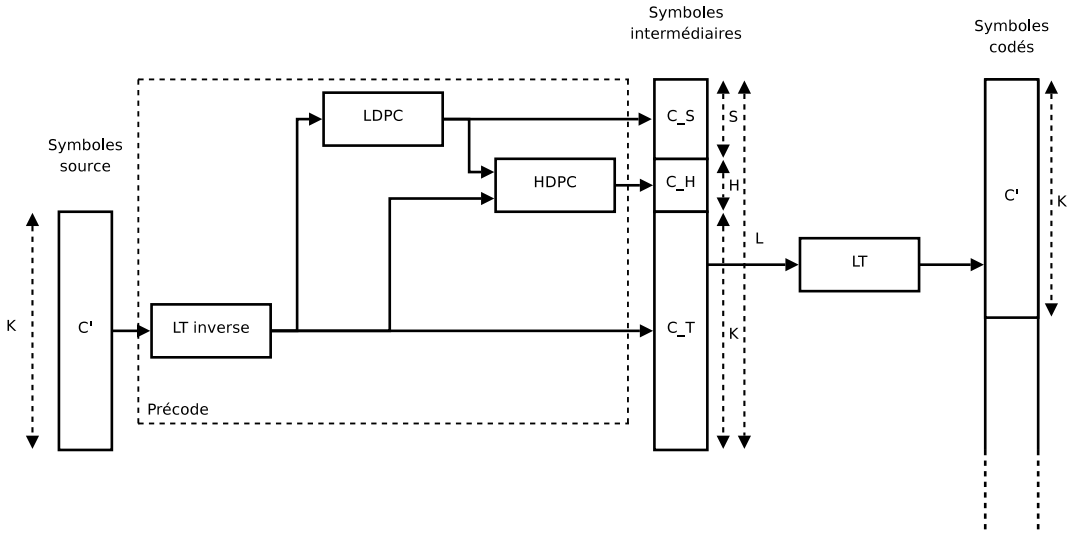


FIG. 2.13 – Structure de l'encodeur des codes Raptor

- un code LDPC qui produira S symboles intermédiaires, appelés symboles LDPC et notés C_S ; l'utilité de ce composant réside dans sa capacité à corriger efficacement les erreurs avec un décodeur itératif [103] ;
- un code de Hamming qui produira H symboles intermédiaires, appelés symboles "Half" et notés C_H ; ce composant est utilisé pour limiter l'effet des "stopping sets" de petite taille [103] ;
- un code LT inverse qui produira les K symboles intermédiaires restant, que l'on appellera symboles "LT-inverse" et que l'on notera C_T ; ce code permet de garantir le caractère systématique du code Raptor.

Ce choix de code répond aux spécifications du précode (codage LT inverse et code à effacements).

Code LT : Le code LT utilisé dans le code Raptor ne suit pas exactement la construction donnée par Luby [58]. En effet la distribution des degrés est différente, et il ne s'agit plus de la distribution Soliton. Afin de réduire la complexité d'encodage et de décodage ($O(K\log(K))$ pour les codes LT), le degré moyen des noeuds du code LT est réduit à une valeur faible et ne dépendant pas de K (contre $O(\log K)$ dans le cas des codes LT). Cela va bien évidement dégrader les performances du code, et en particulier les $K+\log^2(K)/\sqrt{(K)}$ symboles codés ne seront plus suffisants en moyenne pour permettre un décodage complet. Mais grâce au précode, qui est un code à effacements, un décodage partiel du code LT sera suffisant pour permettre un décodage complet du code Raptor.

Encodage des codes Raptor

L'encodage Raptor s'effectue en deux étapes : premièrement l'encodage du précode permet de créer les symboles intermédiaires, puis l'encodage LT produit les symboles codés Raptor. L'encodage LT ayant été détaillé en section 2.6.1 nous ne reviendrons pas sur cette étape.

Le précodage consiste à générer les L symboles intermédiaires C à partir des K symboles sources C'. Soit D le vecteur résultant de la concaténation du vecteur des symboles

sources C' avec $S+H$ symboles nuls. L'encodage du précode est effectué en multipliant le vecteur D par la matrice génératrice étendue du précode $\widetilde{G}_{\text{précode}}$. Cette matrice est définie comme étant l'inverse d'une matrice A représentée sur la figure 2.14. La figure 2.15 montre un exemple de matrice A.

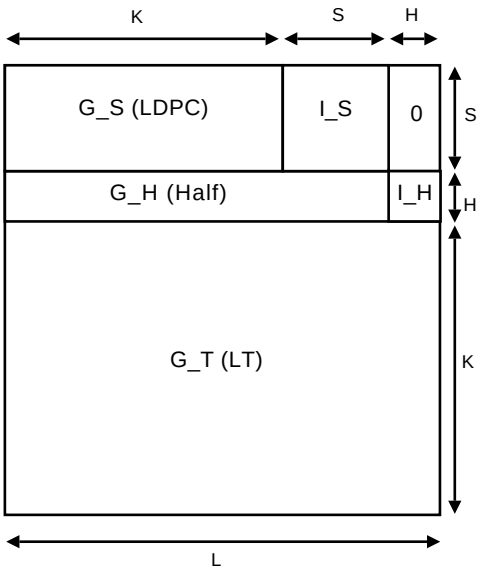


FIG. 2.14 – Structure de la matrice A du précode

Les symboles intermédiaires C doivent être tels que $AC = D$. En effet après l'encodage LT des symboles intermédiaires (afin de produire les symboles de redondance Raptor), on doit retrouver les symboles sources :

$$\begin{array}{rcl} AC & = & D \\ \begin{pmatrix} G_{\text{S}} & I_S & 0 \\ G_{\text{H}} & & I_H \\ & G_{\text{T}} & \end{pmatrix} \begin{pmatrix} C_{\text{T}} \\ C_{\text{S}} \\ C_{\text{H}} \end{pmatrix} & = & \begin{pmatrix} 0 \\ 0 \\ C' \end{pmatrix} \end{array}$$

Ainsi on a :

$$\begin{array}{rcl} G_{\text{S}} C_{\text{T}} & = & C_{\text{S}} \\ G_H \begin{pmatrix} C_{\text{T}} \\ C_{\text{S}} \end{pmatrix} & = & C_{\text{H}} \\ G_T \begin{pmatrix} C_{\text{T}} \\ C_{\text{S}} \\ C_{\text{H}} \end{pmatrix} & = & C' \end{array}$$

En choisissant cette matrice, l'encodage LT inverse ainsi que la production de symboles Half et LDPC sont bien effectués. La matrice génératrice du précode est alors $\widetilde{G}_{\text{precode}} = A^{-1}$.

Comme $\widetilde{G}_{\text{précode}}$ est définie comme l'inverse de A, encore faut-il que A soit inversible. En effet A est une matrice carrée binaire de taille $L \times L$, et n'a à priori aucune raison d'être

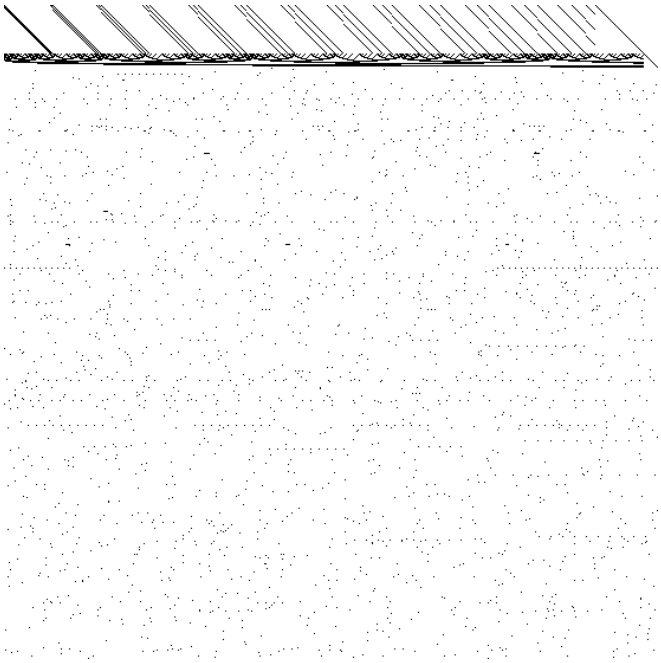


FIG. 2.15 – Exemple de matrice A du précode ($k = 500$, $L = 553$)

inversible. Les $S + H$ premières lignes de A étant fixées par les codes Half et LDPC, il ne reste plus que la partie correspondant à la matrice génératrice du code LT pour adapter la matrice A. L'astuce afin d'obtenir une matrice A inversible repose dans le choix de la matrice G_{LT} qui est générée à partir d'un PRNG. Le RFC 5053 fournit une table de graines d'initialisation du PRNG du code LT pour chaque dimension du code Raptor. Selon toute vraisemblance, cette table a été définie de manière à ce que les graines soient telles que la matrice A correspondante soit toujours inversible.

Décodage des codes Raptor

Le décodage des codes Raptor consiste à récupérer les K symboles sources à partir d'un ensemble de symboles codés. Ce décodage s'effectue en deux étapes : décodage du code code LT, puis le décodage du précode.

Comme pour les codes LT, il est proposé dans le papier original de Shokrolahi de décoder les codes Raptor avec un décodeur itératif. Cependant, les capacités de correction de ces codes, lorsqu'ils sont décodés avec un algorithme itératif, se dégradent rapidement à mesure que leur dimension diminue. Le décodage ML est donc considéré comme pertinent pour des codes dont la dimension est de l'ordre de quelques milliers de symboles, alors que pour des dimensions de l'ordre de plusieurs dizaines de milliers de symboles le décodage itératif sera préféré [103].

La technique de décodage proposée dans le RFC5053, qui considère des dimensions inférieures à $K_{max} = 8192$, est donc un décodage ML. Cette résolution est effectuée en deux étapes : d'abord on décode le code LT à l'aide d'une variante du pivot de Gauss [104], puis on déduit les symboles sources à partir des symboles intermédiaires par une multiplication matricielle.

Le système du code LT possède L inconnues, i.e. les symboles intermédiaires. Chaque symbole codé reçu permet d'ajouter une contrainte à ce système. De plus les symboles

intermédiaires ne sont pas indépendants. En effet le précode définit des relations entre ces symboles, ce qui nous permet d'ajouter $S+H$ contraintes supplémentaires. Ainsi, à partir de K symboles codés reçus, le système à L inconnues possède $K+S+H=L$ contraintes. Un décodage ML est donc envisageable si la matrice est de rang plein. Une fois les symboles intermédiaires reconstruits, une multiplication du vecteur des symboles intermédiaires par la matrice A permet de retrouver les symboles sources.

Capacités de correction des codes Raptor

Si les performances théoriques des codes Raptor sont largement accessibles, les résultats précis sur les performances pratiques des codes Raptor sont rares.

D'après une présentation de Shokrolahi sur les codes Raptor, un décodage ML sur un code de dimension $k=1024$ permet d'obtenir pour un surcoût en nombre de symboles $\varepsilon=1\%$ et une probabilité d'erreur bloc $P_{\text{erreur bloc}}<10^{-3}$. Une comparaison entre des codes Raptor à rendement fixe et un ensemble de codes "Generalized Repeat-accumulate" est présentée dans [82]. Les résultats montrent qu'avec un décodage ML, ces deux classes de codes possèdent des capacités de correction sensiblement équivalentes pour l'ensemble des paramètres considérés.

Dans le chapitre 4, nous présenterons quelques résultats sur les capacités de correction des codes Raptor, obtenues à partir de l'implémentation que nous avons réalisée suivant les spécification du RFC 5053.

Performances des codes Raptor en termes de vitesse

Concernant la complexité de décodage, les résultats se font aussi rares, rendant difficiles les comparaisons avec d'autres codes. On peut trouver une étude du décodage itératif pour des codes LT et Raptor ayant des dimensions de l'ordre de plusieurs centaines de milliers de symboles [24]. Cette étude confirme l'avantage en termes de complexité des codes Raptor sur les codes LT, et le fait que le temps de décodage dépend linéairement de la dimension du code.

Dans l'article originale [103] on peut lire la phrase suivante : *"The Raptor implementation of Digital Fountain reaches speeds of several gigabits per second, on a 2.4-GHz Intel Xeon processor, while ensuring very stringent conditions on the error probability of the decoder, even for very short lengths."*. La dimension du code, la taille des symboles considérée, le type de décodage et la signification de "several Gbps" n'étant pas précisés, il est difficile d'interpréter cette information.

Afin de réduire la complexité du décodage ML, il est proposé d'utiliser des techniques brevetées [104] dites d'*inactivation*. Ces techniques réduisent la complexité du décodage ML en tirant profit du caractère creux du système. Ces techniques semblent réduire significativement la complexité du décodage [83]. Qualcomm annonce même une complexité de décodage linéaire (qui nous semble plutôt être une complexité super-linéaire).

Les codes Raptor sont donc d'excellents codes à effacements, qui en plus d'avoir de très bonnes capacités de corrections, possèdent la capacité de produire un nombre potentiellement infini de symboles codés. Si le décodage itératif permet d'exploiter efficacement ces codes pour de grandes dimensions, le décodage ML reste nécessaire pour conserver de bonnes capacités de correction sur des codes de petites tailles (quelques milliers de symboles sources).

Chapitre 3

Méthodes et outils d'évaluation des codes à effacements

Contents

Ce chapitre présente une méthodologie d'évaluation des codes à effacements. Nous verrons sur la base de quelles métriques ces évaluations sont effectuées, et nous présenterons les outils qui permettent d'obtenir ces résultats.

3.1 Introduction

Les travaux présentés dans cette thèse étant des propositions de codes et d'algorithmes de décodage performants, il convient de décrire dans un premier temps les métriques sur lesquelles reposent ces évaluations, ainsi que les méthodes utilisées pour les obtenir.

Le domaine des codes à effacements connaît depuis plusieurs années un essor particulier. Un grand nombre de travaux ont vu le jour présentant de nouveaux codes et de nouveaux algorithmes de décodage. Cependant il est parfois, si ce n'est souvent, difficile de comparer deux propositions. En effet, les résultats présentés n'utilisent pas toujours les mêmes métriques ni les mêmes conditions expérimentales, rendant difficile une comparaison équitable.

Les méthodes d'évaluation peuvent être catégorisées en deux ensembles : les méthodes dites analytiques et les méthodes basées sur la simulation. Avec une méthode analytique, il

faut modéliser le système dans une premier temps, puis résoudre le problème posé par des méthodes de calcul analytiques. A l'opposé, les méthodes reposant sur la simulation vont se baser sur des mesures d'un ensemble de résultats d'expériences, desquelles sera déduit le comportement moyen du système.

Nous allons dans un premier temps présenter les métriques les plus communément utilisées pour évaluer les codes à effacements, puis nous introduirons les méthodes permettant d'obtenir ces métriques.

3.2 Métriques de capacités de correction

Le rôle d'un code correcteur étant de corriger les erreurs et/ou les effacements, les capacités de correction d'un code constituent donc la première classe de métriques. Il faut noter que la pertinence des ces métriques dépend fortement de l'application visée et de la façon dont est utilisé le code.

3.2.1 Distance à la capacité

D'un point de vue théorique, on désire qu'un code soit le plus proche possible de la capacité du canal. Un code atteignant la capacité du canal est appelé code parfait (voir section 2.1). Sur le canal à effacements, un code parfait est un [n,k]-code qui est capable de corriger $n-k$ effacements, i.e. il atteint la borne de Singleton. La première façon d'évaluer un code est de mesurer la distance qui le sépare du code parfait. Cette grandeur est appelée la distance à la capacité, $\triangle$, et elle se calcule comme la différence entre la capacité du code, C_{code}, et la capacité du canal, C_{canal}. Dans le cas du canal binaire à effacements la capacité est égale à $C_{\text{canal}} = 1-p$, où p est la probabilité d'effacement sur le canal. La capacité du code est égale à $1-P^*$, où P^* est la probabilité d'effacement maximum corrigible par le code. Il en découle une expression de la distance à la capacité :

$$\triangle = C_{\text{code}} - C_{\text{canal}} = (1-P^*) - (1-p) = p - P^* \tag{3.1}$$

La distance à la capacité permet d'évaluer les capacités de correction du code du point de vue de la théorie de l'information. On déduira souvent cette grandeur à partir des probabilités d'effacements qui peuvent être corrigées par le code. Elle pourra donc être obtenue de manière analytique ou à partir de simulations.

3.2.2 Probabilité d'erreur de décodage

En pratique on s'intéresse souvent à la fiabilité d'un système et en particulier à la probabilité qu'il ne remplisse pas sa fonction. Ainsi pour un code à effacements on s'intéresse à la probabilité que le décodage échoue. Sur les canaux à effacements on considère qu'un décodage a échoué lorsqu'il subsiste un symbole (source) effacé n'ayant pas été reconstruit. Dans les cas où un décodage partiel peut être exploité, la probabilité d'effacement symbole par symbole après le décodage pourra également être considérée.

Ces probabilités d'erreur de décodage sont le plus souvent exprimées en fonction de la probabilité d'effacement du canal, le rendement étant lui fixé. Cependant, on peut aussi bien les exprimer en fonction du taux d'effacements du canal. Le taux d'effacements détermine le nombre exact d'effacements pour une expérience, alors que la probabilité

d'effacement va donner le paramètre du tirage aléatoire déterminant pour chaque symbole, s'il est effacé ou non.

Comme indiqué plus haut, les métriques basées sur les probabilités d'erreurs servent à mesurer la qualité du service fourni à la couche supérieure. Elles seront particulièrement appropriées pour évaluer les performances d'un système, dans le cas où la probabilité de panne est cruciale et dans le cas où la qualité du canal est connue à l'avance.

Probabilité d'erreur bloc : La probabilité que le décodage échoue est appelé probabilité d'erreur bloc ou BER ("Block Error Rate"). Elle correspond à la probabilité que le décodage ne soit pas complet (s'il subsiste des symboles effacés non reconstruits), étant donné un ensemble de symboles reçus (voir figure 3.1).

Probabilité d'erreur symbole : Comme nous l'avons fait remarquer plus tôt, un échec du décodage ne signifie pas pour autant que les données ne seront pas utilisables. En effet, si le code est systématique, une partie des symboles sources a pu être reçue. De plus certains décodeurs, tels que les décodeurs itératifs, permettent une reconstruction progressive des symboles à mesure que les symboles sont reçus. Même si le décodage s'est terminé par un échec au niveau du bloc, une partie des symboles reconstruits pourront être exploités [78].

Terminologie associée aux courbes de probabilités d'erreur : Pour les codes parfaits la probabilité d'erreur de décodage se déduit directement de la dimension du code et du nombre de symboles reçus. Il existe donc une frontière nette entre la zone de succès de décodage et la zone d'échec de décodage. Dans le cas général, cette frontière n'est pas aussi nette, et les courbes représentant la probabilité d'erreur de décodage en fonction de la qualité du canal possèdent une forme particulière. La courbe (voir figure 3.1) est habituellement découpée en trois zones :

- lorsque la probabilité d'effacement du canal est supérieure à la capacité du code, les effacements sont trop nombreux pour être corrigés, la probabilité d'erreur de décodage est donc égale à 1.
- le plancher d'erreur ("error floor" en anglais), qui correspond à la zone où la probabilité d'effacement est faible. Cette zone montre le très faible taux d'échec de décodage "intrinsèque" au code, puisque persistant mêmes si les conditions du canal s'améliorent. On cherchera à construire un code avec un plancher le plus bas possible. On notera que la zone de plancher n'est pas toujours visible, puisque la probabilité d'erreur dans cette zone peut être trop faible pour être capturée par les expériences.
- la zone de "waterfall" qui se situe entre les deux zones précédentes. Dans cette zone, la probabilité d'erreur bloc évolue rapidement des valeurs faibles du plancher d'erreur à la valeur maximum 1. Plus cette zone sera proche de la limite théorique, meilleur sera le code.

3.2.3 Ratio d'inefficacité de décodage

Dans la section précédente, nous avons considéré la probabilité d'erreur en fonction de la quantité d'informations reçue. Dans cette section nous allons considérer la quantité d'informations nécessaire pour permettre le succès du décodage. Pour cela il faut utiliser le ratio d'inefficacité de décodage, qui est défini comme le rapport entre la quantité moyenne

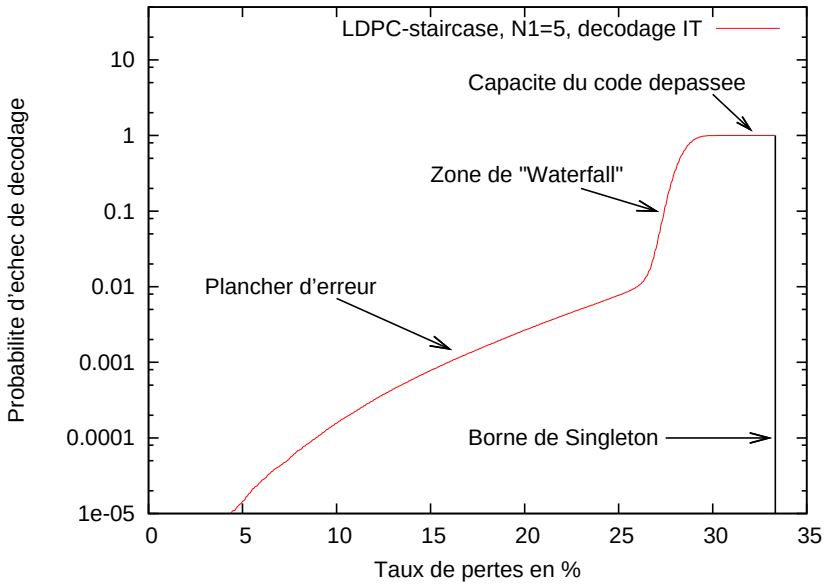


FIG. 3.1 – Zones remarquables sur une courbe de probabilité d'erreur bloc en fonction du taux d'effacements ($k = 1000$, $R = 2/3$).

de symboles nécessaire pour décoder et le nombre de symboles sources, i.e. la dimension du code, k. Le ratio d'inefficacité s'exprime comme :

$$Ineff = \frac{Nb_{\text{Symboles Nécessaires}}}{k} \tag{3.2}$$

Plus le ratio d'inefficacité est bas, meilleurs sont les capacités de correction (un ratio d'inefficacité égale à 1 étant celui d'un code parfait). A partir du ratio d'inefficacité on peut définir le surcoût en transmission ("overhead" en anglais) noté ε qui correspond à la distance entre le ratio d'inefficacité du code et celui d'un code parfait. La ratio d'inefficacité d'un code parfait étant égal à 1, le surcoût s'exprime alors :

$$\varepsilon = \text{Ineff} - 1 \tag{3.3}$$

Le décodage réussira donc en moyenne à partir de $k(1+\varepsilon)$ symboles reçus. Le surcoût en symboles pourra être aussi considéré. Il s'agit du nombre de symboles en plus de k qui sont nécessaires au décodage, nombre qui est égal à $k\varepsilon$.

3.3 Métriques de complexité algorithmique

La complexité d'une solution va déterminer la faisabilité de son utilisation en pratique. Les codes correcteurs sont souvent un composant branché en série dans la pile de communication : si leur débit est trop faible, ils peuvent devenir un goulot d'étranglement qui ralentira l'ensemble du système. De plus, on voit apparaître des plate-formes faisant usage des codes à effacements et qui possèdent des capacités réduites en termes de puissance de calcul, de mémoire, ou encore d'énergie (par exemple dans le "Digital Video Broadcasting - Satellite Handheld", en français Diffusion Vidéo Numérique - Satellite Portable (DVB-SH) où les terminaux sont des récepteurs mobiles).

L'implémentation des codes correcteurs peut aussi bien être faite en logiciel qu'en matériel. Les codes à effacements de niveau applicatif étant généralement implémentés en logiciel, nous nous concentrerons sur les aspects critiques de la complexité pour une telle implémentation.

3.3.1 Complexité théorique

Afin d'estimer le coût d'algorithmes, il est courant d'utiliser la complexité théorique. Cette complexité exprime le nombre d'opérations effectuées par l'algorithme en fonction de la taille des entrées. Elle est obtenue en décomposant l'algorithme en opérations de base comme les additions et les multiplications. On utilise habituellement la notation en $O(.)$ pour cette complexité. La fonction f est un grand "O" de g si :

$$\lim_{n \to +\infty} \sup \left| \frac{f(n)}{g(n)} \right| < \infty \tag{3.4}$$

On dit alors que f est dominée par g. La fonction de coût d'un algorithme est alors comparée avec une fonction usuelle dont on connaît le comportement lorsque n grandit. Ainsi on dira d'un algorithme qu'il a un coût constant si $f(n) \in O(1)$, linéaire si $f(n) \in O(n)$, quadratique si $f(n) \in O(n^2)$.

La complexité théorique nous donne un aperçu de la façon dont la complexité de l'algorithme va évoluer en fonction de la taille de l'entrée. Mais cette complexité ne prend pas en compte le coût des opérations de base. Elle constitue donc une première évaluation du coût de l'algorithme, mais ne sera la plupart du temps pas suffisante pour déterminer si un algorithme d'encodage ou de décodage est plus rapide qu'un autre.

3.3.2 Vitesse et débit

Une manière plus pratique d'évaluer la complexité d'un algorithme, est de mesurer son temps d'exécution. Dans le cas des composants des systèmes de communication on considère plus généralement le débit. D'une manière générale, le débit (moyen, minimum ou maximum) atteignable par un encodeur ou un décodeur, est une métrique qui va déterminer si une implémentation d'un algorithme est utilisable ou non.

Lorsque deux solutions sont comparées sur la base de leurs débits pour un ensemble de paramètres donnés (longueur et dimension du code, taille des symboles, paramètres du canal), il faut prendre la précaution d'effectuer les expériences sur des plate-forme identiques, d'autant plus que les algorithmes peuvent être optimisés pour certaines architectures de microprocesseurs (la taille des mémoires caches a en particulier une très grande influence).

Les métriques basées sur la vitesse et le débit de décodage ont l'avantage de présenter la complexité en pratique. Cependant elles nécessitent l'implémentation d'un codec optimisé, ce qui peut demander une quantité importante de travail, là où la complexité théorique ne nécessite qu'une analyse de l'algorithme.

3.3.3 Consommation mémoire

Outre la complexité en calcul d'un algorithme il est possible de considérer sa consommation mémoire. Cette complexité va représenter la quantité d'information maximale qui sera stockée par l'algorithme. Même si les capacités en mémoire vive des machines

actuelles augmentent significativement, les codes AL-FEC travaillent sur des quantités de données qui peuvent être très volumineuses (plusieurs Go pour des fichiers vidéo). De plus les systèmes à environnement contraint tels que les terminaux mobiles possèdent des capacités mémoire réduites qui doivent souvent être partagées avec d'autres applications (par exemple le décodage d'un flux vidéo).

Si la consommation totale de mémoire est importante, les architectures actuelles possèdent une mémoire hiérarchisée dont les capacités et les temps d'accès varient de plusieurs ordres de grandeur. L'utilisation intelligente de ces espaces mémoire permet des gains significatifs en termes de vitesse.

3.4 Méthodes d'évaluation de performances

Les métriques de performances peuvent s'obtenir de deux façons : de manière analytique, ou bien à partir de simulations.

3.4.1 Méthodes analytiques

Les méthodes analytiques consistent à utiliser la connaissance du code afin de modéliser son décodage, et à en déduire certaines métriques. Ce types de méthodes dépend fortement du code étudié. Pour certains codes (par exemple les codes Reed-Solomon) , des arguments théoriques d'algèbre linéaire sont souvent suffisants pour déterminer les caractéristiques d'un code donné. Alors que pour les codes à matrice creuse décodés par un algorithme itératif, des outils statistiques sont nécessaires. L'analyse des stopping sets et l'analyse par évolution de densité des codes LDPC (voir section 2.5.5) sont deux exemples de méthodes analytiques.

L'étape de modélisation nécessite parfois des hypothèses qui vont donner au résultat un caractère général. Par exemple pour l'analyse par évolution de densité des codes LDPC, il faut faire l'hypothèse que le code est de longueur infinie. Pour l'énumération des stopping sets des codes LDPC on considère des codes de longueur finie, mais on considère des ensembles de codes plutôt qu'un code en particulier.

Les méthodes analytiques constituent un ensemble d'outils pour évaluer les codes sous un ensemble d'hypothèses plus ou moins réalistes. Elles ont l'avantage d'être calculables rapidement permettant une optimisation des performances du code en fonction de ses différents paramètres. Cependant elles demeurent la plupart du temps insuffisantes pour déterminer les performances dans un environnement réel.

3.4.2 Simulations

Afin d'évaluer les performances effectives d'un système dans des conditions précises, il faut faire appel aux techniques de simulations. Ces méthodes reposent sur la mesure du comportement d'un système dans des conditions particulières. En particulier la dimension, la longueur du code et la taille des symboles doivent être fixées. Ces résultats peuvent être exploités directement, ou alors être produits en plus grand nombre afin d'en déduire le comportement moyen du système.

Simulations de type Monte-Carlo

Une méthode de Monte Carlo est une technique calculant une grandeur à partir de processus aléatoires. Ces méthodes reposent sur l'exécution d'un grand nombre d'expériences et nécessitent donc une grande quantité de calculs. Les méthodes de Monte Carlo ont été introduites [73] pour résoudre des problèmes dont la résolution analytique était trop complexe. Ces méthodes reposent sur le principe suivant lequel l'observation d'un grand nombre de réalisations d'expériences permet de capturer le comportement moyen du système.

Afin d'estimer $\widetilde{g}$ la valeur moyenne d'une fonction g de variable aléatoire X, on effectue un échantillonnage en N points différents tirés au hasard. La technique la plus simple et la plus couramment utilisée est la technique basée sur la loi des grands nombres. Soit $\widetilde{g_N}$ la moyenne empirique obtenue à partir de N tirages aléatoires indépendants, elle s'écrit :

$$\widetilde{g_N} = \sum_{i=1}^{N} \frac{g(X_i)}{N}$$

La loi des grands nombres nous dit que :

$$\forall \varepsilon > 0, \lim_{N \to +\infty} \mathbf{P}(|\widetilde{g_N} - \widetilde{g}| \geq \varepsilon) = 0$$

Dit autrement, en augmentant le nombre de mesures de la fonction g on diminue la probabilité que la moyenne empirique s'éloigne au plus de ε de la valeur moyenne, $\widetilde{g}$.

A partir de cette technique il est possible de construire une méthode d'évaluation des performances d'un code. Soit f la métrique observée sur le code et soit X la variable aléatoire permettant de générer une transmission. Une transmission va comprendre un ensemble d'effacements, mais aussi un possible ordonnancement des symboles. On notera que la répartition des effacements entre les symboles sources et les symboles de redondance peut avoir un impact sur les performances [79]. En effectuant N expériences indépendantes, on obtient un ensemble de valeurs $F = \{f(X_i)\}_{0 \leq i \leq N}$. A partir de cet ensemble, on peut calculer la moyenne empirique, mais aussi des intervalles de confiance et plus généralement toute mesure de la statistique descriptive (médiane, variance, etc...).

Il est généralement admis que pour obtenir une précision à 10^{-b} près, il est nécessaire d'effectuer au moins 10^{b+2} expériences. Lorsque l'on s'intéresse à la probabilité d'erreur, il est souvent nécessaire d'obtenir une précision de 10^{-5}, ceci implique d'effectuer 10^7 expériences différentes (cela peut parfois représenter plusieurs jours de simulations sur une station de travail).

Modèle de canal et ordonnancement des symboles : Comme indiqué précédemment, l'identité des symboles effacés a un impact sur l'issue du décodage. En effet tout les symboles ne sont pas nécessairement équivalents, en particulier pour les codes non MDS. Pour un nombre d'effacements donné, des répartitions différentes pourront induire des issues du décodage totalement différentes. La répartition des effacements va dépendre de deux éléments : le modèle de perte du canal et l'ordonnancement des symboles lors de la transmission. Deux types de modèles de perte peuvent être considérés :

– les modèles sans mémoire, représentant des pertes indépendantes. La perte d'un symbole est indépendante des autres symboles et est uniquement déterminée par un tirage aléatoire suivant une certaine loi de probabilité.

– les modèles "Markoviens" représentant la dépendance entre les pertes. Dans ce type de modèle, la qualité du canal varie entre plusieurs états au cours de la transmission. La perte d'un symbole dépend alors de l'état courant du canal. Ces modèles sont habituellement représentés avec une chaîne de Markov. Le modèle le plus simple étant une chaîne à deux états : "mauvais" et "bon", auxquels on adjoint des probabilités de transitions. A partir de ce modèle et d'un générateur aléatoire, une suite d'états du canal est calculée et un schéma de pertes en est déduit. Ce genre de modèle est particulièrement utile pour modéliser les pertes en rafale. Une étude de ce types de canaux est présentée dans .

L'ordre de transmission des symboles peut être choisi par l'émetteur afin d'augmenter la robustesse du système. Trois modes de transmission pourront être considérés :

– La transmission dans un ordre séquentiel. Les symboles sont transmis dans leur ordre naturel à la sortie de l'encodeur. Lorsque l'encodeur est systématique, les symboles sources sont généralement envoyés avant les symboles de parité, mais l'inverse peut être considéré.
– La transmission dans un ordre aléatoire. Les symboles subissent une permutation aléatoire à la sortie de l'encodeur. L'avantage de ce type de transmission est que quel que soit le modèle de perte du canal, d'un point de vue du code, la répartition sera équivalente à un schéma de pertes indépendantes. Ceci est particulièrement utile pour réduire l'impact des pertes en rafale sur les codes qui y sont sensibles. En faisant l'hypothèse que ce type d'ordonnancement est utilisé on peut s'abstenir de considérer le modèle de perte, puisqu'ils seront tous équivalents.
– Une permutation déterministe des symboles, aussi appelé entrelacement. Un exemple d'entrelacement est d'envoyer dans un premier temps les symboles d'indice i tel que $i \bmod B = 0$ puis les symboles d'indice i tel que $i \bmod B = 1$, et ainsi de suite. Ainsi, si B est plus grand que la longueur des rafales de pertes, celles-ci seront dispersées sur l'ensemble des symboles.

Les simulations de type Monte-Carlo possèdent l'avantage d'être capables d'estimer les performances d'un code en particulier, avec une précision arbitraire. Cependant cette précision a un coût en temps de calcul, et la précision des estimations est souvent limitée par la puissance de calcul disponible. Il faut également noter que ces simulations reposent sur des modèles de transmission pouvant être trop grossiers pour capturer le comportement des transmissions réelles.

Exemple de méthode pratique de simulation de Monte Carlo

Nous détaillerons dans ce paragraphe une méthode efficace réduisant le nombre de simulation à effectuer. Supposons que nous souhaitons obtenir les capacités de correction du décodage ML avec une précision à 10^{-5} près, pour un code LDPC de dimension $k = 1000$ et un rendement $R = 2/3$. Il faut alors considérer le nombre minimal de symboles, n_{ML}, pour lequel le décodage ML termine avec succès. Cette quantité peut être encadrée par la dimension du code k et le nombre de symboles , n_{IT}, nécessaires au succès du décodage IT : $k \leq n_{ML} \leq n_{IT}$.

Pour s'abstenir de considérer le modèle de pertes, on utilise un ordre de transmission aléatoire. Un autre avantage de l'ordre de transmission aléatoire est que pour tester le succès du décodage avec n_r symboles reçus aléatoirement, il suffit de considérer les n_r premiers symboles de la transmission. Ainsi en incrémentant n_r on ajoutera, à l'ensemble des symboles reçus, un symbole choisi aléatoirement parmi les $n - n_r$ symboles restants.

Chaque expérience correspond à un ordre de transmission donné, et il faut trouver pour chacune de ces expériences le plus petit n_r tel que le décodage ML termine avec succès. Une solution triviale est de partir de n_{IT} (resp. k) et de décrémenter (resp. incrémenter) n_r, jusqu'à trouver la valeur limite, n_{ML}. Cette méthode nécessite $n_{IT} - n_{ML}$ (resp. $n_{ML} - k$) tentatives de décodage ML. Une autre solution est d'effectuer une recherche dichotomique. On prend comme intervalle de départ $[k, n_{IT}]$. A chaque itération on divise l'intervalle en deux et on effectue une tentative de décodage ML. Si le décodage réussit, alors n_{ML} se trouve dans l'intervalle inférieur, sinon n_{ML} se trouve dans l'intervalle supérieur. La recherche termine quand l'intervalle est réduit à un élément. Cette technique à un coût de $O(\log_2(n_{IT} - k))$. L'avantage de la deuxième technique sera d'autant plus important que la dimension du code sera grande.

Afin d'obtenir la précision souhaitée, il est nécessaire effectuer au moins 10^7 expériences (si on cherche une précision de 10^{-3} avec des conditions différentes. Chaque expérience correspond alors à un ordre de transmission différent, i.e il suffit de changer la graine du PRNG générant la transmission aléatoire. Une fois les expériences effectuées, nous avons 10^5 valeurs de n_{ML}. On calcule alors le ratio d'inefficacité comme étant : $Ineff = n_{ML}^{moyen}/k$. Il est également possible d'obtenir la probabilité d'échec de décodage en fonction du nombre de symboles reçus [1]. En effet, pour chaque n_r on compter le nombre d'expériences i tels que $n_{ML}^{(i)} > n_r$. Rapporté au nombre total d'expériences, on obtient la probabilité d'échec du décodage pour n_r symboles reçus.

Simulations sur traces réelles

Le meilleur moyen de déterminer les performances d'un système en situation réelle est de le tester dans de telles conditions. Sans pour autant l'implémenter et l'intégrer dans un système de communication, il est parfois possible de récupérer des traces provenant d'applications réelles. Par exemple, il est possible de déduire un schéma de perte en observant le flux de paquets à la sortie de la couche transport. Cette trace de pertes sera alors utilisée pour effectuer une simulation sur une plate-forme expérimentale.

Outre la faible charge de calcul, l'avantage des simulations sur traces réelles est qu'il n'est pas nécessaire de faire l'hypothèse d'un modèle de transmission. Néanmoins, il est nécessaire d'effectuer ces simulations sur des traces suffisamment longues et nombreuses de manière à limiter le biais provoqué par un cas particulier. Il est également à noter que ces tests nécessitent que le champs d'application soit connu. On étudie alors les performances (et l'adéquation) d'une solution de code correcteur avec un champs d'application donné et non les performances intrinsèques de ces codes.

3.5 Conclusions

Dans ce chapitre nous avons présenté des métriques permettant l'évaluation des codes à effacements. Un ensemble de méthodes et d'outils permettant de mesurer ces métriques ont été proposées. Il faut noter que les métriques à utiliser pour l'évaluation d'un système de codage dépendent fortement du cas d'application ainsi que des terminaux auxquels ils sont destinés.

[1] De la même manière on peut obtenir la probabilité d'échec de décodage en fonction du taux de perte du canal, en calculant le taux de perte correspondant aux nombre de symboles reçus.

Toutes ces métriques doivent nous permettre de répondre à la question : *des codes A et B, lequel est le meilleur ?* Ceci va bien souvent dépendre du contexte d'utilisation et des contraintes associées. Si nous sommes intéressés pas une seule caractéristique, le ratio d'inefficacité par exemple, le choix sera immédiat. Cependant, la plupart du temps le choix doit prendre en compte plusieurs critères et ces critères sont souvent interdépendants rendant le choix encore plus difficile.

Si seule la capacité de correction est prise en compte, les codes MDS tels que les codes Reed-Solomon s'imposent. Cependant si la complexité de la solution doit également être considérée, ces codes deviennent inutilisables à mesure que leur dimension augmente. Des concessions sur les capacités de correction pourront alors être faites afin obtenir une complexité inférieure, en choisissant des codes LDPC par exemple. Il n'existe donc pas un code universel permettant de répondre à toutes les situations et un compromis sur les performances est souvent nécessaire.

Première partie

Contributions sur les codes à effacements hautes performances

Chapitre 4

Décodage hybride IT/ML des codes LDPC

Contents

Ce chapitre présente un type de décodage performant pour les codes LDPC utilisés sur le canal à effacements. Baptisé décodage hybride IT/ML, il consiste en l'application d'un décodage itératif (IT) dans un premier temps, puis d'un décodage par maximum de vraisemblance (ML) si nécessaire. Cette technique de décodage permet ainsi d'atteindre les capacités de correction d'un décodage ML avec une complexité réduite. En effet l'utilisation préliminaire du décodage IT permet la plupart du temps de compléter le décodage, et dans le cas échéant de simplifier le système que devra résoudre le décodeur ML.
L'implémentation de ce décodeur est critique afin d'obtenir des vitesses de décodage élevées. Nous décrirons donc une façon efficace de l'implémenter et nous présenterons les optimisations possibles de cet algorithme. Cette solution offre une grande flexibilité puisque chaque récepteur peut décider localement du meilleur compromis entre coût en calcul et capacités de corrections.
Les travaux présentés dans ce chapitre ont menés à la publication d'un article [68] et d'un rapport de recherche [67]. Ils ont également largement amélioré les performances de notre codec LDPC [118].

4.1 Introduction

Lorsqu'ils sont décodés par une méthode itérative, les codes LDPC permettent d'obtenir de très bonnes capacités de correction tout en ayant une complexité linéaire. Si les codes LDPC de longueur infinie sont capables d'approcher de près la capacité du canal avec un décodage IT, les codes LDPC de longueurs courtes et moyennes possèdent des performances inférieures.

Il a été montré [86][76] que le décodage ML des codes LDPC permet une amélioration significative des capacités de corrections. Cependant l'utilisation d'un décodeur ML implique une augmentation de la complexité du décodage. En effet, le décodage ML revient à résoudre un système linéaire par une méthode du type élimination de Gauss (GE) ; la complexité du décodage devient donc cubique.

Afin de combiner les avantages des deux types de décodage, nous proposons une approche hybride : le décodage itératif permettra de réduire la complexité globale du décodage tandis que le décodage ML permettra d'atteindre des capacités de corrections supérieures.

L'approche d'un décodage hybride IT/ML a été introduite parallèlement et indépendemment par E. Paolini et al.[84] [82]. Ces travaux mènent aux mêmes conclusions : le décodage hybride IT/ML des codes LDPC permet d'obtenir d'excellentes capacités de correction tout en conservant une complexité réduite.

4.2 Décodeur hybride IT/ML

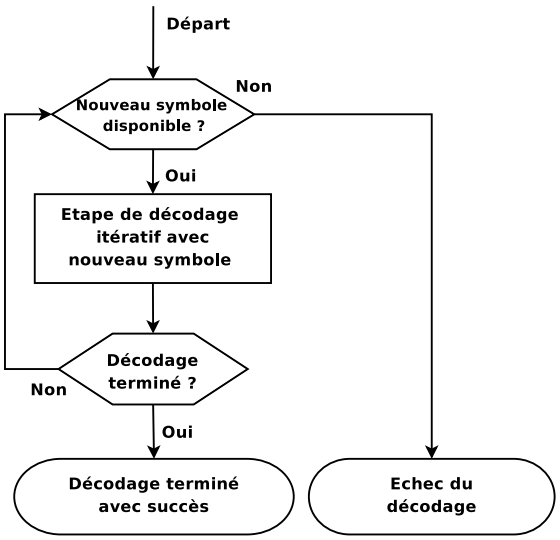
4.2.1 Principes

Le décodage hybride IT/ML repose sur une utilisation conjointe d'un décodeur à complexité linéaire mais à capacités de correction sous-optimales, le décodeur IT, et d'un décodeur à complexité quadratique mais qui permet d'obtenir les capacités de correction maximales offertes par le code, le décodeur ML.

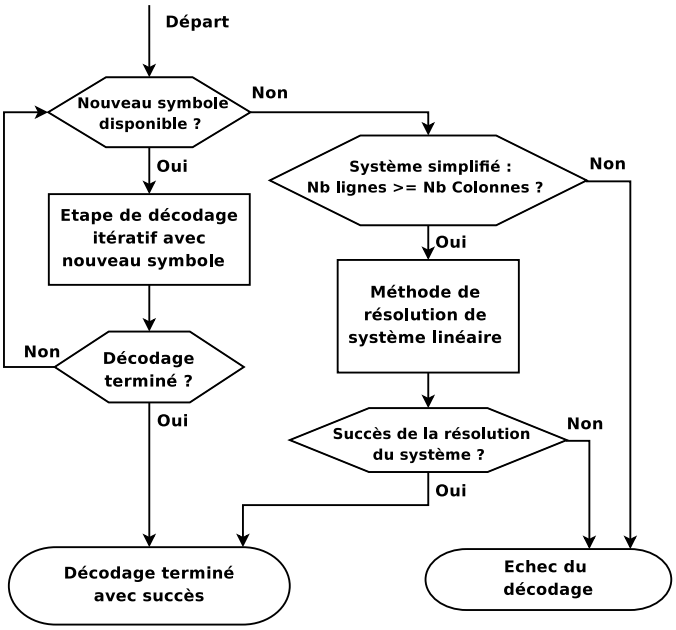
Le principe de fonctionnement est le suivant. A mesure que le récepteur reçoit les symboles, il les soumet au décodeur itératif. Si le décodage peut être achevé par le décodeur IT, alors le mot de code est marqué comme décodé. Si le mot de code ne peut pas être décodé, le récepteur passe alors en mode décodage ML. Le décodage ML va alors tenter de résoudre le système linéaire simplifié par le décodage IT. Ce système pourra être significativement plus petit que le système initial, puisqu'un certain nombre de symboles aura pu être reconstruit pendant la phase de décodage IT. Si l'on détecte que le décodage ML ne pourra pas résoudre le système, par exemple s'il y a plus d'inconnues que d'équations, alors le décodage s'arrête. Sinon un algorithme de résolution de système linéaire (par exemple élimination de Gauss ou décomposition LU) est appliqué sur le système. Si la résolution termine avec succès, le mot de code est marqué comme décodé.

Les diagrammes de la figure 4.1 présentent le décodage itératif seul, et le décodage hybride IT/ML.

Déclenchement du décodage ML : Il est à noter qu'une information clef est le fait que les récepteurs puissent savoir si de nouveaux symboles sont attendus ou non. Dans nombre de champs d'application, ceci est faisable. Dans les quelques cas contraires, une stratégie pourra être mise en place, telle qu'une attente limitée dans la durée avant de



(a) Décodage itératif seul.



(b) Décodage hybride IT/ML.

FIG. 4.1 – Comparaison des deux décodeurs.

tenter un décodage ML. Enfin le récepteur peut décider de déclencher le décodage ML, au lieu d'attendre des symboles supplémentaires, lorsque le nombre de symboles reçus excède un seuil prédéfini pour lequel il sait que le décodage ML aboutira avec une très forte probabilité et avec des temps de calcul compatibles avec les ressources disponibles (énergie, puissance de calcul).

4.2.2 Résolution du système linéaire du décodage ML

Nous allons maintenant détailler comment le décodage peut être implanté efficacement. Même si différentes techniques de résolution de système linéaire existent, celles ci n'ont pas d'importance sur le fonctionnement général du décodeurML. Nous considérerons dans

un premier temps que nous utilisons la méthode d'élimination de Gauss, notée GE.

Soit N_{sc} le nombre de symboles connus, N_{sr} le nombre de symboles reçus et N_{sd} le nombre de symboles reconstruits par le décodage IT. Une condition nécessaire au décodage est que le nombre de symboles reçus N_{sr} soit supérieur à la dimension du code k, $N_{sr} \geq k$. Dans la pratique N_{sr} est de l'ordre de k, puisque le surcoût de décodage est proche de 1. N_{sd} dépend de l'efficacité du décodage itératif, et est petit devant k en général.

Si le décodage itératif est impérativement effectué sur la matrice de parité, le décodageML peut être effectué aussi bien sur la matrice de parité que sur la matrice génératrice.

Décodage ML sur matrice de parité : Soit X le vecteur contenant les n symboles sources et parité du mot de code, et soit H la matrice de parité du code. Soit $\overline{X}$ (resp. $\overline{H}$) les coordonnées de X (resp. les colonnes de H) correspondant aux symboles connus (reçus ou reconstruit[1]). Soit $\widetilde{X}$ (resp. $\widetilde{H}$) les coordonnées de X (resp. les colonnes de H) correspondant aux symboles inconnus. Le système $HX = 0$ peut alors être décomposé de la façon suivante :

$$\widetilde{H}\widetilde{X} = \overline{HX} \tag{4.1}$$

Le décodage ML revient alors à résoudre ce système linéaire dans lequel $\widetilde{X}$ est l'inconnue, $\widetilde{H}$ est la matrice du système et $\overline{HX}$ constitue le second membre.

Le vecteur $\overline{X}$ (resp. $\widetilde{X}$) est donc de taille N_{sc} (resp. $n - N_{\mathrm{sc}}$). La matrice du système initial H possède $n-k$ lignes, chaque symbole reconstruit par le décodage itératif retire une ligne au système. La matrice $\widetilde{H}$ possède donc $n-k-N_{\mathrm{sd}}$ lignes et $n-N_{\mathrm{sc}}$ colonnes. La dimension du système est donc de l'ordre de $(n-k)\times(n-k) = n(1-R)\times n(1-R)$. Notons que la taille du système dépend linéairement de la dimension du code, n, et de $(1-R)$.

Décodage ML sur matrice de génératrice : Soit X (resp. Y) le vecteur contenant les k symboles sources (resp. $n-k$ symboles de parité), et soit G la matrice génératrice du code. Par définition on a : $Y = GX$. Soit $\overline{X}$ (resp. $\overline{Y}$) le vecteur dont les coordonnées sont les symboles sources (resp. de parité) connus. Soit $\overline{G}$ la sous matrice formée des lignes correspondant aux symboles de parité manquants et des colonnes des symboles sources connus. Soit $\widetilde{X}$ (resp. $\widetilde{Y}$) le vecteur dont les coordonnées sont les symboles sources (resp. de parité) manquants. Soit $\widetilde{G}$ la sous matrice formée des lignes correspondantes aux symboles de parité connus et des colonnes des symboles sources manquants. On peut alors décomposer le système à résoudre pour le décodage ML :

$$\widetilde{G}\widetilde{X} = \overline{Y} + \overline{GX} \tag{4.2}$$

où $\widetilde{X}$ est l'inconnue, $\widetilde{G}$ est la matrice du système et $(\overline{Y} + \overline{GX})$ est le membre constant.

Dans le cas où l'ordre de transmission des symboles est aléatoire, ou quand les pertes sont indépendantes, on peut raisonnablement faire l'hypothèse que le rapport entre le nombre de symboles source (resp. de parité) reçus et le nombre de symboles reçus au total est sensiblement égal à $k/n = R$ (resp. $(n-k)/n = 1-R$). Ainsi, $\overline{Y}$ a une taille égale à $N_{sc}(1-R)$ et celle de $\overline{X}$ est égale à $N_{sc}R$. $\widetilde{X}$ a donc une taille de $(k - N_{sc}R)$ et la matrice $\widetilde{G}$ est donc de taille $N_{sc}(1-R)$ lignes et $(k - N_{sc}R)$ colonnes.

[1] Dans le cas d'un décodeur ML utilisé sans décodage IT préliminaire, le vecteur $\overline{X}$ ne sera composé que des symboles reçus.

La taille du système à résoudre est donc de l'ordre de $k(1-R) \times (k-kR) = k(1-R) \times k(1-R)$. Il faut noter que dans ce cas, la taille du système est toujours inférieure à $k \times k$ quel que soit le rendement.

4.3 Implémentation et optimisations

Nous présentons dans cette section, plusieurs optimisations qui ont permis d'améliorer significativement les vitesses de décodage.

Structures de données et représentation des matrices : Si les matrices de parité des codes LDPC sont creuses, il n'en est pas de même des matrices génératrices et des matrices manipulées par le décodage ML. En effet, les matrices génératrices n'ont a priori aucune raison d'être creuses et possèdent la plupart du temps une densité sensiblement égale à $1/2$. Lorsque, lors du décodage ML, on applique un algorithme de résolution sur la matrice de parité simplifiée, on effectue des opérations qui ont tendance à remplir la matrice du système. Ainsi au fur et à mesure des itérations, la matrice du système va perdre son caractère creux.

Pour le décodage itératif, on utilise une notation creuse reposant sur des listes doublement chaînées d'éléments : chaque élément d'une liste représente un 1 dans la matrice, et il possède des pointeurs vers les éléments suivants de sa ligne et de sa colonne. Cette notation permet un parcours efficace du graphe du Tanner du code, tout en réduisant la taille mémoire de la structure.

Cette notation est inadaptée à la représentation de matrices denses. Ainsi pour ces matrices on utilise une notation dense dans laquelle la matrice est représentée sous la forme d'un tableau de mots binaires dont chaque bit représente une entrée de la matrice. Cette notation doit être automatiquement adoptée dès lors que le décodage ML est considéré, que ce soit sur la matrice de parité ou sur la matrice génératrice.

Il est à noter que ce choix de structure de données peut impacter d'un ordre de grandeur les vitesses de décodage. Il n'est donc pas à négliger.

Décodage ML sur la matrice de parité ou sur la matrice génératrice ? Nous avons vu dans la section précédente que le décodage ML pouvait s'effectuer sur la matrice de parité ou sur la matrice génératrice. Le succès du décodage ne dépend pas de la matrice sur laquelle on effectue le décodage, cependant ce choix a un impact sur la complexité de décodage. En effet, on a vu que les tailles des matrices des systèmes correspondants sont différentes : $n(1-R) \times n(1-R)$ pour le système issu de la matrice de parité et $k(1-R) \times k(1-R)$ pour celui issu de la matrice génératrice.

Si le choix de la méthode ne repose que sur la taille du système, on aura intérêt à choisir la matrice génératrice puisque, quel que soit le rendement $R \in]0,1[$, la taille du système issu de la matrice de parité sera toujours supérieure à celle du système issu de la matrice génératrice.

Cependant la taille n'est pas le seul critère à prendre en compte. En effet, la densité du système influe sur la complexité de résolution : plus la matrice est dense, plus la résolution est coûteuse. Le système issu de la matrice de parité hérite du caractère creux de la matrice originale, alors que celui issu de la matrice génératrice sera dense comme celle-ci.

Les expériences que nous avons effectuées ont montré que le décodage ML sur la matrice de parité est plus rapide pour les rendements supérieurs à $1/2$, alors que le décodage sur la matrice génératrice prend l'avantage lorsque le rendement est faible. En particulier, lorsque le rendement devient très faible, i.e. $n >> k$, le décodage sur matrice génératrice prend tout son sens puisque le système est alors de taille $k \times k$ contre $n-k \times n-k \approx n \times n$.

Préparation du système linéaire : Lorsque le décodage ML est enclenché, il doit récupérer l'état du décodeur IT pour bénéficier du travail déjà effectué par celui-ci. Cette étape consiste en la construction du système qui se compose de la matrice du système et du second membre.

Dans le cas du décodage sur la matrice de parité, la matrice du système est la matrice $\widetilde{H}$ qui est simplement une sous matrice de H, le second membre quant à lui étant égal au vecteur des sommes partielles (la somme des symboles connus dans chaque équation) utilisé par le décodage itératif.

Dans le cas du décodage sur la matrice génératrice, la matrice du système est une sous matrice $\widetilde{G}$ de la matrice génératrice qui pourra éventuellement être précalculée. Le second membre peut s'obtenir directement en effectuant le produit $\overline{GX}$ et en le sommant aux symboles de parités connus $\overline{Y}$. La matrice $\overline{G}$ étant dense, ce produit peut être coûteux (i.e de complexité quadratique).

Lorsque l'on considère le décodage des codes LDPC-staircase, ou n'importe quel autre code de la famille des codes "Repeat-Accumulate", une méthode alternative est possible. En effet, en exploitant la structure triangulaire inférieure de la partie droite de la matrice de parité, on peut effectuer cette opération en temps linéaire. La matrice génératrice se calcule comme $G = H_2^{-1} H_1$. Le vecteur des sommes partielles issues du décodage itératif est égal à :

$$P = \overline{H}(\overline{X}^T | \overline{Y}^T)^T = \overline{H_1 X} + \overline{H_2 Y}$$

en effectuant la multiplication par la matrice H_2^{-1} on obtient :

$$H_2^{-1} P = \overline{GX} + \overline{Y}$$

Le second membre du système associé à la matrice génératrice se déduit ainsi :

$$\overline{GX} = \overline{Y} + H_2^{-1} P$$

La matrice H_2 étant une matrice escalier, la matrice H_2^{-1} est en fait une matrice triangulaire inférieure pleine. La multiplication d'un vecteur $U = \{u_0, \ldots, u_n\}$ par une telle matrice peut s'effectuer de manière linéaire, grâce à une suite récurrente. Soit $V = \{v_0, \ldots, v_n\}$ le résultat de $V = H_2^{-1} U$. Alors $v_0 = u_0$ et $\forall j > 0, u_j = u_j + \sum_{i=0}^{j-1} u_j$. La somme $s_j = \sum_{i=0}^{j-1} u_j$ peut se calculer comme $s_j = s_{j-1} + u_{j-1}$. De cette manière, le calcul de V ne nécessite que deux sommes par coordonnée. Le produit $H_2^{-1} P$ peut donc être effectué en temps linéaire quand H_2 est une matrice escalier (plus généralement une matrice triangulaire inférieure).

Algorithme de résolution de système linéaire Il existe un grand nombre d'algorithmes de résolution de systèmes linéaires. Nous avons choisi d'utiliser une approche basée sur décomposition LU pour résoudre les systèmes linéaires lors du décodage ML. De plus, nous avons choisi de ne pas stocker la matrice U, et d'appliquer directement les opérations

correspondantes sur le second membre. Ceci permet une économie de mémoire puisque seule une matrice dense, la matrice L, doit être stockée lors du décodage. La résolution se divise alors en deux phases : dans un premier temps une étape d'élimination en avant permet de trianguliariser le système, puis une étape de substitution à rebours permet de déduire le vecteur inconnu. Lors de la phase d'élimination, des opérations sont effectuées sur les lignes et sont appliquées en parallèle sur le second membre.

Lorsque le décodage ML est effectué sur la matrice de parité, le système initial est creux. La complexité de la résolution de systèmes linéaires creux peut être réduite si on prend soin de retarder le remplissage de la matrice [36]. Un moyen efficace pour cela est d'utiliser des techniques de réordonnancement de pivot. Ces techniques consistent à choisir le pivot de manière à réduire le remplissage de la matrice. Une méthode simple est de choisir le pivot qui appartient à la ligne (et/ou la colonne) la plus creuse. Si de telles méthodes permettent de réduire efficacement le remplissage, elles nécessitent le maintient d'une structure de donnée supplémentaire conservant en mémoire le poids des lignes et des colonnes de la matrice. La mise à jour de cette structure de données peut se faire de plusieurs manières :

- à la volée après chaque modification d'une ligne, ceci nécessite de parcourir l'ensemble de la ligne ;
- de manière non systématique, en ne mettant à jour le poids de la ligne qu'après un certain nombre de modifications ;
- de manière heuristique, en estimant le poids d'une ligne à partir du poids des lignes à partir desquelles elle a été obtenue ;
- suivant une approche hybride combinant les approches précédentes, par exemple en utilisant l'heuristique la majorité du temps, et le calcul effectif du poids après un certain nombre de modifications.

Dans le cas des matrices LDPC ayant des colonnes de poids différents (ce qui est le cas avec les LDPC-staircase puisque la partie gauche possède des colonnes de poids $N1$ alors que la partie droite possède des colonnes de poids 2), on peut utiliser un réordonnancement des colonnes. En mettant les colonnes les plus creuses au début de la matrice, on limitera le remplissage de la matrice lors des premières itérations de l'étape d'élimination en avant. Cette technique peut être vue comme une méthode de réordonnancement de pivot suivant les colonnes, mais qui a l'avantage de ne pas nécessiter de calculs supplémentaires.

Les techniques de réordonnancement ne doivent pas être systématiquement utilisées. En effet lorsque les opérations sur les lignes sont peu coûteuses (matrice et/ou symbole de tailles de taille réduite), le surcoût des opérations nécessaires au réordonancement n'est pas compensé par les gains apportés par celui-ci au niveau du nombre d'opérations. Nos observations sur des codes de dimension 1000 et avec des symboles de 1024 octets ont montré une amélioration de quelques pourcents de la vitesse de décodage.

4.3.1 Simplification du système permis par le décodage IT

L'utilisation préliminaire du décodage itératif permet de réduire la complexité du décodage ML qui sera éventuellement effectué. En effet, même s'il est incomplet, le décodage itératif permet de reconstruire un certain nombre de symboles, réduisant ainsi la taille du système à résoudre par le décodage ML.

Cette simplification est effectué de deux manières. Pour chaque symbole reconstruit par le décodage itératif :

– une variable et donc une colonne est retirée du système ;
– la contrainte à partir de laquelle le symbole a été reconstruit devient inutile (tous les symboles impliqués sont connus) ; on peut donc la retirer du système en enlevant une ligne au système linéaire.

Pour estimer l'importance de la simplification du système, on utilise le *ratio de simplification* qui est égal au rapport entre la taille du système simplifié et la taille du système initial ($n(1-R) \times n(1-R)$ pour le décodage ML sur la matrice de parité). La figure 4.2 présente le ratio de simplification du système représenté par la matrice de parité en fonction du taux de pertes. Pour chaque taux de pertes on a mesuré le ratio de simplification moyen obtenu sur 500 itérations que nous avons tracé avec les valeurs minimum et maximum. Pour les taux de pertes faible, le décodage itératif est toujours capable de décoder et dans ce cas la taille du système simplifié est nulle. A mesure que le taux de pertes augmente le décodage itératif est de moins en moins capable de compléter le décodage et simplifie de moins en moins le système. Pour un taux de perte de 25.5% le système linéaire voit sa taille réduite de 30% en moyenne.

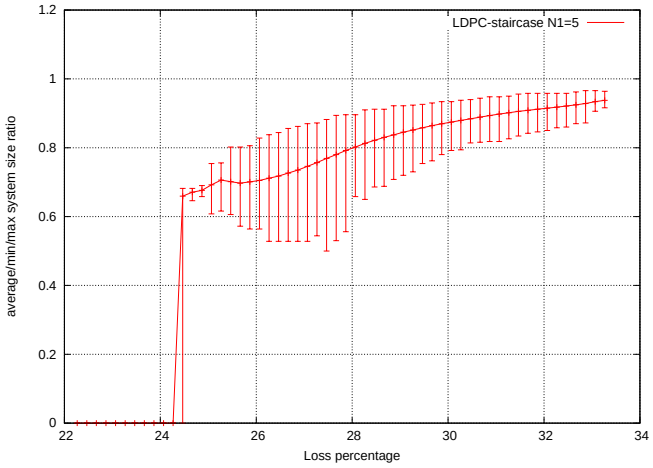


FIG. 4.2 – Ratio de simplification du système de la matrice de parité en fonction du taux de perte (LDPC-staircase, $N1 = 5$, $k = 1000$, $R = 2/3$).

Ainsi le décodage itératif peut faire diminuer sensiblement la taille du système que devra résoudre le décodage ML et ainsi réduire la complexité de celui ci.

4.4 Performances

Nous évaluons les performances des codes LDPC-staircase décodés avec un décodeur hybride IT/ML et nous les comparons à deux codes à effacements de référence : les codes Reed-Solomon et les codes Raptor. Nous montrons que la solution basée sur le décodage hybride IT/ML des codes LDPC-staircase surpasse largement les codes Reed-Solomon et atteint quasiment les performances des codes Raptor pour des tailles d'objets de plusieurs milliers de symboles.

4.4.1 Conditions expérimentales

Les performances du décodage hybride IT/ML vont être évaluées sur un code LDPC-staircase à l'aide de simulations de type Monte-Carlo (voir section 3.4.2). Suivant les recommandations du chapitre suivant (chapitre 5), nous fixons le paramètre déterminant le degré des noeuds symboles à $N1 = 5$. Nous utilisons le codec C++ LDPC, version 2.1 [118] auquel nous avons ajouté le support du décodage hybride IT/ML.

Nous effectuerons des comparaisons avec les code Reed-Solomon en utilisant le codec Reed-Solomon open-source proposé par Luigi Rizzo [97] (voir section 2.4.1). Ce codec travaille sur le corps fini GF(2^8) ce qui limite la longueur du code à $n = 255$. Lorsqu'une longueur supérieure est nécessaire, nous utilisons plusieurs codes RS de longueur ≤ 255, et nous répartissons les symboles suivant l'algorithme standardisé de FLUTE/ALC [81].

Les codes Raptor à rendement fixé seront également comparés. Nous avons développé à cet effet un codec, lors d'un partenariat INRIA/Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), qui est conforme aux spécifications du RFC 5053 [60] reposant sur le décodage ML[2].

4.4.2 Capacités de correction

Nous allons évaluer les capacités de corrections des codes considérés. Il faut noter que les performances en termes de correction obtenues avec le décodeur hybride IT/ML ne sont pas impactées par le décodage IT et ne dépendent que du décodage ML.

Ratio d'inefficacité

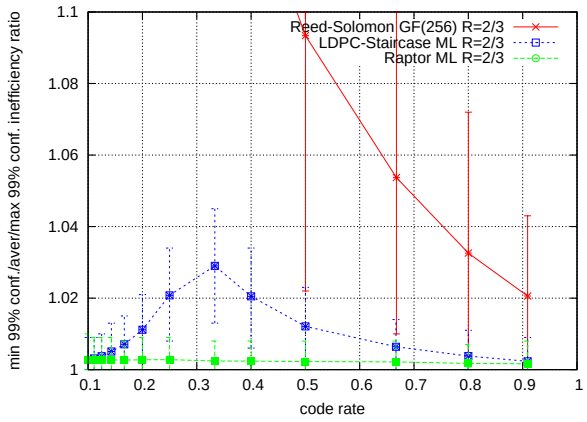
Nous commençons par étudier le ratio d'inefficacité, qui est représenté en fonction du rendement sur la figure 4.3(a) et en fonction de la taille de l'objet en termes de symboles sur la figure 4.3(b). La figure 4.3(a) montre que les codes Reed-Solomon sont inadaptés pour les rendements faibles. Les codes Raptor possèdent des performances stables autour de $0,21\%$ quel que soit le rendement (ceci n'a rien de surprenant étant donné que ces codes sont "sans rendement"). Les codes LDPC-staircase conservent un ratio d'inefficacité faible hormis entre $0,2$ et $0,5$, où le ratio d'inefficacité est moins bon (il atteint presque 3%), il reste sinon nettement sous la barre des 1%[3].

Les faibles performances des codes Reed-Solomon sont également mis en avant sur la figure 4.3(b), où l'on peut observer une augmentation rapide du ratio d'inefficacité à mesure que la taille de l'objet augmente. Ce phénomène est lié à la nécessité de découper l'objet en petits blocs encodables par le code Reed-Solomon travaillant sur GF(2^8)[4]. Les codes LDPC-staircase et les codes Raptor, qui sont quant à eux des codes "grand blocs", permettent d'obtenir d'excellentes performances. Si les codes Raptor sont meilleurs, la différence avec les codes LDPC-staircase est très faible, puisque ceux-ci convergent rapidement sous la barre des $0,6\%$ d'overhead, i.e. extrêmement proche de l'optimal.

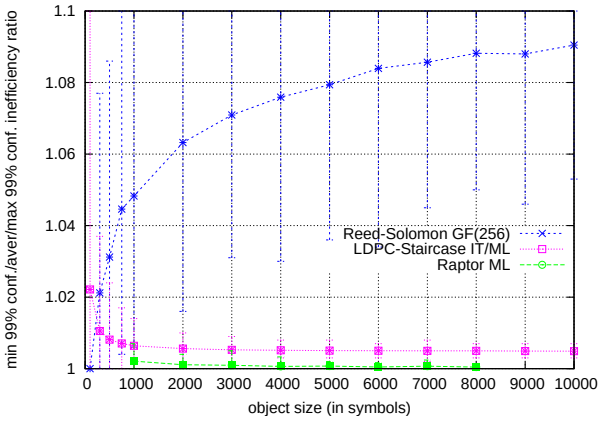
[2] Copyright © CNES (« logiciel de codage/décodage Raptor ») : Tous droits réservés.

[3] Il est à noter qu'utiliser un rendement plus faible que nécessaire pour ensuite n'utiliser qu'un sous-ensemble des symboles de redondance(technique de poinçonnage ou "puncturing") est une solution qui permet de parer à l'éventuelle dégradation des performances dans l'intervalle $[0,2;0,5]$

[4] L'utilisation du corps fini GF(2^{16}) n'est pas réaliste d'un point du coût des opérations.



(a) Objet de 1000 symboles



(b) Rendement R=2/3

FIG. 4.3 – Ratio d'inefficacité en fonction du rendement et de la taille de l'objet.

Probabilité d'erreur de décodage

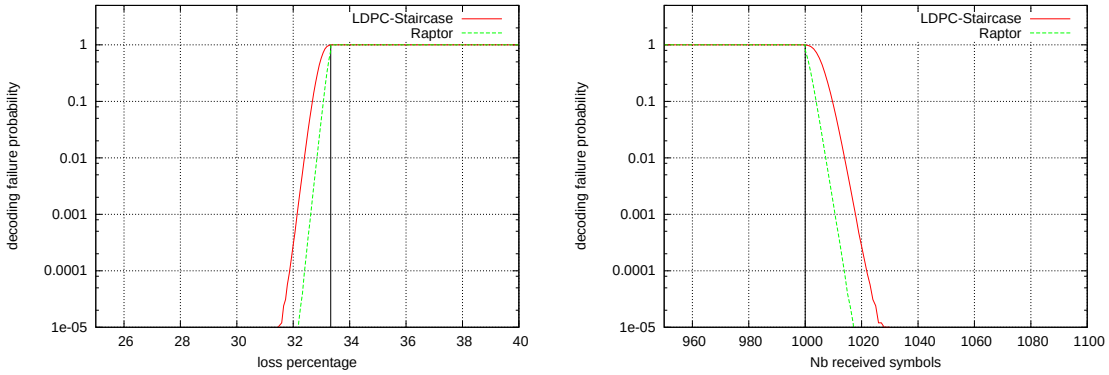
Nous allons maintenant comparer les capacités de corrections grâce à la métrique de probabilité d'erreur de décodage. Nous avons exprimé cette probabilité en fonction du taux de perte sur la figure 4.4(a) et en fonction du nombre de symboles reçus sur la figure 4.4(b), pour les codes LDPC-staircase et les codes Raptor[5].

La figure 4.4(a) montre que les LDPC-staircase sont proche de l'idéal avec une pente raide dans la région de "waterfall". De plus, aucun plancher d'erreur n'apparaît au dessus de 10^{-5}, ce qui est une très bonne performance.

Sur la figure 4.4(b) on observe que la réception de quelques symboles supplémentaires, en plus des k symboles minimaux, permet d'atteindre rapidement des probabilités d'échec en deçà de 10^{-5}.

La table 4.1 présente un récapitulatif des résultats. Afin d'obtenir une probabilité d'échec de décodage inférieure à 10^{-4}, il suffit de recevoir, en plus des $k = 1000$, 22 symboles avec les codes LDPC-staircase, contre 14 symboles avec les codes Raptor, soit une différence de 8 symboles. Pour le rendement considéré, les performances de ces codes sont proches de l'idéal, et les différences entre ces codes sont très limitées.

[5]Les performances des codes Reed-Solomon étant si médiocres que nous ne les avons pas présentées sur ces courbes afin d'améliorer la lisibilité.



(a) En fonction du taux de pertes (b) En fonction du nombre de symboles reçus

FIG. 4.4 – Probabilité d'échec de décodage ($k = 1.000$ symboles, code rate $2/3$).

	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 $2,2\%$	$31,86\%$	$0,90 \times 10^{-4}$

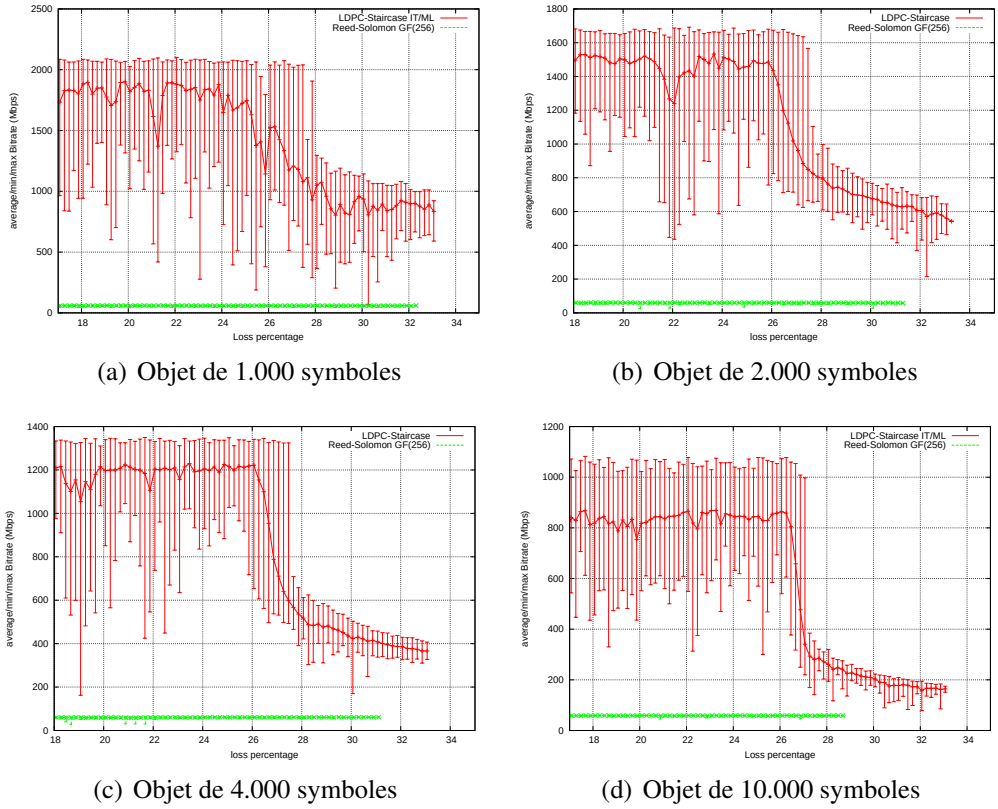
TAB. 4.1 – Overhead et taux de perte nécessaires à l'obtention d'une probabilité d'échec inférieure à 10^{-4} ($k = 1.000$ symboles, code rate $2/3$).

4.4.3 Vitesse et débit de décodage

Nous allons nous intéresser à la vitesse de décodage des codes LDPC par un décodeur hybride IT/ML. La complexité du décodeur itératif étant linéaire et celle du décodeur ML étant cubique, la vitesse de décodage va dépendre de l'utilisation du décodeur ML. Le décodeur ML n'est utilisé que quand le décodeur itératif échoue, c'est à dire quand le taux de perte est trop élevé pour être corrigé par le décodeur IT.

Nous avons effectué les tests sur un système GNU/Linux, utilisant un noyau 2.6.27-11/64 bits et un processeur Intel Dual Core Xeon 5120/1.86 GHz(1066 MHz). Puisque le rendement est supérieur à $1/2$, le décodeur ML utilisé est celui considérant la matrice de parité. Nous n'avons pas effectué de comparaison avec les codes Raptor, étant donné que notre codec n'est pas optimisé en termes de vitesse de décodage. Cependant, il est raisonnable de penser que les performances des codes Raptor seront proches de celles des codes LDPC-staircase puisque les codes Raptor utilisent des matrices binaires creuses similaires à celles des codes LDPC.

On considère le cas d'un objet de 1MO, divisé en 1.000 symboles de 1024 octets. Nous avons donc tracé sur la figure 4.5(a) le débit du décodeur en fonction du taux de perte du canal. On voit que pour des taux de perte faibles, le décodage IT est suffisant et il garantit une vitesse de décodage élevée de l'ordre de 1,8Gb/s (30 fois plus rapide que les codes Reed-Solomon) grâce à sa complexité linéaire. Puis, au fur et à mesure que le taux de perte s'approche de la limite théorique, le débit diminue jusqu'à se stabiliser autour de 850 Mb/s. Ceci s'explique par l'usage de plus en plus fréquent du décodage ML. Ce débit est toujours 14 fois plus rapide que celui des Reed-Solomon. Ce débit diminue à mesure que l'on augmente la taille de l'objet, mais reste toujours supérieur à ce qu'on obtient avec les codes Reed-Solomon.



(a) Objet de 1.000 symboles (b) Objet de 2.000 symboles

(c) Objet de 4.000 symboles (d) Objet de 10.000 symboles

FIG. 4.5 – Vitesse de décodage en fonction de la taille de l'objet (code rate $2/3$).

4.5 Travaux relatifs

Comme précisé dans l'introduction, l'utilisation d'un décodeur hybride IT/ML pour les codes LDPC sur le canal à effacements a été parallèlement et indépendamment proposée par Paolini et al. [84] [82]. Les résultats obtenus montrent que les codes GeIRA ("Generalized Repeat-accumulate", une classe de code englobant les LDPC-staircase) atteignent d'excellentes capacités de correction qui sont sensiblement égales à celles des codes Raptor, et cela même pour des petites dimensions.

Il est possible d'exploiter la faible densité de la matrice de parité afin d'effectuer une implémentation efficace du décodage ML des codes LDPC. Cette approche a été présentée par Burshtein et Miller dans [20] et par Shokrollahi et al. dans un brevet [104]. Grâce à une permutation des lignes et des colonnes, le système est transformé en un système "presque triangulaire" (le système se décompose alors en une matrice triangulaire inférieure creuse, et une matrice dense). Le système décrit par la matrice dense est résolu par une élimination de Gauss, ce qui permet de débloquer le décodage et de déduire le reste des variables grâce à une étape de substitution à rebours sur la sous matrice triangulaire. Ainsi l'opération coûteuse, qu'est l'élimination de Gauss, est effectuée sur un système de taille réduite. La complexité de cet algorithme dépend fortement de la taille de ce système, et donc de la possibilité de trianguläriser par permutation une grande partie de la matrice.

4.6 Conclusions

Nous avons présenté un nouveau type de décodeur pour les codes LDPC sur le canal à effacements. Ce décodeur appelé décodeur hybride IT/ML est capable d'obtenir les capacités de correction du décodage ML avec une complexité réduite grâce à un emploi préliminaire du décodage IT. Le décodage IT permet de décoder la plupart du temps, et le cas échéant réduit la taille du système que devra résoudre le décodage ML. Nous avons également présenté des techniques d'optimisation permettant d'obtenir des débit de décodage élevés.

Grâce au décodage hybride IT/ML les codes LDPC-staircase se rapprochent des codes idéaux et sont très proches des codes Raptor. De plus, les débits de décodage obtenus sont élevés (au moins un ordre de grandeur supérieur aux codes Reed-Solomon) et cela même quand le décodage ML est nécessaire. Le décodage hybride IT/ML permet de construire un système de codage de longueur courte pour la correction d'effacements s'approchant des codes idéaux et supportant des débits élevés. Nous verrons dans les prochains chapitres comment on peut construire des codes dont la structure permet un décodage ML plus rapide.

De futurs travaux s'attelleront à améliorer les techniques de décodage ML. Par exemple une parallélisation des opérations permettra d'exploiter pleinement les capacités des machines multicoeur.

La construction d'algorithmes de décodage aux capacités intermédiaires est une piste à considérer. Une possibilité est d'utiliser un algorithme de décodage permettant d'aller au delà du décodage IT en effectuant des combinaisons des lignes de la matrice de parité. L'objectif est de trouver une combinaison dont le résultat est une équation de degré 1, et qui permet donc de reconstruire un nouveau symbole. La construction de cette combinaison de lignes pourra s'appuyer sur le parcours de cycles dans la matrice de décodage. En effet, un cycle dont toutes les lignes impliquées sont de degré 2, sauf une qui sera de degré 3, permet de construire une telle combinaison.

Chapitre 5

Adaptation des code LDPC-staircase au décodage hybride IT/ML

Contents

Ce travail se concentre sur les codes LDPC-staircase pour le canal à effacements. Nous avons souligné dans le chapitre précédent que le décodage par maximum de vraisemblance (ML) permet d'améliorer sensiblement les capacités de correction de ces codes. L'utilisation d'un décodeur hybride IT/ML permet alors de réduire sensiblement la complexité d'une telle solution.
Dans ce chapitre nous nous intéresserons à l'influence du paramètre N1, qui détermine le degré des nœuds symboles sources, sur les performances des codes LDPC-staircase. En effet nous montrons qu'en augmentant la valeur de N1, les capacités de correction des codes LDPC-staircase décodés en ML s'approchent de l'optimal.
Ce travail a fait l'objet d'une publication [27].

5.1 Introduction

Les codes LDPC sont habituellement décodés à l'aide d'un décodeur itératif (IT) qui est sous optimal en termes de capacité de correction mais qui a l'avantage de posséder une complexité linéaire. Le décodage par maximum de vraisemblance (ML) permet quant à lui d'atteindre la capacité de correction optimale permis par le code au prix d'une complexité accrue. L'utilisation conjointe de ces deux techniques sous la forme d'un décodeur hybride IT/ML permet alors de conjuguer les avantages des deux solutions (voir chapitre 4).

Les codes LDPC-staircase sont une famille de codes appartenant à la classe des "Repeat-Accumulate", dont la construction, très simple, est décrite dans le RFC5170 [99] (voir section 2.5.4). La matrice de parité H de ces codes peut être divisée en deux parties : la

partie gauche H_1 dont les colonnes correspondent aux symboles sources, et la partie droite H_2 dont les colonnes correspondent aux symboles de parité. La partie droite H_2 possède une structure escalier inférieure tandis que la matrice H_1 est une matrice régulière remplie à l'aide d'un PRNG. Mis à part la graine servant à initialiser le PRNG, la matrice de parité est uniquement caractérisée par le degré des colonnes de H_1 est appelé $N1$. Afin d'optimiser les performances sous le décodage itératif et de limiter la complexité du décodage, il a été choisi dans [98] de fixer la valeur de ce paramètre à $N1 = 3$.

Nous allons montrer que lorsque l'on considère un décodage ML, cette valeur par défaut du paramètre $N1$ n'est plus optimale. Nous avons donc analysé l'influence de ce paramètre sur différentes métriques, telles que l'inefficacité de décodage, la probabilité d'échec de décodage ainsi que la vitesse de décodage pour les décodeurs IT et ML. Nous verrons qu'il est possible d'adapter ce paramètre en fonction du type de décodeur supporté par le récepteur ainsi que de la capacité de correction désirée.

5.2 Influence du paramètre N1 sur les capacités de correction

Dans cette section nous étudions les capacités de correction des codes LDPC-staircase en fonction du paramètre $N1$. Ces résultats sont issus de simulations de type Monte-Carlo (voir section 3.4.2) effectuées avec un codec supportant les deux types de décodage IT et ML.

5.2.1 Capacités de correction du décodage ML

Commençons par étudier l'évolution de l'inefficacité de décodage en fonction de la valeur de $N1$. La figure 5.1 présente le ratio d'inefficacité du décodage ML en fonction de $N1$, pour deux rendements $R = \{2/3; 2/5\}$. Lorsque $N1$ augmente, le ratio d'inefficacité décroît rapidement pour se retrouver très proche de 1.

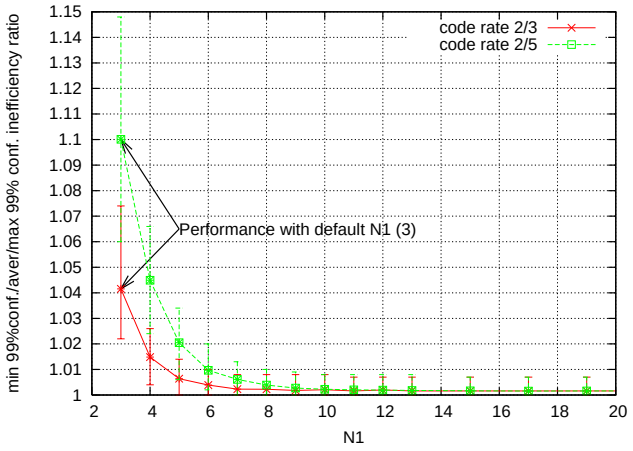


FIG. 5.1 – Ratio d'inefficacité en fonction du paramètre $N1$ pour le décodage ML (code de dimension $k = 1000$).

La figure 5.2 montre l'évolution du ratio d'inefficacité en fonction du rendement du code, pour différentes valeurs de $N1$. L'augmentation de $N1$ permet une réduction du ratio

d'inefficacité, pour la plupart des rendements considérés. De plus, pour les rendements faibles, les différentes courbes sont confondues. Cela vient du fait que pour ces rendements, la valeur spécifiée de $N1$ est trop faible pour remplir suffisamment H_1, et des entrées sont donc ajoutées par l'algorithme de création de la matrice[1].

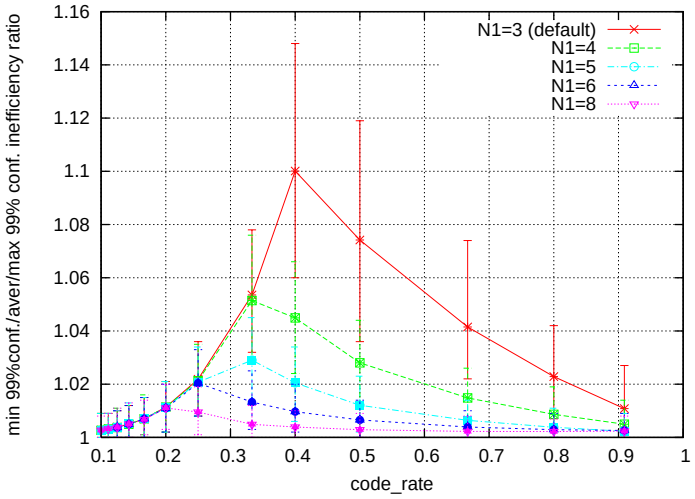


FIG. 5.2 – Ratio d'inefficacité en fonction du rendement pour le décodage ML (code de dimension $k = 1000$).

Afin d'évaluer les performances de correction, on peut également s'intéresser à la probabilité d'échec du décodage (voir section 3.2.2). La figure 5.3 montre la probabilité d'échec de décodage en fonction du ratio d'effacements pour deux rendements $R = \{2/3; 2/5\}$ et plusieurs valeurs de $N1$. Deux phénomènes sont observables. Premièrement l'augmentation de $N1$ fait que le plancher d'erreur descend et finit par devenir trop bas pour être visible. En effet pour $N1 > 4$ le plancher d'erreur n'est plus visible et est donc inférieur à 10^{-4}. Deuxièmement lorsque l'on s'intéresse à la région de "waterfall", la courbe a tendance à se rapprocher de celle du code idéal (droite verticale) à mesure que l'on augmente $N1$.

Toutes ces observations permettent de conclure que l'augmentation de la valeur de $N1$ améliore les capacités de correction des codes LDPC-staircase quand ils sont décodés avec un décodeur ML.

5.2.2 Capacités de correction du décodage itératif

Les observations précédentes nous incitent à utiliser une grande valeur de $N1$. Cependant l'efficacité du décodage itératif repose sur le caractère creux de la matrice, et augmenter $N1$ va provoquer une "densification" de la matrice menant à une dégradation de la capacité de correction de ce décodage. Nous avons donc effectué des simulations afin d'observer l'impact de ce paramètre sur les capacités de correction des codes LDPC-staircase lorsqu'ils sont décodés avec un décodeur itératif.

La figure 5.4 présente le ratio d'inefficacité en fonction du paramètre $N1$. Comme prévu, le ratio d'inefficacité augmente avec $N1$.

[1] Le nombre d'entrées non-nuls dans chaque ligne de H_1 doit être supérieur ou égal à 2. Dans le cas contraire des entrées sont ajoutées aléatoirement dans les lignes, augmentant ainsi la densité de H_1.

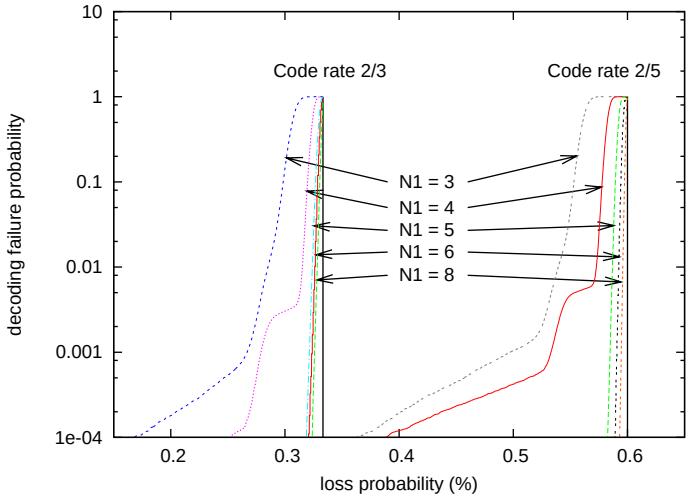


FIG. 5.3 – Probabilité d'échec de décodage en fonction du ratio d'effacements pour le décodage ML (code de dimension $k = 1000$).

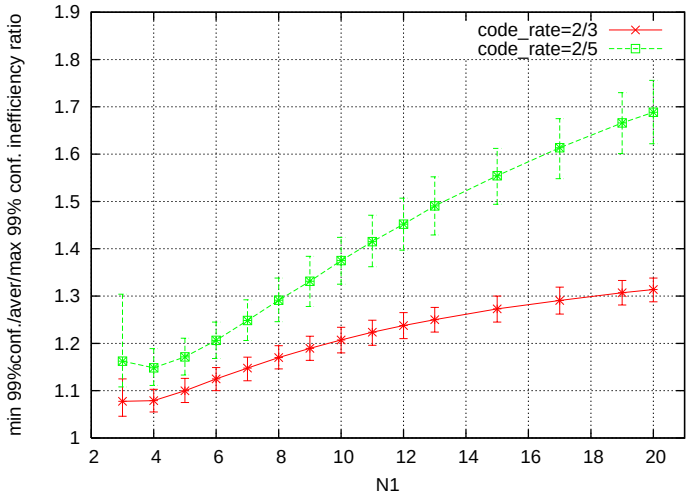


FIG. 5.4 – Ratio d'inefficacité en fonction du paramètre $N1$ pour le décodage IT (code de dimension $k = 1000$).

Sur la figure 5.5 nous avons tracé le ratio d'inefficacité en fonction du rendement pour différentes valeurs de $N1$. L'augmentation de $N1$ induit une augmentation du ratio d'inefficacité pour l'ensemble des rendements considérés. On observe le même phénomène que pour le décodage ML lorsque l'on regarde le ratio d'inefficacité pour les rendements faible : la valeur spécifiée de $N1$ n'a pas d'impact sur la densité de la matrice générée par l'algorithme de création de matrice de parité LDPC-staircase.

La probabilité d'échec du décodage en fonction du ratio d'effacements est présentée sur la figure 5.6 pour plusieurs valeurs de $N1$. L'augmentation de $N1$ a un impact négatif dans la région du "waterfall". En effet la courbe s'éloigne de celle du code parfait à mesure que l'on augmente $N1$. Cependant l'augmentation de $N1$ a un effet positif qui est d'abaisser le plancher d'erreur, jusqu'à le faire sortir de l'intervalle considéré. Pour des valeurs de $N1 > 4$ le plancher d'erreur est inférieur à 10^{-4}.

L'augmentation de la valeur de $N1$ a donc un aspect globalement négatif sur les capaci-

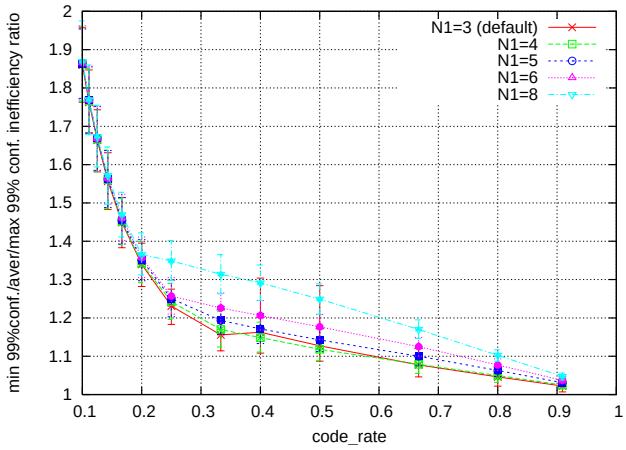


FIG. 5.5 – Ratio d'inefficacité en fonction du rendement pour le décodage IT (code de dimension $k = 1000$).

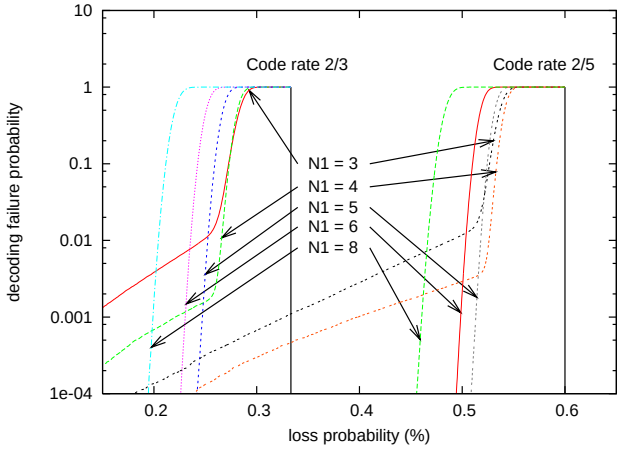


FIG. 5.6 – Probabilité d'échec de décodage en fonction du ratio d'effacements pour le décodage IT (code de dimension $k = 1000$).

tés de correction des codes LDPC-staircase décodés itérativement. Cependant l'augmentation de $N1$ a le bénéfice de faire chuter le plancher d'erreur. Ceci peut avoir son importance dans les systèmes où la probabilité d'erreur est un aspect critique et où l'efficacité globale du système (le ratio d'inefficacité) est considérée comme secondaire.

A la vu de ces résultats, nous considérerons que la valeur $N1 = 5$ constitue un bon compromis entre les dégradations des performances IT et l'amélioration des performances ML. La table 5.1 montre les performances des codes LDPC-Staircase pour deux rendements usuels.

5.3 Influence du paramètre N1 sur la vitesse de décodage

Nous nous intéressons maintenant à l'évolution de la complexité du décodage en fonction de la valeur de $N1$. Nous considérons que le récepteur utilise un décodeur hybride IT/ML sur matrice de parité tel que décrit dans le chapitre 4. Ces tests ont été effectués sur un PC 4xIntelXeon5120 @ 1.86GHz/4 GB RAM GNU/Linux. Pour chaque point, nous

rendement	overhead moyen	overhead pour une proba. d'échec $\leq 10^{-4}$
2/3	0.63%	2.21%
2/5	2.04%	4.41%

TAB. 5.1 – Overhead moyen et overhead pour une probabilité d'erreur inférieure à 10^{-4} en fonction du rendement ($k = 1000$, N1=5, décodage ML).

avons effectué 200 tests aboutissant à un décodage réussi, et nous avons tracé la moyenne ainsi que le minimum et le maximum du débit.

Nous comparons ces résultats à ceux de l'excellent codec Reed-Solomon open-source proposé par Luigi Rizzo [97] (voir section 2.4.1). Ce codec travaille sur le corps fini GF(2^8) ce qui limite la longueur du code à $n = 255$. Lorsqu'une longueur supérieure est nécessaire on utilise plusieurs codes RS de longueur ≤ 255 et on partitionne l'objet source en blocs suivant l'algorithme standardisé de FLUTE/ALC [81].

Les résultats en termes de débit de décodage sont présentés sur la figure 5.7. Nous observons sans surprise que la valeur de $N1$ a un impact significatif sur la complexité du décodage. Ceci est causé par plusieurs phénomènes :

- la complexité du décodage itératif dépend du nombre de symboles présents dans chaque équation, qui détermine le nombre d'opérations XOR et d'opérations élémentaires sur la matrice de parité effectués au cours du décodage. Ceci est la raison pour laquelle les courbes ne sont pas superposées pour les taux de perte faibles (là où le décodage itératif est suffisant).
- la complexité du décodage ML dépend de la de la taille du système ainsi que de la densité du système initial. Plus le taux de perte est élevé plus le décodage itératif sera inefficace et plus le système sur lequel devra travailler le décodage ML sera grand. De plus, un décodage IT inefficace rend le décodage ML nécessaire pour des taux de perte de plus en plus faible, faisant ainsi chuter plus tôt le débit. La densité de la matrice sur laquelle le décodage ML va être effectué sera d'autant plus importante que $N1$ sera grand, augmentant ainsi la complexité.

Même si la complexité de décodage augmente avec $N1$, cette solution est en moyenne au moins un ordre de grandeur plus rapide que celle basée sur le codec Reed-Solomon :

- Reed-Solomon : 54 Mb/s en moyenne (la référence)[2] ;
- LDPC-staircase, $N1 = 5$, taux de perte faible, où le décodage IT est suffisant : 1.75 Gb/s (32.4 fois plus rapide) en moyenne ;
- LDPC-staircase, $N1 = 5$, taux de perte élevé, où le décodage ML est absolument nécessaire : 550 Mb/s (10 fois plus rapide) en moyenne [3] ;

5.4 Conclusions

Dans ce chapitre nous avons montré que les codes LDPC-staircase, que l'équipe PLANETE a standardisé à l'IETF (RFC5170 [99]), peuvent approcher les capacités de

[2] Puisque l'ordre de transmission est une permutation aléatoire, le pourcentage moyen de symboles de redondance reçus est constant, quelle que soit la probabilité d'effacement, et ne dépend que du rendement du code. La complexité de décodage du code Reed-Solomon, qui ne dépend que du nombre de symboles sources à reconstruire, est donc constante durant ces tests

[3] Ce débit est différent de celui indiqué en section 4.4.3. En effet les mesures ont été effectuées à plus d'une année d'intervalle, et le décodeur ML a été amélioré entre temps.

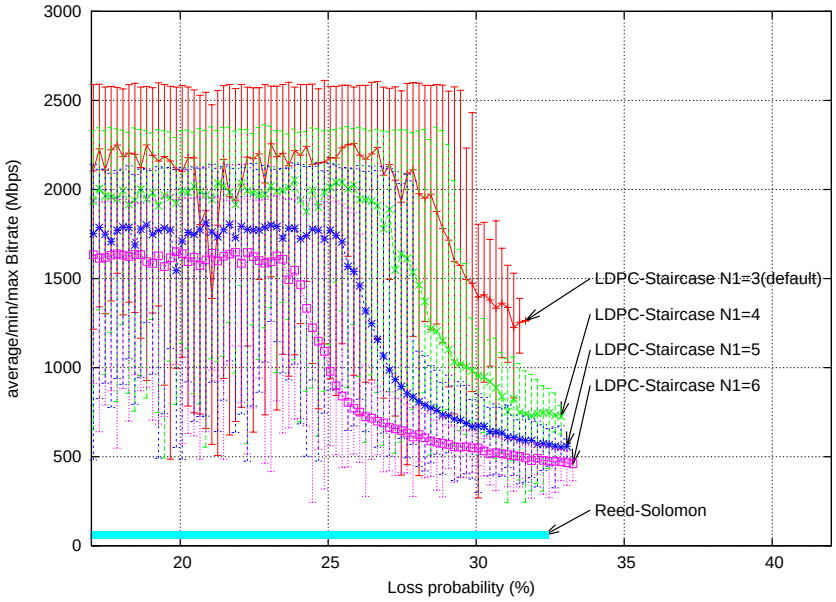


FIG. 5.7 – Débit en fonction du taux d'effacements (décodeur hybride IT/ML $k = 1000$ $R = 2/3$).

correction des codes idéaux, quand ils sont associés à un décodeur hybride IT/ML. Ceci est rendu possible par le contrôle du paramètre $N1$, qui détermine le degré des nœuds symboles sources. Par exemple, avec un code de dimension $k = 1000$, un rendement $R = 2/3$, et $N1 = 5$, le surcoût moyen de décodage s'élève à seulement $0,63\%$, et une probabilité d'échec du décodage de 10^{-4} est atteinte avec un surcoût égal à $2,21\%$.

Malgré l'augmentation de $N1$ cette solution reste très rapide. Les tests que nous avons effectués (i.e. code de dimension $k = 1000$, symboles de 1024 octets, rendement $R = 2/3$, $N1 = 5$) montrent que le décodage hybride est en moyenne 32,4 fois (décodage itératif) et 10,2 fois (décodage ML) plus rapide que le codec de référence Reed-Solomon sur GF(2^8).

Nous avons considéré que la valeur de $N1 = 5$ constitue un bon compromis entre les capacités de correction en ML, les capacités de correction en IT et la complexité de décodage. Dans le cas d'un décodeur hybride IT/ML nous choisirons donc la valeur $N1 = 5$ pour la construction du code LDPC-staircase.

Pour conclure, ce travail montre que malgré leur simplicité de construction, les codes LDPC-staircase utilisés avec un décodeur hybride IT/ML sont des codes efficaces, qui disposent : de bonnes capacités de corrections, d'une vitesse d'encodage naturellement élevée (grâce à leur structure héritée des codes "Repeat-Accumulate"), et de vitesses de décodage qui restent un ordre de grandeur supérieures à celles des code Reed-Solomon. Ils sont de plus flexibles, puisqu'il est possible d'adapter le paramètre $N1$ en fonction des types de décodeurs utilisés par les récepteurs (IT seulement ou hybride IT/ML), ou encore augmenter $N1$ pour de très petits objets.

Chapitre 6

Codes LDPC avec structure bande

Contents

Dans ce chapitre nous présenterons une nouvelle classe de codes pour le canal à effacements, les LDPC-Band, qui ont été conçus de manière à optimiser le décodage hybride IT/ML. En effet ces codes possèdent à la fois une matrice de parité creuse, qui permet une utilisation efficace d'un décodeur itératif, ainsi qu'une matrice génératrice possédant une structure bande, permettant ainsi de réduire la complexité du décodage par maximum de vraisemblance. Lorsqu'ils sont décodés avec une décodeur hybride, ces codes constituent un bon compromis entre la complexité et les capacité de corrections. Nous présenterons une méthode de construction de ces codes basée sur une représentation polynomiale des matrices.
Le travail présenté dans ce chapitre a été effectué en collaboration avec Alexandre Soro et Jérôme Lacan de l'ISAE et a fait l'objet d'une publication [107].

6.1 Introduction

Les codes LDPC sont une famille de codes capable d'atteindre d'excellentes performances de correction en étant décodés avec un algorithme à complexité linéaire, tandis qu'un décodage par maximum de vraisemblance permet de s'approcher encore plus près de

la capacité du canal au prix d'une complexité accrue. Nous avons montré que le décodage hybride IT/ML des codes LDPC constitue une solution performante et flexible en termes de capacité de correction et de complexité de décodage (voir chapitre 4).

Il a été montré que les codes aléatoires (codes dont la matrice génératrice est une matrice binaire remplie aléatoirement) possèdent des capacités de correction très proches de l'optimal [112]. Malheureusement la complexité de décodage de ces codes augmente rapidement avec la taille du code. En effet ces codes doivent être décodés avec un décodage par maximum de vraisemblance, qui peut être effectué par une inversion de matrice, dont la complexité est $O(k^2)$ opérations sur les lignes et les symboles associés de cette matrice.

Afin de réduire la complexité de décodage des codes aléatoires, Studholme et Blake [111] ont montré qu'une capacité de correction similaire à celle des codes aléatoires peut être obtenue quand les entrées non-nulles de la matrice génératrice sont concentrées dans une bande de largeur $2\sqrt{k}$ et dans laquelle chaque colonne contient $2\log k$ éléments non-nuls. Ces codes sont appelés "Windowed Erasure". Grâce à cette modification, la complexité de décodage est réduite à $O(k^{3/2})$ opérations sur les lignes.

La proposition de ce chapitre repose sur le décodage hybride IT/ML, dont le décodage ML est effectué sur la matrice génératrice (meilleure option pour les rendements faibles, i.e. $\leq 1/2$, voir chapitre 4). Notre objectif est de construire des codes de telle manière que leur matrice de parité et la matrice génératrice associée possèdent de bonnes propriétés pour les décodages associés. Plus précisément :

- la matrice de parité devra être creuse de manière à supporter efficacement le décodage itératif,
- la matrice génératrice devra avoir une structure bande afin de réduire la complexité du décodage ML, comme expliqué dans [112].

La combinaison de ces deux contraintes doit mener à la construction d'un code ayant des capacités de corrections proches de celles observées avec des codes LDPC standard, avec les décodages IT et ML, tout en ayant une complexité de décodage ML réduite.

L'idée d'encoder les codes LDPC en suivant une "fenêtre" (la matrice correspondante possède une structure bande) a été proposé par Haken, Luby et al. dans un brevet [45]. L'objectif ce brevet est d'optimiser les accès mémoire pendant l'étape d'encodage, en les localisant dans une fenêtre qui glisse le long du fichier à encoder. Indépendamment de la question de savoir si notre proposition tombe ou non dans le champs de ce brevet, nous observons, que d'un point de vue purement scientifique l'objectif de [45] diffère complètement de l'approche discutée dans ce chapitre, ainsi que des outils théoriques que nous avons déployés pour atteindre notre objectif.

6.1.1 Matrices et polynômes

Notre approche repose en partie sur l'utilisation des relations qui peuvent exister entre certaines matrices et les polynômes à une variable. Nous commencerons donc par présenter les objets et les propriétés qui seront utilisés dans la suite du propos.

Définition Une matrice de Toeplitz est une matrice carré $T = \{t_{i,j}; i, j = 0 \ldots n-1\}$ où $t_{i,j} = t_{i-j}$ (voir figure 6.1). Elle est donc définie par n éléments $\{t_k\}_{0 \leq k \leq n-1}$. Dans une telle matrice, les diagonales descendantes sont toutes constantes. On définit $T(X) \in \mathbb{F}_2[x]^{n-1}$ le polynôme associé à cette matrice qui s'écrit :

$$T(X) = \sum_{k=0}^{n-1} t_k X^k$$

$$T = \begin{pmatrix} t_0 & t_1 & t_2 & \dots & & \dots & t_{n-3} & t_{n-2} & t_{n-1} \\ 0 & t_0 & t_1 & t_2 & \dots & & \dots & t_{n-3} & t_{n-2} \\ 0 & 0 & t_0 & t_1 & t_2 & \dots & & \dots & t_{n-3} \\ \vdots & & & & & & & & \vdots \\ & & & & & & & & \\ \vdots & & & & & & & & \vdots \\ 0 & \dots & & & \dots & 0 & t_0 & t_1 & t_2 \\ 0 & \dots & & & & \dots & 0 & t_0 & t_1 \\ 0 & \dots & & & & & \dots & 0 & t_0 \end{pmatrix}$$

FIG. 6.1 – Matrice de Toeplitz

On notera $\mathbb{F}_2[x]^n = \{p \in \mathbb{F}_2[x] | degree(p) \leq n\}$, l'ensemble des polynômes de degré inférieur ou égal à n. Afin de rendre cet ensemble stable par rapport au produit interne, le produit de 2 éléments de $\mathbb{F}_2[x]^n$ sera tronqué en prenant le résultat modulo x^{n+1}.

Théorème 6.1.1 *Soit $G \in M_{n\times n}(\mathbb{F}_2)$ et soit $\{g_i\}_{0\leq i\leq n-1} \in (\mathbb{F}_2[x]^{n-1})^n$ les polynômes associés aux colonnes de $G = (g_0(x), g_1(x), \dots, g_n(x))$. Ces polynômes peuvent s'écrire, $\forall i \in [0..n]$, $g_i(x) = \sum_{j=0}^{n-1} g_{i,j}x^j$.*

Soit T une matrice de Toeplitz et $t(x)$ sont polynôme associé, alors les colonnes du produit des matrices sont égales aux produits des polynômes associés modulo x^n :

$$TG = (t(x)g_0(x) \bmod x^n, t(x)g_1(x) \bmod x^n, \dots, t(x)g_n(x) \bmod x^n) \tag{6.1}$$

$$\begin{pmatrix} t & & & & \\ & t & & & \\ & & t & & \\ & & & \ddots & \\ & & & & t \end{pmatrix} \begin{pmatrix} g_{0,0} & g_{1,0} & & g_{n-1,0} & g_{n,0} \\ g_{0,1} & g_{1,1} & \cdots & g_{n-1,1} & g_{n,1} \\ g_{0,2} & g_{1,2} & & g_{n-1,2} & g_{n,2} \\ & & & & \\ \vdots & \vdots & & \vdots & \vdots \\ & & & & \\ g_{0,n-1} & g_{1,n-1} & \cdots & g_{n-1,n-1} & g_{n,n-1} \end{pmatrix}$$

$$= \begin{pmatrix} t.g_0 & t.g_1 & \cdots & t.g_{n-1} & t.g_n \\ & & & & \end{pmatrix}$$

Preuve Soit $T = \{a_{i,j}\}_{0\leq i\leq n, 0\leq j\leq n}$, $G = \{b_{i,j}\}_{0\leq i\leq n, 0\leq j\leq n}$, avec $b_{i,j} = g_{j,i}$, et soit le produit $TG = C = \{c_{i,j}\}_{0\leq i\leq n, 0\leq j\leq n}$

commençons par calculer le produit matriciel :

$$
\begin{aligned}
c_{i,j} &= \sum_{k=0}^{n} a_{i,k} b_{k,j} = \sum_{k=0}^{n} a_{i,k} g_{j,k} && (6.2)\\
&= \sum_{k=0}^{j} a_{i,k} g_{j,k} + \sum_{k=j+1}^{n} a_{i,k} g_{j,k} && (6.3)\\
&= \sum_{k=0}^{j} u_{i-k} g_{j,k} + \sum_{k=j+1}^{n} 0 \times g_{j,k} && (6.4)\\
&= \sum_{k=0}^{j} u_{i-k} g_{j,k} && (6.5)
\end{aligned}
$$

d'autre part, le produit polynomial nous donne :

$$
\begin{aligned}
g_j(x)u(x) \bmod x^n &= \left(\sum_{j=0}^{n-1} g_{i,j} x^j\right)\left(\sum_{j=0}^{n-1} u_j x^j\right) \bmod x^n && (6.6)\\
&= \sum_{i=0}^{n-1}\left(\sum_{k=0}^{i} g_{j,k} u_{i-k}\right) x^i && (6.7)\\
&= \sum_{i=0}^{n-1} c_{i,j} x^i && (6.8)
\end{aligned}
$$

les éléments des colonnes de la matrice C sont donc bien égaux aux coefficients des produits des polynômes t et $\{g_i\}_{0\leq i\leq n-1}$.

6.2 Construction du code

6.2.1 Structure globale du code

Considérons un code systématique de dimension k et de longueur n. G la matrice génératrice de ce code est de taille $k \times n$, et H la matrice de parité associée est de taille $(n-k) \times n$. Soit $M \in \mathscr{M}_{k\times(n-k)}(\mathbb{F}_2)$ formée des $(n-k)$ dernières lignes de G. Soit $A \in \mathscr{M}_{(n-k)\times k}(\mathbb{F}_2)$ la matrice de taille $(n-k) \times k$ composée des k premières colonnes de H et soit $U \in \mathscr{M}_{(n-k)\times(n-k)}(\mathbb{F}_2)$ la matrice composée des $(n-k)$ dernières colonnes de H. G et H peuvent ainsi s'écrire :

$$G = (Id|M)^T \qquad H = (A|U)$$

Nous allons supposer que U est une matrice de Toeplitz, afin de profiter des relations existant avec les polynômes vues dans la section précédente. Nous faisons également l'hypothèse qu'elle possède une structure triangulaire inférieure, afin de permettre un encodage à complexité linéaire (voir section 2.5.4).

La matrice U est donc complètement déterminée par sa première colonne, que l'on choisit comme étant égale à $(1, u_1, \ldots, u_{n-k-1})$. Ainsi les éléments de la diagonale de U sont tous égaux à 1, U est donc de rang plein. De plus on considère le polynôme associé :

$$u(x) = 1 + \sum_{k=1}^{n-k-1} u_k x^k \tag{6.9}$$

les coefficients de U sont les suivants :

$$\{u_{i,j}\} = \begin{cases} u_{i-j} & si \quad i > j \\ 0 & si \quad i < j \\ 1 & si \quad i = j \end{cases}$$

et la matrice U possède la forme suivante :

$$U = \begin{pmatrix} 1 & 0 & 0 & \dots & & 0 & 0 \\ u_1 & 1 & 0 & & & & \\ u_2 & u_1 & 1 & & & \vdots & \vdots \\ \vdots & u_2 & u_1 & \ddots & & & \\ u_i & \vdots & u_2 & & \ddots & 0 & \\ \vdots & u_i & \vdots & & & 1 & 0 \\ u_{n-k-1} & \vdots & u_i & & & u_1 & 1 \end{pmatrix}$$

Après avoir déterminé la matrice U, nous passons à la construction de la matrice M. On rappelle que notre objectif est de construire un code tel que M possède une structure bande. Soit B la largeur de cette bande. La matrice M peut alors s'écrire sous la forme suivante :

$$M = \begin{pmatrix} m_{0,0} & m_{0,1} & \dots & m_{0,B-1} & 0 & \dots & 0 & 0 \\ 0 & m_{1,0} & \ddots & \dots & m_{1,B-1} & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \ddots & \dots & \ddots & 0 & \vdots \\ \vdots & \vdots & 0 & m_{k-1,0} & \dots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & m_{k,0} & \ddots & \dots & m_{k,B-1} \end{pmatrix}$$

Pour $i = 0, \dots, (k-1)$, on défini le polynôme $m_i(x) = \sum_{j=0}^{B-1} m_{i,j} x^j$, dont les coefficients correspondent aux éléments de la i-ème ligne de M. On remarque que l'on peut déduire de la matrice M une relation entre la longueur du code, sa dimension et la largeur de la bande :

$$n - k = k + B - 1 \tag{6.10}$$

Puisque la matrice G est sous forme systématique, on sait qu'il existe une relation entre cette matrice et la matrice de parité H (voir section 2.3), et plus particulièrement entre U, M et A :

$$UM^T = A \tag{6.11}$$

d'après le théorème 6.1.1 on déduit que $\forall i \in \{0, \dots, k-1\}$ la i-ème colonne de A est composée des coefficients du polynôme $a_i(x)$ qui est tel que $a_i(x) = u(x)m_i(x)$ (voir figure 6.2). Ce résultat va nous permettre de construire des matrices génératrices bande prenant en compte les contraintes sur les colonnes de la matrice de parité.

On rappelle que le poids de Hamming d'une séquence est le nombre d'éléments non-nuls dans celle-ci. De la même manière on définit le poids de Hamming d'un polynôme $a_i(x)$ comme étant le poids de Hamming de la séquence formée par ses coefficients, que l'on notera $W(a_i)$.

$$H = \left(\begin{array}{cccc|cccc} a_{0,0} & & & & u_0 & & & \\ a_{0,1} & a_{1,0} & & & u_1 & u_0 & & \\ a_{0,2} & a_{1,1} & a_{2,0} & & u_2 & u_1 & u_0 & \\ a_{0,3} & a_{1,2} & a_{2,1} & & u_3 & u_2 & u_1 & \\ \vdots & a_{1,3} & a_{2,2} & \ddots & \vdots & u_3 & u_2 & \ddots \\ & \vdots & a_{2,3} & & & \vdots & u_3 & \\ & & \vdots & & & & \vdots & \\ & & & & & & & \end{array}\right)$$

FIG. 6.2 – Matrice de parité avec une structure bande dans la partie gauche, et triangulaire inférieure dans la partie droite

6.2.2 Construction des matrices

Afin de construire un code supportant efficacement le décodage hybride IT/ML, la première contrainte est d'obtenir une matrice de parité supportant le décodage itératif. Ce type de décodage fonctionne efficacement lorsque la matrice de parité sur laquelle il travaille est creuse. Dit autrement, il faut que les colonnes et les lignes de la matrice de parité soient de faible degré, i.e. de faible poids de Hamming.

En utilisant l'approche polynomiale, et en considérant les polynômes associés, cela revient à dire que les polynômes u et $\{a_i\}_{0\leq i\leq k-1}$ sont de faible poids de Hamming. Plus précisément, on défini un ensemble de degré $\{d_i\}_{0\leq i\leq n_d}$ que l'on considère comme assez faible pour permettre un décodage itératif efficace. Nous verrons plus tard comment le poids des lignes peut être contrôlé.

La seconde contrainte concerne de décodage ML. Comme expliqué précédemment, la matrice génératrice doit posséder une structure bande afin de réduire la complexité du décodage. La traduction polynomiale de cette contrainte, est que les polynômes $\{m_i\}_{0\leq i\leq k-1}$ sont tous de degré inférieur ou égal à B. Ces polynômes devront aussi avoir un degré proche de B de manière à ne pas obtenir une largeur de bande inférieure à la largeur cible B :

$$B - deg(m_i) \leq \delta_B \tag{6.12}$$

δ_B étant la marge tolérable sur la largeur de bande.

Ainsi les coefficients égaux à 1 de la matrice M seront tous concentrés dans une bande de largeur B. Aucune autre contrainte n'est imposée à la matrice génératrice, de manière à ce qu'elle soit suffisamment "aléatoire" pour offrir de bonnes capacités de corrections avec le décodeur ML.

Le point central de la construction du code est de trouver des polynômes $m_i(x)$ de degré inférieur ou égal à B, tel que $a_i(x)$ et $u(x)$ ont peu de monômes, où $a_i(x) = m_i(x)u(x)$. Ces polynômes seront appelés *polynômes candidats*. La recherche de ces polynômes candidats en fonction des paramètres du code (n,k,B,δ_B), des degrés cibles des colonnes $\{d_i\}_{0\leq i\leq n_d}$, peut être effectuée grâce à une recherche exhaustive.

On suppose qu'après cette recherche, l'ensemble des polynômes candidats est non vide. On utilise alors ces polynômes pour remplir les lignes de la matrice génératrice, et on calcule les colonnes correspondantes de la matrice de parité. Ces polynômes peuvent être choisis aléatoirement parmi les polynômes candidats ou de manière à ce qu'une distribution de degrés spécifique pour les colonnes de la matrice de parité soit obtenue.

D'une manière générale il est préférable d'introduire de la diversité dans la matrice, en entrelaçant les différents polynômes. En effet une suite de lignes obtenues à partir du même polynôme va engendrer la création d'un motif régulier dans la matrice, ce qui peut avoir un impact négatif sur le décodage ML. Nous avons observé que l'entrelacement permet une amélioration substantielle des performances en termes de capacité de correction du décodage ML. Les expériences ont également montré que la façon dont les polynômes sont permutés dans la matrice n'a qu'une influence limitée sur les performances du code. Une permutation aléatoire semble donc suffisante.

Une fois que le polynôme u et que l'ensemble des polynômes m_i ont été sélectionnés, le degré des colonnes de la matrice de parité est fixé, par le degré de u pour la partie droite, et par les degrés des m_i pour la partie gauche. Il est alors possible d'ajuster le degré des lignes de manière à atteindre, ou du moins s'approcher, d'une distribution spécifique. En effet la permutation de deux polynômes m_i et m_j ne changera pas la distribution des degrés des colonnes, et aura une influence limité sur les performances du code, mais pourra modifier le degré des lignes de la matrice. Néanmoins, le nombre de permutations possibles rend le champs de recherche extrêmement grand, et il n'est pas garantie que l'on puisse atteindre la distribution cible pour les degrés lignes.

6.2.3 Optimisations

La construction précédente contraint la longueur du code à être égale à $n = 2k + B$. De plus ces codes se sont montrés peu efficaces avec le décodage ML en termes de capacité de correction. En effet, les premiers symboles de redondance ne protègent qu'un petit nombre de symboles source. La structure bande fait que le premier symbole de parité ne pourra protéger que le premier symbole source, étant donné que les autres coefficients de la première colonne de M sont obligatoirement nuls.

Ce problème peut être résolu en utilisant un second ensemble de polynômes, de degré inférieur ou égal à $B/2$, pour construire les premières et dernières ligne de la matrice de parité. Cet ensemble de polynôme doit satisfaire les même contraintes que celle imposées précédemment pour polynômes candidats, exception fait de leur degré. Ainsi en utilisant de tels polynômes pour construire les $B/2$ premières lignes et les $B/2$ dernières lignes de M (voir figure 6.3), on augmente le nombre de connexions entre les premiers symboles de redondance et les symboles sources. En plus d'améliorer les capacités de correction, cette modification permet de retirer la dépendance de la longueur du code, n, vis à vis de la largeur de la bande B. Ainsi la longueur du code peut maintenant être $n = 2k$.

Une autre limite de cette construction de codage est qu'il ne supporte maintenant qu'un rendement, $R = 1/2$. La seconde optimisation consistera donc à permettre une flexibilité de ce rendement. Pour cela nous allons ajuster le décalage de positions entre les polynômes des lignes consécutives de la matrice génératrice. Soit f la valeur de ce décalage entre deux lignes consécutives. Pour un rendement $R = 1/2$ la valeur de ce décalage est $f = 1$, en effet d'une ligne à l'autre, le polynôme est placé une position vers la droite par rapport au précédent. Nous posons $f = 1/R - 1$. Si f n'est pas un entier, il suffit alors d'utiliser une famille d'entiers $\{f_i\}$ tels que la moyenne des valeurs de cette famille est f. Afin de conserver l'optimisation précédente, nous fixons à $B/(2f)$ le nombre de lignes, au début et à la fin de la matrice M, à construire avec des polynômes de degré $B/2$.

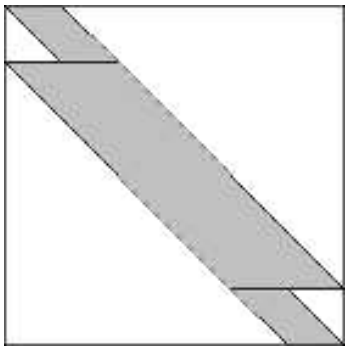


FIG. 6.3 – Matrice bande de largeur B, ayant les $B/2$ premières et $B/2$ dernières lignes remplacées par des lignes de largeur de bande $B/2$

Analyse de l'évolution de densité : Nous avons étudié la faisabilité d'une approche du type évolution de densité (DE) [94] afin d'optimiser les performances du code lorsqu'il est décodé itérativement (voir section 2.5.5), mais avons alors fait face à plusieurs obstacles.

On rappelle que pour effectuer une étude avec l'évolution de densité, on doit faire l'hypothèse que le code est de longueur infinie. La largeur de la bande dépendant de la longueur du code, il nous faudrait alors considérer une bande de largeur infinie. L'évolution de densité nous renverrait alors vers des distributions optimales des codes "Repeat-Accumulate" [33], dont notre code est un cas particulier.

Avec notre construction, la distribution des degrés colonnes peut être immédiatement adaptée. Cependant, obtenir une distribution de degré précise sur les lignes est beaucoup moins évident, puisque comme expliqué plus tôt cette distribution dépend fortement du choix des polynômes et de leurs permutations. De plus comme la bande est relativement étroite, une forte irrégularité dans les distributions jointes [51] peut apparaître. Dit autrement, du point de vue des nœuds, les distributions des degrés des lignes et des colonnes ne seront pas totalement indépendantes.

D'un autre coté, si nous faisons l'hypothèse d'une bande de largeur finie, l'hypothèse des cycles de longueur infinie, nécessaire à l'évolution de densité, n'est plus valable. Cette hypothèse est vraie dans un code de longueur infinie, puisque chaque nœud de contrôle peut être potentiellement connecté à n'importe quel nœud symbole du code. Ce qui n'est pas le cas dans notre approche, puisqu'à cause de la structure bande, un nœud de contrôle sera potentiellement connecté à un ensemble fini de nœuds symboles.

Tous ces éléments font que l'approche par évolution de densité n'est pas triviale, nous en sommes donc restés à une méthode empirique pour le choix des degrés.

6.3 Analyse théorique

6.3.1 Capacités de correction théorique

Influence de la largeur de bande sur les performances ML : Concernant les performances de ces codes avec un décodeur ML, il est nécessaire de trouver une largeur de bande qui permet d'avoir une bonne vitesse de décodage tout en conservant de bonnes capacités de correction. La première contrainte nous pousse à choisir une bande la moins large possible. Cependant cette largeur doit être suffisamment grande pour satisfaire la

seconde contrainte. Il nous faut donc étudier l'influence de cette largeur de bande afin de trouver un compromis.

Studholme et Blake [111][112] ont étudié l'influence de la largeur de la bande sur des matrices binaires aléatoires. La conclusion de leurs travaux est que la taille de cette bande doit être environ de $2\sqrt{k}$ et que le nombre d'éléments égaux à 1 dans chaque colonne doit être au moins $2\log k$. Ces valeurs garantissent que la probabilité d'obtenir une matrice de décodage de rang plein est proche de celle obtenu avec des matrice aléatoires non contraintes. Notre approche considérant des configurations spécifiques des matrices bande, ces valeurs doivent être considérées comme des minimums.

Capacités de corrections du décodage itératif : Les LDPC-Band étant construits sur la base des codes "Repeat-Accumulate", il est raisonnable de penser que leurs performances en termes de capacité de correction soient proche de celles de ces dernier. Néanmoins, la contrainte bande va réduire la diversité des connexions dans le graphe de Tanner correspondant. On peut donc s'attendre à ce que la version contrainte possède des performances inférieures à la version non contrainte que sont les codes RA.

Un code LDPC-Band ayant une largeur de bande $B = n - k$ sera en fait un code RA normal (bonnes capacités de correction), alors qu'à l'opposé , un code ayant une largeur de bande $B = 1$ sera un simple code à répétition (capacités de correction médiocres). La capacité de correction en décodage itératif sera donc également affectée par la largeur de la bande.

6.3.2 Complexité théorique

Le décodeur hybride IT/ML comprend deux types de décodage. Le premier, le décodage itératif, travaillant sur la matrice de parité, est rapide et possède une complexité linéaire en la taille du code, $O(k)$. Lorsque ce premier décodeur échoue, il laisse la main au décodeur ML qui va essayer de résoudre le système linéaire représenté par la matrice de décodage issue de la matrice génératrice. Dans le cas général, la complexité d'un tel décodage est en $O(k^2)$ opérations sur les lignes. Cependant, grâce à la structure bande de la matrice génératrice, la matrice de décodage possède également une structure bande. En effet la matrice de décodage est constituée d'un sous-ensemble des lignes et des colonnes de la matrice génératrice, et elle possède donc également une structure bande de même largeur.

Grâce à une décomposition LU optimisée [44] de la matrice de décodage, la complexité de décodage tombe à $O(kB)$ opérations sur les lignes. Cela veut dire que pour un code de dimension $k = 2000$ et une bande de largeur $B = 200$, le gain théorique en vitesse est d'un ordre de grandeur. Cette complexité est à mettre en regard avec celle du décodage ML des codes "Windowed Erasure" qui est de $O(k^{2/3})$ opérations sur les lignes. Il faut noter qu'en général, la taille du système, qui doit être résolu par le décodeur ML sur matrice génératrice d'un système hybride, est inférieure à k (voir chapitre 4).

6.4 Résultats de simulation

Des tests ont été effectués de manière à confirmer les bénéfices théoriques de notre solution. Nous avons effectué ces tests sur un PC 4xIntelXeon5120 @ 1.86GHz/4 GB RAM/ GNU-Linux, avec une version améliorée du codec libre [118] supportant les décodages IT,

ML et hybrides IT/ML . Les LDPC-Band ont alors été comparés avec deux autres types de codes :

1. les codes LDPC-staircase, qui sont des codes “Repeat-Accumulate” [33] (voir 2.5.4) . Suivant les recommandations proposées dans [27] concernant le décodage hybride IT/ML, le paramètre $N1$, qui représente le degré des colonne sources, a été fixé à 5.
2. les codes ”Windowed Erasure“ [111]. Conformément aux spécifications de l'article original, la matrice génératrice possède une bande de largeur $2\sqrt{(k)}$ contenant $2\log k$ éléments placés aléatoirement dans chaque colonne. Ces codes, tels que décrits dans [111], ne supportent pas le décodage itératif.

Nous avons choisi de comparer les LDPC-Band aux LDPC-staircase à cause de la similarité de leur structure. En effet, même si la partie droite de la matrice peut différer par son degré, les codes LDPC-Band et les codes LDPC-staircase présentent une structure similaire de code “Repeat-Accumulate” [33]. Cette comparaison nous permettra d'évaluer l'influence de la contrainte bande sur les performances du code.

De la même manière la comparaison avec les ”Windowed Erasure“ nous permettra d'évaluer l'influence de la contrainte de supporter un décodage itératif, en plus d'un décodage ML sur une matrice bande.

La matrice du code LDPC-Band utilisée a été obtenue avec la méthode présentée dans la section 6.2. Les polynômes ainsi que les permutations ont été choisis de manière aléatoire.

6.4.1 Capacités de correction

Les tables 6.1 et 6.2 présentent les ratio d'inefficacité de codes de rendement $R = 1/2$, pour le décodage IT et le décodage ML.

Type de décodeur	IT	ML
LDPC-staircase $N_1 = 5$	14.24%	1.21%
LDPC-Band - B = 100	18.39%	2.97%
LDPC-Band - B = 200	14.75%	1.24%
”Windowed Erasure“ - b = 63	NA	0.17%

TAB. 6.1 – Inefficacité moyenne en fonction du type de décodage ($k = 1000$ $R = 1/2$)

Type de décodeur	IT	ML
LDPC-staircase $N_1 = 5$	13.95%	1.15%
LDPC-Band - B = 200	16.23%	1.19%
”Windowed Erasure“ - b = 89	NA	0.79%

TAB. 6.2 – Inefficacité moyenne en fonction du type de décodage ($k = 2000$ $R = 1/2$)

Comme attendu, la largeur de la bande possède une grande influence sur la capacité de correction du code, aussi bien en itératif qu'en ML. Avec une bande de largeur $B = 200$, les LDPC-Band ont une inefficacité de décodage itératif sensiblement égale à celle des LDPC-staircase, alors qu'ils sont légèrement moins bons en décodage ML. De plus, même avec

une largeur de bande sensiblement inférieure, les codes "Windowed Erasure" sont meilleurs que les codes LDPC-Band. Cela est vraisemblablement dû à l'absence de contraintes liées à la matrice de parité des LDPC-Band, qui permet d'avoir une véritable distribution aléatoire des entrées dans la bande. Ces résultats montrent que les LDPC-Band ont une capacité de correction inférieure, mais néanmoins très proche de celle des LDPC-staircase, alors qu'ils sont plus contraints.

6.4.2 Vitesses de décodage

La figure 6.4 montre le débit moyen du décodage, en fonction du taux d'effacements du canal, pour les différents codes. A cause du décodage hybride, les performances des LDPC-Band et LDPC-staircase peuvent être divisées en deux parties. Dans la région où le pourcentage de perte est faible, le décodeur itératif est suffisant pour récupérer les effacements, et le débit est élevé. Lorsque la pourcentage de perte augmente, le décodage ML est de plus en plus souvent nécessaire, réduisant ainsi le débit [1]. Quant aux codes "Windowed Erasure", le débit du décodeur ne dépend pas du taux de perte puisque ces codes ne peuvent être décodés qu'avec un décodeur ML.

Lorsque le décodeur itératif est utilisé, les codes LDPC-Band et LDPC-staircase possèdent des débits de décodage similaires (avec un léger avantage pour les LDPC-staircase), et ils sont significativement plus rapides que les "Windowed Erasure" (qui utilisent alors un décodeur ML).

Lorsque le décodeur ML est nécessaire, les LDPC-Band surpassent les deux autres classes de code. En effet ils sont environ deux fois plus rapides que les "Windowed Erasure" et environ quatre fois plus rapides que les codes LDPC-staircase. La différence entre les LDPC-Band et les LDPC-staircase est une conséquence directe de la structure bande qui accélère la résolution du système linéaire. La différence entre les LDPC-Band et les "Windowed Erasure", quant à elle, vient de la taille du système à résoudre. En effet pour les LDPC-Band, le décodeur ML n'a à résoudre qu'un système de taille environ $((1-R)k) \times ((1-R)k)$ (voir section 4.2.2), moins une éventuelle simplification (voir section 4.3.1), alors que pour les "Windowed Erasure" le système est de taille $k \times k$. En effet les "Windowed Erasure" ne sont pas systématiques, ce qui veut dire que les symboles sources ne sont pas transmis et sont donc tous inconnus pour le décodeur. Alors qu'avec un code systématique Rk symboles sources sont reçus en moyenne, réduisant ainsi le nombre de symboles à reconstruire.

Le tableau 6.3 présente les vitesses de décodage ML moyenne pour les 3 codes comparés en fonction de la dimension du code k. Le LDPC-Band est toujours plus rapide que les autres codes. De plus le débit chute plus vite en fonction de la taille pour les deux autres codes. Les vitesses moyennes de décodage itératif sont présentées dans le tableau 6.4. Le débit des LDPC-Band est inférieur à celui des LDPC-staircase, mais en reste très proche.

[1] On peut remarquer que le débit augmente légèrement avec le pourcentage d'effacements lorsque le décodeur ML est utilisé. Lorsque le taux de perte diminue, le nombre de lignes de matrice de décodage augmente alors que le nombre de colonnes diminue, puisque plus de symboles de parité sont reçus et que moins de symboles sources sont manquants. Alors que la complexité de décomposition de la matrice reste globalement la même, la taille du membre constant augmente avec le nombre de lignes, induisant ainsi une complexité supplémentaire.

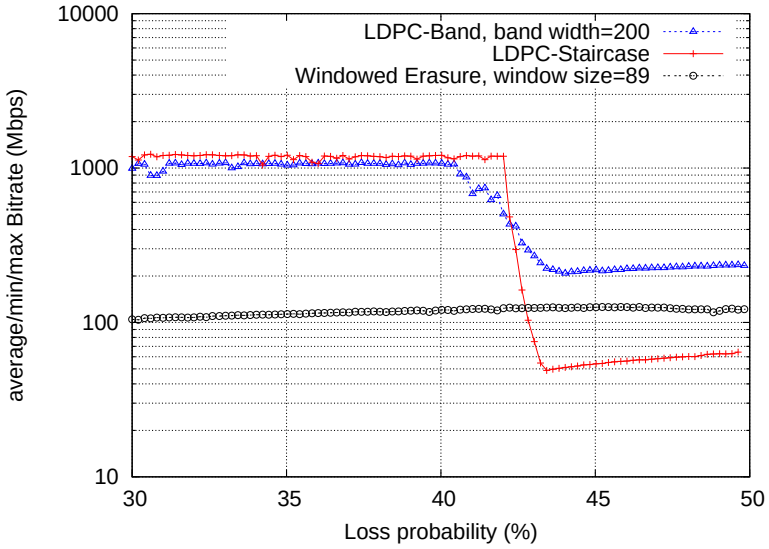


FIG. 6.4 – Débit en fonction du pourcentage de perte (k=2000 r=1/2 et taille des symboles =1024 octets)

k	1,000	2,000	4,000
LDPC-Band	326 Mbps	235 Mbps	150 Mbps
LDPC-staircase	125 Mbps	60 Mbps	30 Mbps
Windowed Erasure	220 Mbps	120 Mbps	68 Mbps

TAB. 6.3 – Débit de décodage ML moyen en fonction de la dimension du code, pour r=1/2 et taille symboles = 1024 octets.

6.5 Conclusions

Dans ce chapitre nous avons proposé une approche qui permet la construction de matrices génératrices et matrices de parité optimisées pour un décodage hybride IT/ML efficace. Le décodage itératif effectué sur la matrice de parité constitue alors une solution à faible complexité pour récupérer les symboles effacés. Lorsque les conditions de transmission deviennent trop difficiles, le décodage ML permet de s'approcher de la capacité du canal. La complexité de ce décodage est réduite à $O(kB)$ grâce à une structure bande, de largeur B, de la matrice génératrice. Cette construction permet au décodeur hybride de conserver des vitesses de décodage élevées, même lorsque le taux de perte est important. Grâce à une représentation polynomiale des matrices, nous avons présenté une méthode pratique pour construire de telles matrices.

Les résultats présentés montrent que les LDPC-Band sont très proches, en termes de vitesse, des codes LDPC-staircase lorsque le décodage itératif est utilisé, tout en étant

k	1,000	2,000	4,000
LDPC-Band	1100 Mbps	1050 Mbps	900 Mbps
LDPC-staircase	1300 Mbps	1200 Mbps	1000 Mbps

TAB. 6.4 – Débit de décodage IT moyen en fonction de la dimension du code, pour R=1/2 et taille symboles = 1024 octets.

sensiblement plus rapides lorsque le décodeur ML est nécessaire. En termes de capacités de correction, les LDPC-Band sont légèrement moins bons que les codes LDPC-staircase sous décodage IT. Lorsque le décodage ML est utilisé, les LDPC-Band sont comme les LDPC-staircase, proche de la capacité du canal, avec un léger avantage pour les LDPC-staircase.

De futurs travaux auront pour but d'améliorer la construction des matrices au niveau des distributions des lignes. Une analyse évolution de densité des LDPC-Band devra être conduite de manière à améliorer la capacité de correction sous décodage IT sans dégrader les performances du décodage ML.

Une approche similaire pourra être étudiée dans le cas où le décodage ML est effectué sur la matrice de parité. Une structure bande dans H_1 permettra d'obtenir une matrice de décodage possédant également une structure bande, réduisant ainsi la complexité de décodage. En effet une permutation des colonnes dans la matrice de décodage, permet de retrouver une structure bande à partir du sous-ensemble des colonnes de deux matrices bandes, H_1 et H_2. L'avantage est que cette approche impose moins de contraintes sur la construction du code. Le degré de liberté supplémentaire pourra par exemple être mis à profit pour construire plus facilement des matrices bande irrégulières.

Chapitre 7

Codes LDPC quasi-cycliques pour le canal à effacements

Contents

Ce chapitre présente l'utilisation des codes LDPC Quasi-cycliques sur le canal à effacements. Nous montrons que la structure particulière de la matrice de parité confère à ces codes deux avantages :

– *un décodage ML à complexité réduite, grâce à un remplissage plus lent de la matrice lors de sa résolution par des techniques d'élimination ;*
– *une amélioration des capacités de correction du décodage itératif via une diminution du ratio d'inefficacité, tout en conservant d'excellentes performances en ML.*

Les travaux présentés dans ce chapitre constituent une étude préliminaire permettant d'apprécier les avantages de ces codes. Une analyse plus approfondie ainsi que des optimisations de ces codes seront prochainement effectuées.
Ce travail est effectué en collaboration avec Valentin Savin du CEA-LETI.

7.1 Introduction

Le décodage hybride IT/ML des codes LDPC permet d'atteindre des capacités de correction proches de l'optimal tout en conservant une vitesse de décodage élevée la plupart du temps. Cependant, lorsque le décodage ML est nécessaire, la complexité accrue qui en résulte induit inévitablement une diminution du débit de décodage.

Pour les systèmes linéaires possédant une structure spécifique la complexité de la résolution du système peut être réduite. Nous avons montré dans le chapitre précédent que

l'utilisation d'une structure bande permet de réduire significativement la complexité de la résolution du système. Cependant cette structure implique des contraintes sur le code qui tendent à dégrader leurs capacités de correction.

Une autre famille de codes LDPC possédant une matrice structurée est constituée par les codes LDPC quasi-cycliques. En appliquant la construction quasi-cyclique [117] aux codes "Repeat-Accumulate", Tanner introduisit en 1999 les codes "Repeat-Accumulate" quasi-cycliques [114] qui peuvent être vus comme des codes LDPC quasi-cycliques. Les matrices de parité de ces codes possèdent une structure cyclique par bloc. De nombreux travaux ont montré l'intérêt de ces codes pour différents canaux [53]. En particulier, il a été montré [40] que cette approche permet de construire des codes LDPC ayant une longueur minimale de cycle élevée, améliorant ainsi les capacités de correction de ces codes.

Dans ce chapitre nous présentons une analyse préliminaire des codes LDPC-QC munis d'un décodeur hybride IT/ML pour le canal à effacements. Nous commencerons par introduire les codes LDPC-QC et les propriétés associées à ces codes, puis nous présenterons les résultats sur les capacités de correction obtenues par simulations. Nous terminerons par une analyse de la complexité du décodage ML de ces codes, et nous mettrons en avant les bénéfices apportés par la structure de la matrice de parité lors de la résolution du système linéaire au sein du décodage ML.

7.2 Codes LDPC quasi-cycliques

Les codes LDPC quasi-cycliques ont été introduit par Tanner sous le nom de "quasi-cyclic reapeat-accumulate codes" [114]. Comme leur nom l'indique, ils appartiennent à la classe des codes quasi-cycliques qui sont eux même une généralisation des codes cycliques.

7.2.1 Codes quasi-cycliques

Les matrices des codes quasi-cycliques peuvent être représentées à l'aide de matrices circulantes. Une matrice circulante A est une matrice carrée $p \times p$ dans laquelle chaque ligne est un décalage cyclique d'une position à droite de la ligne précédente. Il existe un isomorphisme entre l'algèbre des matrices circulantes binaires $p \times p$ et l'algèbre des polynômes dans l'anneau quotient $\mathbb{F}_2[X]/(X^p - 1)$ [1]. On associera à la matrice circulante A, définie par sa première ligne $\{a_0, a_1, \ldots, a_{p-1}\}$ (voir figure 7.1), le polynôme $a(X) = \sum_{i=0}^{p-1} a_i X^i$.

Un code $\mathscr{C}$ de longueur $n = t.p$ et de dimension $k = s.p$ est dit quasi-cyclique de période t si le décalage à droite de t positions d'un mot du code $\mathscr{C}$ est encore un mot du code. La matrice de parité d'un tel code peut s'écrire comme une matrice composée de matrices circulantes de tailles $p \times p$. p sera alors appelé *facteur d'expansion*. En utilisant la notation polynomiale des matrices circulantes on peut représenter la matrice de parité par une matrice $(t-s) \times t$ de polynômes de degrés strictement inférieurs à p (voir figure 7.2).

Les matrices de parité d'un code LDPC devant être creuses, les matrices circulantes des codes LDPC-QC doivent elles aussi être suffisamment creuses. Ceci implique que les polynômes associés doivent être soit nuls soit posséder un faible nombre de monômes.

[1] On se limite ici au cas binaire, mais ces résultats restent valables pour n'importe quel corps fini.

$$A = \begin{pmatrix} a_0 & a_1 & a_2 & \dots & & & & \dots & a_{p-3} & a_{p-2} & a_{p-1} \\ a_{p-1} & a_0 & a_1 & a_2 & \dots & & & & \dots & a_{p-3} & a_{p-2} \\ a_{p-2} & a_{p-1} & a_0 & a_1 & a_2 & \dots & & & & \dots & a_{p-3} \\ \vdots & & & & & & & & & & \vdots \\ a_2 & \dots & & & & \dots & a_{p-3} & a_{p-2} & a_{p-1} & a_0 & a_1 \\ a_1 & a_2 & \dots & & & & \dots & a_{p-3} & a_{p-2} & a_{p-1} & a_0 \end{pmatrix}$$

FIG. 7.1 – Matrice circulante

$$H = \begin{pmatrix} h_{0,0} & h_{0,1} & h_{0,2} & \dots & h_{0,t-1} \\ h_{1,0} & h_{1,1} & h_{1,2} & \dots & h_{1,t-1} \\ \vdots & & & & \vdots \\ h_{t-s-1,0} & h_{t-s-1,1} & h_{t-s-1,2} & \dots & h_{t-s-1,t-1} \end{pmatrix}$$

FIG. 7.2 – Représentation polynomiale d'une matrice de parité d'un code quasi-cyclique.

7.2.2 Codes LDPC-QC de type I

Nous allons nous concentrer sur un type particulier de codes LDPC quasi-cycliques, appelés codes LDPC-QC de type I [106], définis par des polynômes qui sont soit nuls soit ne possédant qu'un seul monôme. Ainsi les matrices circulantes composant la matrice de parité sont soit des matrices nulles soit des matrices identité décalées cycliquement à droite. A chacune de ces matrices circulantes est associé un entier $q \in [0..(p-1)]$ correspondant au degré du monôme. Ainsi la matrice $p \times p$, associée à la valeur q, notée S_q, est en fait une matrice identité décalée cycliquement de p positions à droite.

Un code LDPC-QC de type I peut donc être défini par une matrice B de taille $(t-s) \times t$ à valeurs dans $\{ [0..(p-1)] \cup \{-1\} \}$ qui sera appelée *matrice de base* du code LDPC-QC. A partir de cette matrice B et d'un facteur d'expansion p, la matrice de parité étendue est construite de la manière suivante : pour chaque entrée q de B, la sous matrice carrée associée dans H est égale à la matrice nulle si $q = -1$, ou à S_q si $q \in [0..(p-1)]$. Les matrices de parité ainsi obtenues hériteront de la même distribution de degré au niveau des lignes et des colonnes que la matrice de base B [2].

Longueur des cycles dans les codes QC-LDPC de type I

La taille des cycles dans un code LDPC-QC peut être analysée en fonction des cycles dans la matrice de base et des coefficients correspondants.

Considérons la matrice de base B , à partir de laquelle est construite H, en utilisant un facteur d'expansion p. Alors, pour chaque cycle C^B dans la matrice de base B, il correspondra p cycles différents $\{C_0^H, C_1^H, \dots, C_{p-1}^H\}$ dans la matrice étendue H. On dira que ces cycles *vivent au dessus* du cycle C^B. Soit C un cycle de longueur s dans la matrice de base et $\{b_0, \dots, b_{s-1}\}$ la liste des entrées composant ce cycle. Alors, tous les cycles vivant au dessus de C auront la même longueur.

[2] Seules les entrées ≥ 0 sont prise en compte pour calculer le degré de la matrice de base

$$B = \begin{pmatrix} & & & & & & \\ & b_0 & & \leftarrow & & b_{s-1} & \\ & \downarrow & & & & \vdots & \\ & b_1 & \rightarrow & b_2 & & & \\ & & & \downarrow & & & \\ & & & b_3 & \rightarrow & b_4 & \\ & & & & & & \end{pmatrix}$$

FIG. 7.3 – Cycle de longueur s dans la matrice de base d'un code quasi-cyclique.

Soit p le facteur d'expansion et soit b la somme alternée des valeurs des coefficients du cycle :

$$b = \sum_{i=0}^{s-1} (-1)^i b_i \tag{7.1}$$

La longueur de ces cycles, L, s'exprime de la manière suivante [40] :

$$L = \frac{s\,p}{\text{pgcd}(b,p)} \tag{7.2}$$

Ainsi, si b et p sont premiers, alors $\text{pgcd}(b,p) = 1$, et les cycles vivants au dessus de C seront de longueur maximum égale à $L_{\max} = sp$.

En utilisant cette information, et grâce à un choix judicieux des coefficients de la matrice de base, il est possible de construire des codes dont la longueur des cycles est la plus grande possible. Ceci permet de construire des codes dont le "girth" (i.e. la longueur du plus petit cycle) est grand. Il est donc possible d'utiliser cette construction pour optimiser les capacités de correction du code, puisque la distance minimale dépend linéairement du "girth" et que le décodage itératif fonctionne d'autant mieux que les cycles sont longs.

Cependant, comme nous le verrons en section 7.3, un choix aléatoire des coefficients est déjà suffisant pour permettre une amélioration significative des capacités de corrections obtenues avec un décodeur itératif.

Un cas particulier : les LDPC-QC staircase de type I

Dans ce travail nous considérons un ensemble de code LDPC-QC basés sur une construction issue des codes LDPC-staircase [99] auxquels nous les comparerons. En effet la matrice de base que nous utilisons est obtenue à partir d'une matrice d'un code LDPC-staircase, en remplaçant les entrées nulles par -1 et les entrées non-nulles par un entier compris dans $[0\,..\,q_{\max}]$. Etant donné que nous nous intéressons aux performances sous le décodage hybride IT/ML nous fixons le paramètre $N1 = 5$ pour les deux codes (meilleur compromis pour le décodage hybride IT/ML, voir section 5). Nous appelons cet ensemble de codes les LDPC-QC-staircase de type I.

7.3 Evaluation des capacités de correction

Afin d'estimer les capacités de correction des codes nous avons mesuré le ratio d'inefficacité des ensembles de codes à l'aide de simulations de type Monte Carlo. Nous avons

utilisé un ensemble de codes LDPC-QC staircase de type I ayant une matrice de parité de base de taille 5×10 et les coefficients ont été choisis aléatoirement dans l'intervalle $[0..50]$.

La figure 7.4 présente le ratio d'inefficacité du décodage itératif en fonction de la dimension du code. On observe que le code quasi-cyclique est meilleur que son homologue non structuré pour l'ensemble des tailles considérées. La différence atteint 2% pour les codes de dimension $k = 4000$.

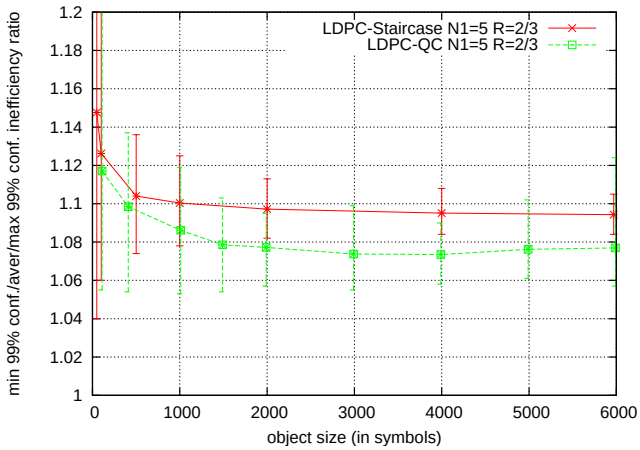


FIG. 7.4 – Ratio d'inefficacité en fonction de la dimension du code, en décodage IT ($R = 2/3$)

Le ratio d'inefficacité du décodage ML en fonction de la dimension du code est présenté sur la figure 7.5. On peut observer que les performances des deux ensembles de codes sont sensiblement égales, avec un léger avantage pour les codes non structurés.

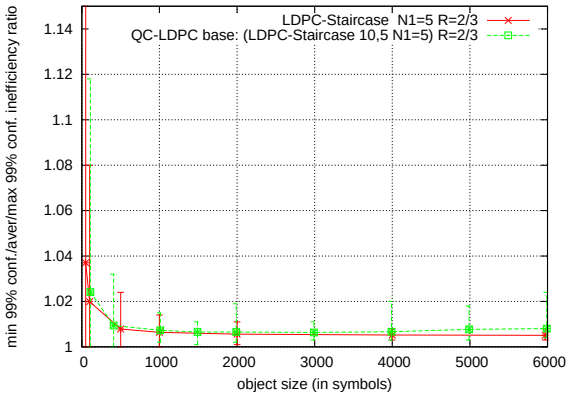


FIG. 7.5 – Ratio d'inefficacité en fonction de la dimension du code, en décodage ML ($R = 2/3$)

La structure quasi-cyclique a donc un impact globalement bénéfique sur le ratio d'inefficacité. L'amélioration des performances en décodage IT est à mettre sur le compte d'une longueur des cycles accrue par la structure quasi-cyclique. L'impact négatif limité sur les performances en décodage ML est le résultat de la structuration de la matrice qui réduit la probabilité d'obtenir une matrice de rang plein permettant le succès du décodage ML.

7.4 Evaluation de la complexité du décodage

Nous présentons maintenant des résultats concernant la réduction de la complexité induite par la structure de la matrice.

7.4.1 Vitesse de décodage

Nous avons effectué des simulations de type Monte Carlo sur un PC utilisant le système GNU/Linux avec processeur Intel Dual-Core Xeon 5120 cadencé à $1,86$ GHz et 4 Go de RAM. Le débit de décodage en fonction du taux de perte pour un code de dimension $k = 5990$ (la matrice de base est de taille 5×15 et le facteur d'expansion est 599) et un rendement $R = 2/3$ est présenté sur la figure 7.6 pour une taille de symbole $s = 1024$ octets et sur la figure 7.7 pour une taille de symbole $s = 10240$ octets. Nous avons choisi un code de dimension moyenne ($k = 5990$) car l'amélioration apportée par la structure quasi-cyclique est d'autant plus importante que le code est grand.

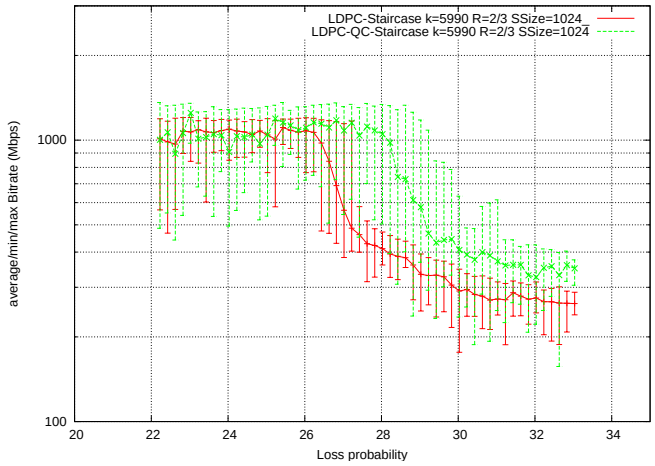


FIG. 7.6 – Débit de décodage en fonction du taux de perte, décodage hybride IT/ML et une taille de symbole $s = 1024$ octets ($R = 2/3$)

La complexité du décodage itératif ne semble pas être affectée par la structure de la matrice puisque les débits observés pour les taux de pertes faibles sont sensiblement égaux (1.1 Gb/s pour $s = 1024$). Le débit chute plus tôt avec le code LDPC-staircase qu'avec le code LDPC-QC-staircase. Cela vient du fait que le décodage IT est plus efficace avec le code quasi-cyclique (voir section précédente) et que le recours au décodage ML sera plus tardif. Enfin, lorsque le taux de perte est élevé et que le décodage ML est nécessaire, le débit obtenu avec le code quasi-cyclique (350 Mb/s pour $s = 1024$ et 460 Mb/s pour $s = 10240$) est significativement plus élevé que celui obtenu avec le code non structuré (260 Mb/s pour $s = 1024$, 230 Mb/s pour $s = 10240$). Ce gain est lié à la structure de la matrice de parité du code quasi-cyclique qui simplifie la résolution du système, ainsi que nous le montrerons dans la section suivante. Le gain en débit en ML est d'autant plus important que la taille des symboles est grande. En effet plus les symboles sont grands, plus les opérations sur ceux-ci sont coûteuses (le coût des accès mémoire augmente puisque les symboles ne tiennent plus dans le cache), et comme nous allons le voir dans la section

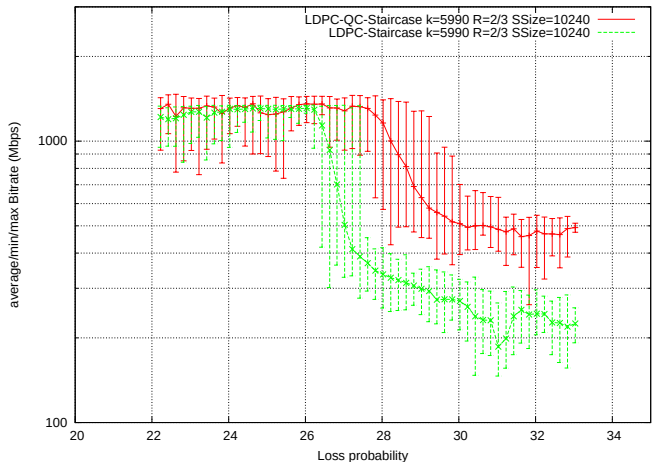


FIG. 7.7 – Débit de décodage en fonction du taux de perte, décodage hybride IT/ML et une taille de symbole $s = 10240$ octets ($R = 2/3$)

suivante le nombre d'opérations symboles effectuées par le décodage ML est inférieur dans le cas des codes QC.

7.4.2 Analyse détaillée de la complexité de la résolution du système

Nous allons maintenant nous intéresser plus en détail à la résolution du système linéaire afin d'expliquer le gain en complexité obtenu grâce à la structure du code.

Dans un premier temps nous avons observé le nombre total d'opérations XOR sur les symboles durant le décodage en fonction de la dimension du code (figure 7.8).

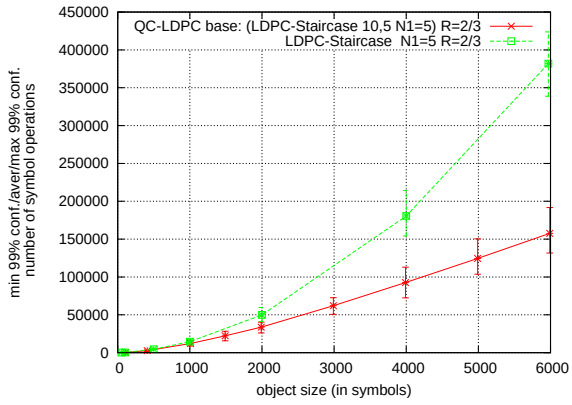


FIG. 7.8 – Nombre moyen d'opérations XOR sur les symboles en fonction de la dimension du code, lors d'un décodage ML ($R = 2/3$).

Le nombre d'opérations symboles augmente moins vite pour le code quasi-cyclique que pour le code non structuré. Pour un code de dimension $k = 6000$ le décodage ML du code LDPC-QC-staircase nécessite en moyenne 159.400 opérations symbole alors que celui du LDPC-staircase nécessite plus de 203.972 opérations symbole. La structure par bloc de la matrice a donc un impact bénéfique. Pour une matrice de base donnée, cet avantage est

d'autant plus grand que la dimension du code est grande , i.e. que le facteur d'expansion est élevé. Cet avantage peut être expliqué par une différence de "remplissage" de la matrice au cours de l'étape de triangularisation. Ce phénomène est observable sur la figure 7.9 où nous avons tracé le nombre moyen d'opérations sur les lignes par itération de l'algorithme de triangularisation ; ainsi que la densité de la matrice du système pour chaque itération. Une opération sur les lignes consiste à sommer deux lignes de la matrice du système, mais également à sommer les symboles associés aux coordonnées correspondantes dans le second membre du système.

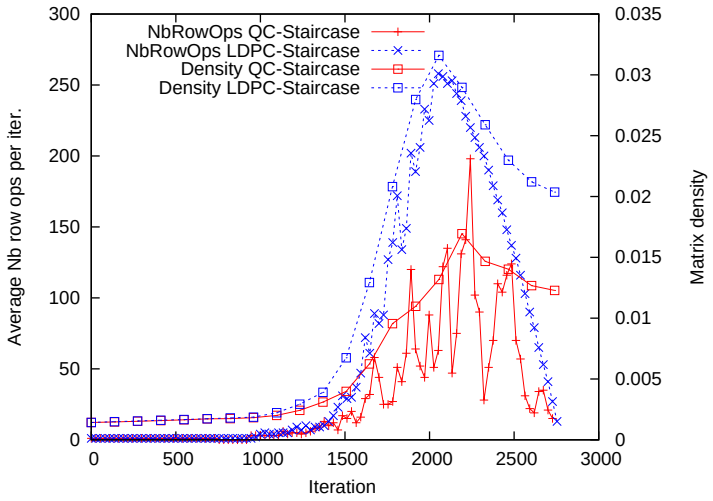


FIG. 7.9 – Nombre moyen d'opérations XOR sur les symboles par itération et densité de la matrice au cours de la triangularisation du système ($k = 5990$, $R = 2/3$).

Le nombre d'opération sur les lignes et la densité de la matrice sont deux grandeurs fortement liées. En effet à chaque itération d'élimination, il y aura d'autant plus d'opérations sur les lignes qu'il y aura d'entrées à éliminer, et donc que la matrice sera dense. De plus chaque opération sur les lignes va ajouter des entrées dans les colonnes d'indice supérieur à l'itération courante, augmentant ainsi le nombre d'entrées à éliminer au cours des itérations suivantes.

Si les deux codes possèdent des courbes d'allures similaires, l'amplitude est différente. Ceci va induire une réduction de la complexité pour deux raisons :

- le nombre d'opérations symboles par itération du code quasi-cyclique est significativement inférieur à celui du code non structuré (au maximum on atteint 258 opérations symboles par itération pour le LDPC-staircase contre 198 pour le code QC). Ceci va induire une réduction du nombre total d'opération symboles effectués lors la phase d'élimination en avant ;
- la densité de la matrice à la fin de l'étape d'élimination en avant est également inférieure dans le cas du code quasi-cyclique. Ainsi le nombre d'entrées non nulles dans la matrice triangularisée est plus faible, ce qui entraîne une réduction du nombre d'opérations symboles à effectuer lors de l'étape de substitution à rebours.

Le remplissage réduit de la matrice quasi-cyclique peut s'expliquer grâce à la structure par bloc. En effet la structure quasi-cyclique induit un retard sur le remplissage de la matrice, réduisant ainsi le nombre d'opérations sur les lignes.Montrons cette propriété : considérons une itération i d'élimination d'une colonne appartenant au premier bloc de

la matrice ($i < p$). La ligne que l'on choisi pour pivoter a tous ses éléments d'indice $< i$ nuls, alors que ceux d'indices $> i$ peuvent être non nuls. Lorsque l'on va sommer cette ligne à d'autres lignes, des entrées vont être éliminées dans la colonne i, mais d'autres entrées vont être ajoutées dans les colonnes suivantes. Dans le cas d'un code non structuré, ces entrées non nuls nouvellement créées peuvent être situées à n'importe quelle position $> i$. A l'inverse, pour le code quasi-cyclique ces entrées seront créées uniquement dans le bloc suivant et non pas dans le bloc courant. En effet dans chaque ligne d'une matrice quasi-cyclique de type I il y a au plus une entrée non nulle par bloc. La matrice va donc se remplir, mais l'effet de ce remplissage ne se fera sentir que lorsque l'on arrivera aux itérations d'élimination dans le bloc suivant.

7.5 Conclusions

Nous avons montré que l'utilisation d'une structure quasi-cyclique permet de construire des codes LDPC efficaces pour le canal à effacements. En effet de tels codes apportent une amélioration significative des capacités de correction en décodage itératif par rapport aux codes non structurés. De plus le décodage ML bénéficie d'une réduction de la complexité apportée par la structure bloc des matrices de parité. Nous avons observé que cette réduction de complexité provenait d'un remplissage plus lent de la matrice lors des étapes d'élimination de la résolution du système.

Les travaux présentés dans ce chapitre constituent une étude préliminaire attestant de l'intérêt de ces codes pour une utilisation sur le canal à effacements en utilisant un décodeur hybride IT/ML. Des travaux en cours conduisent à effectuer une étude plus précise des cycles de ces codes sous la forme d'un énumérateur. Il serait intéressant d'optimiser le choix des coefficients de la matrice de base de manière à maximiser la taille de ces cycles.

La structure quasi-cyclique devrait également permettre une optimisation de l'implémentation du décodeur. Premièrement, il n'est plus nécessaire de stocker la matrice de parité du code, puisque la matrice de base est suffisante pour calculer à la volée les positions des entrées non nulles, permettant ainsi un gain en mémoire. De plus il serait intéressant d'étudier la possible parallélisation des opérations de décodage tel qu'il est effectué sur d'autres canaux [47].

Dans ce travail nous nous sommes limités à l'étude d'un type particulier de codes LDPC QC, l'étude d'autres ensembles de codes LDPC-QC, tels que LDPC-QC de type II, étant en cours.

Chapitre 8

Codes LDPC généralisés à “faible rendement”

Contents

Ce chapitre présente un méthode de codage à faible rendement pour les mécanismes de correction d'effacements de niveau applicatif. Ce système est capable de s'adapter aux conditions du canal en adoptant dynamiquement différents rendements. En effet il est capable de changer à la volée le rendement, depuis le rendement maximum intrinsèque du code LDPC interne jusqu'au rendement minimum permis par la construction du code LDPC généralisé (ou GLDPC).
Plus précisément, le message est d'abord encodé en utilisant la méthode d'encodage des codes à structure triangulaire inférieure. Si des symboles de redondance supplémentaires sont nécessaires, l'encodeur passe en mode GLDPC de manière à produire ces “extra-symboles“ à la demande.
Des distributions irrégulières optimisées d'extra-symboles sont étudiées grâce à la méthode d'évolution de densité, afin d'améliorer les capacités de corrections. Nous montrons également que l'augmentation du nombre d'extra-symboles permet d'améliorer la probabilité de décodage, qui tend asymptotiquement vers 1.
Les travaux présentés dans ce chapitre ont été effectués en collaboration avec Valentin Savin du CEA-LETI et ont abouti à une publication [28].

8.1 Introduction

8.1.1 Motivations

Comme le souligne la section 2.6, certaines applications peuvent nécessiter un grand nombre de symboles de redondance. C'est le cas par exemple des applications de type carrousel [5], ou pour des transmissions sur des canaux dont la probabilité d'effacement ne peut pas être connue à l'avance. Il peut alors être intéressant de produire les symboles de redondance à la volée. Deux motivations peuvent être avancées :

- produire ces symboles a un coût (en termes de calcul) et limiter la production des symboles inutiles peut être une solution pour réduire ce coût ;
- un système qui nécessite que la totalité des symboles soit produits avant de pouvoir transmettre introduit une latence. Dans le cas d'un rendement faible, cette latence peut devenir prohibitive. La production des symboles de redondance à la volée permettra de réduire cette latence[1].

Afin de répondre aux besoins de codage à faible rendement capable de produire la redondance à la volée, nous proposons une alternative aux précédents schémas de codage. Ce nouveau code est basé sur un code LDPC-staircase [99] transformé en un code LDPC généralisé.

8.1.2 Codes à faible rendement

Les codes MDS tels que les codes Reed-Solomon [91] (voir section 2.4) permettent de coder avec un rendement arbitraire. En effet pour une dimension donnée, la longueur du code, et donc le rendement du code, n'est limitée que par la taille du corps fini utilisé. Il est donc possible de créer des codes de Reed-Solomon pour n'importe quel rendement désiré. Cependant cela peut nécessiter l'utilisation de grands corps finis, ce qui peut poser des problèmes de vitesse d'encodage/décodage (comme précisé en section 2.4 l'utilisation de corps finis avec plus de 2^{16} éléments n'est pas réaliste d'un point de vue pratique).

Les codes Raptor [103] et les codes LT [58] sont des codes permettant de produire à la volée un nombre potentiellement infini de symboles de redondance. De tels codes sont appelés codes sans rendement (voir section 2.6). Même s'ils sont décodables avec des algorithmes itératifs, le décodage ML à complexité potentiellement élevée, devient indispensable lorsqu'il s'agit de codes de longueurs courtes ou moyennes (voir section 2.6).

Les codes GLDPC présentés dans ce chapitre forment une classe de codes à faible rendement décodables avec un algorithme itératif garantissant une complexité linéaire. Un décodage ML de ces codes peut être envisagé et devrait permettre d'augmenter les capacités de correction.

8.1.3 Codes LDPC généralisés

Comme leur nom l'indique, ces codes constituent une généralisation du principe des codes LDPC [42]. Les codes LDPC généralisés ont été introduits par Tanner [113] et ont récemment fait l'objet d'études pour leur application à différents canaux [57][18][46][74][75].

[1] Rappelons que la transmission des symboles de redondance d'un code LDPC-staircase doit se faire dans un ordre aléatoire de manière à obtenir les meilleures capacités de correction.

Cette généralisation s'obtient en considérant que les contraintes représentées par les noeuds de contrôle ne représentent plus des contrôles de parité mais n'importe quel code linéaire, appelé sous-code ou encore code composant. En particulier, il devient possible de créer plusieurs symboles de redondance par noeud de contrôle. La figure 8.1 présente le graphe d'un code GLDPC généralisé. On remarque que le nombre de symboles de redondance est supérieur au nombre de noeuds de contrainte, alors que ces nombres sont égaux dans un code LDPC.

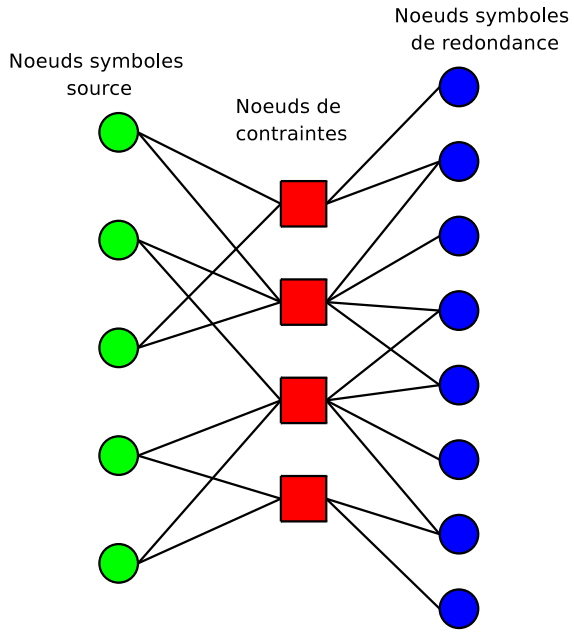


FIG. 8.1 – Exemple du graphe biparti d'un code LDPC généralisé

Tout comme les codes LDPC, les codes GLDPC peuvent être décodés avec des décodeurs itératifs dont la complexité dépend linéairement de la longueur du code ainsi que de la complexité de décodage du sous-code.

8.2 Schéma de codage proposé

Considérons H une matrice de parité binaire avec M lignes et $N = K + M$ colonnes. Cette matrice peut s'écrire $H = (H_1|H_2)$, où H_1 est une matrice de taille $M \times K$, et H_2 est une matrice de taille $M \times M$. Nous supposons dans ce travail que H_2 est une matrice escalier (double diagonale inférieur). Cependant l'approche peut être étendue en considérant n'importe quelle matrice triangulaire inférieure. Le graphe biparti associé à H est donc constitué de M *noeuds de contrainte* et N *noeuds symboles* correspondants respectivement aux lignes et aux colonnes de la matrice H.

Définition du code MDS interne : Considérons un code Reed-Solomon (RS) [91] systématique (il est à noter que n'importe quel autre code MDS défini sur un corps fini du type GF(2^p) serait utilisable). Chaque élément de ce corps fini correspond alors à un groupe de p bits. Sur ce corps, la somme de deux éléments revient à faire la somme de deux polynômes à coefficients sur $\mathbb{F}_2$, ce qui correspond en fait à une somme exclusive

(XOR) bit-à-bit. Considérons un tel $[k+1+E,k]$-code de Reed-Solomon. Ainsi, k est le nombre de symboles sources(la dimension du code), $k+1+E$ est le nombre de symboles codés (la longueur du code), et le nombre de symboles de redondance pouvant être produits par le code est égal à $1+E$.

Sans perte de généralité, on peut supposer que le premier symbole de parité est la somme XOR des k symboles sources. Plus précisément, puisque le code RS est systématique, quels que soient les symboles sources $(x_1,\ldots,x_k)$, les symboles de redondance sont calculés de la manière suivante :

$$(y_1,y_{1,1},\ldots,y_{1,E}) = (x_1,\ldots,x_k)G \tag{8.1}$$

où G est une matrice de taille $k\times(1+E)$ a coefficients dans GF(2^p). Le premier symbole de redondance y_1 est donc la somme exclusive des symboles sources $x_1,\ldots,x_k$ si et seulement si tous les coefficients de la première colonne de G sont égaux à 1. Ceci peut être obtenu en multipliant G à gauche par la matrice inversible appropriée, ce qui changera la manière dont l'encodage est effectué, mais pas les propriétés MDS du code.

Application au code LDPC-staircase externe : Pour la clarté de la présentation, supposons dans un premier temps que chaque ligne de la matrice H possède $k+1$ coefficients égaux à 1 (i.e. chaque noeud de contrainte est de degré $k+1$). Pour une séquence donnée de symboles sources $s_1,s_2,\ldots,s_K$, on peut calculer la séquence des symboles de redondance en "descendant les escaliers" de la manière suivante :

- Considérons les $k+1$ noeuds symboles correspondant à la première ligne de la matrice de parité. Les k premiers noeuds symboles correspondent à des symboles sources, tandis que le $(k+1)$-ème (i.e, le coefficient 1 sur la diagonale de H_2) est la somme XOR des k premiers symboles. Nous pouvons alors produire les $E-1$ symboles de redondance $p_1,p_{1,1},\ldots,p_{1,E}$, à partir des k premiers symboles. On note que p_1 est la valeur du $(k+1)$-ème symbole (ils sont tout les deux égaux à la somme XOR des k symboles source). Les symboles $p_{1,1},\ldots,p_{1,E}$ sont appelés extra-symboles de redondance, et ils ne seront pas utilisés dans la suite du processus d'encodage.
- Considérons les $k+1$ noeuds symboles correspondant à la deuxième ligne de la matrice de parité. Les k premiers noeuds symboles représentent soit des symboles source, soit le symbole de redondance de la ligne précédente p_1. Le $(k+1)$-ème symbole (i.e., le coefficient 1 sur la diagonale de H_2) est la somme XOR des k premiers symboles. On encode la séquence des k premiers symboles de la ligne en utilisant le code Reed-Solomon systématique. Soit $p_2,p_{2,1},\ldots,p_{2,E}$ les symboles de redondance produits. On note que p_2 est la valeur du $(k+1)$-ème symbole (ils sont tout les deux égaux à la somme XOR des k symboles source). Les symboles $p_{2,1},\ldots,p_{2,E}$ sont appelés extra-symboles de redondance, et ils ne seront pas utilisés dans la suite du processus d'encodage.
- On continue de la même manière le processus d'encodage jusqu'à la dernière ligne de la matrice de parité.

On remarque que la séquence $s_1,s_2,\ldots,s_K,p_1,\ldots,p_M$ est en fait un mot de code du code LDPC ayant pour matrice de parité H.

L'avantage de cette approche est qu'un ensemble d'extra-symboles de redondance est disponible, ce qui fait que ce code devient un code à faible rendement. Ce code est ainsi adapté pour toutes les situations qui peuvent bénéficier d'un code à faible rendement (ou

de manière extrême sans rendement). De plus les extra-symboles de redondance peuvent être produits à la demande, juste avant leur transmission, au lieu d'être produits en avance.

Généralisation : Il est aussi important de remarquer que l'hypothèse des noeuds de contrainte de degré constant égal à $k+1$ n'est pas indispensable. En effet, dans le cas où les noeuds de contrainte possèdent des degrés différents, on peut choisir k tel que le maximum des degrés des noeuds de contrainte est égale à $k+1$. Puis, pour tout noeud de contrainte de degré $k'+1$, avec $k' < k$, on peut utiliser un padding avec des zéros de manière à fournir une séquence de longueur k à l'encodeur systématique du code Reed-Solomon .

Le code LDPC ayant pour matrice de parité H sera appelé *code initial*, et son rendement $r = K/N$ sera appelé *rendement de base*. Supposons que $e(m)$, $m = 1,\ldots,M$, extra-symboles de redondance Reed-Solomon sont générés à partir du noeud de contrainte de la m-ème ligne, avec $0 \leq e(m) \leq E$. Ainsi la séquence :

$$\begin{aligned} s_1, s_2, \ldots, s_K, p_1, p_{1,1}, \ldots, p_{1,e(1)}, \cdots \\ \cdots, p_M, p_{M,1}, \ldots, p_{M,e(M)} \end{aligned} \tag{8.2}$$

est un mot du code GLDPC. Nous appellerons ce dernier le *code étendu*. On peut noter que les noeuds variable correspondant aux extra-symboles de redondance $p_{m,i}$ sont tous de degré 1 dans le graphe biparti associé au code étendu (voir figure 8.2).

Soit f_e la fraction de noeuds de contrainte avec e extra-symboles de redondance :

$$f_e = \frac{\text{card}\{m = 1,\ldots,M \mid e(m) = e\}}{M} \tag{8.3}$$

et $\bar{f}$ le nombre moyen d'extra-symboles de redondance par noeud de contrainte :

$$\bar{f} = \sum_{e=0}^{E} f_e \cdot e \tag{8.4}$$

Le rendement R du code étendu peut être calculé comme :

$$R = \frac{K}{N + M\bar{f}} = \frac{r}{1 + (1-r)\bar{f}} \tag{8.5}$$

Par exemple, si le rendement de base du code initial est $r = 1/2$ et que le nombre moyen d'extra-symboles de redondance par noeud de contrainte est $\bar{f} = 6$, on obtient un code étendu avec un rendement $R = 1/8$.

8.3 Conception du code

Cette section se focalise sur la conception des codes étendus. Le code initial considéré est un code LDPC-staircase décrit dans [99] (voir section 2.5.4). Ainsi, $H = (H_1|H_2)$, où chaque colonne de H_1 contient 3 coefficients égaux à 1 (meilleur choix pour le décodage itératif), et H_2 est une matrice escalier inférieure (double diagonale). On considère un rendement de base $r = 1/2$. Nous supposons également, que le degré des noeuds de contrainte est constant (ce qui est vrai pour le rendement considéré). Les noeuds de contrainte doivent alors être de degré 5. On en déduit que pour chaque ligne la dimension du code est $k = 4$.

Construire le code étendu nécessite de définir la distribution du nombre d'extra-symboles par noeud de contrainte. Nous considérons en particulier deux types de distribution d'extra-symboles :

- des distributions régulières (où "skyscraper") : $f_E = 1$, et $f_e = 0$ pour $e \in \{0, 1, \ldots, E-1\}$ (voir figure 8.2). Ainsi, chaque noeud de contrainte possède le même nombre E d'extra-symboles de redondance.
- distribution uniforme : $f_e = 1/(E+1)$, pour $e \in \{0, 1, \ldots, E\}$ (voir figure 8.3). Le nombre moyen d'extra-symboles par noeud de contrainte est $\bar{f} = E/2$.

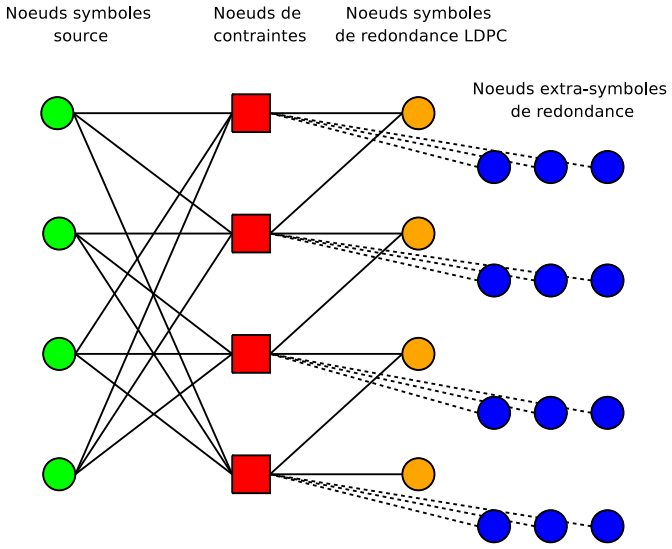


FIG. 8.2 – Graphe d'un code GLDPC avec distribution régulière d'extra-symboles

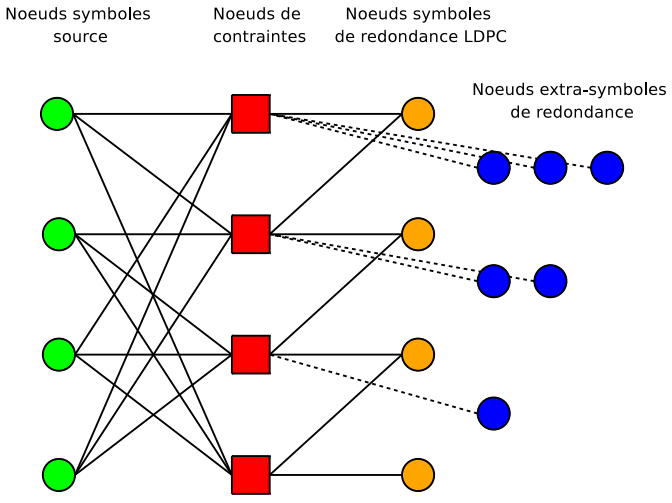


FIG. 8.3 – Graphe d'un code GLDPC avec distribution uniforme d'extra-symboles

Une distribution irrégulière est telle que le nombre d'extra-symboles varie d'un noeud de contrainte à l'autre. Une distribution uniforme est donc une distribution irrégulière. D'autre distributions irrégulières peuvent être considérées, mais nous montrerons que les performances de la distribution uniforme sont très proches de celle des distributions

irrégulières optimales. Une autre raison qui nous pousse à considérer ces deux types de distributions est qu'elles peuvent être "emboîtées" quand E varie de 0 à une valeur maximale. C'est à dire qu'il est possible d'augmenter le nombre d'extra-symboles tout en conservant le même type de distribution. Ceci permet de générer les extra-symboles de façon incrémentale en conservant les performances associées à la distribution.

Pour une distribution régulière avec E extra-symboles par noeud contrainte, l'ajout d'un extra-symbole produira également une distribution régulière, mais avec $E+1$ extra-symboles par noeud contrainte. Il en est de même pour une distribution uniforme : en ajoutant selon une distribution particulière au plus un extra-symbole à chaque noeud de contrainte, on obtiendra également une distribution uniforme. Voici une description de l'algorithme permettant d'obtenir une redondance incrémentale avec une distribution uniforme.

- Envoyer les symboles de redondance du code LDPC. Mettre $E_{courant} = 0$.
- Si davantage d'extra-symboles sont nécessaires :
  - choisir aléatoirement $M/2$ lignes de H ;
  - générer puis envoyer un extra-symbole de redondance pour chacune de ces lignes.
- Mettre $E_{courant} = 1$.
- Tant que davantage d'extra-symboles sont nécessaires, pour chaque $e \in \{0, 1, \ldots, E_{courant}\}$:
  - choisir aléatoirement $M/(E_{courant}+1) - M/(E_{courant}+2)$ lignes parmi les $M/(E_{courant}+1)$ lignes ayant e extra-symboles de redondance ;
  - pour chacune de ces lignes, envoyer $E_{courant}+1-e$ extra-symboles de redondance supplémentaires (ainsi chacune de ces ligne possède maintenant $E_{courant}+1$ extra-symboles de redondance). Mettre $E_{courant} = E_{courant}+1$.

Nous allons montrer dans la prochaine section que ces codes étendus s'approchent de très près de la capacité du canal si le nombre d'extra-symboles de redondance est suffisamment grand.

8.4 Optimisation de la distribution irrégulière du nombre d'extra-symboles par évolution de densité

Dans cette section nous utilisons l'approche évolution de densité [94] (voir section 2.5.5) afin d'optimiser la distribution d'extra-symboles f pour une matrice de base donnée et un rendement étendu cible R.

Le décodage itératif des codes étendus est similaire à celui du code de base LDPC. En effet le décodage revient à rechercher les noeuds de contrainte pour lesquels le décodage du sous-code est possible. La différence est que dans le cas des codes GLDPC chaque sous code est capable de reconstruire $e(m)+1$ symboles effacés contre 1 dans les codes LDPC. Une fois ces symboles reconstruits, on recommence à chercher des noeuds de contrainte décodables. Il est important de noter que les extra-symboles de redondance ne fournissent de l'information qu'à un seul noeud de contrainte, celui auquel il participe, et qu'ils ne peuvent donc pas relayer d'information durant le décodage itératif. De plus, une fois que tous les symboles source d'un noeud contrainte ont été récupérés, les extra-symboles associés n'ont plus aucune utilité.

A cause de l'inutilité des extra-symboles reconstruits, dans cette analyse le degré d'un noeud de contrainte doit toujours être considéré en fonction du code LDPC de base (i.e. on omet les extra-symboles de redondance). Le degré maximum des noeuds de contrainte est

noté d_c et le degré maximum des noeuds variable (de base) est noté d_s. Dans notre cas, ces degrés sont ceux associés aux LDPC-staircase : $d_s = 3$, $d_c = d_s \frac{K}{M}$. Comme d'habitude, λ et ρ désignent les polynômes de distribution des degrés respectivement des noeuds variables et des noeuds de contrainte du code LDPC de base :

- λ_d est la fraction d'arêtes connectées à un noeud variable LDPC de degré d, et $\lambda(X) = \sum_{d=1}^{d_s} \lambda_d X^{d-1}$
- ρ_d est la fraction d'arêtes connectées à un noeud de contrainte LDPC de degré d, et $\rho(X) = \sum_{d=1}^{d_c} \rho_d X^{d-1}$

On note aussi :

- P_l, la probabilité qu'un noeud variable LDPC envoie un effacement à l'itération l
- Q_l, la probabilité qu'un noeud de contrainte LDPC envoie un effacement à l'itération l

Ainsi P_0 est la probabilité d'effacement du canal, puisqu'on suppose que chaque symbole est soit reçu, soit complètement effacé.

Considérons un noeud de contrainte m de degré $k+1$, et soit $e(m) = e$ le nombre d'extra-symboles de redondance de m. Soient $n, n_1, \ldots, n_k, t_1, \ldots t_e$ les symboles participant à m. Les $k+1$ premiers sont des symboles LDPC (source et redondance), alors que les e derniers représentent les extra-symboles. Le noeud contrainte m peut reconstruire la valeur du symbole LDPC n si et seulement si le nombre d'effacements dans la séquence $n_1, \ldots, n_k, t_1, \ldots t_e$ est inférieur ou égal à e. A l'itération l, le symbole LDPC est effacé avec une probabilité P_l, alors que les extra-symboles de redondance sont toujours effacés avec la probabilité P_0, la probabilité d'effacement du canal. Il en suit que la probabilité que le noeud de contrainte m reconstruise la valeur du symbole LDPC n à l'itération $l+1$, notée $\bar{Q}_{\ell+1}(k,e)$, peut être obtenue comme :

$$\bar{Q}_{\ell+1}(k,e) = \sum_{\substack{0 \le i \le k, 0 \le j \le e \\ i+j \le e}} \mathrm{C}_k^i P_\ell^i (1-P_\ell)^{k-i} \mathrm{C}_e^j P_0^j (1-P_0)^{e-j} \tag{8.6}$$

Puisque la probabilité que le noeud contrainte m envoie un effacement au symbole LDPC n à l'itération $l+1$ est $(1-\bar{Q}_{\ell+1}(k,e))$, en faisant la moyenne sur toutes les valeurs possible de k et e, on obtient :

$$Q_{\ell+1} = 1 - \sum_{k=0}^{d_c-1} \rho_{k+1} \sum_{e=0}^{E} f_e \bar{Q}_{\ell+1}(k,e) \tag{8.7}$$

Inversement, un noeud variable LDPC n de degré d participant aux noeuds contraintes $m, m_1, \ldots, m_{d-1}$, envoie un effacement au noeud de contrainte m ssi il a été effacé par le canal, et qu'il a reçu des effacements de tous les noeuds de contraintes $m_1, \ldots, m_{d-1}$. Puisque cela arrive avec la probabilité $P_0 \cdot Q_{\ell+1}^{d-1}$, et en faisant la moyenne sur toutes les valeurs possibles des degrés d, on obtient :

$$P_{\ell+1} = P_0 \sum_{d=1}^{d_s} \lambda_d Q_{\ell+1}^{d-1} = P_0 \lambda(Q_{\ell+1}) \tag{8.8}$$

Ainsi, on peut suivre la probabilité d'effacement P_ℓ à chaque itération $\ell \ge 0$ en utilisant les équations (8.6), (8.7), (8.8). Nous pouvons reconstruire tous les symboles effacés dans

le cas où la fraction de symboles effacés est P_0 ssi $\lim_{\ell \to +\infty} P_\ell = 0$. La probabilité seuil P_{th} est définie par :

$$P_{\text{th}} = \max\{P_0 \mid \lim_{\ell \to +\infty} P_\ell = 0\} \tag{8.9}$$

et la distance à la capacité est :

$$\Delta = 1 - R - P_{\text{th}}. \tag{8.10}$$

où R est le rendement du code étendu. L'inefficacité du décodage est définie comme le rapport entre le nombre de symboles reçus et le nombre de symboles source. L'inefficacité de seuil peut être calculée par :

$$\mu_{\text{th}} = \frac{1 - P_{\text{th}}}{R} = \frac{1 - P_{\text{th}}}{r}(1 + (1 - r)\bar{f}) \tag{8.11}$$

où r est le rendement de base.

Les performances théoriques des distributions régulières et uniformes sont présentées en termes d'inefficacité seuil et de distance à la capacité sur les figures 8.4 et 8.5. L'axe horizontal inférieur montre le nombre moyen d'extra-symboles de redondance par ligne ($\bar{f}$), tandis que l'axe horizontal supérieur montre le rendement du code étendu correspondant (R). L'inefficacité du code initial (sans extra-symboles de redondance) est $\mu = 1.1023$. En augmentant le nombre d'extra-symboles de redondance, le rendement étendu décroît jusqu'à 0.059, tandis que l'inefficacité augmente jusqu'à 1.1604 pour les distributions régulières et jusqu'à 1.1205 pour les distributions uniformes. Ceci correspond à une augmentation de "seulement" 2% par rapport à l'inefficacité du code initial. Il est important de noter que l'inefficacité est biaisée par la faiblesse du rendement. En fait, sur la figure 8.5, on peut voir que la distance à la capacité, i.e. la distance entre les codes étendus et un code idéal (caractérisé par $\Delta = 0$), décroît jusqu'à 0.0071 quand le nombre moyen d'extra-symboles de redondance devient égal à 15.

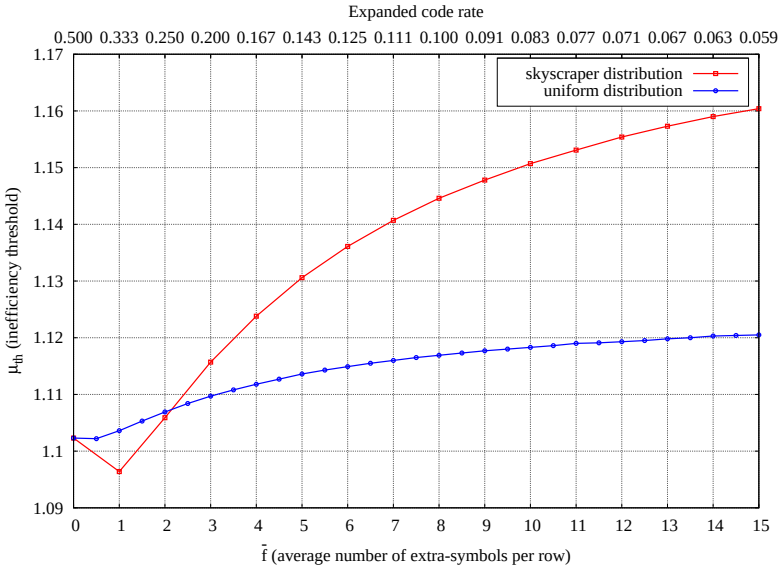


FIG. 8.4 – Seuil d'inefficacité en fonction du nombre moyen d'extra-symboles de redondance par ligne.

On note qu'il est possible d'optimiser la distribution f des extra-symboles, en optimisant la fonction qui associe à f la distance à la capacité correspondante en utilisant

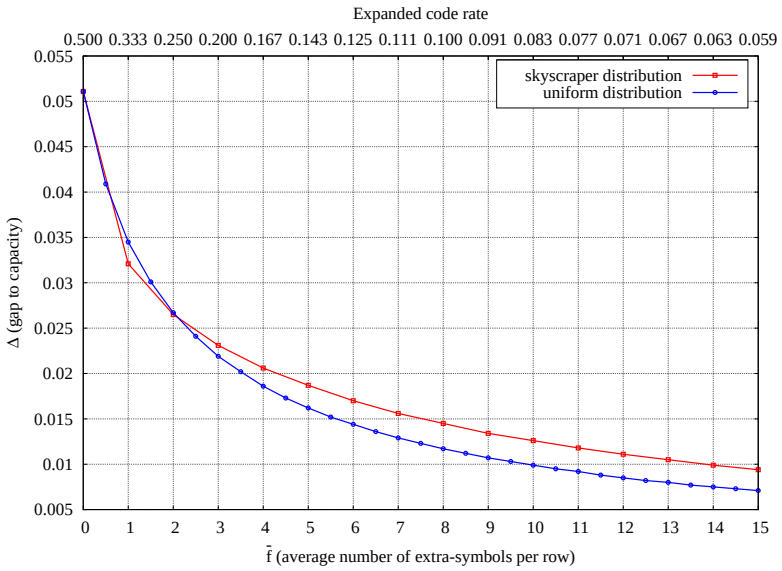


FIG. 8.5 – Distance à la capacité en fonction du nombre moyen d'extra-symboles de redondance par ligne.

f_0	f_1	f_2	f_3	f_4	f_5	f_6	$\bar{f}$	Δ
0.2382	0.0511	0.0047	0.2446	0.1614	0.2405	0.0595	3	0.0213

TAB. 8.1 – Exemple de distribution irrégulière optimisée ($\bar{f} = 3$)

l'algorithme d'évolution différentielle [110]. Nous avons effectué cette optimisation pour différentes valeurs du nombre moyen d'extra-symboles $\bar{f}$. Le tableau 8.1 donne un exemple d'une distribution irrégulière optimisée, avec $\bar{f} = 3$, obtenue après 100 itérations de l'algorithme d'évolution différentielle. Pour chaque $\bar{f}$, l'amélioration des performances apportée par la distribution optimisée par rapport à la distribution uniforme n'était pas significative. Par exemple, pour $\bar{f} = 3$ la distance à la capacité de la distribution optimisée était de 0.0213 (contre 0.0219 pour la distribution uniforme), et pour $\bar{f} = 10$ la distance à la capacité de la distribution optimisée était de 0.0092 (contre 0.0219 pour la distribution uniforme). Nous pensons donc qu'une distribution uniforme constitue le meilleur choix (voir section 8.3).

8.5 Résultats des simulations

Nous avons effectué des simulations de type Monte-Carlo pour les distributions régulières et uniformes pour différentes valeurs de K. Les performances en termes d'inefficacité de décodage moyenne sont présentées sur la figure 8.6 pour la distribution régulière et sur la figure 8.7 pour la distribution uniforme. La figure 8.8 montre la distance à la capacité de la distribution uniforme pour différentes valeurs de K. Sur les figures 8.6 et 8.7 on observe que pour un rendement étendu donné, l'inefficacité décroît à mesure que la dimension du code augmente. De plus pour une dimension fixée, l'inefficacité augmente à mesure que le rendement étendu augmente. Les performances obtenues avec un code de dimension 50000 sont assez éloignées des performances d'un code de longueur infini représenté par

la courbe “DE threshold”. Sur la figure 8.8, on constate que la distance à la capacité des codes GLDPC diminue à mesure que le rendement étendu augmente. Ces tests confirment les bonnes performances des codes étendus GLDPC utilisant des distribution uniformes pour la génération des extra-symboles de redondance.

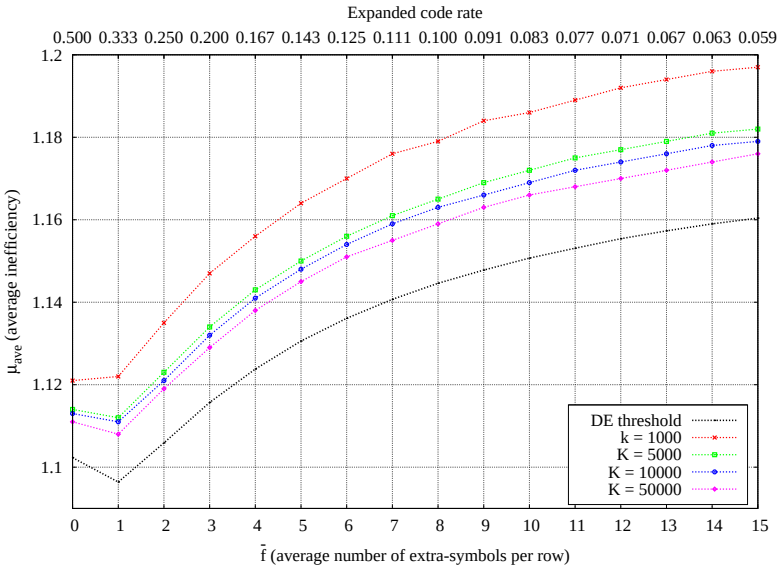


FIG. 8.6 – Inefficacité moyenne en fonction du nombre moyen d’extra-symboles par ligne dans le cas d’une distribution régulière

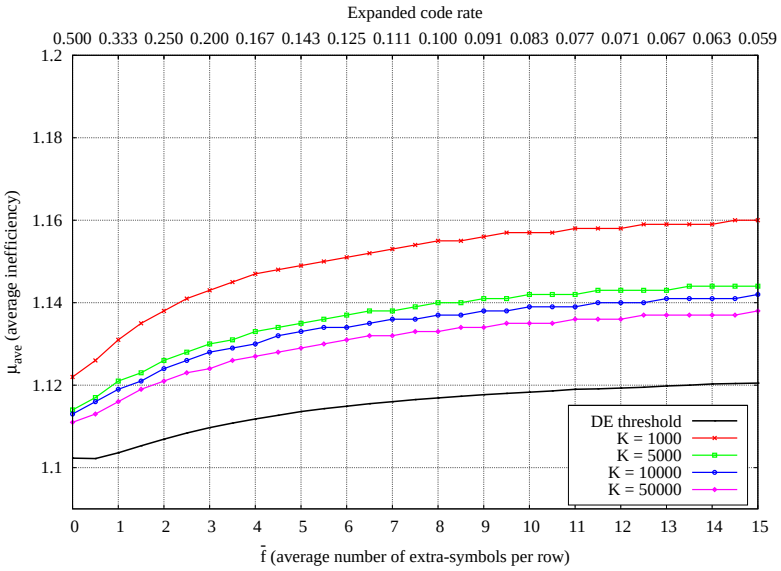


FIG. 8.7 – Inefficacité moyenne en fonction du nombre moyen d’extra-symboles par ligne dans le cas d’une distribution uniforme.

8.6 Conclusions

Dans ce chapitre nous avons proposé un schéma de codage qui est capable de produire de la redondance de manière incrémentale afin de s’adapter aux mauvaises conditions

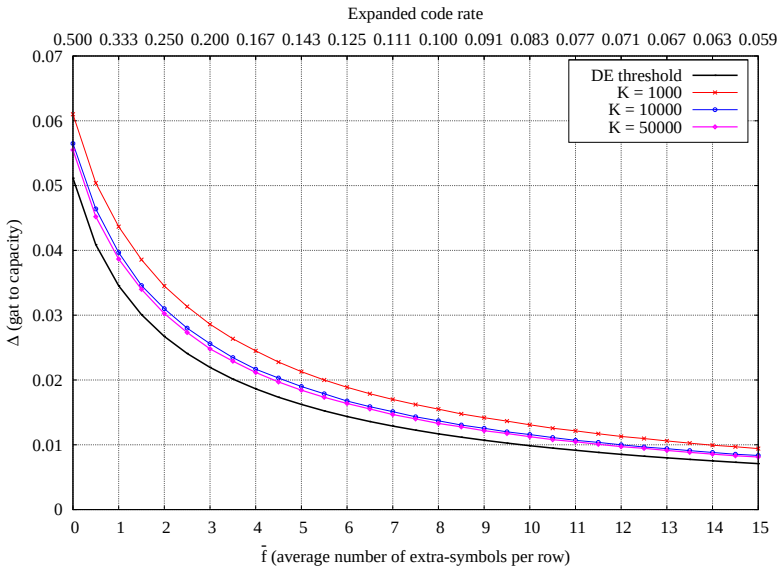


FIG. 8.8 – Seuil de probabilité d’effacement en fonction du nombre moyen d’extra-symboles par ligne dans le cas d’une distribution uniforme.

du canal ou pour effectuer une distribution de contenu de type fontaine ou carrousel (e.g. FLUTE/ALC [81]). Le schéma de codage se base sur les codes LDPC-staircase, et des extra-symboles de redondance sont produits à la demande en étendant le code initial en un code LDPC généralisé.

Les performances de plusieurs distributions d’extra-symboles ont été considérées . Les tests ont montré qu’une distribution uniforme offre des performances très proches des distributions optimisées, tout en permettant d’utiliser facilement la fonctionnalité de “redondance incrémentale”. Nous avons également montré qu’en augmentant le nombre d’extra-symboles de redondance, le code étendu s’approche des performances d’un code parfait de type MDS.

Notre proposition peut donc être utilisée pour construire des codes applicatifs grand bloc à faible rendement performants, et se décodant avec un algorithme itératif. De plus il est vraisemblable que les performances de ces codes pourront encore être améliorées grâce à l’usage d’un décodeur hybride IT/ML (voir chapitre 4). Ceci constitue une piste intéressante pour les travaux futurs.

Deuxième partie

Codes à effacements et fonctionnalités cryptographiques

Chapitre 9

VeriFEC : vérification d'intégrité conjointe au décodage

Contents

Nous pensons que tôt ou tard, la vérification d'intégrité du contenu ainsi que l'authentification de la source seront nécessaires dans les systèmes utilisant les codes correcteurs d'effacements. De manière à sauvegarder les ressources du terminal client, qui peut être un appareil mobile et autonome, nous avons conçu un système hybride qui réunit le décodage AL-FEC avec la vérification d'intégrité et l'authentification de la source. Plus précisément, notre système est capable de détecter une corruption aléatoire de l'objet avec une probabilité proche de 1 *avec un surcoût quasiment nul en termes de calcul. Le cas des attaques intelligentes est également étudié, et des contre-mesures efficaces sont proposées. Finalement nous proposons une généralisation du système à d'autres type de code à effacements. Ce travail a fait l'objet d'une publication [29]. Ces travaux ont grandement bénéficié des discussions que nous avons eu avec Valentin Savin du CEA-LETI, Philippe Gaborit de l'équipe PI2C de l'université de Limoges, Nicolas Sendrier et Jean-Pierre Tillich de l'équipe SECRET de l'INRIA.*

9.1 Introduction

Nous nous intéressons dans ce chapitre à deux fonctionnalités cruciales dans les réseaux de communications modernes : la correction d'erreurs et la garantie d'intégrité des données. Ces deux fonctionnalités sont liées au problème de l'altération des données transmises. Ces altérations peuvent être soit accidentelles (perturbation du signal sur le médium physique) soit intentionnelles (corruption des données par un attaquant). Au niveau de la couche applicative, ces fonctionnalités reposent habituellement sur l'utilisation de codes AL-FEC pour la correction d'erreurs et de fonctions de hachage cryptographiques pour la vérification d'intégrité. Ces deux types d'outils étant souvent amenés à travailler sur les mêmes ensembles de données, le développement d'un système remplissant conjointement ces deux fonctionnalités dans le but de réduire la consommation de ressources de calcul nous est apparu comme une voie intéressante à explorer.

Ajout d'un service de vérification d'intégrité/authentification : A l'intérieur des réseaux fermés (par exemple les infrastructures "Digital Video Broadcasting - Handheld", en francais Diffusion Vidéo Numérique - Portable (DVB-H)), lancer une attaque de type déni de service (DoS pour "Denial of Service" en anglais) ou injecter un trafic malicieux nécessitent un équipement onéreux, ce qui limite les risques. Mais la situation est totalement différente dans les réseaux ouverts, tels que Internet ou des points d'accès Wifi. Dans ces situations la vérification d'intégrité et l'authentification de la source sont nécessaires afin de permettre au récepteur de vérifier que ce qu'il a reçu est bien ce que l'émetteur légitime a envoyé.

Ces vérifications d'intégrité/authentification de la source peuvent être effectuées suivant une approche par paquet (le récepteur vérifie individuellement chaque paquet) ou bien par objet (le récepteur vérifie l'objet une fois ce dernier totalement reçu). Ce travaille se concentre sur le dernier cas. La solution traditionnelle consiste à signer le hash de l'objet avec une fonction cryptographique asymétrique. Dans le cas de la transmission de gros objets, le temps de calcul associé à la signature est faible comparé au temps de calcul du hash de l'objet. Cette solution sera considéré comme la solution de *référence* à laquelle nous comparerons notre solution.

Objectifs de ce travail : Notre travail explore une solution alternative en ajoutant une fonctionnalité de vérification d'intégrité à un schéma de codage existant, de manière à minimiser le surcoût en calcul et en transmission. Le système résultant, appelé VeriFEC, devra être capable de :

- détecter une large majorité des objets corrompus avec un coût réduit en calcul (i.e. autoriser une vérification préliminaire discriminante à bas coût),
- détecter tous les objets corrompus avec un coût proche de celui d'une vérification d'intégrité standard (i.e. vérification complète),
- conserver la capacité de correction du schéma de codage AL-FEC original.

La corruption peut être soit accidentelle (i.e. causée par des erreurs de transmissions qui n'ont pas été détectées par les systèmes de correction et de détection d'erreurs des couches inférieures [109]) soit intentionnelle (i.e. montée par un attaquant intelligent). Dans ce travail nous considérerons dans un premier temps, le cas de corruptions aléatoires, puis nous étudierons le cas d'attaques intelligentes.

9.2 Analyse du problème et observations

Cette section introduit le modèle d'attaque ainsi que le phénomène propagation de corruption qui constitue le coeur de notre proposition. Nous discuterons ensuite de l'usage potentiel de ce phénomène, du point de vue de l'attaquant comme du point de vu du récepteur.

9.2.1 Modèle d'attaque

Considérons un canal de transmission non sécurisé. Nous supposons dans un premier temps que l'attaquant est capable de corrompre un nombre illimité de symboles choisis au hasard (ce qui prend en compte le cas d'erreurs non détectées par les couches inférieures et les attaquants avec des capacités limitées). Dans un second temps (section 9.5) nous considérerons le cas des attaques intelligentes montées par un attaquant possédant des ressources illimitées.

Un premier objectif pour l'attaquant peut être de *corrompre l'objet sans que le récepteur ne le remarque*. Cette corruption sera de toutes façons détectée par l'utilisation d'une fonction de hachage cryptographique sur l'ensemble de l'objet, et la probabilité de détection ne sera limitée que par la robustesse de la fonction de hachage contre des attaques malicieuses. Un autre objectif pour l'attaquant peut être de monter une *attaque de type déni de service (DoS)*, soit en envoyant un grand nombre d'objets factices qui seront reçus en plus de l'objet légitime, ou simplement en corrompant le plus grand nombre d'objets possible. Cette attaque est triviale à mettre en place. Le défi pour le récepteur sera d'*identifier rapidement les objets corrompus et de s'en débarrasser avec le moins de calcul possible*.

Ce travail se concentre essentiellement sur le deuxième type d'attaque, où l'attaquant essaye d'épuiser les ressources du récepteur.

9.2.2 Phénomène de propagation des corruptions

Le phénomène : De manière à corriger les effacements, le décodeur itératif reconstruit les symboles manquants en utilisant les symboles reçus. Considérons l'équation suivante

(l'une des équations de contrainte définies par la matrice de parité du code LDPC) :

$$S_0 \oplus S_1 \oplus S_2 \oplus S_3 = 0$$

Supposons que les valeurs s_1, s_2, s_3 des symboles correspondants ont été reçues, mais pas s_0 qui demeure inconnue. Ainsi s_0, la valeur du symbole S_0, est reconstruite de la manière suivante :

$$s_0 = s_1 \oplus s_2 \oplus s_3$$

Si de plus le symbole S_3 a eu sa valeur corrompue en $s'_3 = s_3 \oplus \varepsilon$ et que les autres symboles ont été reçus correctement ; alors S_0 est décodé comme :

$$s_1 \oplus s_2 \oplus s'_3 = s_1 \oplus s_2 \oplus s_3 \oplus \varepsilon = s_0 \oplus \varepsilon = s'_0$$

S_0 a donc hérité de la corruption de S_3. Ainsi, si un symbole corrompu est utilisé lors du décodage, le symbole reconstruit héritera de la corruption. De plus, chaque symbole nouvellement reconstruit peut être utilisé récursivement pour reconstruire d'autre symboles, et une succession de corruptions peut avoir lieu. Nous appelons ceci le *phénomène de propagation de corruptions*.

Interprétation en termes de mots de code :

Ce phénomène peut être étudié d'un point de vue "mot de code". Rappelons que dans le contexte des codes AL-FEC travaillant sur $\mathbb{F}_2^t$, un mot de code est un vecteur formé des bits à une certaine position dans l'ensemble des symboles sources et parité. Par exemple l'ensemble des bits en i-ème position dans les symboles, forme aussi un mot de code binaire. Ainsi le code LDPC travaillant avec des symboles composées de t bits peut être vu comme la mise en parallèle de t codes LDPC travaillant sur des symboles composés d'un seul bit. Dit autrement, le décodage d'un code LDPC sur $\mathbb{F}_2^t$ équivaut au décodage parallèle de t codes LDPC binaires partageant tous le même schéma d'effacements.

On remarque que les vecteurs obtenus à la sortie d'un décodage réussi d'un code à effacements sont toujours des mots du code. Pour un code LDPC de matrice de parité H, cela veut dire qu'un vecteur w décodé est tel que $Hw = 0$.

Supposons que le mot de code w transmis a été corrompu. La sortie du décodeur est nécessairement un autre mot du code w' ($w' \neq w$). Puisque nous considérons un code linéaire, la différence des deux mots de code (et en particulier $e = w - w'$) est aussi un mot de code. Ainsi une corruption réussie peut être vue comme l'addition d'un mot de code e, appelé le *mot de code de corruption*, au mot de code transmis w.

Observations expérimentales :

Afin de quantifier l'importance de ce phénomène, nous avons effectué des expériences avec le codec de référence LDPC version 2.0 en C++ [118]. Nous avons utilisé un objet composé de 20.000 symboles[1], ainsi qu'un rendement $R = k/n = 2/3$ (i.e. $n - k = 10.000$ symboles de parité sont créés). Les symboles sont transmis dans un ordre aléatoire de manière (1) à effectuer les expériences sans se préoccuper du modèle de perte, et (2) à être certain qu'un décodage aura lieu[2]. L'attaquant choisit aléatoirement plusieurs symboles et

[1] La taille des symboles n'a pas d'importance et n'est donc pas spécifiée.

[2] Si l'ordre de transmission n'est pas assez aléatoire, un récepteur pourra facilement mélanger l'ordre dans lequel les symboles sont donnés au décodeur sans nécessiter l'utilisation d'une file d'attente supplémentaire.

y ajoute une corruption. Nous comptons ensuite le nombre de symboles *source* corrompus à la sortie du décodage[3]. Le test est répété 2.000 fois pour chaque valeur du nombre de symboles corrompus par l'attaquant, et nous traçons les min/moyenne/max/intervalles de confiance à 90 %.

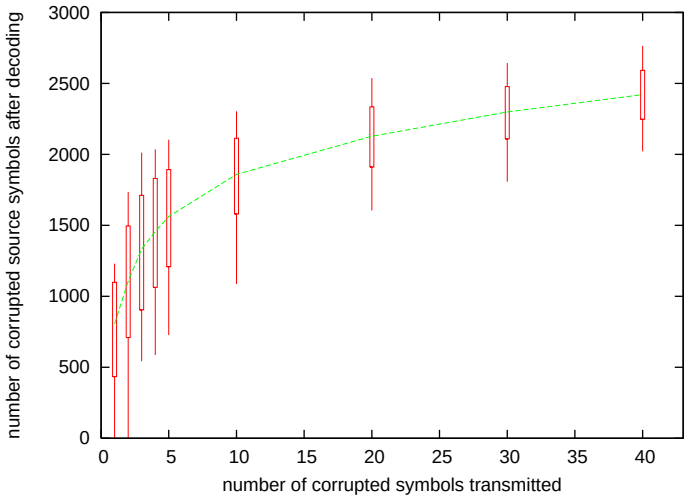


FIG. 9.1 – Nombre de corruptions à la sortie du décodeur (min/moyenne/max/intervalles de confiance à 90%) en fonction du nombre de symboles corrompus durant la transmission.

La figure 9.1 montre que même une corruption d'un unique symbole peut provoquer la corruption de plus de 700 symboles après décodage (i.e. $3,5\%$ de l'objet). Cependant certaines expériences montrent très peu de symboles corrompus, ce qui veut dire que les symboles corrompus n'ont été utilisés que pour reconstruire un petit nombre de symboles. Dans certain tests, il n'y a aucun symbole corrompu après le décodage. Cela veut dire que les symboles corrompus par l'attaquant n'ont pas été utilisés lors du décodage (par exemple parce que le symbole a déjà été reconstruit lorsqu'il est reçu).

9.2.3 Premières conclusions

D'après ces premières observations, plusieurs conclusions peuvent être tirées.

Pour l'attaquant :

Pour l'attaquant, une corruption massive de l'objet peut être obtenue avec un nombre limité d'attaques sur les symboles transmis (un unique symbole est souvent suffisant). Cependant, l'attaquant aura des difficultés à créer une corruption limitée et ciblée de l'objet. Ceci n'est pas totalement impossible, mais cela reste exceptionnel. Ce dernier aspect sera traité en section 9.5.

Pour le récepteur :

Le récepteur peut voir l'importance du phénomène de propagation de corruption soit comme un problème (e.g., un fichier vidéo corrompu présentera de nombreux défauts)

[3]Nous ne prenons pas en compte le nombre de symboles de parité corrompus après décodage, puisque l'objectif de l'attaquant est de corrompre l'objet et non pas les symboles de parité temporaires.

ou comme un avantage (la détection de corruptions sera plus facile). Dans notre cas, l'efficacité de VeriFEC repose largement sur ce phénomène.

9.3 Notre solution : VeriFEC

Cette section introduit le système hybride VeriFEC qui réunit un code AL-FEC et un système de vérification d'intégrité/authentification en une seul entité.

9.3.1 Principes

L'idée consiste à tirer avantage du phénomène de propagation de corruptions en utilisant une vérification d'intégrité de l'objet en deux étapes. La vérification d'intégrité se fera à l'aide d'une fonction de hachage. Dans un premier temps, une vérification préliminaire à faible complexité pourra détecter une vaste majorité des objets corrompus. Puis, si la vérification préliminaire n'a rien détecté d'anormal, une vérification complémentaire sera effectuée afin d'obtenir une probabilité de détection égale à 1[4].

La vérification d'intégrité préliminaire consiste à vérifier uniquement un sous-ensemble des symboles sources à la sortie du décodeur (figure 9.2). Grâce au phénomène de propagation de corruption, nous savons que la plupart des attaques aléatoires (même celles n'impliquant qu'un unique symbole, le pire cas) déclencheront un grand nombre de corruptions dans l'objet décodé. Puisque seul un sous-ensemble de l'objet est vérifié, la vérification préliminaire ne pourra jamais atteindre une probabilité de détection de 1, mais nous allons montrer dans la section 9.4 qu'en pratique une large majorité des attaques sont détectées. La deuxième vérification consiste à vérifier les symboles sources restants. Ainsi, un objet passant avec succès les deux vérifications est certifié comme étant intègre.

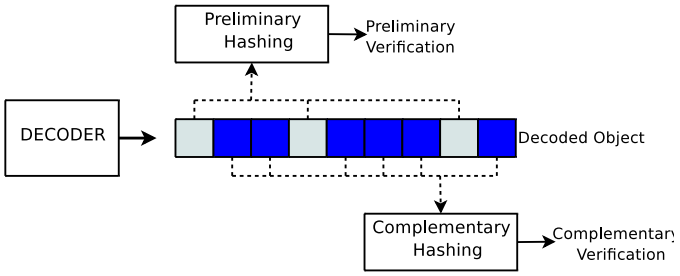


FIG. 9.2 – Vérification d'intégrité préliminaire et complémentaire de VeriFEC.

Dans ce travail nous nous limitons au cas d'un récepteur unique. Cependant VeriFEC n'inclut pas de mécanisme qui pourrait limiter son champ d'application. Puisque VeriFEC n'utilise pas de canal retour (information envoyée du récepteur à l'émetteur), il peut s'appliquer à des systèmes de diffusion à destination d'un très grand nombre de récepteurs.

9.3.2 Détails

Côté émetteur :

L'émetteur effectue un encodage FEC et envoie l'ensemble des symboles sources et parité comme d'habitude. En parallèle il sélectionne un sous-ensemble V de N_{verif}

[4]Cette probabilité n'est limitée que par la fonction de hachage utilisée pour la vérification d'intégrité.

symboles sources à l'aide d'un PRNG initialisé à l'aide d'une graine $graine\_verif\_prel$. Soit $\overline{V}$ le sous-ensemble complémentaire, contenant tous les symboles sources qui ne sont pas dans V. L'émetteur calcule alors de hash de l'ensemble V, appelé $hash\_prel$, et le hash de $\overline{V}$ appelé $hash\_compl$. Le triplet $\{graine\_verif\_prel; hash\_prel; hash\_compl\}$ est ensuite envoyé au récepteur. Puisque la sécurité de ce triplet est cruciale, l'émetteur le signe avec un crypto-système à clef publique [95], de manière à ce que le récepteur puisse vérifier l'intégrité du triplet et en authentifier la source (on suppose que le récepteur connaît la clef publique de l'émetteur, par exemple grâce à un système de type PKI). Le triplet signé peut alors être envoyé en ligne (en utilisant le canal non sécurisé) où hors-ligne (via une page web par exemple). Un canal sécurisé point à point peut parfois exister entre l'émetteur et le récepteur, sur lequel le triplet peut être envoyé. Cela ne doit pas être considéré comme la solution habituelle puisqu'elle ne passe pas à l'échelle.

Côté récepteur :

Le récepteur effectue un décodage FEC standard de l'objet en utilisant les symboles reçus. En parallèle le récepteur récupère le triplet $\{graine\_verif\_prel; hash\_prel; hash\_compl\}$ et vérifie sa signature. Grâce à cette vérification, l'émetteur est également authentifié. Le récepteur peut maintenant procéder à une vérification d'intégrité en deux étapes. A partir de la graine $graine\_verif\_prel$ reçue, le récepteur sélectionne le sous-ensemble V des symboles source, puis calcule le hash de ce sous-ensemble et compare sa valeur à $hash\_prel$. Si les deux hash diffèrent le récepteur est certain d'avoir détecté une corruption. Autrement le récepteur ne peut pas encore conclure. Il calcule alors le hash du sous-ensemble complémentaire $\overline{V}$ et le compare au hash $hash\_compl$ reçu. Si les deux hashs différent, le récepteur est alors certain que l'objet a été corrompu, sinon il est certain que l'objet n'a pas été corrompu.

Notons que dans certains champs d'applications, le récepteur pourrait se satisfaire de la vérification préliminaire seulement et ne pas calculer le hash sur $\overline{V}$. Ceci pourrait être pertinent lorsque l'on cherche à se protéger d'attaques aléatoires (erreurs non détectées/corrigées par les couches basses par exemple).

9.4 Évaluation de performances contre les attaques aléatoires

Nous avons implémenté une classe VeriFEC qui dérive de la classe sous-jacente LDPCFecSession du codec C++ open source LDPC-staircase version 2.0 [118]. Nous avons utilisé les primitives cryptographiques de la librairie OpenSSL version 0.9.8c. Plus précisément les signatures électroniques utilisent RSA-1024 et la fonction de hachage est une des suivantes : MD5 [96], RIPEMD-160 [34], SHA-1 et SHA-256 [80].

Nous avons effectué des expériences afin d'estimer les performances de la vérification préliminaire de VeriFEC en termes de probabilité de détection de corruption et de surcoût en calcul. La configuration est celle de la section 9.2.2 : l'objet est composé de $k = 20.000$ symboles et le rendement du code est égal à $R = 2/3$ (sauf en section 9.4.3). Nous supposons que l'attaquant ne veut pas être détecté par la vérification préliminaire et ainsi il se limite à la corruption d'un seul symbole (cas le plus difficile à détecter). Nous supposons également que l'attaquant choisit au hasard le symbole à corrompre (le cas des attaques intelligentes sera traité en section 9.5).

9.4.1 Dépendance vis-à-vis du ratio de vérification

Nous étudions dans un premier temps le nombre N_{verif} de symboles sources qui doivent être vérifiés (i.e. le nombre de symboles dans V) de manière à obtenir la probabilité de détection de corruption désirée pour la vérification préliminaire. Plus N_{verif} sera grand, plus la probabilité de détection sera grande. Ainsi une détection complète est effectuée quand $N_{verif} = k$. Cependant nous voulons aussi garder le surcoût en calcul le plus bas possible, et de ce point de vu N_{verif} doit être le plus petit possible. Afin de trouver un compromis nous avons effectué des expériences au cours desquelles pour chaque ratio de vérification (i.e. N_{verif}/k), nous avons mesuré le pourcentage de corruptions détectées sur 50.000 tests.

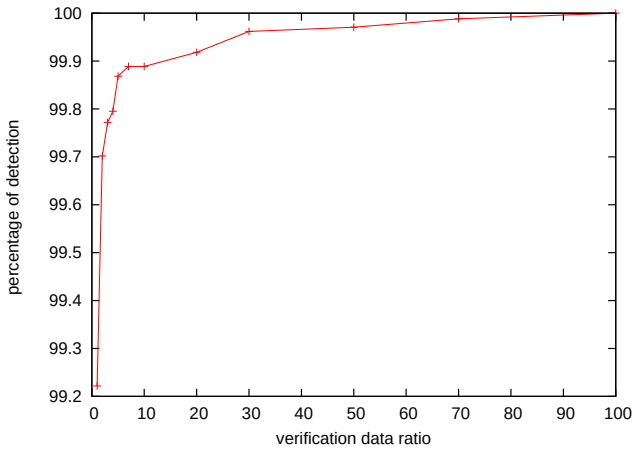


FIG. 9.3 – Probabilité de détection de la vérification préliminaire en fonction du ratio de vérification, N_{verif}/k, sachant qu'un seul symbole, choisi aléatoirement, est reçu corrompu par le récepteur.

Comme attendu, la figure 9.3 montre que la probabilité de détection augmente avec N_{verif}. La vérification de seulement 1% de l'objet décodé permet déjà la détection de $99,22\%$ des attaques. Nous pensons que *la vérification de 5% des symboles pour atteindre une probabilité de détection de 99,86% est un bon compromis entre la détection et le coût en calcul*. Ce ratio sera utilisé dans la suite de ce travail.

9.4.2 Gains en calcul

Maintenant que le ratio de vérification a été fixé à 5%, nous pouvons étudier les gains sur le surcoût en calcul par rapport à la solution de référence (i.e. le codec FEC standard et une vérification d'un hash signé de l'objet dans son intégralité). A cet effet, nous avons mesuré les différents temps de calculs au niveau du récepteur, pour différentes fonctions de hachage, et nous avons calculé la moyenne sur 200 tests. Les expériences ont été effectuées sur un système GNU/Linux avec processeur Intel Dual-Core Xeon 5120 cadencé à $1,86$ GHz et 4 Go de RAM. La taille des symboles a été fixée à 1024 octets, ce qui veut dire que l'objet a une taille de 20 Mo et que l'on y ajoute 10 Mo de parité.

Vérification préliminaire seule :

Le premier scénario correspond au cas où l'objet est corrompu et où la vérification préliminaire la détecte (le cas le plus probable).

	MD5	*RIPEMD-160*	*SHA-1*	*SHA-256*
Récepteur : débit				
FEC+hash signé (s)	651 Mb/s	473 Mb/s	586 Mb/s	337 Mb/s
VeriFEC (s)	865 Mb/s	861 Mb/s	858 Mb/s	828 Mb/s
surcoût relatif (△%)	**-24.7**△%	**-45.1**△%	**-31.7**△%	**-59.2**△%
Récepteur : temps de vérification seul				
Vérification du hash signé (s)	0.0657 s	0.1646 s	0.0937 s	0.3032 s
vérification de VeriFEC (s)	0.0037 s	0.0086 s	0.0051 s	0.0157 s
surcoût relatif (△%)	**-94.4**△%	**-94.8**△%	**-94.6**△%	**-94.8**△%

TAB. 9.1 – Débit et temps de calcul de la vérification préliminaire de VeriFEC et du système standard FEC + Hash signé.

Le tableau 9.1 présente une comparaison du coût de la solution classique avec le coût de VeriFEC avec vérification préliminaire seule. Nous pouvons voir que les surcoûts relatifs ($\triangle = (\text{après} - \text{avant})/\text{après}$) sont très significatifs, et plus particulièrement pour les fonctions de hachage les plus récentes qui nécessitent une charge de calcul importante. Avec SHA-256, le gain relatif pour le décodage FEC et la vérification est de $59,2\triangle\%$ en faveur de VeriFEC (avec SHA-1 le gain est déjà de $31,7\triangle\%$). Si on se concentre sur le processus de vérification seul, on observe que VeriFEC réduit le coût de $94,8\triangle\%$, ce qui est en accord avec l'amélioration théorique de 95% (puisque seulement 5% des symboles sont vérifiés).

Vérification complète :

Le deuxième scénario correspond au cas où la vérification préliminaire n'a pas détecté de corruption, ce qui veut dire que soit l'objet n'est pas corrompu, soit que la vérification préliminaire n'a rien détecté. Le coût de la solution classique est comparé à celui de VeriFEC quand les vérifications préliminaire et complémentaire sont effectuées.

	Solution classique (FEC+hash signé)	*VeriFEC (prel+compl avec hash)*	*surcoût relatif*
Emetteur FEC + hash temps création	0.2946 s	0.2975 s	+0.98 △%
Récepteur FEC + temps vérification	0.3880 s	0.3911 s	+0.80 △%
Récepteur temps . de vérification seul	0.1724 s	0.1753 s	+1.68 △%

TAB. 9.2 – Temps de calcul total pour VeriFEC contre la solution classique FEC + hash signé.

Un léger surcoût en calcul est attendu puisque les morceaux de données qui sont utilisées par la fonction de hachage durant les deux étapes de vérifications ne sont pas nécessairement contiguës en mémoire. Nous avons mesuré ce surcoût en utilisant la fonction de hachage RIPEMD-160. Le tableau 9.2 montre que le surcoût reste faible, $+1.68\triangle\%$ (si on considère uniquement le temps de vérification sans décodage FEC).

Gains en calculs en fonction du ratio de corruption des objets :

Nous nous intéressons maintenant au cas de la transmission de plusieurs objets et aux bénéfices de VeriFEC en fonction du ratio de corruption des objets (i.e. la proportion d'objets corrompus par l'attaquant). Dans le cas de la solution classique "FEC + hash signé", le coût en calcul est constant. A l'opposé, le coût de vérification de VeriFEC varie grandement : si très peu d'objets sont corrompus, la vérification complémentaire (coûteuse) sera presque toujours effectuée ; par contre, si le nombre d'objets corrompus est grand, alors la plupart des corruptions seront identifiées par la vérification préliminaire peu coûteuse, économisant ainsi le temps de calcul.

Introduisons quelques notations :

- T_{Verif} : temps moyen pour effectuer une vérification de l'objet,
- $T_{\text{Preliminary\_Verif}}$: temps de la vérification préliminaire,
- $T_{\text{Compl\_Verif}}$: temps de la vérification complémentaire,
- $P_{\text{Corruption\_Objet}}$: ratio de corruption des objets,
- $P_{\text{Detection\_Pre\_Verif}}$: probabilité de détection de la vérification préliminaire.

Avec le système VeriFEC, le temps de vérification moyen est donné en fonction du ratio de corruption des objets par :

$$T_{\text{Verif}} = T_{\text{Preliminary\_Verif}} + T_{\text{Compl\_Verif}} \times (1 - P_{\text{Corruption\_Objet}} \times P_{\text{Detection\_Pre\_Verif}})$$

Nous avons utilisé $P_{\text{Detection\_Pre\_Verif}} = 0,9986$ (section 9.4.1). Nous avons mesuré expérimentalement les autres paramètres et, avec RIPEMD-160, nous avons trouvé qu'en moyenne : $T_{\text{Preliminary\_Verif}} = 0,0091s$ et $T_{\text{Compl\_Verif}} = 0,1662s$.

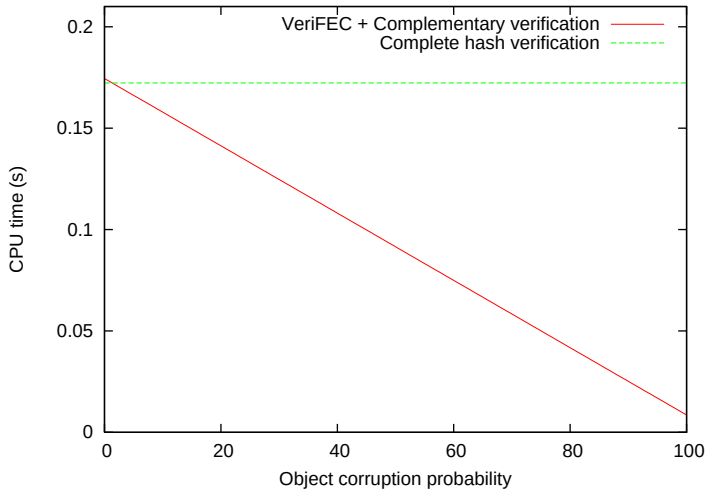


FIG. 9.4 – Temps de vérification en fonction du ratio de corruption des objets.

La figure 9.4 montre les courbes pour chacune des solutions sans inclure le temps de décodage FEC (qui est identique dans les deux cas). S'il n'y a pas de corruption, notre

système introduit un léger surcoût. Ce surcoût devient nul quand $1,3\%$ des objets transmis sont corrompus. Puis, plus le ratio de corruption devient grand, plus notre système est efficace.

9.4.3 Dépendance vis-à-vis de la dimension et du rendement du code

Nous allons maintenant analyser l'influence de la dimension du code (le nombre de symboles source) et du rendement sur la probabilité de détection de la vérification préliminaire. Puisque ces deux paramètres ont été fixés dans les expériences précédentes, nous allons maintenant nous assurer que le système VeriFEC reste efficace pour différentes dimensions et différents rendements.

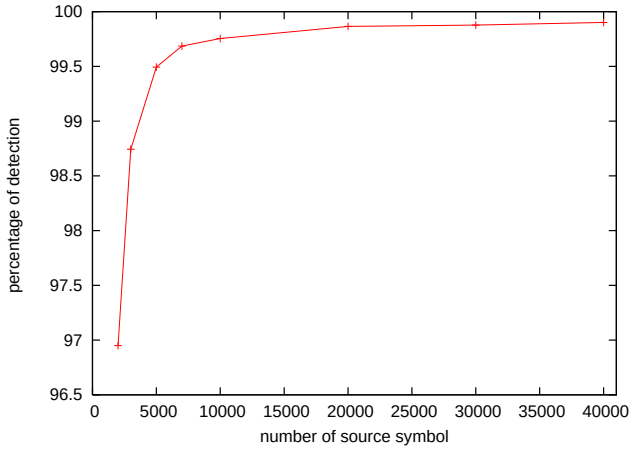


FIG. 9.5 – Probabilité de détection de la vérification préliminaire en fonction de la dimension du code ($R = 2/3$).

La figure 9.5 montre l'évolution de la probabilité de détection en fonction de la dimension du code pour un rendement $R = 2/3$. La probabilité de détection augmente rapidement avec la dimension du code, et pour une dimension de $k = 4000$, la probabilité de détection est déjà de $98,75\%$. Ainsi VeriFEC correspond bien aux conditions d'utilisations des codes LDPC-staircase sous-jacent qui sont des codes AL-FEC grand blocs, travaillant avec des dimensions de plusieurs milliers de symboles [98].

Concernant le rendement du code, la figure 9.6 montre la probabilité de détection en fonction du ratio d'expansion qui est l'inverse du rendement. Cette probabilité reste comprise entre $99,53\%$ et $99,90\%$ quand le rendement varie entre $0,33$ et $0,91$.

Nous pouvons ainsi conclure que la dimension du code ainsi que le rendement n'affectent pas significativement l'efficacité de VeriFEC.

9.5 Cas des attaques intelligentes

Dans cette section, nous considérons le cas d'un attaquant intelligent et possédant une puissance de calcul potentiellement illimitée. Le bénéfice principal de VeriFEC étant d'avoir une vérification préliminaire détectant les attaques avec une grande probabilité, nous ne considérons que les attaques qui la réduisent significativement, i.e. celle qui mènent à des probabilités de non détection significativement plus grandes que $1,4.10^{-3}$

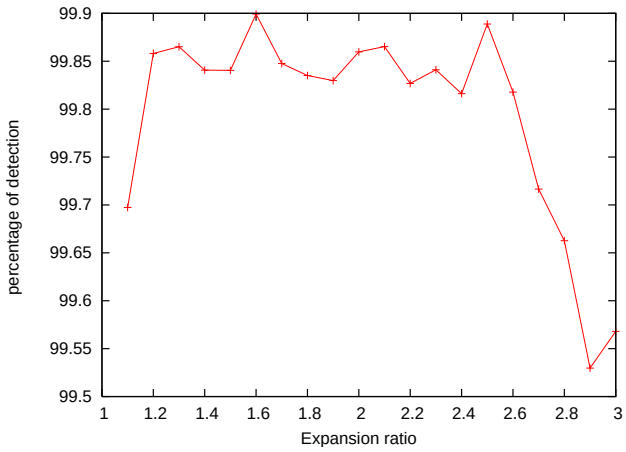


FIG. 9.6 – Probabilité de détection de la vérification préliminaire en fonction du rendement du code ($k = 20.000$).

(voir section 9.3). Il faut noter qu'au final toutes les attaques seront détectées par la vérification complémentaire.

9.5.1 Prévention des attaques simples par extension de l'ensemble V des symboles vérifiés

Supposons dans un premier temps que le code LDPC utilisé est connu de l'attaquant, autrement dit que le triplet $\{k, n, seed\}$ déterminant complètement le code LDPC est transmis en clair [99]. Dans ce cas, la vérification d'un sous-ensemble des symboles source lors de la vérification préliminaire n'est plus suffisante. En effet, un attaquant intelligent pourra choisir un mot de code de corruption avec un unique "1" dans la partie source. Pour trouver un tel mot de code, l'attaquant a juste besoin d'encoder le vecteur source possédant un unique bit à "1" (puisqu'il connaît le code) et de récupérer les bits de parités associés (il y aura plusieurs bits de parité égaux à "1"). Ensuite l'attaquant ajoute au mot de code transmis le mot de code de corruption calculé et le transmet au récepteur[5]. La probabilité de détection si on ne vérifie que les symboles sources est alors égale à N_{verif}/k (i.e. le ratio de vérification).

Une première contre-mesure est de *choisir le sous-ensemble vérifié V sur l'ensemble des symboles sources et parité*. Le *sous-ensemble complémentaire $\overline{V}$ reste le même* et contient uniquement les symboles sources qui ne sont pas dans V. En fait l'algorithme de décodage itératif reconstruit déjà une grande partie, si ce n'est la totalité, des symboles de parité. Le surcoût associé peut donc être négligé.

Une autre contre-mesure est de cacher le code LDPC utilisé. Cette technique sera décrite en détail dans la section suivante.

9.5.2 Prévention des attaques par mot de poids faible

Nous allons maintenant décrire un nouveau type d'attaque utilisant des *mots de code de poids faible* ou LWC ("Low Weight Codewords") et nous proposerons des contre-mesures.

[5]Evidemment, l'attaquant peut ajouter le mot de code de corruption à n'importe lequel des s mots de code binaire, s étant la taille en bit des symboles.

Le besoin de mots de poids faible

Sans perte de généralité, concentrons nous sur un des s mots de code binaire (nous supposons que l'attaquant a reçu les n symboles, et connaît donc les s mots de code correspondants). Soit $SS(.)$ la fonction qui sélectionne le sous-ensemble de N_{verif} bits du mot de code aux positions sectionnées pour le sous-ensemble V. Une corruption sur un mot de code w ne sera pas détectée si l'attaquant créé un mot de code $w' \neq w$ tel que $SS(w') = SS(w)$[6]. Connaissant le sous-ensemble vérifié, trouver un mot de code de corruption possédant des bits nuls aux positions correspondant au sous-ensemble vérifié V, est suffisant pour monter une attaque réussie sur la vérification préliminaire.

Une contre-mesure triviale est de cacher à l'attaquant le sous-ensemble vérifié. Ceci peut être fait de plusieurs façons : en envoyant la graine *graine_verif_prel* sur un canal sécurisé, ou en le chiffrant, ou encore en l'envoyant à la fin de la transmission avec un moyen pour le récepteur de vérifier que les paquets n'ont pas été excessivement retardés. En effet, si le sous-ensemble vérifié est révélé une fois que les symboles ont été reçus, il sera trop tard pour que l'attaquant effectue une attaque. Cacher le code constitue aussi une contre-mesure simple.

Si la graine *graine_verif_prel* est cachée, l'attaquant peut toujours espérer que le sous-ensemble vérifié V n'inspectera pas les bits non nuls du mot de code de corruption, e. La probabilité de succès associée à une telle attaque dépend du poids de Hamming de e, $H_w(e)$, et de la taille du sous-ensemble vérifié N_{verif}. La probabilité de non détection (PND) est en fait la probabilité au cours de N_{verif} tirages, de ne tirer aucun élément d'un ensemble de cardinal $H_w(e)$ parmi n, et peut s'exprimer comme :

$$PND = \begin{cases} \prod_{i=0}^{N_{verif}-1} \frac{(n - H_w(e) - i)}{n - i} & \text{si } H_w(e) \leq n - N_{verif} \\ 0 & \text{si } H_w(e) > n - N_{verif} \end{cases}$$

La figure 9.7 montre que la probabilité de non détection chute rapidement à mesure que le poids de Hamming du mot de code de corruption augmente. Pour un ratio de vérification de 5%, les mots de code de corruption ayant un poids de Hamming supérieur à 200 auront une probabilité de non détection inférieure à 10^{-3}, i.e. inférieure à la probabilité de non détection intrinsèque de la vérification préliminaire de VeriFEC. Ainsi, seuls les mots de code de poids inférieurs à 200 représentent une menace. Ceci nous amène au problème de la recherche de mot de code de poids faible (LWC).

Recherche de mots de poids faible

L'attaquant a besoin de mots de code de poids faible pour effectuer son attaque. Il est connu que les codes LDPC possèdent de tels mots de code (voir section 2.5). L'attaquant a alors deux possibilités :

- il peut tirer avantage des s mots de code binaires extraits des paquets transmis pour y trouver un LWC ;
- ou alors il peut essayer de trouver des mots de poids faible à partir de la définition du code lui même. Ceci suppose soit qu'il connaît le code LDPC, soit qu'il est capable d'inférer le code utilisé à partir des paquets transmis.

[6] On considère que la fonction de hachage est résistante aux collisions, i.e. la probabilité d'obtenir deux objets différents possédant le même hash peut être négligée.

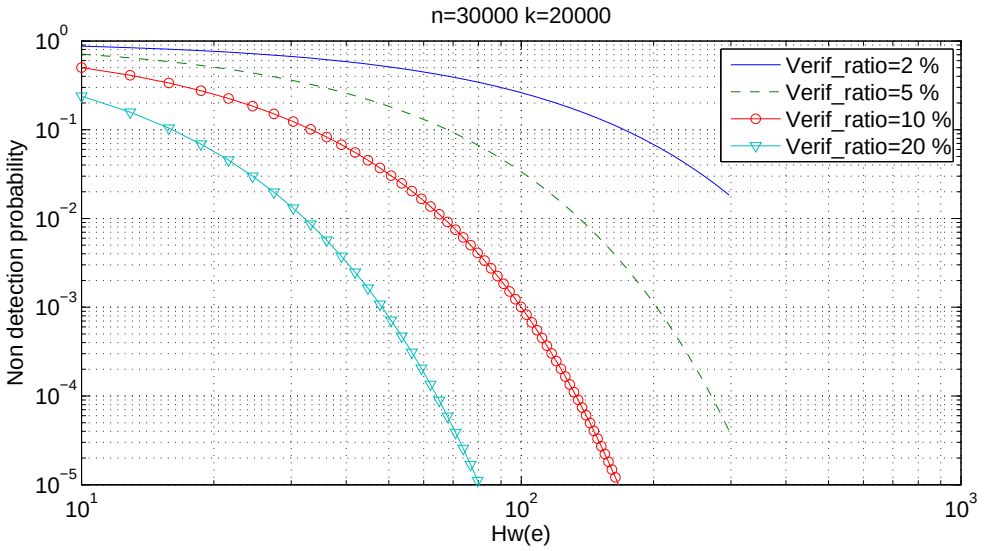


FIG. 9.7 – Probabilité de non détection (PND) de la vérification préliminaire en fonction du poids de Hamming du mot de code de corruption pour différents ratios de vérification (n=30.000, k=20.000).

Considérons la première possibilité. Ici la probabilité qu'un des s mots de code transmis soit un LWC, ou bien qu'une combinaison linéaire de ces s mots de codes soit un LWC doit être considéré. Soit N_t le nombre de mots de code de poids t dans le code $\mathscr{C}$. Ce nombre peut être approximé par $N_t \simeq \frac{C_n^t}{2^{n-k}}$. En supposant que les s mots de code transmis sont linéairement indépendants (le pire cas), ils engendrent un espace vectoriel T de dimension s, et on peut donc produire à partir de cet ensemble 2^s mots de code différents. Soit $N_t^{\leq}$ le nombre de mots de code de poids inférieur ou égal à t. On peut donner une majoration de cette grandeur :

$$N_t^{\leq} \simeq \sum_{i=1}^{t} \frac{C_n^i}{2^{n-k}} \leq \frac{tC_n^t}{2^{n-k}} \leq \frac{t}{2^{n-k}} \frac{n^t}{t!}$$

La probabilité qu'un mot de code de poids inférieur à t appartienne à cet ensemble T est :

$$P_{s,t} = \frac{Card(T)}{Card(\mathscr{C})} * N_t^{\leq} = \frac{2^s}{2^k} * N_t^{\leq} \leq \frac{t}{2^{n-s}} \frac{n^t}{t!}$$

Cette probabilité tend vers zéro quand n tend vers l'infini. Dans notre cas w (resp. s) est deux (resp. un) ordre de grandeur plus petit que n, ainsi la probabilité qu'un LWC soit transmis est très faible et nous pouvons donc la négliger.

Considérons maintenant la seconde possibilité. Trouver un LWC à partir de la définition d'un code LDPC (par exemple à partir de la matrice de parité) peut être effectué par une recherche exhaustive, ou avec des algorithmes plus élaborés [23]. La complexité de tels algorithmes peut être un obstacle pour un attaquant avec une capacité de calcul limitée. Cependant, afin d'obtenir une sécurité inconditionnelle, nous considérons que l'attaquant est capable de trouver un LWC à partir de la connaissance du code. Cela mène au problème de la dissimulation du code LDPC.

Dissimuler le code

Intéressons nous maintenant au problème de la dissimulation du code à l'attaquant. Changer de code à chaque transmission est trivial avec les codes LDPC-staircase, puisque ces codes sont générés à la volée grâce à un PRNG et à une graine de 32 bits qui peut être facilement changée à chaque transmission [99]. Comme pour la graine *graine_verif_prel*, la graine utilisée pour la génération du code peut être facilement dissimulée.

Cependant un attaquant peut faire appel à des *techniques de reconnaissance de code* [119] pour deviner le code utilisé. Le nombre de mots de code requis pour une reconnaissance de code LDPC dans un environnement bruité a été étudié dans [26] (dans notre cas on suppose que les mots de code interceptés ne contiennent pas d'erreur). Le problème de la reconnaissance d'un code LDPC est équivalent à trouver sa matrice de parité.

Les codes LDPC-staircase possèdent une matrice de parité de la forme $H = (H_1|H_2)$, où H_1 est une matrice de taille $(n-k) \times k$ avec une distribution régulière des degrés ligne et colonne, et H_2 est une matrice de taille $(n-k) \times (n-k)$ escalier . Soit N_1 le degré des colonnes de H_1, i.e. le nombre de '1' par colonne, et $t = N_1 \frac{R}{1-R}$ le degré des lignes de H_1, où $R = k/n$ est le rendement du code (ces degrés résultent des spécifications du code [99]). Notons E l'ensemble de telles matrices. H_1 défini un graphe biparti régulier avec k noeuds gauche de degré N_1 et $n-k$ noeuds droit de degré t. Un dénombrement de ces codes est présenté dans [26], et d'après l'équation (8) on a :

$$log_2(card(E)) \sim \frac{N_1(t-1)}{t} log_2(n) \tag{9.1}$$

Le nombre de mots de codes nécessaires pour la reconnaissance du code est donc de l'ordre de $\log_2(n)$. Supposons que s, le nombre de mots de code connus de l'attaquant, est tel que $s < \frac{N_1(t-1)}{t} log_2(n) - C$, où C est une constante dont on ajustera la valeur. Le nombre de codes potentiels (i.e. les choix possibles) est alors de l'ordre de 2^C et ainsi la probabilité de choisir le bon code parmi les codes potentiels (i.e. lancer avec succès une attaque) est de 2^{-C}. En choisissant $C = 10$, on rend cette probabilité de détection égale à $2^{-10} = 0.00098$, i.e. légèrement plus petite que la probabilité de non détection de la vérification préliminaire (voir section 9.4.1).

Considérons les même conditions que celles utilisées en section 9.2.2. Ainsi $k = 20000$ et $R = 2/3$, il suit que $n = 30000$ et que $t = 6$ ($N_1 = 3$ est la valeur par défaut avec les codes LDPC-staircase utilisant le décodage itératif). On a donc $\frac{N_1(t-1)}{t} log_2(n) - C = 27.18$, ce qui veut dire qu'il est suffisant de prendre $s < 27$ bits.

Pour conclure on peut dire qu'en pratique, quand $k = 20000$ et $R = 2/3$, *utiliser des symboles de taille égale à 3 octets, dissimuler le code LDPC (i.e. la graine associée) et dissimuler* graine_verif_prel*, empêche les attaques intelligentes contre la vérification préliminaire de VeriFEC.*

9.6 Travaux relatifs

Dans [59], les auteurs introduisent un système qui corrige les erreurs et vérifie l'intégrité des symboles avec une "très haute" probabilité quand les erreurs sont aléatoires. Puis ils étendent leur système au cas plus complexe des attaques intelligentes, à l'aide d'une technique appelé "scrambling". Si on se limite à la première contribution, la solution repose sur l'utilisation d'un décodeur spécifique pour le canal q-aire symétrique. Cette

solution diffère complètement de la nôtre qui conserve le même algorithme de décodage itératif sur le canal à effacements, et vérifie un sous-ensemble des symboles après décodage. L'objectif est également différent puisque VeriFEC n'essaie pas de corriger ni de localiser les corruptions.

L'idée de détection de corruptions à partir du calcul d'un hash partiel de l'objet a été présentée dans [30]. Cette technique est appelée *Striped Hashing*. L'objectif de ce travail est de fournir une solution capable de vérifier l'intégrité du code exécuté sans pour autant ralentir le système. Le sous-ensemble des octets inclus dans le *striped hash* suit un motif particulier, qui permet de maximiser la probabilité de détection d'octets consécutifs corrompus dans les données. L'efficacité du *striped hashing* est nettement inférieure à celle de notre système. En effet, la corruption d'un unique octet pourra être détectée par VeriFEC avec une probabilité élevée grâce au phénomène de propagation. A l'inverse, la solution de *Striped Hashing* ne détectera la corruption qu'avec une probabilité égale au ratio de données pris en compte dans le hash partiel.

Dans [54] les auteurs présentent un système qui est capable de vérifier à la volée les symboles reçus (i.e. avant le décodage). Ce système est basé sur une fonction de hachage homomorphique qui possède la propriété suivante : le hash d'une combinaison linéaire de symboles peut être obtenu comme une fonction des hashs des symboles correspondants. Grâce à cette fonction de hachage homomorphique, le récepteur peut calculer le hash de n'importe quel symbole encodé à partir des valeurs des hashs des symboles sources. Il peut donc vérifier l'intégrité de n'importe quel symbole reçu uniquement à partir d'un ensemble de k hashs au lieu de n. L'avantage de cette solution est que seul les symboles corrects seront utilisés pour le décodage (i.e. les symboles corrompus sont détectés et effacés). Ainsi, on est sûr que l'objet décodé ne contiendra pas d'erreurs si le décodage réussit. Cependant cette solution nécessite l'emploi d'additions sur $\mathbb{Z}_q$ qui sont beaucoup plus coûteuses que les opérations XOR utilisées pas les codes LDPC-staircase. D'après les auteurs, ces opérations ajoutent un surcoût d'environ 500% en temps de calcul. Ceci va à l'encontre de notre objectif de réduction de complexité. De plus, il existe un surcoût significatif en transmission puisque les hashs de chaque symbole source doivent être transmis, alors que VeriFEC nécessite uniquement l'envoi du triplet $\{graine\_verif\_prel; hash\_prel; hash\_compl\}$.

9.7 Généralisation aux codes MDS

Ce système peut être facilement généralisé pour les codes à effacements MDS tels que les codes Reed-Solomon. En effet, contrairement aux codes LDPC, ces codes ne possèdent pas de mot de poids faible, puisque le poids minimum d'un mot d'un $[n,k]$-code MDS est $H_{min} = n-k+1$.

Cette propriété est particulièrement intéressante et peut être utilisée pour effectuer une vérification d'intégrité à complexité réduite à la sortie d'un décodeur qui ne retourne que des mots de code, et ceci sur n'importe quel canal. En effet si l'on considère des mots de code d'un code MDS, on sait que deux mots de code diffèrent en au moins $n-k+1$ symboles sur les n. Ainsi une borne supérieure de la probabilité de non détection d'une vérification partielle peut être calculée comme la probabilité au cours de N_{verif} tirages sans

remise, de ne tirer aucun élément d'un ensemble de cardinal $n-k+1$ parmi n :

$$PND \begin{cases} \leq & \prod_{i=1}^{N_{verif}} \frac{k-1-i}{n-i} & \text{si } N_{verif} < k \\ = & 0 & \text{si } N_{verif} \geq k \end{cases}$$

Il faut noter que la vérification de k symboles d'un mot de code est suffisante pour garantir à 100% l'intégrité de du mot de code. Cependant cela revient à vérifier l'intégrité d'une quantité de données de la même taille que les données originales.

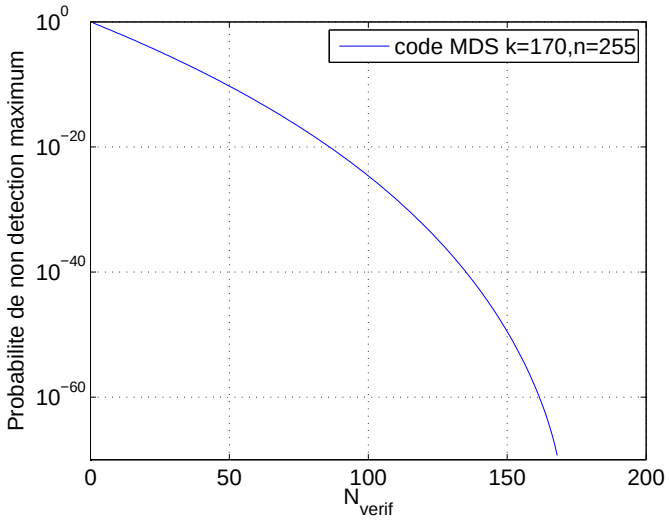


FIG. 9.8 – Probabilité de non détection maximum, pour une vérification partiel d'un mot de code MDS ($k = 170$, $n = 255$).

La figure 9.8 montre l'évolution de la grandeur $\prod_{i=1}^{N_{verif}} \frac{k-1-i}{n-i}$, qui est la probabilité de non détection maximale, en fonction du nombre de vérifications pour un $[255, 170]$-code MDS (caractéristiques communes des codes Reed-Solomon utilisés en pratique). On voit que la probabilité maximale de non détection chute rapidement et que la vérification de 27 symboles permet déjà d'atteindre une probabilité de $6,5431.10^{-6}$.

Grâce au poids important des mots de codes, l'utilisation d'un système type VeriFEC avec des codes MDS permet donc d'obtenir une meilleure probabilité de détection pour un nombre de symboles vérifiés donné. Cependant il faut noter que l'intérêt d'un système type de VeriFEC avec des codes MDS peut être moins important qu'avec des codes haut débit tels que les LDPC-staircase. En effet le débit des codes MDS est de plusieurs dizaines de Mb/s, ce qui est généralement inférieur à celui des fonctions de hachage. Ainsi, la réduction du coût de la vérification d'intégrité aura un impact limité sur les performances globales du système. Néanmoins, même si le gain relatif en complexité est réduit, il peut exister des cas où cela a du sens. Par exemple, lorsque le décodeur du code MDS possède un débit élevé (implémentation matérielle) ou quand l'opération de vérification d'intégrité est très coûteuse, ce qui risque de devenir le cas avec la complexification des fonctions de hachage cryptographiques de nouvelle génération.

9.8 Conclusions

Nous avons montré dans ce chapitre qu'une fonctionnalité de détection de corruptions et d'authentification de la source, peut être efficacement adjointe à un code AL-FEC LDPC-staircase. Le système proposé, VeriFEC, vérifie l'intégrité de l'objet décodé en deux étapes : la première étape détecte une majorité des corruptions avec un coût en calcul faible, tandis qu'une seconde étape de vérification permet d'atteindre une probabilité de détection égale à 100%.

Les résultats expérimentaux montrent que *VeriFEC détecte 99,86% des attaques aléatoires les plus difficiles à détecter (où un seul symbole est corrompu) pour un coût en calcul inférieur à 6% de celui de la vérification complète de l'objet à l'aide d'un hash signé*, et ceci sans modifier les capacités de correction. Si l'attaque aléatoire est moins subtile (par exemple si plusieurs symboles sont corrompus), la probabilité de détection de la vérification préliminaire augmente jusqu'à presque 100%. Le cas des attaques intelligentes dont le but est de réduire la probabilité de détection de la vérification préliminaire a également été étudié. Nous avons montré que les attaques utilisant les mots de code de poids faible peuvent être évitées en réduisant la taille des symboles et en cachant un petit nombre des paramètres clefs du système.

Globalement, grâce à son faible surcoût en calcul, VeriFEC peut être d'une grande aide pour limiter l'impact des attaques aléatoires ou intelligentes visant à provoquer un déni de service. De plus, si les menaces sont uniquement constituées d'attaques aléatoires, et si une probabilité d'intégrité élevée est suffisante, l'utilisation de la vérification préliminaire seule prend tout son sens. On note que ceci reste un cas particulier.

Ce système peut-être généralisé à d'autres codes LDPC à condition que ceux-ci puissent être cachés au lieu d'être complètement déterminés par la paire $\{n;k\}$. Il peut également être utilisé avec des codes MDS tels que les Reed-Solomon, mais comme la vitesse de décodage de ces codes est généralement faible comparé à celle des fonctions de hachage, le gain relatif sera limité.

Des travaux futurs exploreront d'autres techniques pour la dissimulation du code aux attaquants potentiels, comme par exemple l'ajout d'un bruit connu aux symboles transmis. Un objectif est de relâcher la contrainte sur la taille des symboles (voir 9.5.2), actuellement relativement importante (3 octets maximum).

Chapitre 10

Chiffrement et codage Reed-Solomon conjoints

Contents

Ce chapitre présente un système de codage et de chiffrement conjoint à complexité réduite. Grâce aux relations existant entre les codes Reed-Solomon et les systèmes de “secret-sharing”, nous montrons qu’un chiffrement partiel des symboles encodés est suffisant pour empêcher l’accès au contenu par des utilisateurs illégitimes. Nous présentons plusieurs attaques possibles sur ce système et nous proposons des parades appropriées.

10.1 Introduction

Les systèmes de communications modernes ont souvent à faire face à deux enjeux primordiaux : la qualité de service et la confidentialité de l’information transportée. Les codes correcteurs, et en particulier les codes à effacements, sont souvent proposés pour résoudre la problématique de qualité de service, alors que la problématique de confidentialité est laissée aux outils de chiffrement.

Le chiffrement et le codage possèdent deux objectifs diamétralement opposés : alors que le chiffrement tend à limiter l’accès à une information, le codage tend à rendre l’information la plus disponible possible. Malgré cette différence d’objectifs, les codes et les fonctions de chiffrement possèdent des caractéristiques communes. En effet, comme le

codage à effacements, le chiffrement se situe dans la pile protocolaire à un niveau proche du niveau applicatif (par exemple OpenSSL [115] se situe entre la couche application et la couche transport) et ils travaillent le plus souvent sur le même flux d'information. De plus, les codes correcteurs comme les fonctions de chiffrement travaillent avec le même type d'objets (par exemple des éléments de corps fini) et le même type d'opérations [49] (par exemple les opérations XOR ou l'inversion dans un corps fini pour l'opération "SubBytes" d'AES [77])[1]. Il est donc naturel d'essayer de combiner un système de chiffrement à un code à effacements afin de réduire la complexité globale du système.

Cette approche de combinaison des deux fonctionnalités, même si elle est difficile du fait de l'antagonisme des fonctionnalités, a fait l'objet de plusieurs travaux [69] [43]. Le système résultant est communément appelé "Crypt-coding".

Au lieu de proposer une nouvelle construction d'une fonction de crypt-coding en partant de zéro, dont la sécurité est souvent difficile à démontrer, nous nous proposons dans ce chapitre de combiner deux fonctions existantes et éprouvées afin de construire un système de "crypt-coding". Ce système utilise des codes à effacements MDS, en l'occurrence les codes Reed-Solomon (voir section 2.4), ainsi qu'une fonction de chiffrement. La sécurité de la proposition reposera sur celle du système de "secret sharing" (voir section 10.2.2).

10.2 Conception du système proposé

10.2.1 Objectifs

L'objectif de ce travail est de créer un système combinant les fonctionnalités d'un code à effacements avec celle d'une fonction de chiffrement, dans le but d'économiser les ressources en calcul. Nous comparerons notre proposition à la solution classique qui consiste à chiffrer d'abord le contenu et à le coder ensuite (voir figure 10.1). La solution classique nécessite donc le chiffrement de k symboles, puis l'encodage par le code à effacements de k symboles sources en n symboles codés.

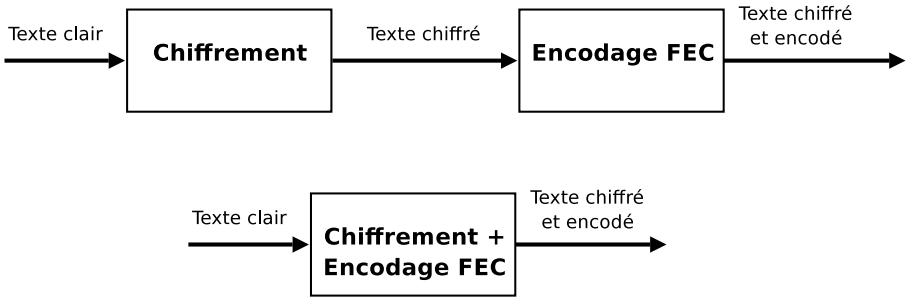


FIG. 10.1 – Solution classique (chiffrement puis codage) et solution conjointe (chiffrement et codage dans une même fonction).

Le système proposé devra satisfaire les contraintes suivantes :

– tout récepteur légitime (i.e. possédant la clef) doit être capable de récupérer le contenu et de corriger autant de pertes qu'avec le code à effacements original,

[1] On notera que les codes correcteurs peuvent être utilisés pour construire des systèmes de chiffrement asymétriques dit post-quantiques, tel que le crypto-système de McEliece [71]. Ces systèmes font usage de problèmes complexes inhérents aux codes correcteurs afin de garantir leur robustesse. Cependant ces systèmes font uniquement du chiffrement et n'effectuent pas de correction.

– tout récepteur illégitime (i.e. ne possédant pas la clef) doit être incapable de récupérer le contenu (même partiellement),
– le coût en calcul de l'étape d'encodage/chiffrement du système combiné devra être inférieur à celui de la solution classique,
– le coût en calcul de l'étape de décodage/déchiffrement du système combiné devra être inférieur à celui de la solution classique.

Notre proposition est basée sur une idée simple qui consiste à ne chiffrer qu'une seule partie du contenu encodé. Le chiffrement d'une partie des données après qu'elles aient subi une transformation a été présenté dans [70] et [50]. Le cas particulier où la transformation préliminaire est un code à effacements a été présenté dans [21] pour le cas des codes Tornado [22] et dans [41] pour le cas des codes LDPC. Ces deux exemples seront présentés plus en détail dans la section 10.4.

10.2.2 Mécanisme de "secret sharing"

Un mécanisme de "secret sharing" est un système, proposé par Shamir en 1979 [101], permettant le partage d'un secret entre n entités. Le secret est partagé en n morceaux distribués aux n entités. Ce système est tel que le secret ne peut être révélé que si au moins t entités parmi les n réunissent leurs morceaux de secret. Tout groupe de taille strictement inférieure à k ne peut obtenir aucune information sur le secret.

En 1981 McEliece remarqua [72] que le système de secret sharing de Shamir était en fait un cas particulier des codes Reed-Solomon. Considérons un code de Reed-Solomon systématique de dimension k et de longueur n. Soit $\{a_0,\ldots,a_{k-1}\}$ le secret à partager. Le code Reed-Solomon encode ce vecteur en un nouveau vecteur $\{D_0,\ldots,D_{n-1}\}$. A cause du caractère systématique du code, $\{D_0,\ldots,D_{k-1}\}$ est égal au secret $\{a_0,\ldots,a_{k-1}\}$, tandis que $\{D_k,\ldots,D_{n-1}\}$ correspond aux $n-k$ morceaux de secret. En effet, si l'on dispose de k morceaux de secret parmi $\{D_k,\ldots,D_{n-1}\}$, alors le décodage à effacements du code Reed-Solomon nous retournera $\{a_0,\ldots,a_{k-1}\}$ qui est le secret. A l'opposé, si l'on dispose de moins de k morceaux de secret, il sera impossible de décoder et le secret demeurera inconnu.

Ceci peut être illustré avec la construction à base de matrice de Vandermonde des codes Reed-Solomon (voir section 2.4.1). La matrice de Vandermonde est $V = \{\alpha^{ij}\}_{0\leq i\leq n-1;0\leq j\leq k-1}$ α étant un élément générateur du corps fini considéré. Le décodage à partir de r symboles connus, revient à résoudre un système linéaire $V'X = Y$ ayant k inconnues et r équations. Si $r < k$ alors le système est sous-contraint et il est impossible de résoudre, même partiellement, le système.

Pour un corps fini donné, l'utilisation d'un code de Reed-Solomon systématique va limiter le nombre de symboles de redondance que pourra produire notre système et donc limitera le rendement minimum atteignable. Il serait donc préférable d'utiliser un code non systématique, afin de s'affranchir de cette contrainte. Ceci présente l'inconvénient d'augmenter la complexité de l'encodage et potentiellement du décodage, mais nous pensons que ce surcoût sera compensé par les autres avantages apportés par notre solution. Dans la suite nous considérerons que le code de Reed-Solomon utilisé est non-systématique et donc que les symboles sources ne sont pas présents dans l'ensemble des symboles codés.

10.2.3 Chiffrement partiel

Nous proposons d'utiliser la propriété de "secret-sharing" du code Reed-Solomon pour limiter l'accès au contenu encodé. En chiffrant suffisamment de symboles encodés, il est possible d'empêcher un récepteur non légitime de récupérer le contenu(voir figure 10.2). En effet le chiffrement de $n-k+1$ symboles, laissera $k-1$ symboles en clair, ce qui est insuffisant pour obtenir une information sur le contenu original à cause de la propriété de "secret sharing" du code. Le chiffrement de $n-k+1$ symboles codés peut ne pas être suffisant pour empêcher l'accès au schéma de décodage direct. En effet un attaquant pourra essayer de deviner la valeur du symbole qui lui manque pour effectuer le décodage. La complexité de cette attaque dépend du corps fini utilisé et est en fait proportionnelle au nombre de valeurs possibles pour le symbole manquant (une étude plus détaillée des attaques sera présentée en section 10.3). Il faut donc prévoir une marge de sécurité sur le nombre de symboles à chiffrer. On fixe alors à $n-k+t$ le nombre de symboles encodés à chiffrer, où $t \geq 1$ détermine la *marge de chiffrement*. On appelle $C = \frac{n-k+t}{k}$ le ratio de chiffrement, qui mesure le rapport des quantités de données à chiffrer avec notre proposition et la solution classique.

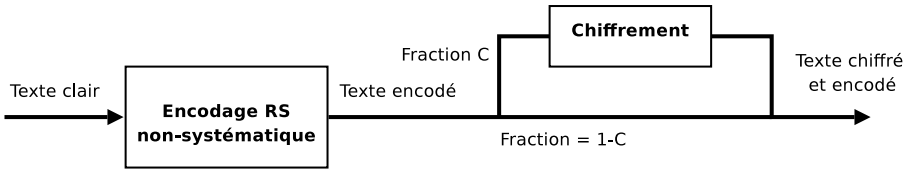


FIG. 10.2 – Encodage et chiffrement partiel.

Le nombre de déchiffrements à effectuer dépend du nombre de symboles codés reçus en clair. L'utilisation d'une permutation aléatoire pour l'ordre de transmission des symboles permettra d'avoir une fraction constante de symboles à déchiffrer, quel que soit le schéma de pertes du canal.

10.2.4 Gain théorique

Ce système a la particularité de nécessiter le chiffrement de $n-k+t$ symboles au lieu de k. L'avantage que l'on va en tirer dépend du rendement R du code utilisé. En effet le coût du chiffrement ne sera réduit que si :

$$\begin{aligned} n-k+t &< k \\ R &> 1/2+t/(2n) \end{aligned}$$

Ainsi, le système proposé n'est avantageux que si le rendement est supérieur à $1/2$.

La figure 10.3 présente le ratio de chiffrement $C = \frac{n-k+t}{k}$ du système proposé ainsi que le ratio de chiffrement correspondant à la solution classique en fonction du rendement R. Le ratio de chiffrement de la solution classique est constant et égal à 1 (puisque la quantité d'informations à chiffrer est k quel que soit R). Le ratio de chiffrement est en faveur de notre système quand le rendement dépasse $1/2$, et la quantité de données à chiffrer est d'autant plus faible que le rendement s'approche de 1 et inversement.

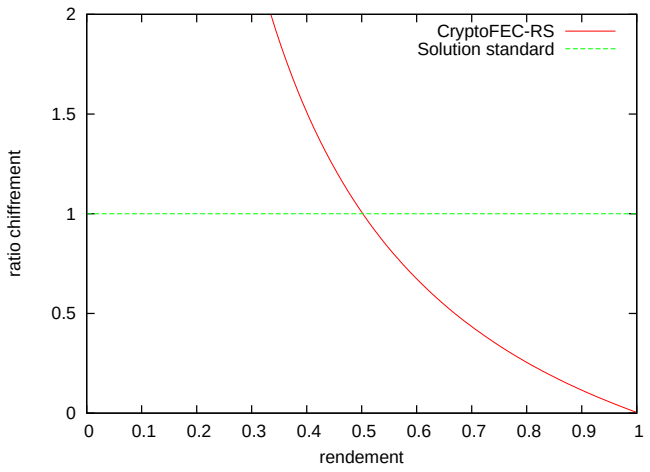


FIG. 10.3 – Ratio de chiffrement code RS ($n = 255$, $t = 1$).

10.3 Analyse de la sécurité du système

Nous allons maintenant nous intéresser à la sécurité du système. Nous faisons l'hypothèse que la fonction de chiffrement est robuste. En d'autres termes la fonction de chiffrement est résistante aux attaques à texte chiffré connu.

10.3.1 Attaques par "force brute"

Comme introduit en section 10.2.3, il est possible d'attaquer notre système en devinant la valeur de certains symboles. En effet dans le cas où l'attaquant a reçu la totalité des symboles envoyés (le pire cas), il dispose de $n-k+t$ symboles codés chiffrés et de $k-t$ symboles codés en clair. S'il arrive à deviner la valeur de t symboles chiffrés il disposera de k symboles codés en clair et pourra donc décoder et récupérer le contenu original.

Pour effectuer cette attaque, il faut que l'attaquant soit capable de reconnaître le contenu parmi des textes aléatoires. Supposons que le code soit basé sur le corps fini $\mathbb{F}_q$ qui possède q éléments. L'attaquant procède en essayant toutes les valeurs possible du t-uplet des symboles manquants, et s'arrête quand le résultat du décodage est reconnu comme un texte clair. Cette attaque est en fait une attaque par "force brute" sur les valeurs de symboles manquants. L'espace de recherche contient q^t éléments, l'attaquant aura donc besoin en moyenne d'effectuer $q^t/2$ décodages différents pour retrouver le contenu en clair. La complexité de l'attaque est donc exponentielle en la marge t. La valeur de t peut donc être facilement augmentée de manière à obtenir une complexité suffisamment grande pour prévenir toute attaque de ce type. Cependant la valeur de t a une influence sur l'efficacité du système car le ratio de chiffrement C dépend linéairement de celle-ci. Un compromis devra donc être trouvé entre l'efficacité du système et la complexité des attaques de type "force brute".

10.3.2 Attaques avec texte clair partiellement connu

Les contenus à encoder sont rarement des données aléatoires, et la plupart du temps, une partie de celles-ci peut être connue par l'attaquant. Par exemple un fichier vidéo possédera des entêtes dont les valeurs sont fixées ou sont limitées à un petit nombre de

valeurs (par exemple l'identifiant du codec vidéo). On doit alors supposer que l'attaquant connaît un sous-ensemble des symboles source.

Ceci est une menace pour la sécurité de notre système. En effet la connaissance d'une partie des symboles sources va diminuer le nombre de variables du système linéaire à résoudre. Ceci va impliquer une diminution du nombre de symboles codés nécessaire au décodage. Supposons que g symboles sources sont connus de l'attaquant. Le système linéaire ne possède donc plus que $k-g$ variables. $k-g$ symboles codés seront donc suffisants pour résoudre le système et récupérer le reste du contenu original. Si $g \geq t$ alors le décodage pourra être effectué à partir des $k-t$ symboles codés clairs. Sinon le décodage n'est pas faisable directement, mais la complexité d'une attaque de type "brute-force" sera diminuée.

Encore une fois, on pourra utiliser le paramètre t pour ajuster la sécurité du système en fonction de la quantité d'information prévisible dans le contenu et la sécurité du système face aux attaques "brute-force" avec texte clair partiellement connu.

10.4 Travaux relatifs

L'idée d'un chiffrement partiel du contenu encodé par un code à effacements a été proposée par Byers dans [21]. Le code à effacements considéré est un code Tornado [22]. Contrairement à notre proposition, ce travail implique la modification du code considéré et ainsi ne garantit pas que les capacités de correction du système combiné soient les mêmes que celles du système original. Une proposition similaire est présentée dans le brevet [41] où les auteurs proposent d'ajouter une fonctionnalité de chiffrement à un code LDPC pour le canal à effacements.

On pourra citer les approches présentées dans [70] et [50], où l'objectif est de réduire la taille des données à chiffrer en utilisant une transformation des données originales. La transformation préliminaire des données a comme unique objectif de permettre un chiffrement à posteriori à bas coût, alors que dans notre système, la transformation préliminaire a une utilité propre et indépendante du chiffrement puisqu'elle permet la correction d'effacements.

10.5 Conclusions

Dans ce chapitre nous avons présenté un système combinant la correction d'effacements des codes Reed-Solomon et la protection du contenu. Grâce à un chiffrement partiel du contenu codé par un code Reed-Solomon, ce système empêche la récupération du contenu si la clef utilisée par la fonction de chiffrement est inconnue. Ce système permet une réduction de la complexité, puisque la quantité de données devant être chiffrées est inférieure à la taille du contenu original. La réduction de complexité est d'autant plus grande que le rendement du code s'approche de 1. Une analyse de la sécurité du système a été proposée par l'étude d'attaque de type "force brute" et d'attaque à texte clair partiellement connu. Des contre-mesures efficaces ont été proposées pour faire face à ces menaces.

Troisième partie

Applications des codes à effacements

Chapitre 11

Utilisation des codes linéaires pour le calcul distribué tolérant aux fautes

Contents

Les plate-formes de calcul pair-à-pair peuvent être l'objet d'un large nombre de fautes et d'attaques. Dans ce chapitre nous proposons une généralisation d'une approche dites de "disk-less checkpointing" pour la tolérance aux fautes dans les systèmes de calcul hautes performances. Notre contribution est double :

- *Premièrement, au lieu de se restreindre aux contrôles de parité en deux dimensions qui ne peuvent tolérer qu'un nombre limité de fautes, nous proposons l'utilisation des codes linéaires pour tolérer un plus grand nombre de fautes.*
- *Deuxièmement, nous comparons l'utilisation des codes LDPC et des codes Reed-Solomon pour différents modèles de fautes présents sur les systèmes pair-à-pair. Notre méthode de "disk-less checkpointing" basée sur les codes LDPC est bien adaptée quand on ne considère que les déconnexions de pairs, mais elle n'est pas adaptée au problème des pairs malicieux. Notre proposition de "disk-less checkpointing" basée sur les codes Reed-Solomon est quant à elle capable de tolérer les fautes malicieuses, mais est limitée aux opérations sur les corps finis.*

Ce travail a été effectué en collaboration avec Thomas Roche et Jean-Louis Roch du LIG et a fait l'objet d'une publication [116].

11.1 Introduction

En rassemblant des milliers de ressources, les plate-formes pair-à-pair (P2P) sont de formidables systèmes de calcul et de stockage de données à grande échelle. Cependant, de telles plate-formes peuvent être sujettes à plusieurs types de fautes. Dans ce travail, nous nous concentrerons sur les erreurs par arrêt, ou panne franche (par exemple des déconnexions de pairs), et sur les erreurs par valeur [7].

Dans le domaine des systèmes distribués, on appelle byzantin le comportement d'un pair qui ne répond pas conformément aux spécifications du système [56]. Ainsi, un pair qui renverra des résultats erronés ou qui ne renverra pas de résultat du tout sera qualifié de byzantin. Dans certains contextes, ce comportement byzantin peut être modélisé statistiquement. Cependant, comme les plate-formes P2P opèrent habituellement dans des environnements non contrôlés, elles peuvent être les cibles d'attaques à grande échelle utilisant les vulnérabilités d'un système d'exploitation particulier. Ainsi, afin de prendre en compte le pire des cas, on doit considérer que les attaques sont malicieuses [100], ce qui rend impossible toute modélisation ou quantification de ces attaques.

Le système BOINC [6] est un système de calcul distribué P2P très populaire (une de ses utilisations est le support du projet "seti@home" pour la recherche d'une intelligence extraterrestre via l'analyse des signaux radio). Afin de faire face au possible comportement byzantin des pairs, BOINC utilise une solution basée sur la réplication des tâches et la recherche d'un consensus sur les résultats. Sur le projet "seti@home" chaque tâche est répliquée 2 fois par défaut. Malgré le surcoût important d'une telle solution, la probabilité qu'un résultat erroné soit détecté comme tel est loin de 100%. Ainsi une grande quantité de ressources est "gaspillée" afin d'obtenir une tolérance toute relative aux erreurs.

Lorsque de longs calculs sont effectués sur une plate-forme pouvant être victime d'erreurs, des solutions dites de "checkpointing" sont utilisées afin de permettre de revenir à un état de calcul intermédiaire sans erreurs dans le cas où une erreur est détectée. Ceci peut être effectué en stockant les résultats intermédiaires sur un support stable. Lorsque l'on veut se libérer de la contrainte de support de stockage stable, on utilise des systèmes dits de "diskless checkpointing" qui utilisent des calculs redondants pour garantir la propriété de "checkpointing".

Une technique commune de "diskless checkpointing" est proposée par la tolérance aux fautes basée sur l'algorithme, ou ABFT pour "Algorithm-based Fault Tolerance", qui a été introduit en 1984 par K.H Huang et al [48]. L'ABFT était à l'origine dédiée aux opérations sur les matrices pour lesquelles elles constituent une solution efficace en termes de temps, d'espace et de complexité. L'ABFT est basée sur la conception conjointe d'une technique de codage avec un algorithme donné. A cause des propriétés algébriques des opérations matricielles, l'ABFT pour les algorithmes de multiplication matricielle repose sur l'utilisation de codes linéaires. Étant donné la complexité du produit matriciel, $O(n^3)$, et la taille des matrices rencontrées dans certains problèmes, il est souvent utile, voire indispensable, d'effectuer les calculs sur des plate-formes disposant d'une grande puissance de calcul, telles que les plate-formes P2P.

L'utilisation de l'ABFT sur des plate-formes de calcul globales a été proposé par J.L. Roch et al. [100] pour les calculs exacts. G. Boscila et al [15] ont présenté une nouvelle méthode d'ABFT pour la multiplication matricielle sur les plate-formes de calcul hautes performances, ou HPC ("High Performance Computing"). Cette méthode est basée sur un contrôle de parité au niveau des lignes et des colonnes couplé avec l'algorithme SUMMA

[120]. Cette méthode sera présentée plus en détail en section 11.2. Son principal avantage est qu'elle effectue le calcul tout en ayant des résultats intermédiaires consistants vis à vis du système de "disk-less checkpointing". L'utilisation de contrôles de parité permet la tolérance d'au plus une erreur par valeur ou au plus 3 erreurs par arrêt. Si ces capacités sont adaptées aux systèmes HPC de taille modérée, elles ne le sont plus dans le cas des plate-formes de calcul grande échelle de type P2P.

En se focalisant sur le domaine des plate-formes de calcul P2P, la première contribution de ce travail est une généralisation de la technique de G. Boscila et al en utilisant des codes linéaires de manière à tolérer efficacement un plus grand nombre d'erreurs (section 11.3). Puis nous proposons l'utilisation de deux classes de codes correcteurs : les codes LDPC [42] pour les opérations sur les flottants (section 11.4), et les codes Reed-Solomon qui résistent aux attaques malicieuses pour les calculs exacts (section section 11.5). A notre connaissance il s'agit de la première proposition d'ABFT pour plate-formes P2P basée sur les codes LDPC.

11.2 ABFT appliquée aux calculs hautes performances

11.2.1 Un systèmeABFT consistant pour la multiplication matricielle

Rappelons les principes du système ABFT proposé en [15] pour les multiplications matrice-matrice. Ce système est basé sur deux idées. Premièrement, quand les matrices en entrée contiennent des sommes de parité sur les lignes et les colonnes alors, en l'absence d'erreurs, la matrice résultante possède aussi des sommes de parité valides pour les lignes et les colonnes. Deuxièmement à chaque étape durant la multiplication, les sommes de parité sont toujours consistantes (les matrices intermédiaires possèdent aussi des sommes de parité valides). Ainsi si A et B sont les matrices en entrée et C_R et C_C sont les matrices de contrôle de parité, les matrices encodées respectivement en A_F et B_F sont les suivantes :

$$A_F = \begin{pmatrix} A & AC_R \\ C_C^T A & C_C^T A C_R \end{pmatrix} \quad B_F = \begin{pmatrix} B & BC_R \\ C_C^T B & C_C^T B C_R \end{pmatrix}$$

On peut écrire le produit des matrices encodées :

$$\begin{pmatrix} A \\ C_C^T A \end{pmatrix} \times \begin{pmatrix} B & BC_R \end{pmatrix} = \underbrace{\begin{pmatrix} AB & ABC_R \\ C_C^T AB & C_C^T ABC_R \end{pmatrix}}_{(AB)_F} \tag{11.1}$$

L'équation 11.1 présente le produit des matrices encodées et prouve la consistance des sommes de parité pour la multiplication complète. En effet la matrice résultante, $(AB)_F$, est bien la matrice AB avec les sommes de contrôle valides.

De plus dans cette proposition, la multiplication matricielle est effectuée dans sa version produit extérieur : (`for k=1 :n,` $C_k = C_k + A_{:,k}B_{k,:}$ `end`). Les matrices intermédiaires C_k étant composées de sommes et de produits de matrices consistantes, elles sont aussi consistantes. La figure 11.1 présente les détails de ce produit avec les sommes de contrôle sur une matrice 3×3.

L'algorithme SUMMA ("Scalable Universal Matrix Multiplication Algorithm") [120], est un algorithme efficace de multiplication matricielle en parallèle, qui est largement

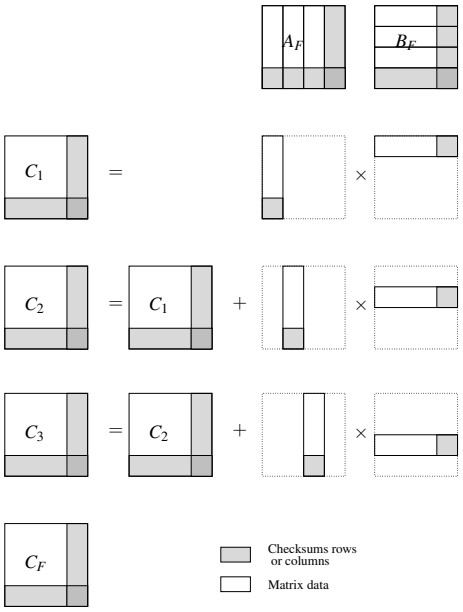


FIG. 11.1 – Produit extérieur de matrices consistant vis à vis des sommes de contrôles

utilisé (par exemple dans ScaLAPACK [35]). Considérons un ensemble de p processus effectuant la multiplication matrice-matrice distribué sur une grille carré de $\sqrt{p}$ par $\sqrt{p}$. Soit $n \times k$ (resp. $k \times m$) la taille de A_F (resp. B_F). Les colonnes de A_F (resp. les lignes de B_F) sont traitées comme des blocs de nb lignes (resp. colonnes) .

La matrice A_F (resp. B_F) est divisée en blocs de taille $\sqrt{p}$ par $\frac{k}{nb}$ (resp. $\frac{k}{nb}$ par $\sqrt{p}$). A la i-ème étape de la multiplication (comme décrit sur la figure 11.1), $\forall\, j$ le (i, j)-ème bloc de A_F est envoyé aux processus de la j-ème ligne de la grille et le (i, j)-ème bloc de B_F est envoyé aux processus de la j-ème colonne de la grille. Ensuite chaque processus effectue en local une multiplication matrice-matrice de taille $\frac{n}{\sqrt{p}}$ par nb par $\frac{m}{\sqrt{p}}$, dont il ajoute le résultat à son bloc local. Le calcul s'arrête après nb itérations. Après chacune des nb itérations de produit extérieur, une synchronisation globale est effectuée pour s'assurer que les données sont consistantes vis-à-vis de la somme de contrôle. Le paramètre nb peut varier de 1 (un état consistant après chaque calcul de ligne) à n (seulement un état consistant à la fin du calcul global).

Les calculs redondants sont effectués par les processus correspondants au dernier bloc de ligne de A_F et au dernier bloc de colonne de B_F. En effet ces blocs ne contiennent que des données relatives aux sommes de contrôle. Si un processus s'arrête en erreur au cours de la i-ème itération de l'algorithme, son état peut être récupéré à partir de l'information des autres processus. En fait, la somme de contrôle en 2 dimensions permet de récupérer jusqu'à 3 arrêts en erreur de processus pendant la même itération[1].

Le coût en nombre de processus de la tolérance aux fautes est de $2 \times \sqrt{p} - 1$. Ainsi la fraction de processus utilisés pour la tolérance aux fautes ($\frac{2\times\sqrt{p}-1}{p}$) décroît quand le nombre de processus augmente. Ce système est donc fortement "scalable" (le surcoût décroît avec la taille du système).

[1] Il est garanti que n'importe quelle configuration de 3 arrêts de processus pourra être récupérée. Cependant il est possible que le système puisse récupérer certaines configurations de plus de 3 arrêts de processus.

11.2.2 Certification de résultat

Il est important de noter qu'avec ce système, la multiplication matrice-matrice bénéficie d'une vérification à posteriori efficace. Que l'on travaille sur les corps finis ou sur les flottants, il est toujours possible de vérifier que le résultat $C = AB$ est correct avec une probabilité élevée : on choisit au hasard un vecteur x de taille m et on vérifie que $Cx = A \times Bx$. La coût de cette vérification est égal à celui de trois produit matrice-vecteur (complexité $O(n^2)$), ce qui est négligeable devant le coût du produit matrice-matrice (complexité $O(n^3)$). De plus, chacun des produits intermédiaires C_k de l'algorithme SUMMA peut être certifié de la même manière puisque $C_k = A_k B_k$, où A_k (resp. B_k) est la sous-matrice de A (resp. B) composée des k premiers blocs de nb colonnes (resp. blocs de nb lignes).

11.2.3 Limitations

Pour une probabilité d'erreur donnée, le nombre d'erreurs par valeur et d'arrêt en erreurs va augmenter à mesure que la taille de la plate-forme augmente. Cependant le systèmeABFT que nous venons de décrire possède des capacités de correction limitées : trois arrêts en erreur de processus et une erreur par valeur. Ainsi, il n'est pas adapté aux réseaux P2P qui sont sujet à des erreurs et/ou des déconnexions fréquentes.

11.3 Généralisation aux codes linéaires et extension aux réseaux de calcul pair à pair

De manière à répondre aux besoins des plate-formes P2P, nous proposons une généralisation du systèmeABFT précédent. En observant que les sommes de contrôle sont en fait un cas particulier des codes linéaires, nous proposons de généraliser ce système en employant des codes linéaires plus flexibles de manière à augmenter le nombre d'erreurs et de pannes tolérables par le système.

11.3.1 Codes linéaires par bloc et multiplication matricielle

Les codes linéaires sont une classe de codes utilisés pour la détection et la correction d'erreurs. Grâce à l'ajout de redondance aux données originales, ils sont capable de détecter et corriger des erreurs par valeur et des effacements. Ces codes et les notions associées ont déjà été introduits dans le chapitre 2. Nous rappelons ici brièvement les principaux résultats en insistant sur le cas des erreurs par valeur. Un code bloc linéaire $\mathscr{C}$ de dimension n et de longueur $n+r$ est un sous espace vectoriel de dimension n de l'espace vectoriel $\mathbb{F}^{n+r}$, où $\mathbb{F}$ est un corps fini. La quantité $R = n/(n+r)$ est appelé le rendement du code. Les éléments de $\mathbb{F}$ sont appelés *symboles*, et les éléments de $\mathscr{C}$ sont appelés des *mots du code*.

Les capacités de correction d'un code linéaire sont habituellement estimées par la caractéristique appelée *distance minimale*. La distance minimale (notée d_{min}) d'un code linéaire est le nombre minimum de symboles non-nuls dans un mot de code non nul. Un code avec une distance minimale d_{min} sera capable de détecteur $d_{min} - 1$ erreurs et d'en corriger $\lfloor \frac{d_{min}-1}{2} \rfloor$. De manière générale, un mot de code corrompu avec t erreurs et e effacements pourra être corrigé si $d_{min} \geq 2t + e + 1$. Les codes Maximum Distance

Séparable sont tels que $d_{min} = r+1$: leur distance minimale est maximum par rapport au nombre de symboles de redondance r.

Le codage proposé en [15] peut être facilement généralisé avec n'importe quel code linéaire dont le corps fini est compatible avec les opérations arithmétiques des matrices. Les matrices de contrôle de parité C_C et C_R sont remplacées par les matrices génératrices du code linéaire choisi. Grâce à la structure algébrique des codes linéaires, le produit (ainsi que la somme) de deux matrices encodées est toujours une matrice encodée consistante par rapport au code. Dans la suite, nous ne considérerons que le codage des colonnes avec la matrice C_C, le codage des lignes avec la matrice C_R étant analogue.

11.3.2 ABFT reposant sur les codes linéaires

Grâce à la généralisation précédente, un produit matriciel consistant vis à vis du code peut être construit. Le nombre d'erreurs tolérables par le système sera déterminé par le taux de redondance du code linéaires utilisé, qui pourra être ajusté en fonction des besoins.

Nous nous intéressons maintenant au produit matriciel effectué sur une grande plate-forme HPC. Considérons le produit des matrices A et B, effectué par l'algorithme SUMMA comme présenté dans la section 11.2 et un code linéaire de dimension n et de longueur $n+r$. La matrice A, de taille $n \times k$, est encodée en une matrice A_F de taille $(n+r) \times k$.

Comme en section 11.2, on considère un ensemble de p ressources vues comme une grille de taille $\sqrt{p}$ par $\sqrt{p}$. Le nombre de fautes qui peuvent être tolérées entre chaque itération de l'algorithme SUMMA dépend de la quantité de redondance ajoutée par le code. Dans le cas d'un code MDS, l'ajout de r symboles de redondance permettra au système de tolérer e effacements et t erreurs tant que $r = d_{min} - 1 \geq 2t + e$. Puisqu'un processus est en charge d'effectuer le calcul de $(n+r)/\sqrt{p}$ symboles d'une colonne, une unique panne franche d'un noeud (arrêt en erreur) provoquera l'effacement de la même quantité de symboles, tandis qu'une erreur par valeur provoquera la corruption de cette même quantité de symboles. De manière à tolérer E pannes franche de noeuds (représentées par des effacements) et T noeuds renvoyant des résultats erronées, nous avons besoin d'avoir :

$$r \geq \frac{2T(n+r)}{\sqrt{p}} + \frac{E(n+r)}{\sqrt{p}}$$

ou bien :

$$r \geq \frac{n(2T+E)}{\sqrt{p} - (2T+E)}$$

Ainsi la fraction de processus utilisés pour la tolérance aux fautes est :

$$\frac{(r\sqrt{p}/(n+r))\sqrt{p}}{p} = r/(n+r) = 1-R$$

Cette fraction ne dépend que du rendement du code. On peut ainsi dire que ce système est faiblement "scalable" (le surcoût ne varie pas en fonction de la taille du système) ; mais contrairement à [15], le pourcentage de calculs redondants est constant, ce qui est indispensable pour la tolérance aux fautes sur les plate-formes de calcul P2P.

11.3.3 Adaptation de l'ABFT aux plate-formes P2P

Pour les multiplications de matrices de grandes dimensions, l'utilisation d'un système ABFT basé sur les codes linéaires, est une solution intéressante pour corriger les erreurs par

valeur, les pertes d'information ainsi que pour certifier les résultats intermédiaires. Toutes ces opérations sont effectuées en un temps en $\tilde{O}(n^2)$, tandis que la multiplication elle même nécessite un temps de calcul en $O(n^3)$. La version produit extérieur de la multiplication matrice-matrice nous permet d'effectuer tout cela en permettant un mécanisme efficace de "checkpoint" tolérant aux fautes présenté sur la figure 11.2.

Considérons un utilisateur U proposant une multiplication matrice-matrice sur un réseau de calcul P2P. La propriété de "disk-less checkpointing", qui repose sur la synchronisation globale des ressources participant au calcul n'est pas adaptée aux plate-formes P2P, où toutes les données (A_F, B_F et les matrices C_k) sont distribuées sur les pairs. Après la distribution des données sur le réseau P2P par le téléchargement des matrices d'entrées et des tâches (on pourra utiliser pour cela un système de téléchargement P2P tel que Bittorent [1]), les pairs vont travailler sur les données qu'ils ont en leur possession (on pourra utiliser des codes "régénérants" [38] pour garantir efficacement la persistence des données en cas de déconnexion de pairs). Le mécanisme de "checkpoint" est assuré en envoyant les résultats intermédiaires à l'utilisateur U (après chaque itération ou après un certain nombre d'itérations). Lorsque U a reçu suffisamment de blocs correspondant à la matrice intermédiaire C_k de l'itération k, il peut décoder C_k, corrigeant les erreurs potentielles et récupérant les blocs manquant. U effectue ensuite une vérification à posteriori sur le résultat décodé afin de certifier de manière probabiliste le résultat intermédiaire C_k. Si la vérification à posteriori n'est pas passée avec succès, il peut envoyer un message *stop* aux pairs afin de recommencer le calcul à partir du dernier état certifié.

On note que le produit extérieur s'intègre parfaitement à ce protocole P2P. En effet, grâce à la nature hautement parallèle de l'algorithme, et à la granularité ajustable des tâches, ce système supportera bien la division et la répartition des tâches en ligne (particulièrement adapté pour les techniques de *vol de travail* [100] par exemple). De plus, il n'est pas nécessaire que l'utilisateur partitionne les matrices en bloc en fonction du nombre de ressources (nombre qui peut être difficile à estimer dans un contexte P2P). On pourra par exemple imaginer un mécanisme récursif de division de la matrice par une approche de *vol de travail*, qui répartira automatiquement et de manière équitablement la charge de travail des pairs.

11.4 Codage LDPC pour les plate-formes de calcul pair à pair non malicieuses

On considère un réseau P2P où les pairs sont susceptibles de partir avant d'avoir terminé leur tâche ou bien de renvoyer un résultat erroné. Ce cas peut être résolu par l'utilisation de codes LDPC (voir section 2.5). Même si la distance minimale de ces codes est faible, ces codes possèdent des capacités de corrections proche de la capacité du canal lorsqu'ils sont décodés avec un algorithme itératif à complexité linéaire.

Puisque les codes LDPC sont en fait une mise en parallèle de plusieurs sommes de parité, comme celles utilisées dans [15], les propriétés numériques des sommes de contrôle avec les flottants sont toujours valides lorsque l'on corrige des effacements. Cependant la correction d'erreurs par valeur avec les codes LDPC doit être restreinte à l'arithmétique exact des corps finis. En effet, à notre connaissance, le décodage itératif des erreurs avec des valeurs flottantes n'a pas été étudié, et cette généralisation n'est pas immédiate. En effet la propagation des erreurs d'arrondi dans les algorithme de passage de message risque

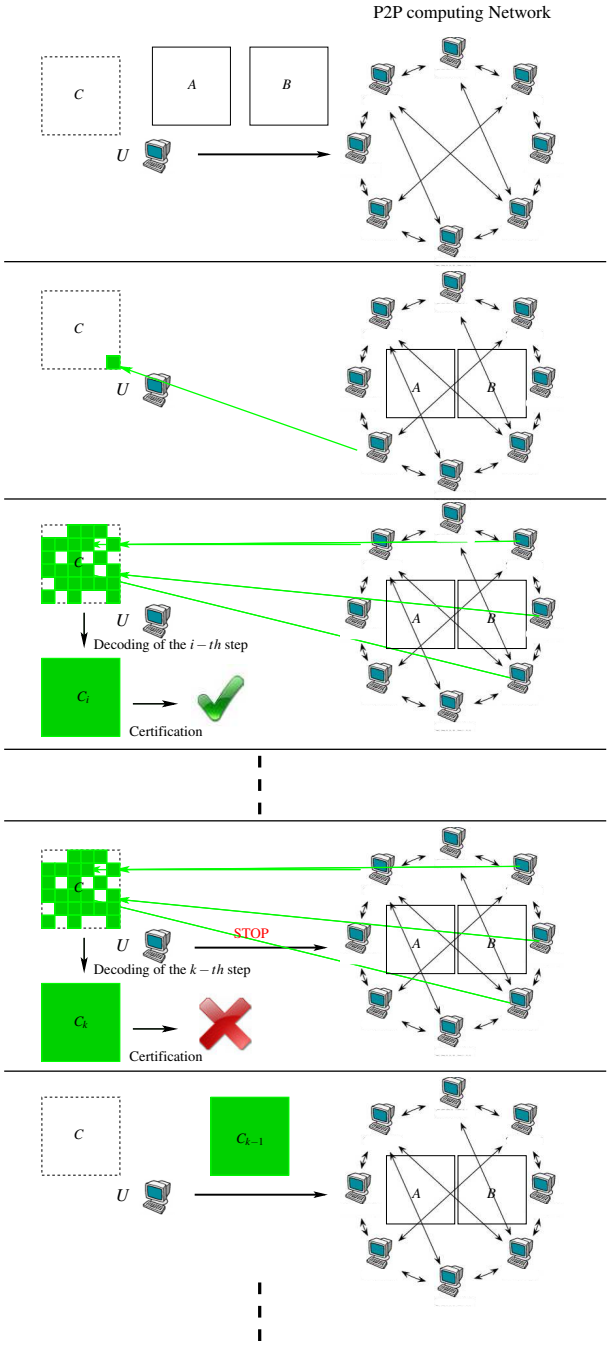


FIG. 11.2 – Protocole de “checkpoint” tolérant aux fautes

d'empêcher l'algorithme de converger, voir de le faire diverger.

En se basant sur les résultats publiés [94] [85] sur les capacités de correction des codes LDPC, on peut dire que l'utilisation d'un système d'ABFT basé sur les codes LDPC permet la tolérance d'un nombre de déconnexions et d'erreurs par valeur non malicieuses proche de l'optimal avec une probabilité élevée (voir tableau 11.1). L'algorithme utilisé fonctionne avec une complexité $O(n)$ [42], ce qui est du même ordre de complexité que le système de somme de contrôle de [15] : le décodage complet de la matrice nécessite le décodage de n mots de code et a donc une complexité totale de $O(n^2)$.

11.5 Codage Reed-Solomon pour les plate-formes de calcul pair à pair malicieuses

Nous considérons maintenant le modèle de faute byzantine dans le contexte des réseaux P2P sujet à des attaques malicieuses. En effet la plupart du temps, on ne peut pas avoir une confiance totale dans les pairs d'un réseau P2P, ce qui limite la confiance que l'on peut avoir dans le résultat renvoyé. Même si la multiplication matrice-matrice possède une vérification à posteriori efficace (voir section 11.2), il faut encore être sur que cette vérification n'a pas été forgée par des pairs malicieux. De plus en supposant que l'on peut avoir confiance en la vérification, il est toujours possible de mettre en place pour un faible coût, des attaques de type déni de service, ou DoS. En effet la faible distance minimum des codes LDPC signifie qu'il est possible de forger un mot de code valide avec un faible nombre de coordonnées modifiées (i.e. peu de pairs contrôlés par l'attaquant). Ainsi les codes avec des grande distances minimum (par exemple des codes MDS) seront dans ce cas plus adaptés. Supposons que l'on fasse confiance à une partie des ressources de la plateforme de calcul. On s'attend à ce que la taille (et ainsi la puissance de calcul) de cette partie fiable de la plate-forme, soit petite devant la taille de la partie non-fiable. La partie fiable sera utilisée pour le décodage, et pour la vérification à posteriori du résultat. Par exemple sur la figure 11.2, l'utilisateur U est fiable alors que le reste de la plate-forme P2P ne l'est pas.

Codes Reed-Solomon : Les codes Reed-Solomon, ou RS, (voir 2.4) sont des codes MDS introduit en 1960 par I. Reed et G. Solomon [91]. Ils sont toujours capables de corriger jusqu'à $r/2$ erreurs (là où les codes LDPC ne sont capables de corriger qu'avec une certaine probabilité), tout en ayant une complexité de décodage supérieure à celle des codes LDPC ($O(n \log n^2)$ [11], voir table 11.1). Les codes RS sont bien adaptés pour tolérer les erreurs malicieuses dans le contexte du calcul exact. Ainsi, avec une redondance suffisamment grande, l'utilisation d'un mécanisme d'ABFT de multiplication matricielle en calcul exact basé sur les codes RS, empêchera les attaques malicieuses non-massives [100] : le forgeage d'un résultat erroné nécessitera que l'attaquant prenne le contrôle d'un grand nombre de pairs. Cependant, la non-compatibilité de l'arithmétique sous-jacente fait que les solutions ABFT basées sur les codes RS ne sont pas adaptées aux opérations sur les flottants.

	RS	LDPC
Surcoût en calcul	r/n	r/n
Complexité de décodage (par colonne)	$O(nlog^2(n))$	$O(n)$
Complexité de décodage (matrice complète)	$O(n^2log^2(n))$	$O(n^2)$
Tolérance aux fautes (nb. erreurs par colonne)	$\lfloor r/2 \rfloor$	$0.15(n+r)$ erreurs confiance : $1-10^{-6}$ ($n=r=0.5\times10^6$) [94]
Tolérance aux fautes (nb. effacements par colonne)	$\lfloor r/2 \rfloor$	$0.49(n+r)$ effacements confiance : $1-10^{-4}$ ($n=r=0.25\times10^7$) [85]

TAB. 11.1 – Comparaison des codes RS et LDPC pour l'ABFT

11.6 Conclusions et travaux futurs

Dans ce chapitre nous avons présenté une généralisation d'un mécanisme de multiplication matricielle ABFT basé sur les codes linéaires qui est adapté à l'utilisation sur les plate-formes de calcul P2P. Le protocole proposé permet de tolérer le comportement erratique des réseaux P2P. Lorsque l'on considère le calcul exact, l'utilisation des codes Reed-Solomon fournit une tolérance aux fautes, non seulement dans le cas de déconnexions de pairs, mais également dans le cas d'erreurs par valeur (même dans le cas de pairs malicieux). Pour les calculs sur les flottants, l'utilisation des codes LDPC permet de tolérer des fautes par déconnexion avec des algorithmes d'encodage et de décodage à complexité linéaire.

Nous projetons d'utiliser ce protocole afin de réaliser une implémentation dans le contexte du calcul exact sur une plate-forme de calcul distribué tel que Grid5000.

L'étude du décodage des codes LDPC dans le cas d'erreurs par valeur dans un contexte de calcul flottant constitue un axe de recherche possible. De plus il serait intéressant de trouver des mécanismes d'ABFT adaptés à d'autres opérations que celles de l'arithmétique des matrices. On pourra par exemple citer l'exemple de la plateforme P2P "seti@home" qui effectue principalement des transformées de Fourier rapides (FFT) et dont l'efficacité pourrait être grandement améliorée par l'emploi d'un système ABFT adapté aux FFT.

Quatrième partie

Conclusions et perspectives

Chapitre 12

Résumé des contributions et perspectives

Ce chapitre résume mes contributions et identifie les perspectives jugées les plus pertinentes.

12.1 Codes et codecs hautes performances pour le canal à effacements

12.1.1 Décodage hybride IT/ML

Une description du décodage hybride IT/ML par élimination de Gauss a été effectuée, et une discussion approfondie de l'implémentation optimisée de cet algorithme a été présentée. Les résultats montrent que le décodage hybride IT/ML des codes LDPC de longueur courte et moyenne permet d'obtenir d'excellentes performances. Ces résultats confirment le fait que les capacités de correction des codes LDPC sont sensiblement améliorées par l'utilisation du décodeur ML et qu'elles s'approchent de celles des codes parfaits. De plus grâce à une implémentation judicieuse du décodeur ML, la complexité du décodage est suffisamment faible pour obtenir des débits élevés bien souvent proches de ceux obtenus par décodage IT. Enfin, l'utilisation d'un décodeur hybride permet de réduire encore cette complexité tout en offrant une grande flexibilité au niveau du récepteur. En effet, à partir d'un même code le récepteur pourra choisir dynamiquement le compromis optimal entre les performances de correction et la complexité de décodage.

Travaux futurs : Nous voyons trois grandes pistes afin d'améliorer les codecs reposant sur le décodage hybride IT/ML.

Les codes travaillant sur le corps fini $\mathbb{F}_2^p$ étant équivalents à la mise en parallèle de p codes travaillant sur $\mathbb{F}_2$ (voir section 2.3.3), il serait intéressant de décomposer le décodage de manière à ne travailler que sur un sous-ensemble des p mots de codes à la fois. Ceci permettra de réduire la taille de la mémoire de travail et également d'accélérer le décodage en réduisant les temps d'accès en lecture/écriture aux données, dont le coût peut facilement devenir un goulot d'étranglement.

Une seconde voie d'amélioration serait de paralléliser les algorithmes afin de bénéficier du gain potentiel offert par les architectures multi-coeurs et/ou GPGPU. Il sera possible

de conduire les recherches suivant deux axes, la conception du décodeur, mais aussi la structure du code. En effet des codes structurés tels que les codes quasi-cycliques peuvent supporter des opérations effectuées en parallèle.

Enfin, le décodage ML ne réussissant pas à chaque tentative, en cas d'échec, il faut retenter un nouveau décodage avec les nouveaux symboles reçus par la suite, le cas échéant. Dans la version actuelle du codec, ceci nécessite de recommencer le décodage depuis le début. Il serait intéressant de pouvoir exploiter les opérations effectuées par la précédente tentative de décodage ML afin de réduire la complexité du décodage. On pourra par exemple s'inspirer des algorithmes de "décomposition LU incrémentale" [92] pour lesquels l'ajout de lignes et de colonnes à la matrice à décomposer ne nécessite pas de reprendre la décomposition depuis le début.

12.1.2 Adaptation des codes LDPC-staircase au décodage hybride IT/ML

Nous avons ensuite démontré le potentiel des codes LDPC-staircase lorsqu'ils sont décodés avec un décodeur hybride IT/ML. Les résultats présentés montrent que ces codes possèdent des capacités de correction très proches de celles des codes Raptor. Grâce au contrôle du paramètre N1, il est possible d'adapter les performances du code en fonction du type de décodeur disponible au niveau des récepteurs. Cette fonctionnalité renforce la flexibilité du décodage hybride IT/ML, puisqu'elle permet de choisir entre les performances en décodage IT (capacités de correction moyennes mais vitesse élevée) et les performances du décodage ML (capacités de correction très proches de l'optimal au prix d'une vitesse réduite). Un enseignement est qu'un code très simple permet d'offrir des performances extrêmement intéressantes avec un codec bien implémenté. Ce résultat renforce l'idée que codes et codecs doivent être conçus de pair.

12.1.3 Codes LDPC-Band

Une nouvelle classe de codes LDPC structurés, appelés LDPC-Band, a été présentée. La structure bande de leur matrice génératrice permet une réduction de la complexité du décodage ML, tandis que leur matrice de parité, creuse, permet un décodage itératif efficace. Ces codes sont ainsi parfaitement adaptés au décodeur hybride IT/ML. Les résultats montrent que ces codes possèdent des capacités de correction légèrement inférieures à celles des codes non structurés, mais grâce à leur structure bande, la vitesse du décodage ML est améliorée jusqu'à un facteur cinq.

Ces travaux ont été effectués en collaboration avec Alexandre Soro et Jérôme Lacan de L'ISAE.

Travaux futurs : En conservant l'idée d'une structure bande pour réduire la complexité du décodage ML, il serait possible de construire un code "Repeat-Accumulate" possédant une structure bande dans la partie gauche de la matrice de parité. Le décodage ML sera alors effectué sur la matrice de parité, après une permutation appropriée des colonnes, afin que cette matrice exhibe une unique bande (en effet, les deux sous-matrices de la matrice de parité sont chacune des matrices bandes). La structure bande du système à résoudre permettra une réduction de la complexité du décodage ML. De plus, la simplicité de la

construction de ces codes facilitera de potentielles optimisations, par exemple sous la forme d'une étude par évolution de densité.

12.1.4 Codes LDPC-QC

Suite aux travaux sur les LDPC-Band, et toujours dans l'optique de codes structurés pour faciliter le décodage ML, nous avons ensuite étudié des codes LDPC Quasi-Cycliques, c'est à dire des codes dont la matrice de parité est composée de matrices circulantes. Notre étude préliminaire a montré qu'outre l'amélioration des capacités de correction en décodage IT, la structure quasi-cyclique permet de réduire la complexité du décodage ML par élimination de Gauss. Ce gain en complexité est rendu possible grâce à un remplissage plus lent du système lors de l'élimination de Gauss.

Ces travaux sont effectués en collaboration avec Valentin Savin du CEA-LETI.

Travaux en cours : Etant donnée une matrice de base, la longueur des cycles dans un code LDPC-QC dépend fortement du choix du facteur d'expansion et donc de la longueur du code. Ainsi pour une matrice de base donnée, seules certaines longueurs permettront de tirer parti de manière optimale de la construction quasi-cyclique. Une méthode permettant d'étendre efficacement le code de base pour des longueurs arbitraires est actuellement à l'étude.

12.1.5 Codes G-LDPC, ou LDPC Généralisés

Nous nous sommes également intéressés à la construction de codes à rendement faible. Une approche LDPC-Généralisée a été utilisée pour étendre les codes LDPC-staircase. Cette construction rend possible l'ajout d'une redondance incrémentale, c'est à dire qu'il est potentiellement possible d'augmenter le nombre de symboles de redondance, sans avoir à fixer le rendementlors de l'initialisation, et de les produire à la volée. Nous avons montré qu'il était possible de décoder ces codes de façon efficace avec un algorithme itératif.

Ces travaux ont été effectués en collaboration avec Valentin Savin du CEA-LETI, Alexandre Soro et Jérôme Lacan de L'ISAE.

Travaux futurs : Cette étude s'est concentrée sur les capacités de correction du code. Des travaux futurs s'attelleront à estimer la complexité du système et en particulier les vitesses d'encodage et de décodage. Dans un but de réduction de complexité il serait intéressant de considérer des "codes MDS composants" autres que les codes Reed-Solomon (tels que les array codes [12]). De plus, les codes LDPC-Généralisés peuvent être décodés par un décodeur ML, et un décodage hybride IT/ML peut également être considéré pour ces codes. Ceci devrait permettre d'améliorer les capacités de corrections. Plus généralement le problème du décodage efficace des codes LDPC-Généralisés constitue une suite logique à ces travaux.

12.2 Codes à effacements et primitives cryptographiques

Le développement conjoint de systèmes effectuant de la correction d'effacements et fournissant des fonctionnalités cryptographiques a été étudié suivant deux axes.

12.2.1 VeriFEC

Nous avons étudié tout d'abord l'ajout d'une fonctionnalité de vérification d'intégrité à un code LDPC-Staircase. Le système résultant, VeriFEC, est capable de détecter la plupart des corruptions aléatoires avec un surcoût très faible (la vérification de 5% de l'objet décodé permet de détecter 99,8% des attaques), tandis qu'une vérification complémentaire permet de certifier à 100 % l'intégrité de l'objet décodé. Le problème des attaques intelligentes a été abordé, et nous avons montré que des contre-mesures permettaient de les réduire au cas des attaques aléatoires.

Ces travaux ont grandement bénéficié des discussions que nous avons eu avec Valentin Savin du CEA-LETI, Philippe Gaborit de l'équipe PI2C de l'université de Limoges, Nicolas Sendrier et Jean-Pierre Tillich de l'équipe SECRET de l'INRIA.

Travaux futurs : Des travaux futurs s'attelleront à relaxer les contraintes à ajouter au système pour se prémunir des attaques intelligentes. De plus il devrait être possible de généraliser ce système à d'autres codes et à d'autres types de canaux, tels que les canaux à erreurs par valeur.

12.2.2 Chiffrement et code Reed-Solomon

Un système de chiffrement conjoint à un codage Reed-Solomon pour le canal à effacements a été proposé. Dans les cas où le rendement du code est supérieur à 1/2, ce système réduit la quantité d'informations devant être chiffrées pour empêcher l'accès aux données sources. Cette contribution montre qu'il est possible d'adjoindre, pour un surcoût limité, un mécanisme de restriction d'accès aux données sur des flux encodés par un code Reed-Solomon.

Application des codes à effacements

12.2.3 Calcul distribué tolérant aux fautes sur plateforme P2P

La dernière contribution a consisté à utiliser des codes correcteurs pour construire un système de multiplication matricielle tolérant aux fautes. A partir de cette construction nous avons proposé une utilisation des codes LDPC et des codes Reed-Solomon pour effectuer des calculs tolérants aux fautes sur des plate-formes P2P.

Ces travaux ont été effectués en collaboration avec Thomas Roche et Jean-Louis Roch du Laboratoire d'Informatique de Grenoble (LIG).

Travaux futurs : Le cas des erreurs par valeurs sur un alphabet flottant ne peut pas être traité par l'emploi direct des codes LDPC. Le développement d'un décodeur LDPC travaillant sur un alphabet flottant et capable de corriger des erreurs par valeurs constitue une perspective intéressante à notre étude.

12.3 Le "mot de la fin"

Les travaux présentés dans cette thèse confirment le potentiel des codes LDPC pour la correction d'effacements, en particulier lorsqu'ils sont décodés avec une approche hybride

IT/ML. Nous avons montré tout d'abord que la complexité du décodage est parfaitement acceptable lorsque l'on considère des codes de longueur courte ou moyenne et que l'on utilise une implémentation optimisée du décodeur. Ce résultat va à l'encontre de la pensée communément admise selon laquelle le décodage ML des codes LDPC serait trop complexe pour être utilisé dans la plupart des applications pratiques. De plus, avec l'augmentation régulière de la puissance de calcul des processeurs, le décodage ML va devenir de plus en plus accessible et l'on pourra l'envisager pour des codes de plus en plus longs.

Nous avons ensuite montré que, malgré leur construction simple, les codes LDPC-staircase permettent d'atteindre d'excellentes performances. Ainsi les performances des codes LDPC-staircase approchent celles des codes Raptor dont la construction est autrement plus complexe (distributions irrégulières, codes concaténés, précodage LT-inverse pour encodage systématique, table de 65535 graines). Les différences entre les capacités de corrections de ces deux codes est souvent si faible que dans de nombreux cas d'application elle ne sera pas visible. Dans les cas où cette différence est invisible, nous pouvons nous poser la question de l'utilité de codes à construction complexe là où des codes à construction simple permettent d'obtenir les mêmes résultats.

Les codes LDPC peuvent servir de base à des constructions de codes plus élaborées. La faible densité de la matrice de parité étant la seule contrainte de ces codes, il est possible d'ajouter des contraintes supplémentaires sur la structure des matrices. Cette structure peut avoir deux conséquences bénéfiques : l'amélioration des capacités de correction et surtout la réduction de la complexité du décodage. Alors que les performances en termes de capacité de correction des codes LDPC sont proches de l'optimal, la question des performances en termes de vitesse de décodage constitue la suite logique dans l'amélioration de ces codes. A cet effet nous pensons que les codes structurés, tels que les codes LDPC-QC, associés à des décodeurs optimisés constituent une option de choix.

Parmi les adaptations possibles des algorithmes, on pourra penser aux possibilités offertes par la parallélisation. En effet les architectures multi-coeurs se démocratisent sur l'ensemble des terminaux. Le décodage sur des plateformes multi-coeurs peut être facilement envisagé en remarquant que celui-ci peut être divisé en deux parties : d'un coté le décodage logique sur la matrice de décodage (par exemple l'inversion de la matrice) et de l'autre les manipulations des données. Il est alors possible d'assigner un premier coeur au décodage logique tandis qu'un ou plusieurs autres coeurs s'occuperaient d'appliquer sur les données, les opérations issues du décodage logique. Il faut noter que le succès ou l'échec du décodage sera uniquement déterminé par le décodage logique. Ainsi, dans le cas d'un échec du décodage, la séparation du décodage logique et de la manipulation des données permettra d'économiser du temps de calcul.

On observe également une généralisation des unités de calcul GPU sur ces terminaux. Outre le caractère multi-coeurs des GPU, ils représentent souvent une puissance de calcul inexploitée. Il semble donc opportun de déporter les opérations liées aux codes correcteurs vers cette ressource, pour libérer de la puissance CPU au profit d'autres applications.

Dans un autre registre, considérons maintenant les systèmes effectuant de la correction d'effacements et remplissant une fonctionnalité cryptographique. Ce développement conjoint permet une réduction du coût algorithmique des opérations cryptographiques tout en conservant les capacités de correction de la fonction de correction d'effacements. La réduction du coût apporté par ce type d'approche ne peut qu'évoluer de manière favorable. En effet, la complexité algorithmique des codes correcteurs aura tendance à diminuer par le biais d'optimisations, ou dans le pire des cas restera identique. En revanche, pour faire face

aux avancées de la cryptanalyse et à l'augmentation de la puissance de calcul disponible, on assiste à une complexification des algorithmes cryptographiques. Ainsi des systèmes hybrides tels que ceux présentés ici devraient voir leur attractivité augmenter dans le futur.

Enfin, une application alternative des codes correcteurs a été présentée sous la forme d'un système de calcul distribué tolérant aux fautes. Ce champ d'application possède ses contraintes propres et ouvre la voie à de nouvelles recherches sur le thème des codes correcteurs. Plus généralement, on voit aujourd'hui une augmentation exponentielle des ressources (puissance de calcul, espace de stockage) offertes par une miniaturisation du matériel d'une part et la mutualisation des ressources d'autre part. Cependant cette augmentation des ressources disponibles va souvent de pair avec une augmentation du nombre d'erreurs/fautes (pannes de disque, erreurs de calculs, bugs, attaques malicieuses). La tolérance aux fautes devient alors un enjeu capital, auquel les codes correcteurs permettent de répondre de manière efficace.

Enfin, les algorithmes d'encodage et de décodage des codes à effacements sont eux aussi susceptibles de fonctionner sur de telles ressources non fiables. Il deviendra donc nécessaire de développer des décodeurs capables de tolérer des erreurs de calculs internes [90]. Pour conclure, nous pouvons dire que les codes correcteurs, et plus particulièrement les codes LDPC, sont ammenés à devenir des composants de plus en plus courant dans les systèmes manipulant de l'information, actuels et futurs.

Bibliographie

[1] Bittorrent, inc. http ://www.bittorrent.com.

[2] IEEE standards for local area networks : carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. *ANSI/IEEE Std 802.3-1985*, 1985.

[3] ETSI “IP Datacast over DVB-H : Content Delivery Protocols (CDP)”. Technical Report TS 102 472 v1.2.1, ETSI, 2006.

[4] 3GPP. Multimedia Broadcast/Multicast Service (MBMS) ; Protocols and codecs. TS 26.346, 3rd Generation Partnership Project (3GPP).

[5] Swarup Acharya, Michael Franklin, and Stanley Zdonik. Dissemination-based data delivery using broadcast disks. *IEEE Personal Communications*, 2 :50–60, 1995.

[6] D. P. Anderson. Boinc : a system for public-resource computing and storage. pages 4–10, 2004.

[7] Algirdas Avizienis, Jean-Claude Laprie, and Brian Randell. Dependability and its threats - a taxonomy. In *IFIP Congress Topical Sessions*, pages 91–120, 2004.

[8] E. Berlekamp. Nonbinary BCH decoding. *Information Theory, IEEE Transactions on*, 14(2) :242–242, Mar 1968.

[9] Elwyn R. Berlekamp. *Algebraic coding theory*. McGraw-Hill, 1968.

[10] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding : turbo-codes. *IEEE, Trans. Inform. Theory*, 44(10) :1261–1271, 1996.

[11] Richard E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley.

[12] M. Blaum and R.M. Roth. New array codes for multiple phased burst correction. *Information Theory, IEEE Transactions on*, 39(1) :66–77, Jan 1993.

[13] Johannes Blömer, Malik Kalfane, Richard Karp, Marek Karpinski, Michael Luby, and David Zuckerman. An XOR-Based Erasure-Resilient Coding Scheme, 1995.

[14] R. C. Bose and D. K. Ray Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1) :68–79, March 1960.

[15] George Bosilca, Remi Delmas, Jack Dongarra, and Julien Langou. Algorithmic based fault tolerance applied to high performance computing. *CoRR*, abs/0806.3121, 2008.

[16] A. Bouabdallah, M. Kieffer, J. Lacan, G. Sabeva, G. Scot, C. Bazile, and P. Duhamel. Evaluation of Cross - Layer Reliability Mechanisms for Satellite Digital Multimedia Broadcast. *IEEE Transactions on Broadcasting*, 53(1) :391–404, 2007. Rapport LAAS n° 07620.

[17] K. Bouchireb and P. Duhamel. Transmission schemes for scalable video streaming in a multicast environment. In *Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08. International*, pages 419–424, Aug. 2008.

[18] J. Boutros, O. Pothier, and G. Zemor. Generalized low density (Tanner) codes. *Communications, 1999. ICC'99. 1999 IEEE International Conference on*, 1, 1999.

[19] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking : an approach to interplanetary internet. *IEEE Communications Magazine*, 41(6) :128–136, 2003.

[20] David Burshtein and Gadi Miller. An efficient maximum-likelihood decoding of LDPC codes over the binary erasure channel. *IEEE Transactions on Information Theory*, 50(11) :2837–2844, 2004.

[21] John Byers, Gene Itkis, Mei Chin Cheng, and Alex Yeung. Securing bulk content almost for free. *Journal of Computer Communications*, pages 280–290, 2005.

[22] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege. A digital fountain approach to reliable distribution of bulk data. *SIGCOMM Comput. Commun. Rev.*, 28(4) :56–67, October 1998.

[23] A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code : application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *Information Theory, IEEE Transactions on*, 44(1) :367–378, Jan 1998.

[24] Pasquale Cataldi, Miquel Pedros Shatarski, Marco Grangetto, and Enrico Magli. Implementation and performance evaluation of LT and Raptor codes for multimedia applications. *Intelligent Information Hiding and Multimedia Signal Processing, International Conference on*, 0 :263–266, 2006.

[25] Peter Clements, Paul Nettle, and Ryan Gallagher. Parity volume set specification 2.0, May 11th 2003.

[26] M. Cluzeau and J.-P. Tillich. On the code reverse engineering problem. *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 634–638, July 2008.

[27] M. Cunche and V. Roca. Optimizing the error recovery capabilities of LDPC-staircase codes featuring a Gaussian Elimination decoding scheme. In *10th IEEE International Workshop on Signal Processing for Space Communications (SPSC'08), Rhodes Island, Greece*, October 2008.

[28] M. Cunche, V. Savin, V. Roca, G. Kraidy, A. Soro, and J. Lacan. Low-rate coding using incremental redundancy for GLDPC codes. In *Satellite and Space Communications, 2008. IWSSC 2008. IEEE International Workshop on*, pages 299–303, Oct. 2008.

[29] Mathieu Cunche and Vincent Roca. Adding integrity verification capabilities to the LDPC-staircase erasure correction codes. *Global Telecommunications Conference, 2009. GLOBECOM'09. IEEE*, 2009.

[30] G. I. David and J. A. Hansen. A preliminary exploration of striped hashing : a probabilistic scheme to speed up existing hash algorithms. Second International Conference on e-Business and Telecommunication Networks, October 2005.

[31] George I. Davida and Sudhakar M. Reddy. Forward-error correction with decision feedback. *Information and Control*, 21(2) :117 – 133, 1972.

[32] C. Di, D. Proietti, T. Richardson, E. Telatar, and R. Urbanke. Finite length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Transactions on Information Theory*, 48 :1570–1579, 2002.

[33] D. Divsalar, H. Jin, and R. J. Mceliece. Coding theorems for "turbo-like" codes. In *Proceedings Thirty-Sixth Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, 1998.

[34] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160 : A strengthened version of RIPEMD. *Fast Software Encryption, LNCS 1039, D. Gollmann, Ed., Springer-Verlag*, 1996.

[35] Jack Dongarra, Kaj Madsen, and Jerzy Wasniewski, editors. *Applied Parallel Computing, Computations in Physics, Chemistry and Engineering Science, Second International Workshop, PARA '95, Lyngby, Denmark, August 21-24, 1995, Proceedings*, volume 1041 of *Lecture Notes in Computer Science*. Springer, 1996.

[36] Jean-Guillaume Dumas and Gilles Villard. Computing the rank of large sparse matrices over finite fields. In *CASC'2002 Computer Algebra in Scientific Computing*, pages 22–27. Springer-Verlag, 2002.

[37] Alessandro Duminuco and Ernst Biersack. Hierarchical codes : How to make erasure codes attractive for peer-to-peer storage systems. *Peer-to-Peer Computing, IEEE International Conference on*, 0 :89–98, 2008.

[38] Alessandro Duminuco and Ernst Biersack. A practical study of regenerating codes for peer-to-peer backup systems. In *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, pages 376–384, June 2009.

[39] P. Elias. Coding for two noisy channels. In *Information Theory, The 3rd London Symposium*, pages 61–76. Buttersworth's Scientific Publications, September 1955.

[40] Marc P. C. Fossorier. Quasi-cyclic low-density parity-check codes from circulant permutation matrices. *IEEE Transactions on Information Theory*, 50(8) :1788–1793, 2004.

[41] Aurélien Francillon, Vincent Roca, Christoph Neumann, and Pascal Moniot. Secure transmission with error correcting code, September 2008.

[42] R. G. Gallager. *Low Density Parity Check Codes*. PhD thesis, MIT, Cambridge, Mass., September 1960.

[43] Danilo Gligoroski, Svein J. Knapskog, and Suzana Andova. Cryptcoding - encryption and error-correction coding in a single step. In Hamid R. Arabnia and Selim Aissi, editors, *Security and Management*, pages 145–151. CSREA Press, 2006.

[44] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[45] A Haken, M. Luby, G. Horn, J. Byers, J. Persch, and M. Mitzenmacher. On demand encoding with a window. US Patent 6486803, November 2002.

[46] S. Hirst and B. Honary. Application of efficient Chase algorithm in decoding of generalized low-density parity-check codes. *Communications Letters, IEEE*, 6(9) :385–387, 2002.

[47] D.E. Hocevar. A reduced complexity decoder architecture via layered decoding of ldpc codes. In *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, pages 107–112, Oct. 2004.

[48] Kuang-Hua Huang and J.A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Transactions on Computers*, 33(6) :518–528, 1984.

[49] Hideki Imai and Manabu Hagiwara. Error-correcting codes and cryptography. *Appl. Algebra Eng., Commun. Comput.*, 19(3) :213–228, 2008.

[50] Markus Jakobsson, Julien P. Stern, and Moti Yung. Scramble all, encrypt small. In *In Proc. of Fast Software Encryption*, pages 95–111, 1999.

[51] K. Kasai, T. Shibuya, and K. Sakaniwa. Detailed representation of irregular LDPC code ensembles and density evolution. *Information Theory, 2003. Proceedings. IEEE International Symposium on*, pages 121–, June-4 July 2003.

[52] R. Khalili and K. Salamatian. A new analytic approach to evaluation of packet error rate in wireless networks. In *Communication Networks and Services Research Conference, 2005. Proceedings of the 3rd Annual*, pages 333–338, May 2005.

[53] Yu Kou, Shu Lin, and Marc P. C. Fossorier. Low-density parity-check codes based on finite geometries : A rediscovery and new results. *IEEE Trans. Inform. Theory*, 47(7) :2711–2736, 2001.

[54] M. Krohn, M. Freedman, and D. Eres. In-the-fly verification of rateless erasure codes for efficient content distribution. In *IEEE Symposium on Security and Privacy*, May 2004.

[55] J. Lacan, V. Roca, J. Peltotalo, and S. Peltotalo. *Reed-Solomon Forward Error Correction (FEC) Schemes*, April 2009. IETF Request for Comments, RFC 5510.

[56] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3) :382–401, 1982.

[57] M. Lentmaier and K.S. Zigangirov. On generalized low-density parity-check codes based on Hamming component codes. *Communications Letters, IEEE*, 3(8) :248–250, 1999.

[58] M. Luby. LT codes. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 271–280, 2002.

[59] M. Luby and M. Mitzenmacher. Verification based decoding for packet based low-density parity check codes. *IEEE Trans. on Information Theory*, 50(1), January 2005.

[60] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer. Raptor Forward Error Correction Scheme for Object Delivery. RFC 5053 (Proposed Standard), October 2007.

[61] Michael Luby, Michael Mitzenmacher, Amin Shokrollahi, Daniel Spielman, and Volker Stemann. Practical loss-resilient codes. pages 150–159, 1997.

[62] Michael G. Luby, Michael Mitzenmacher, M. Amin Shokrollahi, M. Amin, Shokrollahi Daniel, and Daniel A. Spielman. Analysis of low density codes and improved designs using irregular graphs. In *In Proc. of ACM STOC*, pages 249–258, 1998.

[63] D. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, ISBN : 0521642981, 2003.

[64] David J.C. MacKay and Radford M. Neal. Near shannon limit performance of low density parity check codes. *Electronics Letters*, 32 :1645–1646, 1996.

[65] F. J. Macwilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes (North-Holland Mathematical Library)*. North Holland, January 1983.

[66] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15 :122–127, 1969.

[67] Mathieu Cunche and Vincent Roca. Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme. Research Report RR-6473, INRIA, 2008.

[68] Mathieu Cunche and Vincent Roca. Le RFC 5170 en pratique : conception et évaluation d'un codec AL-FEC LDPC-staircase hautes performances. In *CFIP'2009*, Strasbourg France, 10 2009.

[69] Chetan Nanjunda Mathur, Karthik Narayan, and K. P. Subbalakshmi. High diffusion cipher : Encryption and error correction in a single cryptographic primitive. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 309–324, 2006.

[70] M. Matyas, M. Peyravian, A. Roginsky, and N. Zunic. Reversible data mixing procedure for efficient public-key encryption.

[71] R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44 :114–116, January 1978.

[72] R. J. McEliece and D. V. Sarwate. On sharing secrets and Reed-Solomon codes. *Commun. ACM*, 24(9) :583–584, 1981.

[73] Nicholas Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247) :335–341, 1949.

[74] N. Miladinovic and M. Fossorier. Generalized LDPC codes with Reed-Solomon and BCH codes as component codes for binary channels. *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, 3.

[75] N. Miladinovic and M. Fossorier. Generalized LDPC codes and generalized stopping sets. *IEEE Trans. on Comm.*, 56(2) :201–212, 1974.

[76] G. Miller and D. Burshtein. Bounds on the maximum likelihood decoding error probability of low density parity check codes. In *Information Theory, 2000. Proceedings. IEEE International Symposium on*, pages 290–, 2000.

[77] National Institute for Science and Technology (NIST). Advanced Encryption Standard (FIPS PUB 197), November 2001. `http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[78] Christoph Neumann and Vincent Roca. Analysis of fec codes for partially reliable media broadcasting schemes. In *2nd ACM International Workshop on Multimedia Interactive Protocols and Systems (MIPS'04)*, 2004.

[79] Christoph Neumann, Vincent Roca, Aurélien Francillon, and David Furodet. Impacts of packet scheduling and packet loss distribution on fec performances : observations and recommendations. In *CoNEXT '05 : Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 166–176, New York, NY, USA, 2005. ACM.

[80] NIST. FIPS 180-2, Secure Hash Standard, Federal Information Processing Standard (FIPS), publication 180-2. Technical report, Departement of Commerce, August 2002.

[81] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh. FLUTE - File Delivery over Unidirectional Transport. RFC 3926, Internet Engineering Task Force, octobre 2004.

[82] E. Paolini, G. Liva, B. Matuz, and M. Chiani. Generalized IRA erasure correcting codes for hybrid iterative/maximum likelihood decoding. *Communications Letters, IEEE*, 12(6) :450–452, June 2008.

[83] E. Paolini, B. Matuz, G. Liva, and M. Chiani. Pivoting algorithms for maximum likelihood decoding of LDPC codes over erasure channels. *Global Telecommunications Conference, 2009. GLOBECOM'09. IEEE*, 2009.

[84] Enrico Paolini, Gianluigi Liva, Michela Varrella, Balazs Matuz, and Marco Chiani. Low-complexity ldpc codes with near-optimum performance over the bec, 2008.

[85] H. D. Pfister, I. Sason, and R. Urbanke. Capacity-Achieving Ensembles for the Binary Erasure Channel with Bounded Complexity. In *Fourth ETH–Technion Meeting on Information Theory and Communications*, 2004.

[86] H. Pishro-Nik and F. Fekri. On decoding of low-density parity-check codes over the binary erasure channel. *Information Theory, IEEE Transactions on*, 50(3) :439–454, March 2004.

[87] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. Technical Report CS-96-332, University of Tennessee, July 1996.

[88] J. Postel. User Datagram Protocol. RFC 0768, Internet Engineering Task Force, August 1980.

[89] J. Postel. Transmission Control Protocol. RFC 0793, Internet Engineering Task Force, September 1981.

[90] G. Robert Redinbo. Fault-tolerant decoders for cyclic error-correcting codes. *IEEE Trans. Comput.*, 36(1) :47–63, 1987.

[91] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2) :300–304, 1960.

[92] Y. Remion. Décomposition LU « incrémentale » d'une matrice régulière. Technical report, LERI, IUT Léonard de Vinci de Reims, 2004.

[93] T. J. Richardson and R. L. Urbanke. Efficient encoding of low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2) :638–656, 2001.

[94] Thomas J. Richardson, M. Amin Shokrollahi, and Rüdiger L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inform. Theory*, 47 :619–637, 2001.

[95] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. Technical Report MIT/LCS/TM-82, 1977.

[96] Ronald L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments, April 1992. RFC 1321.

[97] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2), April 1997.

[98] V. Roca and C. Neumann. Design, evaluation and comparison of four large block FEC codecs, LDPC, LDGM, LDGM staircase and LDGM triangle, plus a Reed-Solomon small block FEC codec. Research Report 5225, INRIA, June 2004.

[99] V. Roca, C. Neumann, and D. Furodet. *Low Density Parity Check (LDPC) Forward Error Correction*, June 2008. IETF RMT Working Group, Request for Comments, RFC 5170.

[100] Jean-Louis Roch and Sebastien Varrette. Probabilistic certification of divide & conquer algorithms on global computing platforms. Application to fault-tolerant exact matrix-vector product. In ACM publishing, editor, *Parallel Symbolic Computation'07 (PASCO'07)*, London, Ontario, Canada, July 2007.

[101] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11) :612–613, 1979.

[102] Claude E. Shannon. *A Mathematical Theory of Communication*. CSLI Publications, 1948.

[103] Amin Shokrollahi. Raptor codes. *IEEE/ACM Trans. Netw.*, 14(SI) :2551–2567, 2006.

[104] Lassen Soren Karp Richard Shokrollahi Amin M. Systems and processes for decoding a chain reaction code through inactivation, April 2006.

[105] R. Singleton. Maximum distance q -nary codes. *Information Theory, IEEE Transactions on*, 10(2) :116–118, Apr 1964.

[106] R. Smarandache and P.O. Vontobel. On regular quasicyclic LDPC codes from binomials. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, pages 274–, June-2 July 2004.

[107] Alexandre Soro, Mathieu Cunche, Jérôme Lacan, and Vincent Roca. Erasure codes with a banded structure for hybrid Iterative-ML decoding. *Global Telecommunications Conference, 2009. GLOBECOM'09. IEEE*, 2009.

[108] Alexandre Soro and Jérôme Lacan. FNT-based Reed-Solomon erasure codes. *CoRR*, abs/0907.1788, 2009.

[109] Jonathan Stone, Michael Greenwald, Craig Partridge, and James Hughes. Performance of checksums and CRC's over real data. *IEEE/ACM Trans. Netw.*, 6(5) :529–543, 1998.

[110] R. Storn and K. Price. Differential evolution : a simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4) :341–359, 1997.

[111] Chris Studholme and Ian Blake. Windowed erasure codes. *Information Theory, 2006 IEEE International Symposium on*, pages 509–513, July 2006.

[112] Chris Studholme and Ian Blake. Random matrices and codes for the erasure channel. *Algorithmica*, April 2008.

[113] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27(5) :533–547, 1981.

[114] R. Michael Tanner. On quasi-cyclic repeat-accumulate codes. In *in Proc. 37th Allerton Conf. Communication, Control and Computing*, pages 249–259, 1999.

[115] The OpenSSL Project. OpenSSL : The open source toolkit for SSL/TLS. `www.openssl.org`, April 2003.

[116] Mathieu Cunche Jean-Louis Roch Thomas Roche. Algorithm-based fault tolerance applied to p2p computing networks. In *The First International Conference on Advances in P2P Systems (AP2PS'09)*, 2009.

[117] R. Townsend and Jr. Weldon, E. Self-orthogonal quasi-cyclic codes. *Information Theory, IEEE Transactions on*, 13(2) :183–195, Apr 1967.

[118] M. Cunche C. Neumann V. Roca and J. Labouré. *An Open-Source LDPC Large Block FEC Codec*. URL : http ://planete-bcast.inrialpes.fr/.

[119] Antoine Valembois. Detection and recognition of a binary linear code. *Discrete Applied Mathematics*, 111(1-2) :199–218, 2001.

[120] Robert A. van de Geijn and Jerrell Watts. Summa : scalable universal matrix multiplication algorithm. *Concurrency - Practice and Experience*, 9(4) :255–274, 1997.

[121] Hakim Weatherspoon and John D. Kubiatowicz. Erasure coding vs. replication : A quantitative comparison. *Lecture Notes in Computer Science*, 2429 :328–338, 2002.

[122] S. B. Bhargava V. K. Wicker. *Reed-Solomon Codes and Their Applications*. John Wiley and Sons, 1994.

[123] V. V. Zyablov and M. S. Pinsker. Decoding complexity of low-density codes for transmission in a channel with erasures. *Probl. Peredachi Inf.*, 48 :18–28, 1974.

RÉSUMÉ

Nous assistons au développement rapide des solutions de diffusion de contenus sur des systèmes, où en plus des traditionnelles corruptions de l'information dans les couches basses, se pose le problème des pertes de paquets d'informations. Le besoin de fiabiliser ces systèmes de transmission a conduit à l'émergence de codes correcteurs d'effacements, qui grâce à l'ajout d'informations redondantes, permettent de reconstruire l'information perdue.

Dans cette thèse nous abordons le problème de la conception de codes à effacements ayant de bonnes capacités de correction et dont les algorithmes de décodage possèdent une complexité permettant d'atteindre des débits élevés. Pour cela, nous avons choisi de travailler conjointement sur les codes et sur leur implémentation au sein d'un codec logiciel, et plus particulièrement sur les algorithmes de décodage.

La première partie de nos travaux montre que des solutions basées sur les codes LDPC permettent d'obtenir d'excellents résultats. En particulier lorsque ces codes sont décodés avec un décodeur hybride IT/ML qui permet d'obtenir des capacités de corrections proches de l'optimal, tout en conservant une complexité acceptable. De plus, nous montrons que grâce à l'utilisation de codes LDPC structurés la complexité du décodage ML peut être largement réduite.

Nous étudions ensuite le développement de systèmes combinant un code à effacements et des fonctionnalités cryptographiques. Les systèmes résultants permettent de réduire la complexité globale du système tout en garantissant un niveau de sécurité élevé.

Finalement, nous présentons une technique de tolérance aux fautes basée sur des codes correcteurs pour des applications de multiplications matricielles. Cette technique nous permet de construire un système de calcul distribué sur plateforme P2P tolérant efficacement aussi bien les pannes franches que les erreurs malicieuses.

TITLE

Codes AL-FEC hautes performances pour les canaux à effacements : variations autour des codes LDPC

ABSTRACT

Abstract (english version).