AFF3CT : Un environnement de simulation pour le codage de canal

Adrien CASSAGNE[1,2], Mathieu LEONARDON[1], Olivier HARTMANN[1], Thibaud TONNELLIER[1], Guillaume DELBERGUE[1], Valentin GIRAUD[1], Camille LEROUX[1], Romain TAJAN[1], Bertrand LE GAL[1], Christophe JEGO[1], Olivier AUMAGE[2] et Denis BARTHOU[2]
[1]Laboratoire IMS, CNRS UMR 5218, Bordeaux INP, France
[1]INRIA / Labri, Université de Bordeaux, France

Abstract

Dans cet article nous présentons un environnement de simulation de Monte Carlo pour les systèmes de communications numériques. Nous nous focalisons en particulier sur les fonctions associées au codage de canal. Après avoir présenté les enjeux liés à la simulation, nous identifions trois problèmes inhérents à ce type de simulation. Puis nous présentons les principales caractéristiques de l'environnement AFF3CT.

1 Les enjeux de la simulation

Malgré l'immense variété des systèmes de communications qui nous entourent, chacun d'entre eux repose sur un modèle abstrait unique imaginé par le génial inventeur de la théorie de l'information, Claude Shannon. En effet, dans son article de 1948 [5], parmi plusieurs contributions originales, Claude Shannon propose de modéliser un système de communication à l'aide de 5 blocs élémentaires : une source d'information, un émetteur, un canal, un récepteur et une destination. Ce modèle fût ensuite largement adopté puis complété avec d'autres blocs comme montré sur la Figure 1. Dans ce modèle de chaîne de communication, les blocs de codage et de décodage de canal sont critiques car ils déterminent le taux d'erreur que l'on peut espérer atteindre. De plus, le décodeur est bien souvent un des principaux contributeurs à la complexité calculatoire du système complet. Malheureusement, la plupart du temps, il n'existe pas de modèle mathématique qui permette de prédire les performances de décodage d'un couple codeur/décodeur. La seule solution simple est d'effectuer des simulations de Monte Carlo de l'ensemble de la chaîne de communication : des données sont générées de manière aléatoire, encodées, modulées, bruitées, décodées et les performances sont estimées en mesurant le taux d'erreur binaire et le taux d'erreurs trame après l'étape de décodage. Bien que présentant l'avantage d'être universel, la simulation de Monte Carlo pose néanmoins trois problèmes majeurs pour les concepteurs

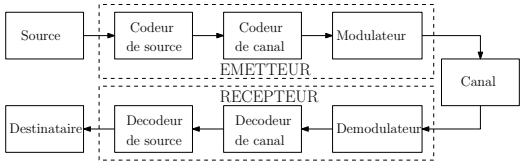

FIGURE **1. Chaine de communications.**

de systèmes de communications numériques.

Reproductibilité : Reproduire les performances des systèmes proposés dans la littérature est bien souvent une tâche fastidieuse et hasardeuse. Cela concerne notamment le décodeur qui fait largement appel à des paramètres déterminés de manière empirique (format des données manipulées, heuristiques de calculs, séquencement des opérations, etc...). Ces ajustements sont très rarement rapportés de manières exhaustives dans les publications scientifiques. Il est de plus très rare que les auteurs partagent le code source de leur simulateur. Ainsi, une grande partie du temps est consacrée à "réinventer la roue" uniquement pour être en mesure de se comparer de manière fiable avec les travaux passés.

Le temps de simulation : Afin d'estimer précisement un taux d'erreurs, il faut observer une centaine de trames erronées. Cela signifie qu'il est nécessaire de simuler la transmission de $\sim 100 \times 10^7 = 10^9$ trames. En gardant à l'esprit la complexité calculatoire non-négligeable des algorithmes de décodage, il n'est pas rare que plusieurs semaines voire plusieurs mois soient nécessaires pour obtenir une estimation des performances d'un système de communications numériques.

Hétérogénéité algorithmique : Comme indiqué précédemment, pour chaque schéma de codage, il existe plusieurs algorithmes de décodage. S'il est plutôt facile de décrire un schéma de codage unique, il en est tout autrement de construire un simulateur qui supporte différents schémas de codage ainsi que les algorithmes de décodage associés. Cette difficulté vient de l'hétérogénéité de structure des codes et des décodeurs : les turbo codes [3] reposent sur une représentation en treillis, les codes LDPC [4] sont basés

sur des factor graph quand les codes polaires [2] sont décodés de manière efficaces sur des arbres binaires.

Le defis de la reproductibilité pousse vers une solution *portable* (multi-OS) et *Open-source*. De plsu, les temps de simulation trop long suggèrent de disposer d'un outil *rapide*. Enfin, l'hétérogénéité algorithmique impose une certaine *flexibilité*. Ces différents constats nous ont ainsi amenés au développement d'AFF3CT : A Fast Forward 3rror Correction Tool [1].

2 AFF3CT

L'ambition d'AFF3CT est de fournir à la communauté scientifique du codage de canal et des systèmes de communications numériques au sens large un outil de simulation open-source, portable rapide et flexible. Celui-ci doit permettre aux chercheurs et aux ingénieurs du domaine de reproduire facilement les résultats de la littérature et de passer plus de temps sur les problèmes de codage de canal ou d'algorithmiques que sur des difficultés de développement logiciel.

L'outil AFF3CT est écrit en C++11 avec une approche modulaire de telle manière que l'on puisse facilement ajouter de nouveaux blocs. L'approche objet, portée par le C++, permet également de spécialiser les blocs de la chaîne en fonction du scenario que l'on souhaite valider. Il est possible de simuler une chaîne similaire à celle de la Figure 1 De plus, il est possible de simuler une chaîne incluant un récepteur itératif dans lequel le démodulateur et le décodeur échangent des informations de manière itérative. Il est tout à fait possible d'imaginer d'autres types de simulations comme par exemple la génération d'EXIT chart ou bien encore l'analyse de décodeurs par évolution de densité.

Flexibilité : AFF3CT peut être utilisé pour la simulation des performances de décodage d'un schéma de codage. Il facilite le test de nouveaux codes ou algorithmes de décodage. On peut également l'utiliser pour expérimenter des optimisations algorithmiques en vue d'une implantation matérielle ou logicielle. Dans tous les cas, il est nécessaire de pouvoir aisément changer de type de code, de modulation, de type de canal, etc... A ce jour, AFF3CT supporte déjà un grand nombre de configurations pour chacun des blocs constituant la chaîne. Du point de vue du codage de canal, AFF3CT supporte la plupart des codes intégrés dans les standards : les turbo codes, les codes LDPC, les codes polaires ainsi que les codes BCH et les codes convolutifs. Pour aller encore plus loin dans la flexibilité, AFF3CT peut également être compilé et utilisé comme une bibliothèque externe. Dans ce cas, l'utilisateur peut créer lui même sa simulation en instanciant uniquement les blocs dont il a besoin. Dans l'optique d'une implantation matérielle d'un bloc de la chaîne, il est également possible de faire du "Hardware in the loop". Cela signifie que l'utilisateur peut échanger des trames avec un circuit FPGA via une interface UART. Une fois les traitement effectués sur le FPGA, ce dernier renvoie les données traitées à AFF3CT via la même interface. Cela permet de faciliter la vérification du fonctionnement d'un bloc matériel implanté sur un circuit FPGA.

Rapidité : La flexibilité s'obtient souvent au prix d'une vitesse de simulation amoindrie. Dans le cas d'AFF3CT, nous utilisons massivement les techniques de parallélisation afin de profiter au mieux des ressources de calculs présentes sur les processeurs actuels. Les blocs les plus critiques en terme de complexité calculatoire utilisent des instructions SIMD. De plus, l'ensemble des simulations peuvent s'exécuter sur plusieurs "threads" à la fois. Enfin, il est possible de lancer AFF3CT sur plusieurs noeuds de calculs au sein d'un supercalculateur. Les simulations de Monte Carlo étant massivement parallèle par essence, l'accélération observée est quasiment égale au nombre de noeuds utilisés. Nous avons ainsi pu simuler un turbo décodeur s'éxecutant sur 50 noeuds à un débit de 50 Gb/s. Certains décodeurs sont d'ailleurs parmi les plus rapides de la littérature à ce jour[1].

Portabilité et libre accès : AFF3CT a été conçu dans un esprit d'ouverture au sein d'une communauté scientifique qui peine parfois à partager l'accès à ses codes sources. Ainsi AFF3CT est sous une license libre de type MIT. Cela signifie que n'importe qui peut le télécharger [1], le modifier et même l'utiliser à des fins commerciales. L'idée ici est que tout le monde puisse y avoir accès afin de pouvoir y contribuer librement qu'importe le contexte. L'outil a déjà été téléchargé par des équipes à travers le monde.

Dans une optique de portabilité, il nous a semblé opportun de supporter différentes architectures de processeurs (x86, ARM, Xeon Phi) ainsi que les trois systèmes d'exploitations les plus répandus (Windows, MacOS et Linux). Enfin AFF3CT peut être compilé avec différentes suites de compilation : gcc, icc, clang, et msvc.

Ce travail a été supporté par l'ANR NAND (ANR-15-CE25-0006-01).

Références

[1] AFF3CT : A Fast Forward 3rror Correction Tool. http ://aff3ct.github.io.

[2] E. Arikan. Channel polarization : A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7) :3051–3073, 2009.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding : Turbo-codes. 1. In *IEEE ICC'93 Geneva.*, volume 2, pages 1064–1070. IEEE, 1993.

[4] R. Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1) :21–28, 1962.

[5] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 1948.