



Rapport de synthèse
Surveillance réseau sur NetFPGA
Projet 3A



Rédacteurs :

Benoit Fontaine
Tristan Groléat
Franziska Hubert

étudiant 3A, filière INFO
étudiant 3A, filière INFO
étudiant 3A, filière ELEC

Encadrants :

Matthieu Arzel
Amer Baghdadi
Sandrine Vaton
Sylvain Gombault

enseignant-chercheur, département ELEC
enseignant-chercheur, département ELEC
enseignant-chercheur, département INFO
enseignant-chercheur, département RSM

Octobre 2009 – Mars 2010

Table des matières

Introduction	2
1 Organisation	5
1.1 Ressources	5
1.2 Trac	7
1.3 Réunions et division des tâches	8
2 Mise en œuvre	9
2.1 L'architecture de notre solution	9
2.1.1 Le rôle du NetFPGA	9
2.1.2 La séparation logiciel/matériel	9
2.2 L'algorithme	10
2.3 Le développement dans l'environnement NetFPGA	11
2.3.1 Contexte	11
2.3.2 La transformation du NetFPGA en hub Ethernet	11
2.3.3 L'insertion d'un module de surveillance du trafic	11
2.4 La communication entre l'ordinateur et le NetFPGA	12
Conclusion	13
Bibliographie	15

Introduction

Les volumes de trafic en cœur de réseau ne cessent de croître à une vitesse exponentielle. En effet un nombre croissant d'applications requièrent des débits de plus en plus importants. On peut citer le peer-to-peer ou le streaming vidéo (Youtube, Dailymotion) qui permet maintenant de regarder des vidéos en haute définition (720p essentiellement). De plus le déploiement actuel de la fibre optique jusqu'à l'utilisateur terminal ne va pas calmer le rythme car il va rendre possible l'utilisation de nouveaux services encore plus gourmands en bande passante.

Les débits actuellement utilisés sont 10 Mb/s, 100 Mb/s (Fast Ethernet), 1 Gb/s (Gigabit Ethernet) et 10 Gb/s. Cependant le groupe de travail IEEE P802.3ba [2] travaille depuis Novembre 2007 à l'implémentation des normes 40 Gb/s et 100 Gb/s Ethernet. Cette augmentation de débit présente un enjeu supplémentaire pour la surveillance de trafic dont l'objectif est la détection des anomalies dans le trafic réseau comme les dénis de service, les scans ou les virus.

La surveillance doit ainsi s'adapter aux nouvelles vitesses des réseaux, mais aussi faire face à la diversification des menaces de sécurité. Ces deux facteurs doivent être considérés lors du développement de nouvelles techniques au niveau software, hardware, mais aussi au niveau des protocoles pour adapter la surveillance trafic aux nouveaux besoins.

Dans le cadre du projet de 3ème année à Télécom Bretagne d'une durée de 4 mois (de novembre 2009 à mars 2010), nous nous sommes intéressés à la réalisation d'algorithmes de surveillance de trafic utilisant l'accélération matérielle de cartes dédiées. Pour ce projet qui réunit les départements informatique, électronique et réseaux de l'école, nous disposons d'une carte NetFPGA. Il s'agit d'une carte intégrant un FPGA¹ connecté à 4 ports Ethernet, permettant de l'intégrer facilement dans un réseau. Notre but était de réaliser sur cette carte un algorithme de surveillance de trafic. Nous avons choisi de détecter les attaques par SYN Flooding. Il s'agit d'attaques qui tirent partie d'une faiblesse du protocole TCP et qui consistent à envoyer de très nombreux paquets TCP d'un type spécial (SYN), à un serveur pour le surcharger et le rendre inutilisable.

¹Field Programmable Gate Array : circuit intégré programmable en Verilog ou VHDL, langages décrivant la manipulation des bits par des portes logiques

Les objectifs du projet étaient multiples :

Prendre en main le NetFPGA C'est-à-dire installer l'environnement de développement.

Analyser et utiliser l'architecture du NetFPGA C'est-à-dire être capable d'intégrer notre projet dans cette architecture.

Implémenter un algorithme de surveillance de trafic C'est-à-dire réaliser effectivement l'algorithme, permettant de se rendre compte des gains apportés par l'accélération matérielle aux performances.

Les résultats de notre projet pourront servir ensuite dans le cadre plus large d'un projet Européen auquel participe Télécom Bretagne par le biais de l'institut Télécom : le projet DEMONS (DEcentralized, cooperative, and privacy-preserving MONitoring for trustworthinesS) qui commence en 2010 pour 3 ans, et auquel participe plus particulièrement Sandrine Vaton. Ce projet traite de la surveillance du trafic, et insiste sur la nécessité d'utiliser l'accélération matérielle. Le NetFPGA pourra donc servir dans ce cadre à Télécom Bretagne, et l'algorithme de détection d'attaques par SYN flooding pourra être un bon point de départ.

Chapitre 1

Organisation

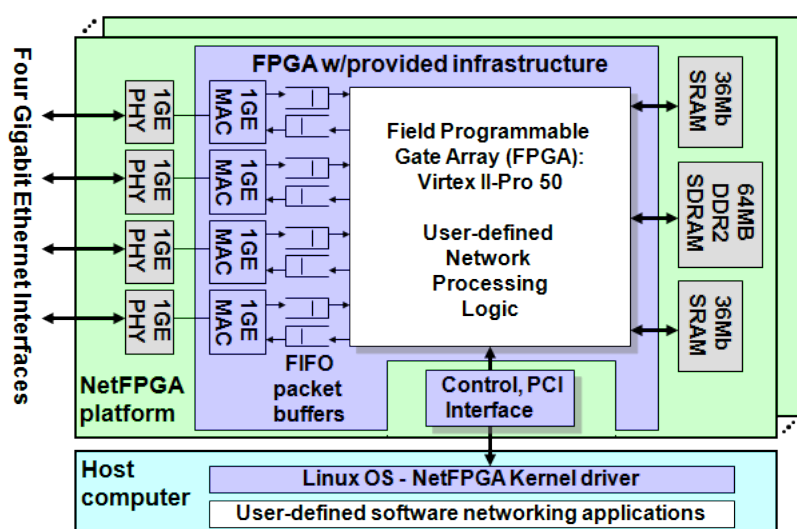
1.1 Ressources

Ressources humaines

Ce projet possède l'originalité de s'étaler sur trois filières : ELEC, INFO et RSM. Deux élèves de filière INFO et une élève de filière ELEC travaillent sur le projet. Deux encadrants sont de filière ELEC, un de filière INFO et un de filière RSM.

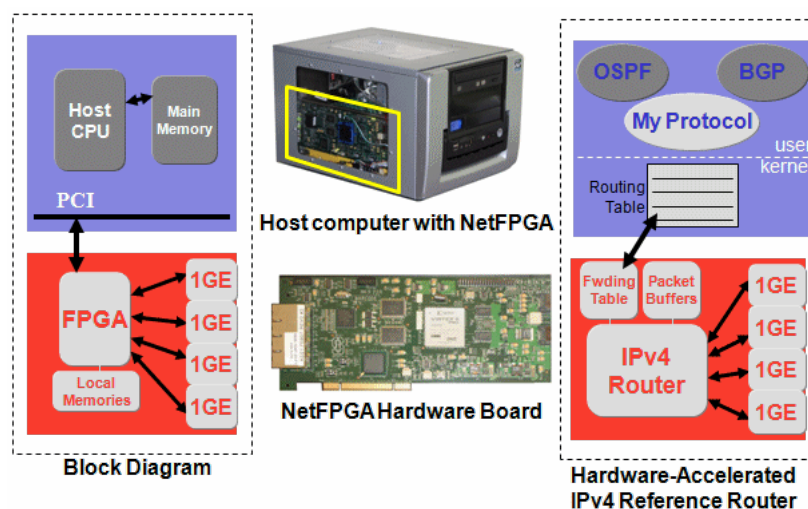
Ressources matérielles

L'élément principal de notre projet est le NetFPGA. Le NetFPGA est une carte PCI dotée d'un FPGA Xilinx Virtex2-Pro 50, de 4 ports Gigabit Ethernet et d'une mémoire embarquée.



Il a été conçu à l'université de Stanford à Palo Alto, CA et permet aux étudiants et chercheurs de réaliser des prototypes de systèmes réseau à haute vitesse en bénéficiant d'une accélération matérielle. Les étudiants de Stanford ont déjà travaillé à réaliser un routeur IP à l'aide du NetFPGA. Une présentation de leurs travaux est disponible sur le site officiel [6].

La carte NetFPGA possède une interface PCI standard et peut donc être connectée à un ordinateur de bureau ou un serveur. Le NetFPGA permet de décharger le processeur de la machine hôte du traitement des données. Le CPU de la machine hôte a accès à la mémoire principale et peut effectuer des opérations DMA de lecture et d'écriture des registres et mémoires du NetFPGA. Contrairement aux autres projets open-source le NetFPGA offre une approche matériellement accélérée. Le NetFPGA fournit une interface matérielle directement connectée aux quatre ports Gigabit et à de multiples banques de mémoire installées sur la carte.



Le département électronique nous a mis à disposition un de ses laboratoires (au bâtiment K2) ainsi qu'une station de travail dotée d'une interface réseau, de la carte NetFPGA et d'une distribution Linux Ubuntu Karmic 9.10. Cette station est connectée au réseau Salsa-Lite de l'école. Il est à noter que ceci impose de taper sur le portail captif son identifiant et mot de passe école à chaque fois que l'on est inactif plus de quelques dizaines de minutes sur Internet.

Nous avons très vite eu besoin d'une nouvelle carte réseau afin de tester le NetFPGA sans devoir changer à chaque fois les câbles et perdre du même coup notre connexion vers le Net.

Nous avons découvert par la suite que le NetFPGA ne fonctionne qu'avec des interfaces 1000BaseT (Gigabit) et pas avec les interfaces classiques 100BaseT. Or nos ordinateurs portables ne supportent que le 100BaseT. Nous avons donc emprunté une seconde station de travail disponible dans le laboratoire qui dispose elle d'une interface Gigabit. Nous pouvons ainsi simuler un réseau de deux ordinateurs avec le NetFPGA situé en coupure pour l'analyse des communications.

La première étape dans le déroulement du projet fut de faire fonctionner le NetFPGA sur les ressources qui nous ont été attribuées. En effet l'environnement supporté officiellement par celui-ci est différent du notre. Nous disposons d'une Ubuntu récente alors que le site du NetFPGA recommandait une CentOS plus ancienne. Il a donc fallu effectuer des modifications dans le code de la partie software du NetFPGA pour le faire fonctionner sur un noyau et des bibliothèques plus récentes. Cette phase était essentielle sans quoi le projet n'aurait jamais pu continuer.

1.2 Trac

Afin de rendre la gestion du projet plus aisée, nous utilisons le moteur **Trac** couplé au système de gestion de versions **Subversion**. Trac est un système Open Source de gestion complète de projet par Internet.



Il inclut :

Un wiki Nous écrivons l'ensemble de la documentation du projet dans ce wiki. Dès que nous découvrons ou mettons en place quelque chose de nouveau, un article y est tout de suite consacré. Ainsi des pages sont consacrées aux outils que nous utilisons, d'autres aux algorithmes de surveillance, d'autres à l'utilisation et au développement sur NetFPGA et enfin une partie est dédiée à l'organisation avec un planning des événements.

Subversion Subversion est un logiciel de gestion de versions. Nous gérons avec cet outil le code du NetFPGA ainsi que les rendus écrits en \LaTeX (comme celui-ci). Cela nous permet de travailler simultanément sur les mêmes fichiers et de garder un historique de nos modifications avec la possibilité d'y associer des commentaires (les messages de "commit").

Un système de tickets Les tickets sont très pratiques pour communiquer et se coordonner sur une tâche à accomplir ou un problème à résoudre. Il est possible de laisser des commentaires, ainsi les discussions ne sont pas perdues mais stockées dans Trac et disponibles pour de futurs contributeurs au projet. De cette manière on peut vite voir si un problème auquel on est confronté n'a pas déjà été résolu. Chaque personne associée au ticket est notifiée par email des actions prises sur celui-ci.

Une gestion des feuilles de route Les feuilles de route permettent de s'organiser sur les échéances. Une feuille de route correspond en fait à une échéance. On lui donne un intitulé, une description, une date de fin puis on y associe un certain nombre de tickets. Au fur et à mesure que les tickets sont fermés, la barre de progression de la feuille de route se met à jour et nous montre le chemin qui nous sépare encore de sa réalisation totale. Par exemple une feuille de route a été créée pour la réalisation de ce rapport de synthèse et cinq tickets y ont été associés.

Un historique La page d'historique est très importante car elle permet de suivre l'évolution du projet en un simple coup d'œil. Cette page regroupe les dernières modifications de tickets, de feuilles de route, de commits subversion et du wiki. Il est même possible d'accéder à

son contenu au format RSS afin d'être en permanence au courant de l'évolution du projet grâce à son lecteur de flux RSS préféré.

Trac [5] nous permet donc de travailler de façon collaborative et de mutualiser nos contributions au projet. C'est un outil essentiel à tout projet comptant s'inscrire dans le temps. La page web de notre Trac est disponible à l'adresse suivante : <http://trac.benoute.fr/netfpga> [3].

1.3 Réunions et division des tâches

Les réunions sont un élément important non seulement pour s'organiser mais aussi pour produire et partager les connaissances acquises. Nous organisons plusieurs réunions chaque semaine. Il est possible de répartir ces réunions en trois catégories :

Réunions avec les encadrants Nous avons décidé en début de projet d'effectuer une réunion chaque lundi avec nos encadrants. C'est le moment de leur faire part de notre progression mais aussi de nos problèmes.

Réunions de travail Nous nous réunissons au moins deux fois par semaine dans notre laboratoire de projet. Ces réunions sont en fait des sessions de travail. Elles durent en général un après-midi voire une journée entière. C'est durant ces sessions que se fait le développement du projet. Soit nous nous repartissons des tâches et nous travaillons en parallèle, soit nous travaillons à plusieurs sur la même tâche.

Réunions d'organisation Ces réunions ont lieu lorsqu'il est nécessaire de s'organiser sur une échéance. Aucun développement n'est réalisé, seule l'organisation est traitée. Ces réunions sont plus courtes et durent de quelques dizaines de minutes à une heure.

Chapitre 2

Mise en œuvre

2.1 L'architecture de notre solution

2.1.1 Le rôle du NetFPGA

Nous souhaitons faire de la surveillance de trafic sur un lien à très haut débit en utilisant l'accélération matérielle fournie par le NetFPGA. Pour cela, nous avons mis le NetFPGA en coupure du lien à surveiller de manière à ce que tous les paquets qui passent par ce lien traversent le NetFPGA. Nous utilisons donc deux des quatre interfaces Ethernet que possède le NetFPGA.

Ceci signifie que le rôle premier du NetFPGA doit être de transmettre tous les paquets qu'il reçoit sur une interface à l'autre interface. Pour simplifier l'utilisation et le test de notre solution, nous avons décidé de ne pas imposer les deux ports à utiliser sur le NetFPGA. Il faut donc que le NetFPGA copie tous les paquets qu'il reçoit sur n'importe quelle interface sur toutes les autres interfaces.

Ce comportement est celui d'un élément bien connu en réseau : le hub Ethernet. Notre première tâche au niveau du développement a donc été de transformer le NetFPGA en simple hub Ethernet. Nous aurions pu compliquer son comportement en lui faisant jouer le rôle de switch Ethernet, qui apprend à qui il est connecté, ou même celui de routeur IP. Mais cela aurait été peu utile car nous ne voulons utiliser que deux interfaces, et cela aurait compliqué la configuration et le test du NetFPGA.

2.1.2 La séparation logiciel/matériel

Maintenant que nous avons décidé que le NetFPGA agirait comme un hub Ethernet, le problème suivant est de savoir comment la surveillance du trafic doit être faite. Il y a trois grandes étapes dans cette surveillance :

1. la détection des paquets qui nous intéressent, ici les paquets TCP SYN ;
2. le comptage de ces paquets ;
3. l'analyse des résultats, et le possible déclenchement d'alertes.

Si nous ne faisons pas la première étape sur le NetFPGA, nous sommes obligés de renvoyer tous les paquets reçus à l'ordinateur. Or la communication avec l'ordinateur est assez lente. La

conséquence serait donc une réduction importante du débit maximal auquel le NetFPGA peut fonctionner. Pour utiliser les capacités d'accélération matérielle du NetFPGA, nous ferons donc la première étape dans le NetFPGA.

La seconde étape n'est déclenchée qu'à la réception d'un paquet qui nous intéresse, ici un TCP SYN. Et la seule information qui nous intéresse est l'adresse IP source du paquet. Il serait donc envisageable d'envoyer l'adresse IP source à la réception de chaque paquet TCP SYN à l'ordinateur pour qu'il le traite. Mais ceci ne fonctionnerait que si nous recevons suffisamment peu de paquets SYN. Or le but de notre projet est de détecter les attaques qui se font par envois massifs de paquets TCP SYN. Il est donc indispensable de pouvoir garantir le fonctionnement de notre solution avec une grande quantité de paquets TCP SYN reçus. C'est pourquoi nous effectuerons aussi la seconde étape sur le NetFPGA.

La troisième étape en revanche est déclenchée quand nous le souhaitons. La réaction immédiate aux attaques de SYN flooding n'entre en effet pas dans le cadre de notre projet : nous souhaitons seulement détecter ces attaques. C'est pourquoi nous pouvons n'analyser les résultats que périodiquement, à un intervalle fixe. Cette analyse sera faite par l'ordinateur qui pourra utiliser sa grande puissance de calcul pour faire des analyses complexes, générer des statistiques...

En résumé, nous effectuerons la détection et le comptage sur le NetFPGA, et l'ordinateur accèdera périodiquement aux données du NetFPGA pour les analyser.

2.2 L'algorithme

La détection des paquets qui nous intéressent consiste simplement à lire les bons champs des paquets reçus, nous n'avons donc pas besoin d'algorithmes particulier pour cela. En revanche le comptage est plus complexe : nous ne pouvons en effet pas mettre en place un compteur dans la mémoire du NetFPGA pour chaque adresse IP qui nous envoie des paquets TCP SYN. Pour économiser de la mémoire, nous avons décidé d'utiliser un algorithme de "data mining" : ces méthodes nous permettent de stocker beaucoup moins de données, et de pouvoir en retirer des informations avec une incertitude et une probabilité connues.

Cet algorithme devra être implémenté sur le NetFPGA. Nous avons donc choisi un algorithme simple : le Count Min Sketch (CMS). Cet algorithme fonctionne en calculant des haschs des adresses IPs qui nous envoient des paquets TCP SYN, et en remplissant un tableau en mémoire à partir de ces haschs.

Une implémentation de cet algorithme existe déjà en C [4]. Nous devons en faire une implémentation matérielle sur le NetFPGA. La difficulté principale de cette conversion est que l'algorithme utilise des fonctions de génération de nombres aléatoires, qui sont difficiles à réaliser en matériel. Heureusement, ces fonctions ne sont utilisées qu'à l'initialisation de l'algorithme pour générer des fonctions de hachage aléatoirement. Nous générerons donc ces nombres aléatoires lors du développement sur l'ordinateur, puis nous les chargerons dans une mémoire morte (ROM) sur le NetFPGA, qui les utilisera ensuite dans l'algorithme.

2.3 Le développement dans l’environnement NetFPGA

2.3.1 Contexte

Le NetFPGA est programmé à partir du langage Verilog. Il s’agit d’un langage qui spécifie précisément la manipulation des signaux binaires. Il existe un programme Verilog appelé “routeur de référence”, fourni avec le NetFPGA, qui transforme le NetFPGA en routeur. C’est cette référence que les projets comme le notre utilisent comme base.

Le parcours d’un paquet Ethernet dans le “routeur de référence” est divisé en modules. Chaque module reçoit le paquet, fait son traitement puis renvoie le paquet, éventuellement modifié, au module suivant. Si les modules ont besoin de se communiquer des données extérieures au paquet, ils peuvent ajouter ou modifier des en-têtes spécifiques au NetFPGA au début du paquet. [1]

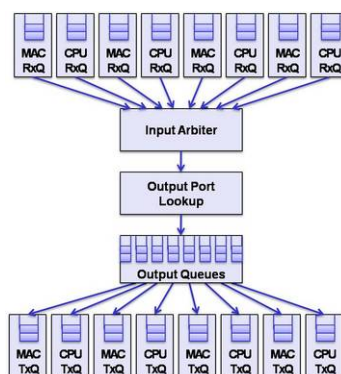


Fig. 2.1 – Modules dans le routeur de référence, selon netfpga.org

Le premier module reçoit les trames Ethernet, les découpe, et leur ajoute une en-tête disant d’où ils viennent. Le module suivant, appelé “output_port_lookup”, modifie l’en-tête pour indiquer par quelles interfaces le paquet doit ressortir. Les modules se succèdent ainsi jusqu’au dernier qui renvoie la paquet sur les bonnes interfaces.

2.3.2 La transformation du NetFPGA en hub Ethernet

Pour transformer le NetFPGA en hub Ethernet, nous sommes donc partis du “routeur de référence”. Ce routeur est contrôlé et configuré par l’ordinateur. Nous avons supprimé toute la partie de contrôle sur l’ordinateur, puis nous avons modifié le module “output_port_lookup”, qui décide vers quelle interface un paquet doit être envoyé. Nous avons effectué ces modifications guidés par un tutoriel pour apprendre à manipuler le NetFPGA trouvé sur Internet.

2.3.3 L’insertion d’un module de surveillance du trafic

Une fois que nous avons un hub Ethernet fonctionnel, il nous a fallu ajouter la fonction de surveillance du trafic. Pour cela, nous avons ajouté un module appelé “network_monitoring” juste après le module “output_port_lookup”. Ce module laisse les paquets le traverser sans les modifier.

Ce module lit les champs qui l'intéressent dans les paquets qu'il reçoit en même temps qu'ils passent. Il ne limite donc pas le débit du trafic reçu. Il est cependant nécessaire qu'il puisse prendre le temps d'enregistrer l'arrivée d'un paquet TCP SYN quand il en détecte un. Pour cela, nous utilisons une file d'attente, qui est ouverte tout le temps sauf quand le module est en train d'enregistrer l'arrivée d'un paquet TCP SYN (application de l'algorithme CMS).

2.4 La communication entre l'ordinateur et le NetFPGA

Le dernier problème qui se pose pour réaliser notre surveillance de trafic est la lecture par l'ordinateur des registres que le module "network_monitoring" du NetFPGA utilise pour enregistrer ses données. Cette lecture est prévue dans l'architecture globale du NetFPGA grâce à un bus de données qui forme un anneau en passant par tous les modules du NetFPGA.

Un module Verilog appelé "generic_register" existe que nous pouvons intégrer dans notre module "network_monitoring". Ce sous-module se connecte au bus en anneau qui passe par notre module. Il permet ensuite d'accéder depuis l'intérieur du module à des registres et de compteurs qui seront accessibles également depuis l'ordinateur. Il existe trois types de registres dans "generic_register" : des registres "hardware" écrits par le module et lus par l'ordinateur, des registres "software" écrits par l'ordinateur et lus par le module, et des compteurs incrémentés par le module et lus par l'ordinateur.

Nous avons choisi des compteurs car le module compte les paquets TCP SYN et l'ordinateur lit les résultats.

Au niveau de l'ordinateur, la lecture s'effectue grâce à des pilotes fournis par NetFPGA qui font passer l'interface de connexion avec le NetFPGA pour une interface réseau. Elle est facilitée par des bibliothèques fournies en C pour lire/écrire des registres à une adresse donnée. Les adresses sont fournies par un fichier d'en-têtes C généré automatiquement lors de la synthèse de notre module. Nous avons développé un petit programme en C permettant de récupérer les données du NetFPGA.

Conclusion

Les objectifs de ce projet étaient multiples :

Prendre en main le NetFPGA Nous avons mis en place l'environnement de développement du NetFPGA pour la première fois à TELECOM Bretagne. Les étapes que nous avons suivies pour le mettre en place sont consultables sur le wiki et le SVN du projet. La machine installée pourra aussi être utilisée pour d'autres projets sur le NetFPGA.

Analyser et utiliser l'architecture du NetFPGA Nous avons documenté dans le wiki du projet la méthode qui permet d'insérer son propre module dans l'architecture du NetFPGA, et d'utiliser ses fonctionnalités : lecture des paquets reçus, communication avec l'ordinateur... Ces informations pourront servir à tous les projets futurs sur NetFPGA à Télécom Bretagne.

Implémenter un algorithme de surveillance de trafic Nous avons développé un algorithme de détection d'attaques TCP SYN utilisant l'accélération matérielle fournie par le NetFPGA. Les performances de cet algorithme peuvent être comparées à celles des implémentations purement logicielles. Ce code peut aussi servir de base à l'implémentation d'algorithmes plus sophistiqués.

Au moment de l'écriture de ce rapport, notre projet n'est pas tout à fait terminé. Nous avons développé la détection et un comptage basique des paquets TCP SYN qui passent par le NetFPGA : un simple compteur enregistre le nombre de paquets TCP SYN qui passent par le NetFPGA, sans se soucier de l'adresse IP source. Ce compteur est ensuite lu par l'ordinateur grâce à un programme en C basique qui affiche la valeur à l'écran.

Nous n'avons pas encore implémenté l'algorithme CMS pour rendre le comptage efficace. Nous avons cependant déjà étudié les difficultés posées par cet algorithme et pensons pouvoir l'implémenter sans problèmes avant la fin du projet. Nous développerons aussi un programme un peu plus sophistiqué (toujours en C, ou en Python, les deux langages supportés simplement par NetFPGA) pour permettre à l'ordinateur de détecter les attaques par SYN flooding.

En plus d'être intéressant, ce projet fut très enrichissant au niveau des connaissances acquises. Tout d'abord nous avons pris conscience de certaines problématiques du domaine du système embarqué comme la gestion parcimonieuse des ressources. Ce point fut à l'origine de la décision de l'implémentation de l'algorithme CMS car il permet de limiter l'empreinte mémoire due au stockage des résultats d'analyse du trafic. Le NetFPGA aura aussi été l'occasion d'utiliser

un FPGA dans un environnement complexe, c'est-à-dire dans un environnement avec beaucoup de code déjà écrit et reposant sur une architecture évoluée. Avant d'être en état de développer, il a fallu se plonger dans cette architecture en lisant documentation et code. Cela nous a permis d'en découvrir les rouages et de comprendre comment réaliser nous-même des choses similaires. Notre formation scolaire nous amène à utiliser la suite Xilinx : ce projet nous a permis de confirmer ces enseignements mais aussi de réaliser que l'on pouvait utiliser ces outils d'une façon différente. Nous avons enfin découvert l'autre gros langage de conception, le Verilog. Ce point est très important car seul le langage VHDL est enseigné à Telecom Bretagne.

Les bénéfices pour l'école ne sont pas moins importants. La phase d'installation du NetFPGA est entièrement maîtrisée ainsi que la phase de développement. Il est donc maintenant possible d'implémenter des fonctionnalités poussées et intéressantes pour le/les groupes qui reprendront le projet l'année prochaine. L'école est aussi à présent en mesure d'utiliser nos acquis pour intégrer le NetFPGA dans le cursus scolaire de 3ème année. Enfin nous avons gardé à jour un large wiki [3], contenant toutes nos découvertes et développements, que l'école pourra exploiter librement.

Les perspectives d'avenir du projet sont variées : premièrement en ce qui concerne la surveillance de trafic, l'algorithme de détection d'attaques que nous avons implémenté pourra être utilisé pour stopper ces attaques. Des mesures de performance plus poussées pourront aussi être effectuées et des algorithmes plus efficaces pourront être testés. Enfin les connaissances acquises sur le NetFPGA pourront servir à tous les projets qui visent à accélérer matériellement des algorithmes ayant à s'intégrer dans un réseau.

Bibliographie

RÉFÉRENCES

- [1] GUIDE NETFPGA : <http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/Guide>, 2010.
- [2] IEEE P802.3BA 40GB/S AND 100GB/S ETHERNET TASK FORCE : <http://www.ieee802.org/3/ba/>, 2010.
- [3] LE TRAC DU PROJET : <http://trac.benoute.fr/netfpga>, 2010.
- [4] S. MUTHU MUTHUKRISHNAN : ALGORITHME CMS : <http://www.cs.rutgers.edu/~muthu/massdal-code-index.html>, 2004.
- [5] SITE DU PROJET TRAC : <http://trac.edgewall.org/>, 2010.
- [6] SITE OFFICIEL DU NETFPGA : <http://netfpga.org>, 2010.