

Titre: Conception, implémentation et caractérisation des performances de la fonction passerelle d'accès AGF de la 5G programmée en langage P4
Title:

Auteur: El Mehdi Makhroute
Author:

Date: 2023

Type: Mémoire ou thèse / Dissertation or Thesis

Référence: Makhroute, E. M. (2023). Conception, implémentation et caractérisation des performances de la fonction passerelle d'accès AGF de la 5G programmée en langage P4 [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie.
Citation: <https://publications.polymtl.ca/55103/>

 **Document en libre accès dans PolyPublie**
Open Access document in PolyPublie

URL de PolyPublie: <https://publications.polymtl.ca/55103/>
PolyPublie URL:

Directeurs de recherche: Tarek Ould-Bachir, & Yvon Savaria
Advisors:

Programme: Génie informatique
Program:

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

**Conception, implémentation et caractérisation des performances de la fonction
passerelle d'accès AGF de la 5G programmée en langage P4**

EL MEHDI MAKHROUTE

Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

**Conception, implémentation et caractérisation des performances de la fonction
passerelle d'accès AGF de la 5G programmée en langage P4**

présenté par **El Mehdi MAKHROUTE**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Samuel PIERRE, président

Tarek OULD-BACHIR, membre et directeur de recherche

Yvon SAVARIA, membre et codirecteur de recherche

Steven CHAMBERLAND, membre

DÉDICACE

Je dédie ce mémoire à mes parents et mes proches

REMERCIEMENTS

Je tiens à exprimer ma sincère gratitude à Tarek Ould-Bachir, mon directeur de recherche, pour m'avoir donné l'opportunité de contribuer à ce projet. Ce travail n'aurait pas vu le jour si le hasard de la vie ne l'avait pas mis sur mon chemin lors du stage de mon collègue Issam Dourid à Polytechnique.

Je tiens à remercier également mon co-directeur Yvon Savaria. Je me considère extrêmement chanceux d'avoir eu le soutien et l'encadrement de mon directeur Tarek et mon co-directeur Yvon tout au long ma maîtrise. Je suis tellement reconnaissant pour leur disponibilité, leur suivi constant, leur orientation, leur expertise et leurs précieux conseils.

Ce travail n'aurait pas été possible sans les collaborations que j'ai pu entamer dans le laboratoire du GRM avec d'autres étudiants membres et équipe technique, particulièrement : Vincent Brouillard, Ayoub Elharti, Hubert Laganière et Rejean Lepage.

Je souhaite également exprimer ma profonde gratitude envers l'organisme MITACS et Kaloom pour le support financier. Notamment, j'aimerais adresser mes sincères remerciements au personnel de Kaloom qui s'est grandement impliqué dans ce projet. Je tiens à exprimer ma reconnaissance envers Ludovic Beliveau, Eric Tremblay, Ebin Mathews et Anandakumar Udayamurthy pour leur soutien technique, leurs conseils avisés et leur expertise précieuse. Leur engagement indéfectible, leur disponibilité et leur patience ont grandement contribué à la réussite de mon mémoire.

RÉSUMÉ

La transition vers la convergence fixe-mobile présente des défis complexes, qui sont abordés en utilisant les technologies de virtualisation des fonctions réseau (Network Function Virtualization : NFV) et les réseaux définis par logiciel (Software-Defined Networking : SDN). Ce mémoire souligne l'importance de la passerelle de réseau à large bande (Broadband Network Gateway : BNG) et de la fonction passerelle d'accès (AGF) dans la mise en œuvre de la FMC dans le noyau de la 5G.

Dans le cadre de cette recherche, l'objectif principal est de concevoir et de déployer une solution AGF en utilisant le langage P4, et d'en caractériser les performances sur un commutateur Tofino. Le travail procède en analysant les spécifications techniques de l'AGF fournies par le Broadband Forum (BBF), à traduire lesdites spécifications en code P4 (Programming Protocol-Independent Packet Processors), avant de procéder à la simulation du plan de données de l'AGF sur des modèles comportementaux de commutateurs. L'implémentation du prototype AGF sur le commutateur Tofino suit, ainsi que l'évaluation des performances et des limitations de la solution, ainsi que la formulation de recommandations pour un déploiement industriel.

Les résultats de l'étude montrent que le commutateur Tofino programmable en P4 peut supporter jusqu'à 285 500 sessions en utilisant uniquement des SRAM, et jusqu'à 400 300 sessions en exploitant les TCAM (Ternary Content-Addressable Memory). Ces conclusions permettent de mieux comprendre les performances et les limites de la solution proposée, ouvrant la voie à un déploiement industriel de la convergence fixe-mobile basée sur le commutateur Tofino.

ABSTRACT

The transition to fixed-mobile convergence presents complex challenges, which are addressed by using Network Function Virtualization (NFV) and Software-Defined Networking (SDN) technologies. This thesis highlights the importance of the Broadband Network Gateway (BNG) and the Access Gateway Function (AGF) in implementing fixed-mobile convergence in the core of 5G.

The main objective of this research is to design and deploy an AGF solution using the P4 language and characterize its performance on a Tofino switch. The work involves analyzing the technical specifications of AGF provided by the Broadband Forum (BBF), translating these specifications into P4 code (Programming Protocol-Independent Packet Processors), and simulating the AGF data plane on behavioral models of switches. The implementation of the AGF prototype on the Tofino switch follows, along with evaluating the performance and limitations of the solution and providing recommendations for industrial deployment.

The results of the study demonstrate that the programmable P4 Tofino switch can support up to 285,500 sessions using only SRAM and up to 400,300 sessions by leveraging TCAM (Ternary Content-Addressable Memory). These findings enhance understanding of the performance and limitations of the proposed solution, paving the way for industrial deployment of fixed-mobile convergence based on the Tofino switch.

TABLE DES MATIÈRES

DÉDICACE	iii
REMERCIEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vi
TABLE DES MATIÈRES	vii
LISTE DES TABLEAUX	x
LISTE DES FIGURES	xi
LISTE DES SIGLES ET ABRÉVIATIONS	xii
CHAPITRE 1 INTRODUCTION	1
1.1 Évolution de l'Internet et de l'Internet mobile	1
1.2 Éléments de la problématique	1
1.2.1 Réseau 5G et SDN	1
1.2.2 Convergence des réseaux fixes et mobiles	2
1.2.3 Langage P4 et commutateurs programmables	2
1.2.4 Commutateur Tofino	3
1.3 Objectifs de recherche	3
1.4 Organisation du mémoire	4
CHAPITRE 2 CONCEPTS DE BASE ET REVUE DE LITTÉRATURE	5
2.1 Introduction	5
2.2 Réseaux définis par logiciel	5
2.2.1 Problématique et définition	5
2.2.2 Virtualisation des fonctions de réseau	7
2.3 Environnement de développement P4	8
2.3.1 Le langage de programmation P4	8
2.3.2 Programmation du plan des données	8
2.3.3 L'architecture PISA	9
2.3.4 Bénéfices et limitations du langage P4	10

2.4	Commutateurs cibles	11
2.4.1	Commutateurs programmables en P4	11
2.4.2	Cibles P4 retenues pour la réalisation du travail	12
2.4.3	Le commutateur logiciel BMv2	13
2.4.4	Le commutateur Tofino	14
2.4.5	Le Modèle Tofino	15
2.5	Convergence des réseaux filaire et mobile	15
2.5.1	La 5G et la FMC	15
2.5.2	UPF	15
2.5.3	BNG	16
2.5.4	AGF	16
2.5.5	Qualité de service	17
2.6	Éléments de méthodologie	17
2.6.1	Mise en œuvre du plan de données en P4	17
2.6.2	Exécution sur des cibles hétérogènes	17
2.6.3	Évaluation de performance et utilisation des ressources	18
CHAPITRE 3 MÉTHODE PROPOSÉE POUR LA MISE EN ŒUVRE DU PLAN DE DONNÉES DE LA FONCTION PASSERELLE D'ACCÈS AGF		20
3.1	Introduction	20
3.2	Analyse fonctionnelle de l'AGF	20
3.2.1	Architecture traditionnelle des réseaux d'accès fixes et FMC	20
3.2.2	Scénarios de déploiement de l'AGF et modes d'opération	22
3.2.3	Pile de protocoles AGF	23
3.3	Environnements de développement et de test de l'AGF	24
3.3.1	Outils logiciels	24
3.3.2	Tests de performance physiques	24
3.3.3	Prototypage sur l'ASIC	25
CHAPITRE 4 CONCEPTION ET IMPLÉMENTATION DE LA FONCTION AGF - BASÉE SUR LE LANGAGE P4 - FLEXIBLE DE HAUTE PERFORMANCE		26
4.1	Conception du prototype AGF	26
4.1.1	Encapsulation/désencapsulation des en-têtes du paquet	26
4.1.2	Conception du pipeline P4	30
4.2	Implémentation du plan de données	32
4.2.1	Parser	32
4.2.2	Blocs de contrôle	34

4.3	Implémentation du plan de contrôle en P4	34
CHAPITRE 5 EVALUATION DES PERFORMANCES : RÉSULTATS ET ANALYSE		36
5.1	Introduction	36
5.2	Banc d'essai	36
5.3	Résultats expérimentaux	38
5.3.1	Débit	38
5.3.2	Taux de perte de paquets	38
5.3.3	Latence	39
5.3.4	Utilisation des ressources matérielles	39
5.4	Conclusions et recommandations	45
CHAPITRE 6 CONCLUSION		46
6.1	Synthèse des travaux	46
6.2	Limitations des travaux	46
6.3	Améliorations futures	47
RÉFÉRENCES		49

LISTE DES TABLEAUX

Tableau 2.1	Exemples de cibles P4	11
Tableau 4.1	Règles du plan de contrôle	35
Tableau 5.1	Sommaire des expériences	37
Tableau 5.2	Le débit mesuré (en Gb/s) en augmentant le nombre de flux pour un seul générateur de trafic	38
Tableau 5.3	Le débit mesuré (en Gb/s) en utilisant deux générateur de trafic avec le maximum de flux possible	38
Tableau 5.4	Statistiques d'utilisation des ressources mémoire pour 286 500	40
Tableau 5.5	Statistiques d'utilisation des ressources mémoire pour 400 300	41
Tableau 5.6	Statistiques d'utilisation des différentes ressources pour 286 500 . . .	43
Tableau 5.7	Statistiques d'utilisation des différentes ressources pour 400 300 . . .	44

LISTE DES FIGURES

Figure 2.1	Architecture haut niveau d'un réseau défini par logiciel	6
Figure 2.2	Comparaison entre le modèle de contrôle traditionnel et le modèle SDN	7
Figure 2.3	L'architecture PISA	9
Figure 2.4	Flot de travail du BMv2	13
Figure 3.1	Architecture d'accès fixe filaire traditionnelle	20
Figure 3.2	Architecture d'accès réseau mobile	21
Figure 3.3	Différentes approches permettant la convergence fixe mobile	21
Figure 3.4	AGF en mode directe	22
Figure 3.5	AGF en mode adaptatif	22
Figure 3.6	Pile protocolaire AGF	23
Figure 3.7	Capture d'écran de l'outil de visualisation P4I	25
Figure 4.1	Différents types de trafic AGF : trafic de contrôle et trafic utilisateur	27
Figure 4.2	Encapsulation/désencapsulation des en-têtes de la fonction AGF . . .	27
Figure 4.3	Format d'en-tête du protocole 5WE	28
Figure 4.4	Format d'en-tête du protocole GTP-U	29
Figure 4.5	Format d'en-tête d'extension GTP-U PDU, direction Upstream . . .	29
Figure 4.6	Format d'en-tête d'extension GTP-U PDU, direction Downstream . .	29
Figure 4.7	Conception du pipeline P4 pour l'AGF	30
Figure 4.8	Analyseur d'entêtes.	33
Figure 5.1	Banc d'essai utilisé pour l'évaluation avec un seul serveur	37
Figure 5.2	Banc d'essai utilisé pour l'évaluation avec deux serveur	37

LISTE DES SIGLES ET ABRÉVIATIONS

3GGP	3 rd Generation Partnership Project
AAA	Authentication, Authorization, and Accounting
AGF	Access Gateway Function
ASIC	Application-specific Integrated Circuit
BBF	Broadband Forum
FMC	Fixed Mobile Convergence
FMIF	Fixed Mobile Interworking Function
FPGA	Field-Programmable Gate Array
HQoS	Hierarchical Quality of Service
LDWG	P4 Language Design Working Group
RAN	Radio Access Network
QoS	Quality of Service
SDN	Software-Defined Networking
SRAM	Static Random Access Memory
TCAM	Ternary Content Addressable Memory
TNA	Tofino Native Architecture
UPF	User Plane Function
VFN	Virtual Fixed Network

CHAPITRE 1 INTRODUCTION

1.1 Évolution de l’Internet et de l’Internet mobile

Depuis les prémisses de son invention en 1960, et alors qu’il était initialement conçu principalement pour faciliter la communication entre les universités et les instituts de recherche, l’Internet est devenu un outil incontournable dans la vie quotidienne. L’Internet grand public a commencé à se populariser dans les années 1990 avec la création de navigateurs web tel que Netscape. S’en est suivi le développement du commerce en ligne, des moteurs de recherche et des premiers réseaux sociaux. L’avènement de la connexion haut débit a permis l’émergence de services multimédias tels que la diffusion en continu de musique et de vidéos en ligne [1]. Aujourd’hui, l’Internet est un outil qui permet l’accès à l’information, la communication, l’éducation, les transactions commerciales et bien plus encore. Il a ouvert de nouvelles possibilités pour les entreprises et les individus, permettant une plus grande collaboration et une meilleure accessibilité de l’information.

Les réseaux cellulaires ont joué un rôle important dans l’essor de l’Internet. Ils ont permis aux utilisateurs de s’y connecter à partir de n’importe quel endroit, en utilisant leur téléphone portable ou une tablette, notamment avec l’arrivée des réseaux 3G. Lancés à la fin des années 90 et au début des années 2000, ils ont ouvert la voie à des applications plus avancées telles que la vidéo en flux continu et la navigation GPS [2]. Les réseaux 4G, lancés à partir de 2010, ont augmenté les performances de connexion, permettant le déploiement d’applications encore plus riches en données. La plus récente génération de réseaux cellulaires, le réseau 5G, est en cours de déploiement depuis 2020. Elle devrait permettre le déploiement d’un large éventail d’applications, notamment celles qui exigent une latence ultra-faible et des débits de données très élevés [3].

1.2 Éléments de la problématique

1.2.1 Réseau 5G et SDN

La cinquième génération des réseaux cellulaires (5G) est la dernière norme de télécommunications mobiles ; elle succède à la 4G, lancée en Suède en 2009. La 5G promet une connectivité plus rapide, plus fiable et plus étendue que la 4G, des latences plus faibles et une capacité accrue pour connecter des milliards d’appareils, notamment ceux issus de l’Internet des objets (*Internet of things : IoT*). La 5G utilise des fréquences plus élevées que la 4G, avec des

ondes millimétriques (3.4-3.8 GHz) qui peuvent transporter plus de données à des vitesses plus élevées, mais qui ont également une portée plus courte. Les réseaux 5G seront constitués de petites cellules qui peuvent être déployées dans les zones à forte densité de trafic, tels que les centres urbains, les stades et les centres commerciaux [3].

Les nombreux défis de la 5G ont été relevés en adoptant les technologies de virtualisation des fonctions de réseau (*network function virtualization* : NFV) et des réseaux définis par logiciel (*software-defined networking* : SDN) [4]. La NFV est une des approches utilisées pour relever un tel défi. Elle constitue une opportunité pour l'implémentation rapide et moins coûteuse des nouvelles fonctions réseau, notamment de la 5G [5]. Le SDN diffère des approches réseau traditionnelles en séparant le plan de données (appelé aussi plan d'utilisateur), qui achemine le trafic, et le plan de contrôle, qui fournit les règles en fonction desquelles les décisions d'acheminement doivent être prises [6].

1.2.2 Convergence des réseaux fixes et mobiles

Une nouveauté importante de la 5G est la liaison de l'accès internet fixe dans le cœur du réseau 5G, permettant la convergence des réseaux fixes et mobiles (fixed mobile convergence : FMC). La FMC permet d'offrir des services transparents et cohérents afin d'améliorer l'expérience utilisateur et d'améliorer la gestion du trafic réseau. La FMC promet des économies importantes de coût et un gain en efficacité opérationnelle. À terme, elle permettra d'assurer aux utilisateurs des combinaisons plus riches et une continuité de service réutilisable indépendamment du type d'accès réseau.

Cependant, une telle transition nécessite une transformation significative au niveau d'accès du réseau fixe, et plus précisément au niveau de la passerelle de réseau à large bande (*Broadband Network Gateway* : BNG) [7]. Ainsi, le Broadband Forum (BBF) a introduit la fonction passerelle d'accès (*Access Gateway Function* : AGF) pour atteindre la convergence fixe-mobile [8]. L'AGF ne se comporte pas seulement comme un BNG traditionnel. Elle doit fournir des fonctions supplémentaires essentielles à l'interaction avec le réseau cœur 5G et assure la connectivité et la gestion des connexions d'accès pour les terminaux utilisateur.

1.2.3 Langage P4 et commutateurs programmables

La séparation des plans de contrôle et de données du SDN a permis l'émergence du langage P4, un nom qui tire son origine des quatre (4) P dans *Programming Protocol-Independent Packet Processors* [9]. Le P4 est un langage servant à la programmation des commutateurs réseau, des routeurs et des pare-feu. Le langage P4 est conçu pour être indépendant des protocoles

de réseau, ce qui signifie qu’il peut être utilisé pour définir le traitement de paquets pour n’importe quel protocole de réseau, qu’il s’agisse de protocoles existants tels que Ethernet et IPv4/IPv6 ou de protocoles personnalisés.

Les commutateurs réseau programmables en P4 sont apparus durant la seconde moitié de la décennie 2010. Leur invention a découlé de l’architecture de traitement de paquets programmable PISA (*Protocol-Independent Switch Architecture*) [9], et en réponse à la demande croissante de flexibilité et de personnalisation dans les réseaux de données. Avant l’arrivée de ces commutateurs, les équipements de réseau étaient généralement propriétaires et leur fonctionnement était limité par les fonctionnalités intégrées dans leur matériel et leur logiciel.

1.2.4 Commutateur Tofino

Les commutateurs Tofino sont un type de commutateur réseau programmable en P4 de haute performance développé par la société Barefoot Networks, maintenant filiale de Intel [10]. Ces commutateurs utilisent une architecture PISA, de sorte qu’ils permettent de personnaliser et de programmer les fonctionnalités de traitement de paquets en fonction de besoins spécifiques de l’usager.

Cependant, le commutateur Tofino présente certaines limitations et contraintes. D’une part, il présente une capacité de stockage limitée pour les tables de flux et les autres données, ce qui peut limiter sa capacité à gérer de grands volumes de trafic. D’autre part, bien que le Tofino soit extrêmement rapide et efficace dans le traitement des paquets, il peut rencontrer des problèmes de saturation avec de très gros volumes de trafic ou des configurations très complexes [11]. Finalement, le développement de programmes P4 pour le Tofino nécessite de les adapter à certaines contraintes syntaxiques que nous préciserons plus loin.

1.3 Objectifs de recherche

Notre travail de recherche s’inscrit dans un effort industriel de réalisation de l’AGF sur le commutateur Tofino. Ce travail a été accompli dans le cadre d’un projet financé par MITACS et la société Kaloom. Il a ainsi pu bénéficier du concours des employés de la compagnie partenaire ainsi que d’une équipe d’étudiants.

Pour mener à bien le projet, nous avons dû analyser en détail les documents de le BBF portant sur la FMC [8] ainsi que celle d’une ébauche du protocole 5WE [12]. Suite à ce travail fastidieux, une spécification de l’AGF a été établie, et son implémentation en P4 réalisée. L’application a ensuite été testée avec un modèle comportemental d’un commutateur programmable en P4, avant de l’être sur un modèle propriétaire du Tofino. L’AGF a par la

suite été déployé sur un Tofino de première génération auquel nous avons accès au laboratoire. Nous avons pu en caractériser les performances et présenter des conclusions quant à sa mise en œuvre.

Ainsi, l'objectif principal de notre travail est de concevoir en P4 une solution AGF de calibre industriel, de la déployer sur un commutateur programmable en P4 et d'en caractériser les performances. Les objectifs spécifiques suivants en découlent :

1. Détailler les spécifications techniques fournies par le BBF pour l'AGF ;
2. Simuler le plan de donnée de l'AGF sur des modèles comportementaux d'un commutateur ;
3. Implémenter un prototype de l'AGF sur le commutateur matériel programmable en P4, à savoir le Tofino ;
4. Évaluer les performances et les limitations de notre solution ; et
5. Proposer des recommandations quant à un déploiement industriel de la solution.

1.4 Organisation du mémoire

Le reste du mémoire est structuré comme suit. Le second chapitre décrit les concepts de base nécessaires à la compréhension du sujet de notre travail. Il offre également une revue de la littérature pertinente au domaine. Le troisième chapitre détaille la méthodologie utilisée dans notre travail de recherche. Le quatrième chapitre décrit les étapes de conception de notre prototype de l'AGF ainsi que son implémentation sur différentes cibles. Le cinquième chapitre présente les résultats obtenus et les analyses faites dans l'évaluation des performances du prototype de l'AGF que nous avons contribué à développer. Il détaille également les limitations de la solution proposée et suggère des pistes d'améliorations futures. Le sixième chapitre conclut le travail, et propose des perspectives pour les travaux futurs.

CHAPITRE 2 CONCEPTS DE BASE ET REVUE DE LITTÉRATURE

2.1 Introduction

Le présent chapitre a pour objectif de fournir une introduction aux concepts de base du SDN et des réseaux programmables en P4, ainsi que de présenter une revue de la littérature pertinente au domaine de recherche. Nous commencerons par expliquer les principes fondamentaux du SDN, y compris l'architecture du plan de contrôle et du plan de données, ainsi que les avantages et les limites de cette approche. Ensuite, nous aborderons les réseaux programmables en P4, qui sont de plus en plus populaires en raison de leur capacité à personnaliser le traitement des paquets dans les réseaux de données. Nous discuterons des caractéristiques clés de P4, telles que la programmabilité de la couche de traitement de paquets et la flexibilité des pipelines de traitement. Enfin, nous passerons en revue la littérature pertinente au domaine de recherche, en examinant les travaux antérieurs sur le SDN et les réseaux programmables en P4, ainsi que les applications et les cas d'utilisation qui ont été développés. Cette revue de la littérature fournira un aperçu de l'état actuel de la recherche dans ce domaine, ainsi que des perspectives pour les futurs travaux de recherche et développement.

2.2 Réseaux définis par logiciel

2.2.1 Problématique et définition

L'architecture du réseau traditionnelle adoptée par Internet est basée sur des routeurs, des commutateurs et des périphériques réseau de fonctionnalités simples. Il s'agit d'un système distribué où chaque nœud collabore avec les autres nœuds en exécutant des logiciels de contrôle distribués complexes et généralement fermés pour assurer la transmission de paquets. Il y avait de bonnes raisons pour lesquelles Internet a adopté cette approche. Parmi elles, la fiabilité et la facilité d'étendre le réseau. Cependant, une telle architecture a ralenti l'innovation et a rendu la gestion du réseau plus difficile, complexe et coûteuse [13]. Par exemple les opérateurs réseau ont besoin de configurer chaque commutateur séparément, en utilisant souvent des interfaces spécifiques aux fournisseurs.

Le Réseau défini par logiciel (*Software-Defined Networking* ou SDN) [14–16] a été introduit pour changer la façon de concevoir et gérer les réseaux. Il s'agit d'un modèle d'architecture de réseau qui permet de gérer les réseaux informatiques de manière centralisée et programmable. L'approche SDN sépare le plan de contrôle (*control plane*) qui est responsable de décider

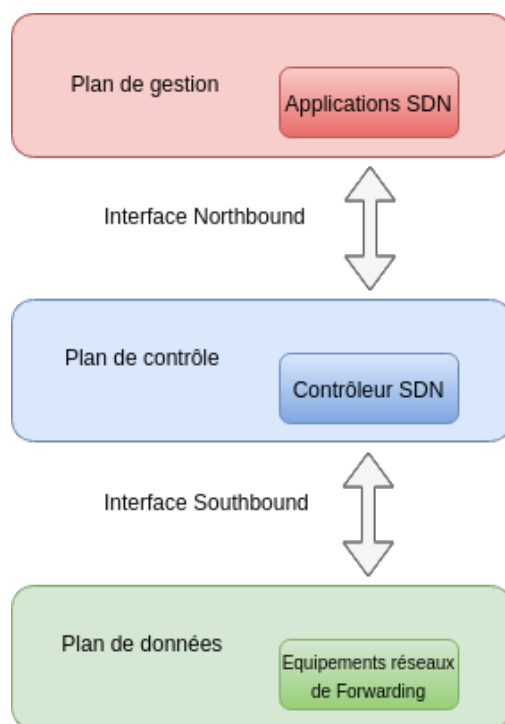


FIGURE 2.1 Architecture haut niveau d'un réseau défini par logiciel, figure adaptée de [13]

comment les paquets doivent être acheminés, de la transmission de trafic qui est gérée par le plan de données (*data plane*) [6], tel qu'illustré à la Figure 2.1. En plus du plan de données et de contrôle, un autre plan peut être considéré, souvent appelé plan de gestion qui interagit avec les contrôleurs SDN afin de définir les politiques de réseau et d'observer le fonctionnement du réseau [17]. Deux éléments clés caractérisent le nouveau paradigme SDN. Premièrement, la séparation du plan de données et du plan de contrôle. Deuxièmement, la centralisation du plan de contrôle de telle sorte qu'un seul contrôleur peut contrôler plusieurs périphériques du plan de données. Une comparaison entre l'architecture réseau traditionnelle et celle du SDN est illustrée dans la Figure 2.2 .

La communication entre les différents plans SDN est assurée par des APIs. L'interface qui fait le lien entre le plan de contrôle et le plan de gestion, généralement mise en œuvre via une API REST, est communément appelée interface *Northbound*. L'API *Southbound*, située entre le plan de contrôle et le plan de données, fournit un canal de communication permettant aux contrôleurs du plan de contrôle d'interagir avec les dispositifs du plan des données. OpenFlow [18] est un exemple bien connu d'API *Southbound*. Il représente l'interface standard open-source entre les plans de contrôle et de données dans une architecture SDN. Le SDN offre des avantages considérables qui sont exploités par la 5G [19]. En découplant les plans

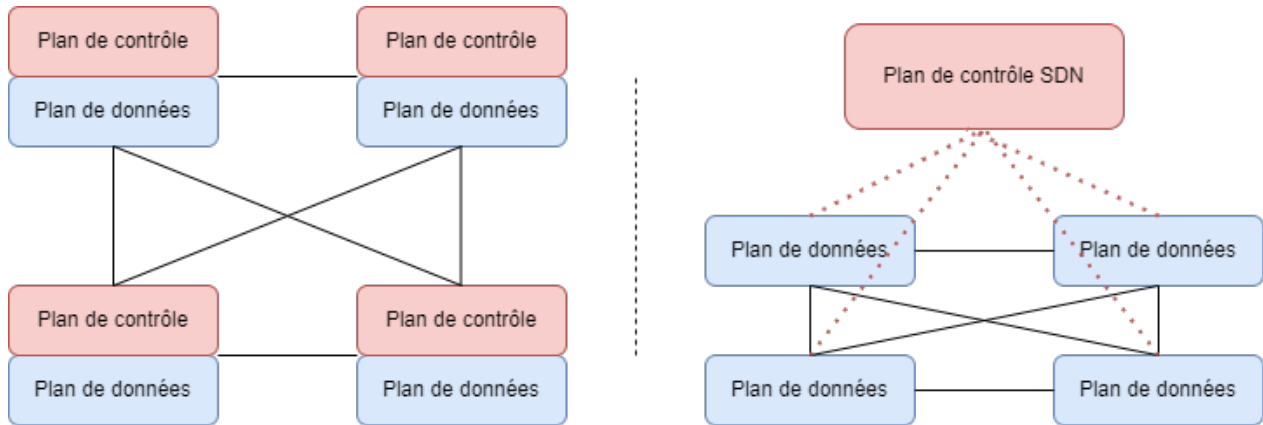


FIGURE 2.2 Comparaison entre le modèle de contrôle traditionnel et le modèle SDN

de contrôle et de données, le SDN favorise la flexibilité, l'évolutivité et la gestion efficace des ressources du réseau. Il améliore la sécurité du réseau grâce à un contrôle centralisé [20]. De plus, il favorise également le *Network slicing* [21], un élément essentiel de la 5G, qui permet un découpage logique du réseau pour divers services sur la même infrastructure, améliorant ainsi la qualité du service et l'expérience de l'utilisateur tout en réduisant les coûts [22].

2.2.2 Virtualisation des fonctions de réseau

Alors que les réseaux traditionnels reposent souvent sur du matériel spécifique pour remplir des fonctions particulières (routeurs, commutateurs, pare-feux ...), la Virtualisation des fonctions de réseau (*Network Function Virtualization NFV*) découple les fonctions de réseau des dispositifs matériels en permettant d'exécuter diverses fonctions réseaux sur des serveurs génériques, qui ne sont pas spécialisés pour une fonction spécifique. Pour ce faire, ces fonctions sont mises en œuvre sous forme d'applications logicielles, qui peuvent être exécutées sur n'importe quel serveur standard [23]. Cela présente plusieurs avantages tels qu'une flexibilité accrue, des coûts réduits et une meilleure évolutivité.

La Virtualisation des fonctions réseau (NFV) est généralement mise en œuvre en association étroite avec la technologie SDN. Les deux technologies peuvent bénéficier l'une de l'autre, ce qui s'avère particulièrement avantageux dans le cadre des applications 5G [4]. Les solutions NFV diffèrent de l'infrastructure traditionnelle basée sur du matériel dédié de fonction fixe. En effet, malgré la grande performance des solutions matérielles dédiées, elles manquent de flexibilité, ce qui ralentit le processus d'implantation des nouveaux algorithmes et protocoles réseau. En conséquence, les solutions basées sur du matériel dédié sont généralement incapables de suivre le taux d'innovation continue que connaît le monde des réseaux informa-

tiques. Bien que la NFV permette d’optimiser les coûts et offre une flexibilité en implémentant les fonctions réseau sur des serveurs basés sur des processeurs à usage général, il est limité en terme de performance, comparé aux solutions basées sur du matériel dédié.

2.3 Environnement de développement P4

2.3.1 Le langage de programmation P4

La séparation des plans de contrôle et de données du SDN a permis l’émergence du langage P4, un nom qui tire son origine des quatre (4) P dans *Programming Protocol-Independent Packet Processors* [9]. Le P4 est un langage servant à la programmation des commutateurs réseau, des routeurs et des pare-feu. Le langage P4 est conçu pour être indépendant des protocoles de réseau, ce qui signifie qu’il peut être utilisé pour définir le traitement de paquets pour n’importe quel protocole de réseau, qu’il s’agisse de protocoles existants tels que Ethernet et IPv4/IPv6 ou de protocoles personnalisés. Comme nous le verrons, cette propriété sera utile dans la mise en œuvre des protocoles proposés pour prendre en charge la convergence des réseaux fixes et mobiles dans le contexte de la 5G.

Le groupe de travail sur la conception du langage P4 (*P4 Language Design Working Group - LDWG*) du consortium du langage P4 a standardisé jusqu’à présent deux normes distinctes de P4 : P4-14 et P4-16. P4-16 est une version améliorée et plus expressive du langage P4 par rapport à P4-14. Cette nouvelle version offre une syntaxe plus riche, permettant une définition plus détaillée et sophistiquée du traitement des paquets dans les commutateurs réseau programmables. De plus, P4-16 introduit de nouvelles abstractions qui simplifient la programmation des commutateurs et rendent le code plus modulaire et réutilisable. Cette version prend également en charge de nouveaux protocoles réseau et gère des situations plus complexes, ce qui la rend plus adaptée aux réseaux modernes en constante évolution.

Il est important de noter que les programmes écrits en P4-14 ne sont pas directement compatibles avec P4-16, nécessitant ainsi des adaptations pour fonctionner avec la version améliorée.

2.3.2 Programmation du plan des données

Pouvoir programmer le plan des données dans les réseaux offre des avantages considérables en matière de flexibilité, d’innovation et d’efficacité. En utilisant des langages de programmation comme P4, les opérateurs de réseau et les chercheurs peuvent définir de manière personnalisée le traitement des paquets, les règles de routage et les actions à entreprendre en fonction de différents protocoles réseau [24]. Cela permet d’adapter le réseau aux besoins spécifiques des

applications, des services et des utilisateurs, améliorant ainsi les performances et la qualité de service. De plus, la programmabilité du plan des données facilite l'introduction rapide de nouvelles fonctionnalités, accélérant l'innovation et l'évolution des réseaux pour répondre aux défis émergents et aux demandes changeantes des utilisateurs. Ainsi, la programmation du plan des données offre une approche plus dynamique et évolutive pour la gestion et l'optimisation des réseaux.

2.3.3 L'architecture PISA

Les commutateurs réseau programmables en P4 sont apparus durant la seconde moitié de la décennie 2010. Leur invention a découlé de l'architecture de traitement de paquets programmable PISA (*protocol-independent switch architecture*) [9], et en réponse à la demande croissante de flexibilité et de personnalisation dans les réseaux de données. Avant l'arrivée de ces commutateurs, les équipements de réseau étaient généralement propriétaires et leur fonctionnement était limité par les fonctionnalités intégrées dans leur matériel et leur logiciel.

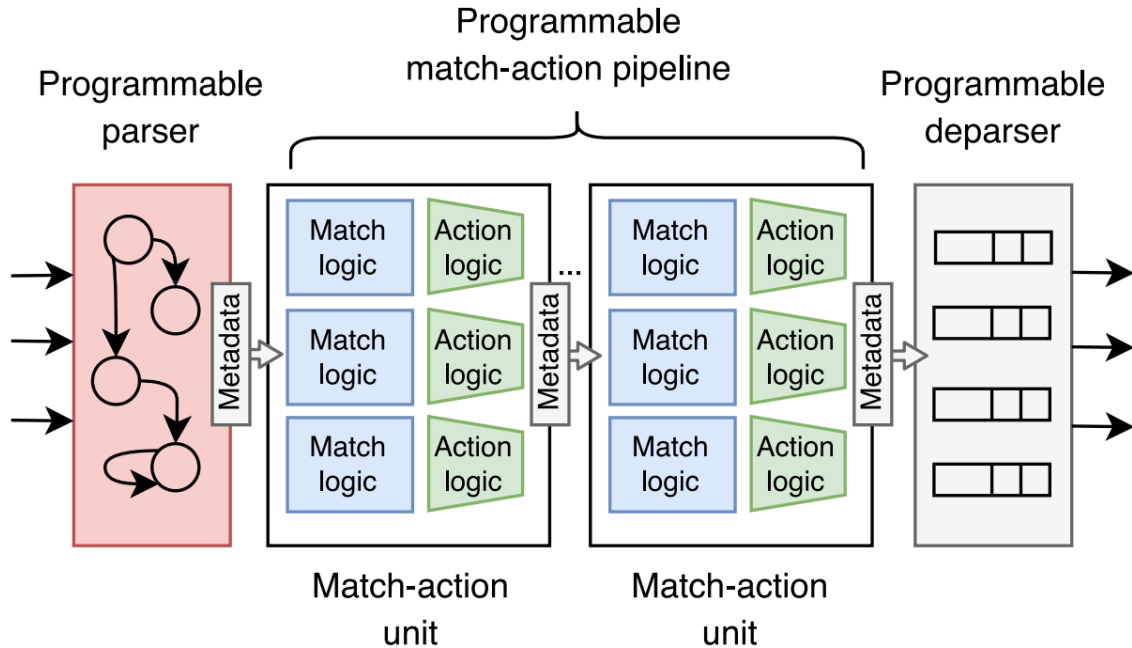


FIGURE 2.3 L'architecture PISA, tirée de [25]

L'architecture PISA est composée de trois éléments programmables, comme illustré à la Figure 2.3. Tout d'abord, un analyseur syntaxique (*parser*) permet de désassembler les paquets entrants et d'extraire les informations nécessaires pour leur traitement. Ensuite, un pipeline de correspondance (*match-action pipeline*) est responsable d'appliquer les règles de traitement

spécifiques en fonction des caractéristiques des paquets. Enfin, un ré-assembleur des entêtes (*deparser*) est chargé de reformater les paquets avant de les renvoyer vers leur destination. Cette architecture offre une grande flexibilité et possibilité de personnalisation, permettant de répondre aux besoins spécifiques des applications et des services dans les réseaux programmables. Cette architecture, contrairement aux puces dédiées (*Application-specific Integrated circuit* : ASIC) typiques, offre une grande flexibilité tout en maintenant des performances élevées [26].

2.3.4 Bénéfices et limitations du langage P4

Le langage P4 présente plusieurs avantages en matière de programmation et de gestion des commutateurs réseau. Tout d'abord, P4 offre une grande flexibilité en permettant aux développeurs de définir le traitement des paquets de manière indépendante des protocoles réseau sous-jacents [27]. Cela signifie qu'un seul programme P4 peut être utilisé pour programmer différents types de commutateurs, ce qui simplifie considérablement le processus de développement et de déploiement des réseaux. De plus, P4 permet de créer des fonctions de réseau personnalisées pour répondre aux besoins spécifiques des applications et des services, ce qui ouvre la voie à une innovation rapide et à une meilleure adaptation des réseaux aux nouvelles exigences [25].

Cependant, le langage P4 présente également certaines limitations. Tout d'abord, sa programmabilité peut entraîner une complexité accrue dans la gestion et la maintenance des réseaux, en particulier lorsque les règles de traitement des paquets deviennent très personnalisées. De plus, le langage P4 est relativement nouveau, ce qui signifie qu'il peut manquer certains outils et bibliothèques bien établis pour faciliter le développement. En conséquence, il est parfois nécessaire d'investir davantage dans l'apprentissage du langage et dans la recherche de solutions pour résoudre des problèmes spécifiques. Enfin, l'efficacité et les performances des programmes P4 dépendent également des capacités matérielles des commutateurs programmables. Par conséquent, certaines fonctionnalités plus avancées ou des opérations complexes peuvent être limitées par les contraintes matérielles, ce qui nécessite une gestion prudente des compromis entre la programmabilité et la performance [25].

Malgré ces limitations, le langage P4 reste un outil puissant pour la programmation des commutateurs réseau et continue de susciter un grand intérêt dans l'industrie et la recherche pour l'innovation et l'évolution des réseaux. C'est la raison pour laquelle nous avons opté pour ce langage lors de la création du prototype développé dans ce mémoire de maîtrise.

2.4 Commutateurs cibles

2.4.1 Commutateurs programmables en P4

Tout dispositif capable d'exécuter des programmes P4 est appelé cible P4. En raison de la propriété d'indépendance de cible du langage P4, plusieurs types de cibles sont introduits pour permettre le traitement des paquets. Les commutateurs programmables en P4 offrent aux utilisateurs une flexibilité et une capacité de personnalisation accrues pour le traitement de paquets dans les réseaux de données. Il en existe plusieurs sur le marché, notamment ceux présentés au Tableau 2.1. Ces cibles sont classées par catégorie (logiciel, FPGA et ASIC). Le Tableau 2.1 liste également les versions P4 supportées, l'architecture et le débit maximal des dites cibles.

TABLEAU 2.1 Exemples de cibles P4

Cible	Catégorie	Version P4	Architecture $P4_{16}$	Débit max
p4c-behavioral	Logiciel	$P4_{14}$	-	-
Bmv2	Logiciel	$P4_{14} - P4_{16}$	V1Model - PSA	1 Gb/s
PISCES [28]	Logiciel	$P4_{14}$	-	-
T4P4S	Logiciel	$P4_{14} - P4_{16}$	V1Model - PSA	-
Tofino Model	Logiciel	$P4_{14} - P4_{16}$	TNA	-
Intel N3000	FPGA	$P4_{14} - P4_{16}$	V1Model	
P4->NetFPGA	FPGA	$P4_{16}$	SimpleSumeSwitch	40 Gb/s
Alveo U280	FPGA	$P4_{14} - P4_{16}$	-	200 Gb/s
Tofino [10]	ASIC	$P4_{14} - P4_{16}$	TNA	6.4 Tb/s
Tofino 2	ASIC	$P4_{14} - P4_{16}$	TNA	12.8 Tb/s
Silicon One Q200 [29]	ASIC	$P4_{14} - P4_{16}$	-	12.8Tb/s

Une cible logicielle pour le langage P4 présente un intérêt essentiel dans le développement et la mise au point des programmes P4 destinés à être exécutés sur des commutateurs réseau programmables. En utilisant une cible logicielle, les développeurs peuvent tester et évaluer leurs programmes P4 sur un environnement de simulation, ce qui permet d'obtenir des résultats rapides et efficaces. Cela facilite grandement le processus de développement en éliminant la nécessité d'avoir un matériel physique dédié pour chaque itération du développement. De plus, les cibles logicielles fournissent une plate-forme de test flexible et facilement reproductible, permettant aux développeurs de vérifier le comportement de leurs programmes P4 dans divers scénarios et configurations de réseau. Les cibles logicielles utilisent le plus souvent une architecture dite PSs (*Portable Switch Architecture*), dont le V1Model-PSA (Version 1 du modèle) est la première version [30].

Une cible FPGA offre des avantages significatifs : Les FPGA étant des dispositifs matériels

reconfigurables, ils permettent aux développeurs de mettre en œuvre directement leurs programmes P4 sur du matériel personnalisé. L'intérêt d'une telle cible réside dans la possibilité de bénéficier d'une performance matérielle dédiée et hautement spécialisée, offrant une accélération significative des opérations de traitement de paquets. En programmant le FPGA avec des programmes P4, les commutateurs réseau basés sur FPGA peuvent être adaptés spécifiquement aux besoins de l'application et aux exigences du réseau, offrant ainsi une flexibilité et une efficacité inégalées. Les outils permettant de convertir du P4 en langage de description matérielle font leur chemin dans l'industrie [31].

Les cibles ASIC offrent des qualités exceptionnelles pour le langage P4. Ces puces sont spécialement conçues pour accélérer les performances de traitement de paquets en exploitant pleinement les programmes P4. Leur architecture matérielle dédiée permet une exécution rapide et efficace des programmes P4, garantissant une latence réduite et une bande passante élevée. Les cibles ASIC offrent également une grande flexibilité, car elles peuvent être programmées pour prendre en charge une variété de fonctions réseau spécifiques, ce qui permet une personnalisation complète des commutateurs réseau en fonction des besoins de l'application.

1. Le commutateur Tofino d'Intel [10] : il s'agit d'une famille de commutateurs réseau programmables en P4, un des premiers du marché, dont l'architecture interne est appelée TNA : *Tofino Native Architecture* ;
2. Le commutateur Silicon One de Cisco [29] : il s'agit d'un commutateur programmable à très haut débit conçu pour permettre à la prochaine génération des produits réseaux Cisco d'être compatibles avec la norme P4.

2.4.2 Cibles P4 retenues pour la réalisation du travail

Pour répondre aux exigences du projet, nous avons sélectionné trois commutateurs pour nos expérimentations. Pour les environnements de développement et de test, nous avons retenu les commutateurs logiciels BMv2 et Tofino Model, offrant une flexibilité dans la programmation et la simulation des réseaux programmables. En outre, nous avons choisi le commutateur ASIC Tofino pour ses performances matérielles dédiées et sa capacité à accélérer le traitement des paquets en exploitant pleinement les programmes P4. Cette combinaison nous permettra de réaliser des évaluations exhaustives et de comparer les performances et les limitations des différentes cibles pour les réseaux programmables. À l'origine, l'inclusion d'une cible FPGA dans le projet était prévue. Cependant, en raison du manque de maturité du produit, nous avons été contraints de reporter son utilisation pour les développements futurs du laboratoire.

2.4.3 Le commutateur logiciel BMv2

BMv2 [32] est un commutateur logiciel développé en C++ qui s'exécute dans l'espace utilisateur du système d'exploitation pour émuler le chemin de données P4. Il s'agit de la deuxième version du modèle comportemental du commutateur logiciel P4 qui a été introduite pour apporter des solutions aux limitations du p4c-comportemental [33] et répondre aux spécifications du langage P4. Contrairement au p4c-comportemental, le code source de BMv2 est statique et indépendant des programmes P4.

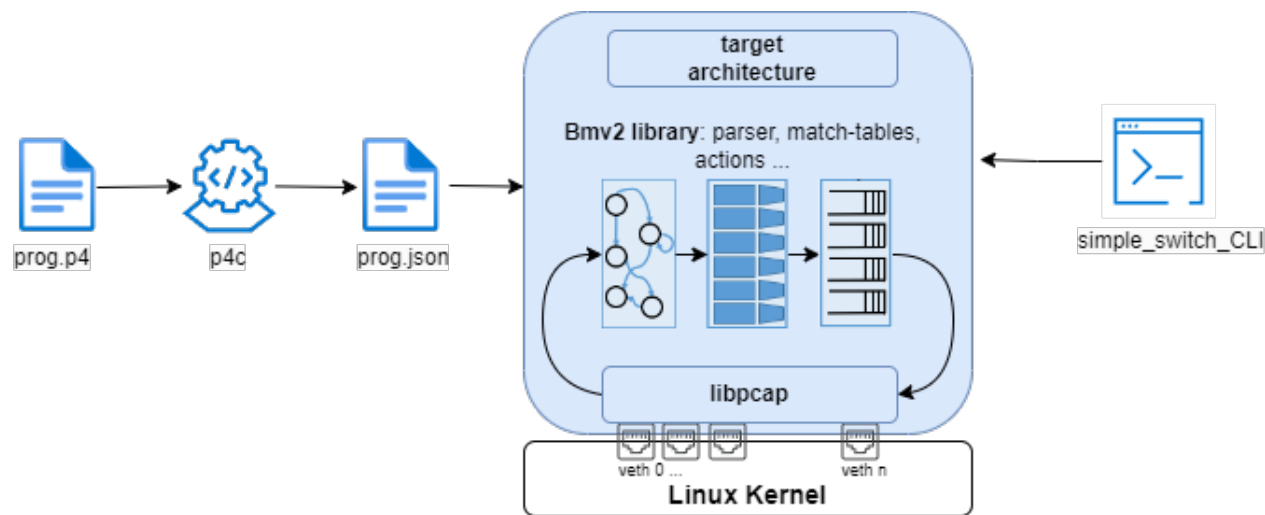


FIGURE 2.4 Flot de travail du BMv2

BMv2 contient plusieurs cibles : *simple_router*, *l2_switch*, *simple_switch*, *simple_switch_grpc* et *psa_switch*. Les deux premières représentent des exemples de différentes cibles qui peuvent être construites avec BMv2. Elles ne prennent en charge qu'une portion des métadonnées standards. Elles ont pour but de montrer comment différentes architectures peuvent être mises en œuvre avec BMv2. Le *simple_switch* est la cible de référence qui vise à mettre en œuvre la spécification complète du P4. Il utilise v1model comme architecture et l'API *Thrift* comme interface pour la communication avec le plan de contrôle. Le *simple_switch_grpc* et *psa_switch* sont similaires au *simple_switch* avec les différences suivantes : le premier utilise l'API P4Runtime basé sur gRPC au lieu de l'API Thrift et le deuxième utilise l'architecture psa au lieu du v1model.

Le flot de travail de BMv2 est résumé dans la Figure 2.3. Le programme P4 est compilé en une représentation JSON par le compilateur p4c. Ce fichier JSON est chargé sur BMv2 et initialise ses structures de données, ce qui permet d'obtenir le comportement de commutation souhaité. BMv2 exécute un serveur Thrift RPC auquel l'interface en ligne de commande

`simple_switch_CLI` peut accéder pour remplir et configurer les tables. La bibliothèque `libpcap` est utilisé afin de recevoir et transmettre des paquets via des interfaces.

BMv2 offre un débit qui peut atteindre jusqu'à 1Gbit/s [34]. Cependant, ce commutateur logiciel ne reste normalement qu'un outil de développement, de test et de débogage. Il n'est pas destiné à des fins de production industrielles. Comme mentionné par le consortium P4 [34], sa performance peut être affectée par un certain nombre de facteurs tels que le matériel, le système d'exploitation, la version du commutateur BMv2 utilisée, le nombre et les tailles des tables d'action-appariement du programme P4. Dans cet environnement, c'est le pipeline d'entrée qui représente souvent un goulot d'étranglement.

2.4.4 Le commutateur Tofino

Un commutateur Tofino est un type de commutateur réseau haute performance développé au départ par la société Barefoot Networks, qui est maintenant une filiale d'Intel. Ce commutateur utilise une architecture de traitement de paquets programmable, qui permet aux utilisateurs de personnaliser et de programmer les fonctionnalités de traitement de paquets en fonction des besoins spécifiques de leur réseau. Le commutateur Tofino est conçu pour être utilisé dans les centres de données et les réseaux de communication haute vitesse, et il peut gérer des débits de données allant jusqu'à plusieurs téraoctets par seconde. Il est également compatible avec différents protocoles de réseau, tels que Ethernet, IPv4, IPv6 et d'autres protocoles personnalisés.

En plus de sa haute performance, le commutateur Tofino est également connu pour son faible temps de latence, ce qui en fait un choix populaire pour les applications qui nécessitent une faible latence, telles que les jeux en ligne, la réalité virtuelle et augmentée, et la diffusion en direct. L'implémentation du plan des données du Tofino à l'aide du langage P4 représente une intéressante solution programmable de haute performance pour répondre aux besoins fonctionnels de l'AGF. En effet l'ASIC Tofino offre une certaine flexibilité, un haut débit (jusqu'à 6.4 Tb/sec), une latence faible et une consommation d'énergie efficace.

La limitation des ressources mémoires disponibles pour les entrées des tables appariement/action représente un réel défi, surtout que l'AGF doit supporter un nombre important d'abonnés. Une approche pour gérer ce problème consiste à utiliser une plateforme hétérogène composée d'un ASIC Tofino et d'une carte FPGA où un programme P4 est divisé en deux sous-programmes. Comme mentionné plus haut, cette avenue n'a cependant pas été considérée dans ce projet à cause du manque de maturité des outils logiciels dédiés à la conversion P4 en langage de description matérielle telle que VHDL ou Verilog.

2.4.5 Le Modèle Tofino

Le Modèle Tofino (*Tofino Model*) est un commutateur logiciel fourni par Intel et conçu pour émuler le comportement du commutateur ASIC Tofino comme le ferait le BMv2, avec une meilleure fidélité quant au comportement de l'ASIC. Il permet aux développeurs de tester et de valider leurs programmes P4 sur des environnements simulés avant de les déployer sur du matériel Tofino réel. Le modèle Tofino offre une flexibilité et une facilité de développement en permettant de bénéficier des caractéristiques matérielles du Tofino, tout en s'exécutant dans un environnement logiciel virtuel.

2.5 Convergence des réseaux filaire et mobile

2.5.1 La 5G et la FMC

La 5G est la plus récente génération de la norme de communication mobile qui a été déployée. Elle vise une connectivité plus rapide, des latences plus faibles et une capacité accrue pour connecter des milliards d'appareils. Les réseaux 5G seront constitués notamment de petites cellules qui peuvent être déployées dans les zones à forte densité de trafic, telles que les centres urbains, les stades et les centres commerciaux [3]. Une nouveauté importante de la 5G est la liaison de l'accès internet fixe dans le cœur du réseau 5G, permettant la convergence des réseaux fixes et mobiles (FMC). La FMC permet d'offrir des services transparents et cohérents afin d'améliorer l'expérience utilisateur et d'améliorer la gestion du trafic réseau. La FMC promet des économies importantes de coût et un gain en efficacité opérationnelle. À terme, elle permettra d'assurer aux utilisateurs des combinaisons plus riches et une continuité de service réutilisable indépendamment du type d'accès réseau.

2.5.2 UPF

L'UPF (*User Plane Function*) est un élément clé de la 5G. Il s'agit d'une fonction responsable du traitement du trafic de données provenant des appareils des utilisateurs. L'UPF assure le routage et l'acheminement des paquets de données entre les appareils connectés et les services de réseau appropriés. En tant que c'est le point d'entrée du trafic de données dans le cœur du réseau, l'UPF gère également des fonctions de sécurité importantes, telles que le contrôle de l'accès des utilisateurs et la protection contre les menaces potentielles [35].

2.5.3 BNG

Le *Broadband Network Gateway* (BNG) est un élément clé des réseaux à large bande et il joue un rôle essentiel dans la gestion des connexions d'accès des utilisateurs. C'est un dispositif qui agit comme une interface entre le réseau d'accès, généralement un réseau à large bande, et le réseau cœur de l'opérateur. Le BNG assure le contrôle de l'accès des utilisateurs au réseau, notamment en gérant l'authentification, l'autorisation et la comptabilité (*Authentication, Authorization, and Accounting* : AAA) des abonnés. Il offre également des fonctions de contrôle de flux, de qualité de service (QoS) et de gestion de la bande passante pour optimiser l'utilisation des ressources réseau. Le BNG est crucial pour permettre la fourniture de services Internet haut débit aux abonnés, en prenant en charge de manière efficace et sécurisée un grand nombre de connexions simultanées.

La conception et la mise en œuvre d'un BNG performant sont d'une importance capitale pour les opérateurs de réseaux à large bande, car il doit être capable de gérer le trafic de manière efficace, tout en garantissant une expérience utilisateur de qualité. En raison de l'évolution des besoins des utilisateurs et des avancées technologiques, le BNG évolue constamment pour prendre en charge de nouvelles fonctionnalités et s'adapter aux exigences croissantes de la connectivité haut débit. Il est crucial que les BNG soient conçus pour être évolutifs et flexibles, de sorte que des implémentations P4 du BNG ont déjà été proposée [7, 36, 37]

2.5.4 AGF

Historiquement, les passerelles BNG ont été utilisées pour fournir l'accès réseau résidentiel, offrant des performances élevées, mais leur mise en œuvre est coûteuse. Cependant, dans l'architecture du réseau 5G basée sur le SDN et la NFV, les BNG ne semblent pas adaptées pour assurer l'accès fixe au cœur du réseau 5G, car elles ne prennent pas en charge les interfaces essentielles pour communiquer avec le réseau 5G et n'offrent pas les fonctionnalités nécessaires pour gérer le trafic des utilisateurs finaux.

En conséquence, la passerelle d'accès (*Access Gateway Function* : AGF) s'est imposée comme une alternative à la BNG dans le réseau 5G [8]. L'AGF propose une solution plus complète, spécialement conçue pour prendre en charge les interfaces nécessaires pour interagir avec le plan de données et le plan de contrôle du réseau 5G. De plus, l'AGF répond aux autres exigences essentielles du cœur de la 5G, telles que la qualité de service et la gestion du trafic et de l'accès.

Ce travail de recherche s'inscrit dans ces efforts déployés pour réaliser la vision de la convergence des réseaux fixes et mobiles, en garantissant une transition harmonieuse et une inter-

opérabilité optimale entre les deux types de réseaux. Aucune implémentation P4 de l'AGF n'est rapportée dans la littérature, faisant du travail présenté ici une contribution originale à l'état de l'art dans ce domaine.

2.5.5 Qualité de service

La qualité de service (*Quality of Service* : QoS) est un autre aspect important de la 5G. Pour répondre aux exigences de la qualité de service de différentes tranches de réseau, il est important d'avoir une implémentation de l'AGF sur un plan de données programmable sensible à la qualité de service du trafic 5G. Le langage P4 permet la classification des flots des paquets et ainsi la mise en oeuvre de la QoS via la programmabilité du plan de données [38]. Dans ce mémoire, nous nous contentons de travailler avec le gestionnaire de trafic du Tofino, plus simple que le système H-QoS (*Hierarchical Quality of Service*) basé sur FPGA proposé par Kundel *et al.* [7], malgré le fait que notre solution ne supporterait pas l'expression d'ordonnancement des paquets.

2.6 Éléments de méthodologie

2.6.1 Mise en oeuvre du plan de données en P4

La conception et l'implémentation efficace du plan de données en P4 sont essentielles pour offrir une expérience utilisateur optimale dans les réseaux. Il a été proposé une mise en oeuvre du plan de données en utilisant deux programmes P4 : l'un servant de modèle de plan de données ouvert pour faciliter l'interface avec le plan de contrôle, et l'autre est optimisé pour les commutateurs matériels afin de minimiser la latence et de maximiser la bande passante [39]. Cette approche permet de tester les changements apportés au plan de contrôle avant leur intégration avec le data plane performant, favorisant ainsi la flexibilité et l'efficacité dans l'évolution des réseaux mobiles.

2.6.2 Exécution sur des cibles hétérogènes

Une implémentation de programmes P4 nécessite généralement un déploiement sur une plateforme hétérogène pour s'adapter aux divers matériels de commutation disponibles. En effet, les commutateurs réseau programmables peuvent varier considérablement en termes de capacités matérielles et de fonctionnalités supportées, ce qui implique que les programmes P4 doivent être conçus de manière à être exécutés efficacement sur différentes plateformes afin d'exploiter au mieux les caractéristiques spécifiques de chaque dispositif.

Considérons à titre d'exemple l'implémentation de l'UPF en P4. En raison de la nature hautement distribuée et du trafic de données massif attendu dans les réseaux 5G, l'UPF doit être capable de gérer un grand nombre de connexions simultanées et de faire face à une variété de scénarios d'utilisation. L'évolutivité, la flexibilité et l'efficacité sont des considérations clés lors de la conception de l'UPF pour permettre une connectivité rapide et fiable dans le paysage en constante évolution des services de communication mobiles.

Kumar *et al.* ont ainsi proposé des conceptions hybrides de pipelines pour l'UPF [40]. Pour ce faire, ils exploitent les fonctionnalités spécifiques aux cibles matérielles ainsi que les possibilités offertes par P4. L'UPF, entière ou désagrégée, s'exécute sur des cibles P4 et alloue les chemins de traitement des paquets dans le matériel P4 ou dans le logiciel DPDK/x86 en fonction des caractéristiques des flux et des exigences de QoS. Cette approche hybride permet d'améliorer la mise à l'échelle jusqu'à 18 fois et de réduire la latence jusqu'à 50%. En outre, l'application orientée QoS de l'UPF hybride est démontrée avec des benchmarks de services de mémoire tampon pour le gNodeB. Ces travaux soulignent l'importance de combiner les avantages de P4 et du matériel spécifique pour atteindre de hautes performances et pour offrir une gestion efficace des flux dans l'UPF et les réseaux mobiles de nouvelle génération.

2.6.3 Évaluation de performance et utilisation des ressources

L'évaluation de la performance et l'utilisation des ressources d'une solution P4 pour un commutateur réseau programmable revêt une importance cruciale. Les solutions P4 conçues pour les commutateurs programmables doivent relever le défi d'exploiter efficacement les ressources matérielles disponibles, telles que le nombre de blocs d'association/action. Il est donc essentiel d'identifier les meilleures approches pour accélérer les éléments du plan utilisateur tout en respectant les limites du dispositif P4 visé. En outre, il est nécessaire d'évaluer les capacités de flux du dispositif et de mesurer avec précision le temps de transit des liaisons, fournissant ainsi une estimation précise des capacités du dispositif et de l'efficacité du code qui y est exécuté. Ces évaluations sont cruciales pour optimiser les performances et garantir une utilisation efficace des ressources dans les réseaux modernes [41].

Cependant, répondre aux exigences de qualité de service dans les réseaux 5G, notamment en ce qui concerne la faible latence, les débits élevés et la disponibilité, constitue un véritable défi. Harkous *et al.* [42] mettent en évidence l'importance d'évaluer les performances des dispositifs programmables P4 en se concentrant sur deux aspects clés. Premièrement, ils examinent la latence de traitement des dispositifs P4 en fonction du nombre de flux, afin de déterminer les limites de chaque dispositif en termes du nombre maximum de flux qu'il peut prendre en charge. Deuxièmement, ils analysent le temps de réponse des dispositifs P4 en réaction

aux commandes du plan de contrôle, ce qui permet d'évaluer leur réactivité globale face aux demandes du réseau. Ces évaluations sont essentielles pour identifier les performances réelles des dispositifs P4 et pour s'assurer qu'ils répondent aux besoins en termes de capacité et de réactivité dans les environnements de communication de nouvelle génération.

Al Sadi *et al.* [43] abordent l'importance de l'évaluation précise de la performance des réseaux, en particulier la mesure du délai des liaisons. Les auteurs proposent une solution basée sur le langage de programmation P4 pour mesurer le temps de transit des liens en utilisant des paquets de sonde légers. Cette approche permet d'obtenir des mesures précises et fiables du temps de transit des liens, même dans les liaisons multi-sauts. Comparé à d'autres approches de mesure, l'utilisation de P4 permet d'obtenir des résultats plus robustes et précis, même dans des conditions de réseau congestionné.

CHAPITRE 3 MÉTHODE PROPOSÉE POUR LA MISE EN ŒUVRE DU PLAN DE DONNÉES DE LA FONCTION PASSERELLE D’ACCÈS AGF

3.1 Introduction

Ce chapitre décrit la méthodologie suivie lors de notre travail de recherche pour concevoir et implémenter le plan de donnée de l’AGF. Le chapitre est divisé en trois parties. Après la description des méthodes utilisées dans la revue de la littérature utile pour notre projet, il convient de commencer par fournir une explication détaillée de l’AGF à partir des documents de normalisation et des spécifications techniques. La deuxième partie décrit l’environnement de développement virtuel P4 ainsi que les outils de test et de déverminage qui ont été utilisés pour valider la fonctionnalité du prototype AGF. Ceci constitue un point de départ utile pour une implémentation d’un plan de données AGF performant sur un commutateur matériel Intel Tofino décrite dans la quatrième partie. Des outils de génération de trafic et de test de performance sont aussi présentés.

3.2 Analyse fonctionnelle de l’AGF

La méthodologie adoptée pour la mise en œuvre de l’AGF découle des spécifications techniques définies conjointement par les organismes de normalisation BBF (*Broadband Forum*) et 3GPP (*3rd Generation Partnership Project*). Ces spécifications définissent le modèle fonctionnel et le cadre architectural de l’AGF [8, 35, 44].

3.2.1 Architecture traditionnelle des réseaux d’accès fixes et FMC



FIGURE 3.1 Architecture d’accès fixe filaire traditionnelle

Afin de comprendre les différents scénarios de déploiement de l’AGF pour réaliser la convergence FMC, commençons par présenter l’architecture traditionnelle des réseaux fixe. Dans la Figure 3.1, le BNG assure la liaison entre les dispositifs d’accès et le réseau des fournisseurs d’accès Internet. La fonction BNG a été spécifiée par le BBF dans plusieurs rapports techniques (TR-101, TR-105 et TR-145) [45–47]. Elle est responsable d’établir et de gérer les

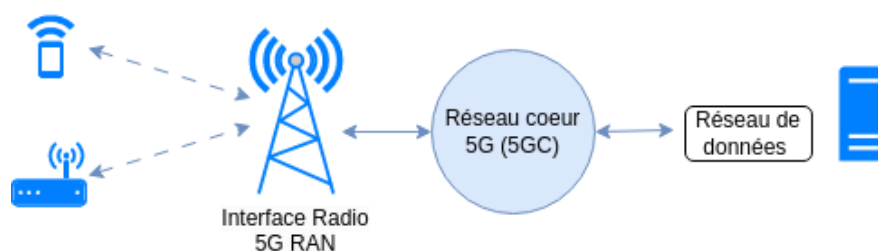


FIGURE 3.2 Architecture d'accès réseau mobile

sessions des abonnés et regroupe le trafic de diverses sessions d'abonnés à partir du réseau d'accès et l'achemine vers le réseau du fournisseur de services. Cette infrastructure d'accès fixe est totalement séparée du réseau mobile 5G. Elle est illustrée à la Figure 3.2, où les appareils mobiles et les passerelles 5G sont liés au réseau coeur 5G via des interfaces radio 5G RAN.

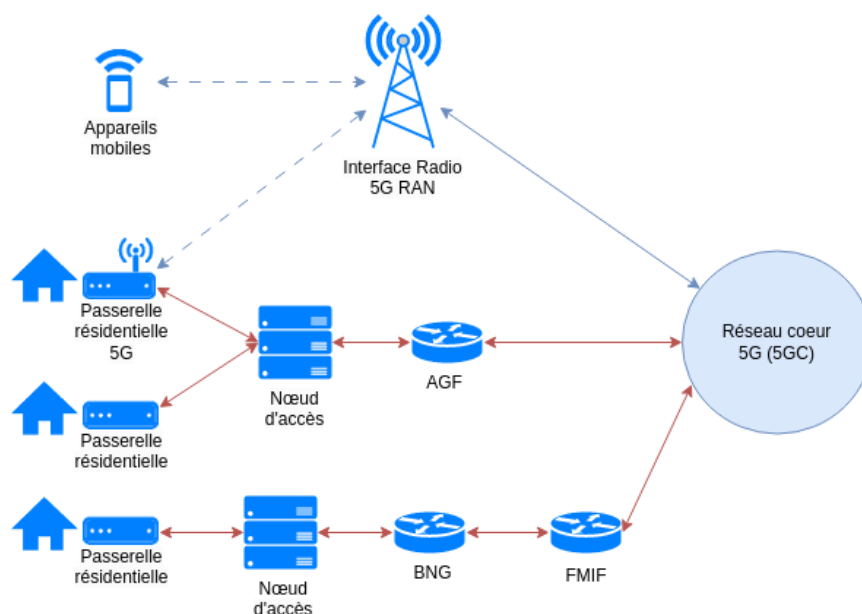


FIGURE 3.3 Différentes approches permettant la convergence fixe mobile

De multiples approches permettent la convergence entre ces deux types d'accès réseaux fixes et mobiles, ont été proposées, ce que résume la Figure 3.3. Une première approche consiste à introduire la fonction d'inter-fonctionnement fixe-mobile (*Fixed Mobile Interworking Function* : FMIF), qui établit une connexion entre la BNG et le cœur de réseau 5G. Cette approche est particulièrement appropriée pour les fournisseurs de services qui ont déjà déployé leur BNG et qui souhaitent apporter des modifications minimales à leur architecture existante. Une

autre approche consiste à mettre en œuvre la fonction de passerelle d'accès (*Access Gateway Function* : AGF), sujet de notre projet. L'AGF remplace la BNG et permet aux fournisseurs de services une intégration plus poussée de leurs réseaux d'accès filaires existant avec le cœur de réseau 5G. Elle supporte à la fois les passerelles résidentielles traditionnelles (FN-RG) et les nouvelles passerelles résidentielles 5G (5G-RG). Ces dernières sont améliorées pour être compatibles avec le plan de contrôle de la 5G. Elle peuvent communiquer directement avec le cœur de la 5G par l'intermédiaire de n'importe quel réseau d'accès. Les différents scénarios de déploiement et modes d'opération de l'AGF sont détaillés dans la section suivante.

3.2.2 Scénarios de déploiement de l'AGF et modes d'opération

On distingue deux modes d'opérations de l'AGF illustrés dans les Figures 3.4 et 3.5 : le mode direct où le 5G-RG est pris en charge et le mode adaptatif où le FN-RG est pris en charge. Pour le premier mode, le 5G-RG se connecte via le réseau d'accès filaire et l'AGF assure la liaison au réseau central 5G à l'aide des interfaces N2 et N3. Le 5G-RG est capable d'accomplir les fonctions NAS via l'interface N1 : Il peut s'authentifier et s'enregistrer directement au réseau central 5G.

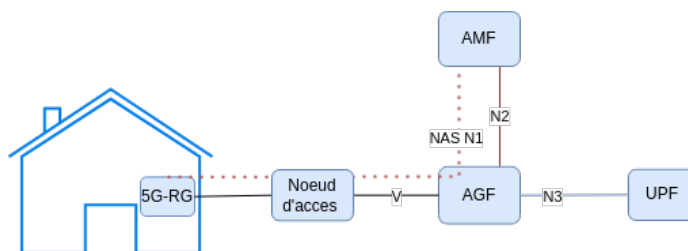


FIGURE 3.4 AGF en mode directe

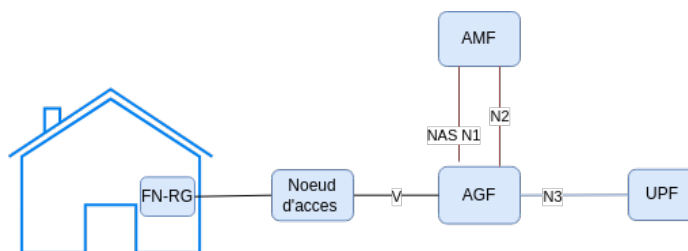


FIGURE 3.5 AGF en mode adaptatif

L'AGF peut aussi s'intégrer au système 5G aussi en mode adaptatif. Ceci est similaire au premier mode, dans la mesure où le FN-RG se connecte via le réseau d'accès filaire et l'AGF

assure la liaison au réseau central 5G à l'aide des interfaces N2 et N3. Toutefois, contrairement au mode direct, le FN-RG ne prend pas en charge l'interface N1. L'AGF agit donc en tant que point final de l'interface N1 au lieu du FN-RG et ainsi assure les fonctions NAS.

A noter qu'une seule instance AGF peut opérer simultanément en mode direct (5G-RG) et en mode adaptatif (FN-RG). Dans notre travail, nous nous focalisons plus sur le mode direct.

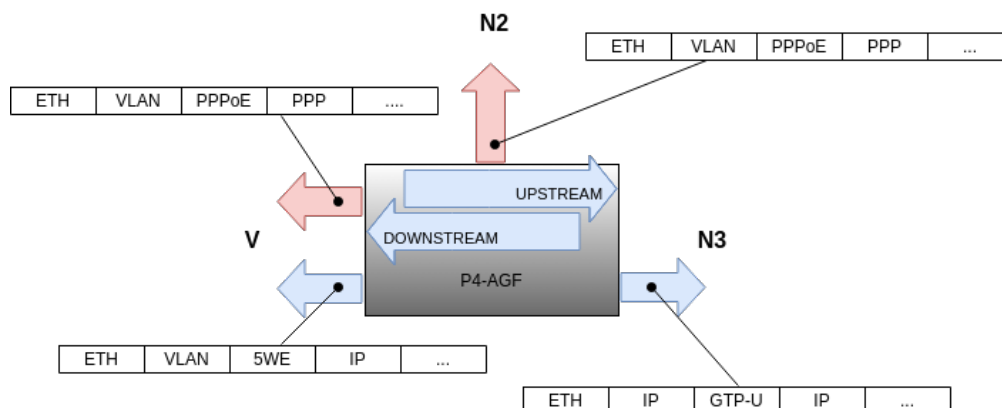


FIGURE 3.6 Pile protocolaire AGF

3.2.3 Pile de protocoles AGF

Le plan de données de l'AGF doit prendre en charge une variété de protocoles pour fonctionner correctement, tel qu'illustré à la Figure 3.6. La pile de protocoles varie en fonction du type de trafic et de l'interface concernée. On distingue deux types de trafic : le trafic du plan de contrôle échangé entre les interfaces V et N2 ; et le trafic du plan de données échangé entre l'interface V et N3.

1. Pile protocolaire pour le trafic du plan de contrôle (interfaces V et N2) : Le trafic du plan de contrôle est minoritaire et uniquement nécessaire lors de l'authentification et de l'établissement du session. Pour ce type de trafic, la pile protocolaire est composée de : Ethernet, Double VLAN, PPPoE et PPP.
2. Pile protocolaire pour le trafic du plan des données (interfaces V et N3) : Le trafic du plan de données représente la majorité du trafic et transporte les données des utilisateurs. Pour ce type de trafic, la pile protocolaire dépend de l'interface. Du côté de l'interface V, elle est composée de : Ethernet, Double VLAN et 5WE [12]. Alors que du côté de l'interface N3, elle est composée de : Ethernet, IP et GTP-u.

3.3 Environnements de développement et de test de l'AGF

Pour réussir la mise en œuvre d'un 5G-AGF fonctionnel, il est essentiel de bien comprendre et analyser les exigences fonctionnelles à remplir par l'AGF. La présente section fournit une analyse de ces requis fonctionnelles telles qu'elles sont spécifiées dans la documentation technique [8, 35, 44]. Après avoir acquis une compréhension approfondie des outils de l'environnement de développement P4, nous avons pu entamer la mise en œuvre du plan de données de l'AGF, conformément aux spécifications techniques évoquées dans la section précédente.

Le Chapitre 4 fournit une description plus complète de cette mise en œuvre. Nous décrivons ici les environnements de développement et de test de notre prototype AGF. Ceux-ci ont été impliqués soit dans la réalisation du prototype sur les commutateurs logiciels (BMv2, Tofino Model), le commutateur matériel (Tofino) ou les deux.

3.3.1 Outils logiciels

Afin de tester la fonctionnalité du programme AGF, nous devons générer des paquets adaptés aux différents scénarios de test. Pour se faire on utilise Scapy [48]. Il s'agit d'une bibliothèque Python très puissante et flexible qui permet de créer, manipuler, forger, émettre et recevoir des paquets pour une multitude de protocoles. Elle permet aussi d'effectuer les tâches classiques liées aux réseaux comme la découverte de réseaux, le scan, le traceroute, le routage, le traçage, les tests unitaires, les attaques réseaux...

L'outil tcpdump [49] a été aussi utilisé pour inspecter les paquets entrants et sortants au niveau de chaque port du commutateur. Cet outil de ligne de commande, couramment utilisé dans l'industrie des réseaux, nous a permis d'examiner les protocoles de chaque paquet ainsi que les enregistrer dans un fichier pour une analyse ultérieure.

Un avantage important de l'implémentation du prototype du plan de données de l'AGF sur un commutateur logiciel comme le BMv2 est la possibilité d'analyser les journaux (logs) générés par le modèle de commutateur. Cet aspect de journalisation du BMv2 a été très utile car il permet de suivre en détail et de comprendre comment les paquets sont traités et transformés à l'intérieur du commutateur selon la logique définie dans le programme P4. Ceci facilite le processus de développement et de débogage.

3.3.2 Tests de performance physiques

Pour évaluer les performances des dispositifs réseau, des générateurs de trafic réseau sont utilisés. Il s'agit des outils logiciels ou matériels qui permettent de générer un trafic réseau

important et de fournir des mesures relatives à la latence, au débit et à d'autres métriques.

Une variété de générateur de trafic open source existent sur le marché, et peuvent même rivaliser avec les solutions propriétaires en termes de fonctionnalités et d'efficacité [50]. Parmi ces solutions on cite : T-Rex [51], Pktgen [52] et Moongen [53]. Ces outils sont basés sur le framework DPDK et sont capables de générer du trafic réseau important avec un débit élevé.

Dans notre projet, nous avons choisi d'utiliser T-Rex en raison de sa flexibilité et de ses diverses fonctionnalités. En effet, T-Rex prend en charge de nombreux protocoles et permet de personnaliser facilement le trafic généré. Étant basé sur DPDK, il permet de générer des paquets à grand débit avec une utilisation raisonnable de CPU [51]. De plus, Il est conçu pour fonctionner plus efficacement avec certains types de cartes réseau, plus spécifiquement la carte "Intel Ethernet Network Adapter XXV710" [54] que nous avons à notre disposition au laboratoire.

	Average	0	1	2	3	4	5	6	7	8	9	10	11
Action Data Bus Bytes	6.3%	1.6%	1.6%	0.0%	17.2%	17.2%	4.7%	4.7%	4.7%	7.8%	12.5%	1.6%	1.6%
8-bit Action Slots	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
16-bit Action Slots	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
32-bit Action Slots	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Exact Match Input Xbar	7.6%	7.0%	10.9%	9.4%	6.3%	1.6%	6.3%	6.3%	6.3%	10.9%	9.4%	11.7%	4.7%
Gateway	3.1%	12.5%	6.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	6.3%	6.3%	6.3%	0.0%
Hash Bit	17.7%	12.5%	24.0%	16.8%	20.4%	10.3%	11.5%	11.5%	11.5%	25.5%	30.8%	24.5%	12.5%
Hash Dist Unit	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Logical Table ID	13.0%	18.8%	12.5%	6.3%	25.0%	6.3%	6.3%	6.3%	6.3%	25.0%	25.0%	12.5%	6.3%
Map RAM	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Meter ALU	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
SRAM	97.0%	100.0%	100.0%	100.0%	100.0%	96.3%	100.0%	100.0%	100.0%	96.3%	100.0%	100.0%	71.3%
Stash	14.1%	12.5%	18.8%	12.5%	18.8%	6.3%	12.5%	12.5%	12.5%	18.8%	18.8%	18.8%	6.3%
Stats ALU	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
TCAM	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Ternary Match Input Xbar	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
VLIW Instruction	8.3%	6.3%	9.4%	3.1%	9.4%	6.3%	6.3%	6.3%	6.3%	18.8%	12.5%	9.4%	6.3%
Exact Match Search Bus	12.0%	18.8%	18.8%	6.3%	12.5%	6.3%	6.3%	6.3%	6.3%	18.8%	25.0%	12.5%	6.3%
Exact Match Result Bus	9.9%	6.3%	12.5%	6.3%	12.5%	6.3%	6.3%	6.3%	6.3%	18.8%	18.8%	12.5%	6.3%
Ternary Result Bus	2.6%	6.3%	0.0%	0.0%	12.5%	0.0%	0.0%	0.0%	0.0%	6.3%	6.3%	0.0%	0.0%

FIGURE 3.7 Capture d'écran de l'outil de visualisation P4I

3.3.3 Prototypage sur l'ASIC

Étant donné les contraintes auxquelles les commutateurs ASIC programmables sont confrontés en termes de ressources disponibles, il est crucial d'évaluer le taux d'utilisation des ressources matérielles par notre prototype implémentant la fonctionnalité AGF. Pour ce faire, nous utilisons dans notre projet l'outil P4 Insight (Figure 3.7), une solution proposée par Intel, qui permet une analyse complète d'allocation des ressources matérielles des programmes P4 compilés pour les ASIC Intel Tofino. Grâce à l'outil Intel P4 Insight, nous pouvons obtenir des statistiques détaillées et des informations critiques sur l'utilisation des ressources et de la mémoire, ce qui nous aide à procéder à des optimisations afin de garantir une utilisation plus efficace des ressources matérielles, particulièrement les mémoires SRAM et TCAM.

CHAPITRE 4 CONCEPTION ET IMPLÉMENTATION DE LA FONCTION AGF - BASÉE SUR LE LANGAGE P4 - FLEXIBLE DE HAUTE PERFORMANCE

Dans ce chapitre, nous décrivons l'architecture de notre solution et nos choix de conception pour la réalisation de la fonction AGF. Nous introduisons d'abord la conception générale du plan de donnée de l'AGF. Pour ce faire, nous présentons le système d'encapsulation et de désencapsulation des en-têtes des paquets d'une part, et décrivons le pipeline de traitement des paquets après l'extraction de ces en-têtes d'autre part. Nous décrivons ensuite les différentes implémentations du plan de données pour les commutateurs BMv2 et Tofino. Finalement, nous décrivons l'implémentation du plan de contrôle de l'AGF.

4.1 Conception du prototype AGF

4.1.1 Encapsulation/désencapsulation des en-têtes du paquet

Comme illustré dans la Figure 4.1, deux types de trafic doivent être traités par le plan de données de l'AGF : le trafic de contrôle et le trafic usager. Le trafic de contrôle est encapsulé dans des paquets PPPoE. Le trafic usager est encapsulé dans des paquets 5WE.

- Trafic de contrôle : L'encapsulation PPPoE permet de transporter les signalisations NAS et AS lors de l'établissement de session. C'est la partie plan de contrôle de l'AGF (AGF-CP) qui est responsable de gérer ce type de trafic. Le plan de donnée se limite à extraire le champ d'en-tête PPPoE pour prendre des décisions concernant son acheminement au port convenable ou son rejet.
- Trafic usager : Un tunnel est établi entre le RG et l'AGF d'une part et l'AGF et l'UPF d'autre part pour chaque abonné, ou plus précisément pour chaque session PDU (*Protocol Data Unit PDU*). (Un seul abonné pourrait avoir plusieurs sessions PDU établies simultanément [44]). La Figure 4.2 illustre la pile protocolaire pour l'encapsulation des en-têtes des paquets.

Les paquets ascendants (*upstream*) qui proviennent du côté de l'abonné vers le réseau du fournisseur d'accès, sont encapsulés par le RG dans un tunnel 5WE. De plus, le RG ajoute un tag VLAN (s-tag) pour identifier le type de service, et modifie les adresses Ethernet source et de destination. Le paquet est ensuite transmis au nœud d'accès, où un deuxième tag VLAN (c-tag), qui représente le *Line ID*, lui est ajouté. Enfin, l'AGF reçoit le paquet avec tous les en-têtes ajoutés pour les traiter.

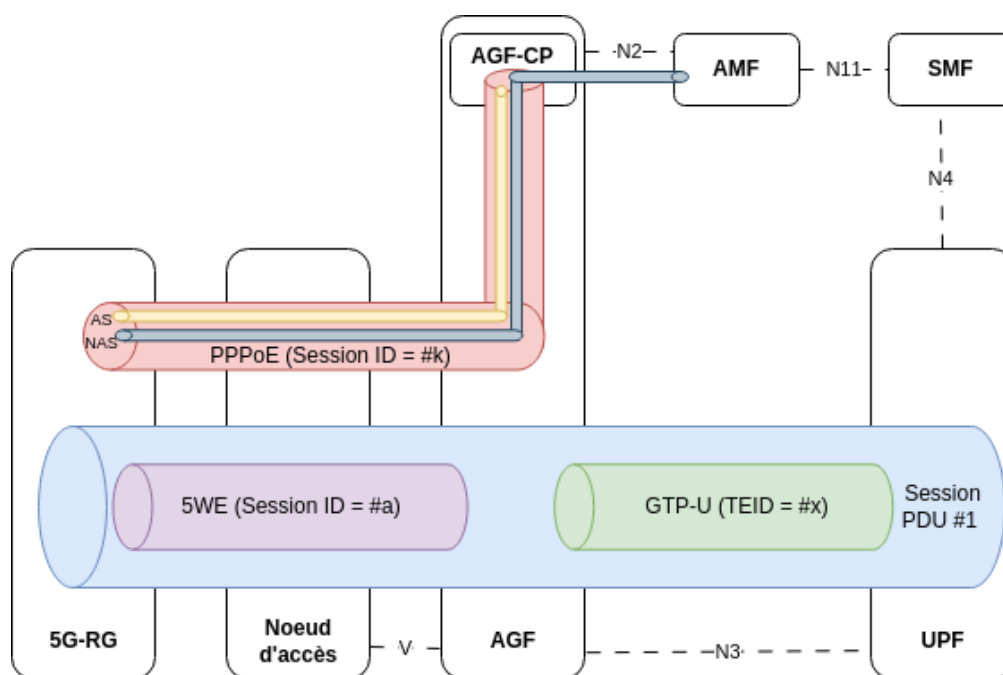


FIGURE 4.1 Différents types de trafic AGF : trafic de contrôle et trafic usager, adapté de [44]

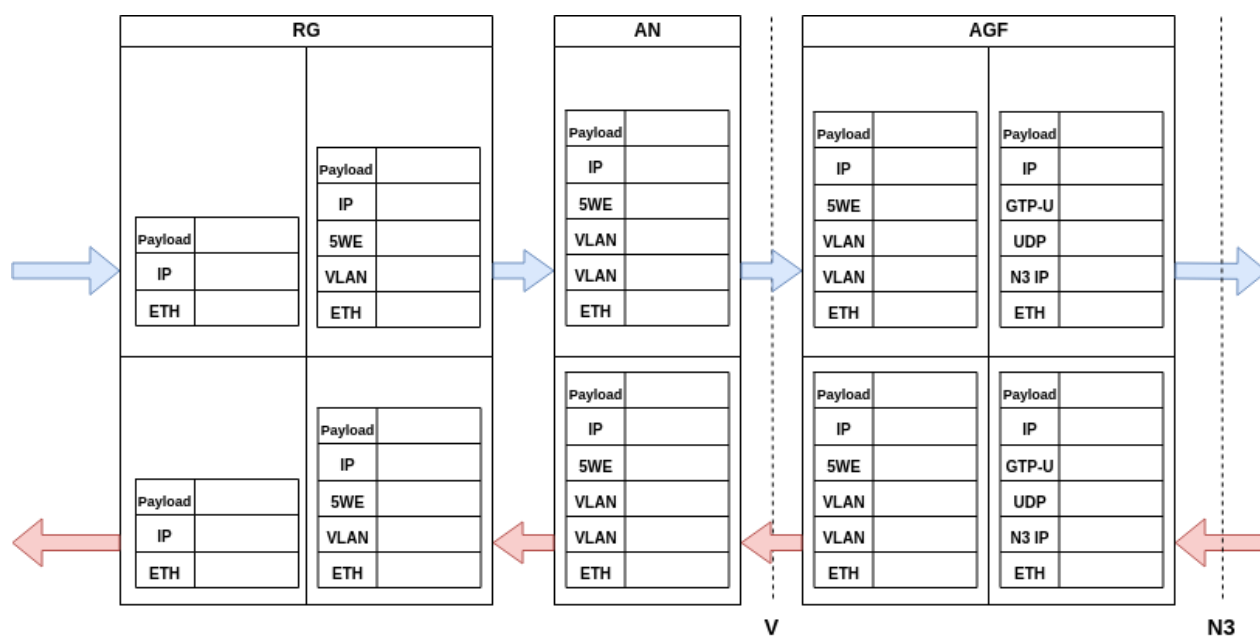


FIGURE 4.2 Conception d'un système d'encapsulation/désencapsulation des en-têtes des paquets Upstream et Downstream via la fonction AGF

De l'autre côté, les paquets descendant (*downstream*) provenant du réseau de données sont encapsulés par l'UPF dans un tunnel GTP-U (*GPRS Tunneling Protocol in the User Plane*). Le GTP-U étant le protocole utilisé pour la mise en tunnel au niveau du plan utilisateur (plan de données) du réseau coeur 5G. Il est basé sur les protocoles IP/UDP et peut transporter non seulement des paquets IP tels que IPv4, IPv6 mais aussi des paquets Ethernet que la 5G prend en charge aussi [55].

Pour les deux types de trafic *Upstream* et *Downstream*, il est essentiel que l'AGF extrait et analyse correctement les en-têtes des paquets entrants afin de permettre le traitement ultérieur. Cependant, ces protocoles (surtout le 5WE) ne sont pas pris en charge par le matériel réseau standard. Ce qui représente un réel défi. Par conséquent, le langage de programmation P4 et ses architectures matérielles cibles offrent un niveau de flexibilité que nous exploitons dans ce projet.

Du côté du réseau d'accès, le principal défi réside dans le traitement du protocole 5WE. Un paquet 5WE est transporté dans une trame Ethernet, et peut être identifié par la valeur du champ EtherType = 0x8863. La structure d'un paquet 5WE est illustrée dans la Figure 4.3. Nous définissons cette structure au niveau du code P4 et spécifions des règles permettant à la machine à états de l'analyseur d'entêtes (*parser*) d'extraire correctement ce type de paquet.

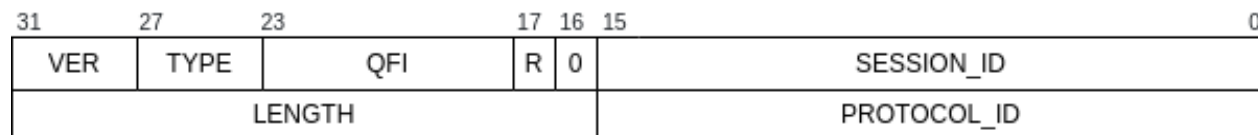


FIGURE 4.3 Format d'en-tête du protocole 5WE pour l'encapsulation des paquets du plan de données selon [12]

Du côté du réseau de données, le protocole GTP-U est assez courant. Il représente de ce fait un défi mineur qui réside principalement dans le fait qu'il est encapsulé dans un segment UDP qui n'indique pas le type du protocole suivant. Dans ce cas pour déterminer le protocole suivant, on utilise plutôt le champ port source UDP qui doit avoir comme valeur 2152 pour le GTP-U. Habituellement, ce port sert à différencier diverses applications de la couche applicative pour le même hôte.

Les Figures 4.4, 4.5 et 4.6 illustrent la structure du protocole GTP-U ainsi que les extensions *PDU Session Container* qui sont aussi déjà présentés en détail dans le chapitre précédent. On distingue deux types d'extension : UL et DL. Ces extensions sont propres à la 5G et ne sont pas présentes dans les normes plus anciennes. Par conséquent, l'AGF doit être en mesure de

l'interpréter convenablement. Semblablement au 5WE, Nous définissons la structure de ces protocoles dans le code P4 (voir l'annexe) et nous spécifions des règles permettant d'extraire correctement ces types d'en-tête par la machine d'état d'analyseur des en-têtes *Parser*.

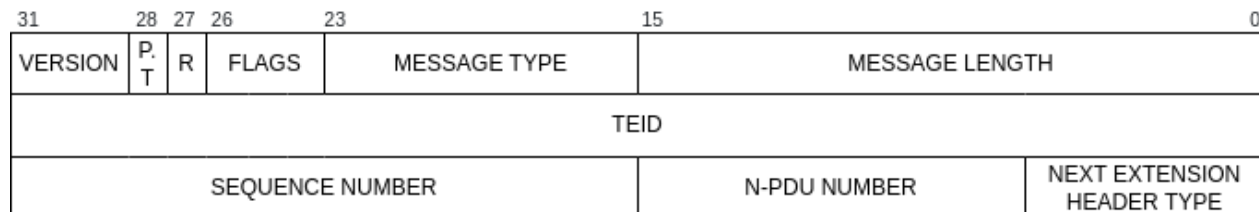


FIGURE 4.4 Format d'en-tête du protocole GTP-U pour l'encapsulation des paquets du plan de données selon [56]

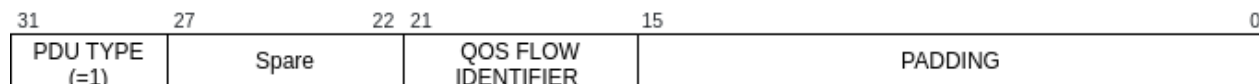


FIGURE 4.5 Format d'en-tête d'extension GTP-U PDU session container pour la direction Upstream selon [57]

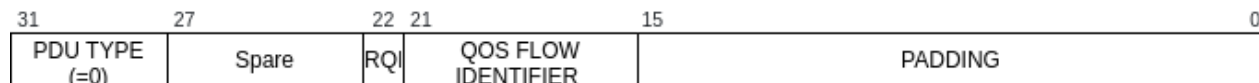


FIGURE 4.6 Format d'en-tête d'extension GTP-U PDU session container pour la direction Downstream [57]

La désencapsulation des paquets *Upstream* peut être effectuée en se basant sur le champ approprié d'en-tête courant pour déterminer le type d'en-tête suivant, tel que l'EtherType pour l'Ethernet et les deux VLANs et le champ 5WE protocole pour le 5WE. Le même principe est appliqué aux paquets *Downstream* mais en se basant sur le champs "Ethertype" pour l'Ethernet, "Protocole" pour l'IP, "Port source" pour l'UDP et "Next extension header" pour le GTP-U. Après l'extraction de ces en-têtes. Ils passent par la pipeline du commutateur P4 où des tables et des actions sont appliquées afin de réaliser la fonctionnalité AGF souhaitée. La conception de ce pipeline est discutée en détail dans la partie suivante. Et finalement le ré-assembleur des entêtes *Deparser* rassemble les en-têtes valides à la fin du pipeline dans le bon ordre pour construire le paquet de sortie.

4.1.2 Conception du pipeline P4

Le flux de contrôle du programme P4 qui met en œuvre la fonction AGF, est présenté dans la Figure 4.7. Par la suite, ce programme peut être mappé par le compilateur sur les ressources du pipeline match-action de la cible programmable en P4. La conception du pipeline proposé est divisée en trois parties :

1. Une partie pour le traitement des paquets du plan contrôle.
2. Une partie pour le traitement des paquets *Upstream*.
3. Une partie pour le traitement des paquets *Downstream*.

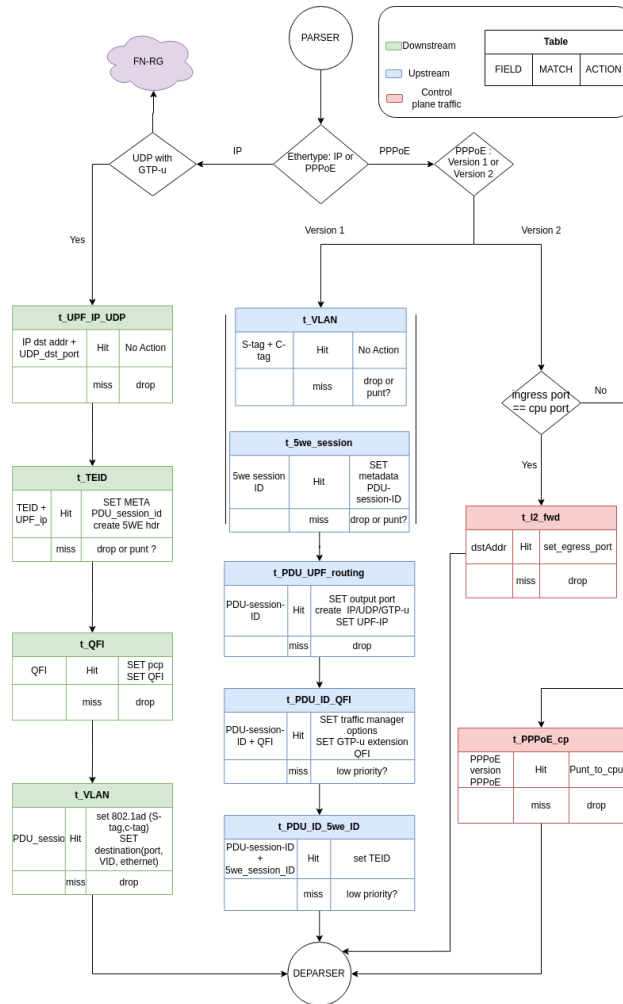


FIGURE 4.7 Conception du pipeline P4 pour l'AGF

La partie responsable du traitement des paquets du contrôle est représentée par une couleur rouge dans le diagramme. Elle comprend deux tables *t_pppoe_cp* et *t_l2_fwd*. D'une part,

la table "*t_pppoe_cp*" examine les champs *Version* et *Code* du PPPoE pour déterminer si le paquet appartient au plan de contrôle de l'AGF. Si *Version* = 1, il s'agit d'un paquet PPPoE du plan de contrôle et il est directement transmis au plan de contrôle (*egress_port* prend la valeur du port CPU) sans aucun traitement supplémentaire par le pipeline P4. Le champ *Code* est aussi examiné et doit correspondre aux valeurs définies par l'IANA [58] selon le type du paquet PPPoE (par exemple PADO = 7, PADI = 9, PADG = 10 ...). Autrement, le paquet est rejeté. D'autre part, des paquets peuvent être aussi injectés à partir du plan de contrôle (via le port CPU). Dans ce cas, la table *t_l2_fwd* est responsable de déterminer le port de sortie (*egress_port*) du paquet en se basant sur son adresse MAC destination. A noter que ce type de trafic, qui est minoritaire par rapport au trafic utilisateur, n'est échangé que pendant l'authentification et l'établissement de session.

La partie responsable du traitement des paquets utilisateur dans la direction *Upstream* est représentée par une couleur bleue. Elle est composée de quatre tables *t_vlan_5we*, *t_pdu_upf_routing*, *t_pdu_id_qfi* et *t_pdu_id_5we_id*.

Dans un premier temps, la table *t_vlan_5we* est appliquée afin de gérer efficacement les sessions PDU des abonnés pour chaque 5G-RG en se basant sur les deux étiquettes VLAN (c-tag et s-tag) et sur le champ "5WE session ID". L'étiquette intérieure c-tag qui représente le Line ID est assignée par le noeud d'accès pour identifier l'abonné, l'étiquette extérieure s-tag est assignée par le RG pour identifier le service et le 5WE ID est assignée par l'AGF pendant l'établissement de session. Ces trois informations sont associées à un identifiant d'une session PDU (PDU session ID). S'il n'y a aucune correspondance, le paquet est rejeté.

Dans un deuxième temps, en se basant sur le PDU session ID, la table *t_pdu_upf_routing* via son action associée construit le paquet GTP-U, assigne une route au paquet (adresse IP de l'UPF destination), modifie le champ ethertype ainsi que les adresses MAC source et destination et définit le port de sortie. Ensuite, la table *t_pdu_id_qfi* est appliquée. Cette table se base sur les deux champs PDU session ID et QFI (du 5WE) pour spécifier le champ QFI de l'extension GTP-U. Le QFI permet d'identifier le flux QoS. Ceci représente la granularité la plus fine de la différenciation de la qualité de service pour une session PDU dans le réseau 5G. Autrement une faible priorité est accordée au paquet (QFI=0). Finalement, la table *t_pdu_id_5we_id* définit le TEID à partir du 5WE session id et du PDU session ID. Ce TEID est utilisé du côté de l'AGF pour la mise en tunnel GTP avec la prochaine UPF (liée à l'AGF via l'interface N3).

La partie responsable de traitement des paquets dans la direction *Downstream* a un comportement opposée de celle du *Upstream*. Elle est représentée par une couleur verte dans la figure 4.7. Et elle comprend les quatre tables *t_upf_ip_dp*, *t_teid*, *t_qfi* et *t_vlan*.

La première table appliquée pour traiter les paquets dans la direction *Downstream* est la table : *t_upf_ip_udp* qui prend comme clés correspondance l'adresse IP destination et le port UDP destination. Cette table n'a aucune action associée mais permet de filtrer les paquets entrants : si l'adresse IP ne correspond pas aux valeurs fournies par le plan de contrôle ou le port UDP ne correspond pas à celui du GTP-U, le paquet est rejeté.

La deuxième table *t_teid* se base sur les champs TEID et l'adresse IP source de l'UPF pour déterminer le "PDU session ID". En cas de correspondance réussie l'action "set_sessionId_5we" permet de construire le champ 5WE est exécutée, sinon le paquet est rejeté.

Ensuite la table *t_QFI* est appliquée. Elle se base sur le champ QFI de l'en-tête d'extension GTP-U PDU Session Container pour déterminer la classe QoS du paquet et ainsi définir les champs PCP de l'entête Ethernet et le QFI de l'entête 5WE.

Finalement, la table *t_vlan* qui se base sur le "PDU Session ID" est appliquée pour spécifier le port destination et remplir les champs EtherType et le double VLAN.

4.2 Implémentation du plan de données

Après avoir finalisé l'étape de conception, nous passons à l'implémentation de notre application en utilisant le langage de programmation P4. L'implémentation du plan de données peut être divisée en deux étapes dépendamment de l'architecture du commutateur utilisé : l'architecture V1model dans le cas du commutateur logiciel BMV2 et l'architecture TNA dans le cas du commutateur matériel Tofino. Il convient de noter que les deux architectures ont plusieurs similitudes et peuvent être interopérables.

Cette section concerne la logique d'implémentation du plan de données basée sur l'architecture V1model pour le commutateur logiciel BMV2 et sur l'architecture TNA pour le commutateur matériel Tofino. On commence par une description de l'implémentation de la machine d'état du "Ingress/Egress Parser" puis on passe à la description de l'implémentation du "Ingress/Egress control blocks". Il convient de noter qu'une description générale sera faite en mettant l'accent plus sur le traitement du trafic Upstream et celui du plan de controle.

4.2.1 Parser

Le rôle principal du Parser est d'extraire les champs d'en-tête des paquets entrants et de les placer dans des métadonnées d'en-têtes définies dans le programme P4. Il s'agit une machine à états finis qui contient un état initial "Start" et deux états finaux "Accept" et "Reject".

La Figure 4.8 décrit la structure du Igress Parser pour les paquets du *Upstream* et du *Control*

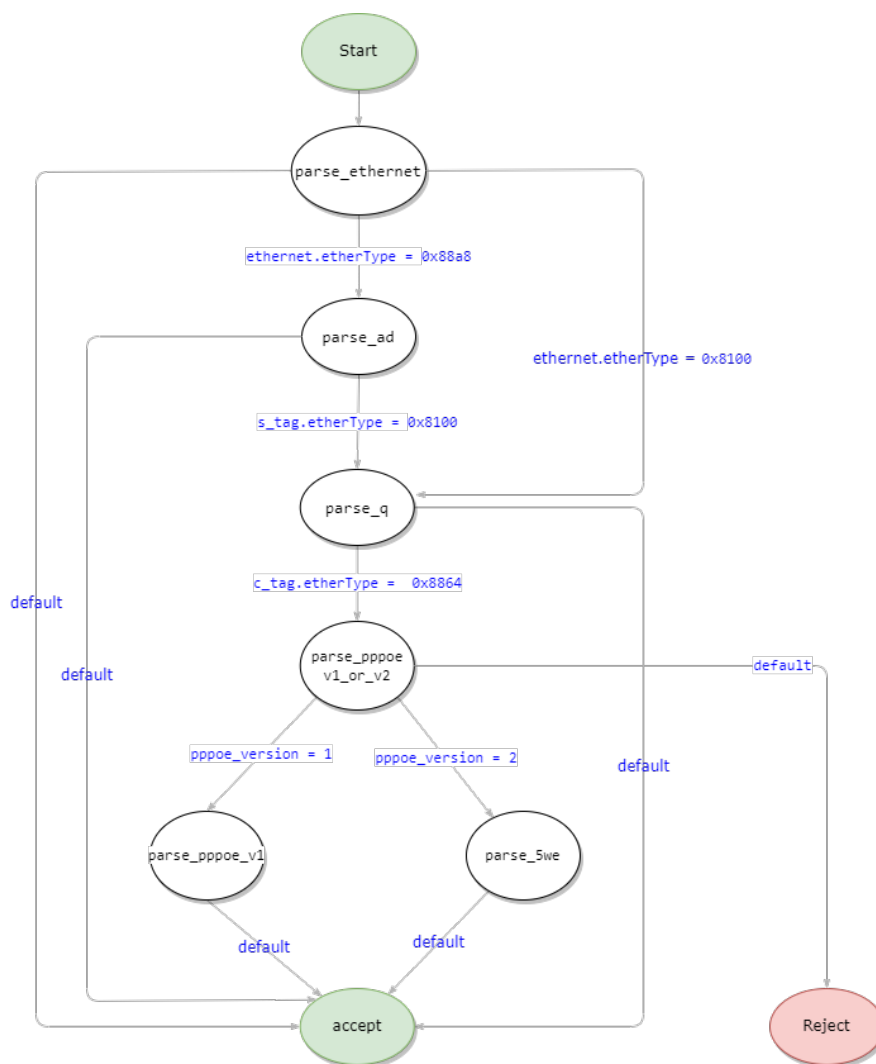


FIGURE 4.8 Analyseur d'entêtes.

Plane. Les transitions entre les états peuvent être conditionnelles ou pas. Lorsque le paquet est arrivé au port du commutateur, l'en-tête du Ethernet est extrait. Si la valeur du champ Ethertype correspond à celle du IEEE 802.1Q (0x8100), on passe à l'en-tête suivant : le s-tag. Un traitement similaire est effectué au niveau du champ Ethertype du s-tag, si sa valeur correspond à celle du IEEE 802.1ad (0x8864), on passe à l'état suivante. Ensuite, le champ version est extrait pour déterminer s'il s'agit d'un paquet *Upstream* du plan de données ou d'un paquet qui appartient au plan de contrôle. Si la valeur du champ version est 1, il s'agit d'un paquet PPPoE du plan de contrôle. Si sa valeur est 2, il s'agit d'un paquet 5WE du plan de données dans le sens montant (*Upstream*). La transition par défaut pour ces états à l'exception de l'état parse-pppoe-v1-or-v2 conduit à accepter le paquet.

4.2.2 Blocs de contrôle

Les blocs de contrôle — également appelés blocs appariement-action (match-action) — se situent directement après le Parser. Ils permettent d'exprimer comment les paquets doivent être traités. C'est dans cette section qu'on définit les actions, les tables et la logique de traitement des paquets.

4.3 Implémentation du plan de contrôle en P4

Étant donné que l'objectif principal de notre travail est l'implémentation et la caractérisation de la performance du plan de données de l'AGF, le plan de contrôle est conçu pour être simple et minimal. Néanmoins, il reste crucial pour la validation et le test du comportement du plan des données, car il fournit à ce dernier les règles nécessaires au traitement des paquets en configurant les tables de correspondance "match-action".

Si P4 est un langage de programmation très approprié pour la description des pipelines de traitement des paquets, il est insuffisant pour configurer ces pipelines pendant l'exécution. Pour remédier à ceci, plusieurs protocoles réseau et interfaces basés sur le RPC peuvent être utilisés : Thrift, gRPC, Barefoot Runtime (BFRT), et P4Runtime [59]. Ce dernier a été introduit par le Consortium P4 et vise à devenir la norme pour la configuration des commutateurs P4. Ces interfaces permettent à un client du plan de contrôle de lire et d'écrire les entrées des tables du plan de données. La structure de ces tables (nom de la table, clés de correspondance, nom et paramètres des actions...) se trouve dans un fichier "p4info" qui est créé lors de la compilation du programme P4. En utilisant ce fichier, le plan de contrôle envoie des messages de requête d'écriture pour ajouter des entrées aux tables.

Dans notre travail, nous utilisons des APIs basées sur Python pour l'implémentation du plan de contrôle. Dans le cas du BMv2, nous configurons les tables via l'API Thrift (car au moment du lancement projet les outils de développement et tests P4 ne prenaient pas en charge entièrement le P4runtime). Dans le cas du Tofino ASIC, nous utilisons l'API Barefoot Runtime. Vu sa complexité, une implémentation du plan du contrôle de l'AGF (AGF-CP) [8] complète permettant une intégration avec les différents composants du plan de contrôle du réseau 5G n'entre pas dans le cadre de notre projet. La configuration des tables dépend du scénario de test et consiste en règles de correspondance et d'action (match-action) résumées dans le Tableau 4.1.

TABLEAU 4.1 Les règles du plan de contrôle pour configurer le plan de données de l'AGF, les clés de correspondance utilisées et les paramètres utilisés par les actions correspondantes dans les méta-données d'un paquet.

Règle	Clé de correspondance	Paramètres
Règle de filtration des paquets	IP dest addr, UDP dest addr	-
Règle de mise en tunnel 5WE et de routage	TEID, UPF IP	session_id, session
Règle de Classification du trafic descendant	QFI	c_pcp
Règle de balisage VLAN	pdu_session_id	port, vlan_id
Règle de maintien de contexte par session	s-tag, c-tag, 5WE Session ID	session_id
Règle de mise en tunnel GTPU	pdu_session_id	port, upf_ip, agf_ip
Règle de Classification du trafic montant	pdu_session_id, QFI	-
Règle de génération de TEID	pdu_session_id, 5we_session_id	id

CHAPITRE 5 EVALUATION DES PERFORMANCES : RÉSULTATS ET ANALYSE

5.1 Introduction

La mise en œuvre d’une fonction AGF performante est une tâche assez exigeante. La difficulté réside principalement dans la gestion d’un nombre important de sessions d’abonnés simultanées, ainsi que dans les contraintes relatives le débit et la latence. Dans ce qui suit, nous examinerons ces critères de performance du prototype AGF proposé basé sur le commutateur ASIC Intel Tofino (Edgecore Wedge 100BF-32X) à travers différents scénarios de test. Nous commençons par la description du banc d’essai. Puis nous introduisons les critères d’évaluation de performance. Et enfin nous présentons nos résultats expérimentaux.

5.2 Banc d’essai

Les bancs d’essai proposés pour évaluer la performance sont ceux présentés dans les Figures 5.1 et 5.2. Nous avons utilisé le commutateur Intel Tofino Edgecore Wedge 100BF-32X comme dispositif sous test (DUT) qui implemente la fonction AGF. Il a 32 ports QSFP28, chacun capable de supporter un débit maximal de 100 Gbit/s. Le débit totale ainsi atteint ainsi 3,2 Tbit/s.

En premier lieu nous avons utilisé un seul serveur pour générer et recevoir le trafic comme présenté dans la Figure 5.1, puis nous passons à deux serveurs afin d’augmenter le nombre de sessions générées comme présenté dans la Figure 5.2. Les serveurs sont connectés au commutateur via des câbles de connexion QSFP28 vers SFP+ 4x25Gbps.

Les scénarios expérimentaux que nous avons conçus, explorent différentes échelles de génération de flux et leur impact sur les performances du commutateur.

Nous commençons par le scénario le plus simple, dans lequel un seul flux est généré. Puis, nous augmentons le nombre des flux générés pour atteindre un total de 1 000. Enfin, nous poussons les limites du générateur de trafic jusqu’à sa capacité maximale, pour atteindre 20 000 flux. Les 3 expériences conçues sont résumées dans le Tableau 5.1.

Par la suite, nous tentons de doubler le nombre de flux maximal en introduisant un serveur supplémentaire. Deux serveurs sont utilisés simultanément, chacun générant 20 000 flux ce qui nous a permis d’atteindre un nombre de 40 000 flux.

Il est à noter que le contenu des paquets (*payload*) est exactement le même pour toutes les

expériences.

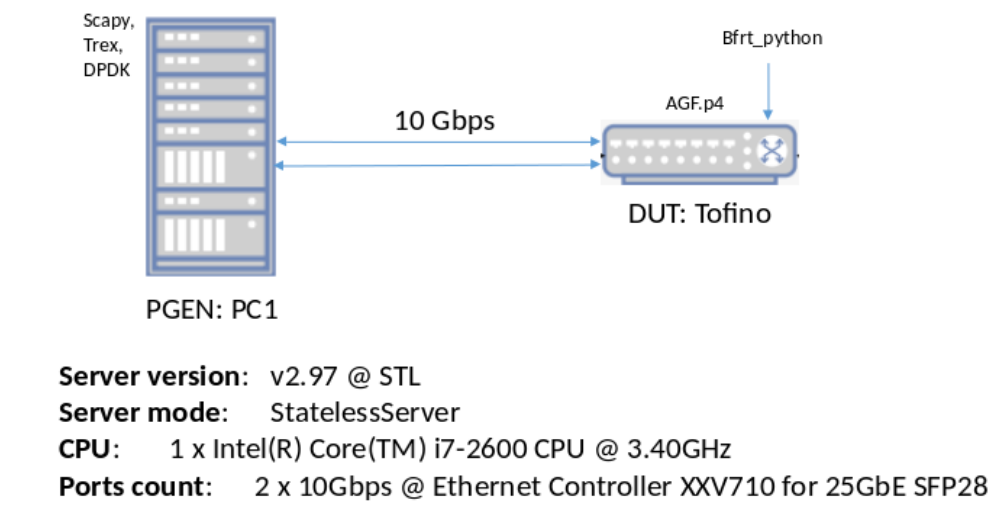


FIGURE 5.1 Banc d’essai utilisé pour l’évaluation avec un seul serveur

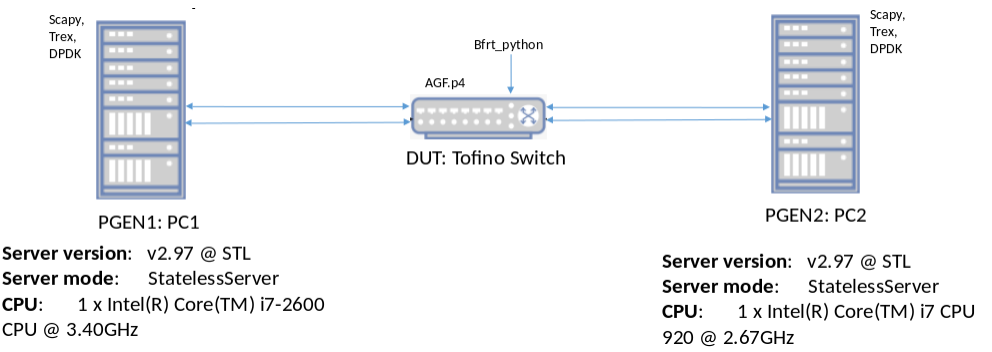


FIGURE 5.2 Banc d’essai utilisé pour l’évaluation avec deux serveur

TABLEAU 5.1 Sommaire des expériences

ID de l’expérience	Nombre de flux
F1	1
F1k	1000
Fmax	20000

5.3 Résultats expérimentaux

5.3.1 Débit

Le tableau 5.2 résume les différents débits générés par le générateur de trafic pour chaque expérience. Au cours de la première expérience avec un seul flux de paquets généré, on a pu atteindre environ 10 Gb/s. Pour la deuxième expérience, le débit a diminué à 8.62 Gb/s. Finalement, une chute de débit a été observé lors de la troisième expérience à 4.65 Gb/s.

La réduction du débit généré est due la diminution de performance du générateur de trafic lorsque nous augmentons le nombre de flux [51].

On souligne aussi que le débit total en transmission est différent de celui en réception. Cela est dû à la différence de taille des paquets 5WE et GTP-U.

TABLEAU 5.2 Le débit mesuré (en Gb/s) en augmentant le nombre de flux pour un seul générateur de trafic

ID de l'expérience	Débit Tx (Gb/s)	Débit Rx (Gb/s)
F1	9.94	6.97
F1K	8.62	5.98
Fmax	4.65	3.27

Le tableau 5.3 présente les résultats de la quatrième expérience qui combine deux serveurs pour atteindre un nombre maximal de flux de 40 000. Le débit total généré a atteint 8.4 Gb/s.

TABLEAU 5.3 Le débit mesuré (en Gb/s) en utilisant deux générateur de trafic avec le maximum de flux possible

ID de PGEN	Débit Tx (Gb/s)	Débit Rx (Gb/s)
1	4.79	3.33
2	3.60	2.5
Total	8.39	5.83

5.3.2 Taux de perte de paquets

Les résultats montrent que le taux de perte de paquets est nul. Aucune perte de paquets n'a eu lieu lors des 4 expériences : le nombre total des paquets entrants est le même que les paquets sortants ($o_pkts = i_pkts$).

Le taux de perte nulle des paquets démontre que le débit utilisé peut être supporté par dispositif sous test (Tofino). Ce qui n'est pas surprenant puisque chaque port du commutateur Tofino peut supporter jusqu'à 100 Gb/s alors que nos expériences sont effectuées avec des débits ne dépassant pas 10 Gb/s.

5.3.3 Latence

Nous utilisons TRex pour la collecte de statistiques relatives à la latence. Durant toutes les expériences, la latence obtenue est toujours nulle et aucune influence du nombre de flux n'a pas pu être observée. Ce comportement n'est pas surprenant vu que T-rex, en tant que générateur de trafic logiciel, n'est pas capable de mesurer la latence avec une précision de l'ordre de la nanoseconde. Comme nous ne disposons pas des outils permettant la mesure de latence avec une telle précision, nous nous contentons des informations relatives à la latence fournies par Intel.

5.3.4 Utilisation des ressources matérielles

Tofino utilise principalement deux types de mémoire pour effectuer la mise en correspondance des tables, à savoir la mémoire statique à accès aléatoire (*Static Random-Access Memory SRAM*) et la mémoire ternaire adressable par le contenu (*Ternary Content-Addressable Memory TCAM*). Ces mémoires sont généralement responsables de stocker les entrées des tables de routage, d'acheminement ou des sessions. Alors que la mémoire TCAM, dont la capacité est plus limitée, est utilisée pour la correspondance de plage, la mémoire SRAM, quant à elle, est utilisée pour la correspondance exacte. Les deux mémoires SRAM et TCAM sont des ressources critiques et leur nature limitée impose une limite au nombre d'entrées qui peuvent être installées simultanément.

Dans le contexte de l'AGF, une session correspond à un flux spécifique de paquets associés à un tuple composé des deux champs : l'adresse IPv4 source et le TEID. La table qui utilise ces deux champs comme clés de correspondance est "t_TEID". Le nombre maximal de règles qui peuvent être installées dans cette table détermine le nombre maximal de sessions que le commutateur Tofino peut prendre en charge.

Le langage P4 permet de spécifier la taille maximale d'une table comme paramètre lors de sa déclaration. Pour évaluer le nombre de sessions maximale que le commutateur Tofino peut stocker, nous avons compilé les tables de correspondance avec différentes tailles. Le nombre d'entrées le plus élevé que nous avons obtenu pour la table "t_TEID" sans avoir d'erreurs de compilation est 286500.

Le Tableau 5.4 présente le taux d'utilisation de la mémoire SRAM et TCAM pour 286500 abonnés, en utilisant uniquement le type de correspondance exacte. L'outil utilisé pour obtenir ces statistiques sur l'utilisation des ressources est P4 Insight (P4I) [25], qui fournit un rapport complet sur la consommation de mémoire et de ressources du commutateur Tofino. Nous constatons que les 12 étages de pipeline sont utilisés (chacun disposant d'une quantité constante de mémoire et de blocs logiques). Nous constatons aussi que la mémoire SRAM a presque atteint sa capacité maximale, alors que la mémoire TCAM n'a pas été utilisée. En effet, notre programme P4 initial utilise exclusivement la correspondance exacte pour toutes les tables. Il en résulte que le compilateur contourne simplement la TCAM et utilise seulement la SRAM, ce qui conduit à un taux d'utilisation de la mémoire TCAM de 0%.

TABLEAU 5.4 Statistiques d'utilisation des ressources mémoire en pourcentage du commutateur programmable en P4 Intel Tofino, implémentant les fonctions AGF pour 286 500 abonnés

Étage de pipeline \ Ressource mémoire	SRAM	TCAM
0	100%	0%
1	100%	0%
2	100%	0%
3	100%	0%
4	96.3%	0%
5	100%	0%
6	100%	0%
7	100%	0%
8	96.3%	0%
9	100%	0%
10	100%	0%
11	71.3%	0%
Moyenne	97.0%	0%

Compte tenu des ressources TCAM disponibles mais non utilisées, nous avons adapté notre programme P4 pour optimiser d'utilisation de mémoire et ainsi augmenter nombre de sessions des abonnés. En effet, selon le guide d'utilisateur "P4 Compiler (bf-p4c) User Guide", nous pouvons forcer le compilateur à utiliser des ressources TCAM pour une table spécifique. Nous avons fait le choix d'utiliser les ressources TCAM pour les trois tables : `t_qfi`, `t_pppoe_cp` et `t_pdu_upf_routing`. Les deux tables `t_pppoe_cp` et `t_qfi` qui ont une petite taille (res-

pectivement 4096 et 64) ne posent pas de problème : TCAM peut être utilisée sans réduire la taille de la table. La table `t_pdu_upf_routing` qui a une taille plus grande (65536) atteint une limite de 43000 entrées lorsque la TCAM est utilisée. En forçant le compilateur à utiliser la TCAM au lieu de la SRAM pour ces tables et en réduisant la taille des tables de 65 536 à 43 000, plus d'espace mémoire a été libéré pour la table `t_TEID` dont la taille est passée de 285 500 à 400 300. L'utilisation des ressources mémoires dans ce cas est présentée dans le tableau 5.5. On constate que l'utilisation de mémoire TCAM est augmentée à 32.3%.

TABLEAU 5.5 Statistiques d'utilisation des ressources mémoire en pourcentage du commutateur programmable en P4 Intel Tofino, implémentant les fonctions AGF pour 400 300 abonnés

Étage de pipeline \ Ressource mémoire	SRAM	TCAM
0	100%	0%
1	100%	0%
2	100%	0%
3	100%	100%
4	96.3%	100%
5	100%	100%
6	100%	83.3%
7	100%	0%
8	100%	0%
9	100%	4.2%
10	100%	0%
11	100%	0%
Moyenne	99.7%	32.3%

Bien que nous nous soyons principalement concentrés sur l'optimisation de l'utilisation de la mémoire, nous analysons aussi la consommation d'autres types ressources matérielles du commutateur Tofino présentées ci-dessous :

- Les crossbars d'entrée (*Input Crossbars*) : multiplexeurs utilisés pour choisir les clés d'entrée pour la recherche de correspondance. On distingue deux blocs séparés : l'un est utilisé pour la correspondance exacte (*Exact Match Input Crossbar*) et l'autre pour la correspondance de plage (*Ternary Match Input Crossbar*). Plus les clés de correspondance des tables sont larges et leur nombre est élevé, plus les ressources Crossbars sont utilisées.

- Les bits de hachage (*Hash Bits*) : bits utilisés pour effectuer des recherches dans les tables de hachage à correspondance exacte.
- Les portes (*Gateways*) : unités spécialisées pour effectuer le branchement conditionnel.
- Les octets du bus de données d'action (*Action Data Bus Bytes*) et les instructions VLIW (*Very Long Instruction Word*) : les instructions VLIW sont utilisées pour encoder les actions des tables que doivent être exécutées. Ces actions utilisent le bus de données d'action.
- Les *Stashes* : petites mémoires utilisées pour enregistrer temporairement les entrées qui ne peuvent pas être placées dans les tables de correspondance exacte hachée en raison de multiples collisions de hachage.

Les Tableaux 5.6 et 5.7 fournissent une comparaison d'utilisation de ces ressources du prototype AGF basé sur le commutateur Tofino pour 286 500 et 400 300 abonnés. Nous constatons qu'aucune de ces ressources ne représentent un facteur limitant : leurs taux d'utilisation ne dépassent pas 20% dans les deux cas. Nous constatons aussi une augmentation du taux d'utilisation dans le cas du 400 300 sessions par rapport au 286 500 sessions.

TABLEAU 5.6 Statistiques d'utilisation des différentes ressources en pourcentage du commutateur programmable en P4 Intel Tofino, implémentant les fonctions AGF pour 286 500 abonnés

Étage de pipeline	EMIC	TMIC	Hash Bit	Gateway	ADBB	VI	Stash
0	7.0%	0%	12.5%	12.5%	1.6%	6.3%	12.5%
1	10.9%	0%	24.0%	6.3%	1.6%	9.4%	18.8%
2	9.4%	0%	16.8%	0%	0%	3.1%	12.5%
3	6.3%	0%	20.4%	0%	17.2%	9.4%	18.8%
4	1.6%	0%	10.3%	0%	17.2%	6.3%	6.3%
5	6.3%	0%	11.5%	0%	4.7%	6.3%	12.5%
6	6.3%	0%	11.5%	0%	4.7%	6.3%	12.5%
7	6.3%	0%	11.5%	0%	4.7%	6.3%	12.5%
8	10.9%	0%	25.5%	6.3%	7.8%	18.8%	18.8%
9	9.4%	0%	30.8%	6.3%	12.5%	12.5%	18.8%
10	11.7%	0%	24.5%	6.3%	1.6%	9.4%	18.8%
11	4.7%	0%	12.5%	0%	1.6%	6.3%	6.3%
Moyenne	7.6%	0%	17.7%	3.1%	6.3%	8.3%	14.1%

* EMIC : Exact Match Input Crossbar

* TMIC : Ternary Match Input Crossbar

* ADBB : Action Data Bus Bytes

* VI : VLIW Instruction

TABLEAU 5.7 Statistiques d'utilisation des différentes ressources en pourcentage du commutateur programmable en P4 Intel Tofino, implémentant les fonctions AGF pour 400300 abonnés

Étage de pipeline	EMIC	TMIC	Hash Bit	Gateway	ADBB	VI	Stash
0	7.0%	0%	12.5%	12.5%	1.6%	6.3%	12.5%
1	10.9%	0%	24.0%	6.3%	1.6%	9.4%	18.8%
2	9.4%	0%	16.8%	0%	0%	3.1%	12.5%
3	10.9%	3%	22.4%	0%	18.8%	12.5%	18.8%
4	6.3%	3%	11.5%	0%	21.9%	9.4%	6.3%
5	6.3%	3%	11.5%	0%	21.9%	9.4%	12.5%
6	7.0%	6.1%	12.5%	0%	21.9%	12.5%	12.5%
7	6.3%	0%	11.5%	0%	4.7%	6.3%	12.5%
8	6.3%	0%	11.5%	6.3%	4.7%	6.3%	18.8%
9	11.7%	1.5%	33.7%	6.3%	12.5%	18.8%	18.8%
10	8.6%	0%	33.7%	6.3%	7.8%	12.5%	18.8%
11	9.4%	0%	16.8%	0%	1.6%	6.3%	6.3%
Moyenne	8.3%	1.4%	18.2%	3.1%	9.9%	9.4%	14.1%

* EMIC : Exact Match Input Crossbar

* TMICTernary Match Input Crossbar

* Action Data Bus Bytes

* VI : VLIW Instruction

5.4 Conclusions et recommandations

Les résultats de l'évaluation expérimentale démontrent la faisabilité et l'efficacité de l'implémentation du plan de données de la fonction AGF sur un commutateur ASIC programmable en P4. Pour un débit maximal d'environ 10 Gb/s et un nombre de flux maximal de 40 000 (limités uniquement par le générateur de trafic plutôt que par le commutateur sous test) aucune perte de paquet n'a été observée et la latence reste très faible (l'ordre de nanoseconde) à mesurer par les outils de mesure à notre disposition. Cependant on constate que les ressources mémoires d'un tel commutateur pour stocker les règles de correspondance, représentent un facteur limitant (400 300 flux peuvent être pris en charge après optimisations).

Ainsi les commutateurs ASIC Tofino sont particulièrement bien adaptés pour traiter des flux de trafic à haut débit lorsque le nombre de règles est relativement faible. Nous les recommandons pour implanter le plan de données de base performant de l'AGF. En outre, ils peuvent être utilisés pour traiter sélectivement certains utilisateurs ou flux de données, en leur donnant la priorité en fonction du volume de leurs trafics de données.

CHAPITRE 6 CONCLUSION

6.1 Synthèse des travaux

Nous avons démontré la faisabilité de l'AGF en analysant ses spécifications techniques et en facilitant son implémentation en P4. L'étude approfondie des spécifications permet de comprendre les exigences et les fonctionnalités de l'AGF, ouvrant ainsi la voie à sa mise en œuvre pratique.

Notre travail a également permis d'évaluer les performances et les contraintes du prototype AGF sur le commutateur Tofino programmable en P4. Cette évaluation permet de mesurer les capacités du commutateur Tofino à supporter le plan de données de l'AGF et à gérer un grand nombre de sessions simultanées. Ces résultats sont essentiels pour évaluer l'efficacité et l'adéquation de l'AGF dans un contexte réel de réseau 5G.

Enfin, ce travail de recherche a permis de dégager des recommandations claires pour un déploiement industriel potentiel de la solution AGF. Ces recommandations tiennent compte des performances du prototype, des limitations identifiées et des besoins spécifiques de l'industrie. Elles permettent d'orienter les futurs déploiements de l'AGF dans des environnements opérationnels, en tenant compte des meilleures pratiques et des stratégies pour optimiser son utilisation dans des réseaux 5G en plein essor.

6.2 Limitations des travaux

Le projet s'est appuyé sur des outils de test et de génération de trafic pour évaluer les performances du commutateur Tofino. Bien que le commutateur lui-même soit capable de gérer plusieurs téraoctets de données par seconde, les serveurs disponibles au laboratoire n'étaient pas en mesure de générer un trafic aussi volumineux. Par conséquent, certaines mesures telles que la latence n'ont pas pu être expérimentalement mesurées faute d'instruments de haute précision.

La capacité mémoire du commutateur Tofino s'est révélée être un facteur limitant pour gérer des millions de sessions. Bien qu'il soit idéal pour implémenter le plan de données de la fonction 5G AGF grâce à sa programmabilité via le langage P4 et ses excellentes performances en termes de débit et de latence, il ne dispose pas de la mémoire nécessaire pour traiter un grand nombre de sessions simultanées.

En ce qui concerne la qualité de service (QoS), le prototype a pu classer les paquets selon

les classes de trafic définies par la norme 5G. Cependant, la fonctionnalité QoS est limitée en raison du nombre restreint de files d'attente disponibles du gestionnaire de trafic du commutateur Tofino, ce qui ne permet qu'une gestion de base de la qualité de service.

Enfin, le plan de contrôle de l'AGF est resté simple et minimal dans le cadre de ce projet pour permettre des tests dans un environnement isolé avec un banc d'essai composé du prototype P4-AGF et des générateurs de trafic. Cependant, pour des tests plus complets et intégrés dans un système 5G pleinement opérationnel, il sera nécessaire de réaliser un plan de contrôle AGF plus complet qui permettra une interaction avec le réseau cœur 5G. Ceci offre des résultats plus riches en reproduisant la complexité de l'écosystème 5G dans un environnement de test plus réaliste.

6.3 Améliorations futures

Pour améliorer les limitations liées à la mémoire et à la qualité de service de l'AGF, plusieurs directions peuvent être explorées. Une première approche consiste à adopter une plateforme hybride matérielle et logicielle, en combinant les plans de données logicielles et matérielles. Cette approche offre la possibilité d'augmenter considérablement le nombre de sessions d'abonnés traitées en parallèle par l'AGF. En exploitant la capacité de mémoire pratiquement illimitée des commutateurs logiciels et la haute performance des commutateurs matériels, il est possible de gérer efficacement les appareils envoyant des données à faibles débits avec les commutateurs logiciels, tandis que les connexions nécessitant des débits plus élevés peuvent être traitées par les commutateurs matériels.

Une seconde piste d'amélioration repose sur l'exploration de techniques d'optimisation du plan de données de l'AGF en exploitant pleinement les capacités de programmabilité du commutateur Tofino. Une analyse fine des ressources matérielles du Tofino, telles que la mémoire, les files d'attente et les blocs d'association/action, pourrait permettre de mieux équilibrer les tâches entre le matériel et le logiciel, optimisant ainsi l'utilisation des ressources disponibles. De plus, des méthodes de compression de données pourraient être étudiées pour réduire la charge mémoire de l'AGF, permettant ainsi de gérer davantage de sessions simultanément.

Une autre voie d'amélioration envisageable est l'utilisation d'une plateforme hétérogène basée sur un commutateur P4 et des FPGAs. Le prototype actuel basé sur le commutateur Tofino permet de classer les paquets grâce au langage P4, mais il présente des limitations en termes de gestion de la qualité de service. Le trafic manager du Tofino ne dispose que d'un nombre limité de files d'attente, alors qu'un plus grand nombre est nécessaire pour mettre en œuvre des mécanismes de qualité de service hiérarchique (*Hierarchical Quality of Service* :

HQoS). En intégrant un FPGA supplémentaire dans l'AGF, il devient possible d'exprimer la classification des flux en fonction de leurs classes QoS via P4, tandis que les fonctions d'ordonnancement et de gestion des files d'attente seront assurées par le FPGA.

Enfin, pour améliorer les performances globales de l'AGF, il serait intéressant de s'intéresser à la conception de mécanismes de gestion du trafic plus sophistiqués et adaptatifs, permettant de faire face à des situations de congestion et de gérer efficacement les différentes classes de trafic en fonction des besoins spécifiques de chaque utilisateur. Ces mécanismes pourraient être développés et testés au sein d'un environnement de test complet, en interagissant avec un réseau 5G pleinement opérationnel, afin de valider leur efficacité et leur adaptabilité à des scénarios réels.

RÉFÉRENCES

- [1] K. Hafner et M. Lyon, *Where Wizards Stay up Late : The Origins of the Internet*. USA : Simon & Schuster, Inc., 1996.
- [2] L. Dryburgh, *Cellular Dreams : How the Cellular Network Created the Future*. Routledge, 2012.
- [3] Y. Ji, J. Zhang, Y. Xiao et Z. Liu, “5G flexible optical transport networks with large-capacity, low-latency and high-efficiency,” *China Communications*, vol. 16, n^o. 5, p. 19–32, may 2019. [En ligne]. Disponible : <https://doi.org/10.23919%2Fj.cc.2019.05.002>
- [4] F. Z. Yousaf, M. Bredel, S. Schaller et F. Schneider, “NFV and SDN—key technology enablers for 5g networks,” *IEEE Journal on Selected Areas in Communications*, vol. 35, n^o. 11, p. 2468–2478, nov 2017. [En ligne]. Disponible : <https://doi.org/10.1109%2Fjsac.2017.2760418>
- [5] D. Lake, N. Wang, R. Tafazolli et L. Samuel, “Softwarization of 5G networks—implications to open platforms and standardizations,” *IEEE Access*, vol. 9, p. 88 902–88 930, 2021.
- [6] S. Denazis, J. H. Salim, D. Meyer et O. Koufopavlou, “Software-defined networking (SDN) : Layers and architecture terminology,” Rapport technique, jan 2015.
- [7] R. Kundel, L. Nobach, J. Blendin, W. Maas, A. Zimmer, H.-J. Kolbe, G. Schyguda, V. Gurevich, R. Hark, B. Koldehofe et R. Steinmetz, “OpenBNG : Central office network functions on programmable data plane hardware,” *International Journal of Network Management*, vol. 31, n^o. 1, aug 2020.
- [8] “TR-456 : AGF functional requirements, Issue : 02,” BroadBand Forum, Rapport technique, 2022.
- [9] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese et D. Walker, “P4 : Programming protocol-independent packet processors,” *ACM SIGCOMM Computer Communication Review*, vol. 44, n^o. 3, p. 87–95, jul 2014.
- [10] “Open Tofino,” <https://github.com/barefootnetworks/open-tofino>, accessed : 2023-03-15.
- [11] K. Choi, M. Kim, J. Kim et B. Kim, “The limitations of programmable forwarding planes : A comparison of tofino and traditional switching architectures,” *Computer Communications*, vol. 159, p. 13–24, 2020.

- [12] D. Allan, D. E. Eastlake et D. Woolley, “5G wireless wireline convergence user plane encapsulation (5WE),” RFC 8822, 2021, available at <https://datatracker.ietf.org/doc/rfc8822/>.
- [13] J. Esch, “Prolog to : “Software-defined networking : a comprehensive survey”,” *Proceedings of the IEEE*, vol. 103, n^o. 1, p. 10–13, 2015.
- [14] M.-K. Shin, K.-H. Nam et H.-J. Kim, “Software-defined networking (SDN) : A reference architecture and open APIs,” dans *IEEE 2012 International Conference on ICT Convergence (ICTC)*, 2012, p. 360–361.
- [15] K. Bakshi, “Considerations for software defined networking (SDN) : Approaches and use cases,” dans *2013 IEEE Aerospace Conference*, mar 2013.
- [16] N. Feamster, J. Rexford et E. Zegura, “The road to SDN : an intellectual history of programmable networks,” *ACM SIGCOMM Computer Communication Review*, vol. 44, n^o. 2, p. 87–98, 2014.
- [17] K. Benzekki, A. El Fergougui et A. Elbelrhiti Elalaoui, “Software-defined networking (SDN) : a survey,” *Security and communication networks*, vol. 9, n^o. 18, p. 5803–5833, 2016.
- [18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker et J. Turner, “Openflow : Enabling innovation in campus networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, n^o. 2, p. 69–74, mar 2008. [En ligne]. Disponible : <https://doi.org/10.1145/1355734.1355746>
- [19] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper et K. Hoffmann, “Applying NFV and SDN to LTE mobile core gateways, the functions placement problem,” dans *Proceedings of the 4th workshop on All things cellular : operations, applications, & challenges*, 2014, p. 33–38.
- [20] D. Kreutz, F. M. Ramos et P. Verissimo, “Towards secure and dependable software-defined networks,” dans *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2013, p. 55–60.
- [21] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca et J. Folgueira, “Network slicing for 5G with SDN/NFV : Concepts, architectures, and challenges,” *IEEE Communications Magazine*, vol. 55, n^o. 5, p. 80–87, 2017.
- [22] N. McKeown, “Software defined networks and the maturing of the internet,” talk, 2014. [En ligne]. Disponible : <https://opennetworking.org/news-and-events/events/software-defined-networks-and-the-maturing-of-the-internet>

- [23] B. Han, V. Gopalakrishnan, L. Ji et S. Lee, “Network function virtualization : Challenges and opportunities for innovations,” *IEEE Communications Magazine*, vol. 53, n^o. 2, p. 90–97, 2015.
- [24] O. Michel, R. Bifulco, G. Retvari et S. Schmid, “The programmable data plane : Abstractions, architectures, algorithms, and applications,” *ACM Computing Surveys (CSUR)*, vol. 54, n^o. 4, p. 1–36, 2021.
- [25] F. Hauser, M. Häberle, D. Merling, S. Lindner, V. Gurevich, F. Zeiger, R. Frank et M. Menth, “A survey on data plane programming with P4 : Fundamentals, advances, and applied research,” *Journal of Network and Computer Applications*, vol. 212, p. 103561, 2023.
- [26] E. F. Kfoury, J. Crichigno et E. Bou-Harb, “An exhaustive survey on P4 programmable data plane switches : Taxonomy, applications, challenges, and future trends,” *IEEE Access*, vol. 9, p. 87 094–87 155, 2021.
- [27] J. Gomez, E. F. Kfoury, J. Crichigno et G. Srivastava, “A survey on TCP enhancements using P4-programmable devices,” *Computer Networks*, vol. 212, p. 109030, 2022. [En ligne]. Disponible : <https://www.sciencedirect.com/science/article/pii/S1389128622001852>
- [28] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown et J. Rexford, “Pisces : A programmable, protocol-independent software switch,” dans *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, p. 525–538.
- [29] “Cisco silicon one q200 and q200l processors data sheet,” <https://www.cisco.com/c/en/us/solutions/collateral/silicon-one/datasheet-c78-744312.html>, accessed : 2023-07-25.
- [30] The P4.org Architecture Working Group, “P4-16 Portable Switch Architecture (PSA),” Technical Report, November 22 2018. [En ligne]. Disponible : <https://opennetworking.org/wp-content/uploads/2020/10/PSA-v1.1.0.pdf>
- [31] “Vitis networking p4,” <https://www.xilinx.com/products/intellectual-property/ef-di-vitisnetp4.html>, consulté le 24 juillet 2023.
- [32] “Behavioural-model-version2 (BMv2), P4 Language Consortium,” <https://github.com/p4lang/behavioral-model>, accessed : 2023-03-15.
- [33] “p4c-behavioral, P4 Language Consortium,” https://github.com/p4lang/p4c-behavioral/tree/master/p4c_bm, accessed : 2023-03-15.
- [34] “Performance of BMv2, P4 Language Consortium,” <https://github.com/p4lang/behavioral-model/blob/main/docs/performance.md>, accessed : 2023-03-15.
- [35] “TS 23.501 : system architecture for the 5g system (5gs),” 3rd Generation Partnership Project (3GPP), Rapport technique.

- [36] T. Osiński, M. Kossakowski, M. Pawlik, J. Palimaka, M. Sala et H. Tarasiuk, “Unleashing the performance of virtual BNG by offloading data plane to a programmable ASIC,” dans *Proceedings of the 3rd P4 Workshop in Europe*. ACM, dec 2020.
- [37] R. Kundel, L. Nobach, J. Blendin, H.-J. Kolbe, G. Schyguda, V. Gurevich, B. Koldehofe et R. Steinmetz, “P4-BNG : Central office network functions on programmable packet pipelines,” dans *2019 15th International Conference on Network and Service Management (CNSM)*. IEEE, oct 2019.
- [38] R. Ricart-Sanchez, P. Malagon, A. Matencio-Escolar, J. M. Alcaraz Calero et Q. Wang, “Toward hardware-accelerated QoS-aware 5G network slicing based on data plane programmability,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, n^o. 1, p. e3726, 2020, e3726 ett.3726.
- [39] R. MacDavid, C. Cascone, P. Lin, B. Padmanabhan, A. Thakur, L. Peterson, J. Rexford et O. Sunay, “A P4-based 5G user plane function,” dans *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*. ACM, oct 2021.
- [40] S. K. Singh, C. E. Rothenberg, J. Langlet, A. Kassler, P. Voros, S. Laki et G. Pongracz, “Hybrid P4 programmable pipelines for 5G gNodeB and user plane functions,” *IEEE Transactions on Mobile Computing*, p. 1–18, 2022.
- [41] R. Kundel, T. Meuser, T. Koppe, R. Hark et R. Steinmetz, “User plane hardware acceleration in access networks : Experiences in offloading network functions in real 5G deployments,” dans *Proceedings of the Annual Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences, 2022.
- [42] H. Harkous, M. He, M. Jarschel, R. Pries, E. Mansour et W. Kellerer, “Performance study of P4 programmable devices : Flow scalability and rule update responsiveness,” dans *IEEE IFIP Networking Conference*, June 2021.
- [43] A. Al Sadi, D. Berardi, F. Callegati, A. Melis et M. Prandini, “P4DM : Measure the link delay with P4,” *Sensors*, vol. 22, n^o. 12, 2022.
- [44] “TR-470 : 5G wireless wireline convergence architecture,” BroadBand Forum, Rapport technique, 2022.
- [45] BBF, “TR-101 : Adsl2/adsl2plus functionality test plan,” Broadband Forum, Rapport technique, 2011. [En ligne]. Disponible : <https://www.broadband-forum.org/pdfs/tr-101-2-0-0.pdf>
- [46] —, “TR-105 : Migration to ethernet-based broadband aggregation,” Broadband Forum, Rapport technique, 2011. [En ligne]. Disponible : <https://www.broadband-forum.org/pdfs/tr-105-2-0-0.pdf>

- [47] —, “TR-145 : Multi-service broadband network functional modules and architecture,” Broadband Forum, Rapport technique, 2012. [En ligne]. Disponible : <https://www.broadband-forum.org/pdfs/tr-145-1-0-0.pdf>
- [48] S. Rohith Raj, R. Rohith, M. Minal et G. Shobha, “Scapy — a powerful interactive packet manipulation program,” dans *International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, 2018, p. 1–5.
- [49] T. T. Group, *tcpdump - dump traffic on a network*, The Tcpdump Group, 1987. [En ligne]. Disponible : www.tcpdump.org/
- [50] P. Emmerich, S. Gallenmüller, G. Antichi, A. W. Moore et G. Carle, “Mind the gap - a comparison of software packet generators,” dans *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2017, p. 191–203.
- [51] “Cisco, TRex Documentation,” <https://trex-tgn.cisco.com/trex/doc>, accessed : 2023-07-15.
- [52] “Intel, The Pktgen Application,” <https://pktgen.github.io/Pktgen-DPDK/>, accessed : 2023-07-15.
- [53] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart et G. Carle, “MoonGen : A Scriptable High-Speed Packet Generator,” dans *Internet Measurement Conference 2015 (IMC’15)*, Tokyo, Japan, oct. 2015.
- [54] I. Corporation, “Intel ethernet network adapter xxv710 product brief,” Intel Corporation Website, 2023. [En ligne]. Disponible : www.intel.com/content/www/us/en/products/details/ethernet/700-network-adapters/xxv710-network-adapters.html
- [55] “System architecture for the 5G System (5GS),” https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf, accessed : 2023-05-22.
- [56] “User Plane Protocol and Architectural Analysis on 3GPP 5G System draft-ietf-dmm-5g-uplane-analysis-04,” <https://www.ietf.org/archive/id/draft-ietf-dmm-5g-uplane-analysis-04.txt>, accessed : 2023-05-18.
- [57] “PDU session user plane protocol ,” https://www.etsi.org/deliver/etsi_ts/138400_138499/138415/16.04.00_60/ts_138415v160400p.pdf, accessed : 2023-05-22.
- [58] “IANA Considerations for PPP over Ethernet (PPPoE) ,” <https://www.rfc-editor.org/rfc/rfc4937.html>, accessed : 2023-05-23.
- [59] “P4Runtime Specification,” <https://github.com/p4lang/p4runtime>, accessed : 2023-07-31.