UNIVERSITÉ DU QUÉBEC À MONTRÉAL

# PROVISIONNEMENT DE TRANCHES COMPATIBLES AVEC L'INFORMATIQUE EN RÉSEAU POUR LA VIDÉO 360° DANS LES RÉSEAUX DE NOUVELLE GÉNÉRATION

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
ZAKARIA AIT HMITTI

JANVIER 2023

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

*Avertissement*

La diffusion de ce mémoire se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.04-2020). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

# REMERCIEMENTS

Au terme de ce parcours, je tiens à exprimer toute ma reconnaissance à ma directrice de recherche, Pr. Halima Elbiaze. Je la remercie de m'avoir encadré, orienté, aidé et conseillé.

J'adresse mes sincères remerciements à tous les professeurs et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je remercie enfin les membres de ma petite famille qui me sont très chers pour leur soutien moral et leur confiance indéfectible dans mes choix. Leurs attentions et encouragements m'ont accompagné tout au long de ces années.

TABLE DES MATIÈRES

# LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ALU *Arithmetic Logic Unit* (Unité arithmétique et logique).

API *Application Programming Interface* (Interface de programmation d'application).

ASIC *Application-Specific Integrated Circuit* (Circuit intégré spécifique à l'application).

CPU *Central Processing Unit* (Unité centrale de traitement).

DRAM *Dynamic Random Access Memory* (Mémoire vive dynamique).

eMBB *enhanced Mobile Broadband Connectivity* (Connectivité haut débit mobile améliorée).

FPGA *Field-Programmable Gate Array* (Réseau de portes programmable sur site).

ICN *Information-Centric Network* (Réseau centré sur l'information).

INCA *In-Network Computing Architecture* (Architecture de l'inforamtique en réseau).

INT *In-band Network Telemetry* (Télémétrie réseau intrabande).

IoT *Internet of Things* (Internet des objets).

IP *Internet Protocol* (Protocole Internet).

MAC *Media Access Control* (Contrôle d'accès au support).

MANO *Management ANd Orchestration* (Gestion et orchestration).

MEC *Multi-access Edge Computing* (Informatique de périphérie multiaccès).

mMTC *massive Machine Type Communications* (Communications massives de type machine).

NAT *Network Address Translator* (Traducteur d'adresse réseau).

NFV *Network Function Virtualization* (Virtualisation des fonctions réseau).

NIC *Network Interface Card* (Carte d'interface réseau).

NPU *Network Processing Unit* (Unité de traitement réseau).

ONF *Open Networking Foundation* (Fondation de réseau ouvert).

P4 *Programming Protocol-independent Packet Processors* (Processeurs de paquets indépendants du protocole de programmation).

PDP *Programmable Data Plane* (Plans de données programmables).

PISA *Protocol Independent Switch Architecture* (Architecture de commutateur indépendante du protocole).

QoE *Quality of Experience* (Qualité de l'expérience).

QoS *Quality of Service* (Qualité de service).

RDMA *Remote Direct Memory Access* (Accès direct à la mémoire à distance).

REST *REpresentational State Transfer* (Transfert d'état représentatif).

ROHC *RObust Header Compression* (Compression d'en-tête robuste).

SDN *Software-Defined Network* (Réseau définit par logiciel).

TCP *Transmission Control Protocol* (Protocole de contrôle de transmission).

UDP *User Datagram Protocol* (Protocole de datagramme utilisateur).

LISTE DES FIGURES

# LISTE DES TABLEAUX

# RÉSUMÉ

De nombreuses applications ont récemment vu le jour grâce aux progrès technologiques dans le réseau et le calcul. Ces applications nécessitent une latence ultra faible et une bande passante élevée. En outre, les ressources de chaque service doivent être gérées, planifiées et isolées en fonction des besoins.

Aussi, la popularité des vidéos 360° a augmenté, offrant aux utilisateurs/clients une expérience de visionnage plus immersive. Cependant, la diffusion continue de vidéos 360° fait face à de nombreux défis. Ce type de services nécessite une haute résolution et un temps de réponse court afin de satisfaire l'intérêt croissant des utilisateurs et améliorer la qualité de leur expérience.

D'un côté, l'informatique en réseau (en anglais, *In-Network Computing* ou INC), également connu sous le nom de *Computing in the Network* ou COIN, permet de répartir les tâches de calcul sur le réseau au lieu de les effectuer sur des serveurs afin de garantir les performances. Ainsi, l'informatique en réseau devrait fournir une latence plus faible, un trafic réseau plus faible et un débit plus élevé. Cela peut aider à provisionner des applications telles que la diffusion continue de vidéo 360°, pour lesquelles une latence et un trafic réseau faibles sont très difficiles à atteindre.

D'un autre côté, le découpage virtuel du réseau permet la coexistence de services et d'applications aux caractéristiques très différentes sur la même infrastructure réseau. Il apporte de la flexibilité aux réseaux de nouvelle génération en permettant le déploiement d'applications avec des exigences différentes sur le même réseau.

Ce mémoire propose une architecture pour le provisionnement de tranches compatibles avec l'informatique en réseau pour la diffusion continue de vidéo 360° dans les réseaux de nouvelle génération. Il donne également un aperçu sur quelques concepts émergents, et il fournit aussi une revue de la littérature sur les notions pertinentes à notre étude et une explication détaillée de la solution.

Nous considérons un réseau défini par logiciel (*Software Defined Network*) compatible avec le langage de programmation P4 (*Programming Protocol-independent Packet Processors*) comme une infrastructure physique. Nous effectuons des simulations

approfondies et construisons un prototype de preuve de concept. La faisabilité de l'architecture est démontrée par l'évaluation de ses performances.

Nos résultats expérimentaux montrent que l'informatique en réseau réduit considérablement la latence et la charge du réseau par rapport au paradigme de l'informatique en périphérie (*Edge Computing*).

Mots clés : L'informatique en réseau, P4, Découpage virtuel du réseau, Provisionnement de tranches, Diffusion continue de vidéo 360°

# CHAPITRE I

# INTRODUCTION

## 1.1 Mise en contexte

L'énorme quantité de données générées par un nombre toujours croissant d'appareils IoT *(Internet of Things)* pourrait radicalement changer le fonctionnement d'Internet. Les ressources limitées des appareils IoT nécessitent que les tâches de calcul intensives soient déchargées vers des installations riches en ressources (par exemple, le nuage). Cependant, de nouvelles applications (telles que la réalité augmentée et l'enseignement à distance à grande échelle) imposent des contraintes strictes de qualité d'expérience, telles qu'une latence ultra-faible et une fiabilité ultra-élevée. Ainsi, le déchargement de tâches vers des ressources distantes peut présenter un risque pour la contrainte de latence.

Nous utilisons les termes en français suivants, informatique en réseau, informatique en périphérie, informatique en nuage, programmation en réseau, réseau définit par logiciel et découpage virtuel du réseau pour indiquer les termes en anglais, *In-Network Computing*, *Edge Computing*, *Cloud Computing*, *In-Network Programmability*, *Software-Defined Network* et *Network slicing*, respectivement.

L'informatique en réseau est une technologie qui permet de répartir les tâches de calcul sur le réseau au lieu de les effectuer sur les serveurs ou les appareils en dehors du réseau, afin de garantir les performances Kunze *et al.* (2021). L'informatique en réseau devrait fournir une latence plus faible, un trafic réseau plus faible et un débit plus élevé. Par conséquent, le service de diffusion continue de vidéo 360° peut bénéficier de l'informatique en réseau. Par exemple, le déplacement des algorithmes de transcodage et de prédiction des mouvements de la tête vers le réseau pourrait réduire la latence et la consommation de la bande passante. De plus, l'exécution d'algorithmes de compression avancés tels que les algorithmes prédictifs basés sur le contexte pour la compression sans perte, le préchargement et la mise en cache proactive dans le réseau peut également réduire l'utilisation de la bande passante pour la diffusion continue de vidéo 360°.

Le découpage virtuel du réseau permet la coexistence de services et d'applications aux caractéristiques très différentes sur la même infrastructure réseau. Aussi, il apporte de la flexibilité aux réseaux de nouvelle génération en permettant le déploiement d'applications avec des exigences différentes sur le même réseau.

## 1.2 Motivation

En raison de l'explosion des communications et des données générées, ainsi que de la demande toujours plus croissante de réponse à faible temps de latence, Internet est passé d'un simple moyen de transport de bits à un vaste réseau de nœuds capables d'agrégation, de calcul et de transmission de paquets. Le paradigme d'infonuagique conventionnel, dans lequel les applications utilisent des serveurs d'un centre de données distant, ne peut pas répondre aux nouvelles exigences de performances imposées

par l'Internet des objets (IoT) et d'autres technologies émergentes.

Pendant ce temps, l'informatique de périphérie multiaccès (*Multi-access Edge Computing* MEC) vise à répondre à ces nouvelles exigences de performances en distribuant les fonctionnalités des applications aux périphériques du réseau, au lieu de les exécuter uniquement dans des centres de données distants.

Contrairement à ces approches, l'informatique en réseau offre la possibilité d'exécuter des fonctions au sein des équipements réseau. Les appareils électroniques qui peuvent mettre à disposition leurs ressources peuvent aller des commutateurs ou routeurs du réseau aux serveurs infonuagiques, en passant par les stations de base cellulaires et les serveurs de périphérie.

Une tranche est un ensemble de ressources virtuelles (par exemple, des machines virtuelles) ou physiques qu'un fournisseur d'infrastructure peut attribuer à un locataire en fonction des exigences de ce dernier Zhang (2019). Le découpage en tranches de réseau permet la coexistence de différents réseaux virtuels pour divers secteurs verticaux (par exemple, e-industrie et e-santé) sur la même infrastructure physique dans les réseaux de nouvelle génération. Nous supposons que les réseaux de nouvelle génération considérés dans cet article sont les réseaux définis par logiciel (Software-Defined Network SDN) qui sont compatibles avec le langage P4.

Le SDN Kfoury *et al.* (2021) permet de séparer le plan de contrôle qui gère le trafic réseau et le plan de données qui transmet le trafic en fonction des décisions du plan de contrôle. SDN permet aux réseaux d'être programmés à l'aide d'interfaces ouvertes et peut répondre aux limites de l'infrastructure réseau actuelle. Cette séparation est bénéfique pour la flexibilité et simplifie la gestion du réseau.

Le langage P4 est utilisé pour définir le comportement des équipements réseau programmables dans le plan de données Gao et Wang (2021). La programmation avec P4 présente certaines caractéristiques principales telles que l'agilité, la conception descendante, la visibilité, une moindre complexité et des performances améliorées. Avec l'émergence de P4, la programmabilité du plan de données du réseau peut être activée pour permettre au gestionnaire de réseau de définir le comportement des commutateurs et des routeurs.

## 1.3 Problématique

En raison des récents progrès en matière de réseau et de calcul, la demande de la diffusion continue de vidéo 360° a considérablement augmenté. La diffusion continue de vidéo 360° nécessite l'utilisation d'un visiocasque de la part de l'utilisateur, lui permettant d'interagir avec la scène vidéo par les mouvements de la tête Yaqoob *et al.* (2020). Fournir des vidéos 360° aux utilisateurs finaux tout en offrant une qualité d'expérience utilisateur élevée est un véritable défi. Les mouvements de tête de l'utilisateur permettent un changement dans l'angle de vue de la scène. Toutefois ils peuvent provoquer le mal de transport et rendre l'expérience immersive très inconfortable en cas d'un temps de latence élevé. Un temps de latence extrêmement faible est l'une des exigences essentielles pour une expérience immersive.

Le temps de latence est directement influencé par le temps de réponse du visiocasque, le temps de traitement au niveau du serveur infonuagique ou le serveur de périphérie, le temps de transmission dans le réseau et d'autres facteurs. Les visiocasques modernes ont un temps de réponse très petit, indétectable par les humains Orlosky *et al.* (2017).

De plus, la diffusion continue de vidéo 360° est une application extrêmement consommatrice de bande passante et de ressources informatiques qui traite une immense quantité de données (c.-à-d., une bande passante de 25 Mb/s et un temps de latence de 40 ms). Ces exigences rendent extrêmement difficile le provisionnement de tranches incluant l'informatique en réseau pour la diffusion continue de vidéo 360° Yaqoob *et al.* (2020).

D'un autre côté, même si nous considérons l'informatique en réseau comme une solution potentielle pour répondre aux défis précédents, les périphériques réseau disposent d'une puissance de calcul et à un stockage limités. En outre, l'exécution d'application sur le réseau ne doit pas être affectée par ces contraintes. À cet égard, il est crucial de trouver le type de traitement approprié à effectuer dans le réseau. Sapio et al., Sapio *et al.* (2017) introduisent trois critères pour identifier les types appropriés tels que :

1. Une diminution significative du trafic réseau.
2. Un changement minimal requis au niveau de l'application.
3. Une qualité du calcul inchangée.

## 1.4 Méthodologie

Dans cette section, nous décrivons la méthodologie suivie lors de la réalisation de ce mémoire. Tout d'abord, nous avons commencé par une revue de la littérature autour des travaux existants dans le domaine des réseaux définis par logiciel.

À travers cette revue de la littérature, nous avons découvert plusieurs autres technologies et concepts très prometteurs tels que, l'informatique en périphérie et l'informatique en réseau. Ensuite, nous avons rassemblé plusieurs problématiques inté-

ressantes, et notre attention s'est portée sur la diffusion continue de vidéo 360°, plus particulièrement, ses exigences en ce qui concerne le temps de latence et la bande passante. À ce stade, nous nous sommes demandé comment peut-on utiliser l'informatique en réseau pour répondre aux exigences de la diffusion continue de vidéo 360°, et aussi, quelle technologie entre l'informatique en périphérie et l'informatique en réseau serait l'idéal.

Dans le but de répondre à notre question, nous avons hypothétisé que l'informatique en réseau serait l'idéal à utiliser, surtout si elle est combinée avec le découpage virtuel du réseau dans un réseau défini par logiciel.

Afin de vérifier notre hypothèse, nous avons proposé une architecture pour le provisionnement de tranches compatibles avec l'informatique en réseau pour les services de diffusion continue de vidéo 360° dans les réseaux de nouvelle génération.

Finalement, nous avons réalisé un ensemble de simulations et de tests afin de mesurer objectivement la pertinence et les performances de notre proposition.

## 1.5 Contribution

Dans ce mémoire, nous étudions le provisionnement de tranches compatibles avec l'informatique en réseau pour les services de diffusion continue de vidéo 360° en utilisant le langage de programmation P4. Dans ce cadre, nous présentons une architecture qui permet le déploiement des tranches compatibles avec l'informatique en réseau, et permettant ainsi l'exécution de fonction de traitement de vidéo 360° au sein même des équipements réseau. Dans l'ensemble, la contribution de ce mémoire consiste à :

1. Décrire la revue de la littérature des technologies permettant d'avoir l'infor-

matique en réseau, la programmabilité en réseau et le provisionnement des infrastructures. Nous discutons les travaux existants dans ces domaines et nous soulignons chaque problématique et contribution.

2. Présenter le cas d'utilisation de la diffusion continue de vidéo 360°. Ce cas d'utilisation est parmi d'autres qui présentent beaucoup de défis dans les réseaux traditionnels.

   Nous proposons une architecture pour le provisionnement et le déploiement des tranches incluant l'informatique en réseau. Aussi, nous construisons une preuve de concept dans le but de montrer la possibilité de réaliser l'architecture proposée dans le monde réel. Nous utilisons le langage de programmation émergent P4 pour programmer les équipements réseaux dans l'implémentation de la preuve de concept.

   À notre connaissance, ce travail est la toute première tentative d'exploiter l'informatique en réseau pour la fourniture de services de diffusion continue de vidéo 360°.

3. Évaluer les performances de notre solution en calculant trois métriques importantes : la latence, la gigue et la charge du réseau. Nos résultats expérimentaux montrent que l'informatique en réseau réduit significativement la latence et la charge du réseau par rapport au paradigme de l'informatique en périphérie.

Mes travaux dans le domaine de la programmation en réseau ont débouché à la publication suivante :

— S. Ouledsidi Ali, Z. Ait Hmitti, H. Elbiaze and R. Glitho (2021, May). *"On Computing in the Network : Covid-19 Coughs Detection Case Study"*. In International Symposium on Ubiquitous Networking (pp. 186-198). Springer, Cham.

Ainsi que d'autres articles qui ont été soumis pour évaluation :

— F. Aghaaliakbari, Z. Ait Hmitti, M. Rayani, M. Gherari, R. Glitho, H. Elbiaze, W. Ajib. *"An Architecture for Provisioning In-Network Computing Enabled Slices for Holographic Applications in Next Generation Networks"*. 2022 IEEE Communications Magazine.

— F. Aghaaliakbari, F. Ghasemi Javid, Z. Tasnim, Z. Ait Hmitti, M. Rayani, M. Gherari, R. Glitho, H. Elbiaze, W. Ajib. *"Demonstration of an In-Network Computing Enabled Architecture for Holographic Streaming"*. 2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN).

— Z. Ait Hmitti, H. Ben Ammar, E. Gelal Soyak, Y. Kardjadja, S. Malektaji, S. Ouledsidi Ali, M. Rayani, M. Saqib, S. Taghizadeh, W. Ajib, H. Elbiaze, O. Ercetin, Y. Ghamri Doudane, R. Glitho. *"Towards Smart Collaborative cOmputing, caching and netwoRking paradIgm for Next Generation communication infrastructures"*. 2022 International Conference on Computer Communications and Networks (ICCCN).

## 1.6 Organisation du mémoire

L'organisation du reste de ce mémoire est la suivante :

Nous commençons par le chapitre 2 où nous donnons un aperçu sur des notions relatives au provisionnement de tranches compatibles avec l'informatique en réseau pour la diffusion continue de vidéo 360°. Ces concepts sont : le réseau défini par logiciel, l'informatique en réseau, le langage de programmation P4 et le découpage virtuel du réseau.

Dans le chapitre 3, nous présentons différents travaux existants dans la littérature et pertinents à notre étude. Plus précisément, nous discutons des articles qui ont proposé des solutions basées sur SDN et NFV (*Network Function Virtualization* ), des travaux qui traitent la programmabilité en réseau et des articles sur le provisionnement des infrastructures qui se concentrent sur l'informatique en réseau et d'autres dont l'environnement est l'informatique en périphérie.

Le chapitre 4 présente notre travail sur le transcodage de la diffusion continue de vidéo 360° en utilisant le langage de programmation P4 au sein des équipements réseau. Dans ce chapitre, nous décrivons le cas d'utilisation de la diffusion continue de vidéo 360°. Puis, nous présentons l'architecture proposée en décrivant ses modules et en détaillant le service d'intégration de tranches et ses interactions avec les autres composants de l'architecture. Nous introduisons également une preuve de concept de notre architecture proposée.

Le chapitre 5 est consacré à l'évaluation de performance de la solution proposée. Nous commençons, dans ce chapitre, par décrire l'environnement et les scénarios de simulation. Ensuite, nous présentons et discutons les résultats.

Dernièrement, le chapitre 6 présente une conclusion générale du mémoire et aborde les perspectives de recherche.

# CHAPITRE II

# GÉNÉRALITÉS

## 2.1 Introduction

Dans ce chapitre, nous présentons les concepts généraux en relation avec le sujet. La deuxième section décrit le réseau défini par logiciel. Dans cette section, nous commençons par une définition, ensuite, nous expliquons l'architecture du réseau défini par logiciel, les avantages et les défis de cette technologie. La section suivante définit l'informatique en réseau, détaille son historique, la compare avec le calcul dans les réseaux traditionnels, puis détaille ses avantages et ses défis. La quatrième section présente le langage de programmation P4. Dans cette section, nous commençons par la définition, ensuite, nous définissons les éléments du langage P4. Dans la même section, nous expliquons comment programmer une cible avec P4, puis nous décrivons les cas d'utilisation de P4. La dernière section présente le découpage virtuel du réseau. Dans cette même section, nous définissons le découpage virtuel du réseau, nous décrivons ses principes et nous discutons des technologies permettant sa réalisation.

## 2.2 Réseau défini par logiciel

### 2.2.1 Définition

L'organisation *Open Networking Foundation* (ONF) est un consortium à but non lucratif dirigé par des opérateurs, qui encourage et démocratise l'innovation dans les réseaux programmables définis par logiciel ONF (2022a). ONF définit SDN comme une architecture réseau où il y a une séparation des plans de contrôle et de données, une centralisation logique de l'intelligence et de l'état du réseau et une abstraction de l'infrastructure physique du réseau à travers des applications.

L'architecture de réseau traditionnelle est repensée par les fournisseurs de réseau et les utilisateurs pour un certain nombre de raisons Stallings (2015) :

— **La demande augmente :** les réseaux d'entreprise et Internet sont saturés à cause d'un certain nombre de tendances (par exemple, l'informatique en nuage, les données massives, le trafic mobile et l'Internet des objets).

— **L'offre augmente :** à mesure que le trafic du réseau augmente, la capacité des technologies de réseau à le gérer augmente également. Aussi, parallèlement à l'augmentation de la capacité de transmission du réseau, les périphériques réseau ont également amélioré leurs performances. En effet, ces périphériques disposent de mémoires plus grandes et plus rapides, permettant des tampons plus grands et un accès plus rapide aux tampons.

— **Les modèles de trafic sont plus complexes :** les réseaux d'entreprise sont de moins en moins adaptés à l'évolution des modèles de trafic à mesure qu'ils deviennent plus complexes.

— **Les architectures de réseau traditionnelles sont inadéquates :** bien que

les techniques de transmission et les dispositifs de réseau aient été améliorés en matière de performances, les architectures de réseau traditionnelles ne peuvent pas suivre le volume complexe, variable et élevé des charges imposées. La variété des applications augmente également les exigences de qualité de service (Quality of Service QoS) et de qualité d'expérience (Quality of Experience QoE) imposées au réseau, ce qui nécessite une gestion du trafic plus sophistiquée et plus agile.

Une architecture SDN utilise des interfaces de programmation d'application (*Application Programming Interfaces* APIs) standardisées pour permettre aux programmeurs réseau de définir et de reconfigurer rapidement la manière dont les données ou les ressources sont gérées dans un réseau Kirkpatrick (2013). Grâce aux APIs, les applications réseau (telles que les systèmes de messagerie, les services de l'informatique en nuage ou les applications de téléphonie) peuvent reconfigurer le réseau et ses composants (tels que les commutateurs, les bâtis de serveurs, les machines virtuelles et autres périphériques finaux) ou récupérer des données spécifiques, selon leurs besoins.

### 2.2.2 Architecture SDN

Les besoins des utilisateurs du réseau ne cessent d'augmenter ce qui pousse les architectures des réseaux traditionnelles à atteindre leurs limites. Les réseaux définis par logiciel Stallings (2015) représentent une approche développée pour s'adapter aux architectures des réseaux modernes en fournissant une architecture en trois plans (c.-à-d. plan de données, plan de contrôle et plan d'applications) comme décrits dans la figure 2.1. Le plan de données est constitué de périphériques réseau (par exemple, des commutateurs physiques) responsables principalement du transfert de trafic. Le

plan de contrôle s'occupe de la gestion des ressources du plan de données, il est constitué de plusieurs contrôleurs SDN. Chaque contrôleur est responsable de gérer un sous-ensemble de périphériques réseau et peut également communiquer avec les autres contrôleurs. Le plan de contrôle communique avec le plan de données à l'aide d'une API *Southbound* (par exemple, *OpenFlow*). Aussi, le plan de contrôle communique avec le plan d'applications grâce à une API *Northbound* (par exemple, REST *(REpresentational State Transfer)* API). Le plan d'applications fait une sorte d'abstraction des deux autres plans et permet de déployer des applications réseau ou autres.

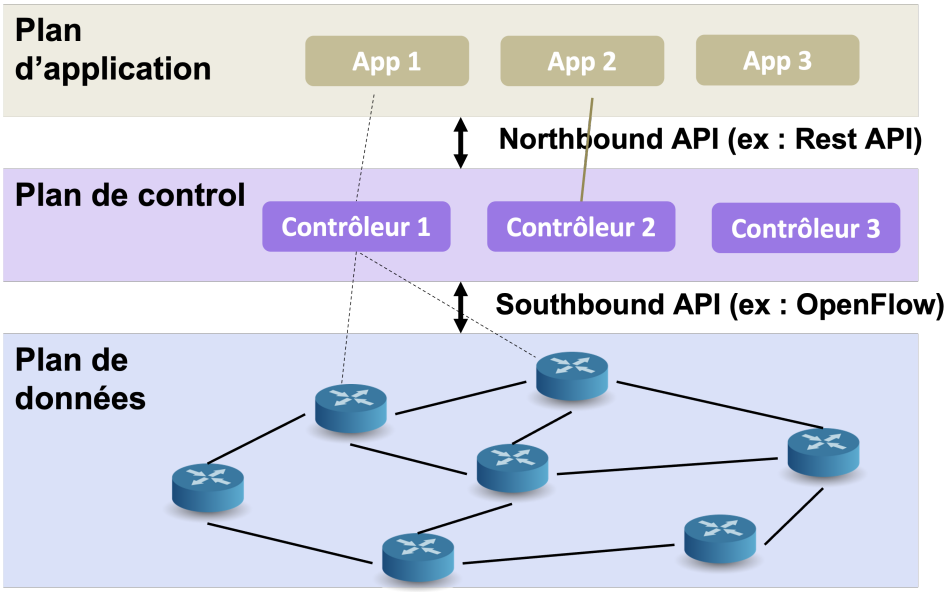



FIGURE 2.1 – l'architecture du réseau défini par logiciel (SDN)

Il existe quatre types d'interfaces SDN Hakiri *et al.* (2014) :

— **Interface *northbound* :** elle permet au contrôleur SDN et à l'application réseau d'échanger des données. Les applications réseau déterminent le type, la

forme et la fréquence des informations échangées.

— **Interface *southbound* :** elle permet la communication entre les contrôleurs du plan de contrôle et les équipements réseau du plan de données (par exemple, *OpenFlow*).

— **Interface *eastbound* :** elle permet d'interconnecter les réseaux IP (*Internet Protocol*) classiques avec les réseaux SDN. Actuellement, cette interface n'est pas normalisée et sa mise en œuvre dépend de la technologie du réseau classique. De plus, un module de traduction doit être fourni entre le SDN et le réseau IP classique.

— **Interface *westbound* :** elle permet la transmission d'informations entre différents plans de contrôle SDN de différents domaines. De plus, elle permet une configuration transparente du flux réseau sur des domaines SDN hétérogènes tout en offrant une vue globale du réseau.

### 2.2.3 Avantages

Dans SDN, le plan de contrôle est découplé du plan de données, ce qui offre un plus grand degré de contrôle sur un réseau. Grâce à ce découplage, la configuration du réseau et les performances seraient améliorées et l'innovation dans les opérations et l'architecture du réseau serait encouragée Xia *et al.* (2014).

SDN utilise l'état instantané du réseau et des politiques définies par l'utilisateur pour contrôler un réseau en temps réel. En plus du transfert de paquets au niveau de la commutation, SDN peut contrôler les caractéristiques des liens entre les équipements réseau du plan de données. En conséquence, les configurations du réseau peuvent être optimisées et les performances du réseau peuvent être améliorées. De plus, SDN offre

une plate-forme pratique pour expérimenter de nouvelles techniques et encourager de nouvelles conceptions de réseau en raison de sa programmabilité et de sa capacité à définir des réseaux virtuels indépendants.

Les avantages de SDN peuvent être catégorisés comme suit Xia *et al.* (2014) :

— Amélioration de la configuration : Afin d'obtenir un fonctionnement cohérent du réseau, les nouveaux équipements doivent être correctement configurés au sein d'un réseau existant. De plus, en raison de l'hétérogénéité entre les fabricants de périphériques réseau, la configuration du réseau implique actuellement un certain traitement manuel, qui prend du temps et est sujet aux erreurs. Aussi, le diagnostic des anomalies (*troubleshooting*) d'un réseau avec des erreurs de configuration nécessite des efforts considérables. Grâce au SDN, les périphériques réseau, tels que les commutateurs, les routeurs, les traducteurs d'adresses réseau (*Network Address Translator* NAT), les pare-feu et les équilibreurs de charge, peuvent être configurés automatiquement à partir d'un seul et même contrôleur, ce qui permet d'améliorer la configuration et la gestion du réseau.

— Amélioration des performances : Un objectif clé des opérations de réseau est de maximiser l'utilisation de l'infrastructure du réseau. Cependant, en raison de la coexistence de différentes technologies et parties prenantes dans un même réseau, l'optimisation des performances du réseau dans son ensemble présente un grand défi. Certaines approches actuelles visent à optimiser les performances d'un sous-ensemble de réseaux ou la qualité de l'expérience utilisateur pour certains services réseau. Ces approches sont basées sur les informations locales, et donc pourraient entraîner des performances sous-optimales, voire des opérations réseau conflictuelles. L'introduction du SDN permet un contrôle centralisé

avec une vue globale du réseau, ainsi qu'un contrôle de rétroaction entre les couches de l'architecture du réseau, ce qui améliorera les performances du réseau à l'échelle mondiale. En conséquence, de nombreux problèmes complexes d'optimisation des performances deviendraient gérables avec des algorithmes centralisés correctement conçus.

— Encourager l'innovation : La mise en œuvre, l'expérimentation et le déploiement d'une nouvelle conception ou idée font face à des défis immédiats. Dans les composants de réseau conventionnels, le matériel du propriétaire empêche la modification à des fins d'expérimentation, ce qui constitue le principal obstacle. Par ailleurs, même lorsque des expérimentations sont possibles, elles sont le plus souvent conduites dans des bancs de test simplifiés qui ne sont pas suffisamment fiables pour être adaptés à des applications industrielles. SDN, d'autre part, permet l'innovation en fournissant une plate-forme réseau programmable qui permet la mise en œuvre, l'expérimentation et le déploiement de nouvelles idées, de nouvelles applications et de nouveaux services générateurs de revenus de manière pratique et flexible, et la haute configurabilité de SDN permet l'expérimentation dans un réglage du monde réel.

### 2.2.4 Défis

SDN prend en charge les modèles de contrôleurs centralisés et distribués. Chaque modèle a des éléments et des exigences d'infrastructure différents ainsi que certains inconvénients Hakiri *et al.* (2014). Les modèles centralisés SDN utilisent un seul contrôleur centralisé pour gérer et superviser un réseau entier. Aussi, un point de décision unique centralise logiquement l'intelligence et les états du réseau. Malgré sa promesse de fournir un point de gestion unique et un meilleur contrôle de la cohérence

de l'état du réseau, le plan de contrôle centralisé présente plusieurs limites.

Premièrement, comme le contrôleur met à jour les commutateurs OpenFlow plus fréquemment que les routeurs traditionnels, la découverte de la topologie peut entraîner une surcharge plus élevée puisque tous les ports doivent être analysés de manière linéaire, ce qui augmente le temps de réponse.

Deuxièmement, la simplicité du modèle centralisé peut se faire au détriment de l'évolutivité du plan de contrôle, ce qui signifie que le regroupement de toutes les fonctionnalités dans un seul nœud peut nécessiter plus de puissance de calcul, d'espace de stockage et de débit de données, ce qui réduit le temps de réponse.

Troisièmement, chaque nouveau flux doit être inspecté par un contrôleur SDN centralisé dès que le paquet est introduit dans le système. Lorsqu'un nouveau flux est programmé, le contrôleur contacte tous les commutateurs le long du chemin, ce qui représente un défi d'extensibilité pour les grands réseaux. La correspondance (*matching*) de flux peut entraîner un grand nombre d'états de transfert dans les tables de flux.

Enfin, les réseaux SDN sont devenus de plus en plus complexes et hétérogènes, car ils fournissent diverses fonctions telles que les applications de sécurité, les pare-feu, la virtualisation du réseau et l'équilibrage de charge. Ces services nécessitent une coordination dans le plan de contrôle pour répondre à des objectifs de contrôle complexes et maintenir un aperçu global du réseau.

Un modèle SDN distribué élimine les points de défaillance uniques et permet une mise à l'échelle en partageant la charge entre plusieurs contrôleurs. Ce modèle a été conçu pour gérer les événements du réseau local dans les centres de données, où une

multitude d'instances de contrôleur partagent de grandes quantités d'informations.

Dans ce modèle, tous les contrôleurs doivent pouvoir voir le réseau de manière uniforme. Aussi, le mappage entre les plans de contrôle et les plans de données doit être automatisé au lieu de la configuration statique qui entraîne une répartition inégale de la charge entre les contrôleurs.

Deuxièmement, il est difficile de déterminer le nombre optimal de contrôleurs distribués pour assurer une extensibilité linéaire du réseau SDN à mesure que sa taille augmente. La majorité des systèmes SDN distribué telles que, Maestro TS et Ng (2010), McNettle Voellmy *et al.* (2012), HyperFlow Tootoonchian et Ganjali (2010), Onix Koponen *et al.* (2010), Devolved Tam *et al.* (2011) and Kandoo Hassas Yeganeh et Ganjali (2012) utilisent des algorithmes locaux pour créer des protocoles de coordination, où chaque contrôleur n'a qu'à répondre aux événements dans son voisinage local. Par conséquent, la synchronisation de l'ensemble des événements locaux et distribués nous permet d'avoir une idée globale sur le réseau.

## 2.3 L'informatique en réseau

### 2.3.1 Définition

L'informatique en réseau fait référence au processus d'interception du trafic réseau et d'exécution de calculs sur un périphérique réseau Tokusashi *et al.* (2019), tel qu'un FPGA (*Field-Programmable Gate Array*) Firestone *et al.* (2018), un ASIC (*Application-Specific Integrated Circuit*) programmable ou une carte d'interface réseau intelligente (*smart Network Interface Card* smartNIC). Ce concept implique l'informatique au sein du réseau, en utilisant des appareils qui existent déjà dans le

réseau et qui ont déjà été chargés de transférer le trafic Zilberman (2019). Comme il y a des commutateurs et des cartes réseau dans le réseau, l'informatique en réseau ne nécessite aucun périphérique supplémentaire. Par conséquent, l'informatique en réseau a une surcharge minimale, car aucun espace ou coût supplémentaire ne sont requis.

Même si l'informatique en réseau et l'informatique en périphérie réduisent la charge de calcul sur l'informatique en nuage, elles diffèrent considérablement. L'informatique en réseau transforme le réseau TCP *(Transmission Control Protocol)*/IP en un réseau de déploiement des applications plutôt qu'en un pipeline de données. En conséquence, les ressources inactives des périphériques réseau peuvent être pleinement utilisées pour effectuer certains calculs de données, réduisant ainsi la charge de transmission de données sur le réseau et la charge de calcul sur l'informatique en nuage Hu *et al.* (2021).

Il a été démontré que l'informatique en réseau peut augmenter considérablement le débit et réduire la latence Jin *et al.* (2017) Dang *et al.* (2020). Selon Tokusashi *et al.* (2018), ses cas d'utilisation vont bien au-delà de la mise en réseau (par exemple, le consensus Dang *et al.* (2015), le traitement des données Jepsen *et al.* (2018) et l'apprentissage automatique Sapio *et al.* (2017)). Les applications basées sur des caches sont les cas d'utilisation les plus populaires pour l'informatique en réseau Jin *et al.* (2017). Il est idéal de gérer les demandes d'informations fréquemment répétées en plaçant un dispositif de calcul dans le réseau qui évite les traversées du réseau Dang *et al.* (2020).

### 2.3.2 Historique

À la recherche de nouvelles solutions pour améliorer les fonctionnalités du réseau, les chercheurs ont tenté de trouver des mécanismes efficaces pour injecter des programmes personnalisés – encapsulés dans des "capsules actives" – dans les nœuds du réseau avec les données classiques. Le code encapsulé dans les paquets est l'idée principale derrière les réseaux actifs (*Active Networks*) Tennenhouse et Wetherall (1996) où les nœuds du réseau traitent et effectuent activement des calculs sur les données des utilisateurs. La technologie de réseau défini par logiciel (SDN) offre une excellente flexibilité dans la réalisation de la programmation du réseau en séparant les plans de contrôle et de données. Alors que le contrôleur SDN interagit avec le plan de données à l'aide du protocole OpenFlow McKeown *et al.* (2008), P4 Bosshart *et al.* (2014) a été introduit pour décrire la façon avec laquelle le paquet doit être traité. Un programme P4 permet de classer un en-tête de paquet et d'exécuter une action sur un paquet entrant. Par conséquent, nous revenons à ce qui était auparavant défini comme un réseau actif et à ce que l'on appelle maintenant l'informatique en réseau.

En résumé, outre les approches d'accélération matérielle, quatre étapes majeures ont marqué le développement de l'informatique en réseau :

1. Milieu des années 1990 : Les réseaux actifs ont introduit des fonctions programmables dans le réseau, ouvrant la voie à un changement de paradigme dans les architectures de réseau.
2. 2004 : SDN a introduit le concept de séparation des plans de contrôle et de données, et des interfaces ouvertes se sont développées entre eux.
3. 2010 : L'API OpenFlow et les systèmes d'exploitation réseau ont adopté des techniques de contrôle et de plan de données avancées et évolutives.

4. 2014 : P4 a été introduit comme une étape vers des commutateurs plus flexibles dont les fonctions sont spécifiées et modifiées dynamiquement dans les réseaux.

### 2.3.3 Le calcul dans les réseaux traditionnels vs l'informatique en réseau

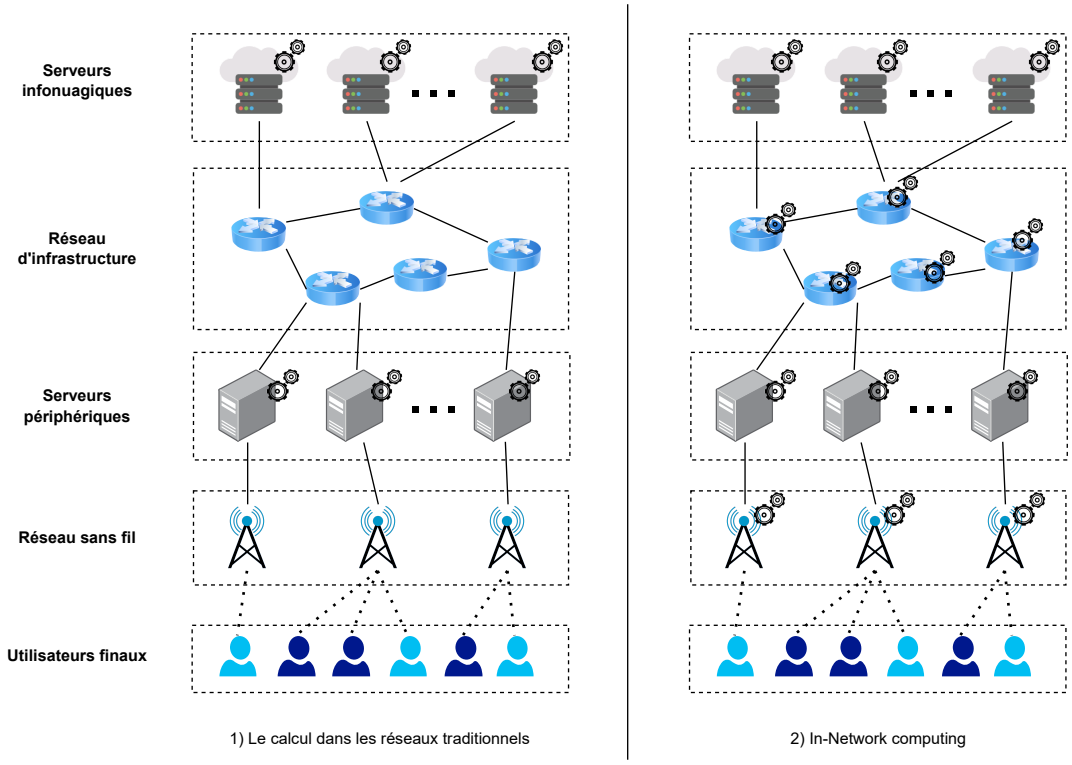



FIGURE 2.2 – le calcul dans les réseaux traditionnels vs l'informatique en réseau

Le modèle informatique traditionnel repose sur des processeurs et les données sont transmises à travers le réseau à des nœuds dotés de ressources de calcul riches Hu *et al.* (2021) (par exemple, l'informatique en nuage et l'informatique en périphérie dans la partie 1) de la figure 2.2). Dans ce modèle, les serveurs avec une capacité de calcul importante sont responsables pour faire le traitement des requêtes des utilisateurs finaux. Les équipements du réseau tels que les stations de base, les routeurs et les

commutateurs assurent uniquement la communication entre les différents dispositifs et le transfert de données. Les utilisateurs finaux ne peuvent décharger leurs tâches que sur des serveurs périphériques ou le nuage, ce qui peut entraîner une latence élevée et une congestion du réseau.

Cependant, les environnements de l'informatique en réseau permettent l'utilisation des ressources des équipements réseau afin d'exécuter des tâches de traitement à faible intensité comme le montre la partie 2) de la figure 2.2. Les équipements réseau peuvent non seulement transférer des paquets et des données, mais également traiter des tâches déchargées par les utilisateurs finaux. Ceci réduit considérablement la latence et allège le réseau. Par conséquent, une certaine qualité de service peut être assurée, principalement pour les utilisateurs finaux contraints par les délais.

### 2.3.4 Avantages

En s'inspirant des réseaux actifs au début de son développement, l'informatique en réseau a fait des progrès substantiels. Selon Ports et Nelson (2019) qui étudie les applications avec des exigences de haute performance, le paradigme de l'informatique en réseau offre des avantages tels que la réduction de la latence, l'augmentation du débit et la réduction du trafic réseau.

Tout d'abord, en déchargeant les tâches de contrôle contraintes au délai des serveurs infonuagiques et périphériques vers des équipements de réseau local qui peuvent effectuer le traitement, le trafic sera considérablement réduit au niveau du nuage et de la périphérie du réseau. De plus, comme le traitement est effectué au sein du réseau, cela signifie que les transactions sont terminées dans leur chemin plutôt que d'atteindre un hôte final, ce qui permet d'économiser la latence introduite par l'hôte

final.

Le débit est un autre avantage de performance de l'utilisation de l'informatique en réseau, qui est amélioré en augmentant le taux de traitement des paquets à l'aide d'une classe de commutateurs, conçus comme des pipelines, déplaçant en continu les données sans blocage. En fait, le traitement des paquets lors du transfert par les équipements réseau permet de diminuer la quantité des données transférées et par conséquent améliore le débit du réseau.

La faible consommation d'énergie est également un avantage de l'informatique en réseau. Étant donné que la surcharge de l'informatique en réseau est faible et que les commutateurs réseau font partie du réseau, sachant que la majeure partie de la consommation d'énergie est déjà payée par le transfert de paquets, l'informatique en réseau est économe en énergie. Plus précisément, l'énergie consommée par les périphériques réseaux lors du traitement des paquets est relativement faible comparée à l'énergie inévitable consommée par leur fonctionnement basique.

### 2.3.5 Défis

Récemment, les progrès des équipements réseau programmables dans le plan de données ouvrent de nombreuses opportunités pour décharger des opérations ou des fonctionnalités vers les commutateurs. Cependant, la contrainte mémoire limitée des commutateurs reste un défi majeur que de nombreuses applications peinent à relever. En outre, de nombreuses applications (par exemple, vidéo de réalité virtuelle à 360 degrés) font face à un compromis entre les performances et l'utilisation de la mémoire. Par exemple, dans le scénario de mise en cache en réseau, le taux de réussite du cache est directement déterminés par la taille de la mémoire de l'équipement

réseau Jin *et al.* (2017) Liu *et al.* (2016).

De même, l'épuisement de la mémoire du dispositif entraîne directement un ralentissement des performances des applications telles que l'équilibrage de charge Miao *et al.* (2017) et la surveillance des performances du réseau Narayana *et al.* (2017). Même la fonction la plus élémentaire des commutateurs, telle que le transfert de paquets, peut souffrir d'un espace mémoire limité. Pour faire face à la contrainte de mémoire, une solution est la conception d'un ASIC de commutation avec une logique interne personnalisée et des câbles pour l'accès externe à la mémoire vive dynamique (*Dynamic Random Access Memory* DRAM). Cependant, cela s'avère coûteux du fait de la conception de nombreux modules DRAM parallèles associés à un contrôleur DRAM. Par conséquent, la DRAM externe sur commutateur n'est pas largement utilisée par les fournisseurs dans les commutateurs à puce unique d'aujourd'hui.

Une autre solution possible consiste à utiliser les ressources inutilisées et la DRAM des réseaux du centre de données. Cependant, il existe des défis techniques en matière de performances, de tolérance aux pannes et d'équilibrage de charge, mais la mémoire du commutateur programmable peut être étendue de manière rentable. Par exemple, les travaux de Kim *et al.* (2020) explorent en particulier la possibilité de réaffecter la DRAM installée sur les serveurs de centre de données pour étendre la mémoire disponible sur les commutateurs du plan de données. Au lieu de construire un autre ASIC de commutation avec une capacité d'accès DRAM externe personnalisée, ils réutilisent simplement un ASIC de commutation programmable existant construit uniquement avec une mémoire sur puce. L'approche proposée conçoit trois primitives de mémoire distante : un tampon de paquets distant, une table de recherche distante et un magasin d'état distant, tirant parti des opérations d'accès direct à la mémoire à distance (Remote Direct Memory Access RDMA). Le résultat expérimen-

tal montre que le système proposé atteint des performances élevées en matière de bande passante, entraînant un petit coût de latence et absolument 0% de surcharge CPU (*Central Processing Unit*).

La consommation d'énergie des dispositifs programmables est l'un des indicateurs de performance clés qui reflètent directement l'efficacité de traitement des paquets de la puce, mesurée en millions de paquets par seconde par Watt (Mpps/Watt) Pongrácz *et al.* (2013).

Par conséquent, cela concerne les chercheurs dans le domaine de l'informatique en réseau. Contrairement aux puces traditionnelles avec un nombre de transistors et une fréquence de commutation inférieurs, les puces d'aujourd'hui sont de plus petit volume et embarquent un grand nombre de transistors qui leur permettent d'avoir une capacité de traitement plus élevée. Par conséquent, cela conduit à plus de dissipation de chaleur qui reflète directement le temps de calcul et génère des coûts de refroidissement supplémentaires. Pour surmonter cette dissipation de chaleur, les travaux de Yang *et al.* (2019) ont changé le fonctionnement des commutateurs tels que les FPGAs. Ces commutateurs fonctionnent selon deux modes :

1. Le mode veille qui réduit la dissipation de chaleur.
2. Le mode basse consommation qui réduit la puissance.

## 2.4 Le langage de programmation P4

### 2.4.1 Définition

La fondation de réseau ouvert (ONF) définit P4 (*Programming Protocol-independent Packet Processors*) comme étant un langage de programmation libre, spécifiant com-

ment les périphériques de plan de données tels que les commutateurs, les routeurs ou les cartes réseau traitent les paquets ONF (2022b).

P4 fournit une API qui ne tient pas compte du matériel réel sur lequel le commutateur est implémenté, augmentant ainsi le niveau d'abstraction pour la programmation des réseaux Bosshart *et al.* (2014). Elle peut être utilisée comme interface entre le contrôleur et les commutateurs. Par conséquent, les futures versions d'OpenFlow devraient laisser le contrôleur contrôler le commutateur, plutôt que de le contraindre. Afin de mettre en œuvre la spécification sur une large gamme de commutateurs matériels et logiciels, il est important de trouver un équilibre entre expression et simplicité dans le langage P4.

Les concepteurs du langage P4 ont trois objectifs de conception principaux Bosshart *et al.* (2014) :

— **La reconfigurabilité :** l'analyse et le traitement des paquets doivent être reprogrammables par le contrôleur.

— **L'indépendance du protocole :** il ne devrait y avoir aucune restriction sur les formats de paquets sur le commutateur. Le contrôleur devrait plutôt être en mesure de spécifier un analyseur de paquets (*packet parser*) qui extrait les champs d'en-tête avec des noms et des types particuliers, ainsi qu'un ensemble de tables de correspondance + action (*match+action*) typées.

— **L'indépendance de la cible :** il ne devrait pas être nécessaire que le programmeur du contrôleur connaisse les détails du commutateur sous-jacent, tout comme un programmeur au langage C n'a pas besoin de connaître les détails du processeur sous-jacent. Lors de la conversion d'une description indépendante de la cible écrite en P4 en un programme dépendant de la cible utilisé pour confi-

gurer le commutateur, un compilateur doit prendre en compte les capacités du commutateur.

### 2.4.2 Les éléments de P4

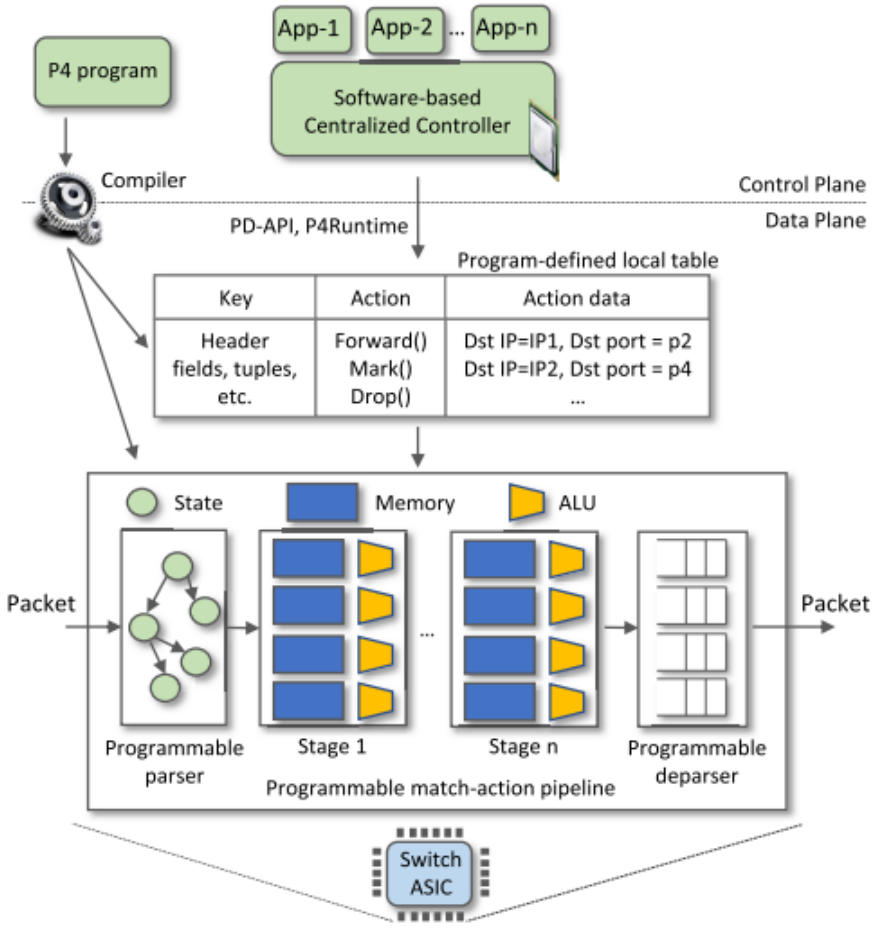



FIGURE 2.3 – un plan de données basé sur PISA et son interaction avec le plan de contrôle Kfoury *et al.* (2021).

La figure 2.3 montre un plan de données basé sur PISA. PISA est l'acronyme de *Protocol Independent Switch Architecture*, qui désigne un modèle de traitement de

paquets. Il inclut un *parser*, un pipeline *match+action* et un *deparser*. Lorsqu'un paquet arrive, il passe premièrement par le *parser* qui extrait les en-têtes du paquet. Ensuite, le pipeline *match+action* exécute des opérations à l'aide de l'ALU (*Arithmetic Logic Unit*) sur les en-têtes du paquet. Finalement, le *deparser* assemble les en-têtes du paquet et les sérialise pour la transmission.

Globallement, P4 contient les éléments clés suivants Kaur *et al.* (2021) :

A) **Format des en-têtes :** les types d'en-tête de P4 sont similaires à la structure C. Le langage P4 a l'avantage de traiter les en-têtes de protocoles standards tels qu'Ethernet, IP, TCP ou UDP *(User Datagram Protocol)* ainsi que les en-têtes personnalisés. Le programme P4 doit définir explicitement tous les types des en-têtes standards et personnalisés et leurs séquences d'analyse.

   1. ***Parser* :** un *parser* spécifie comment les en-têtes du paquet seront analysés et identifie correctement les en-têtes présents dans le paquet entrant. Une machine à états finis représente le *parser*, où les états extraient les structures d'en-tête individuelles et les stockent dans des variables d'exécution. Selon la valeur d'un champ d'en-tête analysé, les transitions peuvent être effectuées de manière conditionnelle.
   2. **Pipeline *match+action* :** il contient plusieurs tables *match+action* qui sont utilisées pour déterminer ce qu'il faut faire lorsque les champs d'en-tête d'un paquet correspondent à des valeurs prédéfinies. De plus, il définit une fonction de contrôle qui détermine comment les tables doivent être exécutées. Aussi, il définit par exemple le port par lequel le paquet va sortir et la nouvelle valeur de l'adresse MAC (*Media Access Control*) source et destination.

3. ***Deparser*** **:** le *deparser* sérialise les en-têtes modifiés dans le paquet et l'envoie au commutateur suivant.

B) **Compilateurs P4 :** les compilateurs P4 traduisent le code P4 en une représentation spécifique à la cible et sont divisés en deux parties : un *front-end* qui convertit le code P4 en une représentation indépendante de la cible et un *back-end* qui mappe la représentation indépendante à la cible. Il existe deux sorties du compilateur P4 :

— Une configuration implémentant le comportement de transfert dans les commutateurs conformément au programme d'entrée P4.

— Une API de contrôleur qui contrôle les commutateurs du plan de données.

C) **L'environnement d'exécution P4 (*P4 Runtime*) :** La communication entre le plan de données et le plan de contrôle est gérée par l'environnement d'exécution P4. L'un des principaux avantages de *P4 Runtime* est la possibilité pour le contrôleur de contrôler n'importe quel plan de données, qu'il soit construit à partir d'ASIC, de FPGA et de NPU (*Network Processing Unit*) fixes ou programmables. De plus, le plan de contrôle ne dépend pas du placement de l'environnement d'exécution P4, par exemple, il peut s'exécuter localement ou à distance comme dans le cas du réseau défini par logiciel.

### 2.4.3 Programmer une cible avec P4

La figure 2.4 donne un aperçu sur le processus général de programmation d'un périphérique P4 Ibanez *et al.* (2019). Trois composants sont fournis par le fournisseur d'un dispositif de traitement de paquets :

— Le périphérique cible de traitement des paquets.

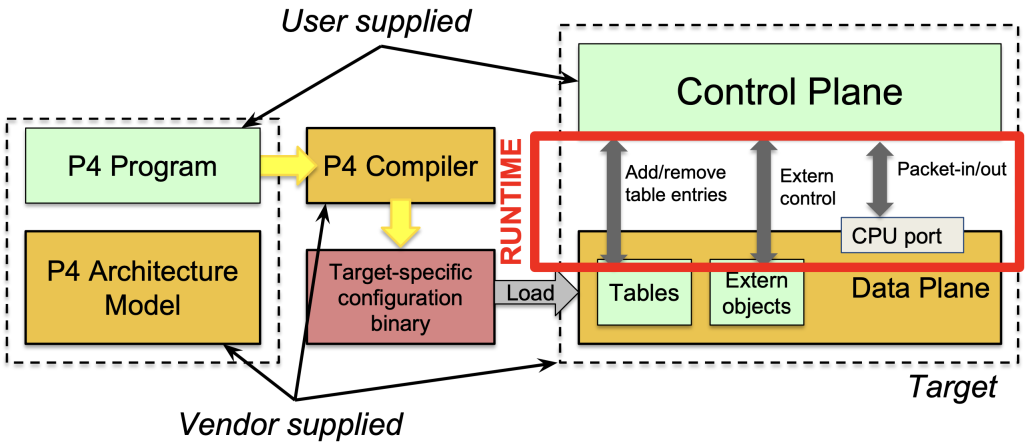



FIGURE 2.4 – le processus de programmation d'une cible P4 Ibanez *et al.* (2019)

— Un modèle d'architecture permettant au programmeur P4 d'accéder aux fonctionnalités programmables d'une cible.

— Un compilateur qui convertit le programme P4 du programmeur en un fichier binaire de configuration spécifique à la cible qui indique à la cible comment traiter les paquets.

Le programmeur fournit un programme principal P4 qui instancie le modèle d'architecture en remplissant les composants programmables, ainsi qu'un logiciel de contrôle (c'est-à-dire, un plan de contrôle) pour contrôler le dispositif de traitement de paquets au moment de l'exécution.

### 2.4.4 Les cas d'utilisation de P4

Il existe des cas d'utilisation intéressants de P4 pour différents réseaux :

— Équilibreur de charge de couche 4 (*SilkRoad* Miao *et al.* (2017)) : *SilkRoad* utilise des primitives matérielles simples disponibles dans les ASICs actuels pour

l'équilibrage de charge avec état. Cette technologie élimine le besoin d'une autre couche logicielle entre le trafic d'application et les serveurs d'application en fournissant un chemin direct entre les deux. Le prototype P4 de *SilkRoad* a été construit sur un commutateur P4 (switch.p4) de base et compilé sur un ASIC de commutation programmable. Le commutateur P4 de base implémente diverses fonctionnalités de mise en réseau nécessaires pour les centres de données infonuagiques typiques dans environ 5000 lignes de code P4.

— Contrôle de la congestion à faible latence (*NDP* Handley *et al.* (2017)) : *NDP* est une nouvelle architecture de transport de centre de données qui fournit des délais d'exécution quasi optimaux pour les transferts courts et un débit élevé dans une variété de scénarios, y compris l'*incast*. Une implémentation avec P4 du commutateur *NDP* a été développée comme une preuve de concept que le traitement *NDP* est suffisamment simple pour être facilement déployé dans des commutateurs programmables.

— Télémétrie réseau intrabande (*In-band Network Telemetry* INT Kim *et al.* (2015)) : La télémétrie réseau intrabande (INT) est une nouvelle abstraction qui permet aux paquets de données d'interroger l'état interne du commutateur, comme la taille de la file d'attente, l'utilisation de la liaison et la latence de la file d'attente. Un prototype de INT en langage P4 a été proposé en utilisant un commutateur logiciel comme plate-forme d'implémentation.

— Cache en réseau rapide (*NetCache* Jin *et al.* (2017)) : *NetCache* est une nouvelle architecture de stockage clé-valeur qui met en cache les données dans le réseau en tirant parti de la puissance et de la flexibilité des commutateurs programmables de nouvelle génération. Le commutateur du plan de données du prototype de *NetCache* a été écrit en P4.

— Agrégation pour les applications *MapReduce* (*DAIET* Sapio *et al.* (2017) : *DAIET* est un système qui effectue l'agrégation de données dans le réseau appliqué aux applications basées sur *MapReduce*. Il représente un exemple de système qui peut être construit à l'aide de P4 pour décharger le calcul sur le plan de données.

## 2.5 Découpage virtuel du réseau

### 2.5.1 Définition

Le découpage virtuel du réseau (*network slicing*) permet de partitionner un réseau physique en plusieurs réseaux virtuels, chaque réseau virtuel étant personnalisé et optimisé pour un type d'application spécifique Zhang (2019) comme le montre la figure 2.5. L'intégration de tranches permet le mappage de ces réseaux virtuels aux ressources physiques. Cela inclut la découverte et l'allocation des ressources. Ces réseaux virtuels sont connus sous le nom de tranches et sont activés par des technologies telles que SDN.

Le découpage en tranches a émergé avec le réseau 5G (cinquième génération) et l'objectif était de permettre trois catégories de services :

— La connectivité haut débit mobile améliorée (*enhanced Mobile Broadband Connectivity* eMBB).

— Les communications massives de type machine (*massive Machine Type Communications* mMTC).

— Les services de communications critiques ultra-fiables.

Le cycle de vie d'une tranche comprend quatre phases Zhang (2019) :

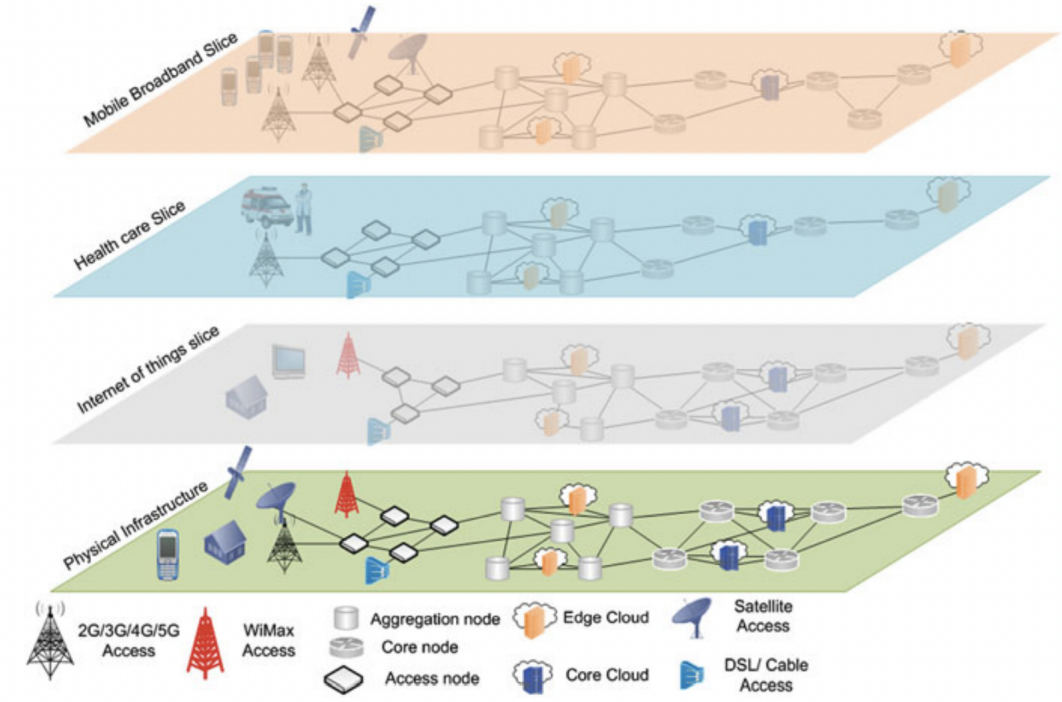



FIGURE 2.5 – découpage virtuel du réseau 5G Kazmi *et al.* (2019)

1. La conception.
2. L'orchestration.
3. L'activation.
4. L'assurance du temps d'exécution.
5. Le démantèlement.

Un catalogue des blocs de construction des tranches de réseau est créé lors de la conception. Des tranches spécifiques sont sélectionnées au cours de la deuxième phase en fonction des besoins des utilisateurs. Les indicateurs de performance (par exemple, l'utilisation des ressources) sont surveillés en permanence et corrélés avec la qualité de service (par exemple, le délai de bout en bout) et les tranches de réseau peuvent être reconfigurées si nécessaire pour respecter les accords sur les niveaux de service. La

gestion du cycle de vie est assurée par la fonctionnalité de gestion et d'orchestration (*Management ANd Orchestration* MANO) dans les réseaux de cinquième génération (5G).

### 2.5.2 Principes de découpage virtuel du réseau

Avec les principes de découpage virtuel du réseau, plusieurs réseaux logiques sont créés au-dessus d'une infrastructure de réseau physique commune pour prendre en charge différentes applications commerciales Kazmi *et al.* (2019). Trois grands principes doivent être suivis lors du découpage d'un réseau :

1. **Isolation des tranches :** l'isolation des tranches active la fonctionnalité selon laquelle les tranches d'un locataire (c'est à dire, un utilisateur qui peut accéder aux ressources partagées avec des privilèges et des droits d'accès spécifiques) sont indépendantes des autres tranches afin de garantir les performances avec la sécurité des différents locataires. Lors du découpage du réseau, l'isolement des tranches est une propriété importante à garantir. En outre, l'isolement limite la capacité du locataire à accéder/modifier d'autres tranches des autres locataires.
2. **Élasticité :** afin d'utiliser efficacement les ressources, l'élasticité permet la modification dynamique des allocations de ressources entre les tranches de locataires. Il est possible d'atteindre l'élasticité en déplaçant les fonctions du réseau virtuel, en augmentant ou en réduisant les ressources allouées et en reprogrammant les fonctionnalités des éléments de contrôle et de données.
3. **Personnalisation de bout en bout :** l'utilisation de ressources personnalisées garantit que les ressources partagées sont utilisées efficacement par différents locataires. Afin de permettre la personnalisation et la virtualisation

des fonctions réseau, le réseau défini par logiciel et l'orchestration du réseau peuvent être utilisés pour faciliter l'allocation des ressources de manière agile.

### 2.5.3 Technologies permettant le découpage virtuel du réseau

Les principales technologies permettant le découpage virtuel du réseau sont les suivantes Kazmi *et al.* (2019) Zhang (2019) :

— **Le réseau défini par logiciel (SDN) :** en utilisant le réseau défini par logiciel, plusieurs instances de client peuvent être prises en charge efficacement par une infrastructure commune. Il est naturel que le réseau défini par logiciel prenne en charge le découpage du réseau puisque le contrôleur SDN fournit un ensemble abstrait de ressources isolées par la virtualisation. Les contrôleurs SDN pourraient orchestrer dynamiquement des tranches de réseau.

— **Virtualisation des fonctions réseau (NFV) :** grâce à la virtualisation, des tranches de réseau peuvent être créées sur des ressources physiques partagées de manière flexible. Dans le contexte du découpage virtuel du réseau, la virtualisation des fonctions réseau permet la gestion des fonctions de réseau virtuelles, le chaînage de services et l'intégration de fonctions de réseau virtuelles contraintes à la latence.

— **Gestion et orchestration (MANO) :** les réseaux en tranches rendent la gestion du réseau plus complexe, en particulier lorsqu'il y a de nombreuses tranches, ce qui rend essentielles les solutions de gestion et d'orchestration automatisées. Le concept de services réseau développé par le groupe ETSI NFV peut être réutilisé pour permettre une gestion automatisée et flexible des tranches. En raison de la tendance technologique actuelle, le *framework* NFV

MANO ETSI (2014) est un choix naturel pour la gestion et l'orchestration des tranches de réseau 5G.

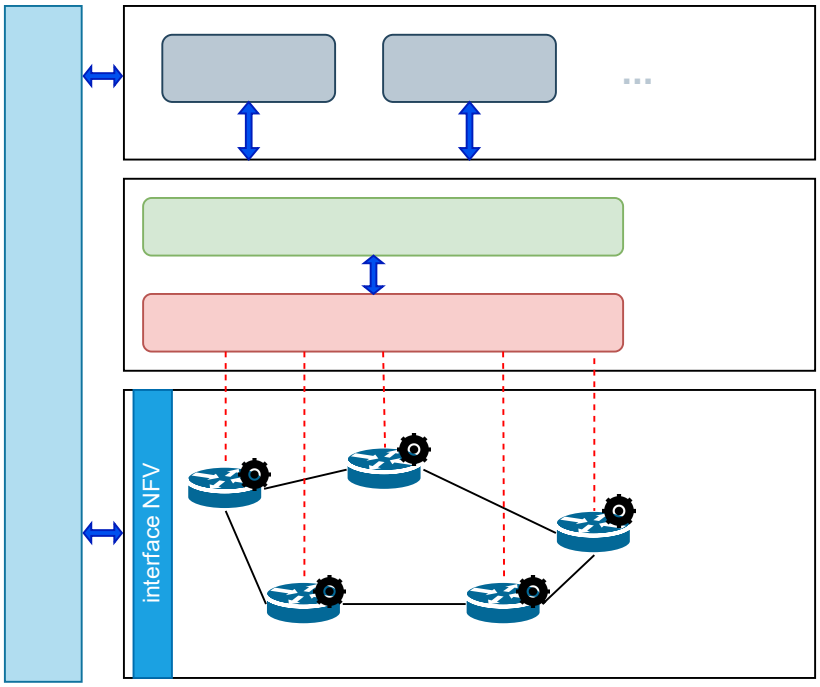



FIGURE 2.6 – architecture du réseau défini par logiciel avec la virtualisation des fonctions réseau

La figure 2.6 inspirée de l'article Ojo *et al.* (2016) représente l'architecture du réseau défini par logiciel avec la virtualisation des fonctions réseau. Le plan de données est constitué de plusieurs périphériques réseau avec des capacités de calcul. Ces périphériques sont capables d'héberger des fonctions réseau virtuelles qui sont compatibles avec le réseau défini par logiciel. Ces fonctions "*SDN-enabled*" sont appelées : éléments de réseau virtuels. Dans l'architecture, il y a également le plan de gestion et d'orchestration contenant un orchestrateur, un gestionnaire d'infrastructure virtualisée et un gestionnaire de fonctions réseau virtuelles. L'orchestrateur est responsable de gérer les ressources et de coordonner les services. Le gestionnaire d'infrastructure

virtualisée contrôle les ressources de l'infrastructure NFV (c'est à dire, l'environnement dans lequel les VNFs s'exécutent et qui comprend la couche de virtualisation) et collecte les métriques de performance. Aussi, le gestionnaire de fonctions réseau virtuelles gère le cycle de vie des VNFs et augmente et diminue leur quantité.

## 2.6 Conclusion

Dans ce chapitre, nous avons défini et détaillé quelques concepts généraux tels que le réseau défini par logiciel, l'informatique en réseau, le langage de programmation P4 et le découpage virtuel du réseau. Dans le prochain chapitre, nous allons présenter une revue de la littérature sur les technologies permettant le déploiement de l'informatique en réseau telles que SDN et NFV, la programmabilité en réseau et le provisionnement des infrastructures.

# CHAPITRE III

# REVUE DE LA LITTÉRATURE

## 3.1 Introduction

Ce chapitre constitue un résumé des travaux qui ont été faits en lien avec les aspects liés à notre projet de recherche. Plus exactement, nous nous intéressons aux articles qui ont présenté les technologies permettant le déploiement de l'informatique en réseau telles que SDN et NFV, ainsi que les travaux concernant la programmabilité en réseau et le provisionnement des infrastructures.

## 3.2 Technologies permettant le déploiement de l'informatique en réseau : SDN et NFV

En incorporant l'informatique en réseau dans une infrastructure réseau contenant l'infonuagique et l'informatique en périphérie, les tâches informatiques déchargées dans le réseau sont exécutées par une infrastructure intégrée qui optimise les ressources de réseau, de cache et de calcul. Chaque tâche de calcul peut être décomposée en un ensemble de sous-tâches. Chaque sous-tâche peut être réalisée en tant que fonction virtualisée sur des conteneurs ou des *unikernels*, déployés dans le réseau.

Selon des travaux connexes antérieurs, la virtualisation est souvent utilisée pour réduire la charge du réseau et accélérer le traitement dans le réseau à travers des boîtiers intermédiaires virtuels (*virtual middleboxes*). Par exemple, les auteurs de Mai *et al.* (2014) ont proposé *NetAgg* pour effectuer l'agrégation le long du chemin sur les nœuds du réseau au lieu de la périphérie. *NetAgg* consiste en un serveur appelé *Agg box* connecté par une liaison haut débit à des commutateurs du réseau. Ces *Agg box* sont utilisés pour exécuter des fonctions d'agrégation spécifiques à l'application à chaque saut afin de réduire le trafic réseau et de résoudre le goulot d'étranglement du réseau, augmentant ainsi les performances de l'application. Deux applications ont été considérées comme cas d'études : le moteur de recherche distribué *Solr* et le traitement qui s'exécute par lot avec *Hadoop*.

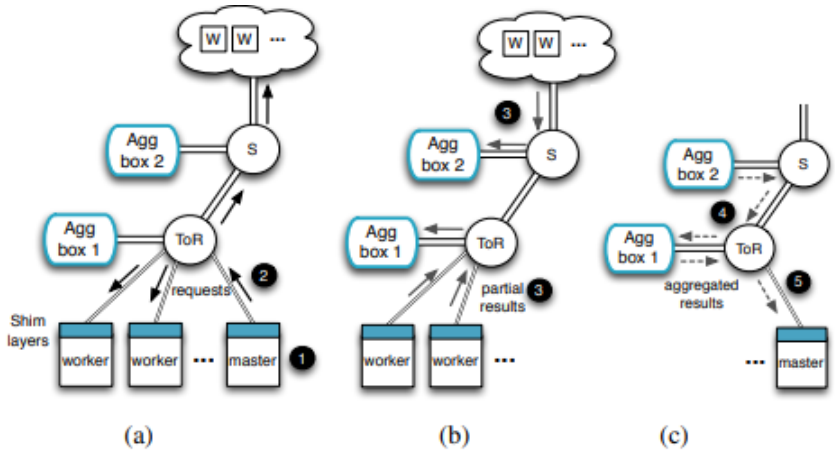



FIGURE 3.1 – flux opérationnel de *NetAgg* Mai *et al.* (2014)

La figure 3.1 montre le flux opérationnel de *NetAgg* déployé dans un centre de données. *NetAgg* comporte deux composantes principales :

— Le serveur *Agg box* : qui est connecté à un commutateur à travers un lien haut débit, et exécute des fonctions d'agrégation sur les données d'application.

— La couche de gestion (appelée *"Shim layer"*) : qui contrôle la redirection des données d'application et gère la collecte de résultats partiels.

Le flux opérationnel entre ces deux composants suit les étapes (de 1 à 5) montrées dans la figure 3.1 :

1. Lorsqu'un client envoi une requête, elle arrive dans un premier lieu au nœud maître. Ce dernier subdivise la requête en plusieurs sous-requêtes (Figure 3.1 (a)).
2. Le nœud maître envoie les sous-requêtes vers les nœuds de travail en passant à travers la couche de gestion qui permet de faire le suivi de ces sous-requêtes sans leur apporter de modification.
3. Les résultats des calculs partiels obtenus de la part des nœuds de travail sont interceptés par la couche de gestion de chaque nœud, qui les redirige vers le premier nœud *Agg box* qui se trouve sur le chemin de retour vers le nœud maître 3.1 (b).
4. Chaque nœud *Agg box* agrège les résultats de calculs partiels, et les envoie ensuite vers le prochain nœud *Agg box* qui se trouve sur le chemin de retour vers le nœud maître 3.1 (c).
5. Le nœud *Agg box* le plus proche du nœud maître envoie les résultats finals agrégés au nœud maître. La couche de gestion du nœud maître se charge par la suite de rediriger le tout vers l'application.

Une autre utilisation de la virtualisation se fait par le biais de commutateurs virtuels, par exemple, dans Zhang *et al.* (2014), les auteurs présentent une plate-forme *SmartSwitch* construite sur des serveurs connectés à des cartes réseau à haut débit. La plate-forme proposée vise à simplifier le développement de services qui fusionnent le calcul et le stockage dans le réseau. Les auteurs appuient le fait que *SmartSwitch*

peut permettre aux routeurs d'effectuer un traitement complexe sur les paquets, ou il pourrait être intégré à un composant réseau pour satisfaire les contraintes de latence. Les auteurs proposent d'abord *MemSwitch*, qu'ils prétendent être plus flexible que les commutateurs conventionnels en offrant une manière dynamique de contrôler la redirection du réseau et l'équilibrage de charge. Deuxièmement, leur *framework* introduit une plate-forme capable de déplacer le stockage d'un serveur infonuagique vers des *middleboxes*. Troisièmement, *MemSwitch* offre un traitement et un filtrage à grande vitesse des données à l'aide de *memcach* (c'est-à-dire, un système général de mise en cache de la mémoire distribuée). Pour résumer, la contribution de ce travail réside dans la proposition d'un *MemSwitch* qui est un commutateur d'équilibrage de charge qui diminue la latence des requêtes, augmente le débit et permet la mise en cache dans le réseau.

D'autre part, à l'ère de l'informatique en nuage, l'informatique en périphérie et de la conteneurisation, la virtualisation des microservices est devenue très populaire dans l'industrie et le milieu universitaire en raison de ses avantages pratiques. Comme les microservices sont des services autonomes qui exécutent une tâche spécifique, ils constituent un excellent choix pour l'informatique en réseau. En effet, les périphériques réseau tels que les commutateurs ont des ressources limitées et peuvent ne pas être en mesure de gérer de grands services monolithiques. Ainsi, il est préférable de les décomposer en microservices pour les rendre plus petits et plus gérables Kecskemeti *et al.* (2016). Outre l'aspect substantiel de l'architecture de microservices, qui est l'évolutivité, elle traite également l'hétérogénéité.

La virtualisation permet le déploiement de microservices sur différents appareils de diverses capacités dans une infrastructure hétérogène. Ainsi, les microservices virtualisés ont été un facteur clé dans l'amélioration des performances des applications

infonuagiques Barik *et al.* (2016) Sun *et al.* (2015). Les applications sont mises en œuvre dans des conteneurs en raison de leur nature légère et de leur déploiement rapide par rapport aux machines virtuelles.

Les auteurs de Amadeo *et al.* (2019) ont proposé une architecture basée sur SDN de services informatiques dans les infrastructures de périphérie composées de trois plans différents : les plans de données, de contrôle et d'application. Le plan de données comprend les nœuds de réseau composant le domaine géré (c'est-à-dire, les nœuds d'entrée, le réseau fédérateur, les routeurs intermédiaires et d'accès). Le plan de contrôle comprend le contrôleur SDN qui gère son domaine selon des politiques d'orchestration spécifiques définies au niveau du plan d'application. De plus, une architecture de plan de contrôle hiérarchique peut être envisagée à des fins d'évolutivité. Globalement, dans le contexte de l'informatique en réseau, le plan de contrôle centralisé est principalement responsable de l'optimisation du réseau en calculant les politiques de réseau optimisées de manière centralisée et en installant les tables de correspondance de flux sur les appareils du plan de données pour prendre en charge les opérations de traitement et de transfert. Par exemple, un pipeline contextuel est conçu pour augmenter les possibilités de satisfaction des services Li *et al.* (2021). Le plan de données utilise d'abord le pipeline de mise en cache et de calcul contextuel pour les paquets entrants afin de rechercher des résultats précalculés qui accéléreront considérablement le processus de service et réduiront les coûts de communication et de calcul. Dans Li *et al.* (2021), le contrôleur SDN collecte périodiquement des informations de surveillance importantes (par exemple, topologie, charge de trafic, listes de fonctions, charge de calcul et contenu) pour améliorer ses politiques. De plus, le contrôleur SDN installe des tables de correspondance de flux sur les dispositifs de plan de données pour prendre en charge les opérations de traitement et de transfert

des paquets de données.

## 3.3 La programmabilité en réseau

Dans le modèle informatique de réseau basé sur SDN, le volet de logique d'application peut être implémenté avec la programmation P4 et peut être déployée sur un commutateur. Ici, le contrôleur SDN joue un rôle crucial en décidant du placement optimal du traitement et de la planification des tâches Tokusashi *et al.* (2019). Plusieurs travaux de recherche proposent des stratégies de placement de traitement optimales en exploitant les capacités du plan de contrôle centralisé. Par exemple, *In-Network Computing Architecture* (INCA) qui instancie les fonctions aux endroits appropriés pour répondre à la contrainte de qualité d'expérience est proposée dans Albalawi *et al.* (2019). Dans INCA, le contrôleur suit la charge de travail de traitement des périphériques réseau ainsi que les routes optimales entre eux. Un autre travail Rüth *et al.* (2018) utilise SDN et P4 dans les systèmes de contrôle industriels pour décharger les tâches de contrôle contraintes au délai du nuage vers des équipements du réseau local qui peuvent effectuer un traitement avec un délai limité.

De même, l'agrégation de données et la mise en cache en réseau sont les fonctionnalités les plus importantes dans la littérature qui tirent parti du réseau défini par logiciel pour mettre en œuvre les services de l'informatique en réseau. Le contrôleur est principalement responsable de la définition de l'arborescence d'agrégation et de l'envoi des règles de traitement correspondantes au commutateur sous la forme d'entrées de table Sapio *et al.* (2017). Pour les paquets de données entrants, le commutateur analyse le message puis agrège les données selon les règles de flux définies par le contrôleur. De même, un autre *framework* est proposé dans Yang *et al.* (2019)

pour intégrer le module d'analyse de charge de paquets et le moteur de traitement d'agrégation de données dans le commutateur, améliorant ainsi le débit du réseau avec une latence réduite. Outre l'agrégation des données, le contrôleur contrôle la cohérence de l'état en maintenant une table d'enregistrement à jour sur les commutateurs du réseau du centre de données Liu *et al.* (2017). Comme la popularité de chaque paire clé-valeur mise en cache change au fil du temps, le contrôle central est utilisé pour conserver les éléments en cache sur les commutateurs Jin *et al.* (2017). En pratique, les opérateurs de réseau utilisent des dispositifs programmables pour modifier la fonctionnalité du plan de données selon les besoins tout en conservant la capacité d'interpréter et de traiter les flux de données à la vitesse du lien. Des cas d'utilisation tels que la mise en cache de données Jin *et al.* (2017), l'équilibrage de charge Grigoryan *et al.* (2019), le serveur de système de noms de domaine Woodruff *et al.* (2019) et l'agrégation de données Sapio *et al.* (2019); Yang *et al.* (2019) peuvent être implémentés dans un périphérique réseau, réduisant le trafic sur le réseau et améliorant par conséquent les performances globales.

Les tables *Match-Action* permettent aux opérateurs de réseau et aux chercheurs d'utiliser un ensemble d'étapes de pipeline contenant des tables de correspondance de largeur prédéfinie et de profondeur dynamique pour faire correspondre un champ particulier dans un paquet entrant sur le pipeline. Les plans de données programmables (*Programmable Data Plane* PDP) permettent aux périphériques réseau d'effectuer des opérations complexes sur des paquets permettant de modifier la façon dont les données sont traitées au niveau matériel.

En outre, les langages spécifiques au domaine créés pour le plan de données programmables, tels que *Protocol-Oblivious Forwarding* Song (2013) et P4 Bosshart *et al.* (2014), permettent de programmer le comportement du plan de données dans

une abstraction de niveau supérieur. Le langage spécifique au domaine fournit de nouvelles approches dans le plan de données programmables da Costa Cordeiro *et al.* (2017), aidant à surmonter encore plus "l'ossification" des réseaux informatiques et permettant le développement d'une nouvelle génération de services réseau.

Étant donné que P4 est la technologie la plus répandue et la plus utilisée pour définir des algorithmes de plan de données, la véritable puissance de P4 réside dans un pipeline de *Match-Action* programmable, décrivant des tables, des correspondances et des actions de manière abstraite et directe avec la liberté de définir n'importe quel type d'en-têtes de protocole.

D'autre part, P4 est un langage spécifique à un domaine qui ne prend pas en charge les opérations complexes telles que les unités de calcul de somme de contrôle ou de hachage, les générateurs de nombres aléatoires, les compteurs de paquets et d'octets, les registres et bien d'autres. Pour rendre ces fonctionnalités externes utilisables, P4 introduit ce qu'on appelle des *externs*. Ces *externs* sont les objets externes décrivant les interfaces que ces objets exposent au plan de données. Par ailleurs, la mise en œuvre d'opérations complexes a été étudiée en tant qu'extension de P4.

Dans da Silva *et al.* (2018), une étude de cas est réalisée sur l'intégration du schéma ROHC (*Robust Header Compression*), qui conduit à l'implémentation d'une nouvelle primitive native. Un autre travail, Scholz *et al.* (2019), propose l'extension de P4 par des fonctions de hachage cryptographiques nécessaires pour construire des applications et des protocoles sécurisés. Les auteurs de Scholz *et al.* (2019) ont proposé une extension de l'architecture de commutateur portable P4 et ont mis en œuvre une preuve de concept pour trois plates-formes : CPU, unité de traitement réseau (NPU) et cibles P4 basées sur FPGA. De plus, l'exécution asynchrone des *externs* a

| Article | Architecture | Objectifs de conception | Contribution |
|---|---|---|---|
| da Silva *et al.* (2018) | P4 | Prise en charge de la compression et de la décompression d'en-tête | Ajout de nouvelles primitives externes à P4 pour prendre en charge la compression et la décompression d'en-tête (ROHC) |
| Kim *et al.* (2020) | P4 | Exploiter l'informatique en réseau pour accélérer la charge de travail des applications scientifiques | NSinC (*Network-accelerated ScIeNtific Computing*) |
| Scholz *et al.* (2019) | P4 | Activation de l'authentification et de la résilience dans les cibles P4 | Extension de P4 avec une fonction de hachage cryptographique externe |

TABLEAU 3.1 – travaux déployant l'informatique en réseau en utilisant le langage de programmation P4

été étudiée dans Horpácsi *et al.* (2019); Laki *et al.* (2020), montrant que les paquets peuvent être traités par le pipeline lors de l'exécution des *externs*.

Le tableau 3.1 résume quelques articles dont l'architecture est basée sur P4 en soulignant leurs objectifs de conception et leurs contributions.

## 3.4 Provisionnement des infrastructures

Très peu de travaux se sont concentrés sur l'utilisation de l'informatique en réseau pour le provisionnement des infrastructures. À notre connaissance, ce travail est la toute première tentative d'exploiter l'informatique en réseau pour la fourniture de services de diffusion continue de vidéo 360°. Pourtant, il y a eu des travaux visant à répondre aux exigences de la vidéo 360° sans utiliser l'informatique en réseau. La plupart de ces travaux reposent sur l'informatique en périphérie.

### 3.4.1 Le provisionnement des infrastructures avec l'informatique en réseau

Dans Vaucher *et al.* (2020), les auteurs ont présenté *ZipLine*, une approche pour concevoir et implémenter la compression à haute vitesse en utilisant un réseau compatible avec P4. Ils ont évalué leur système en mesurant le débit, la latence et les mesures de compression dans des scénarios réels.

Dans Xu *et al.* (2020), les auteurs ont proposé une architecture qui tire parti de l'informatique en réseau pour obtenir de meilleures performances pour ce qui est du haut débit et de la faible latence dans les systèmes IoT industriels. Il s'agit d'une architecture de périphérie mobile alimentée par un réseau centré sur l'information (*Information-Centric Network* ICN). Les tâches contraintes sont déchargées sur les dispositifs ICN, alors que le reste va au *Multi-access Edge Computing* (MEC). Le réseau centré sur l'information découple les informations de leurs sources et fournit un stockage en réseau dans tous les routeurs. Les auteurs ont évalué la faisabilité de leur architecture en utilisant deux cas d'utilisation industriels, le mouvement robotique en réseau et la détection d'incendie en réseau. Cependant, ils n'ont pas discuté de la réduction de la consommation de trafic dans le réseau.

Les auteurs de Tagami *et al.* (2019) se sont concentrés sur la diffusion continue de vidéo panoramique en direct basée sur des tuiles en exploitant les ressources du réseau. Ils ont étudié l'avantage de déplacer les transcodeurs près des utilisateurs finaux. Ce travail a mesuré l'accès au cache, la taille des tuiles et le trafic. En même temps, il n'a pas pris en compte les défis liés aux vidéos 360°, comme la latence.

### 3.4.2 L'informatique en périphérie

L'informatique en périphérie fournit les capacités de l'informatique en nuage à la périphérie du réseau. Elle s'incarne dans trois paradigmes Mouradian *et al.* (2017) :

1. *Cloudlet*.
2. L'informatique en périphérie à accès multiple (MEC).
3. L'informatique géodistribuée (*Fog Computing*).

L'informatique en périphérie peut offrir une faible latence grâce à la communication réduite. Les auteurs de Dong *et al.* (2021) ont proposé une architecture de système flexible avec une abstraction à cinq couches pour les applications vidéo interactives. L'application vidéo interactive est un terme générique pour les vidéos dans les jeux infonuagiques, la réalité virtuelle (VR) et les vidéos 360°. Ces vidéos ont des exigences de latence strictes, qui nécessitent que le temps écoulé entre une interaction de l'utilisateur et la réaction du système soit limité. Un modèle de représentation efficace des tâches est conçu pour augmenter la flexibilité du système. Les performances du système sont analysées à l'aide d'un ensemble de paramètres de mesure de latence. Les résultats ont montré que la latence mesurée de bout en bout satisfait aux restrictions de la plupart des applications vidéo interactives fournies par un serveur périphérique. Ce travail ne traite pas l'efficacité de l'utilisation des ressources réseau à la périphérie et de la limitation de ces ressources.

## 3.5 Conclusion

Dans ce chapitre, nous avons présenté une revue de la littérature en ce qui concerne quelques technologies permettant le déploiement l'informatique en réseau, la pro-

grammabilité en réseau et le provisionnement des infrastructures. Dans le prochain chapitre, nous allons présenter notre étude sur le transcodage de la diffusion continue de vidéo 360° en utilisant le langage de programmation P4. Nous allons expliquer le cas d'utilisation de la diffusion continue de vidéo 360°, l'architecture proposée et la preuve de concept.

# CHAPITRE IV

# TRANSCODAGE DE LA DIFFUSION CONTINUE DE VIDÉO 360° EN UTILISANT LE LANGAGE P4

## 4.1 Introduction

Dans ce chapitre, nous présentons la contribution principale du mémoire. La première section décrit en détail le cas d'utilisation de diffusion continue de vidéo 360°. Ensuite, nous présentons une description globale des modules de l'architecture proposée et une description détaillée du service d'intégration de tranches et de ses interactions avec les autres composants de l'architecture. Après, nous décrivons la preuve de concept.

## 4.2 Cas d'utilisation : la diffusion continue de vidéo 360°

L'informatique en réseau est utilisé dans divers domaines pour répondre aux exigences de différents services, telles que l'agrégation de paquets, l'accélération de l'apprentissage automatique, la télémétrie réseau et le traitement de flux Vaucher *et al.* (2020). La figure 4.1 montre un cas d'utilisation illustratif du concept de l'informatique en réseau pour fournir des applications de la diffusion continue de vidéo 360°.

La diffusion continue de vidéo 360° a récemment attiré une attention et un intérêt croissants en tant que technologie immersive. La forte demande de bande passante et la consommation intensive de ressources de calcul rendent difficile la diffusion continue de vidéo 360° sur les appareils mobiles tout en obtenant une qualité perçue acceptable. La zone visible de la vidéo (connue sous le nom de fenêtre d'affichage de l'utilisateur) est affichée à travers un visiocasque avec une fréquence d'images très élevée et une haute résolution. Une agréable expérience de la diffusion continue de vidéo 360° peut éviter le cybermalaise, grâce à une connectivité ultra-réactive. La résolution de la vidéo immersive en réalité virtuelle (RV) dans le visiocasque doit être extrêmement proche de la quantité de détails que la rétine humaine peut percevoir, ce qui nécessite une bande passante ultra-élevée.

Le transcodage vidéo est un facteur critique dans la compression/décompression des médias et le suréchantillonnage/sous-échantillonnage des vidéos 360°. Bien que le calcul et le stockage centralisés basés sur l'infonuagique ne soient pas adaptés à ce type d'application contraint à la latence, la diffusion continue assistée par l'informatique en périphérie semble être une solution prometteuse pour atténuer le faible taux d'accès au cache qui augmente le volume de trafic. Pour faire face à ce problème, Tagami *et al.* (2019) a introduit le transcodage en périphérie implémenté à l'aide de l'informatique en réseau.

Les vidéos 360° sont transcodées sur les serveurs du fournisseur de flux vidéo pour créer plusieurs couches et tuiles de qualité. Le transcodage est nécessaire, car les vidéos 360° capturées par le diffuseur ne sont pas nécessairement encodées dans ce format. Les flux vidéo transcodés sont ensuite transmis au réseau de distribution de contenu en utilisant le protocole HTTP. La solution proposée dans Tagami *et al.* (2019) permet le partage de contenu en partitionnant chaque image de la vidéo et

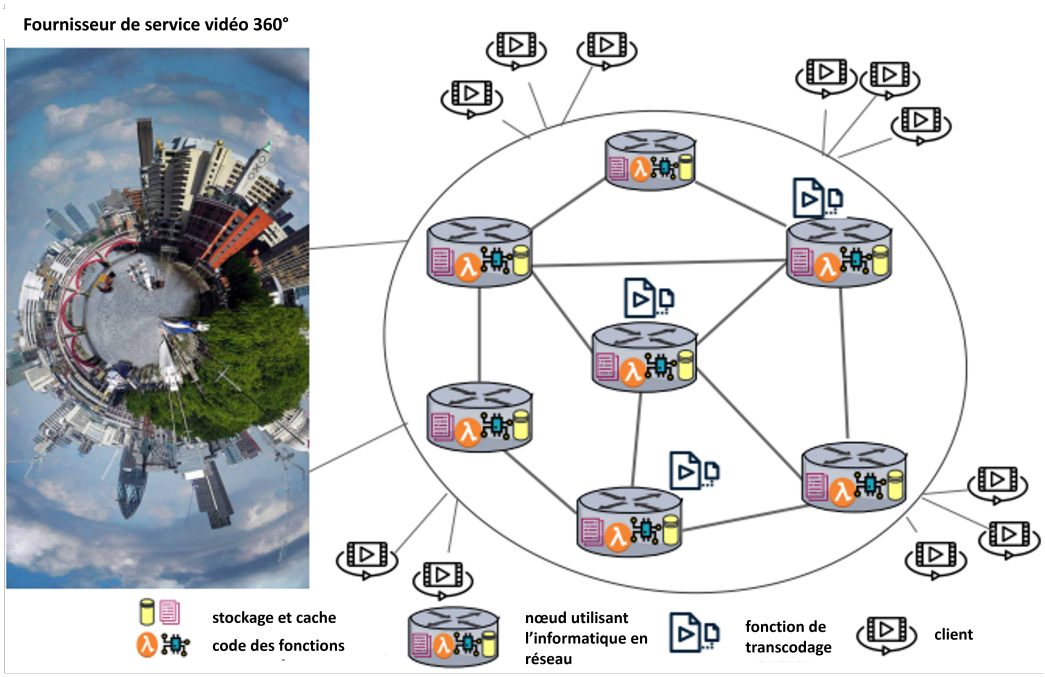



FIGURE 4.1 – la diffusion continue de vidéo 360° en utilisant l'informatique en réseau

en équilibrant la charge à l'aide du transcodage. Une réduction significative (20 à 30%) du volume de trafic du réseau central peut être obtenue en fonction de l'emplacement du transcodeur. Il existe d'autres exemples de l'informatique en réseau. Par exemple, Vaucher *et al.* (2020) ont proposé un (dé)compresseur de données en réseau fonctionnant à la vitesse de la ligne et déchargeant le traitement des nœuds d'extrémité adaptés aux applications à temps critique Vaucher *et al.* (2020). Dans un autre exemple, la diffusion vidéo panoramique en direct basée sur des tuiles se concentre sur l'exploitation des ressources du réseau en divisant une image vidéo en plusieurs tuiles. Ensuite, des transcodeurs proches des consommateurs sont utilisés pour diminuer l'utilisation de la bande passante dans le réseau Tagami *et al.* (2019).

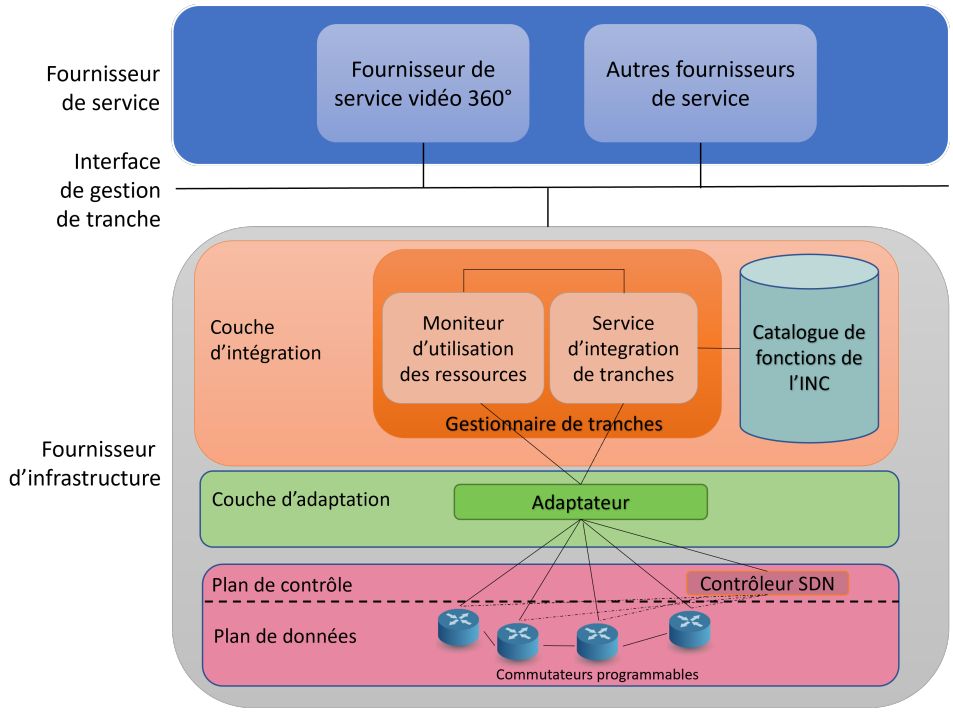



FIGURE 4.2 – architecture proposée

## 4.3 Architecture proposée

### 4.3.1 Modules de l'architecture

L'architecture proposée suppose que les fonctions qui permettent la transmission et les interactions avec la diffusion continue de vidéo 360° ont été développées à l'aide de langages tels que P4, compilées et prêtes à être exécutées dans des commutateurs programmables. L'architecture se concentre plutôt sur les modules nécessaires au provisionnement des tranches utilisant l'informatique en réseau qui permettront de réduire la latence et la charge du réseau.

Il existe deux acteurs clés, comme indiqué dans la Figure 4.2, le fournisseur de service

de diffusion continue de vidéo 360° et le fournisseur d'infrastructures. Ils interagissent par l'intermédiaire de l'interface de gestion des tranches qui permet des opérations telles que la création, la mise à jour et la suppression de tranches. Des normes sont en cours de définition pour ces interfaces et n'importe laquelle de ces normes émergentes pourrait être utilisée.

L'infrastructure comporte quatre couches et nous introduisons ces couches de bas en haut. La quatrième et dernière couche est le plan de données. Il est composé des commutateurs et nous supposons que tous les commutateurs sont programmables, c'est-à-dire capables d'héberger et d'exécuter des programmes. La troisième couche est le plan de contrôle avec le contrôleur du réseau défini par logiciel. La deuxième couche joue le rôle d'adaptateur. Nous envisageons les réseaux de nouvelle génération comme hétérogènes et le but de cette couche est d'atténuer cette hétérogénéité en offrant une interface uniforme à la couche d'intégration de tranches qui est la première couche.

La couche de l'intégration de tranches comporte deux composantes : le catalogue de programmes de l'informatique en réseau et le gestionnaire de tranches. Les programmes de l'informatique en réseau sont stockés dans le catalogue et le gestionnaire de tranches est constitué de deux composants : le moniteur d'utilisation des ressources et le service d'intégration de tranches. Il convient de noter que le catalogue de programmes de l'informatique en réseau s'ajoute au catalogue traditionnel de construction de tranches . Le moniteur d'utilisation des ressources collecte les indicateurs de performance afin de s'assurer que les exigences définies par les fournisseurs de service de diffusion continue de vidéo 360° sont satisfaites. Le service d'intégration de tranches effectue l'intégration proprement dite et l'activation de l'informatique en réseau. Son interaction avec les autres composants de l'architecture sera décrite dans

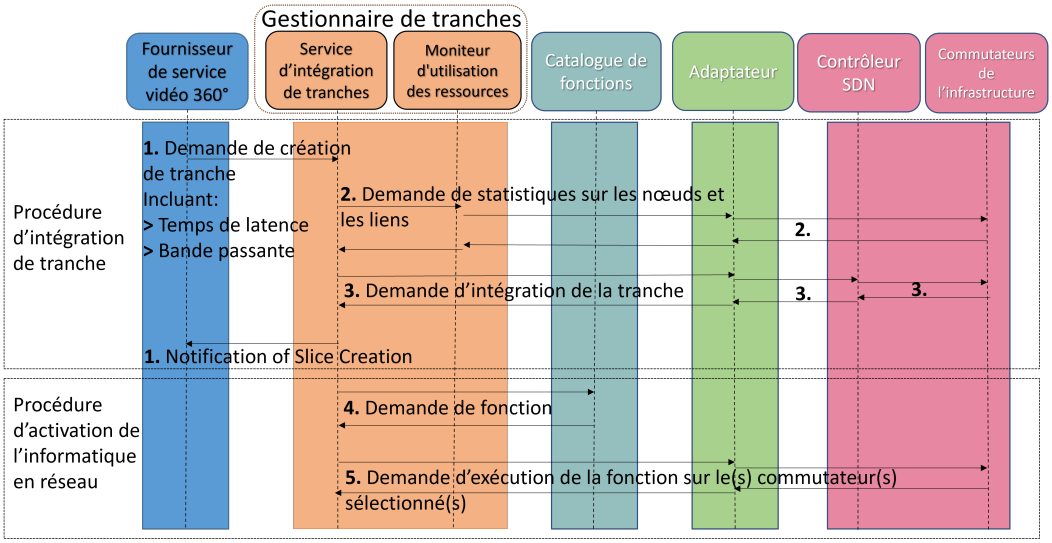



FIGURE 4.3 – diagramme de séquence

la section suivante.

### 4.3.2 Interactions entre les composants de l'architecture

Le service d'intégration de tranche intègre d'abord la tranche, puis active l'informatique en réseau dans cette dernière. Dans cette section, nous utilisons l'exemple d'une vidéo 360° pour illustrer comment le service d'intégration de tranches intègre une tranche demandée et lui rajoute l'informatique en réseau.

La figure 4.3 illustre en détail les interactions du service d'intégration de tranches. Tout d'abord, le fournisseur de service de diffusion continue de vidéo 360° envoie une demande de création d'une nouvelle tranche au service d'intégration de tranches, qui est illustrée dans la séquence 1. Cette demande comprend des exigences, telles que la bande passante et le temps de latence requis. Ensuite, le service d'intégration de tranches demande au moniteur d'utilisation des ressources d'obtenir des indicateurs

de performances, tels que les statistiques du processeur des nœuds du réseau et de la bande passante disponible de chaque lien, (séquence 2). Le moniteur d'utilisation des ressources envoie une demande à l'adaptateur pour collecter les indicateurs de performances directement à partir des commutateurs programmables ou à partir des contrôleurs SDN. L'adaptateur nous permet de mapper notre demande à différents contrôleurs SDN et commutateurs programmables pour couvrir des types hétérogènes d'infrastructures physiques. Ensuite, l'algorithme d'intégration décide d'intégrer la tranche en fonction de la disponibilité des ressources de l'infrastructure, séquence 3. Il existe plusieurs algorithmes existants dans la littérature pour l'intégration Su *et al.* (2019), et n'importe lequel d'entre eux pourrait être utilisé. Enfin, le service d'intégration de tranche envoie une notification de la création de la tranche au fournisseur de service de diffusion continue de vidéo 360°. Lorsque la procédure d'intégration de la tranche est terminée, la procédure d'activation de l'informatique en réseau démarre. Le service d'intégration de tranches sélectionne la fonction de l'informatique en réseau dans le catalogue de fonctions, tel qu'un transcodeur de vidéo, séquence 4. Ensuite, le service d'intégration de tranches exécute un autre algorithme pour décider du placement de la fonction de l'informatique en réseau dans les commutateurs de l'infrastructure, séquence 5. L'algorithme décide de placer la fonction dans un ou plusieurs nœuds de l'infrastructure de telle sorte à avoir un temps de latence ultra-faible et une bande passante élevée, tout en tenant compte des ressources limitées du réseau. Le problème est très brièvement discuté dans la référence Kunze *et al.* (arch), mais selon notre enquête, il n'existe pas un tel algorithme de placement dans la littérature. En outre, d'autres algorithmes de placement peuvent ne pas s'appliquer dans le domaine de l'informatique en réseau. Dans notre prototype, nous avons implémenté un algorithme très simple décrit plus loin.

## 4.4 Preuve de concept

Nous considérons la diffusion continue d'une vidéo 360° comme scénario pour la validation de notre architecture, en mettant l'accent sur la fonction de transcodage. Le transcodage peut être effectué à la réception, à la périphérie ou à l'intérieur du réseau avec le nouveau paradigme de l'informatique en réseau. Une preuve de concept est construite pour démontrer la faisabilité de notre architecture. Des simulations approfondies sont conduites pour évaluer les performances dans le chapitre 5.

Notre point de départ est l'implémentation du transcodage sous forme de fonction capable de s'exécuter à l'aide de l'informatique en réseau. La preuve de concept implémente des versions simplifiées de tous les modules architecturaux décrits dans les sections précédentes. De plus, nous avons effectué des mesures concrètes sur le prototype en marche. Étant donné, qu'aucune implémentation de transcodage avec l'informatique en réseau n'existe jusqu'aujourd'hui (à notre connaissance), nous avons décidé d'implémenter le transcodage dans l'émulateur de commutateur programmable, ce qui nous a donné accès au transcodage depuis le programme écrit en langage P4. On rappelle que P4 est un langage de traitement de paquets indépendant des protocoles de communication, il permet la programmabilité du plan de données. Son modèle abstrait de traitement de paquets est composé d'une machine d'état avec un analyseur, la machine change d'état avec des transitions pilotées par les valeurs dans les champs de l'en-tête du paquet, le traitement se poursuit ensuite en appliquant un ensemble de tables d'action de correspondance dans les pipelines entrants et sortants du commutateur da Silva *et al.* (2018).

Le transcodage est une opération complexe et n'est pas pris en charge nativement par le langage P4. Il existe deux approches pour ajouter des extensions au langage P4 :

ajouter de nouvelles primitives ou utiliser des instances externes da Silva *et al.* (2018). Nous avons décidé de suivre l'approche d'utilisation d'instances externes, car, selon la référence da Silva *et al.* (2018), cela conduit à un code plus élégant implémenté en dehors du cœur du commutateur, réduisant ainsi les éventuels effets secondaires. Un transcodeur logiciel libre en langage de programmation C est utilisé et est appelé depuis un programme P4. Le transcodeur change la résolution du contenu d'entrée sans changer le format du contenu.

La mise en œuvre de la couche inférieure de l'architecture repose sur les commutateurs P4 BMV2 bmv (2022) en tant que plan de données contrôlé avec un simple contrôleur SDN agissant comme un plan de contrôle. Mininet est utilisé comme émulateur. Vu que le contrôleur ne dispose pas d'installations pour collecter des statistiques sur l'utilisation des ressources, un module simple (moniteur d'utilisation des ressources) qui interagit indirectement avec le plan de données est développé pour collecter des statistiques sur l'utilisation des ressources. Un autre module nommé service d'intégration de tranches est développé pour offrir les fonctions très basiques du découpage virtuel du réseau, puisque le contrôleur n'offre pas de fonctionnalité de découpage virtuel du réseau. Nous avons implémenté l'isolation entre différentes tranches par le champ Ethertype de l'en-tête Ethernet, ce qui nous permet de marquer les paquets pour chaque tranche.

Dans la couche d'adaptation, l'adaptateur utilise une API RESTFul à la couche d'intégration, il reçoit des appels REST de la couche d'intégration et mappe ces appels sur les appels de fonction Python proposés par la couche de contrôle (c'est-à-dire le moniteur d'utilisation des ressources, le service d'intégration de tranches et le contrôleur). Le gestionnaire de tranches vérifie si les exigences de tranche peuvent être satisfaites par le réseau physique, il appelle donc une fonction pour apporter

des statistiques en temps réel à partir des commutateurs réseau en envoyant des requêtes à la couche d'adaptateur à travers l'API RESTFul. S'il y a suffisamment de ressources, le gestionnaire de tranches envoie une demande pour déployer la tranche.

À titre expérimental, nous avons créé deux tranches, l'une est compatible avec l'informatique en réseau et l'autre non. Dans la tranche compatible avec l'informatique en réseau, nous utilisons le transcodeur sur les commutateurs programmables, et dans la deuxième tranche, nous avons le programme de transcodage dans les hôtes. Nous avons mesuré le temps de création d'une tranche qui est défini comme étant le temps qui s'écoule à partir du moment où le fournisseur de service vidéo 360° demande la création de la tranche jusqu'à ce que la tranche soit déployée et disponible pour utilisation. Ce temps est de 356,91 ms pour la tranche qui n'utilise pas l'informatique en réseau et de 435,41 ms pour celle qui l'utilise. Ces chiffres sont raisonnables, car les fournisseurs demandent généralement la création de tranches avant les premières demandes des utilisateurs finaux. Nous avons également mesuré le temps de latence moyen de bout en bout et la charge moyenne du réseau des deux tranches créées. Le temps de latence moyen de bout en bout est calculé comme étant le temps de latence moyen de tous les paquets transmis pendant le flux de diffusion continue de vidéo 360°. La charge moyenne du réseau est calculée comme étant la bande passante moyenne utilisée par tous les liens divisée par la bande passante moyenne de tous les liens. Ces mesures sont fournies en exécutant les scénarios 10 fois.

Notre enquête démontre que la tranche compatible avec l'informatique en réseau surpasse la tranche sans l'informatique en réseau en termes de réduction de la charge du réseau par 27,67 %. L'utilisation d'un transcodeur le long du chemin réduit la charge de trafic par rapport au simple envoi des données brutes à travers le réseau. Cette réduction de trafic se traduit par l'obtention d'un gain de bande passante qui

est l'une des exigences de la communication pour le scénario de diffusion continue de vidéo 360°. De plus, nos résultats montrent une réduction de 15,93 % de la latence moyenne des paquets, ce qui est une autre exigence de communication pour la diffusion continue de vidéo 360°. En effet, la tranche compatible avec l'informatique en réseau réduit la taille de la vidéo 360° en utilisant le transcodeur sur les commutateurs.

## 4.5 Conclusion

Dans ce chapitre, nous avons présenté notre architecture de provisionnement de tranches compatible avec l'informatique en réseau pour la diffusion continue de vidéo 360°. Nous avons détaillé cette architecture et avons présenté une preuve de concept qui a démontré sa faisabilité dans le monde réel. Dans le prochain chapitre, nous évaluons les performances de l'architecture proposée et discutons les résultats obtenus.

# CHAPITRE V

# ÉVALUATION DE PERFORMANCES

## 5.1 Introduction

Le présent chapitre évalue les performances de la solution proposée. Nous commençons par décrire l'environnement, la topologie et les paramètres de simulation ainsi que les quatre différents scénarios utilisés dans l'évaluation. Nous illustrons par la suite les résultats obtenus à travers le calcul de la latence, la gigue et la charge du réseau et nous fournissons une analyse de tous ces résultats.

## 5.2 Environnement et scénarios de simulation

Pour la simulation, nous avons utilisé l'émulateur Mininet pour mettre en place une infrastructure SDN compatible avec le langage P4, qui est la couche de ressources physiques de l'architecture proposée. Mininet fournit un émulateur pour créer des infrastructures SDN programmables prenant en charge un trafic utilisateur réaliste à grande échelle. Les simulations s'exécutent sur une machine avec un double processeur Intel Xeon 2X8-Core 2,50 GHz E5-2450v2 et 40 Go de mémoire.

La topologie de simulation est illustrée à la figure 5.1, la topologie contient 11 com-

| Paramètre | Valeur |
|---|---|
| Nombre de commutateurs | 11 |
| Nombre de hôtes | 5 |
| Nombre de fournisseurs de service vidéo 360° | 1 |
| Nombre de serveurs périphériques | 1 |
| Bande passante des liens | 12 Mbps |
| Nombre d'images de la vidéo 360° | 3600 images |

TABLEAU 5.1 – paramètres de simulation

mutateurs, un fournisseur de service vidéo 360°, un serveur en périphérie et plusieurs hôtes finaux.

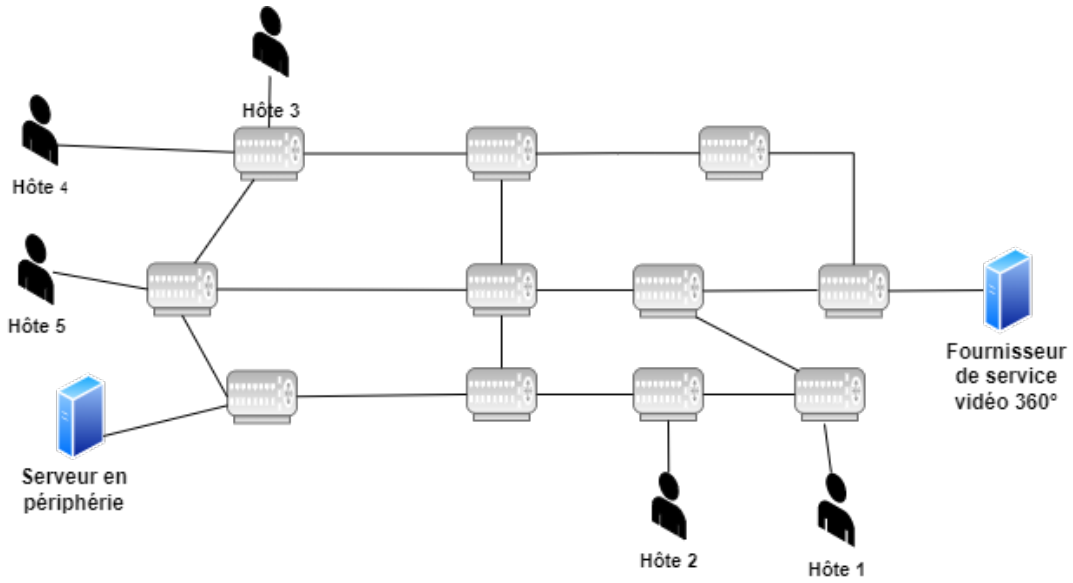



FIGURE 5.1 – topologie de simulation

Tous les commutateurs du réseau sont compatibles avec P4, la capacité CPU des commutateurs, des hôtes et du serveur périphérique est la même, les liens ont une bande passante de 12 Mbps. Une vidéo 360° contenant 3600 images est diffusée depuis le serveur. Le tableau 5.1 présente tous les paramètres de simulation.

Dans notre simulation, nous nous intéressons uniquement au transcodage, car c'est une fonction très importante dans la diffusion de vidéo 360° qui permet de répondre à toutes les exigences des clients qui ont des spécifications de connexion réseau différentes.

L'exécution de la fonction de transcodage sera déléguée soit au serveur périphérique (figure 5.2 (a)), soit à l'hôte final (figure 5.2 (b)) ou aux périphériques réseau (figure 5.2 (c-d)).

**Transcodage dans le serveur périphérique**

Le fournisseur de service vidéo 360° diffuse en continu une vidéo 360°. Afin de s'adapter au besoin de chaque utilisateur, la vidéo va être transcodée au niveau du serveur périphérique, le chemin de la diffusion entre le fournisseur et le serveur suit le chemin le plus court. Par la suite, quand le serveur termine le transcodage, il va envoyer le flux vidéo vers chaque hôte final en suivant le chemin le plus court.

**Transcodage dans les hôtes finaux**

Le fournisseur de service vidéo 360° diffuse en continu une vidéo 360°. Afin de s'adapter au besoin de chaque utilisateur, la vidéo va être transcodée au niveau des hôtes finaux, le chemin de la diffusion entre le fournisseur et chaque hôte suit le chemin le plus court.

**Transcodage dans l'équipement réseau proche du fournisseur de service**

Le fournisseur de service vidéo 360° diffuse en continu une vidéo 360°. Afin de s'adapter au besoin de chaque utilisateur, la vidéo va être transcodée au niveau de l'équipement réseau proche du fournisseur de service. Par la suite, quand l'équipement

réseau termine le transcodage, il va envoyer le flux vidéo vers chaque hôte final en suivant le chemin le plus court.

**Transcodage dans les équipements réseau proches des hôtes**

Le fournisseur de service vidéo 360° diffuse en continu une vidéo 360°. Afin de s'adapter au besoin de chaque utilisateur, la vidéo va être transcodée au niveau des équipements réseau proches des hôtes, le chemin de la diffusion entre le fournisseur et les équipements réseau responsables du transcodage suit le chemin le plus court. Par la suite, après avoir terminé le transcodage, ils vont envoyer le flux vidéo vers chaque hôte final.

Nous considérons et mesurons les métriques suivantes :

— La latence moyenne.

— La gigue.

— L'utilisation de la bande passante.

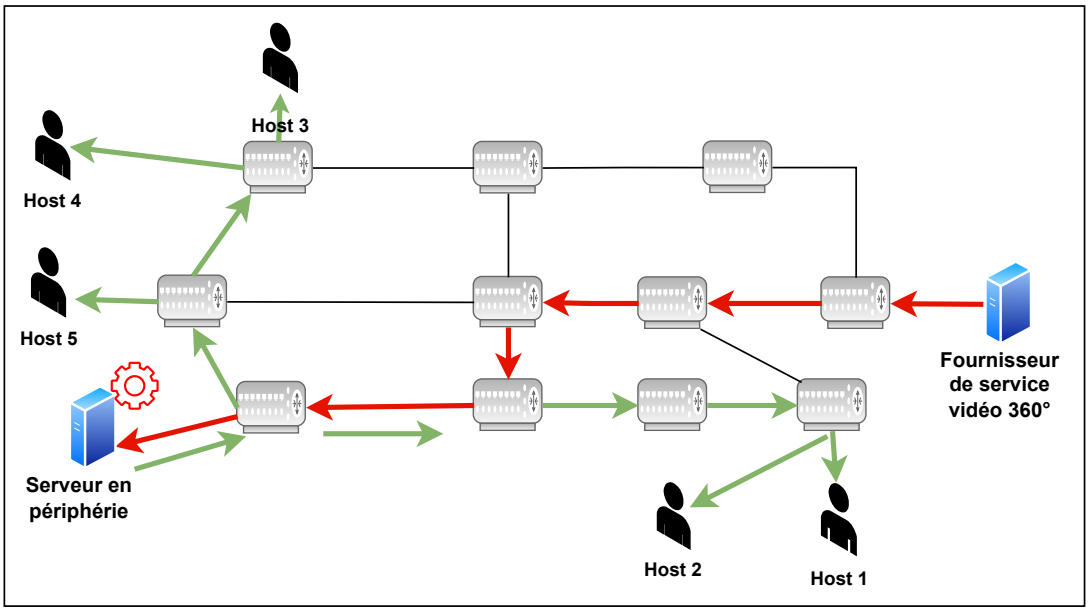
Qui sont les paramètres de QoS les plus fondamentaux pour la diffusion continue de vidéo 360°. Nous exécutons dix fois chaque simulation pour garantir la précision des résultats.

## 5.3 Résultats et discussions

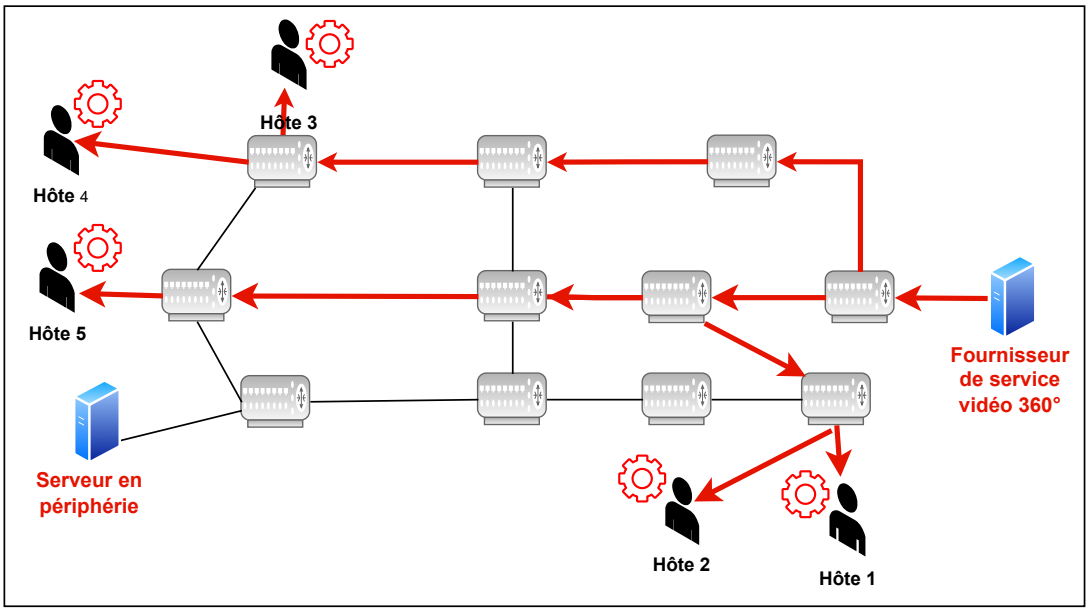
**Analyse de la latence**

Nous analysons la latence tout en plaçant le transcodeur sur le serveur périphérique, sur l'hôte final et sur les périphériques réseau programmables. Le placement sur les commutateurs P4 est effectué en fonction de l'emplacement du fournisseur de service

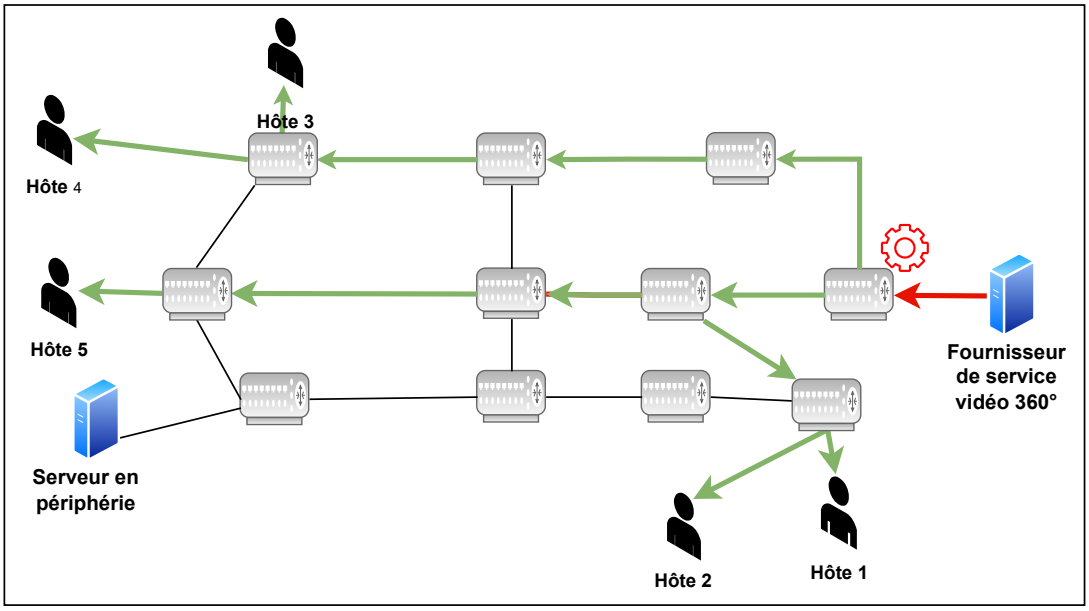
et des hôtes finaux. Nous considérons le nombre de sauts entre les deux comme la



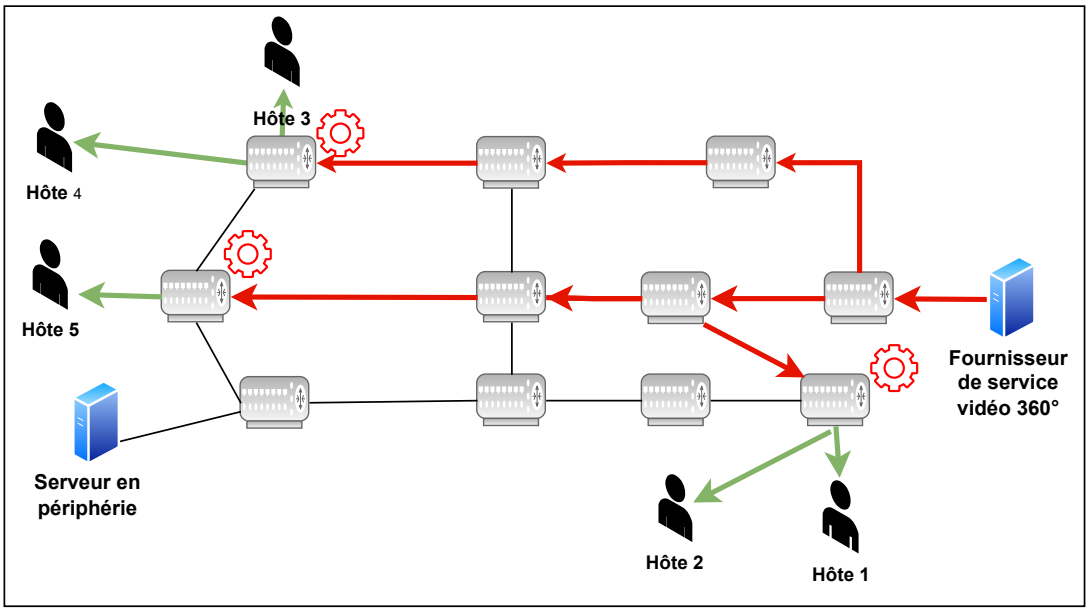

(a) Transcodage dans le serveur périphérique




(b) Transcodage dans les hôtes finaux




(c) Transcodage dans l'équipement réseau proche du fournisseur de service




(d) Transcodage dans les équipements réseau proches des hôtes


FIGURE 5.2 – scénarios de simulation

métrique de la distance.

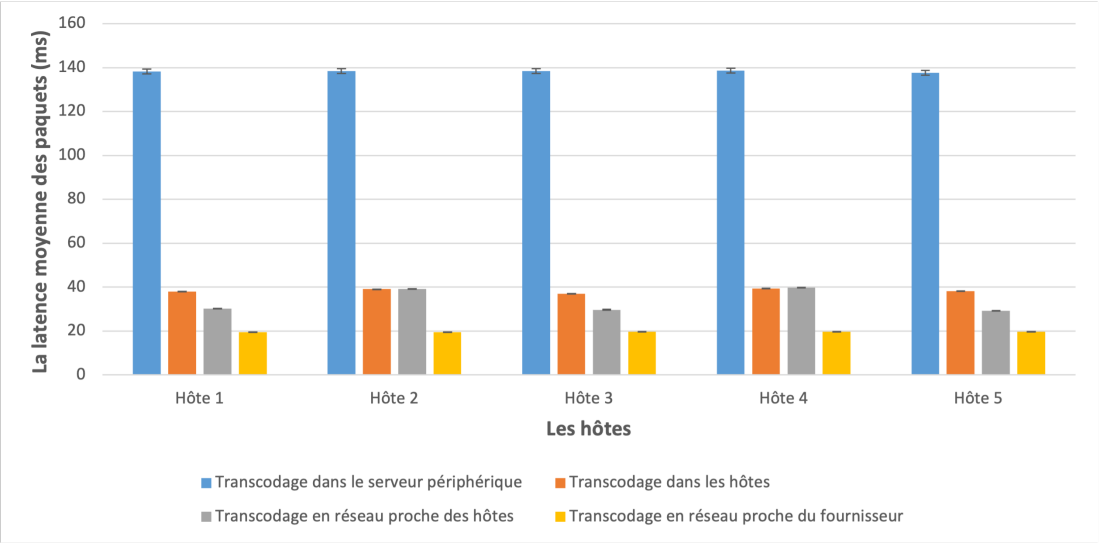



FIGURE 5.3 – résultats de la latence moyenne des paquets pour chaque hôte

Comme on peut le voir sur la figure 5.3, le transcodage sur le serveur de périphérie montre un mauvais résultat au niveau du temps de latence pour tous les hôtes, car les données brutes de la vidéo 360° sont transmises tout au long du réseau pour être transcodées dans le serveur périphérique, ce qui génère un délai plus long, puis les données transcodées de la vidéo 360° sont envoyées aux hôtes finaux. De plus, le transcodage sur l'hôte final réduit considérablement la latence par rapport à la périphérie et cela est dû au fait que nous avons gagné du temps qui a été perdu à attendre les réponses de la périphérie. L'hôte 2 et l'hôte 4 affichent des résultats similaires pour les scénarios où on transcode dans les hôtes finaux et celui où on transcode dans les commutateurs réseau près des hôtes finaux. En particulier, placer le transcodeur près de l'expéditeur (c'est-à-dire le fournisseur de service de diffusion continue de vidéo 360°) génère les meilleurs résultats en réduisant la latence et cela est principalement dû à la taille réduite de la vidéo 360°.

Dans l'ensemble, l'informatique en réseau surpasse l'informatique en périphérie en réduisant considérablement la latence, en transcodant le long du chemin vers les hôtes finaux, et le meilleur placement est celui où on transcode à proximité du fournisseur de service de diffusion continue de vidéo 360°.

### Analyse de la gigue

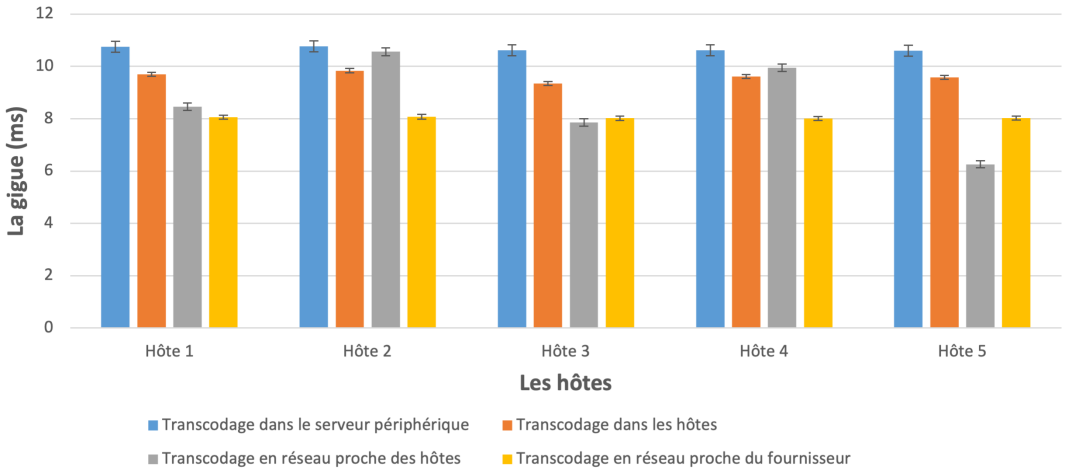



FIGURE 5.4 – résultats de la gigue pour chaque hôte

Une autre exigence importante pour la diffusion continue de vidéos 360° est la gigue. Idéalement, nous devons maintenir les valeurs de gigue en dessous de 15 ms [1] pour éviter le cybermalaise et offrir une expérience agréable. La figure 5.4 décrit la gigue pour différents scénarios dans lesquels il est clairement montré que la contrainte est satisfaite dans tous les scénarios. Concernant la différence des valeurs de gigue d'un hôte à l'autre, cela s'explique par le fait que le nombre de sauts entre le fournisseur de service de diffusion continue de vidéo 360° et chaque hôte diffère comme illustré

1. ITU-T J.1631 (11/2021)

à la figure 5.2.

**Analyse de la bande passante**

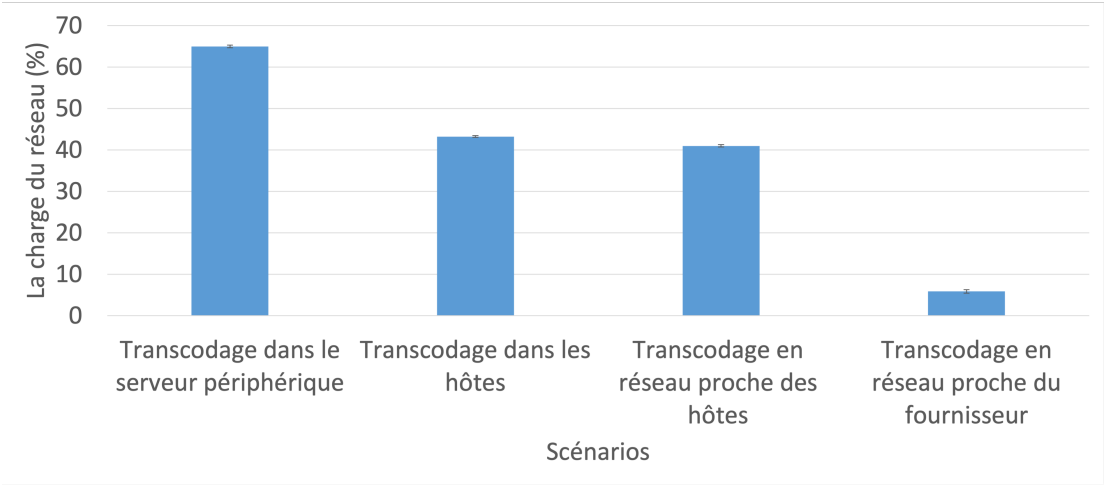



FIGURE 5.5 – résultats de la charge du réseau par scénario

La transmission de données de diffusion continue de vidéo 360° dans leur format brut peut entraîner une congestion importante du réseau, comme le montre la figure 5.5 dans les scénarios de transcodage de la vidéo 360° au niveau du serveur périphérique et des hôtes finaux. Ainsi, le scénario utilisant l'informatique en réseau surpasse les scénarios utilisant l'informatique en périphérie et celui où on fait le traitement au niveau des hôtes en réduisant la charge du réseau. En effet, transcoder la vidéo 360° le long du chemin et la transmettre dans une taille plus petite réduit la charge du réseau tout en conservant une qualité élevée. Il est essentiel de souligner que le transcodage de la vidéo 360° près de l'endroit où ses données sont générées offre les meilleurs résultats en réduisant la charge de trafic de plus de 50 % par rapport au transcodage sur un serveur périphérique, ce qui par conséquent maximise le gain de la bande passante.

## 5.4 Conclusion

Ce chapitre a décrit l'environnement et les scénarios de simulation utilisés pour évaluer les performances de l'architecture proposée. Il a également présenté les résultats avec leurs interprétations. Les résultats ont montré que l'informatique en réseau réduit de manière importante la latence ainsi que la charge du réseau comparée à l'informatique en périphérie. Le chapitre suivant conclut ce mémoire.

# CHAPITRE VI

# CONCLUSION

Ce mémoire propose une architecture pour le provisionnement des tranches compatibles avec l'informatique en réseau pour les applications de diffusion continue de vidéo 360° dans les réseaux de prochaine génération. Il répond à deux défis clés : la réduction de la latence et la réduction de la charge du réseau. Les fondamentaux des applications de diffusion continue de vidéo 360°, l'informatique en réseau et le découpage virtuel du réseau sont présentés. L'état de l'art est évalué. L'informatique en périphérie est le paradigme qui a été le plus largement utilisé jusqu'à présent pour résoudre des problèmes. En effet, cela a aidé à résoudre le défi de la latence. Cependant, comme le montre notre validation, notre architecture proposée conduit à de meilleurs résultats. De plus, elle fournit également de meilleurs résultats en matière de réduction de la charge du réseau.

L'architecture proposée comporte 4 couches : la couche de plan de données, la couche de contrôle, la couche d'adaptation et la couche d'intégration de tranches. Son composant clé, le service d'intégration de tranches, est présenté en détail. Une preuve de concept a été construite pour démontrer la faisabilité de l'architecture.

De plus, des simulations approfondies ont été exécutées pour évaluer les perfor-

mances. Des versions simplifiées de tous les modules logiciels, y compris le service d'intégration de tranches, ont été développées. Le transcodeur qui s'exécute dans les équipements réseau a également été développé en tant qu'une fonction externe de BMV2. Des mesures concrètes telles que le temps nécessaire pour créer des tranches ont également été effectuées sur le prototype.

Plusieurs scénarios ont été envisagés pour les simulations (par exemple, un transcodeur fonctionnant à la périphérie ou un transcodeur fonctionnant dans le réseau). Ils montrent tous que notre architecture proposée surpasse les approches basées sur la périphérie, en ce qui concerne la latence et la réduction de la charge du réseau.

Il existe plusieurs perspectives de recherche intéressantes que nous aimerions poursuivre à l'avenir. Le premier est le problème de placement d'un programme de l'informatique en réseau. En plus de la fonction de transcodage qui est implémentée en tant que programme de l'informatique en réseau dans cet article, il existe plusieurs autres fonctions qui pourraient également être implémentées en tant que programme de l'informatique en réseau (par exemple, codage, décodage, *rendering*).

En supposant que toutes ces fonctions sont disponibles dans le catalogue de fonctions, le problème général est de savoir laquelle d'entre elles exécuter dans le réseau pour répondre à des objectifs spécifiques, en particulier lorsque des réseaux hétérogènes avec des commutateurs, des cartes réseau intelligentes et des accélérateurs matériels sont pris en compte.

Comme indiqué dans le document, la gestion actuelle du cycle de vie des tranches se fait en 5G avec MANO. Cependant, le MANO actuel ne peut pas gérer les tranches qui sont compatibles avec l'informatique en réseau. L'architecture MANO doit donc évoluer afin de pouvoir gérer de manière transparente des infrastructures hétérogènes

avec des tranches compatibles avec l'informatique en réseau et des tranches qui ne le sont pas.

Une troisième et dernière direction consiste à rechercher des architectures plus générales qui intègrent non seulement l'informatique en réseau, comme cela est fait dans cet article, mais également d'autres paradigmes tels que l'informatique en périphérie.

# RÉFÉRENCES

(2022). Designing your own switch target with bmv2. `http://bmv2.org`. Consulté le : 2022-08-14.

Albalawi, A. A., Chakraborti, A., Westphal, C., Kutscher, D., He, J. et Hoole, Q. (2019). Inca : An architecture for in-network computing. Dans *Proceedings of the 1st ACM CoNEXT Workshop on Emerging In-Network Computing Paradigms*, ENCP '19, p. 56–62., New York, NY, USA. Association for Computing Machinery.

Amadeo, M., Campolo, C., Ruggeri, G., Molinaro, A. et Iera, A. (2019). Sdn-managed provisioning of named computing services in edge infrastructures. *IEEE Transactions on Network and Service Management*, *16*(4), 1464–1478. `http://dx.doi.org/10.1109/TNSM.2019.2945497`

Barik, R. K., Lenka, R. K., Rao, K. R. et Ghose, D. (2016). Performance analysis of virtual machines and containers in cloud computing. Dans *2016 international conference on computing, communication and automation (iccca)*, 1204–1210. IEEE.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G. *et al.* (2014). P4 : Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, *44*(3), 87–95.

da Costa Cordeiro, W. L., Marques, J. A. et Gaspary, L. P. (2017). Data plane programmability beyond openflow : Opportunities and challenges for network and service operations and management. *Journal of Network and Systems Management*, *25*(4), 784–818.

da Silva, J. S., Boyer, F.-R., Chiquette, L.-O. et Langlois, J. P. (2018). Extern objects in p4 : An rohc header compression scheme case study. Dans *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 517–522. IEEE.

Dang, H. T., Bressana, P., Wang, H., Lee, K. S., Zilberman, N., Weatherspoon, H., Canini, M., Pedone, F. et Soulé, R. (2020). P4xos : Consensus as a network service. *IEEE/ACM Transactions on Networking*, *28*(4), 1726–1738.

Dang, H. T., Sciascia, D., Canini, M., Pedone, F. et Soulé, R. (2015). Netpaxos : Consensus at network speed. Dans *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, 1–7.

Dong, Y., Song, L., Xie, R. et Zhang, W. (2021). An elastic system architecture for edge based low latency interactive video applications. *IEEE Transactions on Broadcasting*, *67*(4), 824–836.

ETSI, G. N.-M. . v. (2014). *Network Functions Virtualisation(NFV) ; Management and Orchestration*. Rapport technique.

Firestone, D., Putnam, A., Mundkur, S., Chiou, D., Dabagh, A., Andrewartha, M., Angepat, H., Bhanu, V., Caulfield, A., Chung, E. *et al.* (2018). Azure accelerated networking :{SmartNICs} in the public cloud. Dans *15th USENIX*

*Symposium on Networked Systems Design and Implementation (NSDI 18)*, 51–66.

Gao, Y. et Wang, Z. (2021). A review of p4 programmable data planes for network security. *Mobile Information Systems*, *2021*.

Grigoryan, G., Liu, Y. et Kwon, M. (2019). Iload : In-network load balancing with programmable data plane. Dans *Proceedings of the 15th International Conference on emerging Networking EXperiments and Technologies*, 17–19.

Hakiri, A., Gokhale, A., Berthou, P., Schmidt, D. C. et Gayraud, T. (2014). Software-defined networking : Challenges and research opportunities for future internet. *Computer Networks*, *75*, 453–471.

Handley, M., Raiciu, C., Agache, A., Voinescu, A., Moore, A. W., Antichi, G. et Wójcik, M. (2017). Re-architecting datacenter networks and stacks for low latency and high performance. Dans *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 29–42.

Hassas Yeganeh, S. et Ganjali, Y. (2012). Kandoo : a framework for efficient and scalable offloading of control applications. Dans *Proceedings of the first workshop on Hot topics in software defined networks*, 19–24.

Horpácsi, D., Laki, S., Vörös, P., Tejfel, M., Pongrácz, G. et Molnár, L. (2019). Asynchronous extern functions in programmable software data planes. Dans *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 1–2. IEEE.

Hu, N., Tian, Z., Du, X. et Guizani, M. (2021). An energy-efficient in-network

computing paradigm for 6g. *IEEE Transactions on Green Communications and Networking*, *5*(4), 1722–1733.

Ibanez, S., Brebner, G., McKeown, N. et Zilberman, N. (2019). The p4-> netfpga workflow for line-rate packet processing. Dans *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 1–9.

Jepsen, T., Moshref, M., Carzaniga, A., Foster, N. et Soulé, R. (2018). Life in the fast lane : A line-rate linear road. Dans *Proceedings of the Symposium on SDN Research*, 1–7.

Jin, X., Li, X., Zhang, H., Soulé, R., Lee, J., Foster, N., Kim, C. et Stoica, I. (2017). Netcache : Balancing key-value stores with fast in-network caching. Dans *Proceedings of the 26th Symposium on Operating Systems Principles*, 121–136.

Kaur, S., Kumar, K. et Aggarwal, N. (2021). A review on p4-programmable data planes : Architecture, research efforts, and future directions. *Computer Communications*, *170*, 109–129.

Kazmi, S. A., Khan, L. U., Tran, N. H. et Hong, C. S. (2019). *Network slicing for 5G and beyond networks*, volume 1. Springer.

Kecskemeti, G., Marosi, A. C. et Kertesz, A. (2016). The entice approach to decompose monolithic services into microservices. Dans *2016 International Conference on High Performance Computing & Simulation (HPCS)*, 591–596. IEEE.

Kfoury, E. F., Crichigno, J. et Bou-Harb, E. (2021). An exhaustive survey on p4 programmable data plane switches : Taxonomy, applications, challenges, and future trends. *IEEE Access*, *9*, 87094–87155.

Kim, C., Sivaraman, A., Katta, N., Bas, A., Dixit, A. et Wobker, L. J. (2015). In-band network telemetry via programmable dataplanes. Dans *ACM SIGCOMM*, volume 15.

Kim, D., Jain, A., Liu, Z., Amvrosiadis, G., Hazen, D., Settlemyer, B. et Sekar, V. (2020). Unleashing in-network computing on scientific workloads. *arXiv preprint arXiv :2009.02457*.

Kirkpatrick, K. (2013). Software-defined networking. *Communications of the ACM*, *56*(9), 16–19.

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T. *et al.* (2010). Onix : A distributed control platform for large-scale production networks. Dans *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*.

Kunze, I., Wehrle, K., Trossen, D. et Montpetit, M. (2021). Use cases for in-network computing. *Internet Engineering Task Force, Internet-Draft draft-irtf-coinrg-use-cases-00*.

Kunze, I., Wehrle, K., Trossen, D. et Montpetit, M.-J. (2022, March). Use cases for in-network computing. *IETF, Internet-Draft*.

Laki, S., Horpácsi, D., Voros, P., Tejfel, M., Hudoba, P., Pongracz, G. et Molnar, L. (2020). The price for asynchronous execution of extern functions in pro-

grammable software data planes. Dans *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 23–28. IEEE.

Li, X., Xie, R., Yu, F. R., Huang, T. et Liu, Y. (2021). Advancing software-defined service-centric networking toward in-network intelligence. *IEEE Network*, *35*(5), 210–218.

Liu, M., Luo, L., Nelson, J., Ceze, L., Krishnamurthy, A. et Atreya, K. (2017). Incbricks : Toward in-network computation with an in-network cache. Dans *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 795–809.

Liu, Z., Manousis, A., Vorsanger, G., Sekar, V. et Braverman, V. (2016). One sketch to rule them all : Rethinking network flow monitoring with univmon. Dans *Proceedings of the 2016 ACM SIGCOMM Conference*, 101–114.

Mai, L., Rupprecht, L., Alim, A., Costa, P., Migliavacca, M., Pietzuch, P. et Wolf, A. L. (2014). Netagg : Using middleboxes for application-specific on-path aggregation in data centres. Dans *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 249–262.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S. et Turner, J. (2008). Openflow : enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, *38*(2), 69–74.

Miao, R., Zeng, H., Kim, C., Lee, J. et Yu, M. (2017). Silkroad : Making stateful layer-4 load balancing fast and cheap using switching asics. Dans *Proceedings of*

*the Conference of the ACM Special Interest Group on Data Communication*, 15–28.

Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R. H., Morrow, M. J. et Polakos, P. A. (2017). A comprehensive survey on fog computing : State-of-the-art and research challenges. *IEEE communications surveys & tutorials*, *20*(1), 416–464.

Narayana, S., Sivaraman, A., Nathan, V., Goyal, P., Arun, V., Alizadeh, M., Jeyakumar, V. et Kim, C. (2017). Language-directed hardware design for network performance monitoring. Dans *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 85–98.

Ojo, M., Adami, D. et Giordano, S. (2016). A sdn-iot architecture with nfv implementation. Dans *2016 IEEE Globecom Workshops (GC Wkshps)*, 1–6. IEEE.

ONF (2022a). Open networking foundation. https ://opennetworking.org/. Consulté le : 2022-08-14.

ONF (2022b). P4 – language consortium. https ://p4.org/. Consulté le : 2022-08-14.

Orlosky, J., Kiyokawa, K. et Takemura, H. (2017). Virtual and augmented reality on the 5g highway. *Journal of Information Processing*, *25*, 133–141.

Pongrácz, G., Molnár, L., Kis, Z. L. et Turányi, Z. (2013). Cheap silicon : A myth or reality ? picking the right data plane hardware for software defined networking. Dans *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 103–108.

Ports, D. R. et Nelson, J. (2019). When should the network be the computer ? Dans *Proceedings of the Workshop on Hot Topics in Operating Systems*, 209–215.

Rüth, J., Glebke, R., Wehrle, K., Causevic, V. et Hirche, S. (2018). Towards in-network industrial feedback control. Dans *Proceedings of the 2018 Morning Workshop on In-Network Computing*, 14–19.

Sapio, A., Abdelaziz, I., Aldilaijan, A., Canini, M. et Kalnis, P. (2017). In-network computation is a dumb idea whose time has come. Dans *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, 150–156.

Sapio, A., Canini, M., Ho, C.-Y., Nelson, J., Kalnis, P., Kim, C., Krishnamurthy, A., Moshref, M., Ports, D. R. et Richtárik, P. (2019). Scaling distributed machine learning with in-network aggregation. *arXiv preprint arXiv :1903.06701*.

Scholz, D., Oeldemann, A., Geyer, F., Gallenmüller, S., Stubbe, H., Wild, T., Herkersdorf, A. et Carle, G. (2019). Cryptographic hashing in p4 data planes. Dans *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 1–6. IEEE.

Song, H. (2013). Protocol-oblivious forwarding : Unleash the power of sdn through a future-proof forwarding plane. Dans *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 127–132.

Stallings, W. (2015). *Foundations of modern networking : SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional.

Su, R., Zhang, D., Venkatesan, R., Gong, Z., Li, C., Ding, F., Jiang, F. et Zhu, Z. (2019). Resource allocation for network slicing in 5g telecommunication networks : A survey of principles and models. *IEEE Network*, *33*(6), 172–179.

Sun, Y., Nanda, S. et Jaeger, T. (2015). Security-as-a-service for microservices-based

cloud applications. Dans *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, 50–57. IEEE.

Tagami, A., Ueda, K., Lukita, R., De Benedetto, J., Arumaithurai, M., Rossi, G., Detti, A. et Hasegawa, T. (2019). Tile-based panoramic live video streaming on icn. Dans *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, 1–6. IEEE.

Tam, A. S.-W., Xi, K. et Chao, H. J. (2011). Use of devolved controllers in data center networks. Dans *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 596–601. IEEE.

Tennenhouse, D. L. et Wetherall, D. J. (1996). Toward an active network architecture. Dans *Multimedia Computing and Networking 1996*, volume 2667, 2–16. SPIE.

Tokusashi, Y., Dang, H. T., Pedone, F., Soulé, R. et Zilberman, N. (2019). The case for in-network computing on demand. Dans *Proceedings of the Fourteenth EuroSys Conference 2019*, 1–16.

Tokusashi, Y., Matsutani, H. et Zilberman, N. (2018). Lake : the power of in-network computing. Dans *2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 1–8. IEEE.

Tootoonchian, A. et Ganjali, Y. (2010). Hyperflow : A distributed control plane for openflow. Dans *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, volume 3, 10–5555.

TS, Z. C. A. L. C. et Ng, E. (2010). *Maestro : A System for Scalable OpenFlow Control*. Rapport technique, Citeseer.

Vaucher, S., Yazdani, N., Felber, P., Lucani, D. E. et Schiavoni, V. (2020). Zipline : In-network compression at line speed. Dans *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 399–405.

Voellmy, A., Kim, H. et Feamster, N. (2012). Procera : A language for high-level reactive network control. Dans *Proceedings of the first workshop on Hot topics in software defined networks*, 43–48.

Woodruff, J., Ramanujam, M. et Zilberman, N. (2019). P4dns : in-network dns. Dans *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 1–6. IEEE.

Xia, W., Wen, Y., Foh, C. H., Niyato, D. et Xie, H. (2014). A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, *17*(1), 27–51.

Xu, C., Zeng, P., Yu, H., Jin, X. et Xia, C. (2020). Wia-nr : ultra-reliable low-latency communication for industrial wireless control networks over unlicensed bands. *IEEE Network*, *35*(1), 258–265.

Yang, F., Wang, Z., Ma, X., Yuan, G. et An, X. (2019). Switchagg : A further step towards in-network computation. *arXiv preprint arXiv :1904.04024*.

Yaqoob, A., Bi, T. et Muntean, G.-M. (2020). A survey on adaptive 360 video streaming : Solutions, challenges and opportunities. *IEEE Communications Surveys & Tutorials*, *22*(4), 2801–2838.

Zhang, S. (2019). An overview of network slicing for 5g. *IEEE Wireless Communications*, *26*(3), 111–117.

Zhang, W., Wood, T., Ramakrishnan, K. et Hwang, J. (2014). Smartswitch : Blurring the line between network infrastructure & cloud applications. Dans *6th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 14)*.

Zilberman, N. (2019). In-network computing. https ://www.sigarch.org/in-network-computing-draft/. Consulté le : 2022-08-14.