# Système de vidéosurveillance intelligent et adaptatif, dans un environnement de type Fog/Cloud

Hugo Sbai

Université de Lille

**Ecole doctorale Sciences pour l'ingénieur - SPI**

*Laboratoire CRIStAL / Equipe EAST*

# Système de vidéosurveillance intelligent et adaptatif, dans un environnement de type Fog/Cloud

Par SBAI Hugo

Thèse de doctorat en informatique

Dirigée par MEFTALI Samy

Présentée et soutenue publiquement le 20 avril 2018

Devant un jury composé de :

ABOULHAMID El Mostapha (Prof. Université de Montréal. Canada) : Rapporteur
TARI Abdelkamel (Prof. Université de Béjaia. Algérie) : Rapporteur
VANIER Sonia (Maitre de conférences de l'Université Paris 1) : Examinatrice
BOURENNANE El-Bay (Prof. Université de Bourgogne, Dijon) : Examinateur
MEFTALI Samy (Maitre de conférences de l'Université de Lille) : Directeur de thèse
AOUALI Djamel (Directeur de projets Dcarte Engineering) : Co-encadrant
BEESTON Nathalie (Directrice ISOSET) : Invitée

# REMERCIEMENTS

Le travail présenté dans cette thèse a été réalisé dans le cadre d'une collaboration entre le Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL) et l'entreprise Dcarte Engineering. Ainsi, je tiens tout d'abord à remercier Monsieur Olivier Colot, directeur du laboratoire CRIStAL, ainsi que Roberto Rapisarda, directeur de l'entreprise Dcarte Engineering, de m'avoir accueilli et donné les moyens pour effectuer mon travail de recherche.

Je tiens tout particulièrement à exprimer ma profonde gratitude à Monsieur Samy Meftali, Professeur à l'Université Lille 1, pour son encadrement et son suivi. Ses conseils, son soutien, ses encouragements permanents tout au long de ce travail, ses qualités humaines, son dynamisme et sa disponibilité ont joué un rôle déterminant. Qu'il trouve ici, l'expression de ma profonde reconnaissance.

Je remercie très vivement Monsieur Djamel Aouali, pour la confiance qu'il m'a témoignée, de m'avoir accueilli dans son groupe au sein de l'entreprise Dcarte Engineering, et pour avoir supervisé mes travaux. Je tiens à lui exprimer ma gratitude pour ses conseils, ses orientations et ses remarques pertinentes.

Tous mes remerciements à Monsieur El Mostapha Aboulhamid, Professeur à l'Université de Montréal et Monsieur Abdelkamel Tari, Professeur à l'Université de Béjaia, d'avoir accepté de rapporter sur cette thèse. Merci à Monsieur El-Bay Bourennane, Professeur à l'Université de Bourgogne, et Madame Sonia Vanier, Maître de conférence de l'Université Paris 1, d'avoir accepté d'être examinateurs. Merci également à Madame Nathalie Beeston, directrice de l'entreprise ISO SET, d'avoir accepté d'assister à ma soutenance.

Merci à tous mes collègues de l'entreprise Dcarte Engineering, et particulièrement ceux de mon groupe de recherche qui de près ou de loin m'ont aidé, à la réalisation de ce travail. Qu'ils trouvent ici l'expression de ma reconnaissance pour leur témoignage de sympathie et d'amitié. Ce fut, en effet, un plaisir de partager quotidiennement de si bons moments avec eux. Je tiens à remercier tout particulièrement Karim pour son soutient et son aide précieuse dans les moments les plus délicats, sans qui la réalisation de ce travail aurait été compromise. Merci également à Lucie, Idriss, Nathalie, Coralie, Vincent, Angelo et bien d'autres.

Enfin, je tiens à remercier les membres de ma famille, tout particulièrement ma grand-mère, pour leur soutien et leurs encouragements dans les moments les plus difficiles, et qui par leur présence m'ont donné la force d'aller jusqu'au bout de ce travail.

# SOMMAIRE

# TABLE DES FIGURES

# TABLE DES TABLEAUX

## Résumé

Les systèmes de vidéosurveillance utilisent des caméras sophistiquées (caméras réseau, smart caméras) et des serveurs informatiques pour l'enregistrement vidéo dans un système entièrement numérique. Ces systèmes intègrent parfois des centaines de caméras et génèrent une quantité colossale de données, dépassant largement les capacités des agents humains. Ainsi, l'un des défis modernes les plus importants est de faire évoluer un système basé sur le Cloud intégrant plusieurs caméras intelligentes hétérogènes et l'adapter à une architecture Fog/Cloud pour en améliorer les performances.

Les FPGA sont de plus en plus présents dans les architectures FCIoT (FoG-Cloud-IoT). Ils sont caractérisés par des modes de configuration dynamiques et partiels, permettant de s'adapter rapidement aux changements survenus tout en augmentant la puissance de calcul disponible. De telles plateformes présentent de sérieux défis scientifiques, notamment en termes de déploiement et de positionnement des FoGs.

Cette thèse propose un modèle de vidéosurveillance composé de caméras intelligentes plug & play, dotées de FPGAs dynamiquement reconfigurables sur une base hiérarchique FOG/ CLOUD. Dans ce système fortement évolutif, à la fois en nombre de caméras et de cibles trackées, nous proposons une approche automatique et optimisée d'authentification des caméras et de leur association dynamique avec les FoGs. L'approche proposée comporte également une méthodologie pour l'affectation optimale des trackers matériels aux ressources électroniques disponibles pour maximiser les performances et minimiser la consommation d'énergie. Toutes les contributions ont été validées avec un prototype de taille réelle.



## Abstract

CCTV systems use sophisticated cameras (network cameras, smart cameras) and computer servers for video recording in a fully digital system. They often integrate hundreds of cameras generating a huge amount of data, far beyond human agent monitoring capabilities. One of the most important and modern challenges, in this field, is to scale an existing cloud-based video surveillance system with multiple heterogeneous smart cameras and adapt it to a Fog / Cloud architecture to improve performance without a significant cost overhead.

Recently, FPGAs are becoming more and more present in FCIoT (FoG-Cloud-IoT) platform architectures. These components are characterized by dynamic and partial configuration modes, allowing platforms to quickly adapt themselves to changes resulting from an event, while increasing the available computing power. Today, such platforms present a certain number of serious scientific challenges, particularly in terms of deployment and positioning of FoGs.

This thesis proposes a video surveillance model composed of plug & play smart cameras, equipped with dynamically reconfigurable FPGAs on a hierarchical FOG / CLOUD basis. In this highly dynamic and scalable system, both in terms of intelligent cameras (resources) and in terms of targets to track, we propose an automatic and optimized approach for camera authentication and their dynamic association with the FOG components of the system. The proposed approach also includes a methodology for an optimal allocation of hardware trackers to the electronic resources available in the system to maximize performance and minimize power consumption. All contributions have been validated with a real size prototype.

# 1 INTRODUCTION

## 1.1 Contexte et motivation

Les progrès technologiques en informatique et plus précisément dans la conception de capteurs stimulent le développement de nouvelles applications qui transformeront, à terme, les systèmes de vision traditionnels en réseaux de caméras intelligentes omniprésentes. Les diverses applications concernées par les réseaux multi-caméras couvrent un champ très large incluant les maisons intelligentes, la bureautique par détection d'occupation, la sécurité et la surveillance, les réseaux mobiles, les interfaces homme-machine, les bornes interactives et les systèmes de réalités virtuelles. Les capteurs d'images extraient aujourd'hui de précieuses et nombreuses informations sur leur environnement et les événements s'y déroulant. En acquérant ce type de données très riche en informations, ils permettent le développement d'applications d'interprétation fondées sur la vision. Ces applications permettent de nombreuses fonctionnalités qui vont de l'interprétation d'événements en temps réel dans des environnements intelligents par l'adaptation, à l'élaboration de modèles de comportement d'utilisateur basés sur des observations à long terme dans un environnement intelligent. Les réseaux multi-caméras représentent un domaine multidisciplinaire permettant de nombreuses opportunités conceptuelles et algorithmiques, qui sont nécessaires à la vision par ordinateur, au traitement du signal, à l'informatique embarquée, ainsi que pour les réseaux de capteurs filaires et sans fil [NB07].

Aujourd'hui, la vidéosurveillance utilise des caméras sophistiquées (*caméras réseau*, *smart caméras*) et des serveurs informatiques pour l'enregistrement de la vidéo dans un système numérique qui comporte de nombreuses caméras *(~100 caméras*). Ce dernier génère une grande quantité d'informations vidéo qui dépasse très largement les capacités des employés de surveillance. Aujourd'hui, le nombre de caméras et la quantité de données continuent de croitre exponentiellement, de nouvelles approches sont donc nécessaires pour mieux gérer ce mécanisme.

Le tendance actuelle est donc de pousser les ressources élastiques telles que le calcul et le stockage à la limite des capacités réseaux, ce qui motive le déploiement de l'informatique du *Fog* en raison de l'omniprésence d'appareils intelligents connectés s'appuyant sur des services *Cloud* afin de soulager le réseau d'accès et de réduire la latence des échanges de ces applications, tout en gardant une bonne qualité de service (*QoS*). Le *Fog* maintient les données

et le calcul proches des utilisateurs finaux à la périphérie du réseau et fournit ainsi une nouvelle génération d'applications et de services aux utilisateurs finaux avec une faible latence et une bande passante élevée [YLL15].

Les FPGA (*Field Programmable Gate Arrays*) sont des dispositifs en silicium préfabriqués qui peuvent être programmés sur le terrain dans les architectures de plateforme *Fog-Cloud-IoT*. Leurs composants sont caractérisés par des modes de configuration dynamiques et partiels, permettant aux plateformes *Fog-Cloud-IoT* de s'adapter rapidement aux changements survenus lors d'un événement, et d'ouvrir de nouvelles possibilités aux concepteurs pour améliorer la disponibilité et la continuité des services fournis sur ces plateformes. Elles comprennent des blocs logiques programmables mettant en œuvre des fonctions logiques, des routages programmables qui relient ces fonctions logiques, et des blocs d'E/S connectés à ces blocs logiques via l'interconnexion de routage. Les FPGA offrent une solution moins chère et une mise sur le marché plus rapide que les circuits intégrés spécifiques à l'application (*ASIC*) qui nécessitent normalement beaucoup de ressources en termes de temps et d'argent pour obtenir le premier périphérique.

Pour que le traitement de données soit efficace et en temps réel, des chercheurs proposent de distribuer des systèmes de vidéosurveillance basés sur des caméras intelligentes, des boitiers *Fogs* et des serveurs *Cloud*. Néanmoins, tous ces systèmes sont statiques car le nombre et l'emplacement des caméras sont fixes : toute modification du nombre de caméras, de boîtiers *Fogs* ou de leurs emplacements impose des configurations manuelles fastidieuses. Dans des systèmes de vidéosurveillance efficaces, le nombre de caméras et leur emplacement peuvent changer de façon dynamique afin d'adapter le système à de nouvelles situations ou à des besoins particuliers.

L'un des défis modernes est que les systèmes actuels de vidéosurveillance contiennent un certain nombre de caméras intelligentes ayant des puissances de calcul différentes et des quantités d'énergie disponibles différentes : dans chaque objet, dans chaque nuage et dans chaque fog, il y a un nombre limité de ressources.

Dans de tels systèmes les méthodes et outils d'optimisation doivent prendre en compte plusieurs paramètres liés aux caractéristiques physiques du système ainsi qu'à la disposition de ses éléments. Ces paramètres sont :

- *Le volume de contacts de la tâche avec toutes les autres fonctions du système.*
- *La quantité de données que doit recevoir la tâche de chaque capteur dans le système (capteurs de la caméra).*
- *Le coût unitaire (temps) de connexion entre chaque paire d'objets (entre chaque objet et Fog, et entre chaque objet et nuage).*
- *Le coût unitaire (consommation d'énergie) de communication entre chaque paire d'objets et entre chaque objet et Fog ainsi qu'entre chaque objet et nuage.*
- *Le coût de l'énergie pour effectuer la tâche sur chaque objet, sur le nuage et sur le brouillard.*

Le but de telles méthodes étant d'optimiser les performances du système comme les ressources utilisées, le temps d'exécution ou la consommation d'énergie. Il s'agit là d'objectifs souvent contradictoires d'où le recours à des éléments matériels modernes comme les FPGA partiellement et dynamiquement reconfigurables.

## 1.2 Objectifs

Ce travail a pour objectif de proposer une approche permettant d'exploiter efficacement des systèmes de vidéosurveillance déployés dans une multitude de lieux en les faisant évoluer vers des architectures de type Fog/Cloud. Le but étant de proposer des méthodes assez générales pour pouvoir être réutilisées dans un large spectre applicatif mais qui doivent permettre de faire évoluer des systèmes existants sans remettre en cause tous les éléments les constituant initialement, pour ainsi maitriser les couts.

## 1.3 Contributions

Afin de remédier aux divers problèmes se présentant dans les systèmes de vidéosurveillance (ceux ciblés dans cette recherche), plusieurs contributions sont présentées et étudiées, allant de la proposition de modèles, à la réalisation d'un prototype en passant par divers développements logiciels.

Ces contributions peuvent être résumées ainsi:

- Proposer une méthode permettant aux caméras de s'authentifier et de se connecter au Fog auquel elles ont été affectées, afin de maintenir des charges équilibrées et optimiser ainsi les performances globales du système.
- Développer une approche pour affecter efficacement les tâches de suivi (par conséquent les cibles associées) aux différents éléments de calcul disponibles dans le système (caméras, fog et cloud). Une telle approche doit comporter des variantes optimisées à la fois pour les systèmes de petites tailles et pour ceux de tailles importantes.
- Réaliser des prototypes opérationnels pour valider les modèles et approches proposées.

## 1.4 Plan du document

Le chapitre 2 présente un ensemble de travaux en lien avec les différents thèmes de ce travail de recherche. Les travaux ici recensés sont relatifs aux architectures matérielles des systèmes de vidéosurveillance distribués à base de caméras intelligentes ainsi que des architectures hiérarchiques de type Fog/Cloud.

Le chapitre 3 s'efforce de détailler l'architecture du système ciblé ainsi que l'applicatif s'exécutant dessus. Il s'agit de caméras intelligentes communicant avec des composants Fog, qui à leur tour sont connectés à un serveur Cloud. L'application ciblée est le suivi de cibles se déplaçant dans un environnement comportant des obstacles.

Une nouvelle approche permettant la conception de systèmes de vidéosurveillance dynamiques et distribués basés sur des caméras intelligentes sans fil et une architecture mixte *Fog/Cloud,* est proposée dans le chapitre 4. L'algorithme proposé dans ce chapitre permet l'équilibrage de charge adaptative grâce au contrôle d'association entre les fogs et les caméras avec une complexité égale à $log(n)$ où $n$ est le nombre de *Fogs*.

Le chapitre 5 présente une heuristique axée sur ce système de vidéosurveillance, et s'intéressant aux tâches de « *tracking* ». Il propose une solution basée sur une architecture mixte *Cloud* et *Fog* computing, permettant notamment de minimiser deux mesures majeures de performance du système : *le temps de réponse* et *la consommation d'énergie*. L'objectif est de trouver le meilleur compromis entre ces deux paramètres afin de les minimiser. A cette fin, un algorithme prenant en compte les principaux paramètres a été défini. Ce dernier repose sur une méthode d'allocation dynamique de tâches (*de tracking*) qui se fait en cascade entre trois entités principales du système : les caméras, le sous-système *Fog* et le sous-système *Cloud*.

L'objectif du chapitre 6 est de proposer une autre méthode permettant également une assignation dynamique et automatique des tâches de suivi sur les éléments de traitement disponibles d'un système de vidéosurveillance. Ainsi, la technique de programmation linéaire en nombre entier ILP (*Integer Linear Programming*) sera au cœur des développements de ce chapitre.

Le chapitre 7 permettra d'enrichir cette recherche en présentant, très succinctement, certains développements liés au prototype. En effet, certains processus ne seront pas explicités afin de respecter le secret industriel et commercial de Dcarte Engineering qui a déposé plusieurs brevets.

En conclusion, le chapitre 8 s'attachera à mettre en perspective les résultats de cette recherche.

# 2 ETAT DE L'ART

Ce chapitre aborde les travaux existants relatifs au contexte et aux objectifs de la thèse sans toutefois se prétendre exhaustif. Ainsi, dans les sections suivantes, seront présentés les travaux importants liés à des domaines ouvrant un large spectre allant des architectures FPGA aux systèmes de type fog/cloud en passant par les problématiques réseaux telles que les topologies ou les protocoles de communication. Tout au long de ce chapitre, des sections seront dédiées à la discussion des travaux énumérés en vue d'en tirer l'ensemble des enseignements pertinents et justifiant les contributions de ce travail.

## 2.1 Introduction

Les caméras intelligentes sont composées de différents dispositifs discrets, disponibles sur le marché. Elément important d'une caméra, l'acquisition des images est traditionnellement confiée à des imageurs CMOS ou CCD. Ils ont chacun leurs avantages et leurs inconvénients. Les CCDs bénéficient de 25 ans d'expérience en matière d'amélioration de la qualité d'image [DS11]. Les capteurs CMOS sont bien plus récents et connaissent un essor important car ils offrent des performances très intéressantes. Ce sont aussi des capteurs ayant la possibilité d'adresser aléatoirement les pixels et ainsi de récupérer uniquement les parties de l'image désirées ou d'atteindre des fréquences extrêmement rapides en sortie pour de petites résolutions. Ceci est possible car chaque pixel possède son propre circuit de lecture.

Les caméras intelligentes modernes embarquent des ressources de calcul, pouvant être importantes et permettant même des implémentations massivement parallèles dans certains cas. Dans les caméras modernes on trouve ainsi des processeurs de type GPUs ou des circuits FPGAs de diverses tailles et natures. La manière de les programmer est différente mais ils partagent la faculté de posséder un potentiel important de parallélisme.

Les caméras intelligentes étant souvent utilisée en réseau, de manière à pouvoir collaborer au sein de systèmes de vidéosurveillance distribués, leur capacité à communiquer de façon sûre et efficace constitue un critère d'une importance capitale et ayant une incidence directe sur leur performances. Ainsi, la majorité des caméra intelligentes de nos jours sont dotées de multiples interfaces, filaires ou sans fils, supportant plusieurs protocoles de communication.

C'est justement grâce à de telles capacités de communication qu'il est aujourd'hui tout à fait envisageable de concevoir des réseaux de centaines de caméras intelligentes autour d'architectures hiérarchiques de types fog/cloud.

## 2.2 Les architectures matérielles embarquées

Les systèmes embarqués, tels que les caméras intelligentes, peuvent avoir des besoins architecturaux, physiques et opérationnels spécifiques. Consommation d'énergie, limitation de taille, opération temps-réel et autonomie peuvent apparaître comme étant des contraintes fortes, rendant le design de ces systèmes plus complexe comparé aux systèmes de type « *desktop* ». Grâce à l'évolution observée ces dernières années dans le domaine du microélectronique et des technologies VLSI (*Very Large Scale Integration*), une palette variée de dispositifs est

disponible aujourd'hui pour la conception des systèmes embarqués de vision. Chaque famille de dispositifs présente des avantages et inconvénients propres, faisant du choix de composants matériels un facteur prépondérant pour la performance, la flexibilité et la programmabilité d'un système donné. Il est donc essentiel de connaître les particularités et contraintes des dispositifs utilisés afin de mieux prévoir et optimiser le comportement du système en fonction de l'application envisagée. De façon générale, l'architecture matérielle d'une caméra intelligente peut être décomposée en trois parties principales (*voir figure 1*).

***Partie d'acquisition de données :*** composée essentiellement d'un capteur d'images. Néanmoins, d'autres types de dispositifs sensoriels peuvent être intégrés afin d'obtenir des informations supplémentaires sur la scène et l'environnement.

***Partie de traitement des données :*** l'exécution des différentes étapes du traitement de l'information est prise en charge ici. Les résultats obtenus peuvent être envoyés vers un système hôte et/ou être utilisés pour contrôler l'acquisition de nouvelles données.

***Partie de communication :*** responsable de la connexion du système embarqué avec le monde extérieur (système hôte ou réseau).

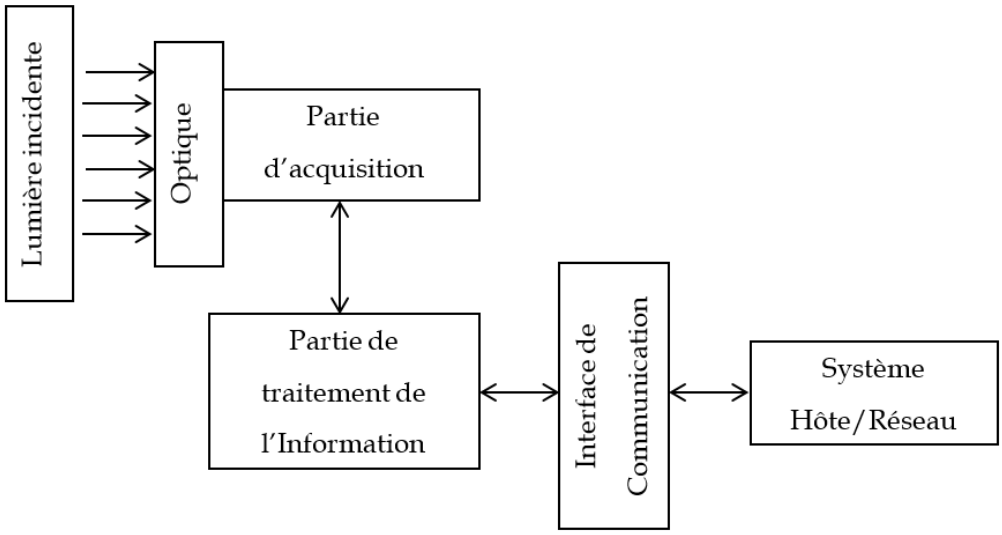


*Figure 1: Schéma simplifié de la structure d'une caméra intelligente*

Les FPGA utilisées pour implémenter des algorithmes de vision par ordinateur dans les caméras intelligentes [DGB11] peuvent atteindre une performance supérieure aux capacités des microprocesseurs modernes.

Cela est particulièrement vrai pour les opérations de traitement d'images basées sur l'accès au niveau du pixel car les FPGA sont massivement capables de traitement parallèle. Le parallélisme peut également être réalisé à l'aide de GPU mais en général, ces derniers ne sont pas adaptés aux caméras de type plug and play à cause des contraintes de consommation d'énergie. Des circuits intégrés dédiés spécifiques à l'application (ASIC) [AIM10] constituent une autre alternative. Cependant, le coût du développement et de la mise en œuvre de ces circuits utilisant un processus de fabrication à la pointe de la technologie poussent les concepteurs à explorer d'autres alternatives. Les FPGA sont facilement reconfigurables et offrent la capacité de connexion avec plusieurs dispositifs externes, comme par exemple les CMOS imageurs et d'autres types de capteurs. Ils sont souvent considérés comme une plate-forme de calcul intéressante pour la mise en œuvre de la caméra intelligente, soit autonome, soit faisant partie d'un système. Une architecture possible d'une caméra intelligente embarquée pour le contrôle et la surveillance intelligente est présentée dans la figure 2.

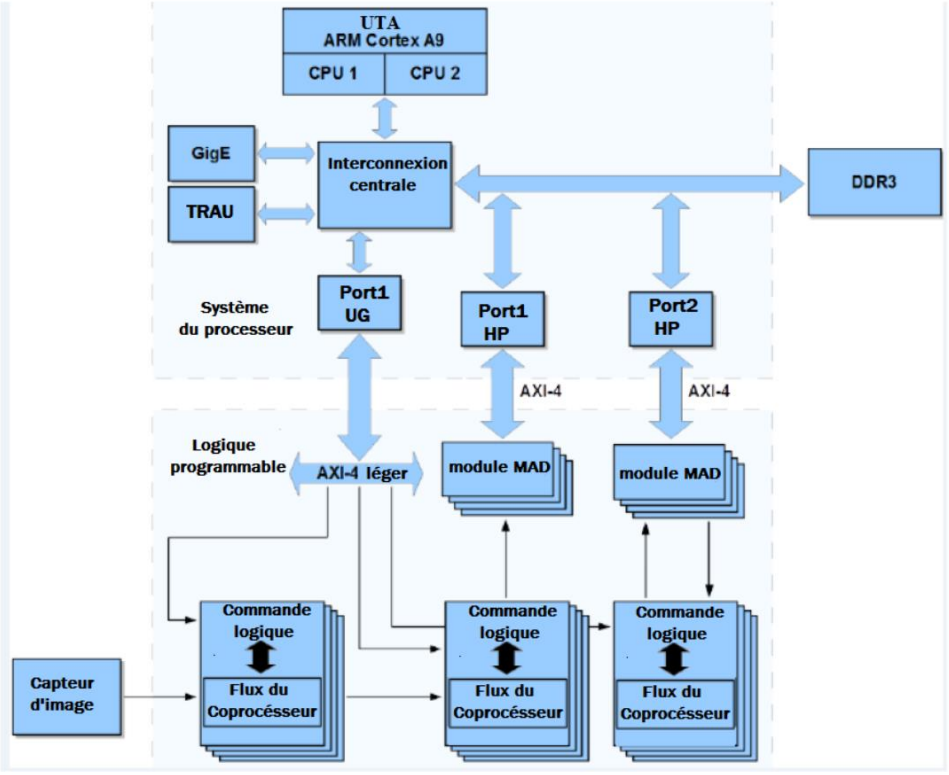


*Figure 2: Architecture d'une caméra intelligente [SZK15].*

Dans des conditions extrêmes, telles qu'un très grand nombre de caméras et/ou l'utilisation de données vidéo à haute résolution ou à fréquence de trame élevée, les exigences de bande

passante ou de puissance de traitement peuvent être impossibles à satisfaire. La manière la plus courante de gérer ces limitations est d'utiliser des algorithmes de compression de plus en plus sophistiqués. En revanche, l'utilisation de tels algorithmes augmente la demande de puissance de calcul pour le codage et le décodage vidéo, ce qui entraîne une augmentation des coûts d'application et de la consommation d'énergie. De plus, la compression dégrade la qualité de l'image, ce qui peut avoir un impact sur la qualité des résultats du traitement. Face aux problèmes et limitations évoqués ci-dessus, les réseaux de caméras intelligentes distribuées attirent de plus en plus l'attention. L'idée correspond à la tendance *Internet of Things* (*IoT*). Dans de tels réseaux, la majeure partie du traitement est effectuée dans le nœud de mesure (*une caméra intelligente*) avec un certain degré d'autonomie. Comme seules les informations utiles du point de vue de l'application sont transmises au serveur central ou à d'autres caméras du réseau, la charge sur l'infrastructure de communication est réduite. L'autonomie de la caméra intelligente permet d'ajuster le taux d'activité du capteur aux conditions actuelles, ce qui réduit la consommation d'énergie. En outre, les réseaux de type *IoT* permettent l'exécution collaborative à l'échelle du réseau des tâches désignées.

Etant donné que les caméras sont destinées au fonctionnement éventuellement prolongé de la batterie, elles sont équipées de microprocesseurs ou de microcontrôleurs de faible puissance mais relativement lents et d'interfaces de communication plutôt lentes. La puissance de traitement permet d'effectuer des opérations simples sur des images basse résolution (*CIF*, *QCIF*, *QVGA*) avec une vitesse de quelques images par seconde. L'interface est dans la plupart des cas basée sur une solution à faible puissance et faible bande passante. L'utilisation d'une telle interface améliore la durée de vie de la batterie, mais est en revanche trop lente pour la transmission d'images à haute définition. De plus, la plate-forme de calcul n'est généralement pas assez rapide pour compresser les images acquises ou le flux vidéo. Les données transmises dans de tels réseaux de capteurs sont généralement limitées aux métadonnées de scènes observées. Comme la faible consommation d'énergie n'est pas toujours l'unique préoccupation et que le traitement vidéo et image plus sophistiqué nécessite davantage de puissance de calcul, les caméras intelligentes incluent généralement des processeurs d'application ou une combinaison plusieurs processeurs pour une flexibilité accrue. De telles solutions sont capables d'effectuer une analyse vidéo en temps réel. Dans de nombreux cas, elles sont également capables de coder les vidéos à la volée. Ainsi, en plus des métadonnées de la scène, elles peuvent transmettre le flux vidéo en direct pour le visionnement, la consignation et l'archivage. Le streaming vidéo nécessite une interface de communication relativement rapide contribuant

à la consommation d'énergie globale du système. L'approche la plus courante consiste à utiliser l'infrastructure Ethernet ou *Wifi* existante. De nos jours, les architectures de type FPGA semblent mieux adaptées pour répondre aux trois besoins évoqués ci-dessus (communication, puissance de calcul et consommation d'énergie) comparé à des systèmes composés de processeurs généralistes.

Aussi, une attention particulière doit être accordée aux GPUs (Unité de Traitement Graphique). Il s'agit d'unités spécialisées dans l'informatique graphique mais pouvant également être utilisées avantageusement pour des calculs généraux. Au niveau architecture, le GPU est vu en tant que périphérique d'entrée sortie, doté de plusieurs unités de calcul (figure 3), piloté par un processeur classique [MV12]. Il est capable d'interpréter et d'exécuter ses propres instructions pour exécuter efficacement les parties parallèles du code. Le GPU sert donc d'accélérateur dans des architectures hétérogènes où il est piloté par un processeur de CPU, en apportant une puissance de calcul supplémentaire considérable.

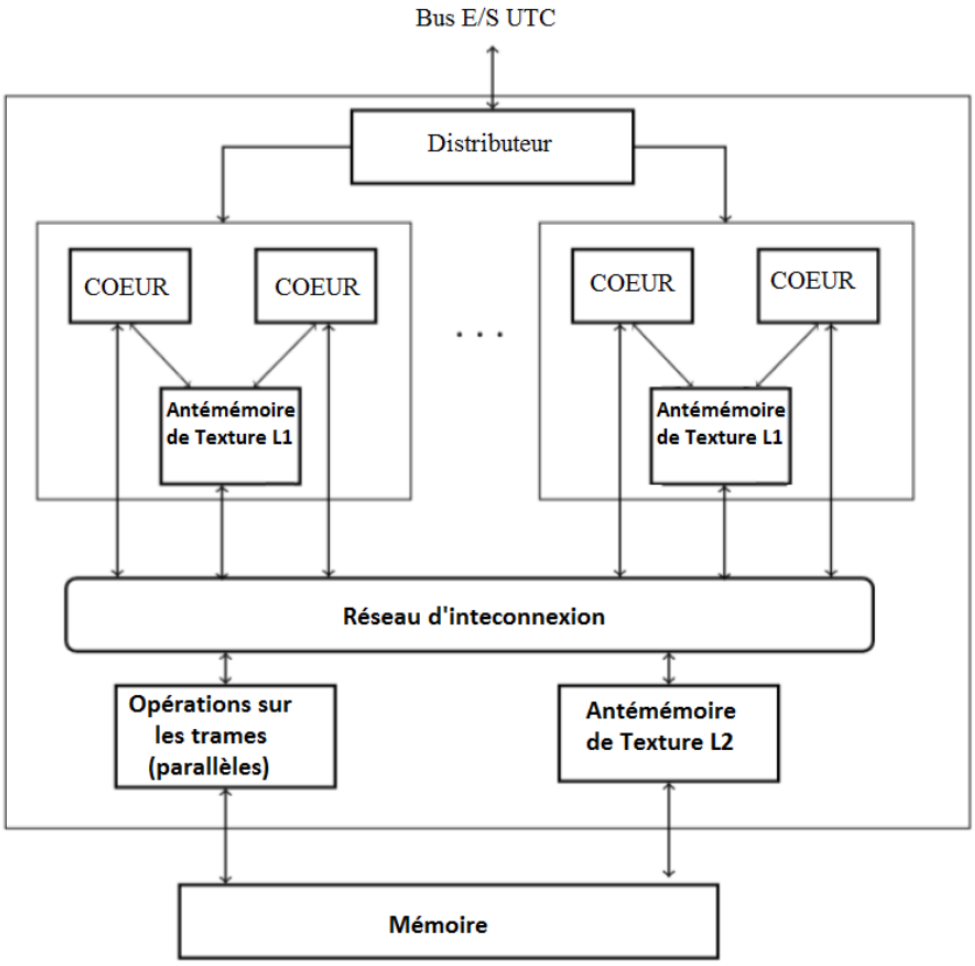


*Figure 3: Schéma simplifié de la structure d'un GPU.*

## 2.3 Les réseaux de smart caméras

Un réseau de caméras est un ensemble de caméras qui surveillent un environnement dans un but particulier. Les réseaux de caméras peuvent être le meilleur moyen d'obtenir des informations urgentes dans des situations où des vies humaines peuvent être en jeu. Ces réseaux seront essentiels pour les applications militaires, environnementales et de surveillance du XXI$^{ème}$ siècle [DCR08].

Les réseaux informatiques posent plusieurs défis de recherche à l'application directe des algorithmes traditionnels de vision par ordinateur. Tout d'abord, les réseaux informatiques contiennent généralement des dizaines à des centaines de caméras étant potentiellement adaptées à de nombreuses applications de type vision. Les recherches sur les réseaux informatiques ont été menées dans des environnements contrôlés où les caméras étaient fixées et leurs relations connues. Plusieurs systèmes ont été développés pour automatiser la surveillance. Un type de réseau de caméras permet de comprendre les relations spatiales entre les caméras du réseau. Par conséquent, des méthodes de découverte automatique de la topologie du réseau sont requises.

### 2.3.1 La topologie

Il existe deux principaux types de problèmes d'estimation de la topologie [RJR10] : le chevauchement et le non chevauchement. Dans le problème de chevauchement, il est supposé que les caméras observent des parties du même environnement à partir de différents points de vue. La relation entre ces échantillons est un graphe non orienté, appelé graphe de vision. Dans le problème sans chevauchement, il est supposé qu'aucune caméra n'observe la même partie de l'environnement. Les relations entre les caméras sont induites par la probabilité qu'un objet dans une caméra apparaisse dans une autre après un certain temps. Ces relations peuvent être modélisées avec un graphe non orienté appelé graphe de communication, où les poids de bord (*la limite de vision*) correspondent aux probabilités et aux temps de transition. Les graphes de vision et de communication (*et leur calcul*) sont fondamentalement différents. Dans ce cas, on considère un réseau hypothétique de *dix caméras* dans la figure 4 [DCR08]. La présence d'une ligne dans le graphique de communication ne permet pas de définir la présence de la même ligne dans le graphique de vision, puisque les caméras proches peuvent être pointées dans des directions différentes (*par exemple, caméra 1 et 3*). De manière similaire, les caméras regardant la même scène peuvent être physiquement distantes (*par exemple, les caméras 3 et 6*). Certains

chercheurs ne construisent pas clairement les graphiques de vision ou de communication. Au lieu de cela, ils modélisent les probabilités de transition et les temps de transition attendus entre les différentes régions des vues de caméra. De tels modèles permettent de dériver à la fois le graphique de vision et le graphique de communication.

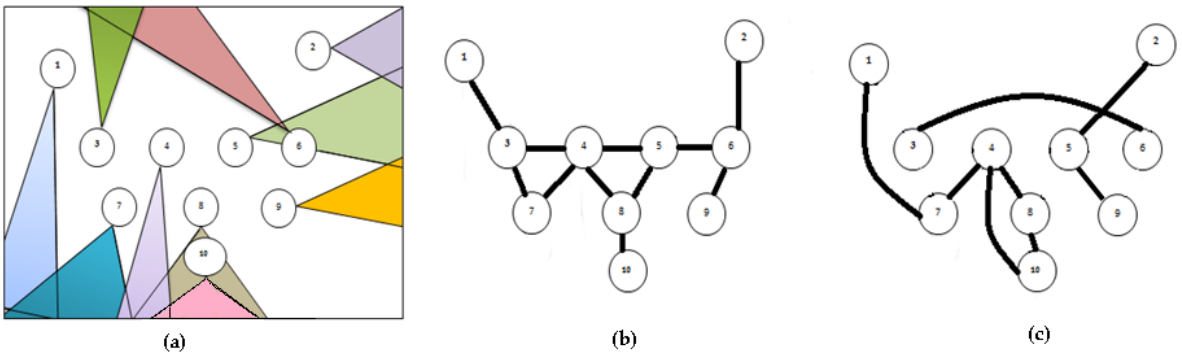


*Figure 4: (a) Un réseau de dix caméras (1-10) et leurs champs de vision correspondants ; (b) Le graphique de communication associé ; (c) Le graphique de vision associé.*

#### 2.3.1.1 Estimation du graphe de vision

Les auteurs de [MSK07] décrivent un algorithme simple d'estimation du graphe de vision basé sur la détection de mouvements. Ils appliquent le test de rapport de probabilité séquentiel (SPRT) pour accepter/rejeter la possibilité que deux caméras observent la même scène sur la base de la correspondance (ou de son absence) des réponses associées. Bien que l'algorithme soit simple, il n'est pas entièrement validé car il est testé sur un réseau de 3 caméras. [HDDCH07] décrit un algorithme efficace qui apprend la topologie quasi-optimale d'un grand réseau de 100 caméras en une heure seulement. L'algorithme est robuste car il n'est pas influencé par les conditions d'éclairage, les angles de caméra et la taille des objets en mouvement. Quelques exemples de vues de caméra correspondantes sont présentés à la figure 5.

L'estimation du graphe de vision se réalise en faisant correspondre les caractéristiques d'une vue de la caméra croisée [DCR08]. Tout d'abord, chaque caméra peut contenir des points de repères distinctifs dans son image qui sont susceptibles de correspondre à d'autres images de la même scène. Le nombre de caractéristiques et la longueur de chacune d'entre elles sont stockés dans une structure à longueur fixe. Chaque caméra compare et fait correspondre ses propres structures stockées avec celles des autres caméras. Un bord dans le graphe de vision est ainsi

établi si suffisamment de correspondances sont trouvées. Le condensé de caractéristiques d'une image est basé sur le détecteur/descripteur populaire [DGL04].

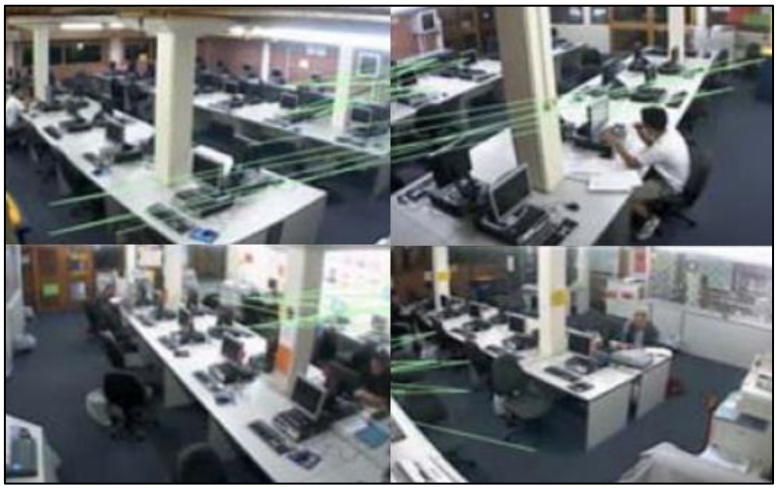

*Figure 5: Groupes d'images dont les vues se chevauchent. Les arcs du graphique de vision sont affichés en vert.*

#### 2.3.1.2 Estimation du graphe de communication

L'utilisation de la version stochastique de l'algorithme Expectation-Maximisation est optimale pour apprendre les trajectoires d'agents plausibles [MD06]. Cette approche n'utilise que des événements de détection provenant des capteurs déployés (équivalant à la détection de mouvement en vidéo). Le modèle suppose que le réseau soit traversé par un nombre fixe d'agents (jusqu'à 10), qui se déplacent entre les nœuds de capteurs, ainsi qu'un nœud de puits externe. Les résultats obtenus à partir de simulations et de données expérimentales suggèrent que la technique produit des graphiques de topologie précis dans diverses conditions et se compare bien aux autres approches [MD06].

#### 2.3.1.3 Estimation combinée du graphique

Des chercheurs exploitent les corrélations temporelles dans les observations des mouvements des agents à travers un réseau [MEB04]. Ils utilisent *Expectation-Maximisation* (*EM*) pour apprendre un modèle de mélange de Gauss (*GMM*) qui modélise les liens entre les zones d'entrée/sortie. Pour chaque vue de caméra, un ensemble de zones d'entrée/sortie est automatiquement appris [ME03]. Une valeur de corrélation croisée est calculée pour chaque lien possible d'une zone de sortie « ***i*** » à une zone d'entrée « ***j*** ». Si la corrélation croisée a un pic clair, il existe un lien réel entre ***i*** et ***j***. De plus, des estimations des temps et des probabilités de transition peuvent être extraites de la corrélation croisée. Si un lien est détecté entre les zones

de deux caméras, alors les deux caméras sont adjacentes (*dans le graphe de communication*) ou se chevauchent (*bord dans le graphe de vision*). En particulier, les deux caméras se chevauchent si le temps de transition est approximativement *nul*. Dans le cas contraire, la cible se déplace dans un chemin invisible et les caméras sont adjacentes dans le graphe de communication. Les résultats expérimentaux semblent prometteurs. La corrélation est efficace pour les environnements monomodaux en général mais n'est pas suffisamment flexible pour gérer les distributions multimodales. De telles relations peuvent se produire, par exemple, lorsque les voitures et les piétons font partie, en même temps, des observations. En général, en fonction des observations et plus le temps de transition est long, plus les correspondances fausses seront générées par la méthode. C'est pour cette raison que d'autres chercheurs améliorent l'approche précitée [TDG05]. Ils utilisent des distributions flexibles, multimodales et explicitement à la correspondance. Ceci est accompli en utilisant une mesure (plus générale) de dépendance statistique pour estimer la correspondance de l'objet. L'approche fait peu d'hypothèses et n'exige pas de supervision.

#### 2.3.1.4 Estimation de la topologie réseau

L'estimation de la topologie réseau de caméras est une tâche qui reste non résolue pour un grand nombre de caméras et de modèles d'activités complexes tels qu'une scène publique encombrée. Des études sont proposées pour favoriser une méthode qui contourne l'inférence de la topologie et le problème de la correspondance [WTG10]. L'utilisation de l'allocation LDA (*Latent Dirichlet Allocation*) permet de regrouper des trajectoires en activité et des chemins de modèles couramment utilisés par des objets sur plusieurs vues de caméra. La méthode a des faibles restrictions, la structure de la scène et le nombre de caméras. L'évaluation est effectuée sur deux grands ensembles de données réelles, dont chacun contient plus de *14 000 trajectoires*. En revanche, la méthode est limitée à l'apprentissage des relations entre les modèles d'activités ; les relations temporelles ne sont pas découvertes automatiquement, elles sont déterminées par un seuil temporel prédéfini, en modélisant les dépendances entre les activités sur les vues de caméra avec un modèle graphique probabiliste temporisé (*TD-PGM*). Les nœuds du modèle graphique représentent des activités dans différentes régions décomposées sémantiquement à partir des différentes vues de caméra, tandis que les arcs dirigés codent des relations causales entre ces activités [LXG09]. L'approche proposée consiste à installer un réseau de caméscopes dans une station de base fixe avec des scènes complexes et diverses, par exemple, de longues files d'attente, des halls, des plates-formes et des escaliers (*voir Figure 6*). La méthode

fonctionne sur une vidéo de *320 × 230* de basse qualité (*seulement 0,7 images par seconde*). [LXG09].

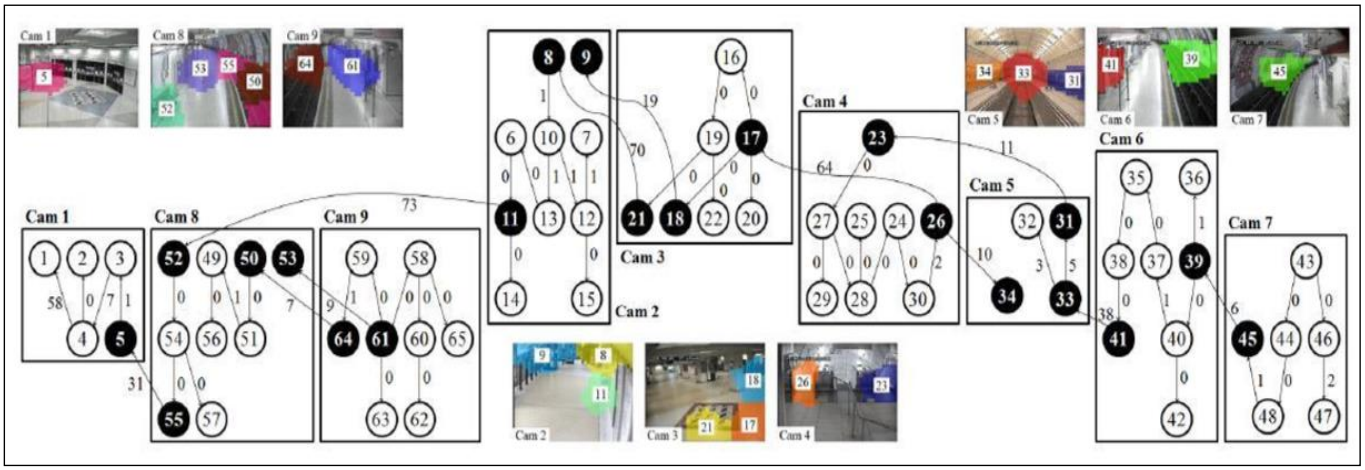


*Figure 6: Le graphe de dépendance d'activité appris. Les bords sont étiquetés avec leurs délais associés. Les régions et les nœuds avec des dépendances inter-caméras détectées sont mis en surbrillance.*

### *2.3.1.5* Discussion

Il ressort des différents travaux survolés dans cette section que pour tirer le meilleur des réseaux composés de plusieurs caméras avec des vues superposées et non chevauchantes, deux approches semblent performantes et pertinentes. Il s'agit de :

- Modéliser les graphiques de vision et de communication indépendamment
- Modéliser le graphique combiné qui permet dériver les graphiques de vision et de communication

Cependant, l'estimation de la topologie réseau semble à proscrire pour des réseaux contenant un nombre significatif de caméras. En effet, cette méthode devient beaucoup trop complexe et le graphique de communication nécessite l'utilisation d'outils sophistiqués pour modéliser les probabilités de transition et les délais.

## 2.3.2 Les protocoles et interfaces de communications

### 2.3.2.1 Interfaces de communication

L'objectif principal d'une caméra intelligente est de traiter en interne les données acquises afin de transmettre seulement les informations pertinentes sur une scène observée ; la capacité de transmettre des images entières en pleine résolution quand cela est nécessaire demeure un point important pour plusieurs applications. Dans ces cas, la bande passante de l'interface de communication doit être assez élevée pour supporter la transmission d'un flot vidéo.

En effet, la bande passante requise est fortement liée aux caractéristiques du capteur d'images qui supporte une certaine résolution et cadence d'acquisition de données. En revanche, la bande passante de transmission n'est pas le seul facteur affectant un protocole de communication. Généralement, une interface de caméra peut être classifiée en fonction de quatre facteurs principaux :

- Bande passante (cadence de réception/transmission de données) ;
- Portée et câbles (filaire ou sans fil, taille des connecteurs, longueur maximale des câbles, portée sans fil) ;
- Déterminisme et réactivité (latences de communication, robustesse, fiabilité) ;
- Interchangeabilité (compatibilité entre fabricants, pilotes logiciels).

Pour la communication filaire, *le tableau 1* illustre les principaux protocoles utilisés et le *tableau 2* l'illustre pour la communication sans fil.

*Tableau 1: Protocoles de communication filaire*

| Protocole | Bande passante théorique (bit/s) |
|---|---|
| Rs-232 serial link | 19200 bit/s. |
| USB 1.x Full-speed | 12 Mbit/s. |
| USB 2.0 Hi-speed | 480 Mbit/s. |
| FireWire or IEEE 1394a/b | 400/800 Mbit/s. |
| Ethernet, Fast Ethernet | 10/100 Mbit/s |
| GigE Vision (Gigabit Ethernet) | 1 Gbit/s |

*Tableau 2: Protocoles de communication sans fil*

| Protocole | Bande passante théorique (bit/s) | Portée sans fil (m) |
|---|---|---|
| WiFi IEEE 802.11a | 54 Mbit/s. | Jusqu'à 10m |
| WiFi IEEE 802.11b | 11 Mbit/s. | ~ 50m en inférieur, ~ 200m en extérieur. |
| WiFi IEEE 802.11g | 54 Mbit/s. | ~ 27m en inférieur, ~ 75m en extérieur. |
| Bluetooth | 1 Mbit/s. | ~ 10-100m. |
| ZigBee (IEEE 802.15.4) | 250 Kbit/s. | ~ 10-30m en inférieur, ~ 150m en extérieur. |

Discussion :

A titre d'exemple, si une caméra est équipée d'un capteur d'image de type *MT9M413* chez *Aptina Imaging* (*ancienne Micron Imaging*), capable de délivrer jusqu'à *660 Mpixels/s*, une interface *Camera link* serait nécessaire afin d'exploiter au mieux les capacités de l'imageur (*5.44 Gbit/s c.à.d. 680 Mbytes/s dans sa « full configuration »*) [FRO].

En revanche, pour une caméra autonome fonctionnant sur piles, le protocole sans fil *ZigBee* peut être préférable, en raison de sa très basse consommation en énergie [HPFA07, KSDH06], même si sa bande passante de *250 Kbit/s* rendra impossible la transmission vidéo en temps réel.

Une solution possible pour réduire les exigences de bande passante est l'utilisation des algorithmes de compression tels que *l'algorithme de Huffman, Lempel-Ziv etc.*

Les coûts d'implémentation d'un système de communication Gigabit Ethernet (*GigE Vision*) peuvent paraître intéressants au premier abord, mais le résultat final pourrait entraver la réactivité du système (*transmission des données par paquets, interruptions fréquentes, surcharge du système hôte pour le paquetage/dépaquetage*), et augmenter le temps de développement. En effet, le protocole GigE Vision est très récent alors que des protocoles comme *Camera link* et *FireWire* ont déjà fait leur preuve sur le terrain [MDP07].

#### 2.3.2.2 Protocoles de communication

Au début des années 1990, alors qu'Internet grandissait rapidement et devenait de plus en plus populaire, les chercheurs étaient mécontents de l'architecture statique TCP/IP [CC83, BC96]. Par conséquent, il existe des piles de protocoles flexibles pouvant être utilisées à la place des

piles de protocoles statiques telles que TCP/IP ou UDP/IP. Les piles de protocoles flexibles divisent les fonctionnalités de mise en réseau en blocs fonctionnels individuels, qui peuvent être dynamiquement liés les uns aux autres afin de former des piles de protocoles arbitraires. L'architecture de piles de protocoles dynamiques (*DPS*) a été développée dans le cadre du projet *EPiCS* [KBNH14] qui étudie le système informatique proprioceptif, par exemple, des systèmes informatiques qui surveillent leur environnement et adaptent leur comportement en fonction de l'observation. L'objectif de l'architecture DPS réside dans la fourniture d'une architecture réseau qui comprend un cadre de surveillance pouvant facilement adapter la pile de protocole sur plusieurs nœuds.

### 2.3.3 Stockage

Le développement des systèmes de vidéo sur IP implique une utilisation de plus en plus importante d'espace disque. Ceci pose un certain nombre de questions, et notamment celle de savoir quel espace disque sera nécessaire et la façon d'assurer un stockage garanti. Les facteurs à prendre en compte dans la détermination des besoins de stockage sont :

- le nombre de caméras ;
- le nombre d'heures d'enregistrement quotidien par caméra ;
- la durée de conservation souhaitée des données ;
- la détection des mouvements (événements) uniquement, ou enregistrement continu ;
- autres paramètres, comme par exemple la fréquence, la compression, la qualité d'image et la complexité.

Exemple :

A titre d'exemple et sans tenir compte du trafic réseau et d'autres problèmes techniques potentiels liés à la taille de fichier, dans le cas d'une solution *JPEG/Motion JPEG* impliquant la réception de fichiers individuels, les besoins de stockage peuvent varier en fonction de la fréquence, de la résolution et de la compression mises en œuvre.

Dans le tableau ci-dessous, les besoins de stockage des caméras 1, 2 et 3 diffèrent en fonction du nombre d'images par seconde et de la résolution.

Calcul :

*Taille de l'image x nombre d'images par seconde x 3600 s* = Ko par heure/1000 = *Mo par heure*
*Mo par heure* x nombre d'heures de fonctionnement par jour/1000 = *Go par jour*

*Go par jour* x durée de conservation = ***besoin de stockage.***

*Tableau 3: Exemples de caméras ayant des besoins de stockage différents*

| Caméra | Résolution | Taille de l'image (Ko) | Images par seconde | Mo/heure | Heures de fonctionnement | Go/jour. |
|---|---|---|---|---|---|---|
| N°1 | CIF | 13 | 5 | 234 | 8 | 1.9 |
| N°2 | CIF | 13 | 15 | 702 | 8 | 5.6 |
| N°3 | 4CIF | 40 | 15 | 2160 | 12 | 26 |

Pour le *MPEG4*, les images parviennent suivant un flux continu et non pas sous forme de fichiers individuels. Les besoins de stockage dépendent du débit, c'est-à-dire de la quantité de données vidéo transmises. Le débit est quant à lui le résultat d'une fréquence déterminée, selon une résolution et un taux de compression donnés et en fonction du degré de mouvement de la scène.

#### 2.3.3.1 Redondance

- **RAID (***Redundant Array of Independent Disks***) :** cette méthode consiste essentiellement à répartir les données sur plusieurs disques durs, de telle manière qu'en cas d'échec sur l'un des disques, les données puissent être récupérées sur un autre disque (*fig.7a).*
- **Réplication des données :** technique commune à de nombreux systèmes réseau. Les serveurs de fichiers sur le réseau sont configurés de manière à ce que les données soient répliquées entre les différents serveurs (*fig.7b*).

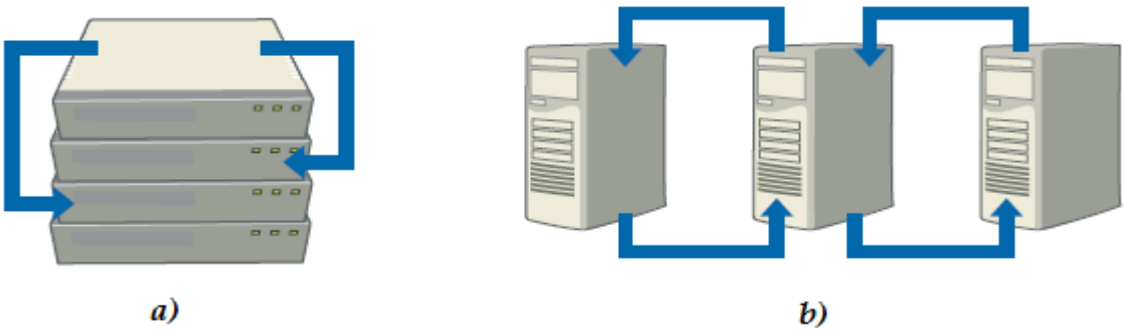


*Figure 7: a) Les données sont répliquées sur les disques durs b) Les données sont répliquées entre les différents serveurs.*

- **Sauvegarde sur bandes magnétiques :** il s'agit d'une méthode alternative ou complémentaire. Il existe toute une panoplie de logiciels et d'équipements prévus à cet effet. En général, pour faire face aux vols éventuels ou aux incendies, les politiques de sauvegarde préconisent en outre de stocker les bandes à l'extérieur.
- **Mise en cluster de serveurs :** Un système en cluster est une architecture composée de plusieurs serveurs formant des nœuds distincts. Chacun de ces nœuds est capable de fonctionner indépendamment des autres, dans le but d'assurer une haute disponibilité des données. Ainsi dans ces architectures quand un des nœuds tombe en panne, un autre nœud du cluster prend le relais afin d'assurer la disponibilité des données. Souvent, les serveurs d'un même cluster partagent en outre la même adresse IP, rendant ainsi la procédure de basculement automatique totalement transparente pour l'utilisateur.

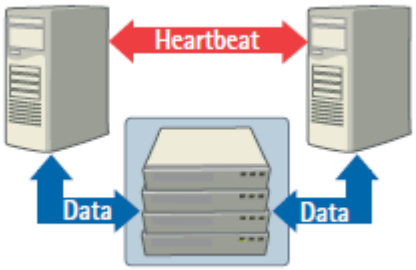


*Figure 8: Mise en cluster de serveurs.*

- **Dédoublement des réceptionnaires vidéo :** une méthode courante visant à garantir la récupération des données en cas de problèmes, et l'archivage externe de la vidéo réseau consiste à envoyer la vidéo simultanément vers deux serveurs différents, situés en des lieux différents. Ces serveurs peuvent bien sûr disposer des techniques RAID, fonctionner en clusters ou répliquer leurs données sur d'autres serveurs éventuellement encore plus éloignés.

#### 2.3.3.2 Les solutions de stockage

Deux approches sont possibles s'agissant de la question du stockage sur disque dur : l'une consiste à confier le stockage au serveur exploitant l'application ; l'autre est une solution indépendante, dans laquelle le stockage n'est pas lié au serveur exploitant l'application.

### *a. Stockage embarqué (DAS - Direct Attached Storage)*

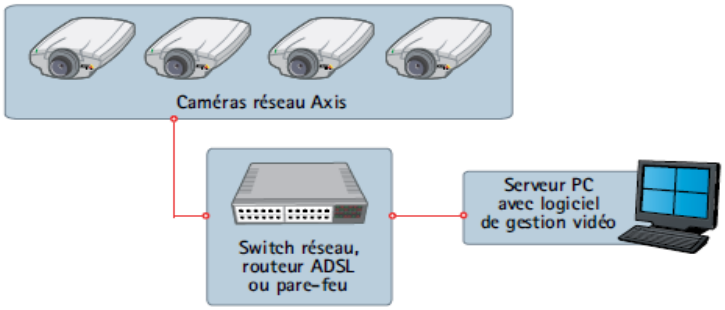


*Figure 9: Stockage embarqué DAS.*

Le stockage embarqué est la solution de stockage sur disque dur la plus fréquemment utilisée dans les installations de petite à moyenne envergure. Le disque dur se trouve sur le PC exploitant l'application de gestion vidéo (serveur applicatif). L'espace disponible dépend du PC et du nombre de disques durs qu'il contient. La plupart des PC accueillent 2 disques durs, certains allant jusqu'à 4. Chaque disque dur peut contenir jusqu'à environ 300 Go, soit une capacité totale d'environ 1,2 To (Téraoctets).

### *b. Stockage déporté NAS (Network Attached Storage) ou SAN (Storage Area Network):*

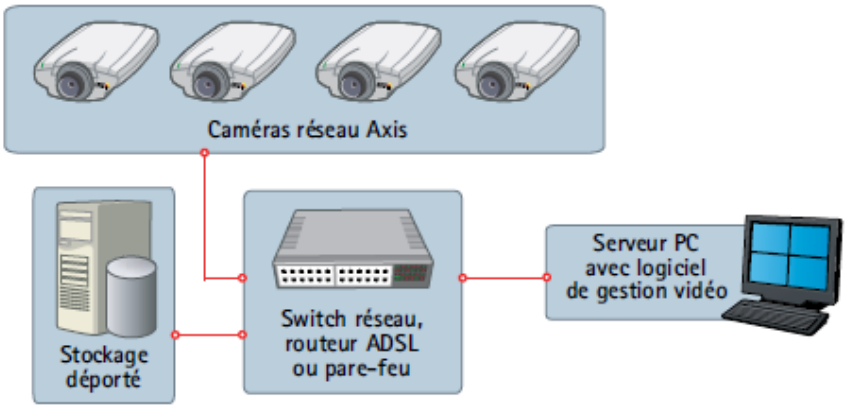


*Figure 10: Stockage déporté NAS ou SAN.*

Dans les applications où la quantité d'informations stockées et les contraintes globales dépassent les limites d'un système de stockage embarqué (*DAS*), un système de stockage séparé est mis en œuvre : NAS *(Network Attached Storage)* et SAN *(Storage Area Network).* Dans le cas d'un système NAS, un même dispositif de stockage directement rattaché à un LAN propose un stockage partagé à tous les clients du réseau. Facile à installer et à administrer, un dispositif de type NAS constitue une solution de stockage bon marché. La capacité de traitement des

données entrantes est cependant limitée. Une solution SAN (figure 10) propose une plate-forme de stockage polyvalente à grande vitesse, reliée par fibres optiques à un ou plusieurs serveurs. Les utilisateurs peuvent accéder à tous les dispositifs de stockage du SAN via les serveurs. La capacité de stockage est configurable jusqu'aux centaines de *To* (*Téraoctets*). Le stockage centralisé des données réduit les contraintes administratives. La différence entre les deux est qu'une plate-forme NAS permet de stocker un fichier entier sur un seul disque dur, tandis que le SAN propose un stockage par blocs sur différents disques durs. Ce type de configuration de disque permet d'exploiter des solutions de taille adaptée, autorisant le stockage d'importantes quantités de données.

## 2.4 Modèles de calcul

Les modèles de calcul flot de données ont vu le jour afin simplifier la conception d'applications composées d'acteurs, en réseau, échangeant des données à travers des liens de communication. On distingue des modèles flot de données statiques et dynamiques, offrant chacun un certain nombre d'avantages.

L'avantage principal des modèles Synchronous Data Flow (SDF) et des modèles statiques en général est la disponibilité de méthodes d'analyse statique pour diverses propriétés importantes telles que les tailles optimales pour les buffers mémoire ou le débit. Quant aux modèles dynamiques, ils sont particulièrement adaptés aux applications dynamiques, où a quantité de données échangée peut être modifiée pendant l'exécution selon des schéma de contrôles conditionnels. Ils sont de nos jours largement utilisés dans les réseaux de caméras intelligentes.

Ainsi, en utilisant un modèle de flot de donnée dynamique, la description d'une application, de vidéosurveillance distribuée, se fait par la décomposition en différents éléments indépendants et inter-communicants, chacun responsable de la réalisation d'une tâche ou traitement sur un ensemble défini de données provenant d'une source (capteur, mémoire, ou autre élément). La communication entre ces éléments est réalisée au moyen de l'échange de jetons (ou tokens), et l'exécution des différentes étapes de l'algorithme est réalisée de façon concurrente, en respectant bien évidemment les différentes relations de dépendance de données.

Il existe de nombreux autres modèles de calcul susceptibles d'être utilisés dans la modélisation d'application de vidéosurveillance distribuée. Les réseaux de processus de Kahn ( KPN ou réseaux de processus ) sont un autre modèle de calcul distribué dans lequel un ensemble de processus séquentiels déterministes communiquent via des canaux de type FIFO infinies. Le réseau de processus résultant présente un comportement déterministe indépendant des temps de

calcul ou de communication. Le modèle a été développé à l'origine pour modéliser des systèmes distribués mais son efficacité a été prouvés à maintes reprises pour la modélisation de systèmes de traitement de signal. Ainsi, les KPN sont aujourd'hui largement utilisés dans la modélisation de systèmes embarqués et de systèmes de calcul haute performance.

## 2.5 Tolérance aux fautes

Les réseaux de caméras intelligentes distribuées peuvent être utilisés dans diverses applications. Les contraintes particulières imposées par ces applications et leurs plateformes associées nécessitent de légères modifications des algorithmes afin de garantir une meilleure tolérance aux fautes :

- Les meilleurs résultats sont obtenus pour le suivi lorsque plusieurs caméras partagent des champs de vision qui se chevauchent. Dans ce cas, les caméras peuvent comparer les pistes qu'elles génèrent pour améliorer la précision de la piste globale générée par le réseau.
- Lorsque nous couvrons de vastes zones, nous ne pouvons peut-être pas nous permettre d'avoir suffisamment de caméras pour fournir des champs de vision qui se chevauchent. Le suivi doit alors estimer la probabilité qu'une personne vue dans une vue soit la même personne vue par une caméra différente à un moment ultérieur.
- Toutes les caméras peuvent ne pas être identiques. Nous pouvons, par exemple, utiliser des caméras de basse consommation et résolution pour surveiller une scène. Nous pouvons aussi utiliser des caméras dans différentes bandes spectrales, telles que l'infrarouge.
- Certaines caméras peuvent bouger, ce qui pose des problèmes pour l'étalonnage et l'élimination de l'arrière-plan.

## 2.6 Algorithmes et traitement distribués

### 2.6.1 Algorithmes distribués

Les algorithmes distribués ont un certain nombre d'avantages et sont une nécessité pratique dans de nombreuses applications. Les algorithmes centralisés limitent l'évolutivité du système, en revanche les algorithmes distribués peuvent, lorsqu'ils sont correctement conçus, fournir un certain degré de tolérance aux fautes.

Deux types d'algorithmes distribués ont été utilisés dans les caméras intelligentes distribuées : les algorithmes de consensus comparant les informations entre les nœuds pour améliorer les

estimations et les algorithmes de coordination transmettant le contrôle entre les nœuds. Consensus et coordination, les algorithmes utilisent différents styles de programmation et présentent des avantages distincts. Les algorithmes de consensus sont généralement considérés comme des systèmes de transmission de messages. Un exemple de consensus pour les caméras intelligentes distribuées est *l'algorithme d'étalonnage de Radke et al.* [RDC08]. Cet algorithme détermine les paramètres d'étalonnage externes (*position de la caméra*) d'un ensemble de caméras avec des champs de vision qui se chevauchent en trouvant des correspondances entre les caractéristiques extraites des scènes vues par chaque caméra. Il est formulé comme un système de passage de messages dans lequel chaque message inclut l'estimation d'un nœud de la position.

Les algorithmes de consensus conviennent bien aux problèmes d'estimation, tels que la détermination de la position. De nombreux algorithmes peuvent être formulés en tant que système de transmission de messages. Une caractéristique importante relative aux algorithmes de consensus est le critère de terminaison. Les systèmes distribués ne peuvent pas fournir une transmission fiable des messages. Par conséquent, la terminaison ne doit pas reposer sur une coordination stricte des messages en itérations.

Les algorithmes de coordination peuvent être considérés comme des systèmes de passage de jetons. Un jeton représente le lieu de contrôle pour le traitement. Les algorithmes de coordination sont bien adaptés à des problèmes tels que le suivi, dans lequel l'identité d'un sujet doit être maintenue sur une période prolongée. Ces algorithmes peuvent être considérés comme des protocoles, chaque nœud maintenant son propre état interne et échangeant des signaux avec d'autres nœuds pour affecter à la fois son propre état et celui des autres nœuds.

Plus récemment, le système de reconnaissance des gestes de *Lin* et *al*. [LLWO04] utilise un jeton pour représenter l'identité du sujet dont les gestes sont reconnus. Certaines extractions de bas niveau sont toujours effectuées localement, mais les phases finales de la reconnaissance gestuelle peuvent se déplacer d'un nœud à l'autre alors que le sujet se déplace et que les fonctions de plusieurs caméras doivent être fusionnées. Un protocole gère le transfert du jeton entre nœuds en s'assurant qu'ils ne sont ni dupliqués ni perdus. Le système de suivi de *Velipasalar* et *al*. [VSCWS06] utilise également un protocole pour échanger des informations sur les cibles. Chaque nœud exécute son propre *traqueur* pour chaque cible dans son champ de vision. Un protocole échange périodiquement des informations entre les nœuds sur la position de chaque cible. Ce système est considéré comme un algorithme de coordination plutôt que

comme un algorithme consensuel car un cycle d'échange d'informations est effectué à chaque période. Bien sûr, le suivi comprend à la fois le maintien de l'identité (*coordination*) et la position d'estimation (*consensus*).

### 2.6.2 Traitement distribué

Dans les applications de surveillance, le nombre de caméras nécessaires augmente avec l'augmentation de la zone à couvrir. Théoriquement, un algorithme distribué de vision est nécessaire dans ce cas. Ce dernier est décentralisé, calcule et diffuse des agrégats de données avec des exigences minimales de traitement et de communication et une bonne tolérance aux fautes. Les besoins de traitement et de communication augmentent lorsque le nombre de caméras augmente (si le traitement est centralisé). Les exigences peuvent être constantes dans une version d'algorithme distribuée (voir figure 11).

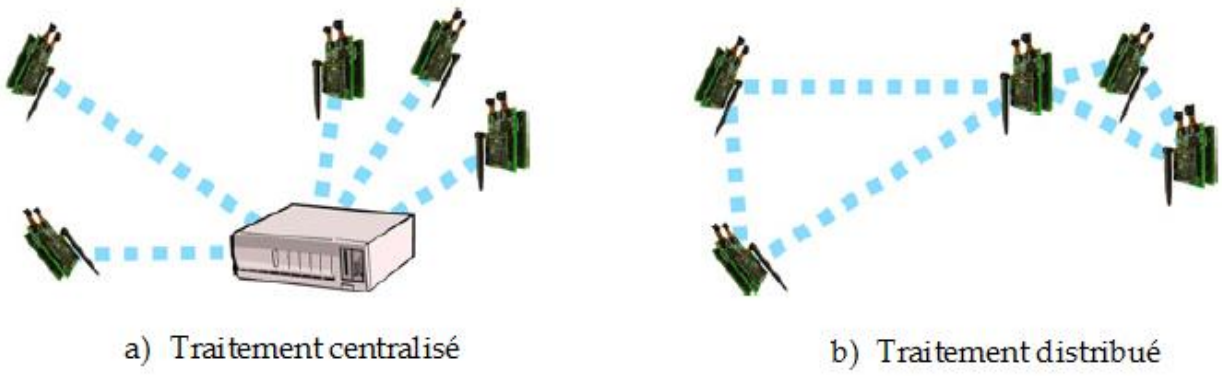


*Figure 11: Traitement centralisé et distribué pour un réseau de caméras.*

## 2.7 La reconfiguration dynamique partielle pour les FPGAs

### 2.7.1 Introduction

Les FPGA (*Field Programmable Gate Arrays*) sont principalement utilisés aujourd'hui à des fins de prototypage rapide [TM13]. Ils peuvent être reconfigurés plusieurs fois pour différentes applications. Des dispositifs *FPGA* ultramodernes tels que les *FPGA Xilinx Virtex* (*voir la figure 12*) supportent en outre une reconfiguration dynamique partielle de l'exécution qui révèle de nouveaux aspects pour le concepteur souhaitant développer de futures applications exigeant du matériel adaptable et flexible. En particulier dans le domaine de l'informatique, les applications haut de gamme bénéficieront des capacités de la nouvelle génération de dispositifs reconfigurables. Ce concept de reconfiguration dynamique partielle permet de créer des systèmes capables de supporter le passage d'une configuration à une autre pendant l'exécution.

Ces systèmes utilisent la flexibilité d'un FPGA en modifiant partiellement la configuration de sorte que seules les fonctions nécessaires à un moment donné soient mises en œuvre sur la puce [MVVL02]. Une fonction peut être remplacée par une autre fonction tandis que les autres restent actives. Pour résoudre le problème de la substitution et de la gestion des E/S, la configuration nécessite un module principal pour contrôler les tâches. Avec un tel système, il est possible d'économiser des ressources telles que les ports d'entrée/sorite et l'énergie en raison de l'externalisation du contrôle de configuration [UGHB04]. Le besoin de la surface de la puce devient plus petit, donc la consommation d'énergie peut être réduite. Néanmoins, les besoins en puissance de telles applications augmenteront avec l'augmentation de la fréquence et de la taille des reconfigurations [BHU03]. La création d'un tel système engendre deux défis: minimiser les ressources utilisées à tout instant et réduire la consommation d'énergie en concevant un système de contrôle qui gère intelligemment le contenu de la configuration des FPGA afin de minimiser le taux de reconfiguration.

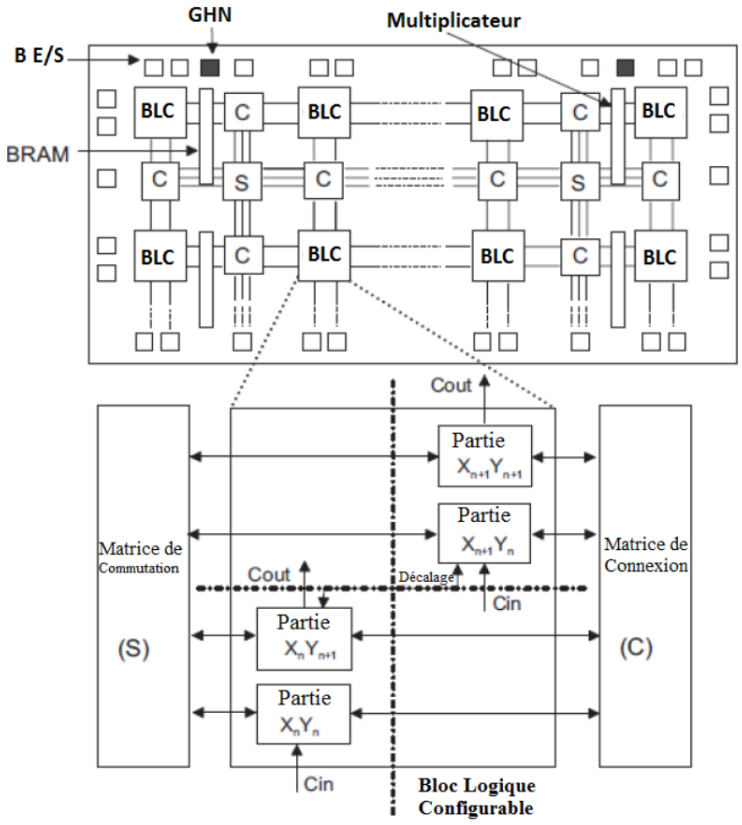


*Figure 12: Architecture des FPGAs Xilinx Virtex.*

La reconfiguration partielle peut être divisée en deux groupes : reconfiguration partielle dynamique (*DPR*) et reconfiguration statique partielle [MGP08]. La reconfiguration partielle dynamique, illustrée à la figure 12, également appelée reconfiguration partielle active, permet de modifier une partie du dispositif alors que le reste d'un FPGA est toujours en cours d'exécution.

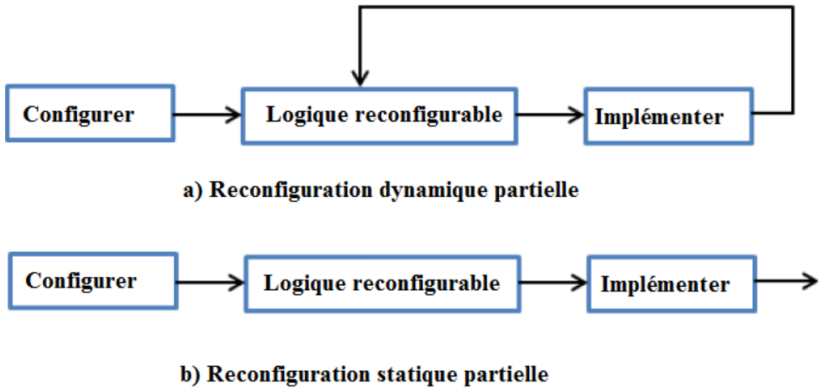


*Figure 13: Les deux groupes de reconfiguration partielle.*

En reconfiguration partielle statique, le périphérique n'est pas actif pendant le processus de configuration. En d'autres termes, alors que les données partielles sont envoyées dans le FPGA, le reste de l'appareil est arrêté et remis en place une fois la configuration terminée, comme indiqué sur la *figure 13*.

## 2.7.2 Les circuits reconfigurables

La principale technologie de mise en œuvre matérielle des systèmes (re-) configurables *RS* est une logique programmable sur site : des réseaux FPGA (*Field Programmable Gate Arrays*) et des *CPLD* (*Complex Programmable Logic Devices*).

Les principales raisons derrière les choix des FPGA et des CPLD étaient:

- Les FPGA offrent une solution moins chère et une mise sur le marché plus rapide que les circuits intégrés spécifiques à l'application (ASIC) qui nécessitent normalement beaucoup de ressources en matière de temps et d'argent pour obtenir le premier périphérique,
- La grande fiabilité des FPGA et des CPLD,

- L'indépendance des développeurs de systèmes et des fabricants de circuits intégrés.

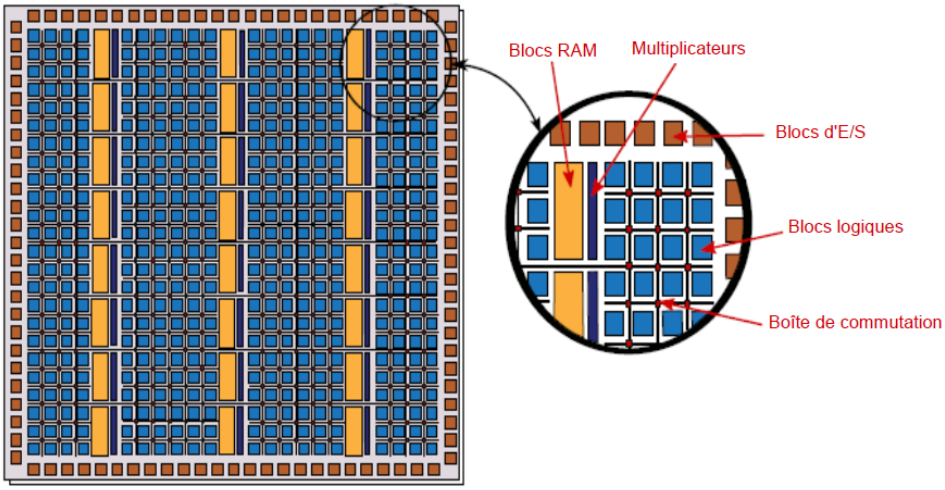


*Figure 14: Les différents blocs d'un FPGA Xilinx*

La nature flexible des FPGA les rend significativement plus grands, plus lents et plus gourmands en énergie que leurs homologues ASIC. Ces inconvénients découlent en grande partie de l'interconnexion de routage programmable des FPGA qui couvre près de 90% de la surface totale de ces derniers. Malgré ces inconvénients, ceux-ci présentent une alternative convaincante pour la mise en œuvre du système numérique en raison de leur gain de temps.

### 2.7.3 Les FPGAs

Les FPGA (*Field Programmable Gate Arrays*) sont des dispositifs en silicium préfabriqués qui peuvent être matériellement programmés sur le terrain pour réaliser presque n'importe quel circuit ou système numérique.

Les FPGA, comme illustré sur la figure 14, comprennent :

- Des blocs logiques programmables mettant en œuvre des fonctions logiques,
- Un routage programmable qui relie ces fonctions logiques,
- Des blocs d'E/S connectés à des blocs logiques via l'interconnexion de routage et des connexions hors puce.
- Des blocs BRAM pour l'implémentation des éléments de mémorisation
- Des ressources spécifiques comme des multiplieurs ou des DSPs

Un exemple généralisé de FPGA est représenté sur *la figure 15* où des blocs logiques configurables (*CLB*) sont disposés dans une grille bidimensionnelle et sont interconnectés par des ressources de routage programmables. Des blocs d'E/S sont disposés à la périphérie de la grille et sont également connectés à l'interconnexion de routage programmable. Le terme « *Programmable/Reconfigurable* » dans les FPGA indique leur capacité à implémenter une nouvelle fonction sur la puce une fois sa fabrication terminée. La reconfigurabilité/programmabilité d'un FPGA est basée sur une technologie de programmation sous-jacente, qui peut provoquer un changement de comportement d'une puce préfabriquée après sa fabrication.

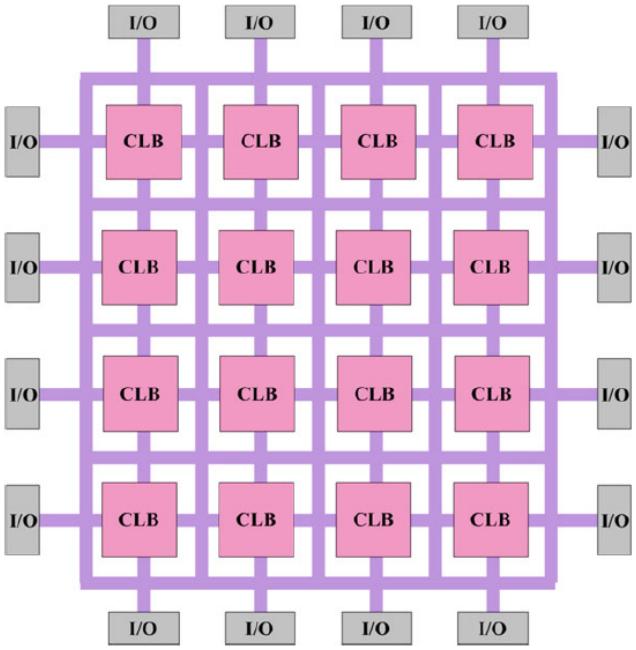


*Figure 15: Vue d'ensemble de l'architecture FPGA.*

A côté des blocs logiques, la première couche du FPGA contient aussi des blocs de mémoire (*BRAMs*) et d'autres blocs dotés de fonctionnalités spécifiques qui sont fréquemment utilisés dans les systèmes implémentés sur FPGA. Le fait d'intégrer ces blocs dans le FPGA permet de réduire la surface requise et d'augmenter la performance de ces blocs par rapport à leur implémentation en utilisant les blocs logiques. Parmi ces blocs, on trouve les multiplicateurs, les blocs DSP, les blocs mémoires et les processeurs embarqués. La deuxième couche d'un FPGA constitue la mémoire de configuration. Pour écrire dans cette mémoire, on utilise des fichiers binaires appelés *bitstreams*. Ces fichiers contiennent les informations de contrôle pour la configuration ainsi que les données de configuration. Ces données permettent de décrire

l'architecture du système se trouvant sur la première couche. *La figure 16* montre la couche mémoire de configuration pour un *Virtex-II Pro*. [LBMYB06]

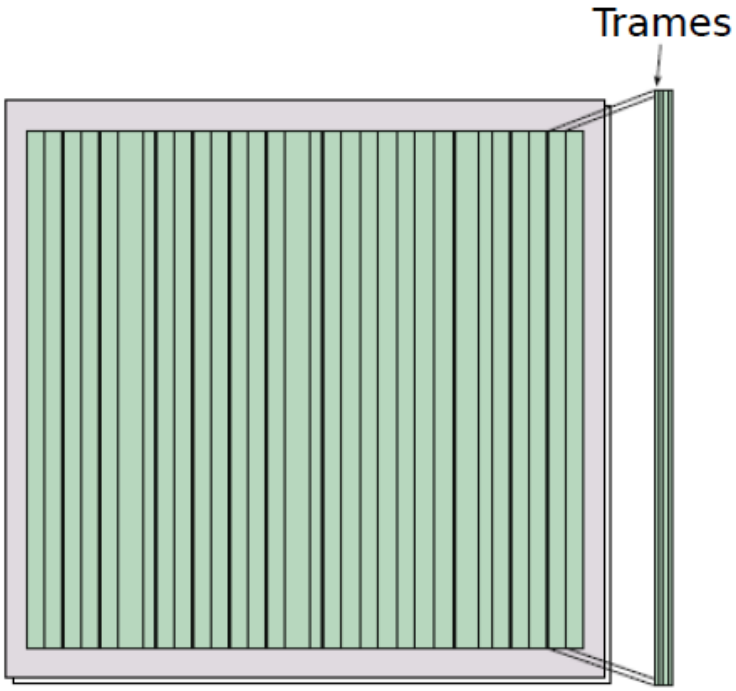


*Figure 16: La mémoire de configuration d'un FPGA Virtex-II Pro de Xilinx.*

## 2.8 Le contrôle de la reconfiguration dynamique partielle dans les FPGA

### 2.8.1 Le modèle de contrôle centralisé

Il existe différentes architectures de contrôle de l'adaptation dynamique des systèmes reconfigurables. Celles-ci peuvent être classées en des architectures centralisées et décentralisées. Le contrôle centralisé des systèmes dynamiquement reconfigurables sur FPGA a été étudié selon le type de l'implémentation du contrôle. Cette implémentation peut être autonome ou par un système d'exploitation (*OS*).

Que ce soit implémenté en autonome ou par un OS, l'utilisation d'un contrôleur centralisé pour l'adaptation dynamique des systèmes reconfigurables pourrait diminuer à la fois la flexibilité du contrôle et la possibilité de le réutiliser et de l'adapter à des systèmes différents. Centralisée ou distribuée, l'utilisation d'un système d'exploitation pour le contrôle d'un système reconfigurable a beaucoup d'avantages : cela permet d'optimiser le partage de ressources entre

différentes applications en rendant possible l'exploitation des ressources reconfigurables par différents processus à travers un système multitâche. La différence entre les systèmes d'exploitation pour les systèmes reconfigurables sur FPGA et ceux traditionnels est qu'ils doivent gérer l'hétérogénéité des tâches (*tâches logicielles et matérielles*) à ordonnancer et à faire communiquer.

Le contrôle décentralisé des systèmes reconfigurables n'a été traité que par quelques travaux sur FPGA. Cependant, il est très utilisé dans d'autres domaines tels que *les systèmes mécatroniques, les systèmes robotiques, etc*. Il y deux catégories de contrôle décentralisé : celle avec des contrôleurs non communicants et celle avec des contrôleurs communicants. La première catégorie de contrôle est utilisée pour des systèmes où une décision faite par un contrôleur n'a pas d'influence sur les autres**.** La seconde est utilisée quand les contrôleurs doivent échanger leurs décisions afin de respecter les contraintes/objectifs globaux du système. Dans ce cas, une coordination entre contrôleurs est nécessaire.

Le contrôle décentralisé peut être complètement distribué ou semi-distribué (*hiérarchique*). Dans cette approche, chaque région du FPGA est contrôlée par un contrôleur matériel. Le rôle de ce dernier est de contrôler les tâches exécutées par la région associée. Une reconfiguration est déclenchée par le contrôleur à chaque fois qu'une tâche gérée par la région contrôlée est requise. L'avantage de cette implémentation matérielle est qu'elle permet de réduire l'impact du contrôle en matière de temps d'exécution en favorisant l'exécution parallèle du contrôle et des autres tâches de l'application.

### 2.8.2 Le modèle de contrôle semi-distribué

En raison de la complexité croissante des applications ciblées par les systèmes reconfigurables à base de FPGA, la conception de contrôle de tels systèmes devient l'un des principaux obstacles rencontrés par les concepteurs. A cet effet, un modèle de contrôle semi-distribué basé sur la séparation entre différentes préoccupations de contrôle (*surveillance, décision et reconfiguration*) et sur la conception dirigée par les modèles (MDE) peut être utilisé afin de diminuer la complexité de celui-ci et de favoriser la réutilisation et l'évolutivité. Ce modèle est composé de contrôleurs distribués gérant chacun l'auto-adaptativité d'une région reconfigurable du système, et d'un coordinateur qui coordonne leurs décisions de reconfiguration afin de respecter les contraintes globales du système. Les implémentations sur FPGA ont montré que le modèle de contrôle semi-distribué est plus flexible, réutilisable et

évolutif que le modèle centralisé, au prix d'une légère augmentation des ressources matérielles requises [TMD12]. L'objectif principal de ce modèle de contrôle est de résoudre les problèmes de conception liés à la complexité, la vérification, la réutilisation et l'évolutivité. Pour atteindre ces objectifs, le modèle de contrôle semi-distribué combine trois points principaux : l'autonomie, la modularité et le formalisme.

### 2.8.2.1 Contrôleurs distribués autonomes modulaires pour l'auto adaptation

Chaque contrôleur réparti est composé de trois modules gérant la surveillance, la prise de décision de reconfiguration et la réalisation de la reconfiguration pour une région reconfigurable donnée, comme le montre la *figure 17*. Le module de surveillance recueille des informations sur le comportement de la région contrôlée et sur d'autres événements externes tels que ceux envoyés par des capteurs (par exemple : le flux vidéo est capté par le module de surveillance). Chaque module de décision prend des décisions locales sur la nécessité ou non d'une reconfiguration de la région contrôlée. Le rôle du module de reconfiguration est d'appliquer des actions de reconfiguration sur la région contrôlée, en chargeant les données de configuration requises (*bitstream*) dans la région reconfigurable via un port de configuration, tel que ICAP pour FPGA Xilinx. Après avoir chargé le train de bits requis, ce module notifie le module de décision afin qu'il mette à jour le mode courant de son automate.

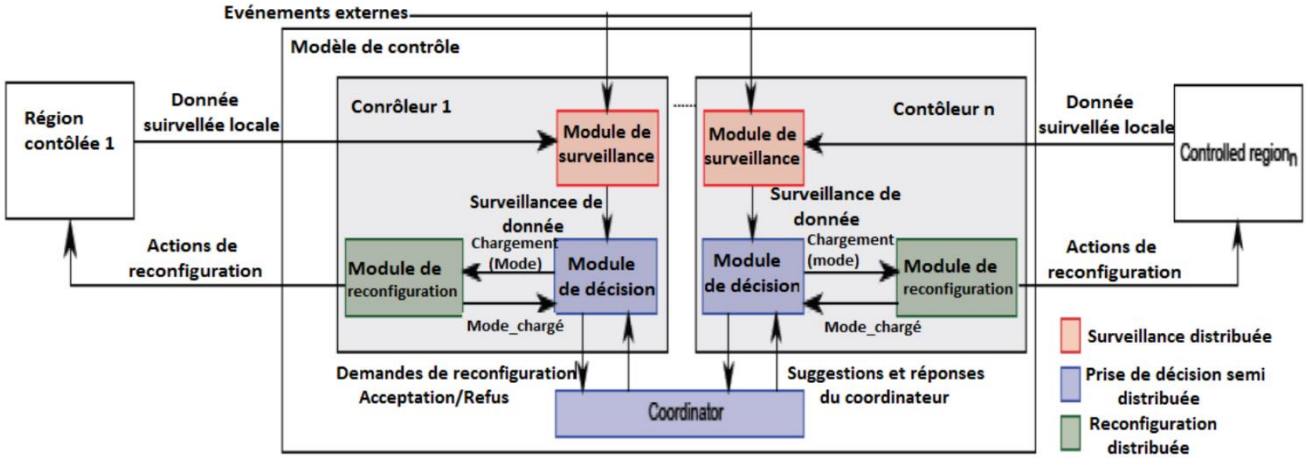


*Figure 17: Aperçu du modèle de contrôle*

#### 2.8.2.2 Prise de décision en utilisant les automates de mode

Le modèle de prise de décision est basé sur le formalisme mode automate [MR03]. Il est composé des automates distribués des contrôleurs et de l'automate du coordinateur. Ces automates communiquent à travers des informations de coordination chaque fois que l'un des contrôleurs estime qu'une reconfiguration de sa région contrôlée est requise. Dans ce cas, il envoie une demande de reconfiguration au coordinateur et attend sa décision. Si la demande implique également la reconfiguration d'autres régions afin de respecter les contraintes globales du système, le coordinateur envoie des suggestions de reconfiguration aux contrôleurs concernés. Ces contrôleurs peuvent accepter ou refuser ces suggestions. Après avoir traité les réponses des contrôleurs, le coordinateur prend sa décision, qui peut être soit l'autorisation, soit le refus de la reconfiguration demandée.

### 2.8.3 Le modèle de contrôle distribué

Le contrôle distribué pour l'adaptation des systèmes basés sur FPGA a été proposé par plusieurs travaux [TRK06]. Un contrôleur matériel a été affecté à chaque région reconfigurable afin de contrôler les tâches qu'elle exécute tout au long de l'exécution de l'application. Cependant, les décisions de reconfiguration ne dépendaient que d'un graphe de tâches et la corrélation entre les régions n'était pas traitée [SHB09]. Un modèle de contrôle distribué dans [PTG10]. Il est focalisé sur l'accès distribué au port de configuration (*ICAP*), permettant d'accélérer le processus de reconfiguration par rapport à l'accès centralisé, sans donner de détails sur les composants de contrôle utilisés (*surveillance, décisions, etc*.) Le contrôle distribué a également été utilisé pour *les systèmes multi-FPGA*, qui ont commencé à susciter l'intérêt et ont été étudiés dans plusieurs travaux. Cependant, un seul contrôleur a été utilisé par *FPGA* [ASJ09] [NTL11], ce qui implique une grande complexité de conception des contrôleurs, et aucun formalisme n'a été proposé pour modéliser le système de contrôle. Pour maîtriser la complexité de la conception, les modèles de contrôle formels sont importants car ils permettent un cycle de conception plus court en améliorant la réutilisation de la conception et en offrant une vérification formelle.

## 2.9 Les architectures de type fog /cloud

### 2.9.1 Définitions

Le brouillard (*Fog*) informatique est un paradigme informatique prometteur qui étend l'informatique en nuage (*Cloud*) en introduisant une certaine forme de hiérarchie. Similaire à

l'informatique en nuage, mais avec des caractéristiques distinctes, le *brouillard* (*Fog*) informatique fait face à de nouveaux défis en matière de sécurité et de confidentialité, en plus de ceux hérités du *cloud computing*. La dernière tendance du paradigme informatique est de pousser les ressources élastiques telles que le calcul et le stockage à la limite des capacités des réseaux ; ce qui motive le déploiement actuel de l'informatique du *brouillard* en raison de l'omniprésence d'appareils intelligents connectés s'appuyant sur des services *cloud*. Le *brouillard informatique* maintient les données et le calcul proches des utilisateurs finaux à la périphérie du réseau et fournit ainsi une nouvelle génération d'applications et de services à ces utilisateurs avec une faible latence et une bande passante élevée. En plus des clouds, les nœuds de *brouillard* peuvent également être connectés à des périphériques pauvres en ressources, tels que les téléviseurs intelligents/décodeurs, les passerelles et les terminaux. Le *brouillard informatique* est généralement associé au *cloud computing*. Par conséquent, les utilisateurs finaux, le brouillard et le nuage forment ensemble un modèle de prestation de services à trois couches, comme le montre *la figure 18*. Le brouillard informatique montre également une forte connexion au cloud computing en termes de caractérisation. Par exemple, les ressources élastiques (*calcul, stockage et mise en réseau*) sont les éléments constitutifs des deux, ce qui indique que la plupart des technologies de cloud computing peuvent être directement appliquées à l'informatique du brouillard. Puisque les appareils intelligents sont généralement insuffisants en puissance de calcul, batterie, stockage et bande passante, les applications et services *IoT* sont généralement soutenus par des serveurs puissants, principalement déployés dans le *cloud*. Cela est dû au fait que le *cloud computing* est considéré comme une solution prometteuse pour fournir des services aux utilisateurs finaux et des applications avec des ressources élastiques à faible coût.

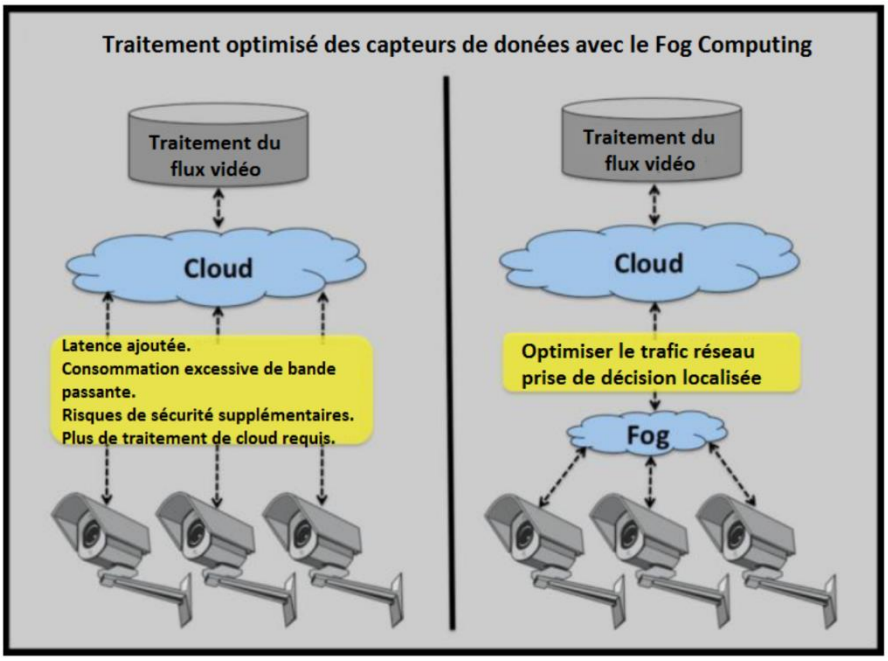


*Figure 18: Un exemple d'architecture Fog /Cloud.*

Dans ce travail, on présente les défis liés aux réseaux de caméras de surveillance actuels, basés sur le cloud, et les solutions proposées afin d'améliorer leur architecture en se basant sur un système fog/cloud. Aujourd'hui, les caméras de surveillance et de sécurité sont déployées dans le monde entier en nombre record afin d'assurer la sûreté et la sécurité des biens, des personnes et des lieux. Ces dispositifs de surveillance génèrent des quantités massives de données. Avec une seule caméra, nous pouvons générer plus d'un *Téraoctet* de données par jour. Les dispositifs de surveillance génèrent des données qui doivent être analysées en temps réel afin d'assurer la sécurité du public.

Comme le nombre de caméras et la quantité de données continuent de croître de façon exponentielle, de nouvelles approches sont nécessaires pour mieux gérer les appareils et les données. Les modèles traditionnels basés sur le *cloud* qui étaient déployés pour les caméras à basse résolution ne sont pas parfaitement adaptés à ces caméras haute définition, en raison de l'énorme quantité de données et des problèmes de latence et de disponibilité du réseau qui en découlent. Le *brouillard informatique* fournit l'architecture nécessaire pour créer des systèmes de surveillance distribués rentables, en temps réel et fournissant une latence intéressante, contribuant ainsi à améliorer les performances des systèmes distribués.

### 2.9.2 Les défis liés au cloud

- Le nuage (cloud) peut difficilement évoluer pour supporter une surveillance à grande échelle dans des applications telles que les bâtiments, les autoroutes, les villes, les gares routières, les aéroports, …etc.
- Des décisions de sécurité rapides doivent être prises sur place en temps réel ; Une latence faible réduit les risques pour la sécurité publique.
- Les caméras haute définition génèrent des téraoctets de données par jour, ce qui est susceptible de créer des goulots d'étranglement pour des systèmes composés d'un cloud seul.
- La vidéosurveillance nécessite une connectivité et une bande passante extrêmement fiables, ce qui peut être difficile dans des situations d'urgence.

### 2.9.3 Les solutions basées sur le fog

- Le Fog rajoute une certaine puissance de calcul dans les systèmes de vidéosurveillance
- Les nœuds Fog offre des relais lors de la transmission de données, notamment vers le Cloud, et contribuent ainsi à assurer une certaine sécurité et plus de déterminisme lors des transmissions.
- La nature potentiellement hétérogène du Fog rend possible le déploiement des tâches en logiciel et/ou en matériel.
- Le Fog peut assurer le stockage temporaire des données sans avoir à les transférer vers le cloud, assurant ainsi une certaine confidentialité
- Etant proches des capteurs vidéo (sources des données) les nœuds Fog peuvent garantir des latences intéressantes.

### 2.9.4 Les principales fonctionnalités du brouillard (Fog)

Le *brouillard informatique* est une architecture permettant de rapprocher les caméras intelligentes des serveurs cloud et d'obtenir de faibles latences. Il fournit un maillage de nœuds de brouillard pour répartir intelligemment le traitement vidéo entre les autres nœuds de brouillard colocalisés avec les caméras et le nuage afin de permettre le suivi en temps réel et la détection d'anomalies à partir de données collectées sur de longs intervalles. Les algorithmes d'analyse vidéo peuvent être exécutés sur des nœuds de brouillard proches des caméras, et bénéficient d'une capacité de traitement hétérogène du brouillard, en exécutant des parties de l'algorithme d'analyse vidéo sur des processeurs rapides (*GPU, Accélérateurs, ... etc.*).

## 2.10 FPGA dans les architectures de type Fog/Cloud

### 2.10.1 FPGA dans les architectures de type Cloud

Les centres de données (*data center*) à grande échelle ont fait d'énormes progrès en matière de gestion de réseau, de virtualisation, d'efficacité énergétique et de gestion des infrastructures. Toutefois, ils ont conservé une structure de base identique aux années précédentes : des serveurs individuels avec des processeurs multi cœurs, des DRAM et du stockage local via des commutateurs Ethernet vers d'autres serveurs. A grande échelle *(des centaines à des milliers de serveurs)*, il y a des avantages significatifs à favoriser l'homogénéité, ce qui réduit les coûts et les erreurs de configuration. Le caractère très dynamique des applications modernes, notamment dans le domaine du traitement vidéo combiné à l'évolution rapide des standards de traitement et de communications engendre un important besoin à des ressources de calcul puissantes pouvant garantir des contraintes de temps réel sur des traitement de plus en plus lourds. Parmi les architectures actuelles, les FPGAs et les GPUs semblent être les plus à mème à répondre à de tels besoins. En effet, leur elles sont, de part leur nature même, tout à fait adaptées aux traitement massivement parallèle, tout en offrant une importante flexibilité se traduisant par la reconfigurabilité des FPGAs et la programmabilité relativement simple des GPUs.

Les GPU et les FPGA sont déployés dans des infrastructures de type data center et semblent fournir de bonnes performances, tel que décrit dans [MS15] [JB13] [OLQWYJ14]. Des chercheurs fournissent des solutions pour un déploiement FPGA à moyen terme dans des centres de données commerciaux pour accélérer l'indexation de la recherche sur le *Web* en utilisant plusieurs accélérateurs [PCC14]. Cette conception consistait en une structure en rack de *48 FPGA* reliés à un réseau secondaire. Bien qu'efficace pour accélérer l'indexation des recherches, cette architecture présentait plusieurs limites importantes :

- Le réseau secondaire (un tore 6x8) nécessitait un câblage coûteux et complexe.
- La gestion des défaillances du tore nécessitait un réacheminement complexe du trafic vers les nœuds voisins, ce qui entraînait à la fois une perte de performance et l'isolement des nœuds sous certaines configurations de défaillance.
- Le nombre de FPGA pouvant communiquer directement, sans passer par un processeur, était limité à un seul rack (c'est-à-dire 48 nœuds).

Pour lutter contre ces défis, une nouvelle architecture d'accélération basée sur FPGA à l'échelle du cloud a été proposée par ces chercheurs : le *Cloud configurable*. Ce dernier élimine toutes

les limitations listées ci-dessus avec un seul design. Cette architecture a été déployée dans la majorité des nouveaux serveurs des centres de données de production de *Microsoft* dans plus de *15 pays sur 5 continents*. Il s'agit d'un Cloud configurable composé de plusieurs FPGAs (multi-FPGA) pouvant s'adapter à tout moment au contexte et aux besoin des applications à exécuter.

La principale différence par rapport aux travaux précités est que le matériel d'accélération est étroitement couplé au réseau du centre de données en plaçant une couche de FPGA entre les cartes réseau des serveurs et les commutateurs réseau Ethernet. *La figure 19* montre comment l'accélérateur s'adapte à un serveur hôte. Tout le trafic réseau est acheminé via le FPGA, ce qui lui permet d'accélérer les flux réseau à haut débit. Une connexion *PCIe* indépendante aux CPU hôtes est également fournie, permettant au FPGA d'être utilisé comme un accélérateur de calcul local. Le commutateur et la topologie réseau standard suppriment l'impact des défaillances sur

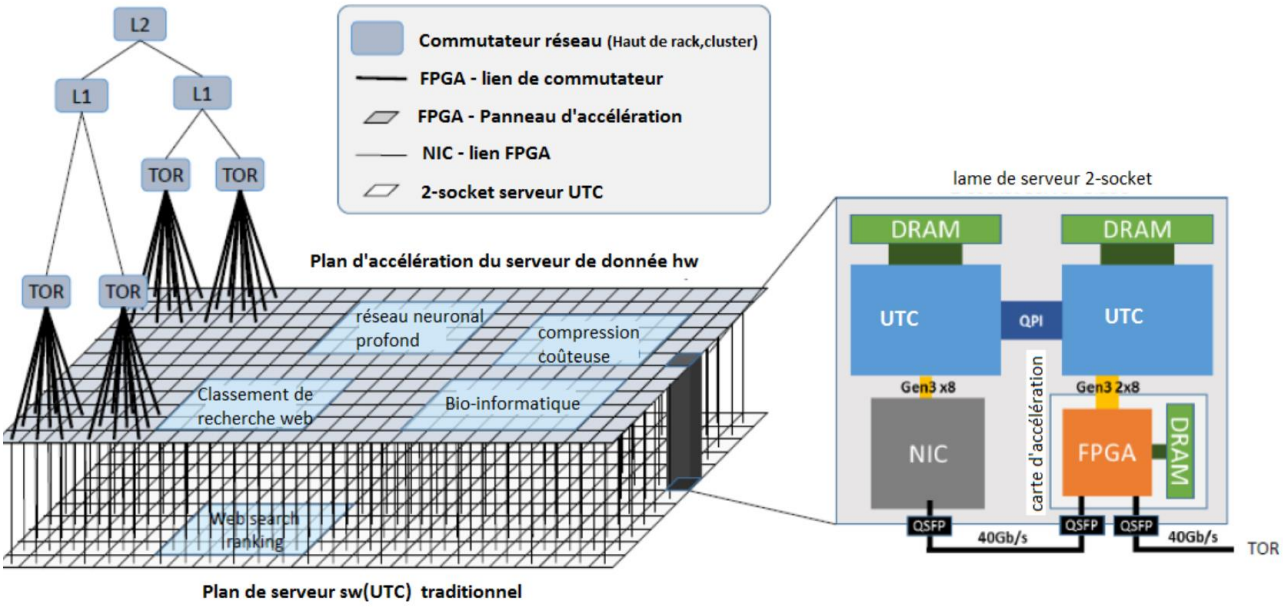


*Figure 19: (a) Plan de matériel programmable découplé, (b) Schéma serveur + FPGA.*

les serveurs voisins, le besoin de câblage non standard.

### 2.10.2 Système des caméras intelligentes basé sur Fog/Cloud

Le concept de «*brouillard informatique*» est proposé comme une solution pouvant améliorer les performances de l'infrastructure *IoT-Cloud*. Ce nouveau concept *FC-IoT (Fog-Cloud-IoT)* est défini comme une infrastructure distribuée dans laquelle une partie du service est gérée et traitée par des composants intelligents du réseau. Cette approche décentralisée (*distribuée*)

permet grâce à ces composants (*intermédiaire entre le cloud et les utilisateurs finaux*) de soulager le réseau d'accès et réduire la latence des échanges de ces applications.

Un des défis les plus importants et modernes est donc la façon de faire évoluer un système de vidéosurveillance basé sur le nuage existant avec plusieurs caméras hétérogènes intelligentes et l'adapter à une architecture *brouillard/nuage* afin d'améliorer les performances. En fait, les systèmes actuels de vidéosurveillance contiennent un certain nombre de caméras intelligentes ayant des puissances de calcul différentes et des quantités d'énergie disponibles différentes.

L'objectif est de proposer une nouvelle approche permettant une assignation dynamique, optimale et automatique des tâches de suivi *Hardware/Software* sur les éléments de traitement d'un système de vidéosurveillance *brouillard/nuage*.

*Le Fog Computing* est une couche applicative et matérielle complémentaire aux solutions cloud existantes. L'architecture de *Fog computing* permet de répondre aux exigences des applications nécessitant un traitement en temps réel ou un traitement massif des données collectées par les objets connectés.

Cette architecture est une infrastructure décentralisée dans laquelle les données, le calcul, le stockage et les applications sont placés efficacement entre la source de données (*capteurs*) et le *cloud*. Afin d'optimiser les temps de réponse des applications *IoT*, *le Fog computing* étend les services de *cloud computing* à la périphérie du réseau pour le traitement des données collectées. Ces équipements de traitement décentralisé peuvent être des routeurs ou des passerelles dotés d'une certaine puissance de calcul. En réduisant le chemin entre les objets connectés et les équipements de traitement de données collectés, le *Fog computing* apporte une capacité supplémentaire au Data Center, allège la charge sur le réseau et améliore la latence des échanges.

## 2.11 Tolérance aux fautes dans les systèmes de type fog /cloud

Le *Cloud Computing* pour les objets connectés est une plate-forme émergente et innovante, qui met l'informatique et le stockage à la disposition des utilisateurs finaux en tant que services. Le cloud computing a été de plus en plus utilisé pour un large éventail d'applications dans divers domaines de recherche, tels que les réseaux intelligents, le commerce électronique, et les applications scientifiques.

### 2.11.1 La tolérance aux fautes dans les Clouds

Le *Cloud Computing* et la virtualisation ont ouvert une nouvelle fenêtre dans la gestion des pannes. La mise en pause, la reprise, et la migration des machines virtuelles (*VMs*) sont des mécanismes puissants pour gérer les défaillances dans un tel environnement. Une machine virtuelle peut être facilement migrée vers un autre nœud lorsqu'une panne est détectée.

Pour assurer la fiabilité et la gestion de défaillances et des pannes, les fournisseurs de services *Cloud* adoptent divers mécanismes pour mettre en œuvre la tolérance aux pannes au niveau du système. La tolérance aux pannes est un problème crucial dans les plates-formes et les applications de cloud computing. Elle permet à un système de continuer à fonctionner, éventuellement à un niveau réduit, lorsque certains sous-composants du système présentent des dysfonctionnements inattendus. En raison de la diversité de la nature et des besoins des différents services déployés dans un environnement cloud, la tolérance aux pannes joue également un rôle essentiel dans le maintien des niveaux de service (*SLA*) et des niveaux de qualité de service ($Q_oS$) souhaités [ZMM10]. Les *SLA* définissent différentes règles pour réguler la disponibilité du service cloud pour les utilisateurs finaux. La virtualisation dans le cloud fournit divers services à différents niveaux d'accès à de nombreux abonnés. La virtualisation, les multiples contrats de type *SLA* et diversification des exigences de $Q_oS$ augmentent considérablement la complexité des environnements cloud.

#### 2.11.1.1 Types de défauts

Les défauts peuvent être de divers types, notamment :

- Défauts matériels transitoires, intermittents ou permanents.
- Bogues logiciels et erreurs de conception.
- Erreurs de l'opérateur.
- Défauts et erreurs induits de l'extérieur.

Dans un environnement de *cloud* classique, les défaillances apparaissent comme une défaillance des ressources, qui sont utilisées par les utilisateurs finaux. Les deux défauts les plus courants dans l'environnement *cloud* sont:

##### 2.11.1.1.1 Les échecs byzantins

Dans ce type de défauts, les composants du système échouent de manière arbitraire, ce qui entraîne un comportement imprévisible du système. Le système peut traiter les demandes de manière incorrecte et produire des sorties incohérentes.

#### 2.11.1.1.2 Les échecs d'écrasement

Lorsque les pannes surviennent, les composants du système cessent complètement de fonctionner ou restent inactifs pendant les défaillances. Par exemple, les pannes dues à des coupures de courant ou à un crash de disque dur.

### 2.11.1.2 La redondance

Une approche pratique de la mise en œuvre de la tolérance aux pannes est la redondance qui implique la duplication des composants matériels et logiciels, de sorte qu'en cas de défaillance d'un composant ou d'un processus, le processus ou le composant de sauvegarde soit disponible. Certains des fournisseurs ont été impliqués dans le développement de solutions informatiques avec une capacité intégrée de tolérance aux pannes. Par exemple, les *ordinateurs Stratus* sont à auto-vérification duplex, c'est-à-dire que chaque ordinateur est dupliqué en interne et s'exécute de manière synchrone. Si l'une des machines tombe en panne, la duplication permet à l'autre machine de la paire de poursuivre les calculs sans délai. De même, les *ordinateurs Tandem* utilisent un certain nombre de processeurs identiques indépendants ainsi que des dispositifs de stockage et des contrôleurs redondants pour assurer une récupération automatique en cas de panne matérielle ou logicielle [HCG10].

### 2.11.1.3 La validation de tolérance aux pannes

Les fournisseurs de services de *cloud* doivent effectuer l'analyse de tolérance aux pannes/disponibilité des services qu'ils fournissent aux utilisateurs finaux. La validation de tolérance de panne d'un service est d'une importance cruciale pour s'assurer que les *SLA* sont correctement respectés. Cependant, en raison de nombreux facteurs impliqués, il est assez difficile de vérifier qu'une machine tolérante aux pannes répondra aux exigences de fiabilité. Pour faciliter l'évaluation de la fiabilité du système, de nombreuses recherches ont récemment été effectuées sur des essais expérimentaux en utilisant une méthodologie connue sous le nom *d'injection de fautes.* L'injection de fautes est une méthode importante pour imiter l'apparition d'erreurs dans un contrôle d'environnement pour prendre les mesures nécessaires. Un certain nombre de modèles stochastiques basés sur des calculs de probabilité ont été développés, ils utilisent des processus de *Markov* et de *semi-Markov* pour effectuer l'analyse de disponibilité d'une machine tolérante aux pannes. Ces modèles ont été implémentés dans plusieurs outils de conception assistée par ordinateur [KK10].

#### 2.11.1.4 Mesures de tolérance aux pannes

La mesure de la tolérance aux pannes est un aspect crucial du paradigme du *nuage*. Des mesures de tolérance aux pannes peuvent être utilisées pour quantifier la fiabilité du système de nuages. Deux des principales mesures existantes pour la tolérance aux pannes du système sont : la disponibilité et la fiabilité [KK10]. La disponibilité est le rapport entre le temps de marche et la somme des temps de disponibilité et d'arrêt du système. La disponibilité peut être quantifiée comme :

$$\text{Disponibilité} = \frac{temps\ de\ marche}{temps\ de\ marche + temps\ d'arrêt} \quad (1)$$

La fiabilité, notée *R(t)*, est la probabilité que le système fonctionne correctement en fonction du temps *'t'* [KK10]. Le manuel de mesure du *TL9000* définit la fiabilité comme «*la capacité d'un élément à exécuter une fonction requise dans des conditions définies pendant une période donnée*» [BA12]. La fiabilité du service peut être formulée comme suit :

$$Fiabilité = \frac{nombre\ de\ réponses\ correctes}{nombre\ total\ de\ requête} \times 100 \quad (2)$$

La disponibilité et la fiabilité du système jouent un rôle central dans le paradigme du *cloud*. Les temps d'arrêts ont de graves répercussions financières. Il a été rapporté qu'une seule heure d'arrêt coûte environ *50 000 dollars* dans un centre de données [BMKZ14]. Par conséquent, la disponibilité des services de cloud est indispensable.

### 2.11.2 Stratégies tolérantes aux fautes dans le cloud

Les pannes sont très courantes dans le cloud, telles que les pannes de disque ou les échecs de liaison. De plus, le nombre de nœuds (*serveurs*) impliqués dans le nuage est de l'ordre de dizaines de milliers ou plus de serveurs ce qui augmente la probabilité et le coût des défaillances. Les travaux s'exécutant sur le cloud peuvent être relativement longs. L'effet de l'échec sur des utilisations à moyen ou long terme peut compromettre la réalisation du contrat *SLA* et entraîner le gaspillage de temps de calcul. Considérons l'exemple d'une tâche qui nécessite 24 heures d'exécution, si après 23 heures le nœud qui exécutait la tâche tombe en panne, alors presque un jour d'exécution sera gaspillé et cela conduira également à la violation du *SLA*. Une façon de résoudre ce problème consiste à exécuter un processus de sauvegarde sur un système différent pour chaque processus principal. Le processus principal dans ce cas est responsable du point de contrôle de l'état actuel des disques redondants. Si le processus

principal échoue, le processus de sauvegarde peut redémarrer à partir du dernier point de contrôle. En règle générale, les systèmes tolérants aux pannes se caractérisent par la capacité de récupération à partir des erreurs commises en amont. Le mécanisme d'avance prend l'état du système au moment où l'erreur a été détectée afin de la corriger, et à partir de là, le système avance. En variante, le mécanisme de restauration utilise le point de reprise pour ramener l'état du système à un état correct antérieur afin de l'avancer. Les opérations entre le point de contrôle et l'état erroné peuvent être rendues inefficaces, conformément à l'exigence de récupération par restauration. Certains systèmes font appel à la récupération aval et à la restauration *rollback* pour différentes erreurs ou différentes parties d'une même erreur.

## 2.12 Analyse et discussion

Les travaux réalisés dans le cadre de cette thèse ciblent un domaine à la croisée de plusieurs disciplines et mettant en œuvre un très large spectre de compétences. L'état et de la taille de l'art est donc primordial, d'où l'intérêt accordé à ce chapitre. Ainsi, il était question dans ce chapitre de regrouper à la fois les définitions de l'ensemble des concepts nécessaires à la compréhension des contributions de cette thèse mais aussi de présenter l'état actuel de la littérature sur les problématiques similaires ou comparables. Cet état de l'art, en plus notamment des priorités de Dcarte Engineering ont guidé les choix et les méthodes proposés dans les chapitres suivants de ce manuscrit.

# 3 ARCHITECTURE DISTRIBUEE POUR UNE VIDEOSURVEILLANCE ADAPTATIVE DE TYPE FOG/CLOUD

L'architecture globale ainsi que le fonctionnement du système et de l'applicatif auquel on s'intéresse dans cette thèse sont présentés dans ce chapitre. Il s'agit, comme détaillé dans les sections qui suivent, d'un ensemble de caméras intelligentes disposant chacune d'une certaine capacité de calcul et d'une énergie limitée. Ces caméras sont disposées dans un système hiérarchique de type Fog/Cloud.

## 3.1 Architecture du système de vidéosurveillance

Le système de vidéosurveillance est basé sur une architecture mixte Cloud/Fog computing : il est composé d'un ensemble de caméras intelligentes (voir Figure 20) ; certaines d'entre elles contiennent un FPGA dynamiquement reconfigurable, d'autres ont simplement un processeur et un GPU. Ces caméras communiquent avec les nœuds Fog. Un nœud Fog relie un certain nombre de caméras en fonction de la stratégie de placement des caméras et de la zone couverte par le système de surveillance. Les nœuds Fog peuvent communiquer entre eux (pour la négociation et la notification) et avec le serveur Cloud qui est composé de plusieurs FPGA de grande taille possédant une grande capacité et puissance de calcul.

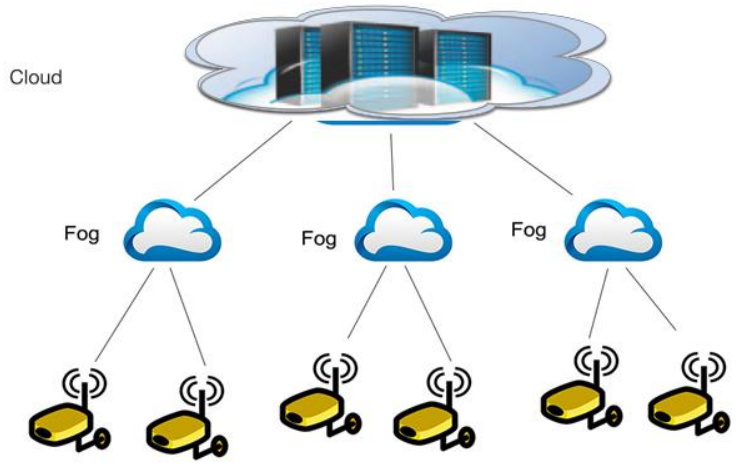


*Figure 20: Architecture générale du système*

Une caméra intelligente est une caméra dans laquelle est ajoutée une composante électronique permettant à la fois d'acquérir et de stocker des images, mais également de traiter de l'information et de communiquer avec les systèmes environnants (réseau, automates, opérateurs...). Ces systèmes sont construits autour d'un microprocesseur. Celui-ci, associé à la mémoire, remplit les mêmes fonctions qu'un ordinateur, mais en plus compact. Les caméras intelligentes sont caractérisées par : le type de capteur, les performances du processeur, la taille de la mémoire, les interfaces d'E/S, les possibilités d'affichage ainsi que le(les) logiciel(s) disponible(s).

## 3.2 Fonctionnement du système

Lorsqu'une nouvelle cible (par exemple, un objet ou une personne devant être suivie) arrive dans le système, toute caméra qui la détecte informe son Fog en envoyant un message. Les nœuds Fogs envoient des messages de notification au Cloud contenant l'identification de nouvelles cibles. Après négociation entre les Fogs, l'un d'entre eux exécute un algorithme (d'allocation de tâches) permettant l'affectation des nouvelles cibles sur les dispositifs de calcul. Dans cette application de suivi, le processus de suivi est exécuté sur les caméras, sur les nœuds Fog ou, en dernier recours, sur le Cloud.

### 3.2.1 Au niveau du Fog

- Les nœuds Fogs partitionnent intelligemment les tâches de tracking entre les caméras, le Fog et le Cloud (le cas échéant).
- Au niveau du Fog (et des caméras), les tâches de tracking sont implémentées dans des processeurs conventionnels ou des accélérateurs (FPGA).
- Les nœuds Fogs reçoivent des flux de données des caméras, en temps réel.
- Les nœuds Fogs exécutent les applications IoT en temps réel avec un temps de réponse très rapide (en millisecondes).
- Les nœuds Fogs fournissent un stockage et un transport sécurisés des données.
- Les nœuds Fogs peuvent déléguer une tâche de tracking au Cloud lorsque cette tâche ne peut être effectuée par aucune caméra ni aucun nœud Fog.

### 3.2.2 Au niveau du Cloud

Le serveur Cloud :

- Prend en charge certaines tâches de tracking.
- Assure un stockage et un transport sécurisés des données.
- Communique avec les nœuds Fog.

## 3.3 Contraintes du système

Le système de vidéosurveillance fonctionne sous un ensemble de contraintes qui peuvent être exprimées comme suit :

- Chaque élément de traitement (caméra, Fog, Cloud) a une quantité d'énergie limitée. Évidemment, cette énergie disponible peut être significativement différente d'un

élément à l'autre selon que l'élément soit connecté à une source d'alimentation ou qu'il soit utilisé sur batterie. L'exécution d'une tâche de suivi consomme une certaine quantité d'énergie, donc la consommation correspondant à toutes les tâches exécutées par un élément de traitement doit être inférieure à la quantité disponible.

- Sur chaque caméra et sur les nœuds Fog, le nombre de ressources est limité : les ressources sont assimilées à des zones sur les FPGA et chaque zone implémente une seule tâche.
- Les ressources au niveau du Cloud sont « théoriquement illimitées ».

### 3.3.1 Architecture du système

Le domaine d'application que nous ciblons dans ce travail est la vidéosurveillance distribuée utilisant des caméras intelligentes hétérogènes. Lorsqu'une cible arrive dans le système, une tâche de suivi $T_k$ est générée, un sous-ensemble de caméras peut effectuer la tâche $T_k$. Ainsi, la tâche doit être affectée à une caméra $C_j$ choisie et on dira que la caméra $C_j$est allouée à la tâche $T_k$. La figure 21 illustre un système composé de trois caméras et six objets à suivre. Le système comporte également un obstacle, schématisé par un trait, et correspondant à tout objet pouvant cacher une cible par rapport à une caméra.

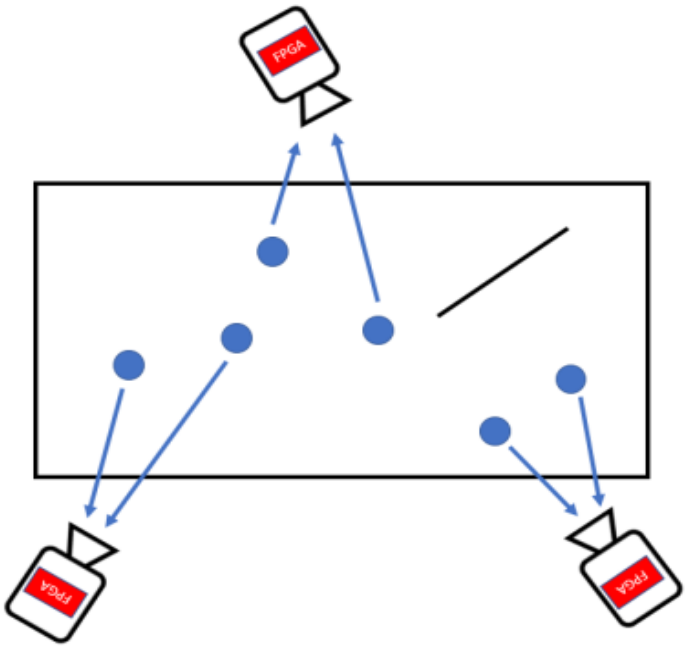


*Figure 21: Système distribué avec trois caméras et six cibles.*

Ce système est très dynamique car les cibles peuvent bouger et de nouvelles cibles peuvent apparaître à tout moment. La figure 22 montre le cas où une cible se déplace derrière un obstacle et quitte le champ de vision de la caméra qui la suivait. Il devient donc nécessaire de choisir une nouvelle caméra qui contient cette cible dans son champ de vision, afin d'y implémenter la tâche de suivi correspondante. Dans l'exemple suivant, il y a un choix entre deux caméras pour l'affectation de la cible.

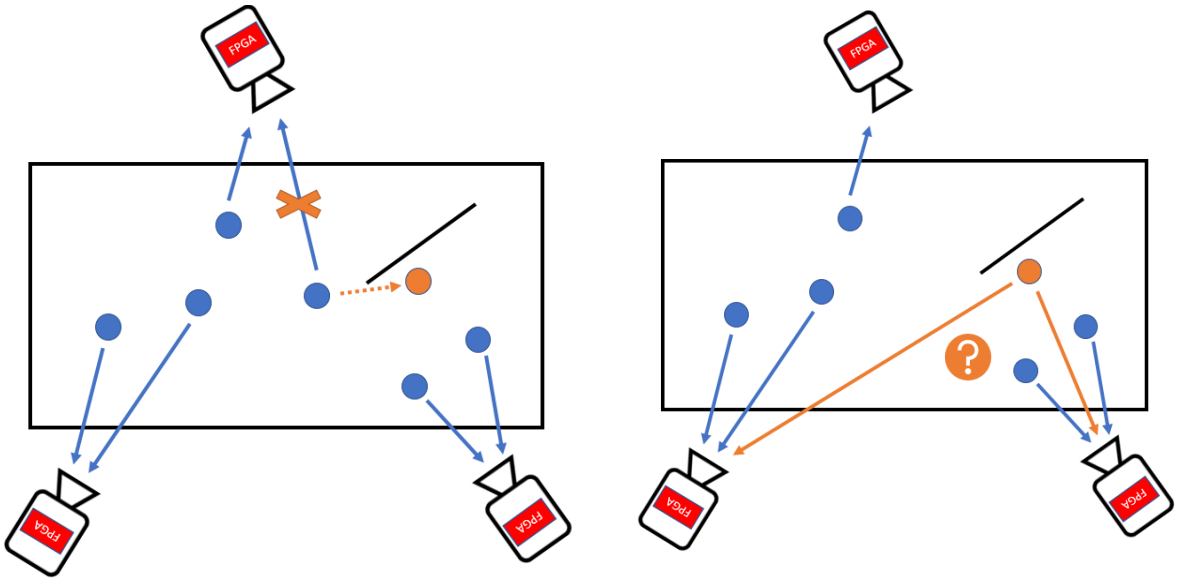


*Figure 22: Affectation dynamique de la tâche de tracking*

Un autre exemple illustrant la dynamicité du système est exposé dans la figure 23. Il s'agit du cas où une caméra tombe en panne et ne peut donc plus assurer le suivi des cibles qui lui ont été affectées. Ces dernières doivent alors être dynamiquement réaffectées à des caméras disponibles et opérationnelles.

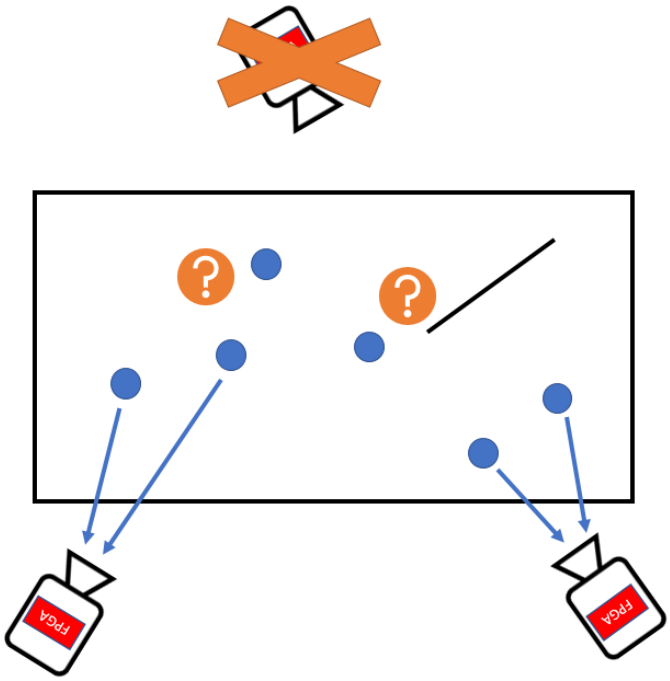


*Figure 23: Panne de l'une des caméras du système*

## 3.4 Discussion et conclusion

Le système ciblé dans ce travail présente la particularité d'être dynamique à la fois au niveau matériel et au niveau fonctionnel (applicatif). Il s'agit d'un système évolutif pouvant s'adapter à divers environnements et à d'éventuels changements dans le système. Cependant, cette dynamicité introduit une complexité supplémentaire liée notamment à la prise en compte du nombre de caméras et/ou des cibles à suivre pendant l'exécution des changements. Ces deux problématiques constituent l'objet de deux chapitres suivants.

# 4 SYSTEME DE VIDEOSURVEILLANCE DYNAMIQUE ET A CHARGE EQUILIBREE BASE SUR UNE ARCHITECTURE FOG/CLOUD

Ce chapitre propose en premier lieu une nouvelle approche d'affectation des caméras dans un réseau de vidéosurveillance distribué [SMAW18]. Plus précisément, il s'agit de comprendre comment une caméra peut se connecter à un FOG via un réseau d'interconnexions tout en respectant l'équilibrage de charge. Le service SDN (Software Defined Networking) offre une approche rentable et flexible pour implémenter un système de connexion et d'équilibrage de charge. Tout en évitant les traditionnelles approches matérielles de mise en réseau, nous proposons dans ce projet la mise en œuvre d'équilibrage de charge à l'aide de logiciels. La nouvelle approche SDN réduit le coût, offre une flexibilité de configuration, réduit le temps de déploiement, fournit l'automatisation de la connexion entre les caméras et les FOG et facilite la construction d'un réseau sans nécessiter la connaissance de tous les fournisseurs spécifiques de logiciel et de matériel informatique.

## 4.1 Introduction

L'idée des SDNs (Software-defined networking) est apparue peu de temps après la publication de Java dans 1995, et a reçu une grande attention en tant que solution permettant de simplifier la gestion de l'Internet depuis 1998 [BMX14]. Le concept principal derrière SDN est de séparer la logique de contrôle de l'équipement de réseau pour contrôler la couche physique. De cette manière, les infrastructures sous-jacentes sont simplifiées, et elles ne suivent que les instructions du contrôleur via l'interface en direction sud (par exemple, OpenFlow [NTH08], etc.). L'avantage de cette structure de réseau en couches est évident. Premièrement, une topologie de réseau peut être recueillie au contrôleur pour la prise de décision, telles que les décisions de routage, moniteur de trafic réseau, et préjugement. Deuxièmement, le contrôleur a une vue globale sur les applications utilisées et les protocoles à exécuter, ce qui permet de gérer plus facilement les interactions. Troisièmement, comme les détails physiques des dispositifs sont abstraits, les composants deviennent plus facilement interchangeables et le réseau plus flexible . De nouvelles fonctions logicielles peuvent être ajoutées au lieu de modifier chaque commutateur. Par conséquent, les entreprises et les opérateurs bénéficient d'une programmation, d'une automatisation et d'un contrôle de réseau flexibles sans précédent, ce qui leur permet de construire des réseaux flexibles et évolutifs qui s'adaptent facilement aux besoins changeants des entreprises.

## 4.2 Les capteurs sans fil intelligents

Les progrès conjoints de la micro-électronique, microtechnique, des technologies de transmission sans fil et des applications logicielles ont permis de produire à coût raisonnable des micro-capteurs de quelques millimètres cubes de volume, susceptibles de fonctionner en réseaux. Ils intègrent une unité de captage chargée de capter des grandeurs physiques (chaleur, humidité, vibrations, rayonnement…) et de les transformer en grandeurs numériques ; une unité de traitement informatique et de stockage de données ; et un module de transmission sans fil (wireless).

Ces micro-capteurs sont donc de véritables systèmes embarqués. Le déploiement de plusieurs d'entre eux, en vue de collecter et transmettre des données environnementales vers un ou plusieurs points de collecte, d'une manière autonome, forme un réseau de capteurs sans fil (*Wireless Sensor Network* ou WSN).

Les défis au niveau des nœuds d'objets intelligents concernent la petite quantité de ressources disponibles, tandis que les défis au niveau du réseau concernent la grande échelle des réseaux d'objets intelligents. Même s'il existe de nombreux exemples de réseaux d'objets intelligents à petite échelle, de nombreux réseaux peuvent potentiellement être très grands - de l'ordre de milliers de nœuds. Chacun de ces capteurs intelligents génère des millions d'informations à transmettre (dans notre cas il s'agit des images acquises). Les données captées par les nœuds sont acheminées grâce à un *routage multi-saut* à un nœud considéré comme un "point de collecte", appelé nœud-puits (ou *sink*). Ce dernier peut être connecté à l'utilisateur du réseau (via Internet, un satellite ou un autre système). L'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises et récolter les données environnementales captées par le biais du nœud-puits.

## 4.3 Les caméras intelligentes

Une caméra intelligente est définie comme un système de vision dans lequel la fonction fondamentale est la production d'une analyse de haut niveau de la scène acquise. Une caméra est appelée « caméra intelligente » lorsqu'elle exécute un traitement d'informations spécifiques à l'application (figure 24). La sortie de ces caméras est soit les caractéristiques extraites des images capturées, soit une description de haut niveau de la scène. Plus de détails sur ce système peuvent être trouvés dans les articles de Wolf [WW09] et Shi et Lichman [SL].

Le Tableau 4 présente un aperçu des plates-formes de caméras intelligentes les plus courantes trouvées dans la littérature. D'autres travaux ont été basés sur des caméras intelligentes intégrées au silicium. Dans ces systèmes, les auteurs proposent une approche dans laquelle la détection d'images et le traitement sont intégrés sur une seule puce de silicium. Ce type d'appareil est appelé « puce de vision ». Le lecteur intéressé peut trouver une description détaillée dans [AM00]. Parmi ces œuvres, le projet phare de P. Dudeck est l'un des plus connu [CBD]. D'autres contributions dans le même axe peuvent être trouvées dans [MB99, ACCPSV02].

*Tableau 4 : Caméras intelligentes dotés de capacités de calcul*

| System | Platform capabilities | | | Application |
|---|---|---|---|---|
| Author (s) | Sensor | CPU | Power | |
| **CMUcam** | CMOS Omnivision | Proc. ARM7 | Battery | Robotic applications |
| **MeshEye** | ADNS-3060 optical mouse sensor + CMOS VGA | Micro-controller AT91SAM7S | Battery | Distributed imaging applications |
| **SeeMOS** | CMOS Cypress Lupa 4000 | FPGA Stratix 60 | Mains | Tracking |
| **LE2I-Cam** | CMOS Micron (MTM9M413) | FPGA Virtex II | Mains | Hight speed imaging |
| WiCa mote | VGA CMOS | Xetal IC3D | Battery | Vehicle detection and speed estimation |
| ITI | LM-9618 CMOS | DSP TMS320C6415 | Mains | Traffic control |

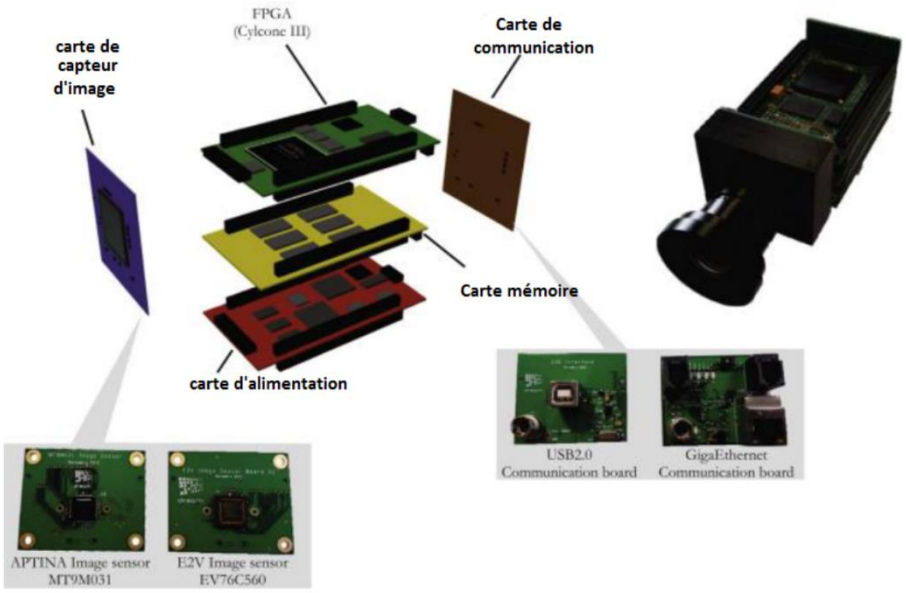


*Figure 24: Exemple d'architecture de caméra intelligente*

## 4.4 La connexion des caméras aux Fogs

Une caméra doit être automatiquement connectée à un Fog pour lui transmettre le flux vidéo ou le résultat du prétraitement réalisé par cette dernière. La connexion entre les caméras et les Fogs doit être faite d'une manière judicieuse afin d'assurer l'équilibrage de charge et une bonne transmission. La plupart des travaux réalisés dans ce domaine utilisent un contrôleur. Cependant certains contrôleurs, comme ceux proposés par Cisco ne sont pas très flexibles du point de vue de la programmation. Nous proposons dans ce travail un algorithme d'équilibrage de charge implémenté dans un contrôleur sous un service SDN. Avant de détailler l'algorithme proposé nous décrivons brièvement le service SDN et les contrôleurs.

### 4.4.1 De l'authentification à la communication dans les réseaux de caméras

Le schéma de transfert d'AP de l'IEEE 802.11 est défini en trois étapes : (a) balayer les nouveaux AP pour l'association, (b) envoyer la sonde d'authentification à l'AP cible qui a un RSSI fort, et (c) commencer la procédure de réassociation pour se connecter avec le nouveau AP. Par conséquent, ce protocole est tout à fait adapté à l'authentification des caméras intelligentes dans un réseau de caméras de type fog, néanmoins l'hétérogénéité des composants de l'infrastructure

peut rendre la communication difficile et fastidieuse en l'absence d'un protocole capable d'abstraire les détails physiques.

Récemment, le protocole OpenFlow a été sollicité dans le réseau sans fil pour fournir le contrôle de paquets à granularité fine, et SDN fait la séparation de l'infrastructure physique sous-jacente et les services de réseau .

### 4.4.2 Pourquoi utiliser le service SDN

Le service SDN est approche développée pour faciliter la gestion de réseau ayant attiré une attention générale. Le déploiement du réseau traditionnel consiste à acheter le matériel et le logiciel ensemble auprès d'un fournisseur et à les configurer conformément aux exigences. Traditionnellement en réseau, il n'est pas possible de séparer le plan de données du plan de contrôle. Par conséquent, le concepteur a dû compter sur le fournisseur pour des corrections de problèmes, des fonctionnalités supplémentaires et des licences. Il n'est pas possible pour le client de modifier le code du logiciel du fournisseur en fonction des besoins du réseau. La nature exclusive de la plupart des fournisseurs des logiciels entraîne un ralentissement de l'innovation et le retard accru dans le déploiement de nouveaux services. SDN tente d'éliminer cette dépendance du fournisseur et constitue un changement de paradigme, passant d'une mise en réseau fermé spécifique à un fournisseur à un système de mise en réseau ouvert à l'hétérogénéité. Dans SDN, un contrôleur centralisé déploie les flux réseau dans les commutateurs réseau de façon transparente indépendamment des fournisseurs.

Par conséquent, le matériel pour les périphériques réseau est disponible à un prix inférieur et la configuration du logiciel peut être modifiée selon les besoins des clients. Ce matériel générique peut être programmé à l'aide d'une interface programmable (API) ou de l'installation d'un système d'exploitation réseau (NOS). Parmi les protocoles utilisés pour la configuration réseau nous pouvons retrouver OpenFlow et NETCONF. OpenFlow permet au contrôleur SDN de communiquer avec des périphériques réseau. De même, le protocole NETCONF permet la configuration et l'automatisation du réseau en utilisant Yet Another Next Generation (YANG) comme langage de modélisation de données. Le dégroupage du matériel et des logiciels réduit considérablement les coûts à mesure que les réseaux se développent. Grâce à l'utilisation de ce dernier, le client peut programmer les flux selon les exigences du réseau. Le projet vise à implémenter un équilibreur de charge pour installer les flux dans un périphérique réseau (Fog) à partir du contrôleur. En utilisant le SDN, le coût d'un équilibreur de charge est réduit, la

dépendance du fournisseur est éliminée et la flexibilité en termes de modification du code est réalisée.

### 4.4.3 Architecture du SDN

Open Network Foundation (ONF) et le groupe de recherche sur les réseaux informatiques (Software-Defined Networking Research Group) ont étudié SDN sous différents angles. Open Network Foundation est une organisation à but non lucratif fondée par Google, Microsoft, Yahoo et certains opérateurs de télécommunications en mars 2011, qui vise à développer la technologie, la normalisation et le marketing liés à SDN. Dans la référence [ONF], la structure et la définition de SDN sont fournies comme suit: «Dans l'architecture SDN, les plans de contrôle et de données sont découplés, l'intelligence et l'état du réseau sont logiquement centralisés et l'infrastructure réseau sous-jacente est découplée des applications.» La figure 25 illustre également l'architecture des réseaux définie par logiciel. Il est clair que SDN est une structure qui sépare le plan de contrôle (ou le système d'exploitation réseau) du plan d'acheminement, qui était précédemment intégré verticalement dans les dispositifs traditionnels.

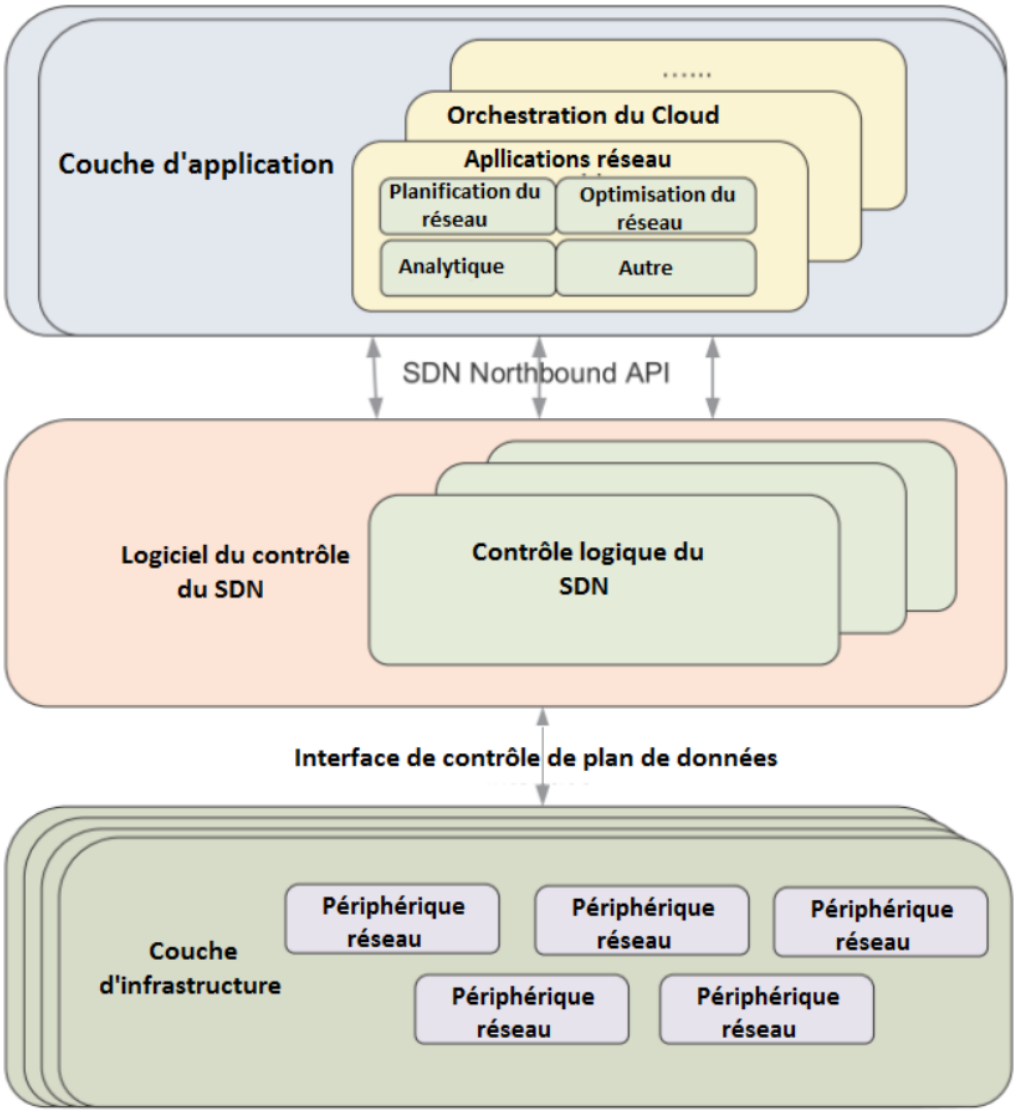


*Figure 25: Architecture du SDN*

Dans SDN, le contrôleur observe et rassemble l'état du réseau entier à partir d'un point de vue central, hébergeant des fonctionnalités telles que les protocoles de routage, le contrôle d'accès, la virtualisation de réseau, la gestion de l'énergie, etc. Comme le contrôleur est un programme fonctionnant sur un serveur, il peut être situé n'importe où et communiquer avec les périphériques réseau via des protocoles orientés vers le sud tels que le protocole OpenFlow via un canal sécurisé. Avec le contrôle centralisé, les informations d'état de commutation peuvent être collectées pour former une topologie de réseau globale et assurer ainsi une gestion en temps réel et une politique cohérente. Il y a eu un travail continu sur le développement des contrôleurs SDN, et les principaux exemples sont NOX, ONIX, Beacon, Maestro, Floodlight etc.

## 4.5 Les contrôleurs réseaux

### 4.5.1 Attachement à un contrôleur

Le point d'accès a besoin d'un contrôleur pour fonctionner. Pour le trouver, il suit une démarche en deux étapes :

- Etablissement d'une liste de contrôleurs candidats (obtention de leurs adresses IP)
- Sélection parmi cette liste.

L'établissement de la liste des candidats peut être réalisé à travers différentes étapes. Celles-ci sont exécutées l'une après l'autre pendant le processus initial de découverte :

- découverte par *broadcast* sur le sous-réseau de rattachement (format de trame spécial que le contrôleur, s'il est localisé sur le même sous-réseau, reconnaîtra et auquel il va répondre) ;
- utilisation d'extensions DHCP spécifiques (champs *TLV* constructeur) ;
- enregistrement DNS ;
- configuration statique dans la mémoire du point d'accès.

Un système intégré d'équilibrage de charge permet au point d'accès de choisir le contrôleur le moins chargé lors de la phase d'initialisation.

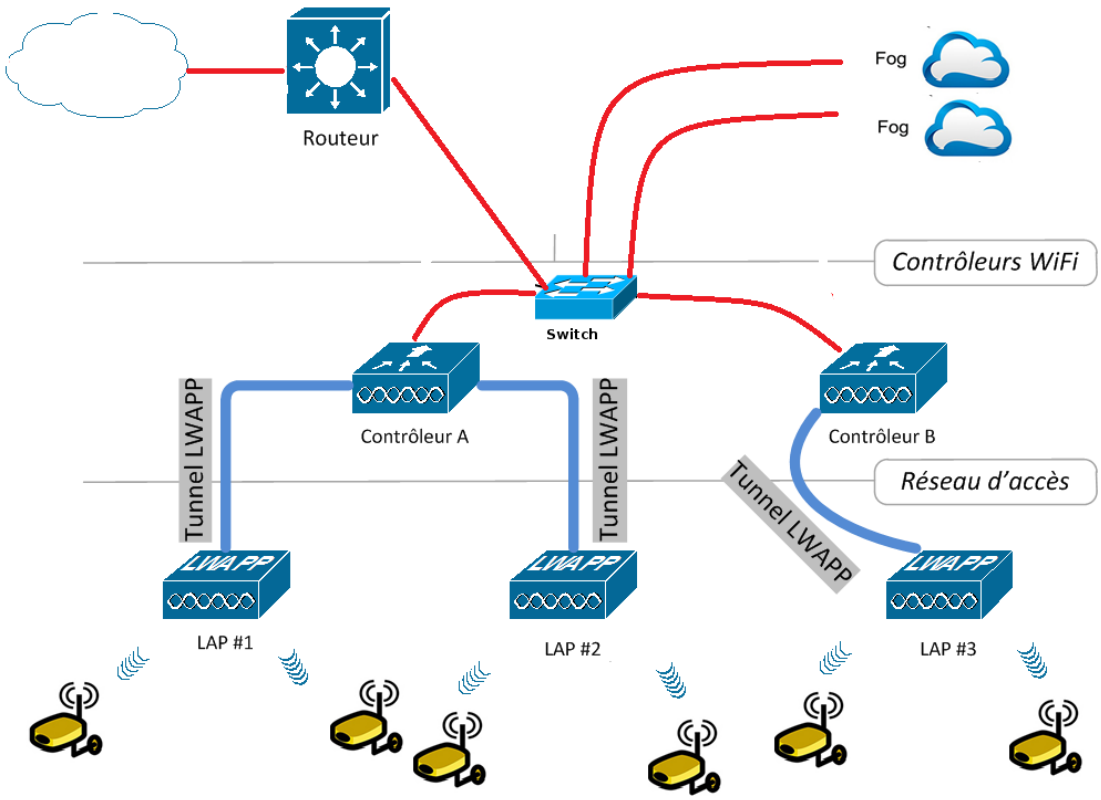


*Figure 26 : Système connecté et complexe nécessitant un équilibrage de charge*

### 4.5.2 Contrôleurs

Le contrôleur est un élément indispensable au fonctionnement d'une solution centralisée : il en constitue le cerveau. Il a en charge le pilotage de tous les points d'accès associés, pour les fonctions telles que :

- la gestion des versions logicielles (*firmwares*) et des configurations ;
- l'application des profils réseau et des politiques de sécurité ;
- la gestion des radiofréquences ;
- la QoS.
- cache DHCP/RADIUS.

Il constitue le point de terminaison des tunnels LWAPP, c'est à dire le point de sortie pour le trafic issu des clients.

## 4.6 Méthode proposée

La méthode proposée est illustrée par la figure 27. Dans ce travail, nous proposons d'utiliser un contrôleur de réseau comme un équipement d'association pour éviter la situation où certains

Fog soient surchargés alors que d'autres sont inactifs. Le contrôleur utilisé présente une interface entre la couche regroupant les Fogs et la troisième couche regroupant les caméras intelligentes.

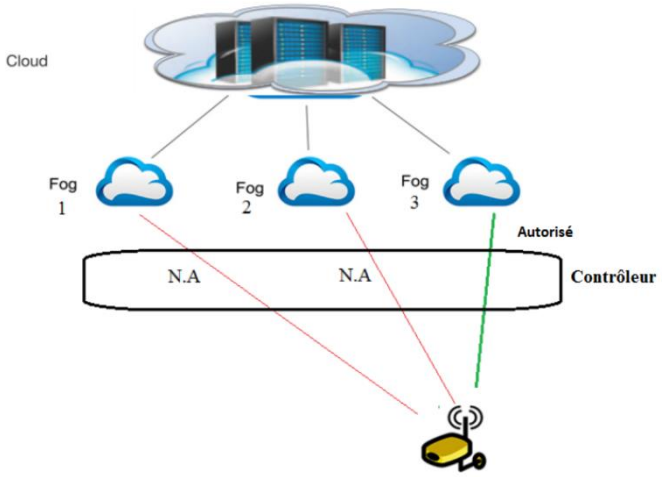


*Figure 27: Gestion du contrôleur*

Comme nous pouvons le voir sur la figure 27, une nouvelle caméra apparaît dans le système et tente de faire une association avec l'un des Fogs. Celle-ci envoie un message au contrôleur et en fonction de la charge de chaque Fog, le contrôleur y connecte la caméra. La ligne rouge de la figure 6 signifie que l'accès à la connexion est refusé par le contrôleur alors que la ligne verte signifie que l'association est acceptée. La mémoire du contrôleur stocke tous les états des Fogs. Si l'un des processus Fogs démarre, il envoie un message au contrôleur, de cette façon, nous obtenons un journal mis à jour contenant tous les états de ces derniers. Nous considérons un WLAN IEEE 802.11 avec un ensemble de Fogs, noté $F$, et $|F|$ indique le nombre de Fogs. Tous les Fogs sont connectés à une infrastructure réseau et à un contrôleur OpenFlow. Dans notre schéma, nous nous concentrons sur le traitement des messages de connexion au Fog. Nous désignons la charge de traitement du Fog α par $L_\alpha$. Selon la configuration et la capacité, chaque $F$ a son nombre de tâches exécutées, noté $P_\alpha$.

Le contrôleur écoute tous les messages arrivant des Fogs et des caméras. Quand une caméra rejoint un WLAN, elle envoie une requête de connexion au contrôleur en même temps que celui-ci écoute chaque canal pour tous les messages de connexion au Fog. Ensuite, il associe la caméra au Fog qui a la charge la plus faible, qui est déterminée par la charge de traitement $P_\alpha$. Dans notre système, nous utilisons un contrôleur pour collecter les RSSI des caméras. Chaque brouillard signale son état et son RSSI au contrôleur en utilisant le protocole OpenFlow. De temps en temps, chaque brouillard envoie sa charge $L$ et la contribution de charge de ses

caméras $L_{\alpha,c}$ au contrôleur, où α désigne le Fog et *c* la caméra. Sur la base des informations ci-dessus, le contrôleur connaît et établit les relations d'association entre les caméras et les Fogs. De la même manière, la charge des Fogs est également connue par le contrôleur. Dans notre système, le contrôleur a une vue globale sur l'ensemble du réseau. Les étapes de l'approche proposée sont résumées comme suit :

- **Étape 1.** Le contrôleur reçoit les événements d'arrivée et charge les messages des Fogs.
- **Étape 2.** Le nombre de nouveaux événements d'arrivée est ajouté dans la table de comptage des arrivées. Ensuite, le contrôleur calcule l'augmentation de l'événement au dernier intervalle de temps. Cette table contient les informations des Fogs : la charge actuelle et la charge maximale, car il est possible que la capacité de traitement des Fogs ne soit pas la même.
- **Étape 3.** Le contrôleur compare le nombre d'événements d'arrivée actuels avec le passé. Lorsqu'un Fog dépasse le seuil des événements d'arrivée, il présente un risque élevé de surcharge. À ce moment, le contrôleur donne au Fog une valeur de charge prédite, puis passe à l'étape 6.
- **Étape 4.** Le contrôleur classe tous les Fogs en trois niveaux de charge : vide, normal et surchargé.
- **Étape 5.** Le contrôleur écoute la demande de la caméra.
- **Étape 6.** Le contrôleur calcule les ajustements des points d'accès en fonction de l'algorithme d'équilibrage de charge décrit dans la sous-section suivante.
- **Étape 7.** Le contrôleur associe la nouvelle caméra au Fog ayant une faible charge.

## 4.7 Algorithme proposé pour l'équilibrage de charge

Dans cette section, nous présentons le mécanisme d'équilibrage de charge adaptatif. Rappelons ce que nous avons décrit dans les sections précédentes : la relation d'association, les RSSI des caméras et la charge des Fogs sont tous collectés par le contrôleur. Sur la base des informations ci-dessus, le contrôleur a une vision globale du réseau sans fil entier. Nous combinons la méthode SDN et Cell-Breathing dans un mécanisme Adaptive Load Balancing. L'algorithme Algo1 montre le pseudo code de notre mécanisme pour le contrôleur. Dans cet algorithme, le niveau de traitement des fogs est d'abord initialisé au niveau de traitement maximal. Le contrôleur recueille et calcule la somme de toutes les utilisations de Fog, et l'état du réseau est initialisé à $S^0$. Tout d'abord, l'algorithme trouve de manière itérative le brouillard le plus chargé. Ensuite, le Fog le plus chargé est ajouté à l'ensemble de surcharge *D* et diminue sa puissance

d'un niveau. En attendant, l'ajustement est enregistré dans une structure de pile. Chaque fois que le Fog le plus surchargé définit le niveau de puissance de la balise, l'algorithme met à jour les états de Fog en simulant la relation d'association entre les Fogs et les caméras. Chaque état est comparé aux états précédents pour trouver l'état optimal. Jusqu'à ce que l'itération se termine, l'algorithme définit le niveau de traitement de la balise Fog en fonction de l'état optimal.

***Algo 1 : Equilibrage de charge***

Entrée : $F$désigne l'ensemble de tous les Fogs, $C$ désigne l'ensemble de toutes les caméras

Initialiser le niveau du traitement pour chaque Fog $P_\alpha$ $o$ù $\alpha \in \{1,2, \dots, |F|\}$

Initialiser somme des états d'utilisation des Fogs $S^{index}, index = 0$

Initialiser la pile Adjustement_Log

Initialiser l'ensemble $D = Null$ des Fogs surchargés

On pose $L_\alpha$ pour désigner la charge du Fog $a$

On pose le drapeau t $End_{Flag} = Faux$

1 $\{C_\alpha, L_{a,c}, P_\alpha\} \leftarrow$ Association_relation_MAJ $(F, C)$

2 **Tant que** $End_{Flag} =$ faux **faire**

3 Trouver $L_\alpha$ la charge *Max* du Fog dans $F$ où $\alpha \in \{1,2, \dots, |F|\}$

4 Ajouter le Fog dans l'ensemble $D$

5 $P_\alpha^* \leftarrow P_\alpha^* - 1$ : Eliminer un Fog ayant la charge *Max* de l'ensemble

6 $Adjustement\_Log.Inserer(a)$

7 $S^{index+1} \leftarrow$ Association_relation_MAJ$(F, C)$

8 **Si** $(S^{index+1} >$ max $\_charge)$ **alors**

9 max_charge_index $\leftarrow index + 1$

10 **FinSi**

11 Si $(\exists$ x $\in D$ $et$ $P_x^* = 0$ $ou$ $(D = F) alors$

12 End_Flag=vrai

13 **Optimisation_charge** (Adjustement_Log, max_charge_index)

14 **End if**

15 **End;**

---

L'algorithme proposé a de bonnes propriétés d'équité avec une complexité relativement peu élevée, O(log$N$), où N est le nombre de brouillards. Les algorithmes à base de round-robin ont une complexité $O(1)$ ou quasi-$O(1)$ (sous des hypothèses pratiques), mais en général ils n'ont pas de bonnes propriétés d'équité.

## 4.8 Exemple expérimental

La figure 28 montre un exemple d'une nouvelle caméra apparaissant dans le système. Une fois que la caméra envoie une demande de connexion, le contrôleur reçoit la demande et lance le programme d'équilibrage de charge pour assurer une association fiable avec l'un des Fogs. Comme expliqué dans la section ci-dessus, le contrôleur détecte le Fog à faible charge puis il connecte la caméra au fog choisi par l'algorithme. Le contrôleur se charge dans cette partie de créer la liaison entre la caméra et le Fog. De cette façon, même si le système redémarre, la caméra se connecte toujours au fog approprié.

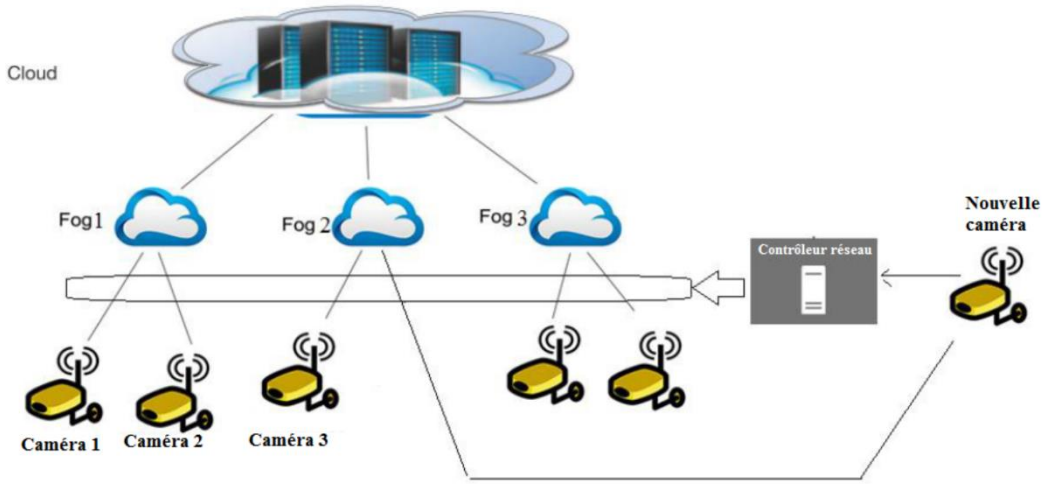


*Figure 28: Association d'une nouvelle caméra*

Après l'exécution, le système détecte que le Fog 2 est celui qui partage le moins de connexions par rapport aux autres, de telle manière que le contrôleur associera la nouvelle caméra à celui-ci. Pour cet exemple, $\alpha \in \{1,2,3\}$, parce que nous avons trois Fogs, donc $|F| = 3$. Nous supposons que cinq caméras ,selon l'exemple, sont connectées; donc pour l'Association_relationship $(F, C)$, $C_\alpha = C_1^1, C_1^2, C_2^3, C_3^4, C_3^5$,. L'algorithme consiste à trouver le Fog avec la charge la plus faible. Initialement, l'ensemble *P* contient tous les Fogs, après la première itération le Fog α = 1 est soustrait de cet ensemble et par conéquent l'ensemble *P* aura seulement deux Fogs. A l'issue de la deuxième itération, le Fog α = 3 sera supprimé. A la fin, seul le Fog α = 2 appartiendra à l'ensemble. Ainsi, la liaison entre la nouvelle caméra sera établie avec le Fog α = 2 par le contrôleur.

## 4.9 Conclusion

Dans ce chapitre, nous avons proposé un système optimisé de vidéosurveillance comportant un schéma d'équilibrage de charge adaptatif à travers le contrôle d'association avec le service SDN en utilisant un réseau sans fil. Le schéma proposé comprend deux parties : la détection d'événement d'arrivée et l'équilibrage de charge adaptatif. Grâce à la technique basée sur SDN, un Fog signale sa situation de charge et de connexion au contrôleur en temps réel. Basé sur la vision globale du contrôleur, nous avons présenté un algorithme portant sur le contrôleur afin d'optimiser la relation d'association entre les Fogs et les caméras.

# 5 HEURISTIQUE POUR UNE ALLOCATION DYNAMIQUE DE TACHES DANS UN SYSTEME DE VIDEOSURVEILLANCE A BASE DE CLOUD/FOG COMPUTING

Dans le chapitre précédent, nous avons vu un algorithme d'équilibrage de charges entre les Fogs afin d'affecter dynamiquement les caméras sur ces boitiers Fog. Le présent chapitre s'intéresse aux tâches de « tracking » dans un système de vidéosurveillance, et décrit une solution basée sur une architecture mixte : Cloud et Fog computing, permettant notamment de minimiser deux mesures majeures de performance du système : le temps de réponse et la consommation d'énergie. En effet, nous présentons une méthode d'allocation dynamique de tâches (de tracking) qui se fait en cascade entre trois entités principales du système : les caméras, le sous-système Fog et le sous-système Cloud. La méthode repose sur l'évaluation d'une fonction de performance des caméras et de la charge de travail des nœuds Fog [SMA218].

## 5.1 Introduction

En fonction du lieu où le système de vidéosurveillance décrit dans le chapitre précédent est installé, il est susceptible d'avoir un nombre important de cibles ainsi que de composants matériels (caméras intelligentes et Fogs). Dans de tels cas l'utilisation de méthodes exactes d'optimisation, dans le but d'affectation des tâches de suivi aux éléments de calcul du système s'avérerait totalement inefficace dû à la nature « difficile » du problème. Dans de tels cas de figure, les heuristiques semblent donc l'unique solution envisageable étant donné leur temps d'exécution acceptable y compris dans des applications en temps réel comme celles ciblées dans le cadre de ce travail. Les sections suivantes ont pour but de décrire une méthode heuristique permettant d'optimiser l'affectation dynamique des tâches aux caméras et aux composants Fogs.

## 5.2 Formulation du problème

La formulation du problème consiste à établir l'ensemble des équations permettant d'exprimer les deux contraintes principales liées aux performances du système : *la consommation d'énergie* et le *temps de traitement*.

### 5.2.1 Consommation d'énergie

Il est important de minimiser la consommation d'énergie cumulée dans toutes les caméras, les nœuds Fogs et le serveur Cloud. La fonction de consommation d'énergie dans le système est définie par :

$$POW^{Sys} = \sum_{i=1}^{n} Pow^{Cam-i} + \sum_{j=1}^{m} Pow^{Fog-j} + Pow^{Cloud}$$

Où:

$POW^{Sys}$ représente la fonction de consommation d'énergie du système.

*Cam-i* représente une caméra.

*n* est le nombre de caméras dans le système.

*Fog-j* représente un noeud Fog.

*m* est le nombre de noeuds Fog dans le système.

$POW^{Cloud}$ la consommation d'énergie par le cloud.

#### 5.2.1.1 Consommation d'énergie au niveau des caméras

La consommation d'énergie au niveau d'une caméra dépend du nombre *de cibles détectées* par celle-ci et du nombre de tâches de tracking qui lui sont affectées.

#### 5.2.1.2 Consommation d'énergie au niveau des nœuds Fog

La consommation d'énergie au niveau d'un nœud Fog dépend du *taux d'arrivée des tâches* (i.e les cibles détectées par les caméras rattachées au Fog), du nombre de délégations qui lui sont attribuées pour exécuter l'algorithme d'allocation et du nombre de tâches de tracking effectuées par le nœud Fog.

#### 5.2.1.3 Consommation d'énergie au niveau du Cloud

La consommation d'énergie au niveau du Cloud dépend du *taux d'arrivée des tâches de tracking* qui lui sont affectées.

### 5.2.2 Temps de traitement

Il est également important de garantir un temps de traitement *minimal* dans l'ensemble du système. Ainsi, la fonction de délai dans le système global est définie comme l'agrégation du :

- Temps de calcul au niveau des caméras, des nœuds Fog ainsi que du Cloud.
- Temps de communication entre :
  - Les caméras et les noeuds Fog.
  - Les noeuds Fog.
  - Les noeuds Fog et le Cloud.

Ceci peut être exprimé par l'équation suivante:

$$DLY^{Sys} = \sum_{i=1}^{n} DLY_{comp}^{Dev-i} + \sum_{j=i}^{m} DLY_{comp}^{Fog-j} + DLY_{comp}^{Cloud} + \sum_{i=1}^{n} DLY_{comm}^{Devi-Fog} + \sum_{\langle j=1 | k=j+1 \rangle}^{m} DLY_{comm}^{Fogj-Fogk} + \sum_{j=1}^{m} DLY_{comm}^{Cloud\_Fogj}$$

Où: $DLY^{Sys}$ représente le délai du système.

$DLY_{comp}^{Dev-i}$ représente le temps de calcul (computation time) des caméras.

$DLY_{comp}^{Fog-j}$ représente le temps de calcul (computation time) des noeuds Fogs.

$DLY_{comp}^{Cloud}$ représente le temps de calcul (computation time) du Cloud.

$DLY_{comm}^{Devi-Fog}$ représente le temps de communication entre cameras et Fogs.

#### 5.2.2.1 Temps de calcul

##### 5.2.2.1.1 Temps de calcul au niveau des caméras

Le temps de calcul dans une caméra augmente avec *le taux d'arrivée des cibles* détectées par la caméra et les tâches de tracking qui lui sont affectées.

##### 5.2.2.1.2 Temps de calcul au niveau des nœuds Fog

Le temps de calcul dans un nœud Fog augmente avec *le taux d'arrivée des tâches* (tracking, négociation, sélection) sur ce nœud. Ainsi, afin d'obtenir un délai de calcul minimal au niveau des nœuds Fog, il est nécessaire de maintenir une charge de travail équilibrée entre ces derniers (c'est-à-dire, éviter autant que possible de submerger un noeud Fog comparativement aux autres noeuds).

##### 5.2.2.1.3 Temps de calcul au niveau du Cloud

Le temps de calcul au niveau du Cloud augmente avec *le taux d'arrivée des tâches de tracking* qui lui sont affectées.

#### 5.2.2.2 Temps de communication dans le système

Le délai de communication entre les caméras et les noeuds Fog est la somme des délais de communication entre les caméras qui détectent la cible et les noeuds Fog. Ainsi, ce délai de communication augmente avec le nombre de caméras détectant la cible et le nombre de cibles elles-mêmes. Cela dépend aussi de la stratégie de placement des caméras. Le délai de communication entre les nœuds Fog augmente avec le nombre de ces derniers qui sont impliqués dans le scénario de négociation. Cela dépend aussi de la stratégie de placement des caméras et de leur distribution entre les nœuds Fog. Le délai de communication entre les nœuds Fog et le Cloud augmente avec le nombre de tâches de tracking déléguées au Cloud.

## 5.3 Méthode d’allocation

L'allocation des tâches est une opération critique en raison de la complexité et du volume des tâches dans les réseaux de caméras intelligentes. Cette opération affecte des objectifs particuliers à chaque caméra de sorte que l'objectif principal de l’ensemble soit atteint. La

solution doit prendre en compte différentes contraintes imposées par la disponibilité des ressources afin de fournir le choix du dispositif le plus approprié pour satisfaire une tâche [TMAP14].

Dans ce système de vidéosurveillance, notre objectif est de trouver la meilleure manière d'attribuer des tâches de tracking aux caméras, aux nœuds Fog et au Cloud afin de garantir le meilleur compromis *entre la consommation d'énergie* et le *temps de traitement*.

## 5.4 Description de la méthode

Lorsqu'une cible arrive dans le système, une tâche $T_k$ de suivi (tracking) est générée et un sous-ensemble de caméras (pouvant voir la cible) peut effectuer la tâche $T_k$. Ainsi, la tâche doit être affectée à une caméra $C_j$ choisie ; on dira que la caméra $C_j$ est allouée à la tâche $T_k$.

Les mesures de performance typiques d'une caméra tiennent compte de la précision et de la rapidité d'exécution de la tâche, de la quantité *d'énergie utilisée*, des *coûts de communication* et de la *durée de vie* de chaque caméra du réseau. Pour cela, afin d'équilibrer la charge de travail entre les caméras, nous définissons une fonction de performance $P_f$ pour chaque caméra $C_j$, relativement à une tâche de suivi $T_k$; cette fonction est représentée par ${P_f}^{C_j}(T_k)$ et définie par une somme pondérée de quatre mesures principales :

- L'énergie restante de la caméra $C_j$ $(Pow^{C_j})$.
- Les ressources disponibles de la caméra $C_j$ $(Res^{C_j})$.
- La fréquence d'images (« frame-rate ») $F_{jk}$ que la caméra $C_j$ peut fournir à la tâche $T_k$.
- La résolution $G_{jk}$ avec laquelle la caméra $C_j$ perçoit la cible relative à la tâche $T_k$.

$$ {P_f}^{C_j}(T_k) = w1 * (Pow^{C_j}) + w2 * (Res^{C_j}) + w3 * (F_{jk}) + w4 * (G_{jk}) $$

$$ \text{Où : } w1 + w2 + w3 + w4 = 1 $$

Nous supposons que chaque caméra peut estimer ses ressources et la quantité d'énergie disponibles et les transmettre au nœud Fog associé. De plus, étant donné les besoins en ressources d'une tâche de tracking $T_k$, une caméra peut calculer la fréquence d'images $F_{jk}$

qu'elle peut fournir à la tâche $T_k$. Elle peut également calculer la résolution $G_{jk}$ associée à la cible visée (liée à la tâche $T_k$).

La méthode d'allocation doit prendre en compte les deux paramètres qui contrôlent les performances du système, principalement : *la consommation d'énergie* et *le temps de traitement*. De ce fait, nous attribuons des poids plus élevés aux deux paramètres $Pow^{C_j}$ et $F_{jk}$. Par exemple, $w1 = w3 = \frac{1}{3}$ $et$ $w2 = w4 = \frac{1}{6}$. Ces poids peuvent être ajustés dynamiquement lorsque la puissance restante de la caméra passe en dessous d'un seuil $T_s$ qui sera fixé.

De plus, la méthode d'allocation doit prendre en compte la charge de travail sur les noeuds Fog afin d'éviter la surcharge d'un nœud Fog comparativement aux autres noeuds. Pour cela, nous définissons une fonction de sélection $S_f$ pour une caméra $C_j$ reliée au noeud Fog $FG_x$, relativement à une tâche $T_{k.}$, par :

$$S_f^{C_j}(T_k) = \frac{P_f^{C_j}(T_k)}{WLoad^{FG_x}}$$

Où : $WLoad^{FG_x}$ représente la charge de travail du noeud Fog $FG_x$, c'est à dire le nombre total de tâches prises en charge par le nœud $FG_x$. Au niveau d'un nœud Fog, on peut avoir une tâche de tracking, une négociation, une sélection (exécution de l'algorithme d'allocation) ou une notification.

En effet, lorsque deux caméras C1 et C2 peuvent effectuer une tâche $T_k$ avec (approximativement) les mêmes fonctions de performance $P_f^{C_1}(T_k)$ et $P_f^{C_2}(T_k)$, la charge de travail des deux nœuds Fog associés (auxquels les caméras C1, C2 sont connectées) devient décisive pour déterminer quelle caméra doit effectuer la tâche de tracking. Ainsi, la caméra qui possède la *plus grande valeur de fonction de sélection* exécute la tâche $T_{k.}$ Le nœud Fog associé à la caméra allouée à la tâche, notifie tous les autres nœuds Fog impliqués dans le processus d'allocation et chaque nœud Fog à son tour, notifie toutes les caméras qui lui sont connectées.

## 5.5 Phases de la méthode d'allocation

La méthode d'allocation opère en trois phases principales : (1) Phase de détection, (2) Phase de négociation et (3) Phase d'allocation.

### 5.5.1 Phase de détection

*Déclenchée par*: l'arrivée d'une cible à suivre.

*Traitement:* Chaque caméra $C_j$ qui peut voir la cible (relative à la tâche $T_k$) envoie un message $Mess^{Tk}$ $(C_j, FG_x)$ à son nœud Fog $FG_x$ pour l'informer de la détection de la cible.

*Sortie:* l'ensemble des nœuds Fog impliqués dans le processus de suivi; soit SFog = { $FG_1$, $FG_2$, … , $FG_r$}

### 5.5.2 Phase de négociation

*Entrée*: l'ensemble des nœuds Fog impliqués dans le processus ($SFog$)

*Traitement*: Tous les nœuds Fog appartenant à $SFog$ s'envoient des messages $Mess^{Tk}$ $(FG_x, FG_y)$ entre eux afin de sélectionner le nœud Fog qui sera délégué pour exécuter la phase d'allocation

*Sortie:* Un nœud Fog sélectionné $FG_s \in \{ FG_1, FG_2, \dots , FG_r\}$

### 5.5.3 Phase d'allocation

*Entrée:* L'ensemble des caméras impliquées dans le processus de tracking, soit $Scam = \{C1, C2, \dots Cq\}$

*Traitement:* Comparaison des fonctions de sélection (et de performance) $Sf^{Cj}$ des différentes caméras dans $Scam$.

*Sortie:* Une camera sélectionnée $Cj \in \{ C1, C2, \dots , Cq\}$ ayant la plus grande valeur de fonction de sélection.

## 5.6 Algorithme d'allocation

Nous introduisons tout d'abord quelques notations utilisées dans l'algorithme.

### 5.6.1 Notations

*Etat* $(C_j)$ désigne à tout moment l'état de la camera $C_j$

*Etat* $(C_j)$ = 1 si la caméra $Cj$ est occupée (allouée à une tâche)

= 0 si la caméra $Cj$ est libre

$Cible(C_j, T_k)$ = 1 si la caméra $C_j$ peut voir la cible relative à la tâche de tracking $T_k$

= 0 sinon

$Mess^{Tk}(C_j, FG_x)$ désigne un message relatif à la tâche $T_k$ envoyé de la caméra $C_j$ à son noeud Fog $FG_x$.

$WLoad^{FG_x}$ désigne la charge de travail associée au noeud Fog $FG_x$; cette charge augmente avec le nombre de scénarios de négociations auquel le noeud participe, le nombre de tâches de tracking qui lui sont affectées et le nombre de délégations qui lui sont assignées.

$Mess^{Tk}$ $(FG_x, FG_y)$ désigne un message relatif à la tâche $T_k$, envoyé du noeud Fog $FG_x$ vers le noeud Fog $FG_y$.

### 5.6.2 Algorithme d'allocation

**Algorithme d'allocation**

*// Phase de détection*

Pour chaque tâche $T_k$

Pour chaque caméra *Cj* connectée au nœud Fog $FG_x$

avec $Cible(C_j, T_k)$ = 1 et $Pow^{C_j} < TS$

Envoyer un message $Mess_{Tk}$ $(C_j, FG_x)$ de la caméra $C_j$ au noeud Fog $FG_x$

*// Phase de négociation*

$SFog = \{FG_1, FG_2, \dots, FG_r\}$ *// L'ensemble des noeuds Fog impliqués dans le processus de négociation associé à la tâche* $T_k$

Pour chaque *Fog* $FG_i$ *dans* $Sfog$

Envoyer un message $Mess^{Tk}$ $(FG_x, FG_y)$ aux autres noeuds Fog dans $SFog$

Sélectionner le Fog $FG_s$ dans $SFog$ ayant la charge minimale

$Fgs = Select\_Fog\ (FG_1, FG_2, \dots, FG_r);\ FG_s \in \{FG_1, FG_2, \dots, FG_r\}$

Où $\boldsymbol{Wload}\ (FG_s) <= \boldsymbol{Wload}\ (FG_i)$ pour $1 <= i <= r$

*// Phase d'allocation*

Sélectionner la caméra $C_j$ ayant la valeur maximale de fonction de sélection

$Cs = Select\_Camera\ (C1, C2, \dots, Cq);\ Cs \in \{C1, C2, \dots, Cq\}$

où $\boldsymbol{Sf}\ (\boldsymbol{Cs}) >= \boldsymbol{Sf}\ (\boldsymbol{Ci})$ pour $1 <= i <= q$

Affecter la tâche $T_k$ à la caméra $Cs$

Etat $(Cs) = 1$ *// Mettre à jour Etat* $(Cs)$

*// Notifier les noeuds Fogs dans* $\{FG_1, FG_2, \dots, FG_r\}$

Pour chaque *Fog* $FG_i$ *dans* $Sfog$

Envoyer un message $Mess^{Tk}\ (FG_s, FG_i)$

*//Notifier toutes les caméras impliquées dans le processus de sélection*

Pour chaque *caméra* $C_i$ *dans* $\{C1, C2, \dots, Cq\}$

Envoyer un message $Mess^{Tk}(FG_x, C_i)$ où $FG_x \in \{FG_1, FG_2, \dots, FG_r\}$

## 5.6.3 Remarque

Une caméra $C_j$ ne participe pas dans le processus de sélection et de tracking si :

- Elle ne possède pas suffisamment d'énergie $Pow^{Cj} < TS$ (TS est un seuil fixé)
- Elle ne peut pas voir la cible $Cible\ (C_j, T_k) = 0$
- Elle ne possède pas assez de ressources disponibles ($Res^{Cj} < ressources\ requises$)

## 5.6.4 Scénarios alternatifs d'allocation

Deux scénarios alternatifs sont à considérer pour la méthode d'allocation, le premier consiste à affecter la tâche de tracking à un *nœud Fog* et le second représente le cas où la tâche de tracking est affectée au *serveur Cloud.*

- *Allocation à un noeud Fog*

Pour une tâche donnée $T_k$, si toutes les caméras Cj avec *Cible(Cj, $T_k$)= 1* ne peuvent pas exécuter la tâche car : $Pow^{Cj} < TS$ ou $Res^{Cj} <$ ressources requises, on affecte alors la tâche de tracking $T_k$ au nœud $FG_x$ tel que:

- ✓ $FG_x \in SFog = \{FG_1, FG_2, \dots, FG_r\}$,
- ✓ Etat $(FG_x) = 0$
- ✓ $Wload\ (FG_x) <= Wload\ (FG_i)$ pour $1 <= i <= r$

- *Allocation au Cloud*

Pour une tâche donnée $T_k$, si tous les nœuds Fog ne peuvent pas exécuter la tâche à cause de leur charge de travail qui excède un seuil $T_s$ .

## 5.7 Complexité de la méthode

Nous réitérons l'algorithme d'allocation pour chaque tâche $T_k$ associée à une cible qui arrive dans le système. Soit $N$ le nombre total de tâches de tracking générées.

Pour chaque tâche $T_k$, la méthode considère toutes les caméras qui peuvent voir la cible relative à celle-ci. Le nombre de caméras pouvant voir la cible dépend de la surface couverte par ces dernières ainsi que leur stratégie de placement. Nous considérons pour chaque cible, un nombre moyen de caméras $Nb_c = \frac{NbTotal_{cam}}{area}$ .

La communication entre les caméras et les noeuds Fog génère $2 * Nb_c$ messages (les messages envoyés des caméras vers leurs nœuds Fog respectifs et les messages de notification envoyés depuis les nœuds Fog vers les caméras associées) ; la complexité de cette phase est en $O(Nb_c)$.

Le nombre de noeuds Fog dépend du nombre de caméras connectées à chaque noeud Fog. Soit $Nb_F$ le nombre moyen de nœuds Fog impliqués dans la méthode d'allocation relativement à une tâche $T_k$.

La phase de négociation est basée sur la communication entre les nœuds Fog impliqués dans le processus d'allocation. Le nombre de messages générés est $NbF * (NbF - 1)$ *;* la complexité de cette phase est en $O(NbF^2)$.

Pour sélectionner un nœud Fog parmi $NbF$, il s'agit de comparer les charges de travail des nœuds Fog. La complexité de la méthode de sélection est donc proportionnelle au nombre de nœuds Fog, soit une complexité en $O(NbF)$.

Pour sélectionner une caméra et l'allouer à la tâche $T_k$, il s'agit de comparer les fonctions de performance de ces dernières. La complexité de cette phase de sélection est proportionnelle à $NbC$, soit une complexité en $O(NbC)$.

La notification des nœuds Fog est (linéairement) proportionnelle à $NbF$ et génère une complexité de $O(NbF)$.

La notification des caméras impliquées dans le processus d'allocation est (linéairement) proportionnelle à $NbC$ et génère une complexité de $O(NbC)$.

La complexité de la méthode d'allocation étant la somme de toutes les complexités précédemment évaluées, elle dépend du nombre de tâches $N$, du nombre de caméras $(NbC)$ et du nombre de Fog $(NbF)$ et peut être exprimée comme suit :

$$C\ (N, NbC, NbF) \ = \ O(N * NbC) + \ O(N * NbF) + \ O(N * NbF^2)$$
$$= \ O(N * (NbC + \ NbF^2)).$$

## 5.8 Exemple et évaluation de la méthode

Dans cette section, nous présentons un exemple simple d'exécution de la méthode d'allocation et nous discutons différents cas d'exécution de celle-ci.

Par exemple, nous considérons un système de vidéosurveillance contenant $n$ caméras et $m$ nœuds Fog ($n = 20$ et $m = 5$). Nous supposons que les caméras sont distribuées de manière équilibrée sur les noeuds Fog. Ainsi, les caméras *C1, C2, C3 et C4* sont connectées au noeud Fog $FG_1$, les caméras *C5, C6, C7* et *C8* sont connectées au nœud Fog $FG_2$, et ainsi de suite…

### 5.8.1 Phase de détection

Considérons le scénario suivant : Une cible arrive dans le système, elle est détectée par les caméras *C1, C2, C5, C7* et *C10.* Comme décrit précédemment, *C1* et *C2* sont connectées à $FG_1$, *C5* et *C7* à $FG_2$ et *C10* est connectée à $FG_3$.

Soient Scam et SFog les deux ensembles suivants :

$Scam \ = \ \{C1, C2, C5, C7, C10\} \qquad SFog \ = \ \{FG_1, FG_2, FG_3\}$

Chaque caméra dans *Scam* envoie un message de détection à son nœud Fog.

### 5.8.2 Phase de négociation

Les noeuds Fog $FG_1$, $FG_2$ et $FG_3$ négocient entre eux afin de sélectionner le nœud qui doit exécuter l'algorithme d'allocation. La négociation est effectuée via un échange de messages (6 messages sont générés).

Supposons que : $Wload^{FG_2} < Wload^{FG_1} < Wload^{FG_3}$. Dans ce cas, le nœud Fog $FG_2$ est sélectionné pour exécuter l'algorithme d'allocation et par conséquent $Wload^{FG_2}$ augmente.

### 5.8.3 Phase d'allocation

Cette phase repose sur la comparaison des fonctions de sélection et de performance des caméras contenues dans $Scam = \{C1, C2, C5, C7, C10\}$.

Supposons que : $Sf^{C7} > Sf^{C5} > Sf^{C2} > Sf^{C10} > Sf^{C1}$.

Dans ce cas, la caméra $C7$ est sélectionnée pour la tâche de tracking associée à la cible et par conséquent Etat(C7) reçoit la valeur 1 et les nœuds Fog $FG_1$ et Fog $FG_3$ sont notifiés par Fog $FG_2$ (2 messages sont générés).

Simultanément, $FG_1$ notifie les caméras $C1\ et\ C2$, $FG_2$ notifie les caméras $C5\ et\ C7\ et\ FG_3$ notifie la caméra $C10$.

### 5.8.4 Quelques résultats

Le tableau suivant montre des cas d'exécution de la méthode d'allocation selon différentes valeurs des principaux paramètres que reçoit l'algorithme.

*Tableau 5: Quelques résultats d'exécution de la méthode d'allocation*

| ***ligne*** | **Total NBCam** | **Total NBFog** | **\|Scam\|** | **\|SFog\|** | **NBmess Cam-Fog** | **NBmess Fog-Fog** | **NBcomp** | **NB_ Op** |
|---|---|---|---|---|---|---|---|---|
| *1* | 20 | 5 | 7 | 2 | 14 | 3 (2+1) | 9 | 26 |
| *2* | 32 | 8 | 12 | 3 | 36 | 8 (6+2) | 15 | 59 |
| *3* | 48 | 8 | 12 | 3 | 36 | 8 (6+2) | 15 | 59 |
| *4* | 48 | 12 | 15 | 4 | 60 | 15 (12+3) | 19 | 94 |
| *5* | 64 | 16 | 15 | 4 | 60 | 15 (12+3) | 19 | 94 |
| *6* | 64 | 16 | 20 | 5 | 100 | 24 (20+4) | 25 | 149 |
| *7* | 72 | 18 | 12 | 3 | 36 | 8 (6+2) | 15 | 59 |
| *8* | 80 | 20 | 16 | 4 | 48 | 15 (12+3) | 20 | 83 |
| *10* | 80 | 20 | 14 | 4 | 52 | 15 (12+3) | 18 | 85 |
| *11* | 80 | 20 | 20 | 5 | 100 | 24 (20+4) | 25 | 149 |
| *12* | 100 | 16 | 20 | 5 | 100 | 24 (20+4) | 25 | 149 |
| *13* | 100 | 25 | 22 | 6 | 132 | 35 (30+5) | 28 | 195 |
| *14* | 100 | 25 | 15 | 4 | 60 | 15 (12+3) | 19 | 94 |
| *15* | 200 | 50 | 30 | 8 | 240 | 63 (56+7) | 38 | 341 |
| *16* | 200 | 50 | 20 | 5 | 100 | 24 (20+4) | 25 | 149 |
| *17* | 200 | 50 | 16 | 4 | 64 | 15 (12+3) | 20 | 99 |
| *18* | 250 | 62 | 20 | 5 | 100 | 24 (20+4) | 25 | 149 |
| *19* | 300 | 75 | 30 | 8 | 240 | 63 (56+7) | 38 | 341 |
| *20* | 300 | 75 | 16 | 4 | 64 | 15 (12+3) | 20 | 99 |

### 5.8.5 Explication

Les deux premières colonnes "Total Nbcam" et "TotalNbFog" désignent respectivement le nombre total de caméras et le nombre total de nœuds Fog dans le système. Ces deux paramètres déterminent la taille du problème. Nous supposons que chaque nœud Fog connecte quatre caméras distinctes.

Le cardinal de l'ensemble $Scam$, $|Scam|$, représente le nombre de caméras qui détectent une cible par rapport à une tâche de tracking $T_k$. De la même manière, le cardinal de l'ensemble $SFog$, $|SFog|$, représente le nombre de nœuds Fog auxquels les caméras de l'ensemble Scam sont connectées. Notons que $|SFog|$ augmente avec le nombre de caméras dans $Scam$.

*NBmess Cam-Fog* est le nombre de messages échangés entre les caméras dans $Scam$ et leurs nœuds Fogs respectifs pendant la phase de détection et de notification. *NBmess Fog-Fog* est le nombre de messages échangés entre tous les nœuds Fogs dans $SFog$ pendant la phase de négociation et les messages de notification émanant du nœud Fog sélectionné (pour la phase d'allocation) vers les autres nœuds Fog dans $SFog$.

*NBcomp* est le nombre de comparaisons effectuées (comparaison des fonctions de performance des caméras et des charges de travail des nœuds Fog) afin de sélectionner un nœud Fog ou une caméra devant être allouée à la tâche, pendant la phase d'allocation.

*NB_Op* est le nombre total d'opérations effectuées (messages échangés et comparaisons) afin de décider à quelle caméra (ou Fog), la tâche de tracking sera allouée.

On peut voir, sur les lignes coloriées du tableau Tab I (voir lignes 6, 11, 12, 18 par exemple) que le cardinal de $Scam$ et le cardinal de $SFog$ n'augmentent pas automatiquement avec la taille du problème (soit Total NBcam et Total NBFog). Par conséquent, le nombre de caméras (et de nœuds Fogs) qui détectent une cible arrivant dans le système, ne dépend pas du nombre total de caméras (et de nœuds Fog) mais dépend plutôt de la stratégie de placement des caméras dans la zone couverte par le système de vidéosurveillance et du nombre moyen de caméras connectées au même nœud Fog. De ce fait, nous déduisons que le temps de réponse dans le système ne dépend pas de la taille du problème mais dépend de la stratégie de placement des caméras. Ainsi, la stratégie de placement des caméras devrait être bien adaptée à la zone couverte par le système et le nombre moyen de cibles arrivant sur cette dernière. En d'autres termes, le nombre de caméras (et de nœuds Fog) qui détectent une cible donnée devrait être

bien étudié dans la stratégie de placement, de manière à ne pas prévoir un nombre beaucoup plus grand que nécessaire.

De plus, nous affirmons que les performances de la méthode d'allocation se dégradent avec l'augmentation du nombre de tâches assignées au Cloud lorsque les ressources disponibles (et l'énergie restante) dans les caméras et les nœuds Fogs diminuent considérablement.

## 5.9 Conclusion

Dans ce chapitre, nous avons décrit une méthode d'allocation dynamique de tâches de suivi (tracking) dans un système de vidéosurveillance. La méthode proposée considère deux paramètres majeurs pour obtenir les meilleures performances dans le réseau : la consommation d'énergie et le temps de traitement. L'objectif est de trouver le meilleur compromis entre ces deux paramètres afin de les minimiser. Pour cela nous avons defini un algorithme prenant en comptes des principaux paramètres. Apres l'exécution de la méthode d'allocation selon différentes valeurs que reçoit l'algorithme d'allocation, nous déduisons que la complexité de cette approche ne dépend pas de la taille du problème définie par le nombre total de caméras et de nœuds Fog (ce qui est un avantage lors du passage à l'échelle), mais dépend plutôt de la stratégie de placement des caméras.

Pour les systèmes de petites tailles l'usage des approches optimales peut s'avérer intéressant et efficace en terme de résultats tout en assurant des résultats d'exécution acceptables. Le chapitre suivant propose une méthode de programmation linéaire entière pour une allocation de tâches de suivi optimale.

# 6 ALLOCATION OPTIMALE DES TACHES DE SUIVI DANS UN SYSTEME DE VIDEOSURVEILLANCE DISTRIBUEE BASE SUR LE FOG/CLOUD

Plus récemment, les composants matériels, appelés FPGA (Field Programmable Gate Array), ont pris de l'amplitude dans les architectures de plate-forme FCIoT (FoG-Cloud-IoT). Ces composants se caractérisent par des modes de configuration dynamiques et partiels, permettant aux plateformes FCIoT de s'adapter rapidement aux changements survenus lors d'un événement, d'ouvrir de nouveaux problèmes aux concepteurs pour améliorer la disponibilité et la continuité des services lancés sur ces plateformes. De telles plates-formes représentent, en effet, un sérieux défi à surmonter, notamment en termes de déploiement et de positionnement des FoGs.

L'un des défis modernes les plus importants est celui de faire évoluer un système de surveillance vidéo basé sur le Cloud existant avec plusieurs caméras intelligentes hétérogènes et de l'adapter à une architecture de Fog/Cloud afin d'en améliorer les performances. Les systèmes actuels de vidéosurveillance contiennent un certain nombre de caméras intelligentes ayant des puissances de calcul distinctes et différentes quantités d'énergie disponibles. Ainsi, lorsqu'une entreprise décide de faire évoluer un tel système, généralement des éléments de Fogs et éventuellement de nouvelles caméras intelligentes sont ajoutés, sans supprimer ceux déjà existants, afin de maintenir un coût raisonnable. Dans un tel système, les tâches de suivi peuvent être exécutées sur n'importe lequel des éléments de calcul disponibles dans le système, mais leurs affectations peuvent avoir un impact significatif sur les performances de l'ensemble du système.

L'objectif de ce chapitre est de proposer une autre méthode permettant une assignation dynamique, optimale et automatique des tâches de suivi sur les éléments de traitement disponibles d'un système de vidéosurveillance [SMA318].

## 6.1 Fonctionnement du système

Même si le problème que nous voulons aborder est NP complet, il peut néanmoins être modélisé en utilisant la technique de programmation linéaire en nombre entier (Integer Linear Programming, ILP) en raison de sa nature optimale et de la disponibilité des outils de résolution.

Afin de trouver des trajectoires optimales, nous avons utilisé la fonction objective suivante :

$$Min\ \alpha \sum_{i=0}^{a} Com_i \sum_{i=0}^{a} \sum_{j=0}^{n+m} T_{ij} X_{ij} + \beta \sum_{i=0}^{a} \sum_{j=0}^{n+m} P_{ij} X_{ij}$$

Explications :

α et β sont des valeurs constantes devant être fixées par le concepteur de systèmes. Plus α (resp β) est grand, plus le poids du coût de la communication/consommation d'énergie est augmenté dans la fonction objective.

$Com_i$ est la quantité de données reçues par la tâche de suivi de la cible i.

$T_{ij}$ est le temps de communication du tracker de la cible i et du périphérique j (Il est initialisé à zéro (0) pour tous les index j correspondants à des caméras capables de suivre la cible i et à l'infini pour tous les autres index de caméras).

$X_{ij}$ est une valeur booléenne pour tester si une cible est affectée à un dispositif (Camera, Fog, Cloud)

$\forall\ i\ \in 0..a\ et\ \forall\ j\ \in 0..(n+m)$

$P_{ij}$ est la quantité d'énergie consommée par le périphérique $j$ pour exécuter le tracker $i$. Pour définir cette valeur, nous avons choisi d'utiliser des intervalles de temps de 8 heures correspondant généralement au temps de travail. Ainsi, une cible ne sera affectée à un appareil donné que si cette dernière a suffisamment d'énergie pour l'exécuter pendant 8 heures.

| Variables contrôlables | Variables non contrôlables |
|---|---|
| $X_{ij}$ | $Com_i, T_{ij}, P_{ij},$ |

*Sous les contraintes :*

$\sum_{j=0}^{n+m} X_{ij} = 1 \; \forall \; i \in 0..a$

Cette formule a pour fonction de garantir que chaque cible sera affectée à un seul périphérique.

$\sum_{i=0}^{a} C_{ij} X_{ij} \leq CP_j \; \forall \; j \in 1..n+m$

Cette formule a pour fonction de s'assurer que chaque dispositif (sauf le Cloud) ne dépassera pas sa capacité en termes de puissance de calcul.

$$\sum_{i=0}^{a} P_{ij} X_{ij} \leq E_j \; \forall \; j \in 1..n+m$$

Cette formule a pour fonction de s'assurer que chaque appareil (sauf le Cloud) ne dépassera pas sa capacité d'énergie disponible.

$X_{ij} \in 0..1$

$\forall \; i \in 0..a$

et $j \in 0..$n + m  Sont des variables booléennes

## 6.2 Analyse du modèle et complexité

Pour une instance de problème, nous obtenons :

- $variables\_NB = M * (m + n + 1)$ variables.

Où $M$ est le maximum de nouvelles cibles susceptibles d'apparaître dans le système à tout moment.

Dans des applications réalistes, M ne peut pas être élevé en restant toujours autour de deux (2) ou trois (3) nouvelles cibles. Ainsi, dans presque tous les cas pratiques, le nombre de variables pourrait être inférieur à 3 $(m + n + 1)$.

- $contraintes\_NB = (a + 1) + 2 * (n + m)$ contraintes.

La modélisation du problème d'assignation des tâches de suivi par l'approche de programmation linéaire d'entiers proposée présente les avantages majeurs exposés ci-dessous :

(1) C'est une méthode exacte qui, contrairement aux méthodes heuristiques, donne une solution optimale.

(2) C'est un modèle très générique qui permet l'intégration de capteurs et de dispositifs informatiques dans toutes les couches de l'architecture des systèmes (caméras intelligentes, Fog et Cloud).

(3) Il est également capable de gérer n'importe quel nombre de cibles.

(4) Il existe de nombreux outils disponibles permettant la résolution d'un tel modèle. La bibliothèque Cplex est celle que nous avons utilisée dans notre environnement.

(5) La complexité du modèle de programmation linéaire d'entiers proposé reste acceptable car le code **C** utilisant la bibliothèque Cplex pour la résolution de problème est exécuté dans le Cloud, en utilisant seulement une parte des ressources disponibles, théoriquement illimitées.

## 6.3 Evaluation

Afin d'illustrer l'efficacité et le bon fonctionnement de la méthode d'assignation des tâches de suivi proposée, nous avons modélisé et résolu plusieurs instances du problème. La bibliothèque Cplex a été utilisée pour la modélisation et la résolution.

Pour ces expériences, nous avons utilisé un HP SERVER (Proliant DL980 G7) avec huit (8) Xeon Six Core E6540 et deux-cent-cinquante-six (256) Go (Gigaoctet  de mémoire installée et 64 T octets de disque dur comme couche nuageuse. Nous avons utilisé un benchmark proche des problèmes réalistes avec un serveur cloud, 4 brouillards et 20 caméras.

Etant donné que ce travail se concentre uniquement sur la méthode d'affectation qui est exécutée sur le Cloud, nous n'implémentons pas de couches de Fogs et de caméras.

Ainsi, après la résolution du problème initial, nous avons successivement ajouté deux (2), quatre(4), cinq (5), huit (8), dix (10) et douze (12) nouvelles cibles puis résolu le problème d'affectation des tâches après chaque modification de la configuration initiale. Toutes les solutions ont été obtenues très rapidement (en temps réel) et ont pris moins d'un centième de seconde. Ces résultats confirment que les modèles proposés constituent une solution efficace

pour les problèmes de taille moyenne que l'on retrouve dans les systèmes de vidéosurveillance réalistes.

## 6.4 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle méthode automatique permettant une résolution optimale du problème d'assignation dynamique des tâches dans les systèmes de vidéosurveillance Fog/Cloud. L'approche proposée consiste en un modèle ILP exécuté en tant que tâche spécifique sur le serveur Cloud, où il tire parti de l'importante puissance de calcul disponible. Cette méthodologie a donné des résultats intéressants et efficaces en termes de temps d'exécution, et ce pour diverses instances réalistes. Puisque dans notre modèle, certaines variables sont booléennes, le problème est que NP est difficile et son étape de résolution peut être lente selon le nombre de ces variables, particulièrement si nous voulons exécuter la tâche de résolution sur un autre périphérique que le Cloud. Ainsi, pour les applications intégrant beaucoup de capteurs et de composants de Fog, et lorsque nous souhaitons plus de flexibilité concernant la tâche de résolution, nous recommandons l'utilisation de méthodes stochastiques ou heuristiques à la place du modèle linéaire. Nos futurs travaux consisteront à explorer de telles solutions.

# 7 PROTOTYPAGE ET EXPERIMENTATION

Dans le cadre des travaux de cette thèse, un intérêt particulier a été consacré au développement et au prototypage, en raison notamment de son déroulement au sein de l'entreprise Dcarte Engineering. Ainsi, l'ensemble des algorithmes proposés a été implémenté sous différentes versions et des démonstrateurs ont été construits de façon incrémentale au fur et à mesure. Certains aspects liés aux démonstrateurs font notamment l'objet de dépôts de brevets par l'entreprise.

## 7.1 Matériel de prototypage

L'environnement de l'expérimentation pour le démonstrateur est composé de FPGAs hétérogènes de deux types principalement :

- 6 Virtex 6 munis chacun d'un capteur vidéo faisant ainsi office de caméras intelligentes et programmables de façon matérielle
- 3 Virtex 7 pour réaliser les composants Fog
- 1 plateforme multi Virtex 7 comme Cloud

Le tableau ci-dessous illustre les ressources disponibles dans les deux types de FPGAs utilisés et les compare aux produits des vendeurs concurrents. On peut ainsi voir clairement la différence en termes de ressources à l'avantage du Virtex 7 2000T.

*Tableau 6: Ressources disponibles sur les FPGAs utilisés*

| Device / Items | Xilinx Virtex-6 760 | Xilinx Virtex-7 2000T | S2C Dual V7 TAI LM | S2C Quad V7 TAI LM | Altera Stratix4 820 |
|---|---|---|---|---|---|
| Logic Cells | 758,784 | 1,954,560 ↑ | 3,909,120 ↑ | 7,818,240 ↑ | 813,050LEs |
| Block RAM (Kb) | 25,920 | 46,512 ↑ | 93,024 ↑ | 186,048 ↑ | 33,294 |
| Global Clocks | 18 | 24 | 14 | 14 | 16 |
| Max User I/O | 1,200 | 1,200 | 1,200 | 1,440 | 1,120 |
| GTX Transceivers | 0 | 16 | 32 | 48 | - |

L'utilisation de FPGAs de la famille Xilinx est justifié pour la disponibilité mais surtout pour la possibilité d'intégration de reconfigurations dynamiques et partielles.

Certaines caméras intelligentes du commerce auraient aussi pu être utilisées, mais l'un des buts recherchés était d'avoir la possibilité d'implémenter librement des protocoles de communication pour pouvoir ainsi explorer et tester tous les choix d'implémentation pouvant se présenter.

## 7.2 Environnement de simulation

Les simulations ont été réalisées sur une station de travail avec 32 Go de mémoire vive et un processeur quad core 2.5 GHZ, exécutant un système Ubuntu 12.04. Les logiciels suivants ont été utilisés :

- Eclipse SDK
- NesC (pour les capteurs)
- Tiny OS 2.12
- C++
- GlomoSim Simulator
- VHDL
- VIVADO 2014.4 64bits

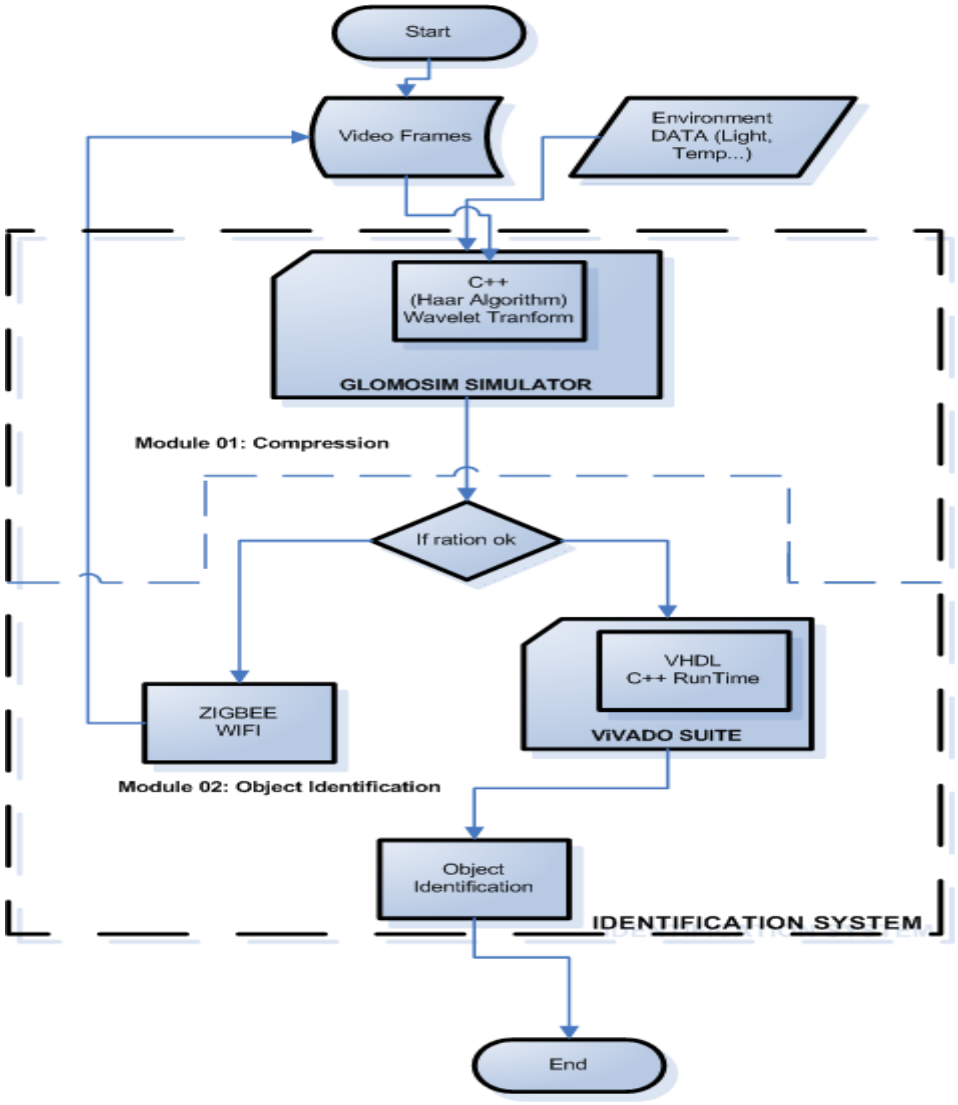


*Figure 29: Flot de simulation de l'identification d'objets*

Ce flot de simulation (figure 29) a permit de tester et de mesurer les performances de plusieurs tâches matérielles (VHDL) de traitement vidéo tout en prenant en compte diverses topologies de connexion entre capteurs et en analysant les flux réseau.

## 7.3 Implémentation d'algorithmes de suivi de cible (Tracking)

Dans un système de vidéosurveillance tel que décrit dans cette thèse, il est d'une importance capitale de sélectionner le bon algorithme à implémenter pour obtenir les meilleures performances possibles.

Au cours des travaux de cette thèse, une étude comparative des différents algorithmes de suivi de cible a été réalisée, en mettant l'accent sur leurs performances lors d'implémentations matérielles ou logicielles (sur GPU notamment). Certains résultats de cette étude ont été présentés dans [SMA118], ils ne figurent néanmoins pas explicitement dans ce manuscrit pour raison de cohérence et de compréhension des principales contributions. L'enseignement principal des diverses implémentations et comparaisons est le constat que les performances de la tâche de tracking et de l'algorithme sélectionné sont fortement liées au degré de parallélisme utilisé mais aussi implicitement au type du FPGA sur lequel l'implémentation est réalisée.

Des optimisations ont également été réalisées sur certaines implémentations, notamment avec l'introduction du pipeline. Ainsi, comme illustré sur les figures 30 et 31, l'introduction du pipeline a permis d'économiser jusqu'à 20% des ressources utilisées. Il s'agit là un aspect important car il permet d'utiliser des FPGAs économiquement intéressants tout en ciblant un nombre de cibles relativement important.

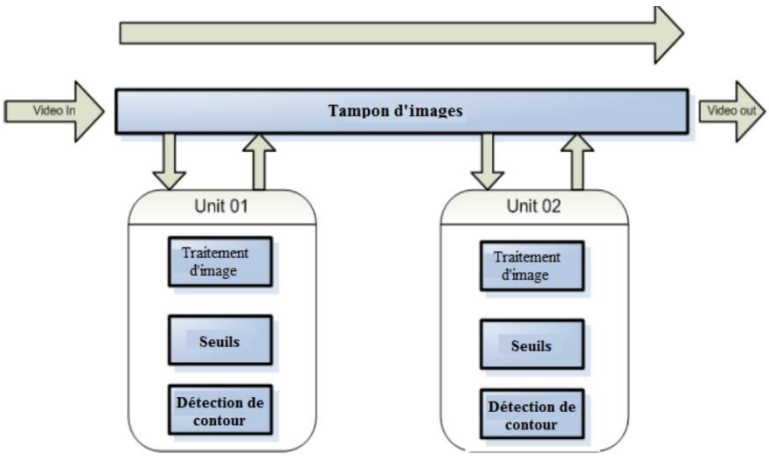


*Figure 30: Implémentation parallèle en 2 unités matérielles d'une même tache de détection de contours dans un algorithme de tracking*

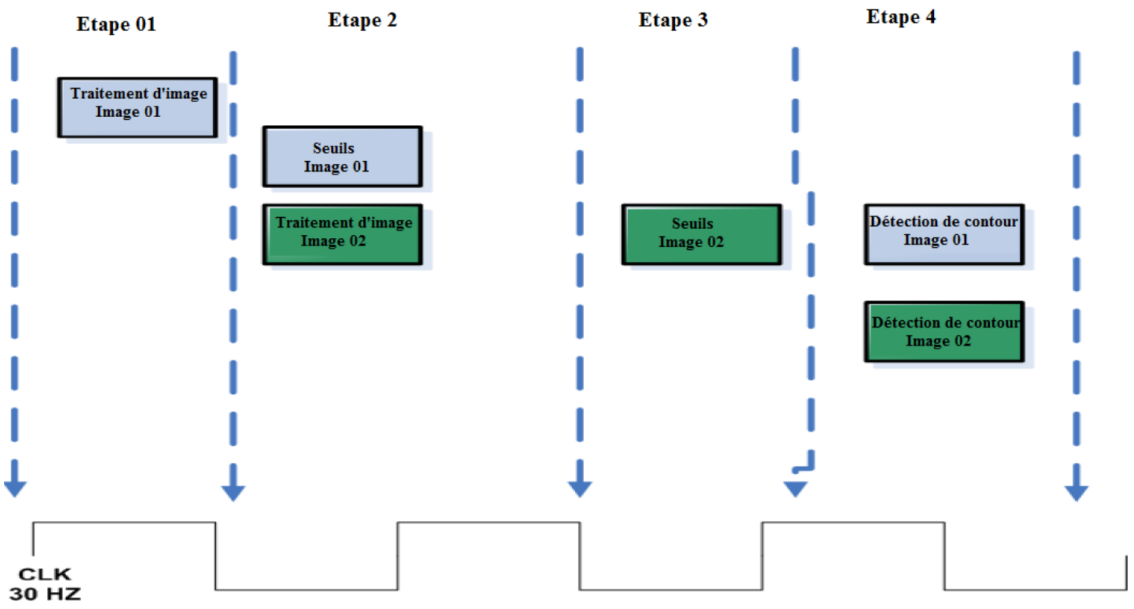


*Figure 31: Implémentation pipelinée sur 4 étages en 2 unités matérielles d'une même tâche de détection de contours dans un algorithme de tracking*

## 7.4 Discussion et conclusion

Le prototype réalisé est fonctionnel avec des cibles apparaissant et disparaissant dynamiquement sur la scène de test. En plus des algorithmes de suivi de cibles, il supporte plusieurs protocoles réseau, permettant ainsi à des caméras hétérogènes de s'authentifier et s'enregistrer « à la volée » dans le système. Cependant, dans l'état actuel il n'est pas possible d'en décrire les détails dans ce manuscrit en raison de leur confidentialité industrielle.

# 8 CONCLUSION ET PERSPECTIVES

## 8.1 Conclusion

Le système de vidéosurveillance basé sur le Fog/Cloud est devenu un outil technologique inestimable pour assurer la sécurité des personnes et des équipements. Elle permet avec l'intermédiaire de logiciels de détecter, d'identifier ou encore de reconnaître automatiquement des objets abandonnés ainsi que des personnes suspectes (qui se démarqueraient par des comportements et/ou attitudes identifiables). Le premier défi rencontré par ce système est celui d'affecter dynamiquement les caméras sur les boitiers Fog tout en traitant les données en temps réels (dans un système où les caméras peuvent être ajoutées, remplacées par d'autres, ou encore changées d'emplacement) et en gardant l'équilibrage de charge entre les Fogs. Le second défi réside dans l'élaboration de la technique permettant d'affecter les cibles à traquer aux caméras (l'allocation de tâches de suivi) dont l'objectif est de minimiser les contraintes principales liées aux performances du système : la consommation d'énergie et le temps de traitement.

Les contributions scientifiques les plus pertinentes pour répondre à ces enjeux techniques ont pour objectif d'améliorer les performances principales de ce système.

Dans un premier temps, il a été présenté une solution basée sur le *fog computing*, solution justifiée au regard des conséquences que la centralisation d'une grande masse d'information à traiter génère. En effet une lourdeur de traitement et un ralentissement au niveau du temps de réponse constituent souvent une contrainte pour beaucoup d'applications, notamment les applications en temps réel. Cette plateforme qui est hautement virtualisée, fournit des services de calcul, de mise en réseau et de stockage entre le Cloud et les terminaux, à l'aide des composants intelligents. Le premier avantage de cette approche est de soulager le réseau d'accès et de réduire la latence des échanges de ces applications, tout en gardant une meilleure qualité de service (QoS). Le second avantage est de rendre les systèmes de vidéosurveillance efficaces et dynamiques, où le nombre et l'emplacement de caméras peuvent librement évoluer, de manière à adapter le système à de nouvelles situations, ou à des besoins particuliers.

Dans cette logique, il a été proposé d'utiliser un contrôleur de réseau comme un équipement d'association de boitiers Fogs afin d'éviter la situation où certains Fogs peuvent être surchargés alors que d'autres sont inactifs. Le contrôleur présente une interface importante entre la couche regroupant les Fogs et la couche regroupant les caméras intelligentes, sans oublier que cet

élément a une vue globale sur l'ensemble du réseau afin d'associer une nouvelle caméra à un Fog à plus faible charge. Apres la combinaison des méthodes SDN (Software-Defined Networks) et Cell-Breathing dans un mécanisme d'équilibrage de charge adaptatif (Adaptive Load Balancing), l'algorithme montre de bonnes propriétés d'équité avec une complexité relativement élevée : $O(logn)$, et $O(1)$ ou quasi $O(1)$ pour les algorithmes à base de round-robin.

L'un des défis modernes les plus importants est de faire évoluer un système de surveillance vidéo basé sur une architecture Fog/Cloud existante avec plusieurs caméras intelligentes hétérogènes, afin d'en améliorer les performances globales. En réalité, les systèmes actuels de vidéosurveillance contiennent un certain nombre de caméras intelligentes présentant des puissances de calcul différentes et détenant différentes quantités d'énergie disponibles. Il est nécessaire de rappeler ici que les dispositifs réseaux (cameras, fog, cloud) à ressources limitées doivent aujourd'hui composer avec la durée de vie de leurs batteries, la bande passante de communication ainsi que leur puissance de calcul. L'allocation des tâches est une opération critique en raison de sa complexité ; les mesures de performance typiques d'une caméra tiennent alors compte de la précision et de la rapidité d'exécution de la tâche, de la quantité d'énergie utilisée, des coûts de communication et de la durée de vie de chaque caméra du réseau. Pour répondre à cet enjeu, il a été proposé de minimiser la consommation d'énergie cumulée dans toutes les caméras, les nœuds Fogs et le serveur Cloud, et aussi de garantir un temps de traitement *minimal* dans l'ensemble du système.

La fonction de performance définie permet à une caméra de calculer la fréquence d'images qu'elle peut fournir à la tâche de suivi, ou encore de calculer et déterminer la résolution associée à la cible visée. Dans ce cadre, il a été attribué des poids plus élevés aux deux paramètres qui contrôlent les performances du système, et une seconde fonction, de sélection des nœuds Fogs, a été introduite.

Rappelons rapidement que l'exécution du scénario d'allocation combine trois phases : (1) *la détection* déclenchée par l'arrivée d'une cible à suivre ; (2) *la négociation* où les nœuds Fog communiquent entre eux afin de sélectionner le nœud qui sera délégué pour cette phase ; et (3) *l'allocation*, où la caméra présentant la plus grande valeur de fonction de sélection est sélectionnée. Après le scénario d'allocation dans notre cas l'algorithme résulte en une complexité qui dépend du nombre de tâches $n$, du nombre de caméras $(NbC)$ et du nombre de Fog $(NbF)$.

Une approche alternative a été proposé consiste à présenter une autre méthode permettant une assignation dynamique, optimale et automatique des tâches de suivi sur les éléments de traitement disponibles d'un système de vidéosurveillance. La problématique a été modélisée en utilisant la technique de programmation linéaire en nombres entiers (Integer Linear Programming, ILP), en raison de sa nature optimale et de la disponibilité des outils de résolution. L'évaluation de cette méthodologie a donné des résultats intéressants et efficaces sur le temps d'exécution pour diverses *instances réalistes*, et a confirmé que les modèles proposés constituent une solution efficace pour les problèmes d'ampleur moyenne que l'on retrouve fréquemment dans les systèmes de vidéosurveillance réalistes.

Certaines variables dans le modèle étudié sont booléennes, le problème est dès lors NP difficile et la procédure de sa résolution peut être lente (proportionnellement au nombre de ces variables), notamment si l'exécution de la tâche de résolution sur un autre périphérique que le Cloud est souhaité. Ainsi, pour les applications incluant assez de capteurs et de composants de Fog, et nécessitant plus de flexibilité pour la tâche de résolution, l'utilisation de méthodes stochastiques plus que le modèle linéaire est conseillée. Nos futurs travaux consisteront à explorer de telles solutions.

## 8.2 Perspectives

### 8.2.1 Architecture Fog/Cloud dynamique pour des applications multiples

Il est parfois nécessaire dans des architectures Fog/Cloud, y compris celles dédiées à la vidéo surveillance, d'exécuter en même temps des applications multiples qui manipulent les mêmes données ou des données complètement différentes. En effet, dans des systèmes distribués de vidéosurveillance tels que celui décrit dans cette thèse, nous pourrions facilement imaginer un sous-ensemble de caméras dédié au suivi de cibles, alors qu'un autre sous ensemble serait chargé d'exécuter des tâches de surveillance et de détection de fumée par exemple.

Dans un tel cas, les méthodes et algorithmes proposés pourraient éventuellement être adaptés, même si l'hétérogénéité augmenterait la complexité et donc les temps d'exécution. Une difficulté doit néanmoins être surmontée pour que notre approche reste applicable ; il s'agit de la non-régularité (en termes de taille) des *bitstream* partiels des tâches. En effet, comme les zones reconfigurables doivent être spécifiées avant l'exécution, il ne sera plus possible d'envisager des zones régulières pouvant accueillir n'importe quelle tâche.

Il s'agit là d'un problème intéressant, bien que complexe, qui mériterait sa place dans nos travaux futurs.

### 8.2.2 Virtualisation logicielle / matérielle

Pour qu'une plateforme ou une méthode soit utilisée à plus ou moins grande échelle, il est impératif de simplifier et d'automatiser au maximum son utilisation. Ceci peut être réalisé par la dissimulation du plus grand nombre de détails possible, notamment architecturaux, à l'utilisateur ( ou programmeur).

Un des aspects intéressants dans un système de vidéosurveillance hétérogène doté de caméras munies de processeurs et d'autres de FPGAs est de permettre au programmeur d'exécuter les différentes tâches, sous forme logicielle ou matérielle, de façon totalement transparente. Autrement dit, avoir à disposition un langage permettant d'évoquer une implémentation sans se soucier du fait qu'elle sera réalisée en logiciel ou en matériel. Ceci passerait certainement par une sorte de virtualisation des ressources de calcul en proposant à la fois un langage adapté ainsi que des outils de compilation spécifiques. Ceci constitue une piste prometteuse pour les futurs travaux.

### 8.2.3 Sécurisation des stockages et de la communication

Les applications mettant en œuvre des communications réseau, notamment sans fil, sont généralement sujettes à des attaques malveillantes. Cela est particulièrement vrai pour les applications de surveillance vidéo. L'aspect « humain » et le droit de respect de la vie privée rendent ces aspects liés à la sécurité encore plus importants dans les systèmes que nous avons utilisés dans cette étude.

Il s'agit là d'une perspective des plus intéressantes surtout en raison de la disponibilité de FPGAs dans nos systèmes et aussi du fait que les ports et protocoles de communication peuvent être implémentés assez librement sur de tels architectures.

Réserver certaines ressources pour l'implémentation de fonctionnalités de sécurité tels que des protocoles particuliers ou des tâches de cryptage pourra donner une valeur ajoutée non négligeable à nos travaux.

# REFERENCES

**[ACCPSV02]** L. Albani, P. Chiesa, D. Covi, G. Pedegani, A. Sartori, M. Vatteroni, VISoc: a smart camera SoC, in: 28th European Solid-State Circuits Conference, Florence, Italy, 2002, pp. 367–370.

**[AIM10]** Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer networks 54(15), 2787–2805 4. Bailey, D.G, 2010.

**[AM00]** A. Moini, Vision chips/ by Alireza Moini, Kluwer Academic Boston, London, 2000.

**[ASJ09]** A. Akoglu, A. Sreeramareddy, and J. G. Josiah, "Fpga based distributed self healing architecture for reusable systems," Cluster Computing, vol. 12, pp. 269–284, September 2009.

**[BA12]** Bauer, E., & Adams, R. Reliability and availability of cloud computing: John Wiley & Sons, 2012.

**[BC96]** B. Carpenter, "Architectural Principles of the Internet," RFC 1958 (Informational), Jun. 1996, updated by RFC 3439. [Online]. Available: http://www.ietf.org/rfc/rfc1958.txt

**[BHU03]** J. Becker, M. Hübner, M. Ullmann: "Real-Time Dynamically Run-Time Reconfiguration for Power-/Cost-optimized Virtex FPGA Realizations", VLSI 03, Darmstadt, Sep 2003.

**[BMKZ14]** Bilal, K, Malik, S., Khan, S. U. & Zomaya. A, 'Trends and Challenges in Cloud Data Centers,' IEEE Cloud Computing Magazine, vol. 1, no. 1, pp. 10-20, 2014.

**[BMX14]** A.N. Bruno, M. Marc, and N. Xuan-nam. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications Surveys and Tutorials, 16(3):1617–1634, February 2014.

**[CBD]** S.J. Carey, D.R. Barr, P. Dudek, Low power high-performance smart camera system based on SCAMP vision sensor, J. Syst. Architect. ISSN 1383–7621, URL http://www.sciencedirect.com/science/article/pii/S13837621130.

**[CC83]** V. G. Cerf and E. Cain, "The dod internet architecture model", Computer Networks, vol. 7, pp. 307–318, 1983. [Online]. Available: http://dblp.uni-trier.de/db/journals/cn/cn7.html#CerfC83

**[DCR08]** Devarajan, D., Cheng, Z. & Radke, R.: Calibrating distributed camera networks, Proceedings of the IEEE 96(10), 1625–1639, 2008.

**[DGB11]** Bailey, D.G.: Design for embedded image processing on FPGAs. John Wiley & Sons, 2011.

**[DGL04]** Lowe, D.G. Distinctive image features from scale-invariant key points, International Journal of Computer Vision 60, 91–110, 2004.

**[DS11]** Debyo Saptono. « Conception d'un outil de prototypage rapide sur le FPGA pour des applications de traitement d'images », Université de Bourgogne, 2011.

**[FRO]** Fabio D. Real de Oliveira, « Conception d'une méthodologie d'implémentation d'applications de vision dans une plateforme hétérogène de type Smart Camera »,

**[HCG10]** Ko. S Y. Hoque, I. Cho, B & Gupta. I, 'Making cloud intermediate data fault-tolerant', In Proceedings of the 1st ACM symposium on Cloud computing, pp. 181-192, 2010.

**[HDDCH07]** Van Den Hengel, A. Dick, A. Detmold, H. Cichowski, A.&Hill, R Finding camera over lap in large surveillance networks, in Proceedings of the 8th Asian conference on Computer vision -Volume Part I, Springer-Verlag, Berlin, Heidelberg, pp. 375–384, 2007.

**[HPFA07]** S. Hengstler, D. Prashanth, S. Fong et H. Aghajan: MeshEye: A Hybrid-Resolution Smart Camera Mote for Applications in Distributed Intel-ligent Surveillance. In International Symposium on Information Processing in Sensor Networks (IPSN), pages 360{369, 2007.

**[JB13]** J. Barr, "Build 3D Streaming Applications with EC2's New G2 Instance Type," Nov 2013.

**[KBNH14]** A. Keller, D. Borkmann, S. Neuhaus, and M. Happe, "Self-awareness in Computer Networks," Hindawi International Journal of Reconfigurable Computing, 2014.

**[KK10]** Koren, I., & Krishna, C. M. Fault-tolerant systems: Morgan Kaufmann, 2010.

**[KSDH06]** R. Kleihorst, B. Schueler, A. Danilin et M. Heijligers, « Smart Camera Mote with High Performance Vision System ». In Workshop on Distributed Smart Cameras (DSC), 2006.

**[LBMYB06]** P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford. Invited paper : Enhanced architectures, design methodologies and cad tools for dynamic reconfiguration of Xilinx FPGAs. In Field Programmable Logic and Applications, FPL'06, pages 1–6, 2006.

**[LLWO04]** C.H. Lin, T. Lv, W. Wolf, I.B. Ozer, A peer-to-peer architecture for distributed real-time gesture recognition, in: Proceedings of the IEEE International Conference on Multimedia and Expo, 2004.

**[LXG09]** Loy. C, C. Xiang, T. & Gong. S, « Modelling activity global temporal dependencies using time delayed probabilistic graphical model, in IEEE International Conference on Computer Vision », pp. 120–127, 2009a.

**[LXG09]** Loy. C, C. Xiang, T. & Gong. S, « Multi-camera activity correlation analysis, in IEEE Conference on Computer Vision and Pattern Recognition, pp. 1988–1995, 2009b.

**[MB99]** T. Moorhead, D. Binnie, Smart CMOS camera for machine vision applications, in: IEEE Conference on Image Processing and its Applications, Manchester, UK, 1999, pp. 865–869.

**[MD06]** Marinakis, D. & Dudek, G. A practical algorithm for network topology inference, in Proceedings of IEEE International Conference on Robotics and Automation., pp. 3108 –3115, 2006.

**[MDP07]** R. Mosqueron, J. Dubois et M. Paindavoine, « High-Speed Smart Camera with High Resolution ». EURASIP Journal on Embedded Systems, p 16, 2007.

**[ME03]** Makris. D, & Ellis. T, « Automatic learning of an activity-based semantic scene model, in Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance », IEEE Computer Society, Los Alamitos, CA, USA, pp. 183–188, 2003.

**[MEB04]** Makris, D. Ellis, T. & Black. J « Bridging the gaps between cameras, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, pp. 205–210, 2004.

**[MGP08]** Matthew G. Parris. Optimizing Dynamic Logic Realizations For Partial Reconfiguration Of Field Programmable Gate Arrays. B.S. University of Louisville .2008.

**[MR03]** F. Maraninchi and Y. Remond, “Mode-automata: a new domain-specific construct for the development of safe critical systems,” Science of Computer Programming, vol. 46, 2003.

**[MS15]** M. Staveley, “Applications that scale using GPU Compute,” in Azure Con 2015, August 2015.

**[MSK07]** Mandel, Z., Shimshoni, I. & Keren, D. Multi-camera topology recovery from coherent motion, in First ACM/IEEE International Conference on Distributed Smart Cameras, pp.243– 250, 2007.

**[MV12]** Vanneschi Marco. High performance computing systems and enabling platforms course notes, 2011/2012.

**[MVVL02]** J.-Y. Mignolet, S. Vernalde, D. Verkest, R. Lauwereins: "Enabling hardware-software multitasking on a reconfigurable computing platform for networked portable multimedia appliances"; Int´l. Conf. on Engineering of Reconfigurable Systems and Algorithms, Las Vegas, USA. June 25-27, 2002.

**[NB07]** Baaziz N., (2007), La vidéosurveillance automatique : sécurisation du contenu et traitements coopératifs, Rapport de recherche, RR 07/06-1, Juin 2007, université du Quebec en Outaouais

**[NTH08]** M. Nick, A. Tom, and B. Hari. Openflow: Enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review, pages 69–74, 2008.

**[NTL11]** X. Y. Niu, K. H. Tsoi, and W. Luk, "Reconfiguring distributed applications in fpga accelerated cluster with wireless networking," in International Conference on Field Programmable Logic and Applications (FPL), pp. 545 –550, 2011.

**[OLQWYJ14]** J. Ouyang, S. Lin, W. Qi, Y. Wang, B. Yu, and S. Jiang, "SDA: Software Defined Accelerator for Large Scale DNN Systems," in Hot Chips 2014, August 2014.

**[ONF]** Open network foundation (onf). software defined networking: the new form for networks [eb/ol]. https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf

**[PCC14]** A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. R. Larus, E. Peterson, G. Prashanth, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A Reconfigurable Fabric for Accelerating Large Scale Data center Services," in International Symposium on Computer Architecture (ISCA), 2014.

**[PTG10]** J.-M. Philippe, B. Tain, and C. Gamrat, "A self-reconfigurable fpga based platform for prototyping future pervasive systems," in Evolvable Systems: From Biology to Hardware,

ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, vol. 6274, pp. 262–273, 2010.

**[RDC08]** R.J. Radke, D. Devarajan, Z. Cheng, Calibrating distributed camera networks, in: Proceedings of the IEEE 96 (10) (2008).

**[RJR10]** Radke, R. J. A survey of distributed computer vision algorithms, in Handbook of Ambient Intelligence and Smart Environments, Springer US, pp. 35–55, 2010.

**[RW08]** B. Rinner and W. Wolf, "An Introduction to Distributed Smart Cameras," Proceedings of the IEEE, vol. 96, no. 10, Oct 2008.

**[SHB09]** C. Schuck, B. Haetzer, and J. Becker, "An interface for a decentralized 2d reconfiguration on xilinx virtex-fpgas for organic computing," Int. J. Reconfig. Comput., vol. 2009, pp. 7:3–7:3, January 2009.

**[SMA118]** Hugo Sbai, Samy Meftali, Djamel Aouali, « A Survey of Video Tracking Algorithms with different Implementations : A comparative Approach », The International Conference on Electrical and Electronics Engineering (ICEEE), february 2018. New York, USA.

**[SMA218]** Hugo Sbai, Samy Meftali, Djamel Aouali, « Dynamic Tasks Allocation in a Fog/Cloud Video Surveillance System », DRASIE Conference, february 2018, Melbourne, Australia.

**[SMA318]** Hugo Sbai, Samy Meftali, Djamel Aouali, « An Optimal Allocation of Tracking Tasks in a SW/HW Fog/Cloud Based Distributed Video Surveillance System », Journal of Communications, mars 2018.

**[SMAW18]** Hugo Sbai, Samy Meftali, Djamel Aouali, Pamela Wattebled, « Dynamic and Load Balanced Video Surveillance System based on a FoG/Cloud Architecture », The 3rd IEEE International Conference on Fog and Edge Mobile Computing (FMEC 2018). Barcelona, Spain, April 2018.

**[SSSMC17]** Sanjay Singh, Sumeet Saurav, Ravi Saini, Atanendu S. Mandal, Santanu Chaudhury, (2017), FPGA-based Smart Camera System for Real-time Automated Video Surveillance, Communications in Computer and Information Science book series (CCIS, volume 711).

**[SZK15]** Szewczyk, Roman and Zieliński, Cezary and Kaliczyńska, Małgorzata: « The Architecture of an Embedded Smart Camera for Intelligent Inspection and Surveillance », January 2015.

**[TDG05]** Tieu.K, Dalley.G & Grimson.W, « Inference of non-over lapping camera net work topology by measuring statistical dependence, in Tenth IEEE International Conference on Computer Vision », Vol. 2, pp. 1842–1849, 2005.

**[TM13]** C. Trabelsi, S. Meftali. Contrôle materiel des systems partiellement reconfigurables sur FPGA : de la modélisation à l'implémentation. Université de Lille. 2013.

**[TMAP14]** Tessens, L.; Morbée, M.; Aghajan, H. and Philips, W., (2014) "Camera selection for tracking in distributed smart camera networks", ACM Trans. on Sensor Networks, 10(2): pp. 1-33.

**[TMD12]** C. Trabelsi, S. Meftali, and J.-L. Dekeyser. Semi-distributed control for fpga-based reconfigurable systems. In 15th Euromicro Conference on Digital System Design, 6, 64, 79-2012.

**[TRK06]** S. Toscher, T. Reinemann, and R. Kasper, "An adaptive fpga-based mechatronic control system supporting partial reconfiguration of controller functionalities," in Proceedings of the first NASA/ESA conference on Adaptive Hardware and Systems, ser. AHS '06, Washington, DC, USA, pp. 225–228, 2006.

**[UGHB04]** M. Ullmann, B. Grimm, M. Huebner, J. Becker: "An FPGA Run-Time System for Dynamical On-Demand Reconfiguration", RAW04, Santa Fé, USA. Apr. 2004.

**[VSCWS06]** S. Velipasalar, J. Schlessman, C.-Y. Chen, W. Wolf, J.P. Singh, SCCS: A scalable clustered camera system for multiple object tracking communicating via message passing interface, in: Proceedings of the IEEE International Conference on Multimedia and Expo, 2006.

**[WTG10]** Wang. X, Tieu. K & Grimson W.E.L. « Correspondence-free activity analysis and scene modeling in multiple camera views, IEEE Transactions on Pattern Analysis and Machine Intelligence » 32, 56–71, 2010.

**[WW09]** W. Wolf, Towards pervasive smart camera networks, 2009. https://pervasive.aau.at/publications/pdf/Rinner_MCNBook2009.pdf

**[YLL15]** S. Yi, C. Li, and Q. Li, (2015), A survey of fog computing: Concepts, applications and issues, in Proceedings of ACM Workshop Mobile Big Data, Hangzhou, China, 2015, pp. 37-42.

**[ZB12]** A. A. Zarezadeh and C. Bobda, "Hardware Middleware for Person Tracking on Embedded Distributed Smart Cameras," Hindawi International Journal of Reconfigurable Computing, Jan 2012.

**[ZMM10]** Zhao, W, Melliar-Smith, PM, & Moser. L E, 'Fault tolerance middleware for cloud computing', In IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 67-74, 2010.