$N°$d'ordre : 3976

THÈSE

présentée à

L'UNIVERSITÉ BORDEAUX I

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET D'INFORMATIQUE

Par **Maïssa MBAYE**

POUR OBTENIR LE GRADE DE

DOCTEUR

SPÉCIALITÉ : **Informatique**

Les systèmes cognitifs dans les réseaux autonomes : une méthode d'apprentissage distribué et collaboratif situé dans le plan de connaissance pour l'auto-adaptation

---

Soutenue le : 17 Décembre 2009

Après avis des rapporteurs :

Nazim Agoulmine	Professeur, Université d'Evry Val d'Essonne
José Neuman Da Souza ..	Professeur, Université Fédérale du Cearà, Brésil

Devant la commission d'examen composée de :

Nazim Agoulmine	Professeur, Université d'Evry Val d'Essonne
André Cotton	Directeur de recherche Thalès Communication
José Neuman Da Souza ...	Professeur, Université Fédérale du Cearà, Brésil
Francine Krief	Professeur, ENSEIRB-MATMECA (*Directrice de thèse*)
Mohamed Mosbah	Professeur, ENSEIRB-MATMECA (*Président du Jury*)
Henry Soldano	Maître de conférences-HDR, Université Paris 13

Titre : *Les systèmes cognitifs dans les réseaux autonomes : une méthode d'apprentissage distribué et collaboratif situé dans le plan de connaissance pour l'auto-adaptation*

Résumé : L'un des défis majeurs pour les décennies à venir, dans le domaine des technologies de l'information et de la communication, est la réalisation du concept de réseaux autonomes. Ce paradigme a pour objectif de rendre les équipements réseaux capables de s'autogérer, c'est-à-dire qu'ils pourront s'auto-configurer, s'auto-optimiser, s'auto-protéger et s'auto-restaurer en respectant les objectifs de haut niveau de leurs concepteurs. Les architectures majeures de réseaux autonomes se basent principalement sur la notion de boucle de contrôle fermée permettant l'auto-adaptation (auto-configuration et auto-optimisation) de l'équipement réseau en fonction des événements qui surviennent sur leur environnement. Le plan de connaissance est une des approches, privilégiées ces dernières années par le monde de la recherche, qui suggère l'utilisation des systèmes cognitifs (l'apprentissage et le raisonnement) pour fermer la boucle de contrôle. Cependant, bien que les architectures majeures de gestion autonomes intègrent des modules d'apprentissage sous forme de boite noire, peu de recherches s'intéressent véritablement au contenu de ces boites. C'est dans ce cadre que nous avons fait une étude sur l'apport de l'apprentissage et proposée une méthode d'apprentissage distribué et collaboratif. Nous proposons une formalisation du problème d'auto-adaptation sous forme d'un problème d'apprentissage d'état-actions. Cette formalisation nous permet de définir un apprentissage de stratégies d'auto-adaptation qui se base sur l'utilisation de l'historique des transitions et utilise la programmation logique inductive pour découvrir de nouvelles stratégies à partir de celles déjà découvertes. Nous définissons, aussi un algorithme de partage de la connaissance qui permet d'accélérer le processus d'apprentissage. Enfin, nous avons testé l'approche proposé dans le cadre d'un réseau *DiffServ* et montré sa transposition dans le cadre du transport de flux multimédia dans les réseaux sans-fil 802.11.

Mots clés : *réseaux autonomes, auto-adaptation, plan de connaissance, systèmes cognitifs, apprentissage artificiel, apprentissage distribué, apprentissage collaboratif, programmation logique inductive.*

Title : *Cognitive systems in autonomic networks : a distributed and collaborative learning method in knowledge plane for self-adapting function*

Abstract : One of the major challenges for decades to come, in the field of information technologies and the communication, is realization of autonomic paradigm. It aims to enable network equipments to self-manage, enable them to self-configure, self-optimize, self-protect and self-heal according to high-level objectives of their designers. Major architectures of autonomic networking are based on closed control loop allowing self-adapting (self-configuring and self-optimizing) of the network equipment according to the events which arise on their environment. Knowledge plane is one approach, very emphasis these last years by researchers, which suggests the use of the cognitive systems (machine learning and the reasoning) to realize closed control loop. However, although the major autonomic architectures integrate machine learning modules as functional block, few researches are really interested in the contents of these blocks. It is in this context that we made a study on the potential contribution of machine learning and proposed a method of distributed and collaborative machine learning. We propose a formalization of self-adapting problem in term of learning configuration strategies (state-actions) problem. This formalization allows us to define a strategies machine learning method for self-adapting which is based on the history of observed transitions and we use inductive logic programming to discover new strategies from those already discovered. We defined, also a knowledge sharing algorithm which makes network components collaborate to improve learning process. Finally, we tested our approach in *DiffServ* context and showed its transposition on multimedia streaming in 802.11 wireless networks.

Keywords : *autonomic networking, self-adaptating, knowledge plane, cognitive systems, machine learning, distibuted machine learning, collaborative machine learning, inductive logic programming.*

Dédicace

A mon cher père, que la terre lui soit légère, le 21 septembre 2008, TOUT a changé

A ma mère et ma grand-mère que j'aime tant !!

ils me manquent...

Remerciements

Je commencerai naturellement par remercier le professeur Francine Krief, pour son encadrement. Elle n'a ménagé ni son temps, ni son expérience, ni ses relations et même sa santé pour que ce travail aboutisse. J'ai pris beaucoup de plaisir à travailler avec une personne aussi sympathique.

Je tiens tout particulièrement à remercier M. Henry Soldano pour son aide et ses remarques si pertinentes.

J'adresse également mes plus sincères remerciements aux professeurs Nazim Agoulmine et José Neuman Da Souza qui m'ont fait l'honneur d'évaluer les travaux de cette thèse.

Je remercie aussi l'ensemble des membres du jury pour leur présence lors de ma soutenance.

Je voudrais remercier aussi tous les membres de l'équipe COMET et tous le personnel administratif du LaBRI.

Si je citais toutes les personnes que j'ai envi de remercier, cette section occuperait une bonne partie du document. Ceci étant dit, je remercie tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail, par leur aide, leurs encouragements ou par leur simple présence. En espérant que chacun s'y reconnaîtra...

Abréviations

AC Autonomic Communication

ACCA Autonomic Communication Coordination Action

ACF Autonomic Communication Forum

ACI Autonomic Computing Initiative

ANA Autonomic Network Architceture

AutoI Autonomic Internet

BA Behavior Aggregate

BIONETS Biologically-Inspired Autonomic Networks

CASCADAS Componentware for Autonomic, Situated-aware Communication and Dynamically Adaptable Services

CIM Common Information Model

DARPA Defense Advanced Research Projects Agency

DMTF Distributed Management Task Force

DSCP DiffServ CodePoint

DT DropTail

FEC Forward Error Correction

FET Future and Emergent Network Technologies

FIFO First In First Out

FP6-7 Framework Programme 6^{th} - 7^{th}

HAGGLE A novel Communication Paradigm for Autonomic Opportunistic Communication

IDS Intrusion Detection System

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

ILP Inductive Logic Programming

IP Internet Protocol

IST Information Society Technologies

JNI Java Native Interface

KIF Knowledge Interchange Format

MAPE Monitoring, Analyzing, Planning, Executing

MIB Management Information Base

PBM Policy-Based Management

PHB Peer-Hop Behaviour

PLI Programmation Logique Inductive (voir ILP)

QoS Quality of Service

RED Random Early Detection

SAC Situated and Autonomic Communication

SFQ Stochastic Fair Queue

SOA Service Oriented Architecture

SUO Standard Upper Ontology

SUO SAS Small Unit Operations Situation Awareness System

TCB Traffic Conditioning Block

TCE Traffic Classification Elements

TCO Total Cost Ownership

TF-AAS Task Force Autonomous & Autonomic Systems

Notations

P_r Critère de performance défini par le concepteur

E_v Environnement (ressources et éléments externes à un équipement à gérer)

E L'ensemble des exemples

e Un exemple, dans le sens apprentissage du terme

E^+ L'ensemble des exemples positifs

E^- L'ensemble des exemples négatifs

B_C L'ensemble des connaissances de base

R Ressource à optimiser suivant le critère de performance P_r

ω, a Une action ou opération de configuration

Ω l'ensemble des opérations de configuration possible sur l'équipement

Ω^* Union de l'ensemble des puissances cartésiennes k^{eme} de Ω

I L'espace des informations de la situation ou du contexte qui peuvent être obtenues de l'équipement.

I_k Un élément d'une partition de I.

ε_t Un état du système

Σ L'ensemble des états

Λ L'ensemble des actions

δ Fonction de transition

h Fonction à apprendre

π Politique qui permet le choix d'actions en fonction des états

ρ Fonction de récompense

$V^\pi(\varepsilon_t)$ récompense différée cumulative d'une politique π quelconque à partir d'un état ε_t

γ Le taux d'escompte

π^* Politique optimale d'un processus d'apprentissage par renforcement

C Un concept

U Un ensemble universel

Liste des publications

Journaux et chapitres de livre

- "*The Knowledge Plane in Autonomic Networks*", **Maïssa Mbaye**, Francine KRIEF, journal International Transactions on Systems Science and Applications (ITSSA), revised version of ISSN paper. **(A paraitre)**
- "*A Collaborative Knowledge Plane for Autonomic Networks*", **Maïssa Mbaye**, Francine Krief, Chapitre de Livre "Autonomic Communication" Vasilakos, A.V. ; Parashar, M. ; Karnouskos, S. ; Pedrycz, Springer ISBN : 978-0-387-09752-7, pages 60-90, Septembre 2009
- "*Autonomic Networking : The Knowledge Plane*", **Maïssa Mbaye**, Francine Krief, System and Information Sciences Notes (ISSN 1753-2310), CODS Series, SIWN - Vol. 1, No. 3, July 2007, pp. 297-302(Chengdu-China)

Conférences internationales avec actes

- "*A Collaborative strategy learning for distributed network self-configuring*" , **Maïssa Mbaye**, Francine Krief, Henry Soldano, In proceeding of 1st International IEEE Conference on Communications and Networking, 3 au 6 Novembre 2009 a Hammamet, Tunisie
- "*A Network Simulator for Autonomic Context-Aware Architecture*", **Maïssa Mbaye**, Francine Krief, The Third IEEE International Conference on New Technologies, Mobility and Security DEMO 2009 Session, 20 to 23 December 2009 in Cairo, Egypt.
- "*Apport de l'apprentissage dans les réseaux autonomiques*", **Maïssa Mbaye**, Francine Krief. Gestion de REseaux et de Services (GRES), Hammamet (Tunisie), L'adaptation dynamique des réseaux et des services, Hermès Science Publications, Volume 1 Novembre 2007
- "*Un plan de connaissance basé sur l'apprentissage pour les réseaux autonomes*", **Maïssa Mbaye**, Francine Krief, 22ème Congrès DNAC : De Nouvelles Architectures pour les Communications, 3 au 5 décembre 2008.
- "*Un module d apprentissage pour l'auto-configuration et l'auto-optimisation des réseaux IP offrant des garanties de qualité de service*", **Maïssa Mbaye**,

Francine Krief. 8èmes Journées Doctorales en Informatique et Réseau - Janvier 2007 (Marne la Vallée France)

- "*Apprentissage distribué et collaboratif de stratégies pour les réseaux autonomes*", **Maïssa Mbaye**, Francine Krief, Poster ´ NOTERE 2009 ´ Montréal, du 29 juin au 3 juillet

Table des figures

Table des matières

CHAPITRE 1

Introduction

Les premiers services téléphoniques étaient gérés par des opérateurs qui se chargeaient de mettre en relation manuellement les personnes qui voulaient communiquer. Durant les années 1920, le réseau s'est agrandi et est devenu tellement complexe que la gestion classique manuelle des communications était devenue impossible. Il a donc fallu retirer le plus possible la main de l'être humain dans le cycle de gestion des communications. Le monde des technologies de l'information et de la communication en général, et des réseaux en particulier, est en train de vivre la même révolution.

Nous assistons à un développement sans précédent des nouvelles technologies. Les réseaux sont devenus accessibles depuis partout (réseaux ubiquitaires), les services disponibles sont de plus plus en variés et exigeants en termes de QoS (voix sur IP, vidéo à la demande, ...), les terminaux connectés de plus en plus nombreux et hétérogènes (Smartphones, ...). Et la tendance actuelle est à la convergence entre les différents réseaux existants.

La conséquence de cette évolution est la complexification croissante de la gestion du cœur du réseau qui doit prendre en compte les exigences des différents services utilisés par les terminaux communicants. En effet, la gestion classique des réseaux se faisait de manière centralisée et l'optimisation des paramètres de configuration était réalisée par des experts. Cependant, le nombre de paramètres à prendre en compte est devenu tellement grand qu'il devient impossible de le gérer de manière rigoureuse à l'échelle de la compétence humaine. La correction des pannes devient aussi un vrai défi car le cœur du réseau s'est aussi complexifié avec la traversée de réseaux de plus en plus hétérogènes.

Durant ces dernières années, les capacités (comme l'avait prévu la loi de Moore)

des ordinateurs de bureau ont considérablement augmenté et le prix de ces équipements a baissé. Leur puissance de calcul se rapproche maintenant de celle des gros serveurs. Les attaques à partir de postes clients constituent maintenant une menace sérieuse même pour les experts de sécurité les plus aguerris. Partout dans le monde, des ordinateurs sont transformés en zombies rendant difficile la remontée jusqu'au véritable coupable. La seule véritable option pour protéger les équipements réseaux, contre ces attaques, est celle de la prévention et de la détection automatiques et évolutives.

La configuration des équipements réseaux est devenue, également, un véritable casse-tête pour les administrateurs. La mise place de tests de stabilité du système après des opérations de configuration, de mise à jour et de mise à niveau sont de plus en plus complexes à faire. Cette complexité entraine souvent une hésitation à changer de systèmes même s'ils sont obsolètes. D'un autre coté, la formation des techniciens experts capables de gérer les équipements et optimiser leurs configurations est certainement une source de dépenses importante pour les entreprises et les opérateurs réseau. L'un des objectifs de la gestion des réseaux étant la diminution du TCO (Total Cost Ownership), la gestion par des experts semble augmenter les coûts avec les temps d'indisponibilités durant les diagnostics et les optimisations, les coûts de formation du personnel. En contre partie, la fiabilité du système diminue. En effet 40% des pannes de services sont dues à des problèmes de configurations de la part des opérateurs [Patterson 2002] qui doivent prendre des décisions rapides [Brown & Patterson 2001].

La gestion autonome est un concept qui vise à permettre aux équipements de se gérer automatiquement (autogestion) tout en respectant des politiques de haut niveau. Ce nouveau paradigme permettra de faire descendre la complexité de gestion au niveau des équipements et de réduire l'activité de l'être humain dans la boucle de gestion des réseaux. Ainsi les administrateurs se limiteront à la tâche de spécification des objectifs *business* : c'est l'équivalent de la révolution des années 20 de la téléphonie, mais cette fois-ci pour la gestion des réseaux de communications. Cette notion se base sur quatre fonctions de base qui sont l'auto-configuration, l'auto-optimisation, l'auto-protection et l'auto-restauration.

L'idée d'utiliser les systèmes cognitifs (apprentissage et raisonnement) capables de s'adapter automatiquement à leur environnement pour réaliser les fonctions au-

tonomes a fait son chemin. Elle a donné naissance au *plan de connaissance*. Le *plan de connaissance*, qui sera vu au chapitre 3, est une structure distribuée qui permet de gérer le réseau en utilisant la connaissance du réseau et des processus cognitifs au lieu de se concentrer sur les tâches de configuration.

Les dernières architectures de gestion autonomes, presque dans leur totalité, incluent des modules contenant des supports cognitifs sous forme d'un plan de connaissance. Ces outils sont utilisés pour fermer la boucle de contrôle laissant ainsi à l'administrateur un rôle de concepteur des politiques de haut niveau. Cependant, parmi ces architectures il n'y a presque pas d'études approfondies sur ce que peuvent apporter réellement ces outils en particulier l'apprentissage. Ils sont en général représentés sous forme de boites noires dans les architectures en faisant des suppositions sur leurs aptitudes ou la possibilité d'utiliser un outils, en particulier qui semble en apparence correspondre à la problématique traitée. En particulier, beaucoup d'architectures existantes supposent que l'apprentissage permet de faire de l'auto-adaptation sans vraiment identifier les outils qui pourraient être utilisés et les éventuels problèmes que ces outils ajouteraient par rapport aux problématiques réseaux.

Cette thèse s'intéresse à cette problématique, à savoir dans quelle mesure les outils d'apprentissage peuvent être utilisés dans le cadre de l'auto-adaptation. Nous proposons ainsi une étude et une série de propositions d'algorithmes permettant de réaliser l'auto-adaptation.

Le présent document de thèse est organisé en quatre chapitres dont les contenus respectifs sont :

Chapitre 2 : Ce chapitre est un état de l'art de la gestion autonome. Il présente d'abord l'historique et les motivations de la gestion autonome depuis ses débuts. Cet historique permet de montrer l'importance pour ne pas dire la nécessité de faire de la gestion autonome dans les réseaux de future génération. Ensuite, nous avons montré les différentes visions du domaine qui se concurrencent d'une part et se complètent d'autre part. Ces visions s'inspirent notamment du système nerveux autonome des mammifères supérieurs, de la société des insectes et des interactions entre les éléments d'un écosystème. Ce chapitre fixe, aussi, la terminologie générale se rapportant aux réseaux autonomes : *Autonomic Computing, Autonomic Net-*

working/Communication, ... Ce chapitre présente aussi, l'architecture de base de la gestion autonome selon la vision IBM avec les éléments de l'entité autonome : ressources gérées, interfaces de contact et gestionnaire autonome. Ce chapitre se termine par la présentation des projets de recherche autour de cette thématique. Cette présentation permet d'introduire le concept des réseaux autonomes en présentant le besoin d'utiliser les systèmes cognitifs pour réaliser les fonctions de la gestion autonome. Elle permet aussi de faire la transition vers le concept qui suggère l'utilisation d'outils cognitifs pour faire de l'auto-adaptation, à savoir le plan de connaissance. Le but général de ce chapitre est de situer le cadre et les objectifs de la thèse.

Chapitre 3 : Ce chapitre présente le plan ce connaissance en se basant pour une grande partie sur l'article de référence [Clark *et al.* 2003b]. Le plan de connaissance est, de manière résumée, une structure distribuée et hiérarchique qui se charge de la collecte des informations, de l'organisation des connaissances du réseau et de la prise de décision intelligente dans le réseau. Elle utilise les outils cognitifs que sont l'apprentissage et le raisonnement pour faire ces tâches. Ce chapitre montre la vision globale des différents attributs du plan de connaissance ainsi que les défis et les considérations architecturales s'y rapportant. Ensuite, nous présentons dans ce chapitre l'ensemble des concepts corrélés au plan de connaissance pour montrer la diversité des tâches que peut réaliser le plan de connaissance et les visions qui se dégagent de l'utilisation du plan de connaissance. Ces tâches s'organisent en deux grands groupes qui sont 1) l'utilisation du plan de connaissance comme système de gestion de base de connaissance (collecte, organisation et distribution de la connaissance), 2) comme système d'auto-adaptation, dans ce cas, il sert à fermer la boucle de contrôle de manière intelligente en utilisant l'apprentissage et le raisonnement. Enfin, dans ce chapitre nous avons positionné le plan de connaissance par rapport au concept *autonomique* pour faire apparaitre à quel point les deux domaines sont liés ou complémentaires. Ce chapitre nous permet de présenter le cadre de conception de la *boucle de contrôle fermée* que fournit le plan de connaissance. Elle nous permet ensuite de nous consacrer, dans la partie suivante, à l'étude et à la proposition d'une approche de réalisation de l'auto-adaptation basée sur un module d'apprentissage.

Chapitre 4 : Dans ce chapitre, nous avons commencé par poser le problème de l'auto-adaptation de manière formelle. Cette formalisation permet de mettre en évidence les différents paramètres importants mais aussi de discuter des conditions de décidabilité du problème. Après cette formalisation, nous avons identifié un objet pour l'apprentissage qui est celui d'apprendre des stratégies, une stratégie étant un ensemble d'actions à appliquer à un état pour améliorer le fonctionnement de l'équipement et améliorer l'utilisation de la ressource et/ou de la QoS. Ensuite, pour identifier les méthodes d'apprentissage intéressantes pour cette problématique, nous avons présenté l'apprentissage artificiel comme un problème d'amélioration de la performance au fur et à mesure de l'exercice d'une activité. Nous avons poursuivi par une classification synthétique des méthodes d'apprentissage suivant plusieurs dimensions(critères). Cette classification permet de faire apparaitre les types d'apprentissage dont les caractéristiques et/ou les motivations correspondent au moins en partie à celle de l'auto-adaptation. Nous avons, ensuite, choisi l'apprentissage par renforcement comme étant la problématique la plus proche de celle de l'auto-adaptation. En effet, l'apprentissage par renforcement peut être vu comme la tâche d'un agent qui apprend une séquence d'actions à effectuer à partir d'un état pour optimiser une récompense. Cet apprentissage est présenté en détail pour faire apparaitre au fur et à mesure les données du problème qui sont indisponibles dans un environnement réseau, et les hypothèses qui sont difficilement applicables dans le cas de l'auto-adaptation vue du coté réseau. Nous proposons, par la suite, un ensemble de mécanismes et d'algorithmes qui permettent de mettre en place un système d'apprentissage pour l'auto-adaptation dans les réseaux.

Chapitre 5 : Ce chapitre décrit notre proposition d'une approche d'auto-adaptation, qui s'inspire en partie sur la formalisation état-actions de l'apprentissage par renforcement et remplace la récompense qui sert à apprécier les états par une estimation de la probabilité de réussite à partir de l'historique des transitions observées. Nous montrons, ensuite, à quel point il peut être pertinent d'utiliser une telle grandeur. Cette estimation est, ensuite, utilisée pour classer les stratégies (couples état-actions) en exemples positifs et négatifs. Nous utilisons la programmation logique inductive pour découvrir des règles intéressantes durant le processus d'apprentissage. Ces nouvelles règles sans label (positif ou négatif) sont ensuite testées en

priorité par le processus d'auto-adaptation. Nous décrivons par la suite la méthode d'auto-adaptation et terminons par un exemple d'exécution. De plus, pour accélérer le processus et pallier au possible déséquilibre de la distribution des observations dans le réseau, nous proposons un algorithme de partage de la connaissance entre les éléments qui apprennent de manière distribuée et autonome. Ce partage permet également une évaluation distribuée de la connaissance. Cette partie se termine par un certain nombre de considérations sur la complexité en nombre de messages de l'algorithme de propagation de la connaissance à travers une vue située et la notion de voisinage qui a un impact sur cette complexité. Ce chapitre présente l'ensemble des propositions de la thèse. Il pose, également, les bases pour une application de la méthode sur des cas servant d'exemples d'application.

Chapitre 6 : Ce chapitre présente deux cas d'utilisation pour la méthode d'apprentissage auto-adaptative. Le premier cas d'utilisation concerne l'infrastructure *DiffServ* qui est l'un des deux environnements de QoS sous IP. Dans ce modèle, les nœuds sont divisés en deux catégories, les nœuds aux bordures du réseau et les nœuds dans le cœur du réseau. Ces nœuds sont configurés afin de remplir des tâches particulière : conditionnement sur les routeurs de bordure et application d'un traitement sur les flux dans le cœur du réseau. Une mauvaise configuration ou bien une configuration fixe peut entrainer des pertes de paquets dues aux changements des flux qui traversent le réseau et qui pourraient être évitées avec une configuration adaptative sans sur-allocation de ressource. Nous avons testé notre méthode d'apprentissage pour l'auto-adaptation dans cet environnement. Afin de réaliser les simulations, nous avons étudiés les simulateurs réseaux existants les plus connus et discuté de leur utilisation dans le cadre de la simulation de plateformes autonomes. Nous avons identifié un ensembles de critères que doivent remplir ces simulateurs pour pouvoir être utilisés dans un tel contexte. Nous avons ensuite décrit un scénario autour dans un environnement *DiffServ* pour la mise en œuvre de notre approche et ceci afin de montrer son intérêt dans un environnement réseau. Les simulations ont été effectuées avec JSIM comme environnement de simulation réseau et Aleph comme environnement de développement de programmation logique inductive. Les simulations montrent que l'apprentissage auto-adaptatif permet d'améliorer de manière automatique le débit des trafics AFxy, BE en évitant des pertes inutiles dues à

une mauvaise configuration de la taille des files d'attentes. Elles montrent aussi l'apport significatif de l'échange de connaissance sur la vitesse de stabilité du système. Le second cas d'utilisation est celui de l'adaptation d'une plateforme de diffusion de flux multimédia (vidéo) sur les réseaux 802.11. Cette plateforme dispose d'un certain nombre de mécanismes d'adaptation afin d'améliorer la qualité de la vidéo reçue. Cependant, il reste à évaluer l'application conjointe de ces différentes adaptations sur l'ensemble en terme de performance. Nous montrons ce qu'une transposition de la méthode que nous proposons pourrait apporter comme solution à ce problème. Une des adaptations qui est proposée dans ce cadre est la combinaison de la FEC avec l'adaptation du débit de la vidéo. Cependant, cette adaptation utilise des politiques fixes pour déterminer le taux de redondance. Ceci, a pour conséquence de ne pas garantir un *overhead* optimal. Nous proposons une amélioration qui consiste à prédire les taux de pertes et s'appuyer sur cette prédiction pour calculer la redondance exacte qu'il faut introduire afin d'anticiper les pertes.

Ce rapport de thèse finit par une conclusion et un ensemble de perspectives concernant nos travaux de recherches futures.

Première partie

Etat de l'art

CHAPITRE 2

Réseaux autonomes

Sommaire

2.1 Introduction

Les premiers services téléphoniques étaient gérés par des opérateurs qui se chargeaient de mettre en relation manuellement les personnes qui voulaient communiquer. Durant les années 1920, le réseau s'est agrandi et est devenu tellement

complexe que la gestion classique manuelle des communications étaient devenue impossible. Il a donc fallu retirer le plus possible la main de l'être humain dans le cycle de gestion des communications. Le monde des technologies de l'information et de la communication en général, et des réseaux en particulier, est en train de vivre la même révolution. En effet, depuis une vingtaine d'années nous assistons à un développement sans précédent des nouvelles technologies. Les réseaux sont devenus accessibles depuis partout (réseaux ubiquitaires), les services disponibles sont de plus en plus variés en termes d'exigences en QoS (voix sur IP, vidéo à la demande, ...) et les terminaux connectés de plus en plus nombreux et hétérogènes (domotique, Smartphones, ...). Et la tendance actuelle est à la convergence des différents réseaux existants.

La conséquence de cette évolution est la complexification croissante de la gestion du cœur du réseau en prenant en compte les exigences de différents services utilisés par les terminaux communicants. En effet, la gestion classique des réseaux se faisait de manière centralisée et l'optimisation des paramètres de configuration par des experts. Cependant, le nombre de paramètres à prendre en compte est devenu tellement grand qu'il devient impossible à gérer de manière rigoureuse à l'échelle de la compétence humaine. La correction des pannes devient aussi un vrai défi pour l'administrateur car le cœur du réseau s'est aussi largement complexifié avec la traversée de réseaux de plus en plus hétérogènes.

L'intégration des réseaux et services reste un verrou technologique à lever du fait que les réseaux sont conçus avec des objectifs et des contraintes différents.

De plus, durant les dernières années, les capacités (comme l'avait prévu la loi de Moore) des ordinateurs de bureau ont considérablement augmenté et leur prix a baissé. Leur puissance de calcul se rapproche maintenant de celle des gros serveurs. Les attaques à partir de poste client constituent, maintenant, une menace sérieuse même pour les experts en sécurité les plus aguerris. Partout dans le monde des ordinateurs sont transformés en zombies rendant difficile la remontée jusqu'au véritable coupable. La seule véritable option pour protéger les équipements réseaux, contre ces attaques, est celle de la prévention et de la détection automatiques et évolutives. Malgré l'évolution des IDS (Intrusion Detection System), il faut constamment mettre à jour les systèmes de détection manuellement. Entre 20% et 50% du temps est consacré à la détermination et à la résolution de problèmes, ce qui consti-

tue un manque à gagner important. La configuration des équipements réseaux est devenue également un véritable casse-tête pour les administrateurs. La mise place de tests de stabilité du système après des opérations de configuration, de mise à jour et de mise à niveau sont de plus en plus complexes à réaliser. Cette complexité entraine souvent une hésitation à faire évoluer le système et une préférence pour le garder même s'il est obsolète. D'un autre coté, la formation des techniciens experts capables de gérer les équipements et d'optimiser leurs configurations est certainement une sources de dépenses importantes pour les entreprises et les opérateurs réseau. L'un des objectifs de la gestion des réseaux étant la diminution du TCO (Total Cost Ownership), avec la gestion par des experts le coût semble augmenter avec les temps d'indisponibilité durant les diagnostics, les optimisations, les coûts de formation du personnel, bien que la fiabilité du système semble diminuer. En effet 40% des pannes de services sont dues à des problèmes de configuration de la part des opérateurs [Patterson 2002] qui doivent prendre des décisions rapidement [Brown & Patterson 2001]. Tous les problèmes qui viennent d'être cités montrent un besoin pour des fonctions de gestion automatisées comme :

- La possibilité de communiquer spontanément avec les objets voisins en utilisant différents types de réseaux (avec ou sans-fil)
- Le monitoring continue des ressources (matérielles et logicielles)
- La configuration et la protection de manière automatique pour maintenir les paramètres de configuration à des valeurs désirées.
- La réduction des coûts de gestion et de fonctionnement des réseaux

La gestion autonome est un concept qui vise à permettre aux équipements de se gérer automatiquement (autogestion) tout en respectant des politiques de haut niveau. Ce nouveau paradigme permettra de faire descendre la complexité de gestion au niveau des équipements et de réduire l'activité de l'être humain dans la boucle de gestion des réseaux. Ainsi, les administrateurs se limiteront à la tâche de spécification des objectifs *business* : c'est l'équivalent de la révolution des années 20 de la téléphonie, mais cette fois-ci pour la gestion des réseaux de communications.

Ce chapitre présente les concepts de la gestion autonome. La première section introduit les concepts de base de la gestion autonome, la seconde section présente les travaux existants dans ce domaine et, enfin, la troisième section discute des moyens pour parvenir à l'autonomie.

2.2 Concepts

2.2.1 Historique et vision de l'*autonomic*

Le concept ***autonomic*** est apparu à une période où le monde de la recherche a commencé à s'inspirer massivement du fonctionnement de la nature pour concevoir les systèmes de demain. En effet, durant la fin des années 1990, plus précisément en 1997, DARPA lance le projet Small Unit Operations Situation Awareness System (SUO SAS) [1]. Bien qu'ayant des objectifs militaires, ceux de concevoir des équipements de communication de localisation des soldats dans un champ de bataille, ce projet fut une des premières pierres dans la construction de l'*autonomic*. Il fut d'abord une preuve de concept et une orientation pour la recherche.

une réflexion fut initiée au début des années 2000 sur la *vision du futur* menée par des géants de l'industrie, notamment MCI Communications, HP et IBM. Cette nouvelle vision partait de l'observation du fonctionnement de la nature et mettait en compétition principalement trois modèles pour la conception des réseaux de demain :

2.2.1.1 Le système nerveux autonome des mammifères supérieurs

Ce système [2] est responsable de la régulation du fonctionnement des mécanismes internes de l'organisme et des organes qui fonctionnent de manière inconsciente. Il contrôle, par exemple, le rythme cardiaque, la température du corps humain sans que le mammifère n'ait à s'en soucier. Ce modèle utilise un système distribué et hiérarchique de contrôle qui s'appuie sur des capteurs et des réactions, ce qui est très proche du modèle d'organisation et de gestion des réseaux d'aujourd'hui. Le paradigme Capteur/Actionneur rappelle aussi le paradigme Agent/Manager qui était bien connu dans les modèles de gestion de réseaux. Cette vision a été soutenue surtout par IBM qui en 2001 créa l'ACI (Autonomic Computing Initiative) dans son manifeste [Horn 2001]. IBM suggère que les systèmes informatiques complexes doivent avoir des propriétés *autonomic*, c'est-à-dire qu'ils doivent pouvoir prendre

1. http ://www.darpa.mil/STO/strategic/suosas.html dernier accès 15 aout 2009
2. Il s'agit de l'*autonomic nervous system* qui donna son nom à l'**autonomic computing** / networking

en charge de manière automatique leur propre maintenance et optimisation. Ceci aura pour effet de réduire le travail de l'administrateur qui pourra se concentrer sur l'expression des objectifs que doit réaliser le système. Ce modèle est très largement développé par le monde de la recherche et de l'industrie car il est bien adapté à des systèmes de gestion active et de contrôle.

2.2.1.2 La société des insectes

Dans les sociétés, la colonie fonctionne avec des décisions individuelles apparemment intelligentes basées sur les interactions simples entre les membres de la colonie. Ainsi, le rôle clair de tous les membres de la colonie et les comportements homogènes des membres de même rôle permettent d'avoir une cohérence globale du fonctionnement du système. Ce modèle ressemble fort aux systèmes contrôlés par des politiques avec un grand nombre d'équipements homogènes. Il semble bien adapté à la conception de système stable, avec des réseaux homogènes comme des réseaux optiques ou IP.

2.2.1.3 Les écosystèmes de Prédateur / Proie et Producteur / Consommateurs

Les interactions entre les éléments d'un écosystème est un modèle qui se prête bien à la conception des réseaux et services associés. Dans ce modèle, les éléments qui offrent des services peuvent être à leur tour des demandeurs et le tout maintien un certain équilibre de l'écosystème. La coordination peut se faire au niveau d'un groupe (de *gnous* par exemple) cependant il n y a pas de supervision globale. Chaque élément a un rôle bien déterminé et dépend des membres de son groupe (les lions chassent ensembles), des autres espèces (les gnous dépendent des arbres qui fournissent des pâturages). Il y a une ressemblance avec les architectures SOA avec les services de base qui sont réutilisés par les autres services pour constituer des services plus complexes. D'un autre coté, ce modèle donne aussi une manière très séduisante de concevoir les communications dans les grands réseaux hétérogènes gérés par des opérateurs différents.

2.2.2 Terminologie

Le monde de la recherche et de l'industrie a tout de suite vu l'intérêt et la nécessité d'une gestion autonome. Le paradigme est donc vite devenu un objet de différentiation marketing des grandes entreprises, ce qui a eu comme effet l'apparition de nombreux termes pour ce concept. Du coté du monde de la recherche, de nouveaux termes sont apparus comme mots clés :

- Autonomic Computing
- Autonomic Networking/Communication
- Autonomic ans Autonomous Systems
- Cognitive Networks
- Ambian Intelligent (AmI) ecosystems
- Global service Ecosystems
- ...

Même s'il y a des variations sur le sens et la portée des termes précédemment cités, la vision et l'objectif sont partagés par tous : rendre les systèmes capables de s'*auto-gérer*. Cependant, il est à noter que le terme *autonomic communication* couvre davantage les domaines liés aux réseaux comme le note le site Autonomic-communication.org :

> *Autonomic Communication (AC) - a new communication paradigm to assist the evolution of communication networks towards functional adaptability, extensibility and resilience to a wide range of possible faults and attacks. Special emphasis is given on the grounding principles to achieve purposeful behavior on top of self-organization (including self-management, self-healing, self-awareness, etc.). Papers are solicited that study network element's autonomic behavior exposed by innovative (cross-layer optimized, context-aware, and securely programmable) protocol stack in its interaction with numerous often-dynamic network groups and communities. The goals are to understand how autonomic behaviors are learned, influenced or changed, and how, in turn, these affect other elements, groups and the network.*

Les principaux objectifs de cette vision sont définis par l'ACI [Horn 2001] regroupant les caractéristiques d'un système autonome autour de huit points :

1. Un système autonome doit disposer d'une connaissance détaillée de ses composants internes (*Know itself*).
2. Un système autonome doit savoir se configurer et se reconfigurer, sans intervention externe, pour s'adapter aux conditions variables et aux changements de son environnement.
3. Un système autonome doit essayer d'optimiser son fonctionnement d'une façon continue afin d'atteindre les objectifs de haut niveau.
4. Un système autonome doit être capable de se restaurer, suite à un dysfonctionnement, sans nuire aux données ni aux délais de leurs traitements.
5. Un système autonome doit être capable de détecter, identifier et se protéger contre les différents types d'attaques : assurer une sécurité de fonctionnement.
6. Un système autonome doit disposer d'une connaissance suffisante de son environnement et faciliter ainsi son adaptation.
7. Un système autonome doit pouvoir fonctionner dans un environnement hétérogène et implémenter des standards ouverts.
8. Un système autonome doit cacher la complexité de son fonctionnement à l'utilisateur : les prises de décision sont transparentes.

Cet ensemble, qui constitue une condition nécessaire pour que les systèmes soient autonomes, peut être traduit en propriétés [Sterritt & Bustard 2003] qui sont représentées par la figure 2.1 ([Sterritt & Bustard 2003]). Ces propriétés traduisent la vision, les objectifs, les attributs et les moyens de ce nouveau concept.

IBM a étendu sa vision [Kephart & Chess 2003, Jacob *et al.* 2002] du concept d'*autonomic computing* pour couvrir l'ensemble des domaines des nouvelles technologies de l'information et de la communication. Par la suite nous utiliserons les termes gestion autonome et réseaux autonomes de manière interchangeable pour exprimer l'application de la vision de la gestion autonome au sein des réseaux.

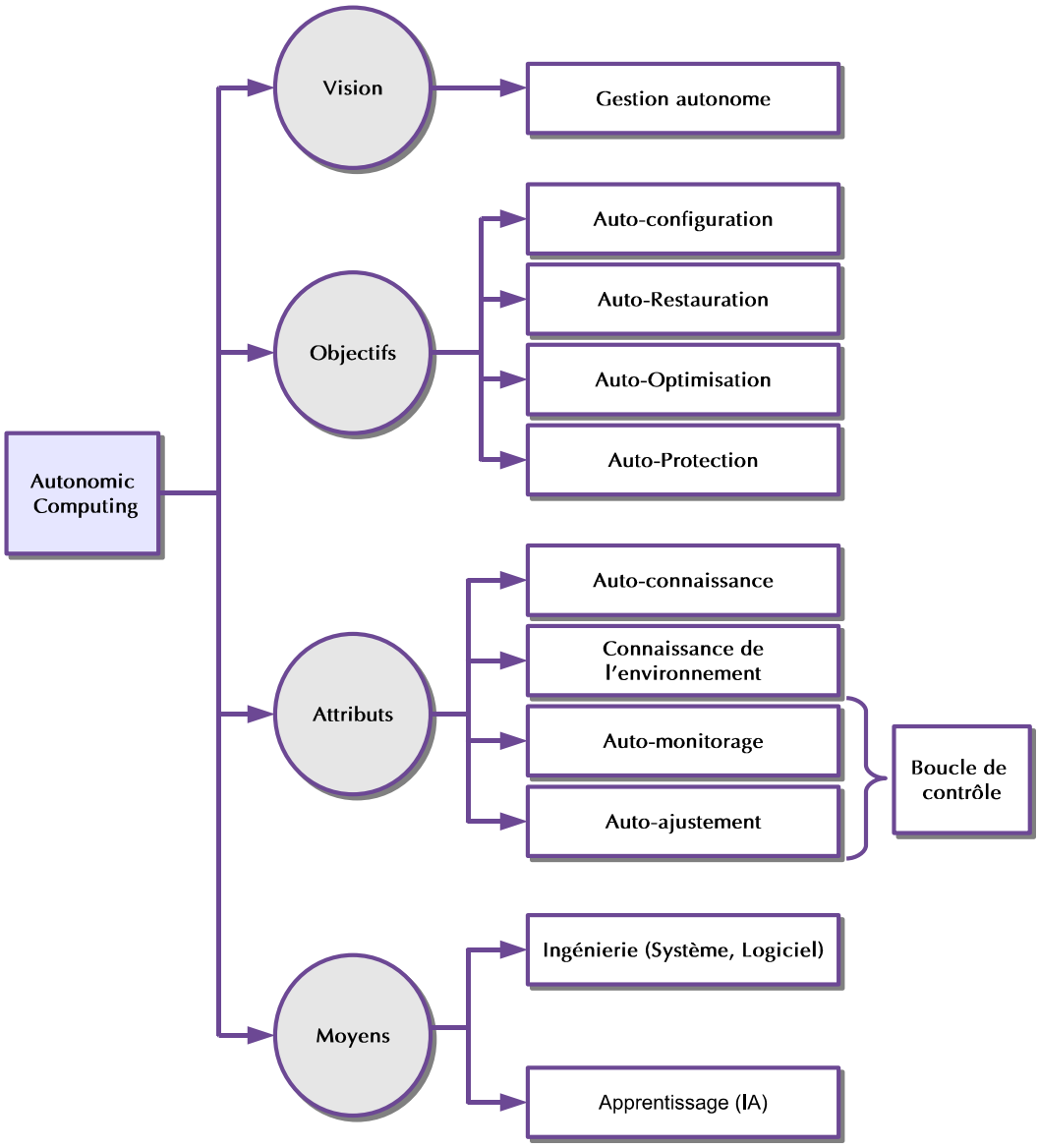

FIGURE 2.1 – Propriétés de la gestion autonome

D'autre part l'AC [3] (Autonomic Communication), ACF [4] (Autonomic Communication Forum) et TF-AAS [5] (Task Force Autonomous & Autonomic Systems) tentent de rassembler les idées autour de l'*autonomic networking* pour proposer des standards [6] .

2.2.3 Objectifs de la gestion autonome

2.2.3.1 Les fonctions de la gestion autonome

Dans un environnement autonome, les composants du réseau doivent remplir les fonctions d'autogestion : l'auto-configuration, l'auto-restauration, l'auto-optimisation et l'auto-protection (figure 2.2 [Kephart & Chess 2003]). Cette suite de fonctions est celle que doit offrir au minimum un système pour être autonome. Cependant, depuis 2001, cette liste a été fortement revue à la hausse avec de nouvelles fonctions qui semblent toutes aussi importantes les unes que les autres.

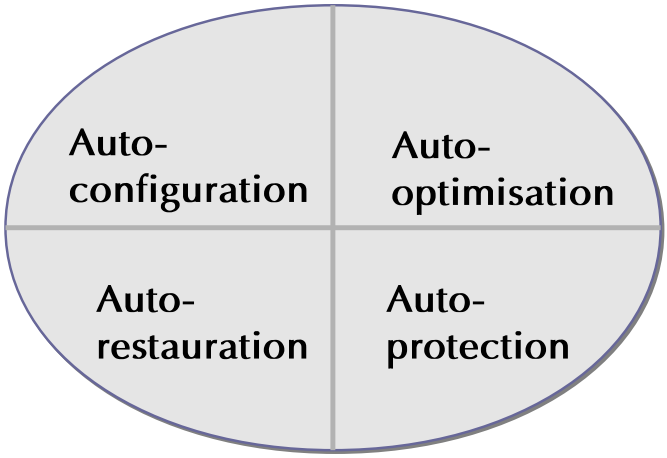

FIGURE 2.2 – Fonctions de base des systèmes autonomes

3. http ://www.autonomic-communication.org/ dernier accès 13 aout 2009
4. http ://www.autonomic-communication-forum.org/ dernier accès 13 aout 2009
5. http ://www.infj.ulst.ac.uk/ãutonomic/IEEE_TF-AAS/index.html
6. Malheureusement ces sites ne sont pas assez mis à jour, il est par conséquent difficile de s'attendre à une sortie prochaine de standards

2.2.3.2 L'auto-configuration : *Self-configuring*

Cet objectif se traduit par la capacité d'un système à se configurer de manière automatique pour s'adapter aux changements de l'environnement ou encore pour aider à la réalisation d'autres objectifs de gestion. Contrairement à la gestion traditionnelle où la fonction de configuration des systèmes avec de nombreux composants hétérogènes est pénible et demande beaucoup de temps aux experts, les systèmes autonomes s'auto-configurent de façon transparente vis-à-vis de l'opérateur en suivant des directives et des politiques de haut niveau. Les politiques spécifient le but à atteindre et non la manière avec laquelle les composants doivent se configurer pour l'atteindre. Un nouvel élément autonome ajouté au système autonome s'auto-configure d'une façon transparente et le reste du système s'adapte à sa présence en se configurant éventuellement.

Ainsi, les erreurs de configuration sont évitées et par conséquent les administrateurs gagnent un temps considérable en ne s'occupant que des objectifs *business*, une fois les politiques de configuration spécifiées.

2.2.3.3 L'auto-optimisation : *Self-optimizing*

L'ajustement des différents paramètres d'un système pour optimiser son fonctionnement et améliorer son rendement est complexe et peut engendrer des dysfonctionnements sans oublier le fait que peu de personnes ont la compétence pour réaliser à bien une telle opération.

Pour remédier à cela, les systèmes autonomes vont continuellement chercher à améliorer leur fonctionnement en saisissant les opportunités d'être plus performants. En même temps, ils surveillent leur environnement afin d'adapter leurs ressources de façon à assurer un fonctionnement optimal par rapport aux besoins spécifiés.

Dans l'immédiat, l'auto-optimisation peut se limiter à une meilleure utilisation des ressources mais avec le temps elle vise à acquérir une certaine expérience afin de s'auto-réguler et de faire le bon choix pour être plus performant et atteindre un objectif qui peut être une QoS optimale pour les utilisateurs et une utilisation optimale des ressources pour le système.

2.2.3.4 L'auto-restauration : *Self-Healing*

Les entités autonomes sont capables de détecter, diagnostiquer et réparer les pannes après les avoir localisées. Ainsi, des actions correctives vont être initiées pour débarrasser le système des anomalies et permettre le bon fonctionnement de ce dernier en évitant ou en réparant des pannes ou de mauvaises opérations qui ont pu avoir lieu.

Ces actions peuvent être le bon choix d'un patch ou encore la demande de création d'un nouveau patch par l'opérateur. Le Self-Healing peut être :

- **Réactif** : des mécanismes qui permettent le retour du système à un état souhaité, suite à un dysfonctionnement ;
- **Proactif** : superviser les signes vitaux d'un système afin de prédire d'éventuels problèmes.

2.2.3.5 L'auto-protection : *Self-Protecting*

L'auto-protection permet à l'entité autonome de se défendre contre des attaques accidentelles ou délibérées. Pour ce faire, les systèmes autonomes doivent acheminer la bonne information au bon moment et vers le bon utilisateur.

De plus, il est nécessaire d'éviter les intrusions ou les accès non autorisés en entreprenant des actions autonomes sans intervention humaine, ce qui permet ainsi de diminuer le coût de gestion de la sécurité de l'infrastructure réseau.

Ainsi, les systèmes autonomes s'auto-protègent de deux manières :

- En *défendant* le système contre les attaques et en *isolant* ou *réparant* les éléments sources de pannes non traitées par l'auto-restauration.
- En *anticipant* les attaques en analysant les rapports d'événements et en initiant les actions adéquates.

2.2.3.6 Autres auto-fonctions

Dans la foulée, d'autres auto-fonctions ont été définies en fonction des besoins. Nous pouvons en citer quelques uns :

- **Auto-localisation** (Self-locating) : Avec cette fonction, les nœuds autonomes établissent et mettent à jour dynamiquement un système de référence pour

identifier leurs voisins et localiser les ressources nécessaires au schéma de coordination [Agoulmine *et al.* 2006].

- **Auto-organisation** (Self-organizing) : Cette fonction exprime le fait que des éléments autonomes ont la capacité de décider de collaborer pour effectuer une tâche. Ces tâches peuvent nécessiter une collaboration parce qu'individuellement, les nœuds, soit ne peuvent pas les réaliser, soit obtiendraient de moins bons résultats [Prehofer & Bettstetter 2005].
- **Auto-conscience et auto-sensibilité au contexte** (Self and Context-aware) : Ces fonctions se réfèrent à la perception et à la réaction cognitives face à un événement qui survient Cet événement peut survenir au niveau interne comme au niveau de l'environnement externe au nœud *autonomic*. Le concept de *context-awareness* est le fondement des autres fonctionnalités opérationnelles : auto-configuration, auto-protection, auto-optimisation, ...
- **Auto-adaptation** (Self-adapting) : Un système qui s'auto-adapte est capable de détecter les changements qui subviennent localement et dans l'environnement et prendre des décisions de configuration localement pour avoir un fonctionnement cohérent et *optimisé*.
- ...

La diversification des fonctions n'est pas sans redondances et/ou des dépendances entre les fonctions autonomes. Pour ce qui est de la redondance, par exemple, l'auto-adaptation et l'auto-organisation réalisent l'ensemble des tâches des autres auto-fonctions. De la même manière, il y a une dépendance entre l'auto-localisation et l'auto-organisation.

L'intérêt peut-être, de ce re-découpage est la possibilité de s'attaquer à des problèmes plus précis de l'autonomie. Dans cette thèse nous nous intéressons donc essentiellement à la fonction d'auto-adaptation et celle d'auto-organisation pour réaliser un environnement autonome.

2.3 Architecture de la gestion autonome

Comme la figure 2.3 ([Kephart & Chess 2003, Jacob *et al.* 2002]) le montre, un élément autonome est essentiellement composé d'une ou de plusieurs ressources gérées, de capteurs, d'actionneurs, d'un gestionnaire autonome et, également d'une

boucle de contrôle qui doit être implémentée par ce dernier afin de réaliser les quatre objectifs de gestion autonome.

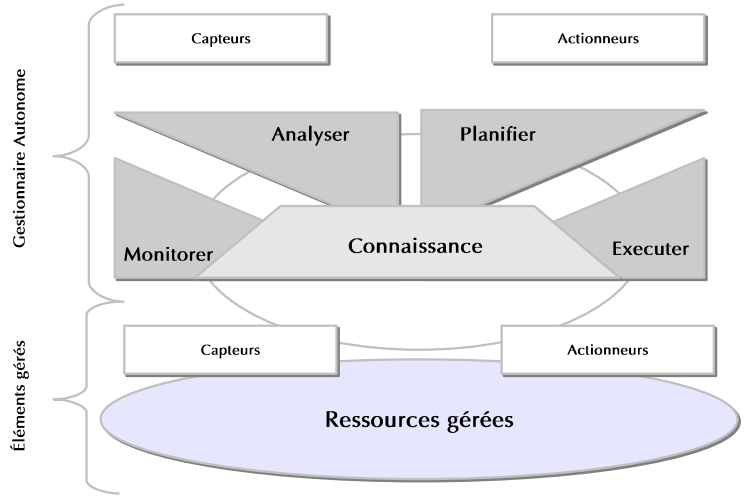

FIGURE 2.3 – Architecture d'un élément autonome

2.3.1 Ressources gérées

Une ressource gérée est un composant matériel ou logiciel qui peut être contrôlé. Contrairement à une ressource traditionnelle, un élément géré dans l'architecture autonome doit permettre à un gestionnaire de le superviser et de le contrôler grâce à des interfaces de gestion dont l'implémentation en points de contacts correspond à des *capteurs* et des *actionneurs*.

Un élément géré peut être une ressource unique telle qu'un serveur, une mémoire, une base de données, une application, une file d'attente, un service ou encore un ensemble de ressources tels qu'une pile de serveurs. Un gestionnaire autonome peut être, lui même, une ressource gérée ou faire partie d'un ensemble de ressources à gérer.

2.3.2 Interfaces de gestion

L'interface de gestion, pour contrôler un élément géré est organisée en capteurs et actionneurs. L'implémentation du comportement d'un capteur et d'un actionneur

spécifique à une ressource est assurée par des points de contacts appelés, également, point de communication qui permettent de réduire la complexité des différentes technologies pouvant être utilisées par les ressources, et ce en offrant aux gestionnaires autonomes une interface standardisée afin de contrôler les ressources dont il est responsable.

- **Capteur** : C'est l'intégration des différents mécanismes capables de collecter des informations sur l'élément géré. Ainsi, un capteur peut supporter un ensemble d'opérations de type *GET* qui permettent de retrouver des informations sur l'état de l'élément géré. Un capteur peut, aussi, disposer d'un ensemble d'événements de gestion qui se traduit par l'envoi de messages asynchrones qui font suite aux changements intervenus sur l'état de l'équipement géré. Ces deux types d'opérations sont caractérisés par différents paradigmes d'interaction. En effet, le premier adopte plutôt le style (*Retrieve-State*) permettant la récupération active de l'état de l'élément géré, tandis que le second utilise une interaction basée sur la réception passive d'une notification (*Receive-Notification*) afin de rendre compte des changements d'état.
- **Actionneur** : C'est l'agrégation des différents mécanismes utilisés pour changer le comportement d'un élément géré. Ainsi, un actionneur peut disposer d'un ensemble d'opérations de type *SET* qui permet la modification de la configuration de l'élément géré. Ce style d'interaction est basé sur l'exécution d'une opération de type *Perform-operation*. Un actionneur peut, également, disposer d'un ensemble d'opérations qui permettent à l'élément géré de faire des demandes, d'initier des requêtes ou encore de consulter d'autres entités grâce à un style d'interaction basé sur l'interpellation (type *Call-Out Request*).

Dans l'architecture de gestion autonome, les opérations des capteurs et des actionneurs sont liées entre elles par la *boucle de contrôle*. En effet, un changement de configuration par un actionneur doit être accompagné d'une notification de ce changement grâce à l'interface du capteur. Cette notification, qui peut être assimilée à un *feedback*, permettra aux processus tels que l'apprentissage d'évaluer l'impact de leurs actions.

2.3.3 Gestionnaires autonomes

Le gestionnaire autonome est le composant de l'architecture qui implémente le concept de boucle de contrôle qui permet de ne plus faire appel à l'intervention humaine. En effet, pour qu'un système soit capable de s'autogérer il doit disposer d'une méthode automatique lui permettant de collecter les informations dont il a besoin, de les analyser afin de déterminer s'il y a besoin ou non de changer un des attributs du système. Si c'est le cas, le gestionnaire autonome va créer un plan comportant une séquence d'actions qui va être exécutée pour réaliser les changements nécessaires.

2.3.3.1 Fonctions d'un gestionnaire autonome

La figure 2.3 est une représentation de la *boucle de contrôle* qui permet de réaliser les actions citées plus haut de manière automatique. Elle est formée d'un ensemble de fonctions indépendantes reliées par une connaissance :

- **Monitoring** : cette fonction offre des mécanismes qui réalisent la collecte, l'agrégation, la corrélation et le filtrage des données. Ces données peuvent porter sur la topologie, les paramètres de configuration, l'état fonctionnel, la capacité offerte ou encore les mesures provenant des ressources gérées dont le gestionnaire est responsable. La collecte des informations se fait par l'intermédiaire des points de contact, et plus précisément de l'interface capteur de la ressource. Ensuite, une quantité importante de données, qui peut être soit statique soit dynamique, est regroupée en symptômes qui vont être analysés dans une seconde étape.
- **Analyse** : cette fonction offre des mécanismes permettant d'observer et d'analyser des situations afin de déterminer si un changement doit être effectué pour satisfaire un objectif ou encore une directive de haut niveau. Ainsi, la fonctionnalité d'analyse est responsable du respect des politiques de haut niveau durant toute l'activité des composants réseau. Pour réaliser cette tâche, la fonction d'analyse implique l'utilisation des modèles comportementaux complexes et des techniques de prédiction afin de permettre au gestionnaire autonome d'apprendre à connaitre l'environnement dans lequel se trouvent les ressources gérées. Ces outils permettent, aussi, d'anticiper sur l'évolution de

l'environnement et sur les besoins en termes de ressources afin de s'y adapter au mieux. Le gestionnaire autonome doit alors être capable de faire des analyses, des raisonnements et des traitements complexes sur les symptômes reçus de la fonction de monitoring. Si des modifications doivent être effectuées, la fonction d'analyse va passer une requête à la fonction de planification. Cette requête contient la description des modifications qui doivent être effectuées afin d'atteindre un fonctionnement respectueux des objectifs de gestion.

- **Planification** : cette fonction offre des mécanismes permettant de planifier les actions nécessaires pour atteindre les objectifs fixés. Ces mécanismes de planification peuvent prendre la forme de règles de politique. Cette fonction permet la création ou la sélection de procédures capables de mettre en œuvre les changements souhaités par la fonction d'analyse sur les ressources gérées. La planification peut avoir plusieurs formes allant d'une simple commande à un enchainement ordonné de tâches. Les modifications ainsi planifiées peuvent être passées à la fonction d'exécution.
- **Exécution** : Cette fonction offre des mécanismes permettant de contrôler l'exécution d'un plan d'actions afin de réaliser les changements nécessaires sur le système. En effet, lorsque le gestionnaire autonome génère un plan (par la fonction planification), un ensemble d'actions doivent être entreprises pour modifier l'état d'une ou de plusieurs ressources gérées dont il a le contrôle. La fonction d'exécution de la boucle de contrôle d'un gestionnaire autonome est responsable de la mise en œuvre de la procédure générée par la fonction de planification par l'intermédiaire d'un ensemble d'actions. Ces actions sont exécutées en utilisant le point de contact actionneur de la ressource concernée par les modifications. Le résultat de ces plans d'exécution permet ensuite d'enrichir la connaissance partagée entre les différentes fonctions.

Les données utilisées par les quatre fonctions assurées par le gestionnaire autonome sont stockées et partagées telle qu'illustré dans la figure 2.3 (Connaissance).Pour l'instant, on ne peut pas encore parler de plan de connaissance, car le bloc fonctionnel de la connaissance offre juste une fonctionnalité de stockage de données, les autres fonctions se chargeant de la mettre à jour et de la traiter. Aussi, l'utilisation du mot connaissance peut laisser croire qu'*information*, *donnée* et *connaissance* sont des termes équivalents, ce qui ne nous parait pas juste. Dans le

chapitre 3, consacré au plan de connaissance, nous discuterons ce point en particulier. Toutefois, nous continuerons à utiliser le terme *connaissance*, ici pour respecter la terminologie définie dans l'architecture IBM.

Ces connaissances partagées peuvent porter sur la topologie, l'historique des événements (*logs*), les mesures, les symptômes mais aussi les politiques.

La connaissance utilisée par le gestionnaire autonome peut être obtenue de diverses manières. En effet, les informations qui enrichissent cette connaissance peuvent être passées au gestionnaire par l'intermédiaire de son interface de type actionneur dans le cas d'un gestionnaire autonome de contact(voir section 2.3.3.2), et ce sous forme de règles de politique qui ne sont autre qu'un ensemble de règles de comportement ou des préférences qui influencent les décisions du gestionnaire. Un autre moyen d'enrichir cette base de connaissance est la récupération d'information d'un service externe susceptible de fournir des informations sur le système sous forme de rapports précisant les différents événements qui sont survenus sur les différents composants.

Enfin, le gestionnaire peut créer lui même sa connaissance par le biais des informations récupérées par la fonction monitoring et la procédure de planification qui en a résulté. Grâce aux informations remontées par les capteurs, il pourra récupérer les conséquences des actions planifiées et exécutées.

Cette connaissance peut être de plusieurs natures :

- les informations sur la topologie du réseau. Celles-ci permettent de spécifier les différents composants formant le système autonome. Ainsi, elles contiennent les constructions de ces composants avec leurs capacités, ce qui facilitera la planification.
- les politiques qui sont les règles qui peuvent être consultées pour déterminer si des changements sont nécessaires ou non pour atteindre les objectifs du système autonome. Ces règles doivent être définies de manière standardisée pour qu'elles puissent être partagées par plusieurs gestionnaires autonomes. Ce partage permet au système de fonctionner en se basant sur un ensemble de règles de politique cohérentes. Cependant, dans ce contexte il y a plusieurs niveaux de cohérence qui sont plus ou moins nécessaires suivant les objectifs du système. Une cohérence interne, du voisinage locale et globale.
- La connaissance qui permet la détermination et la résolution de problème

en s'appuyant sur une adaptation de la configuration des ressources gérées de manière automatique et suivant les changements de l'environnement. Ces informations évolutives permettent au système autonome de reconnaitre les symptômes de mauvais fonctionnement du système.

2.3.3.2 Types de gestionnaires autonomes

Bien qu'ils puissent assurer les mêmes fonctions MAPE (Monitoring, Analyzing, Planning, Executing) de la boucle de contrôle, les gestionnaires peuvent être différenciés selon qu'ils contrôlent des ressources gérées ou d'autres gestionnaires autonomes. Ainsi, ceux qui gèrent directement des ressources gérées sont appelés des gestionnaires autonomes de contact alors que ceux qui contrôlent des gestionnaires autonomes sont appelés des gestionnaires autonomes d'orchestration. Les caractéristiques de ces deux types de gestionnaire peuvent être décrites comme suit :

- **Gestionnaire autonome de contact** : Ce type de gestionnaire réalise les tâche de gestion pour des ressources gérées avec lesquelles ils sont directement en contact avec les interfaces standardisées : les *actionneurs* et les *capteurs*.
  Les tâches que peut réaliser ce type des gestionnaire, sont celles des auto-fonctions à savoir : l' auto-configuration (exemple : installation de logiciels), l'auto-restauration (Exemple : correction des entrées incorrectes d'une table de routage), l'auto-optimisation en ajustant, par exemple, l'allocation d'une ressource dans la but de s'adapter à la variation de la charge observée lors de l'utilisation de la ressource, et enfin, l'auto-protection en isolant, par exemple une ressource suite à une détection de tentative d'intrusion. Tout gestionnaire autonome utilise les règles de politique pour réaliser des objectifs et atteindre des buts. Dans le cas d'un gestionnaire autonome de contact, ces règles de politique sont utilisées afin de déterminer les actions qui doivent être appliquées en priorité sur les ressources contrôlées.
  Un gestionnaire autonome de contact peut gérer une ou plusieurs ressources en utilisant un ou plusieurs points de contact. Il en découle différentes combinaisons qui dépendent du type et du nombre de ressources dont un gestionnaire autonome de contact est responsable. Ainsi, nous pouvons distinguer quatre possibilités (figure 2.4 [Kephart & Chess 2003]). Dans le premier cas, le ges-

tionnaire autonome gère, à travers sa boucle de contrôle, une seule ressource. Cette configuration est la plus simple et représente le cas classique de gestion de ressources tel qu'un routeur, un serveur, une application ou encore un ordinateur de bureau. Dans le deuxième cas, le gestionnaire gère un ensemble de ressources, de même type tel qu'une pile de serveurs qu'il essaie d'optimiser dynamiquement afin d'atteindre un certain degré de performance et de disponibilité. Dans une troisième combinaison, le gestionnaire est responsable d'un ensemble de ressources hétérogènes tels que des bases de données et des serveurs Web qu'il doit contrôler pour accomplir un objectif commun. Enfin, un gestionnaire peut contrôler un système responsable de la fourniture d'un service particulier. Il doit alors contrôler les différents composants du système afin de garantir un niveau de performance conforme aux attentes exprimées en termes de politiques pour le service en question. Ces politiques vont être ensuite dérivées afin de gérer individuellement les ressources.

Le gestionnaire, en même temps qu'il utilise des points de contact pour gérer les ressources, en expose aussi pour permettre sa communication avec le gestionnaire d'orchestration.

- **Gestionnaire autonome d'orchestration** : un gestionnaire autonome de contact qui agit d'une manière isolée ne peut assurer un comportement global autonome pour les ressources dont il a la responsabilité. Les fonctions assurées par l'ensemble de ces gestionnaires autonomes de contact doivent être coordonnées afin d'offrir un comportement autonome global pour tout le domaine. Les gestionnaires d'orchestration apportent cette fonction de coordination. Comme les gestionnaires de contact, ils peuvent réaliser plusieurs fonctions de gestion. L'orchestration peut être intra-discipline, dans ce cas l'orchestration se fait entre gestionnaires de contact avec l'objectif de réaliser une des auto-fonctions (auto-configuration, auto-optimisation, ...). L'orchestration peut être aussi inter-disciplines, dans ce cas, le gestionnaire d'orchestration est responsable d'un ensemble de gestionnaires de contact qui réalisent des fonctions de gestion différentes. Le rôle du gestionnaire d'orchestration est d'autant plus important qu'un composant qui semble bien fonctionner d'une façon isolée ne garantit pas forcément un fonctionnement optimal du système dans son intégralité.

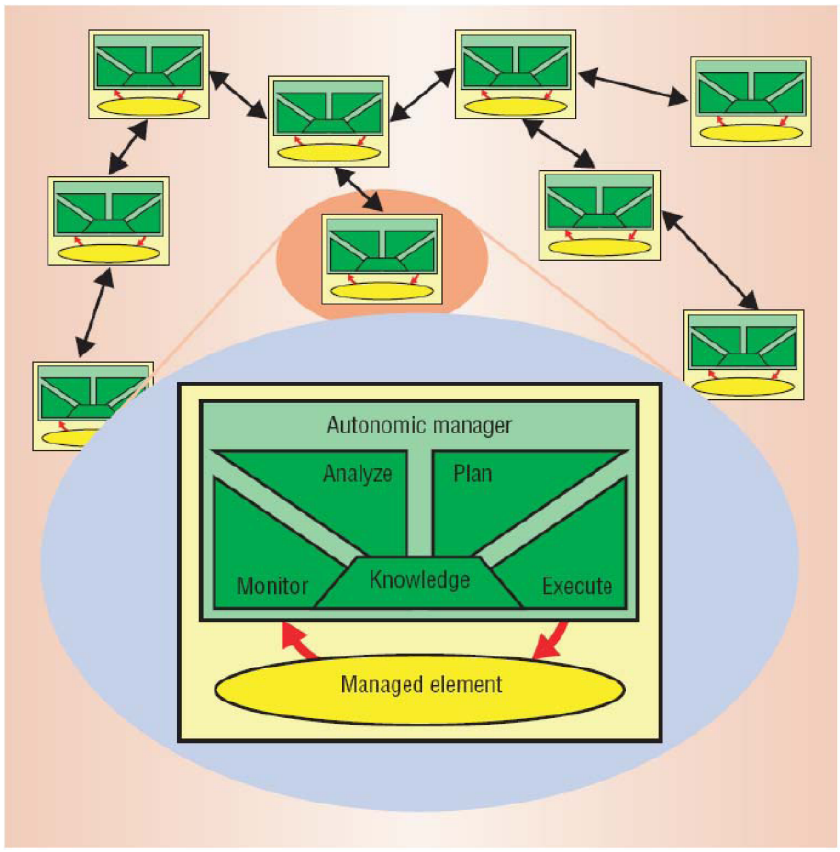

FIGURE 2.4 – Combinaisons possibles d'un gestionnaire autonome de contact

La figure 2.5 [Kephart & Chess 2003, Jacob *et al.* 2002] résume toute ces considérations architecturales.

2.4 Projets de gestion autonome

Plusieurs travaux ont cherché à proposer des systèmes et des architectures qui ont pour objectif de réaliser une ou plusieurs fonctions de gestion autonome avec des mécanismes et concepts offrant des degrés différents d'autonomie.

Au niveau européen, une action de coordination appelé ACCA (Autonomic Communication Coordination Action), qui fait partie des projets ouverts du programme FP6 (Framework Programme 6^{th}) concernant les technologies émergentes (FET : Future and Emergent Network Technologies) dans l'IST (Information So-

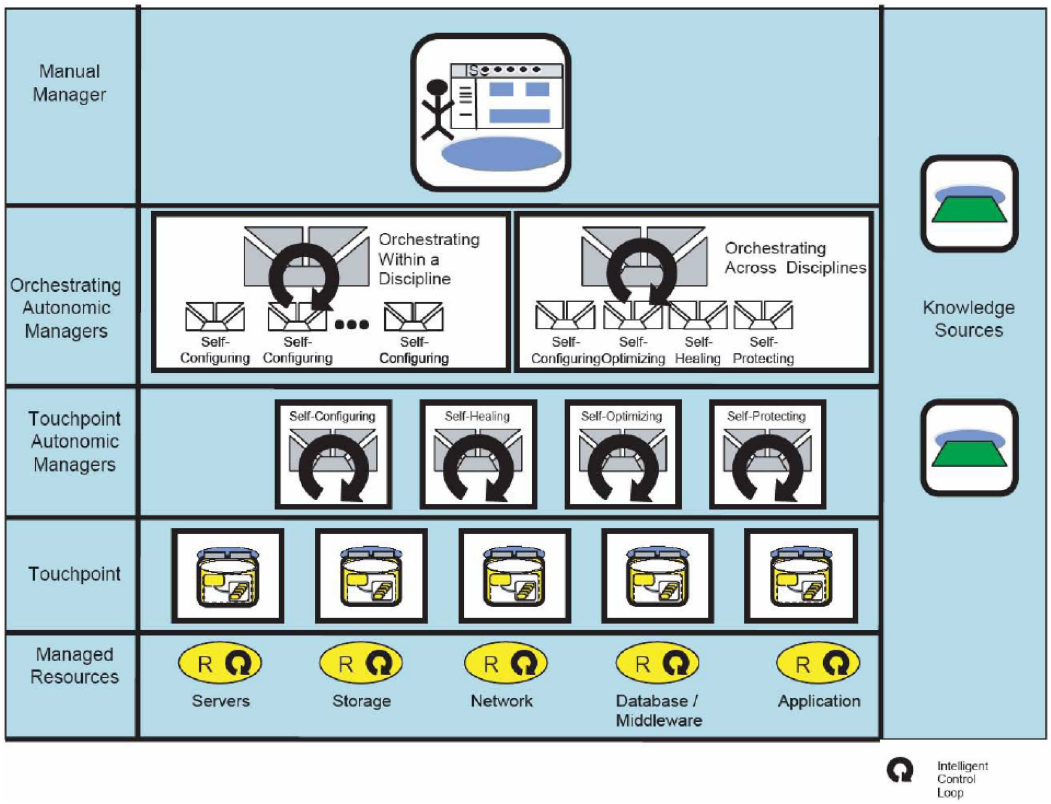

FIGURE 2.5 – Exemple d'architecture de gestion autonome

ciety Technologies), a stimulé de nouvelles initiatives de recherche dans le domaine de la gestion autonome. Ainsi, dans le cadre du SAC (Situated and Autonomic Communication) quatre projets portant sur la gestion autonome ont été proposés : BIONETS (Biologically-Inspired Autonomic Networks), ANA (Autonomic Network Architecture), CASCADAS (Componentware for Autonomic, Situated-aware Communication and Dynamically Adaptable Services) et enfin HAGGLE (A novel Communication Paradigm for Autonomic Opportunistic Communication). Le programme FP7 a propulsé un autre projet AutoI (Autonomic Internet).

Le projet BIONETS (2006-2009) [Pellegrini *et al.* 2006] suppose que les environnements des communications futures seront aussi complexes que les organismes ou encore les écosystèmes biologiques. Ainsi, BIONETS cherche à s'inspirer des systèmes biologiques dans le but de concevoir un environnement qui permet de faciliter l'intégration d'un grand nombre de composants ou encore de ressources hétérogènes, et qui est capable de s'adapter et évoluer de façon autonome. Dans ce cadre, le projet BIONETS vise plutôt la recherche fondamentale en prenant comme modèle les systèmes biologiques.

Le projet ANA (2006-2009) [Tschudin *et al.* 2008] explore de nouvelles méthodes d'organisation et d'utilisation, au delà des technologies traditionnelles de l'Internet. Ces travaux visent la conception et le développement d'une nouvelle architecture réseau qui permet la formation d'entités globales mais aussi de réseaux entiers de façon flexible, dynamique et totalement autonome. L'objectif scientifique de ce projet est l'identification des principes fondamentaux des réseaux autonomes. De plus ce projet a pour objectif de développer une plateforme de démonstration pour tester l'architecture de gestion autonome proposée.

Le projet CASCADAS(2006-2008) [CASCADAS 2006] a comme objectif l'identification, le développement et l'évaluation d'une abstraction (ACE : Autonomic Commmunication Element) d'usage universel pour les services de communication autonomes. Dans ce cadre, ces éléments réalisent d'une façon autonome des fonctions d'auto-organisation et d'auto-adaptation pour la fourniture de services adaptés et situés. Enfin, ce projet essaie d'introduire de nouvelles technologies dans le domaine de l'*autonomic networking* telles que le développement de systèmes auto-adaptatives permettant le déploiement, la composition et l'exécution de services innovants dans un environnement dynamique.

Le projet HAGGLE (2006-2010) [Giordano & Puiatti 2006] propose une nouvelle architecture de gestion autonome des réseaux pour permettre la communication en présence de connectivité intermittente dans les réseaux, et ce en exploitant le concept de communication opportuniste autonome (AOC : Autonomic and Opportunistic Communication), en l'absence d'infrastuctures de communication de bout en bout. Les travaux de recherche réalisés dans ce contexte vont au delà des approches innovantes de Cross Layer en définissant un système qui utilise le *Best Effort* et les informations de contexte (context-aware) pour l'acheminement des messages entre dispositifs mobiles omniprésents, afin de fournir des services quand la connectivité locale est intermittente. Enfin, ce projet présente un environnement ouvert afin de faciliter le développement des applications et des services.

Le projet AutoI (Autonomic Internet) (2008-2009) [AutoI 2008, Abid *et al.* 2008] a pour ambition de proposer une solution *autonomique* qui puisse être déployée à court terme. Il se fixe comme objectif de développer des *overlays* virtuels de ressources qui peuvent couvrir des réseaux hétérogènes, supporter la mobilité, la fiabilité et la QoS. Les services d'adaptation se basent sur des informations et modèles de données ontologiques, ceci dans le but de faciliter le déploiement de nouveaux services et prendre ainsi en compte les NGN (Next Generation Networks). Le but attendu est l'unification des efforts de recherche dans le domaine de l'*autonomic* afin de produire des standards, définir un framework de référence pour l'*autonomic* ainsi garantissant l'interopérabilité et enfin, créer une structure organisationnelle pour maintenir ces deux premiers objectifs.

Le projet 4WARD (2008-2009) [4WARD 2009] fait partie du FP7. L'objectif de ce projet est de développer des solutions réseaux fiables et interopérables offrant un accès direct et ubiquitaire à l'information. Techniquement, il s'attaque au problème de la conception d'architecture intégrés permettant s'affranchir de la méthode *patch-to-patch* d'Internet. Il exploitent aussi la virtualisation des ressources et des réseaux pour permettre une nouvelle abstraction de l'accès au cœur du réseau.

La plupart des grands industriels ont aussi lancés des initiatives dans la foulée d'IBM : par exemple l'initiative "Autonomic System" de FujitsuSiemens Computer [Fujitsu-Siemens 2003], "Dynamic Systems" de Microsoft [Microsoft 07], "Harmonious Computing" de Hitachi [Hitachi 2007] et "N1 Architecture" de Sun MicroSystems [SUNMicrosystems 2007].

Le monde académique s'est également intéressé à ce domaine et plusieurs projets ont ainsi vu le jour : Bio-Networking [BIO-NETWORKING 2007] de l'Université de Californie, AUtonomia [Autonomia 2007] de l'université d'Arizona, *JSPOON* [Konstantinou & Yemini 2003] de l'université de Columbia, et CPN : "*Cognitive Packets Networks*" de l'Imperial College.

D'autres travaux de recherche ont mis en avant certaines fonctions particulières dela gestion autonome. Ainsi, l'architecture OpenWings [Bieber & Carpenter 2003] fait de l'auto-restauration de services suite à des problèmes de réseaux. Dans SELF-CON, [BOUTABA *et al.* 2001] ont préconisés l'auto-configuration de composants grâce aux informations de configuration et de politique associées aux données via un serveur d'annuaire. Dans *Self-Star* [project SELF-STAR 2007], l'auto-optimisation se fait par l'évaluation des performances de systèmes de composants et d'équilibrage de charge alors que les acteurs dans [Menascé *et al.* 2001] proposent une solution d'auto-optimisation par le monitoring dynamique et la re-configuration des paramètres basés sur une métrique de QoS agrégée.

2.5 Enjeux de la gestion autonome

La gestion autonome, pose un très grand nombre de défis [Curran *et al.* 2007, Dobson *et al.* 2006, Greene & Lancaster 2007] dont certains semblent inaccessibles à court terme. Dans cette section, nous tentons d'en énumérer un certain nombre, en particulier ceux qui sont les plus pertinents vis-à-vis de cette thèse.

2.5.1 Gestion distribuée des ressources réseaux

Pendant des années, il a été accepté comme règle que les ressources réseaux pouvaient être gérées de manière centralisée. Cependant, vu l'évolution de la taille et de la complexité des systèmes de communication actuels, cette considération est devenue caduque. L'exemple historique qui illustre bien cette évolution et ce besoin de devenir *autonomic* est l'Internet. En effet, la structure de gestion d'Internet a du passer d'un modèle centralisé à un modèle de gestion et de contrôle totalement distribué pour des raisons typiquement de passage à l'échelle.

Dans un tel contexte, de nouveaux algorithmes *autonomiques* distribués qui

prennent en compte cette complexité tout en restant flexibles doivent être mis en place et analysés. Ceci constitue un des plus grand défis de l'*autonomic*. En effet, ces algorithmes doivent pouvoir garantir un fonctionnement optimal avec un mécanisme de contrôle local du système qui utilise un minimum d'informations de contexte. Parallèlement, il doit réaliser une auto-organisation du système global en optimisant les transactions, en prenant en charge l'auto-correction et en offant une meilleure fiabilité au système, ceci entre des éléments hétérogènes dans un environnement changeant.

Le défi réside dans la contradiction apparente entre ces objectifs. Les systèmes actuels, qui offrent des fonctionnalités intéressantes tels que les systèmes de communication tolérants aux fautes et les primitives pour les communications de groupes, sont des solutions déterministes qui souffrent de leur manque de passage à l'échelle, facteur très important pour les algorithmes des environnements de communication. D'un autre coté, les approches probabilistes offrent le passage à l'échelle mais permettent moins de contrôle sur le système.

La décentralisation des tâches de gestion sur l'ensemble des éléments d'un système de communication pose, également, un problème de gestion de cohérence au niveau des transactions entre les éléments du système. Le domaine des bases de données distribuées a certainement trouvé des solutions à ce problème avec les mécanismes de requêtes transactionnelles. Cependant, cette expérience des bases de données montre que pour avoir une garantie forte de consistance, il faut une barrière de synchronisation, ce qui est difficilement réalisable (voir impossible) dans un contexte distribué et peu fiable tout en permettant le passage à l'échelle. Pour une cohérence dans le fonctionnement du système face aux objectifs fixés par l'administration, au moins une collaboration/coopération distribuée des éléments autonomes est nécessaire. La question est maintenant de savoir comment maintenir le système stable avec les communications supplémentaires engendrées par cette collaboration.

2.5.2 Boucles adaptatives (auto-fonctions)

Dans l'objectif d'atteindre l'autonomie, la boucle de contrôle est certainement l'élément de base le plus important. Il réalise, en effet, les auto-fonctions de base par l'acquisition et le traitement des informations sur les ressources pour prendre

les décisions adéquates en conformité avec les objectifs de haut niveau. Cependant, cet objectif se heurte à un certain nombre de défis avant sa réalisation.

Le premier défi est la conception d'un langage standard d'expression des politiques de haut niveau dans un langage de spécification des objectifs de haut niveau. En effet, il faudrait qu'un tel langage soit suffisamment expressif pour donner des politiques claires, suffisamment riches pour exprimer des objectifs complexes, suffisamment génériques et extensibles pour prendre en compte toutes les technologies et faciliter le raffinement des politiques.

Le second défi est l'optimisation d'un ensemble de paramètres ou d'objectifs qui sont contradictoires. En effet, il peut y avoir un besoin d'optimiser une ressource qui comporte plusieurs paramètres interdépendants les uns avec les autres comme cela arrive souvent. Dans un tel cas, essayer de maximiser l'utilisation de la ressource tout en maintenant les performances des services (c'est-à-dire réaliser un bon compromis en quelque sorte) est un vrai défi, l'optimisation de certains paramètres pouvant être parfois contradictoire ou liée. D'un autre coté, avoir un fonctionnement global optimal de manière automatique en prenant en compte la possibilité qu'il y ait des objectifs business contradictoires entre deux domaines est aussi un défi. Une avancée majeure serait déjà d'avoir des processus automatiques qui découvrent le degré de dépendance de ces paramètres pour construire une ontologie locale à chaque équipement et prendre ainsi des décisions en fonction de cette structure. Aussi, il existe un besoin de standardiser l'ensemble des meta-ontologies (comme ce fut le cas avec la base d'information CIM) sur lesquels les concepteurs pourraient se baser pour aider à concevoir de tels systèmes.

Face à ces défis, deux approches plus ou moins complémentaires sont souvent mises en avant :

- L'approche ingénierie qui se base, principalement, sur les techniques de génie logiciel (génération automatique de code, déploiement automatique) et qui semble atteindre ses limites face aux exigences de passage à l'échelle.
- L'approche qui émerge le plus auprès du monde de la recherche est celle qui se base sur l'apprentissage, et plus généralement, sur les systèmes cognitifs. Cette approche porte le nom de plan de connaissance.

Dans cette thèse, c'est à cette approche que nous allons nous intéresser pour montrer l'intérêt et les limites qui sont derrière cette approche. Le chapitre 3 est

entièrement dédié à ce concept.

2.6 Conclusion

Dans ce premier chapitre, nous avons présenté la gestion autonome comme un paradigme qui vise à rendre les systèmes capables d'effectuer l'ensemble des tâches que l'être humain faisait pour la gestion de l'infrastructure. Le rôle des administrateurs se limitent ainsi à fournir les objectifs de haut niveau business, réduisant ainsi les coûts de possession dus aux interventions d'experts.

La vision de la gestion autonome s'inspire en grande partie du fonctionnement de la nature (le système nerveux des mammifères), et est née du besoin d'une évolution des systèmes vers une autonomie qui garantit un certain nombre de fonctions. Ces fonctions sont : l'auto-configuration, l'auto-réparation, l'auto-optimisation, l'auto-protection.

La première architecture de gestion autonome, proposée par IBM, se base sur la notion de boucle de contrôle fermée avec un gestionnaire autonome qui récupère un ensemble d'informations sur une ou plusieurs ressources à gérer, et prend des décisions de configuration de manière automatique pour atteindre l'objectif des concepteurs de manière optimale. Cette boucle de contrôle fermée qui adapte la configuration des ressources en fonction des changements de l'environnement porte le nom de l'auto-adaptation. Cette fonction est au cœur de la gestion autonome.

Cette notion de gestion autonome, ayant été déclinée au domaine de la gestion des réseaux, a donné naissance au thème des réseaux autonomes. Ce nouveau thème, a fait apparaitre de nombreuses dénominations (communications autonomes, ...) et a permis la proposition de plusieurs architectures autour du concept, ceci dans le but de fermer la boucle de contrôle afin de réaliser la fonction d'auto-adaptation.

Après une étude des ces différentes architectures, il se dégagent deux domaines qui sont utilisés pour réaliser l'auto-adaptation dans les réseaux : l'ingénierie et l'utilisation des systèmes cognitifs. Le domaine de l'ingénierie se base fondamentalement sur les techniques du monde du génie logiciel : génération automatique de codes et de modèles. Cette approche est mise en avant par IBM. L'utilisation de systèmes cognitifs quant à elle a reçu l'attention du monde de la recherche du fait de son caractère innovant, en utilisant les outils de l'intelligence artificielle pour

automatiser les tâches réseaux. Cette dernière approche fait référence à la notion de *plan de connaissance* qui est l'objet du chapitre 3. Comme nous l'avons déjà dit, cette thèse s'est intéressée au problème de savoir comment les outils cognitifs pouvaient aider à réaliser l'auto-adaptation.

CHAPITRE 3

Plan de connaissance

Sommaire

3.1 Introduction

L'idée du *plan de connaissance* est née de la communauté Internet sous la direction du DARPA [Clark *et al.* 2003a], après une constatation des limites de conception de la technologie IP. La réflexion a commencé avec le projet *NewArch* [Clark *et al.* 2003a] dirigé par DARPA en 2003 avec au centre, la question suivante : "*que changerions-nous dans la conception d'Internet si nous devions le reconcevoir à partir de zéro ?* ". En effet, les grands avantages d'Internet que sont sa transparence et son mode d'évolution (extensibilité) qui ont fait son succès sont vite apparus comme des limites après son déploiement à grande échelle. Une bonne compréhension de ces limites permet de préciser l'origine et la motivation liées au concept de *plan de connaissance.*

Le principe de base des communications sur Internet est le bout en bout, c'est-à-dire que les nœuds qui sont dans le cœur du réseau ne font aucune transformation, observation ou filtrage sur les données transportées (les charges utiles). Dans ce type de communication, le rôle des nœuds au cœur du réseau se limite à la retransmission des paquets. Ils ne se soucient ni du contenu des paquets, ni du comportement des applications attendues lors des communications. Les nœuds aux extrémités de la communication concentrent toute l'intelligence du réseau. Ce principe a deux avantages qui ont, en partie, fait le succès d'Internet que sont :

- Le déploiement de nouvelles applications peut se faire sans modifier lourdement les nœuds dans le cœur du réseau. Ainsi, beaucoup d'applications en phase de tests peuvent être déployées sans perturber le fonctionnement normal du cœur du réseau.
- Les nœuds au centre du réseau sont transparents aux nœuds qui communiquent. Les extrémités communiquant n'ont pas à se soucier, par conséquent, du fonctionnement du cœur du réseau ;

Malheureusement, ces avantages, jadis adaptés lorsqu'Internet était une petite structure, se sont révélés être des limites lorsqu'il s'agit de la sécurité après le déploiement à grande échelle. En effet, historiquement ces deux avantages, qui favorisent la propagation d'attaques, de virus et d'autres applications malveillantes, ont été en partie à la source de l'apparition des Firewalls [Clark *et al.* 2003a]. Les Firewalls ont pour objectif clair de casser la transparence en bloquant les applica-

tions inconnues. Ainsi, la perte progressive de cette transparence va rendre difficile le déploiement de nouvelles applications et bloquer l'évolution d'Internet. Dans ces conditions, comment garder les fonctionnalités de transparence tout en résolvant les limites de sécurité ? [Clark *et al.* 2003a] propose une adaptation en fonction du niveau de confiance en intra-domaine. Il s'agit, aussi, de rendre la sécurité transparente pour les utilisateurs dans un réseau considéré comme fiable (intra-domaine), et une sécurité sous contraintes dans un environnement non fiable.

Un autre inconvénient de la communication de bout en bout est le manque d'explications à l'intérieur du réseau. Ainsi, si un problème survient lors d'une communication à l'intérieur du réseau, seuls les nœuds aux extrémités peuvent le signaler à leur vis-à-vis. Cette notification se fait sans forcément une explication sur l'origine du problème ni une suggestion sur la manière de le régler si c'est possible. Un exemple de situation où une explication serait utile est lorsqu'une communication échoue à cause de restrictions du FAI (Fournisseur d'Accès Internet). Si l'utilisateur a seulement comme information la notification de l'échec de la communication, cela ne lui servira pas à grand chose.

Pour aller au delà de toutes ces limites, il semble intéressant de disposer d'une structure transverse (distribuée sur tous les nœuds participant à la communication ou pas) qui se chargerait de collecter toutes les informations sur la nature des données transmises, les comportements attendus des applications ainsi que les données d'authentification. Cette collecte permettrait ainsi d'avoir les données/informations et connaissances là où elles sont nécessaires pour une meilleure gestion des communications. Les nœuds intermédiaires gardent ainsi leurs fonctionnalités de base tout en gagnant plus d'*intelligence* pour les situations éventuelles non prévues au moment de la conception.

Un autre aspect intéressant de la conception d'Internet est son mode d'évolution. En effet, depuis sa création, l'infrastructure Internet a été enrichie de nombreuses fonctionnalités et protocoles qui répondaient à des besoins plus ou moins urgents. Ces évolutions sont basées, le plus souvent, sur des réflexions et des améliorations de proche en proche et des solutions valables sur le moyen ou le court terme. Ce mode d'évolution a résisté à toutes les épreuves jusqu'ici grâce à la conception flexible d'Internet. Cependant au lieu de constituer une limite en soit, cette manière d'évoluer pose un certain nombre de questions. Il est difficile de savoir jusqu'à

quand ce mode d'amélioration d'Internet va durer et encore moins si au final, il ne va pas mettre Internet dans un état instable (à partir d'un point critique de superpositions de fonctionnalités/protocoles). Le comportement d'Internet sera-t-il alors prédictible ? Quel sera l'impact de ces ajouts sur la manière de gérer les réseaux dans le futur ?

Ces questions font apparaître un besoin pour une structure réseau qui permette de gérer l'évolution tout en offrant une sorte d'abstraction vis-à-vis de la conception sous-jacente. Cette structure devrait avoir une certaine faculté à reconnaître les technologies obsolètes, soit en apprenant, soit en récupérant la connaissance où elle se trouve. Une fois cette connaissance "*acquise*", elle doit être capable de faire du raisonnement sur sa connaissance pour décider de lancer une mise à niveau ou une mise à jour ou même parfois rester dans son état initial. Dans un cadre idéal, cette structure aurait des facultés cognitives qui la rendrait capable d'acquérir et de gérer toute la connaissance du réseau.

Cette structure fonctionnelle de la gestion de réseau apporterait une nouvelle manière de voir la gestion des réseaux en rendant les réseaux moins dépendants des êtres humains car les tâches de gestion deviennent de plus en plus complexes. En d'autres termes, elle permettrait de gérer la connaissance du réseau comme un ensemble d'objectifs et elle se chargerait d'acquérir l'expérience pour les atteindre. [Clark *et al.* 2003b] a été le premier à proposer, explicitement, de mettre toutes ces fonctionnalités dans une même structure appelée : plan de connaissance (ou *Knowledge Plane* en anglais).

Les fonctions de gestion de réseaux sont souvent regroupées dans des plans : le plan de données (chargé de la retransmission), le plan de contrôle (établissement et gestion des communications) et le plan de gestion (sécant à ces deux plans). [Clark *et al.* 2003b] propose de créer un nouveau plan pour la gestion de la connaissance du réseau afin de ne pas rendre les plans existants trop complexes.

Le plan de connaissance qui était né dans l'esprit de régler des limites spécifiques à Internet va être adopté (ou intégré), par la suite, dans les architectures de gestion autonomes pour, finalement, y prendre une place assez importante. En parallèle, le concept de connaissance a fait naître de nouveaux concepts telles que la gestion orientées connaissances qui est une gestion de réseaux basée sur les connaissances acquises par le plan de connaissance.

Dans cette partie, nous allons détailler les notions de *plan de connaissance* et de gestion orientée connaissance et présenter les considérations architecturales et les concepts ayant trait au concept de plan de connaissances. Nous finirons la section en citant quelques défis majeurs qui sont à relever par le monde de la recherche pour permettre la réalisation complète du plan de connaissance, et nous discutons de la place du plan de connaissance et dans le cadre des réseaux autonomes.

3.2 Plan de connaissance (*Knowledge Plane*)

3.2.1 Cadres et évolution du concept

La vision de base du plan de connaissance est celle de la conception de nouveaux systèmes réseaux avec de nouvelles techniques avancées permettant d'avoir plus d'intelligence distribuée au niveau des équipements. Cependant, la contrainte qui est posée est celle de garder les avantages des systèmes classiques (extensibilité, facilités de déploiement de nouvelles applications). Les équipements au cœur du réseau auront ainsi une "*connaissance*" du comportement des applications aux extrémités du réseau et s'adapteront en fonction de ces connaissances mais aussi des objectifs de haut niveau qui leur sont fixés par les administrateurs. Cette vision permet d'avoir une adaptation automatique de la configuration des éléments réseaux sous la contrainte des configurations "acceptables" fournies par les administrateurs. Cette adaptation est basée sur une vue globale de la connaissance du réseau qui permet aux équipements de prendre des décisions de bas niveau sans violer les objectifs de haut niveau qu'ils doivent atteindre.

Un tel système est très ambitieux sous plusieurs aspects :

- La définition de la notion de connaissance reste très générale et n'est pas facile à appréhender dans les environnements réseaux ;
- Les techniques d'adaptations automatiques classiques ne pourront suffire pour prendre en compte toutes les situations possibles dans le réseau ;
- La gestion de ces mécanismes dans des domaines d'administration différents n'est pas toujours simple car les limitations (par un firewall ou des politiques d'un opérateur) dans un domaine peuvent annuler des optimisations effectuées en amont ou en aval ;

– Le découpage des activités des éléments du plan de connaissance est un autre défi à relever

A ces questions, la proposition initiale du plan de connaissance donne quelques pistes de réflexions que nous allons à présent détailler.

3.2.2 Les attributs du plan de connaissance

Pour réaliser les objectifs du plan de connaissance sans détruire totalement les architectures existantes, il faudrait une définition complète de l'architecture du plan de connaissance. A défaut, un certain nombre d'attributs ont été proposés comme un ensemble de considérations minimales à prendre en compte dans la spécification des architectures de plans de connaissances. Ces attributs, au nombre de cinq, sont représentés sur la figure 3.1 [Clark *et al.* 2003b]. En analysant ces attributs, on peut se rendre compte qu'ils tentent de répondre uniquement aux deux questions suivantes :

– Quels sont les outils à disposition pour la réalisation du plan de connaissance (framework cognitif) ?
– Quels sont les considérations organisationnelles à prendre en compte pour conserver les avantages du modèle de bout en bout tout en passant à un modèle avec participation de tous les éléments réseau dans le plan de connaissance ?

3.2.2.1 Participation des nœuds situés aux extrémités

Le concept de communication de bout en bout qui domine dans l'environnement de modélisation des réseaux fait que l'essentiel de l'intelligence est refoulée aux extrémités du réseau. Cet état de choses a comme conséquence de fait, que la connaissance du réseau n'est produite et consommée qu'au niveau des nœuds d'extrémités. Lorsqu'une connexion TCP, par exemple, échoue le cœur du réseau ne pourra pas aider à identifier l'origine de l'échec encore moins à résoudre le problème de manière transparente pour les nœuds d'extrémité de la communication.

Vu sous cet angle, la communication de bout en bout constitue un handicap pour un plan de connaissance car, le cœur du réseau serait complètement *aveugle* sur la nature et le comportement attendu des communications qui le traversent.

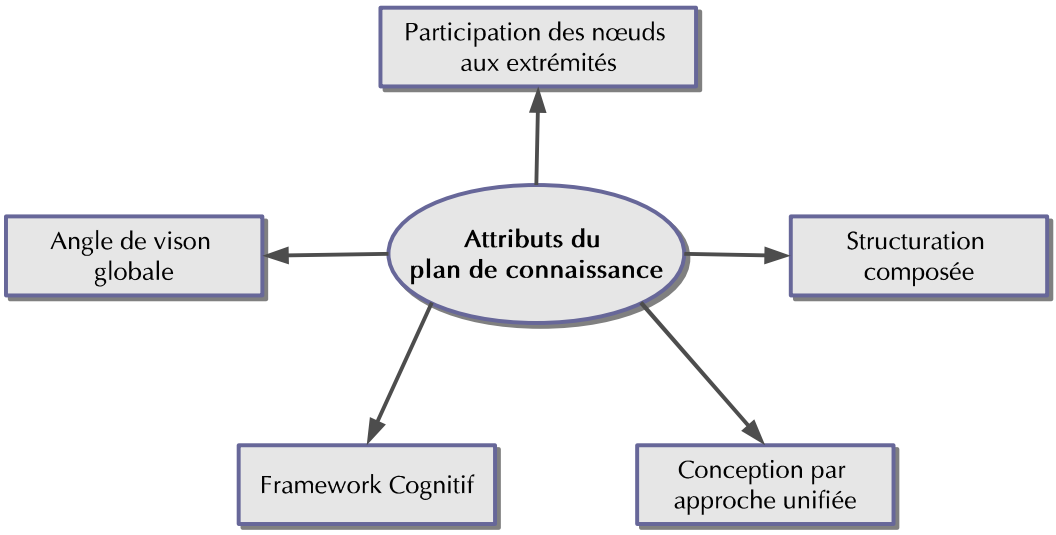

FIGURE 3.1 – Attributs du plan de connaissance

Cependant, le paradigme de communication de bout en bout a été à la base de la réussite des réseaux actuels comme Internet. En effet, il permet le déploiement facile de nouvelles applications sans avoir à modifier le cœur du réseau et est donc une fonctionnalité intéressante à conserver. Dans la conception du plan de connaissance, on ne peut se contenter d'avoir de l'intelligence uniquement aux extrémités du réseau, ni se priver de la facilité de déploiement de nouvelles applications qu'offre le bout en bout.

Lorsque surviennent des événements dans le réseau, les connaissances qui permettent de les expliquer et de s'adapter, peuvent se trouver aussi bien dans le cœur du réseau qu'au niveau des extrémités communicantes. Un des attributs du plan de connaissance est de permettre la consommation d'une partie de la connaissance produite aux extrémités du réseau.

3.2.2.2 Approche globale

Les systèmes de gestion sont en général parcellaires, c'est-à-dire qu'ils ne peuvent gérer que des équipements qui sont dans un domaine particulier du réseau. Cependant, lorsqu'un événement survient à l'intérieur du réseau, son identification et sa localisation peut nécessiter la combinaison/corrélation de données (événements et observations) situées dans différentes parties du cœur du réseau.

Les données provenant de l'extrémité et de l'intérieur du réseau d'un opérateur peuvent ne pas suffire à expliquer une séquence d'événements qui se produit. Par exemple nous pouvons citer le cas d'une erreur qui se produit à cause d'une mauvaise configuration d'une partie du cœur du réseau. Sans possibilité de pouvoir disposer d'informations sur la partie concernée du réseau, il est très difficile (voir impossible) de localiser l'erreur.

Le plan de connaissance doit, donc, avoir la possibilité d'étendre la portée (l'angle) de sa vision en passant d'une vue parcellaire à une vue globale des connaissances du réseau. Cette vue globale permettra de mettre plus facilement en relation des données provenant de différentes parties du réseau afin d'aboutir à un résultat précis et concluant. Entre autre, il doit être capable, dans un cadre idéal, d'adapter la portée des données à prendre en compte en fonction de l'événement ou les données à traiter. Cependant, se pose le problème de la sécurité pour déterminer quel élément a le droit d'accéder à quelles connaissances dans le réseau.

3.2.2.3 Structure composée

La structure du plan de connaissance doit lui permettre de composer à la volée les différents éléments le constituant ainsi que de définir la portée des connaissances du réseau à prendre en compte. Cette composition se fait pour constituer une vue globale mais aussi pour supporter une structure de la connaissance granulaire permettant la gestion de la sécurité de la connaissance.

En effet si deux réseaux, qui n'étaient pas connectés, venaient à l'être, leurs plans de connaissance devraient pouvoir corréler leur connaissances si nécessaire pour obtenir une vue globale tout en gardant un minimum de connaissance privée à chaque réseau. Cette collaboration a comme corolaire la nécessité pour le plan de connaissance de pouvoir fonctionner en présence de données incomplètes (à causes des données privées d'un autre domaine et donc inaccessibles) ou même contradictoires (du fait de contextes différents). En effet, les opérateurs du réseau ne se font pas forcément confiance pour mettre toutes leurs données à disposition les uns et des autres.

Ils peuvent également avoir des politiques d'exploitation différentes dans leur réseaux respectifs, ce qui peut impliquer des politiques contradictoires, comme indi-

qué plus haut. Ces contradictions peuvent aussi être causées par une hétérogénéité des équipements qui produisent des connaissances différentes au cours du temps. Il ne semble donc pas réaliste de faire une supposition sur l'homogénéité des connaissances dans le réseau.

3.2.2.4 Utilisation des systèmes cognitifs

L'une des deux principales nouveautés (avec la séparation du plan de connaissance des autres plans) est l'utilisation des systèmes cognitifs. La motivation derrière l'utilisation de ces outils est de répondre à un certain nombre de situations complexes qui doivent être prises en compte par le plan de connaissance :

- Il doit pouvoir fonctionner en présence de données incomplètes, inconsistantes et potentiellement fausses ou malicieuses ;
- Il doit pouvoir fonctionner de manière appropriée en présence de configurations de haut niveau des opérateurs en conflit ou inconsistantes ;
- Il doit être assez général pour pouvoir prendre en compte l'arrivée de nouveaux éléments dans le réseau dont les technologies n'étaient pas prises en compte lors de sa conception. Il doit, aussi, gérer la complexité d'un environnement hautement dynamique en prenant en compte les changements immédiats et les évolutions à long terme.

Le plan de connaissance doit, donc, prendre des décisions en présence d'informations partielles ou contradictoires et reconnaître et arbitrer des conflits au niveau des politiques et objectifs de haut niveau. Il doit, aussi, répondre aux attaques et problèmes réseaux dans des délais meilleurs qu'un être humain ne pourrait le faire. Il effectue, également, des optimisations dans un environnement avec des métriques à plusieurs dimensions qui impliqueraient des solutions analytiques qui sont trop complexes pour être abordées par l'homme. Il doit permettre, aussi, d'automatiser les fonctions qui doivent aujourd'hui être effectuées par des experts réseaux et des techniciens hautement qualifiés. Les systèmes cognitifs peuvent, ainsi, servir de base au plan de connaissance pour réaliser toutes ces exigences.

Les systèmes cognitifs se fondent sur la notion de *cognition* [Anderson 2003], [Christensen & Nagel 2006] qui est largement explorée par le domaine des sciences cognitives [Stillings 1987], [Garfield & Weisler 1987]. Les sciences cognitives sont

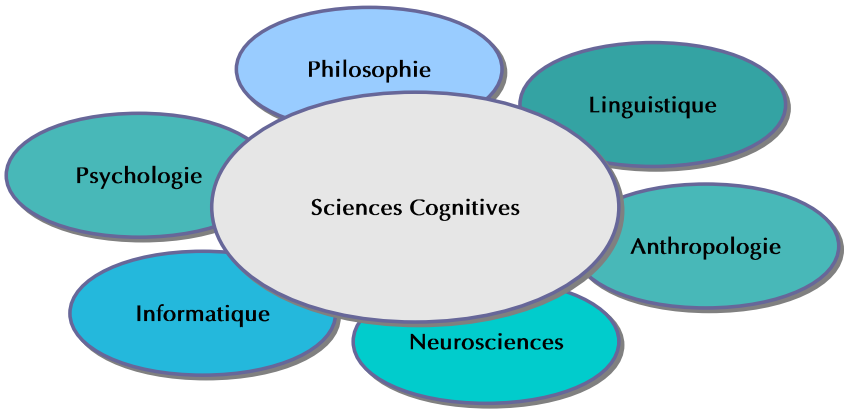

FIGURE 3.2 – Disciplines scientifiques constituant les sciences cognitives

l'intersection de plusieurs disciplines scientifiques telles que l'informatique, la psychologie, la philosophie, la linguistique, l'anthropologie, les neurosciences (voir figure 3.2).

La *cognition* implique la capacité de comprendre comment les choses "pourraient" potentiellement se passer, maintenant mais aussi dans un temps futur, et de prendre cela en considération pour déterminer la meilleure manière d'agir. Dans ce processus, se rappeler de ce qui s'est passé dans le passé permet d'anticiper les événements futures d'où l'importance de la notion de mémoire. Elle permet d'utiliser le passé pour prédire le futur et, donc, assimiler ce qui se passe dans la réalité présente pour s'adapter et améliorer la capacité d'anticipation du système [Vernon *et al.* 2007]. Tout le processus est basé sur une supposition simple qu'il y a une relation continue entre les événements du passé et ceux du futur dans une fenêtre de temps bien définie.

Le système cognitif permet donc d'exhiber le comportement réel du système à travers des perceptions, des actions, des délibérations, des communications, tout ceci en fonctionnant de manière efficace pour des cas non explicitement pris en compte lors de sa conception. La définition des capacités d'un système cognitif n'est pas tout à fait tranchée, toutes étant plus ambitieuses les unes que les autres. [Brachman 2002] dit que c'est un système qui, *en plus de pouvoir raisonner et apprendre à partir de l'expérience pour améliorer ses performances au cours du temps, peut expliquer et justifier les actions qu'il entreprend.* [Hollnagel & Woods 1999]

ajoute qu'un système cognitif devrait pouvoir *voir un problème suivant plusieurs aspects et utiliser sa connaissance de son état interne et celui de son environnement pour planifier et modifier ses actions en conséquence.*

Les fonctions majeures des systèmes cognitifs sont : la perception, l'action, l'anticipation, l'adaptation et la motivation [Vernon *et al.* 2007]. Pour réaliser ces fonctions, l'apprentissage et le raisonnement sous contrôle de la motivation (qui est l'objectif de conception du système) sont nécessaires.

3.2.2.5 Conception par approche unifiée

L'approche de conception descendante qui consiste à découper un processus complexe en sous-processus moins complexes (à implémenter), a permis dans le monde l'informatique de résoudre de nombreux problèmes. Il est, donc, naturel de penser à ce type d'approche pour la conception du plan de connaissance. Pour montrer l'intérêt ou pas d'une approche *top-down*, nous allons présenter des exemples.

Considérons le cas d'un utilisateur qui tente d'installer une nouvelle application réseau et qui se rend compte que l'opération échoue. Une raison pourrait être que son fournisseur d'accès a bloqué la classe de trafic correspondant à ce type d'application. Dans ce cas, pour pouvoir donner une explication claire à l'utilisateur sur les raisons de son échec, le plan de connaissance devrait pouvoir accéder aux contraintes de configuration du fournisseur d'accès Internet afin de déterminer la règle bloquante et informer l'utilisateur de son implication. Cet exemple sous-entend que les informations de configuration et les observations décrivant le comportement d'échec devraient être visibles dans le même Framework, celui du plan de connaissance. Cette relation, implicitement nécessaire dans ce cas, diminue la possibilité de l'approche ascendante de produire des résultats efficaces.

Le plan de connaissance peut, dans certains cas, résoudre des problèmes dans le réseau sans avoir besoin d'accéder à d'autres informations. Le cas le plus évident est celui où le plan de connaissance découvre que le problème peut être résolu par une décision de bas niveau qui n'affecte pas les objectifs de haut niveau programmés par les opérateurs. Par contre, pour décider de la pertinence d'un changement de configuration, le plan de connaissance doit accéder au processus de raisonnement qui soutend la politique de configuration. Cela suppose que les connaissances de

planning doivent être mises dans le même contexte que les connaissances de réparation des problèmes. Ce deuxième cas montre également que la relation entre les différents aspects du plan de connaissance n'est pas figée mais comprend, aussi, des connaissances liées aux politiques globales de configuration.

Enfin, lorsque qu'un équipement récupère une observation de bas niveau qui peut être causée par une anomalie, il ne "sait" pas la pertinence réelle de la connaissance. L'observation en question peut déclencher une opération de réparation, de reconfiguration, une notification de l'opérateur réseau, une alerte de sécurité ou une autre tâche de traitement. Par conséquent, une observation sur la connaissance du réseau ne peut pas être considérée comme concernant exclusivement une classe de tâches. Le découpage par la méthode descendante s'en trouve encore moins efficace dans ce cas précis.

Les trois situations que nous avons présentées montrent les limites que pourraient avoir la méthode descendante par rapport au mode de fonctionnement du plan de connaissance. En effet, elle n'est pas adaptée à l'objectif du plan de connaissance qui est d'avoir une vue globale de la connaissance et des processus du réseau. Cette vue globale permet de prendre en compte le fait que les tâches et connaissances du réseau ne sont pas partitionnées de manière clairement exclusives mais sont au contraire interdépendantes, inter-influentes les unes sur les autres suivant un graphe de dépendance assez complexe.

Au contraire, l'approche unifiée devrait permettre de prendre en compte tout ces aspects dès la conception du plan de connaissance. Une solution basée sur la méthode descendante est plus simple à mettre en œuvre mais une conception intégrée avec une approche unifiée semble correspondre mieux à la philosophie du plan de connaissance.

3.2.3 Aspects fonctionnels et architecture

3.2.3.1 Introduction d'un nouveau plan

Un des aspects les plus importants du plan de connaissance est sa séparation des autres plan existants : le plan de données et le plan de contrôle. Les deux premiers plans sont une subdivision bien connue dans le monde des architectures réseaux. Le plan de données a en charge [Clark *et al.* 2003b] la transmission des données jusqu'à

leurs destinataires alors que le plan de contrôle a pour rôle de mettre en place et de maintenir les communications. Le plan de gestion quant à lui permet de superviser, contrôler, mesurer les performances du plan de données. Le plan de gestion est, aussi, utilisé pour avoir une vue globale de tous les aspects du fonctionnement du réseau afin de prendre les bonnes décisions (figure 3.3).

Contrairement au plan de données, le plan de connaissance ne s'occupe pas de la transmission effective des données mais juste du contrôle des mécanismes qui en sont chargés afin d'avoir un fonctionnement optimal. Le plan de connaissance se différencie du plan de contrôle et de gestion car il n'a pas forcément comme objectif de voir sur toutes les couches du plan de données pour avoir une vue unifiée. Au contraire, c'est une structure parcellaire avec des liens avec les différents éléments réseaux qui peuvent agréger leurs données selon l'objectif de la connaissance à produire. Le plan de connaissance offre une approche qui passe plus facilement pour gérer le plan de contrôle qui, lui même, gère le plan de données.

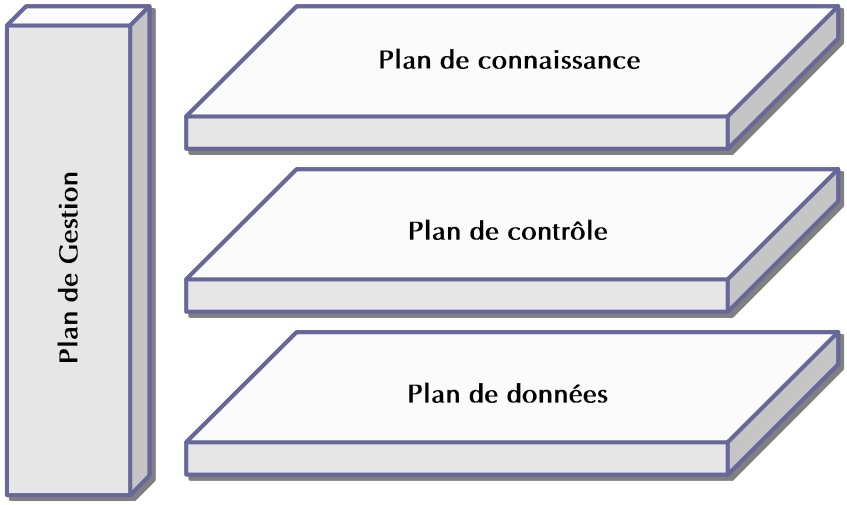

FIGURE 3.3 – Exemple de modèle d'architecture des réseaux autonomes

3.2.3.2 Utilisation des systèmes cognitifs

Un autre aspect important du plan de connaissance est l'utilisation des systèmes cognitifs. Le plan de connaissance a, entre autre, parmi ses contraintes de fonctionner :

- correctement, en présence de données incomplètes, incompatibles ou éventuellement malveillantes ;
- correctement en présence d'objectifs incompatibles entre plusieurs opérateurs réseaux ;
- de manière évolutive et extensible afin de supporter durablement l'avancée des technologies et l'ajout de nouveaux types de matériels

Pour atteindre ces objectifs, les solutions analytiques déterministes risquent de montrer assez vite leurs limites du fait de la difficulté, de cerner le comportement général des systèmes distribués que constituent les applications réseaux. Le plan de connaissance pour répondre à cette problématique utilise des systèmes cognitifs (voir section 3.2.2.4) pour réaliser des systèmes auto-adaptatifs et évolutifs. Ces supports cognitifs sont alimentés en données grâce à une boucle de contrôle (figure 3.4 proposée par les auteurs de [Clark *et al.* 2003b]) qui leur permet d'agir sur le réseau et de récupérer des informations.

3.2.3.3 Aspects structurels et architecturaux

Le plan de connaissance, de part sa définition, est un système distribué dans le réseau afin d'avoir une vue plus ou moins fine de chaque partie du réseau. L'aspect structurel et compositionnel du plan de connaissance fait qu'il a un certain nombre de caractéristiques qui lui sont propres. Il est :

- **Distribué** : les fonctionnalités du plan de connaissance sont logiquement et physiquement décentralisées dans l'ensemble du réseau ;
- **Hiérarchique** : Les grands réseaux composés de sous-réseaux de taille variable sont interconnectés à travers différents domaines. La résolution d'un problème ou la réalisation d'une des fonctions du plan de connaissance peut nécessiter que des parties de sous-plans de connaissance se regroupent pour former un ensemble plus grand et plus complexe. Inversement, le plan de connaissance d'un grand réseau peut se décomposer en sous plans de connaissance pour avoir une vue plus précise d'une connaissance ou bien pour compartimenter le réseau en zones de politiques de sécurité différentes, par exemple. Cette hiérarchie dynamique est nécessaire au niveau du plan de connaissance.
- *Constraint Driven* : L'évolution du plan de connaissance, grâce à ses supports

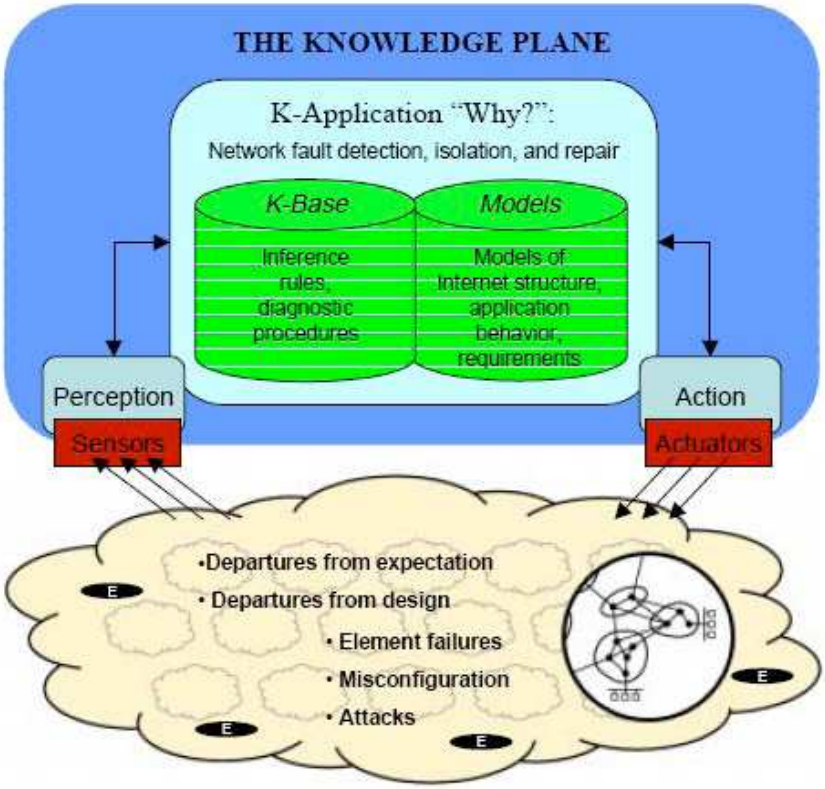

FIGURE 3.4 – Boucle d'adaptation du plan de connaissance

cognitifs, doit être réalisée en respectant les objectifs des concepteurs du réseau. Autrement dit, le système pourra adopter de nouveaux comportements tant que ceux-ci ne sont pas contraires aux objectifs fixés (les contraintes) par ses concepteurs.

3.2.4 Les étapes du plan de connaissance

Comme dans tout système doté d'un mécanisme d'apprentissage, le plan de connaissance évolue en passant par plusieurs étapes avant d'être dans un état complètement fonctionnel. La première étape consiste à découvrir l'environnement. Cette étape n'est pas forcément réalisée par le plan de connaissance mais peut être fournie comme une connaissance à *priori*. Cet étape permet une initialisation du système. La seconde étape est la mise en route des processus d'apprentissage et de raisonnement qui correspond à la phase d'adaptation et de construction de la

connaissance. Le troisième est dernière étape commence lorsque le système a assez d'expérience pour agir et s'améliorer de manière *autonome*.

La réalisation de ces trois étapes nécessaires pour la création, la validation et l'évolution de la connaissance [Clark *et al.* 2003b] est représentée par trois cycles (figure 3.5) : un cycle de reconnaissance-explication, un cycle de reconnaissance-explication-suggestion, et finalement un cycle de reconnaissance-action.

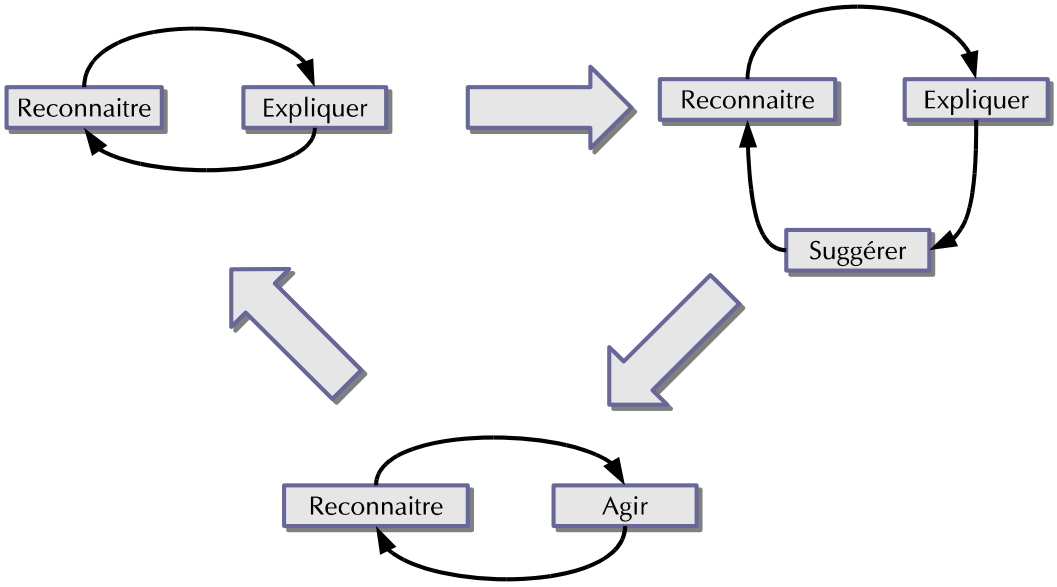

FIGURE 3.5 – Les cycles du plan de connaissance

Le cycle reconnaissance-explication est l'étape durant laquelle le plan de connaissance gagne de l'expérience durant son activité (par l'apprentissage) en se dotant d'une vue sur l'ensemble des états normaux et problématiques de son environnement. Il collecte tout ce qu'il peut avoir comme information pertinente ou pouvant constituer une contrainte sur son activité (objectifs de haut niveau, ressources disponibles, paramètres sur lesquels il peut agir, ...). Ensuite, il tentera de donner une explication pour tous les évènements qui surviennent dans son environnement. C'est l'étape de construction de sa connaissance, de l'apprentissage. Cette étape est cruciale car elle permettra au plan de connaissance d'affiner les décisions qu'il prendra en évitant de réagir face à des situations considérées comme "*normales*".

Le cycle reconnaissance-explication-suggestion est la suite logique du premier. Dans ce cycle, le plan de connaissance, à partir de la connaissance acquise lors de la première étape, fait des suggestions par le mécanisme de raisonnement et en mettant à jour la connaissance en fonction de l'impact des suggestions sur l'environnement.

Le cycle reconnaissance-action lui permet d'agir grâce à sa maturité en termes de fiabilité de sa connaissance. Dans ce cycle, le fonctionnement du plan de connaissance est totalement autonome, et est donc capable de mettre à jour sa connaissance afin de s'adapter à une éventuelle évolution de l'environnement.

Dans un environnement, changeant il peut arriver que la fiabilité de la connaissance du système soit obsolète et donc le processus recommence les cycles de manière incrémentale.

3.2.5 Les services du plan de connaissance

Les sections précédentes montrent à quelle point les limites des services fournis par le plan de connaissance sont diverses et complexes. Cependant, deux cas d'utilisation se dégagent de l'ensemble des publications qui abordent le plan de connaissance. La première utilisation considère le plan de connaissance comme un système de gestion de base de connaissances ne servant qu'à construire, à maintenir les connaissances et à les fournir aux éléments qui en auront besoin. La seconde vision considère le plan de connaissance comme un système actif qui agit sur les éléments à gérer pour permettre l'auto-adaptation du système. Nous détaillons, dans les sections suivantes, ces deux visions.

3.2.5.1 Le plan de connaissance comme système de gestion de la connaissance

Le premier cas d'utilisation du plan de connaissance est la fonction de gestionnaire de la connaissance dans le réseau. Cette fonction de collecte des informations est nécessaire à la construction et à la présentation de la connaissance, ainsi qu'à la gestion de l'hétérogénéité sémantique de la connaissance à travers le réseau, selon le besoin de granularité et de précision de la connaissance. La collecte d'informations

peut se faire à plusieurs niveaux, dans chaque couche de la pile protocolaire, de chaque élément du plan de connaissance (collecte verticale), mais aussi grâce à la distribution et la corrélation entre des connaissances de provenance et de nature différentes. De nombreux de travaux de recherche utilisent cette fonctionnalité en faisant appel au domaine du *Knowledge Engeneering* et aux ontologies. La connaissance est ainsi modélisée sous forme de structures connectées. Dans la réalité, le plan de connaissance doté d'une telle fonctionnalité peut servir à mettre en place un système de détection d'anomalies ou d'intrusions dans un réseau en collectant et fournissant une connaissance plus riche à l'ensemble des éléments qui participent à l'opération de détection. Ainsi, des éléments réseaux traitant des informations de natures différentes peuvent collaborer dans un objectif commun. Le plan de connaissance peut également être utilisé pour répondre à des questions permettant de faire la distinction entre des échecs de communication dûs à des anomalies et ceux causés par les politiques de configuration.

Comme exemple de travaux qui ont cette vision, nous avons :
[Ballani & Francis 2006], [Serrano *et al.* 2007], [Macedo *et al.* 2007], [Kopena & Loo 2008], [Macedo *et al.* 2008] et [Fahy *et al.* 2008]

3.2.5.2 Le plan de connaissance comme système d'adaptation

Le plan de connaissance peut également agir comme un système d'adaptation. Dans ce cas, en plus de constituer une base de connaissance, il pourra agir sur l'environnement pour atteindre un objectif bien précis. Dans cette vision, l'utilisation des systèmes cognitifs est le point le plus important car ils permettent d'améliorer le processus d'adaptation (par l'observation et la reconfiguration) et de prendre en compte de manière automatique les nouveaux cas qui se présentent. Le plan de connaissance, comme système adaptatif, est surtout utilisé pour la résolution de problèmes spécifiques bien identifiés : optimisation, contrôle d'accès, ... Cependant, l'utilisation du plan de connaissance dans ce cadre n'est pas toujours possible car les systèmes cognitifs demandent en général beaucoup de ressources et un certain temps pour la découverte de l'environnement. Cette "*limitation*" est due au fait que le plan de connaissance a d'abord été pensé dans un contexte filaire et porté ensuite dans d'autres architectures. En effet, comme nous l'avons montré dans la

section 3.1, la première proposition du plan de connaissance était surtout pour régler des problèmes ayant traits aux réseaux IP avec, à disposition, une certaine quantité de ressources.

Comme exemple de travaux qui ont cette vision, nous avons :
[Latré *et al.* 2008], [Marbukh 2004], [Omar & Taleb-Bendiab 2006],
[Sawma *et al.* 2008b], [Sawma *et al.* 2008a], [Shen *et al.* 2004],
[Watts *et al.* 2007], [Urra *et al.* 2004].

Il est à noter que la plupart des travaux considérant cette approche sont implicitement hybride car utilisant dans une certaine mesure le plan de connaissance pour construire un modèle de l'environnement, sans toutefois trop insister dessus. Il nous parait, beaucoup plus, cohérent de considérer le plan de connaissance comme un système hybride collectant des informations et les mettre dans un format exploitable pour les éléments réseaux, mais aussi un système qui exécute un processus de prise de décision pour l'auto-adaptation. Cependant, comme pour la gestion autonome, la complexité de la tâche de conception et de réalisation est certainement la cause de ce découpage.

3.2.6 Les défis du plan de connaissance

3.2.6.1 Définition et représentation de la connaissance

Le premier défi auquel les chercheurs du domaine du plan de connaissance devront faire face est la représentation de la connaissance. Dans un premier temps, il s'agira d'identifier ce qui va constituer la connaissance dans le réseau, dans un second temps, de définir une représentation permettant de prendre en compte tous les objectifs du plan de connaissance et les caractéristiques des réseaux tout en étant extensible et flexible.

La définition de la connaissance est un des aspects les plus importants car elle permettra, dès le début, de distinguer les données qui vont intervenir directement dans les processus cognitifs ou qui en sont les résultats directs (que nous appelleront connaissance) de celles qui ne décrivent que la situation de l'environnement en un instant donnée (que nous appellerons information). Tenter de définir la notion de connaissance est plutôt une opération périlleuse car c'est une des nombreuses questions qui ont suscitées des débats philosophiques depuis le temps de la Grèce

antique et qui n'ont pas encore de réponses définitives acceptées de tout le monde. Cependant, il nous est nécessaire de tenter de le faire pour deux raisons :

- Saisir le sens la notion de connaissance dont nous parlons depuis le début du document de manière abstraite ;
- Montrer la différence avec une donnée plate prise isolément (une information).

De manière informelle, la notion de connaissance que nous utilisons dans le langage courant est associée à plusieurs concepts différents dont voici quelques exemples :

- Une relation entre une personne et une expression déclarative, Exemple "*Cathy sait qu'il faut souffrir pour être belle*". Dans cette phrase, l'expression déclarative "*il faut souffrir pour être belle*" constitue une assertion qui peut être vraie, fausse, certaine ou incertaine.
- Une attitude propositionnelle d'un élément. Exemple : "*Cathy pense qu'il faut souffrir pour être belle*". Dans cette situation la connaissance n'est pas une certitude mais juste une certaine perception de l'environnement.
- Une croyance qui juge du degré de précision de la connaissance : "*Cathy croit qu'il faut souffrir pour être belle*".

Dans tous les cas, la notion de connaissance est associée à une certaine représentation de la réalité à l'aide d'une information plus ou moins précise. Pour appuyer encore cette idée le Petit Robert définit le verbe connaitre comme : *Avoir présent à l'esprit (un objet réel ou vrai, concret ou abstrait, physique ou mental) ; être capable de former l'idée, le concept, l'image de.*

Le domaine de l'intelligence artificiel aussi a sa vision de la question. Elle est intimement liée au domaine de la représentation de la connaissance. "*La représentation de la connaissance est le domaine de l'AI qui étudie l'utilisation de symboles pour représenter une collection de propositions qui sont les croyances d'un élément*" [Brachman & Levesque 2004, Davis *et al.* 1993].

Pour ce qui est du plan de connaissance, vu le nombre de modèles et de sémantiques des informations qui sont générés par les équipements et les applications du réseau, c'est un vrai défi de les acquérir, de les traduire et de les représenter sans perte de performance comme le note [Quirolgico *et al.* 2003].

Pour résoudre ce problème, des organismes de standardisation ont proposé des ontologies et des modèles d'informations pour répondre à cette problématique

[Quirolgico *et al.* 2003] :

- le modèle CIM (Common Information Model) définit par le DMTF [1]. CIM est un modèle conceptuel des informations utilisé pour décrire l'ensemble des informations de gestion des équipements informatiques et des réseaux [DMTF 1999].
- La MIB II [McCloghrie & Rose 1991](Management Information Base) définie par l'IETF [2]. La MIB est une collection de spécifications qui définit les propriétés des ressources gérées (Exemple : une carte réseau).
- la SUO (Standard Upper Ontology) [IEEE 2009] de l'IEEE [3]. Cette ontologie est une représentation générale de plusieurs domaines (médecine, finance, ...) qui offre un ensemble de concepts permettant de l'étendre. L'ontologie qui gère la QoS est représentée en utilisant le KIF (Knowledge Interchange Format) [Quirolgico *et al.* 2003]

Ces efforts montrent tout la difficulté qu'il y a à définir ce qui va constituer la connaissance parmi la myriade de caractéristiques disponibles sur les équipements et applications réseaux. Le niveau de granularité de la connaissance est aussi un problème qui se pose en terme de compromis entre précision et performance. [Quirolgico *et al.* 2003] propose, à cet effet, de séparer la connaissance en deux niveaux : le niveau nœud où les informations sont traitées et filtrées et le niveau cognitif qui traite de la connaissance à proprement dite.

Pour résumer cette partie nous pouvons dire que les besoins en termes de définition concernant le plan de connaissance se situent à deux niveaux :

- l'identification et la standardisation des informations qui seront pertinentes pour les processus cognitifs et qui permettront la création de la connaissance ;
- la standardisation d'ontologies définissant l'organisation générale des connaissances afin d'avoir un processus de raisonnement optimisé.

La complexité de la tâche de normaliser un CNKM (Common Network Knowledge Model), qui ne serait pas copie modifiée de CIM (Common Information Base), vient à notre avis du fait que le terme *connaissance* est très ambigüe par rapport aux informations classiques connues dans les environnements réseaux. Pour certains,

1. Distributed Management Task Force
2. Internet Engineering Task Force
3. Institute of Electrical and Electronics Engineers

il n y a pas de différences entre information et connaissance, ce qui est du point de vu des systèmes cognitifs discutable. Le débat reste assez subjectif mais elle est importante pour déterminer quel sera le rôle exact des systèmes cognitifs dans le plan de connaissance et l'*autonomic* en général.

3.2.6.2 Gestion distribuée de la connaissance

La gestion de la connaissance est aussi un défi dans cet environnement réseau qui est fortement distribué. Les éléments du plan de connaissance devant collaborer, des contraintes de passage à l'échelle, de performance, de gestion de la cohérence se posent.

Le passage à l'échelle est une caractéristique très importante et la faisabilité du plan de connaissance en dépend. Un des défis est de définir la manière dont les éléments doivent collaborer de manière efficace sans que cela n'affecte les performances de l'activité principale du réseau : le transport d'informations des utilisateurs. La notion de *vue située* que nous allons présenter plus loin (voir section 3.3.5) est une approche qui offre une première solution mais qui reste encore à être prouvée dans la pratique (à moins de le réduire à un problème de choix de conception). Cependant, il reste une problématique à part qui est celle de déterminer de manière distribuée les limites de la vue située d'un nœud pour prendre en compte le maximum d'informations pertinentes.

Les connaissances étant distribuées dans le réseau, la gestion de la cohérence est aussi un facteur à prendre en compte pour avoir un comportement global cohérent. Aussi il faudra voir jusqu'à quel niveau la cohérence est nécessaire, locale ou globale. En effet, il n'est peut-être pas indispensable dans certains cas d'avoir une cohérence globale du système mais seulement que chaque nœud soit localement cohérent. La condition nécessaire dans ce cas de figure est que les connaissances ne soient pas en compétition à tel point que les optimisations dans une partie du réseau soient annulées par une autre partie. C'est un vrai défi que de garantir cette cohérence sans augmenter drastiquement le nombres de messages à échanger entre les nœuds.

L'hétérogénéité des connaissances dans le réseau ainsi que leurs sémantiques font aussi qu'il y a un besoin d'avoir un système d'inter-médiation pour que les mêmes connaissances aient la même sémantique dans l'ensemble du réseau.

Cette distribution de la connaissance pose aussi le problème de la sécurité d'accès à la connaissance du réseau. En effet, il va être difficile de concilier une collaboration entre les éléments du plan de connaissance se trouvant dans des domaines différents tout en gardant une partie des connaissances privées à chaque domaine. La difficulté vient du fait que la connaissance, qui peut être pertinente pour l'identification ou la corrélation, n'est pas forcement identifiable à l'avance.

3.2.6.3 Supports cognitifs distribués

les systèmes cognitifs constituant le cœur du plan de connaissance, leur utilisation présente aussi un certain nombre de défis, en particulier ceux nombreux en relation avec l'apprentissage [Dietterich & Langley 2003], [Friend *et al.* 2007].

Pour certains problèmes réseaux, comme la reconnaissance des défauts de configuration ou la détection d'intrusion, l'apprentissage supervisé présente un intérêt dans l'optique d'automatiser la résolution des problèmes dans le plan de connaissance. Cependant, beaucoup d'autres problèmes réseaux ne trouvent pas de solutions dans l'application d'un paradigme unique du monde de l'apprentissage. C'est le cas par exemple du domaine de la détection des anomalies en temps réel qui peut être formulé comme des problèmes d'apprentissage non supervisé et d'apprentissage pour l'interprétation. L'apprentissage non supervisé, de manière résumée, est le cas d'un système qui apprend à partir d'un ensemble de données pour découvrir une fonction, un concept, des similarités qui décrivent cet ensemble.

Un autre aspect du problème de choix de paradigme constituant un défi dans le plan de connaissance, est la gestion des niveaux d'abstraction. La détection d'anomalies dans les réseaux par exemple peut être traitée (pour ne pas dire qu'elle l'est souvent) par l'apprentissage en considérant un seul niveau d'abstraction. Cependant, le problème de l'anomalie peut se trouver à plusieurs niveaux d'abstraction : paquets individuels, connexions, protocoles, flux, ou même répartie sur l'ensemble du réseau. Un défi important est de développer des méthodes d'apprentissage supervisé ou non qui peuvent prendre en compte plusieurs niveaux d'abstraction. Ces niveaux d'abstraction permettront, par exemple, au plan de connaissance d'identifier à différents niveaux le même problème pour s'assurer qu'il y a effectivement une anomalie et donc éviter les alarmes inutiles.

Un autre défi est l'exploitation des avantages des types d'apprentissage que sont l'apprentissage à froid et l'apprentissage incrémental. En effet, l'apprentissage à froid (*batch learning*) consiste à réunir les données avant de lancer l'algorithme d'apprentissage. Ce type d'apprentissage a un avantage très intéressant qui est celui de donner, dès la première réponse, une explication du concept en se basant sur l'ensemble de données représentatives du concept qui lui est fourni. Cependant, son véritable inconvénient est le facteur temps d'apprentissage et taille mémoire nécessaires avec ce type d'apprentissage. Comme l'environnement des réseaux est très dynamique, cela impliquerait un renouvellement périodique du processus d'apprentissage pour s'adapter aux changements du réseau. D'un autre coté, l'apprentissage en ligne (in-line) prend les données nécessaires à l'apprentissage de manière incrémentale. L'avantage est que ce type de fonctionnement est bien adapté aux réseaux qui produisent des informations au fil du temps. Cependant, l'apprentissage incrémental demande un certain temps pour atteindre un état stable lorsque c'est possible. Par conséquent, il y a une forte probabilité de faire des choix erronés durant la période de l'apprentissage.

Un autre défi lié à l'apprentissage et au plan de connaissance est la distribution des mécanismes d'apprentissage. La nature fondamentale du plan de connaissance est que c'est un système distribué dans l'ensemble du réseau. La plupart des algorithmes d'apprentissage sont effectués de manière centralisée, en regroupant les données sur une machine pour être ensuite lancés. Dans le contexte des réseaux, chaque nœud dispose localement de données pour apprendre et l'un des défis importants est de déterminer comment et dans quels objectifs les éléments du plan de connaissance distribué dans le réseau vont collaborer pour atteindre les objectifs de haut niveau.

3.3 Concepts corrélés au plan de connaissance

La complexité du plan de connaissance à rendu très difficile la proposition d'architecture complète prenant en compte toutes les fonctionnalités. Comme avec la gestion autonome, la communauté de la recherche a divisé le problème en sous problèmes pour, ensuite, tenter de faire des propositions. Ce découpage a donné naissance à un certain nombre de concepts qui s'inspirent, de prêt ou de loin, du

plan de connaissance. Les sections qui suivent présentent brièvement les concepts les plus répandus.

3.3.1 Plan d'information

Le plan d'information (Information Plane) est un concept qui s'inspire directement du plan de connaissance. Il s'attaque au problème de la gestion des grands réseaux avec des infrastructures fortement distribuées pour le collecte, le stockage, la propagation et la découverte des informations décrivant l'état du système et la réaction en fonction des informations remontées [Wawrzoniak *et al.* 2004, Chun *et al.* 2004]. Le concept est dénommé plan d'*Information* à la place de plan de *Connaissance* car il ne prévoit pas l'utilisation de systèmes cognitifs. L'objectif du plan d'information est plus modeste, peut être parce que les motivations implicites sont d'avoir un système de transition qui puisse être déployé à court terme vers le plan de connaissance.

Pour réaliser de tels objectifs, le plan d'information mise sur les trois fonctions que sont :

- la collecte d'informations à travers le réseau,
- l'évaluation d'instructions/requêtes entre les capteurs et les actionneurs sur l'état du réseau
- et enfin, une réaction appropriée en fonction des conclusions de l'évaluation des instructions.

Deux grands défis se présentent à la réalisation du plan d'information :

- La prise en compte de l'information distribuée à travers le réseau qui varie constamment. La conséquence est qu'il est difficile de concevoir un système exploitant des données aussi instables dans le temps et dans l'espace.
- L'expressivité des requêtes envoyées qui permet de spécifier de manière déclarative les comportements (normaux, attendus et désirés) du réseau. De telles requêtes ne peuvent être que complexes quand on connait la difficulté de définir ce qu'est un comportement réseau.

Le premier prototype de plan d'information a été réalisé à *PlanetLab*. Ce prototype nommé *Sophia* [Wawrzoniak *et al.* 2003, Wawrzoniak *et al.* 2004], modélise les fonctions du plan d'information avec un ensemble de capteurs de reporting sur

l'état du système local ou distribué (Exemple : charge ou usage des ressources d'un nœud particulier ou de l'ensemble du réseau) et fournit un environnement de programmation déclaratif.

Sophia modélise les actions qui peuvent être prises sur le réseau avec un langage logique (Prolog) et des actionneurs qui sont distribués dans le réseau pour interroger les éléments du réseau.

L'ensemble du système fonctionne avec tous les éléments qui jouent en même temps des rôles d'agent et de gestionnaire dans la mesure où ils peuvent être sollicités pour fournir des informations ou bien évaluer une requête d'un autre élément au niveau local.

Il est à noter que malgré la dénomination de plan d'information, *Sophia* reste un système de collecte d'informations uniquement.

L'intérêt majeur d'un tel système est l'utilisation d'un langage expressif (langage logique) pour la collecte d'informations et l'envoi de requêtes aux actionneurs distribués.

Cependant l'architecture réelle du système reste encore à définir avec des paramètres importants jouant sur le passage à l'échelle du système tels que la fréquence des requêtes pour maintenir le système à jour, le mode de communication des requêtes (*braodcast*, *multicast*). Le processus d'évaluation des requêtes en cas d'informations contradictoires ou incomplètes reste, également, un problème majeur non résolu avec cette solution.

Le second prototype (plus abouti) a été implémenté sous le nom de *iPlane* [Madhyastha *et al.* 2006]. Il s'attaque à un problème un peu plus concret qui est celui de la prédiction de performance des chemins sur Internet. *iPlane* est un service distribué qui effectue continuellement des mesures sur le réseau (délai, latence, bande passante, taux de pertes sur les liens) et génère en sortie un plan annoté d'Internet, avec un ensemble riche d'attributs de routeurs et des liens.

Des informations structurelles qui peuvent être la topologie au niveau réseau comme au niveau système autonome (AS) sont utilisées pour prédire des chemins entre deux nœuds quelconques du réseau. Ces chemins prédits sont combinés ensuite à des informations caractéristiques de chaque segment (constituant ces chemins) pour prédire, enfin, des propriétés de chemins de bout en bout pour un certain nombre de métriques comme la latence, la bande passante disponible et les taux de

perte.

*i*Plane peut aussi analyser des anomalies ou avoir une vue globale du réseau en faisant une corrélation des observations de différentes parties du réseau. L'ensemble du système est présenté comme un service réseau interrogeable avec des langages SQL-Like pour obtenir des informations sur l'état du réseau (voir Figure 3.6 tirée de [Madhyastha *et al.* 2006]).

Le système a été testé sur divers environnements et semble donner, dans l'ensemble, de bons résultats qui montrent que le système est fonctionnel (voir pour plus de détails [Madhyastha *et al.* 2006]).

En marge de ces deux implémentations, il existe d'autres réalisations [Huebsch *et al.* 2003, Campbell *et al.* 2005] qui peu à peu s'éloignent de l'idée du plan d'information pour s'orienter vers le domaine de l'interrogation d'informations de performance (*Performance Quering*).

3.3.2 Réseaux de Connaissances

Les réseaux de connaissances (*Knowledge Networks*) s'attaquent au problème de savoir comment mettre ensemble des connaissances contextuelles issues du réseau physique et des "connaissances sociales" issues des utilisateurs des réseaux de communication pour former un espace de connaissances exploitables par les éléments et applications "autonomiques".

Le concept part du principe qu'un ensemble des données contextuelles plates ne suffit pas pour avoir un comportement autonome dans les réseaux mais qu'il faudrait une organisation des informations de contexte en "réseaux de connaissances" pour pouvoir être exploitée par les éléments réseau et les applications. En effet, une bonne organisation sémantique des connaissances du réseau qui sont hétérogènes, distribuées et de sources différentes, peut permettre une exploitation efficace à travers le réseau par les différents équipements et applications (qui peuvent être très hétérogènes).

Les réseaux de connaissances apparaissent, donc, comme une forme d'*overlay* de connaissances, distribuée dans un réseau et faisant partie intégrante de l'ensemble de l'infrastructure, organisées et corrélées. Cette structure contient donc des données sémantiquement riches sur lesquelles les éléments de l'infrastructure autonome

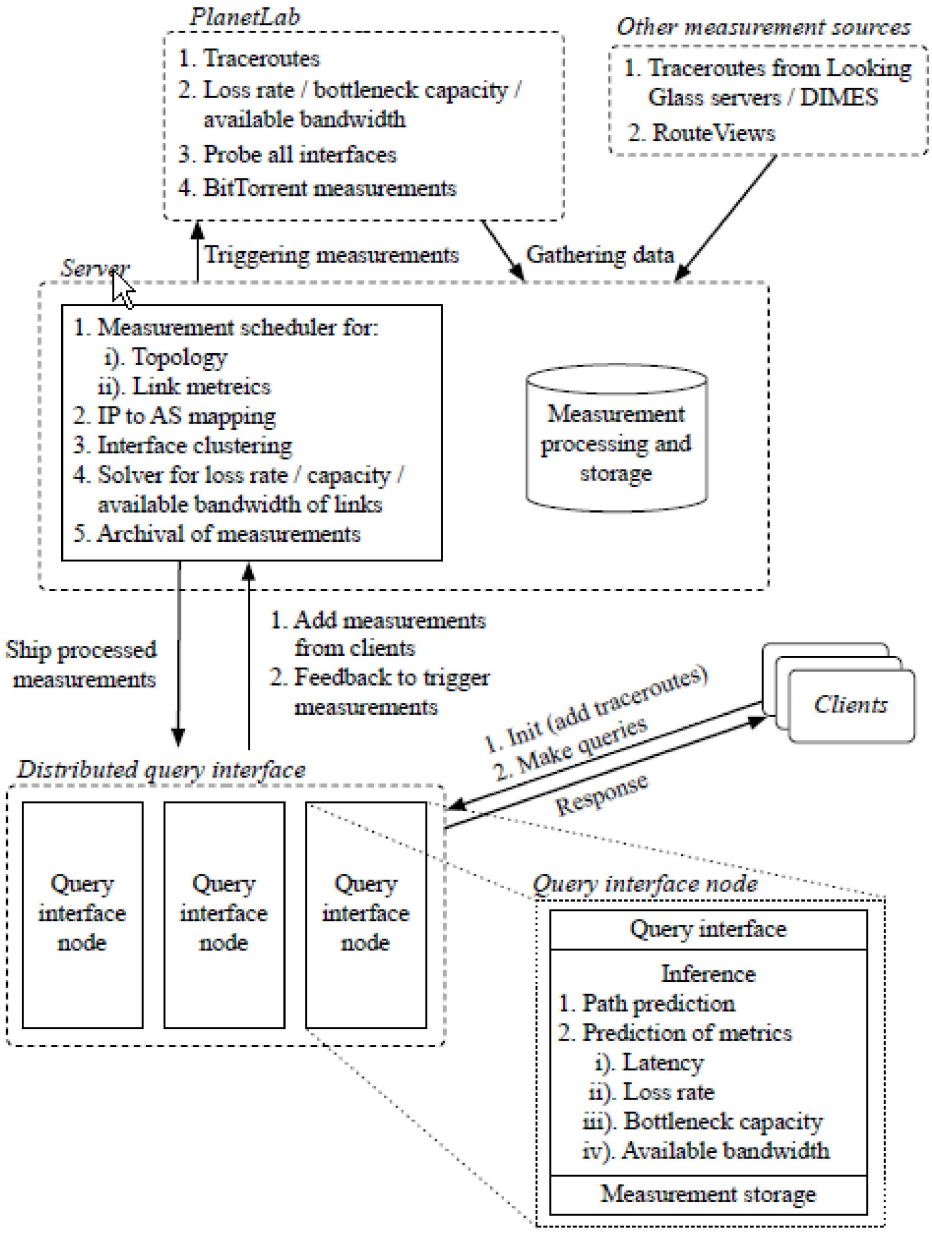

FIGURE 3.6 – Architecture de *i*Plane

peuvent interagir. Ici, implicitement, c'est la fonction de *self-awareness* qui est visée sans toujours être explicitement dit ou en la remplaçant par des concepts très proches qui sont le *context-awareness* et le *situation-aware*.

Autrement dit, c'est une infrastructure de connaissances distribuées représentant une sorte de système nerveux autonome de communication, tous les stimuli et les informations sont nécessaires à la coordination du fonctionnement du système de flux ainsi qu'à son organisation.

[Mulvenna *et al.* 2006] présentent quatre raisons qui peuvent justifier le besoin de réseaux de connaissances :

- Il y a un besoin fondamental d'outils expressifs et flexibles pour promouvoir le *context-awareness*. En effet, les informations de contexte peuvent avoir différents niveaux de granularité et il faut un modèle qui permette l'expressivité et la flexibilité pour prendre en compte et traiter les différents niveaux existants. Une bonne organisation de la connaissance, comme le suggère le concept de réseaux de connaissances, pourrait permettre d'avoir des structures conceptuellement simples et flexibles.
- Les informations de contexte ne sont pas simplement des données locales sur chaque élément, mais une combinaison/corrélation de toutes les connaissances. Pour une orchestration des activités distribuées, un bon compromis est de permettre aux équipements qui ont besoin de plus d'informations locales, de pouvoir naviguer dans la structure, sous forme de réseau, de la connaissance disponible dans l'ensemble du réseau. Ce cas de figure est d'ailleurs exploité dans [Greer *et al.* 2008].
- L'utilisation des structures *Cross-Layer* montre que le niveau service et le niveau réseau ne devraient pas fonctionner de manière séparée chacun suivant son propre objectif. Cependant, des échanges entre ces différents niveaux ne se feraient pas sans problème d'interopérabilité, à moins de passer par un processus de médiation. Les réseaux de connaissances pourraient constituer une structure dont la sémantique est compréhensible par l'ensemble des niveaux de gestion du réseau.
- L'auto-organisation est une des fonctions importantes des réseaux autonomes car elle permet aux équipements de coordonner leurs activités en échangeant des informations. Les réseaux de connaissances peuvent permettre de gérer les

différents aspects liés à l'hétérogénéité des éléments réseaux et des applications qui communiquent (exemple d'utilisation dans [Baumgarten *et al.* 2007]).

Les auteurs dans [Mulvenna *et al.* 2006] font la distinction entre les concepts de plan de connaissance et celui des réseaux de connaissances, notamment en mettant en avant le fait que la vision plan de connaissance de [Clark *et al.* 2003b] "*prône*" une séparation de la gestion de la connaissance dans un plan spécifique alors que, dans la proposition qui concerne les réseaux de connaissances, ils penchaient pour une approche *Cross-Layer* de la gestion de la connaissance.

Cependant, cette différentiation est moins claire dans la réalité. La figure 3.7 (tiré de [Mulvenna *et al.* 2006]) représente le fonctionnement global d'une architecture avec un réseau de connaissances. Elle définit une structure de médiation entre des couches qui ne communiquent pas directement. Un autre aspect, qui met fortement en relation les deux concepts, est que les réseaux de connaissances utilisent fortement les ontologies dans l'objectif de raisonner sur les connaissances afin d'en sortir des connaissances exploitables par le niveau qui demande l'information. Ce raisonnement nous ramène à un début d'utilisation des systèmes cognitifs et donc en fin de compte ne constitue que la réponse à une partie des problèmes liés à la connaissance.

De plus, dans la définition du plan de connaissance, le principe de séparer le nouveau plan, du plan de contrôle et du plan de données, n'est pas dans l'optique de les déconnecter. D'une part, les informations qui permettent la construction de la connaissance dans le réseau viennent des différentes couches de gestion mais aussi de différents endroits, et, d'autre part, le plan de connaissance a dans ses fonctionnalités la possibilité de fournir la connaissance à tous les éléments de gestion. Le *Cross-Layering* est donc bien le modèle de fonctionnement des éléments du plan de connaissance.

3.3.3 Réseaux Cognitifs

Les réseaux cognitifs (Cognitive Networks) [Ramming 2004], [Boscovic 2005], [Thomas *et al.* 2005] représentent un concept qui vient du monde des communications sans fil. Il est apparu presque en même temps qu'un autre concept qui est celui de la radio cognitive [Thomas *et al.* 2005]. Bien que ces termes semblent proches,

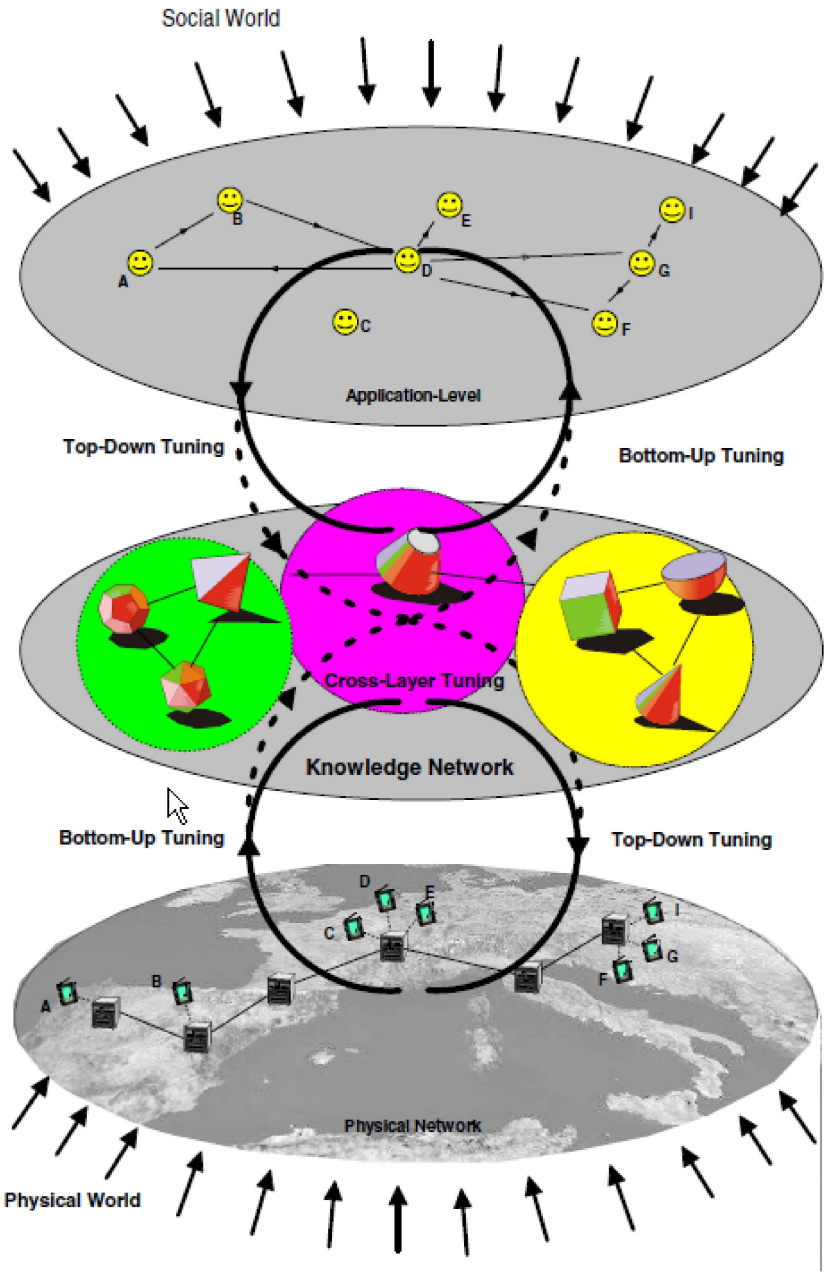

FIGURE 3.7 – Les Réseaux de connaissances

car, dans la réalité, ils sont souvent cités ensemble comme dans [Nolan & Doyle 2007, Boscovic 2005], ils sont assez différents pour constituer deux domaines de recherche à part. Pour montrer cette différence, nous allons commencer par présenter le

concept de radio cognitive.

Dans le domaine des communications sans fil, la gestion du spectre est régulée. Certaines fréquences sont attribuées aux opérateurs qui les utilisent exclusivement tandis que d'autres sont libres. Cependant, des études comme celle de [McHenry & Steadman 2005] ont montré que les fréquences allouées aux opérateurs sont largement sous utilisées alors que les fréquences libres (comme celle de la IEEE 802.11) sont devenues des goulots d'étranglement. Le concept de radio cognitive a pour objectif de permettre l'accès dynamique au spectre en adaptant les paramètres d'émission en fonction des changements de l'environnement . Cette notion de radio cognitive a été introduite par [Mitola III 2000] et le concept y est présenté comme faisant référence à un système radio intelligent qui a la capacité de capter des données de l'environnement externe, d'apprendre de l'historique et de prendre des décisions pour ajuster ses paramètres d'émission en fonction de l'état de l'environnement. Le domaine est très florescent depuis quelques années car la radio cognitive se veut une approche permettant d'offrir une certaine flexibilité afin d'obtenir des gains importants en terme d'efficacité d'utilisation du spectre.

Pour ce qui est des réseaux cognitifs, l'origine du nom est sans doute inspirée de celui de la radio cognitive mais le concept lui même prend ses sources dans le plan de connaissance et les réseaux autonomes [Qusay 2007] (*autonomic computing* plus exactement). Il existe plusieurs définitions de cette notion de réseaux cognitifs qui ont été proposées. [Ramming 2004] considère qu'un réseau cognitif est un réseau qui doit être *self-awareness* et pouvoir prendre des décisions de configuration dans le but de réaliser un objectif précis dans un contexte bien défini. Il avance aussi qu'un réseau qui doit s'autogérer a besoin de technologies cognitives et d'une connaissance des objectifs de la tâche qu'il est entrain d'effectuer. [Sifalakis *et al.* 2004] définit le terme "*cognitif*" lorsqu'il est associé au terme réseau comme désignant un réseau capable de s'auto-adapter en réponse à des événements et des conditions en se basant sur le raisonnement artificiel et une connaissance acquise au fur et à mesure par le réseau. Il existe également beaucoup d'autres variantes de ces définitions, cependant, il ressort de toutes ces définitions un certain nombre d'attributs des réseaux cognitifs que sont : la capacité d'apprendre et de raisonner dans un environnement distribué [Friend *et al.* 2007] afin d'avoir un comportement autonome [Strassner 2007].

Le plan de connaissance est donc utilisé pour ses supports cognitifs mais son

objectif est d'atteindre l'autonomie. Il est à noter que même si les réseaux cognitifs ont des racines très profondes dans les problématiques des réseaux sans-fil, ils peuvent être utilisés comme approche dans d'autres domaines comme les réseaux autonomes. D'ailleurs, la modélisation des réseaux cognitifs, comme le soulève [Thomas *et al.* 2005], s'appuie sur l'approche des boucles de contrôle largement utilisées dans les réseaux autonomes. Le plan de connaissance intervient en ajoutant au cycle OODA (Observe, Orient, Decide, Act), de l'apprentissage pour le rendre plus efficace. Notre approche s'inscrit beaucoup plus dans cette vision des réseaux cognitifs

3.3.4 Plan d'inférence

Le concept de plan d'inférence (*Inference Plane*) a été développé comme une première étape vers le plan de connaissance [Strassner *et al.* 2007]. Il a pour ambition de proposer une technologie fonctionnelle à court terme sans avoir à remettre en cause radicalement les technologies déjà existantes. Il a aussi pour objectif de répondre à certains aspects de la proposition initiale du plan de connaissance considérés par les auteurs du plan d'inférence comme des incohérences et/ou des manquements.

En effet, le plan de connaissance est supposé fonctionner en respectant les objectifs business des concepteurs du réseau tout en optimisant le fonctionnement du réseau. Cependant, aucune définition ou indication n'est donnée sur la manière de spécifier ces objectifs dans le système pour gérer les services et ressources réseaux quelque soit la situation. Le plan d'inférence répond à ce problème grâce à des notions importées de la gestion par politique, et en particulier de l'architecture DEN-ng [Strassner 2004].

Un second aspect du plan de connaissance, qui fait parti de la motivation du plan d'inférence, est la liaison forte de la conception du plan de connaissance à l'architecture d'Internet en particulier et des réseaux filaires en général. La gestion des problématiques des réseaux sans fil comme le paradigme des réseaux ad hoc, la mobilité, les topologies dynamiques ne sont pas prises en compte. Le plan d'inférence tente d'apporter une solution à cela avec une architecture de gestion sous forme de grille de calcul(Voir figure 3.8 tirée de [Strassner *et al.* 2007]).

La conception du plan d'inférence s'est faite dans la logique de son intégra-

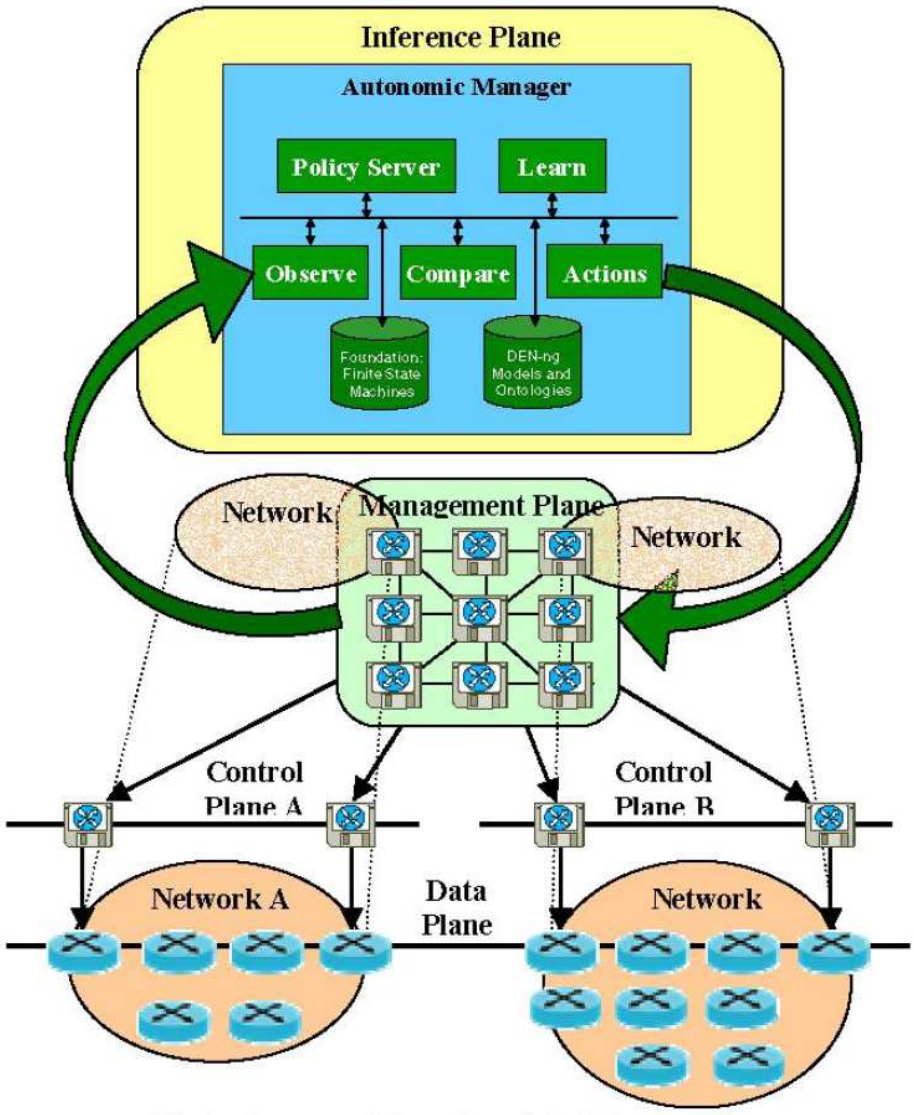

FIGURE 3.8 – Vue conceptuelle du plan d'inférence

tion à l'architecture autonomique FOCALE [Strassner *et al.* 2006]. L'architecture globale est organisée sous forme de grille virtuelle d'éléments programmables qui gèrent plusieurs réseaux de types différents tout en coordonnant leurs actions. La connaissance est organisée sous forme de sémantiques riches.

De ce fait, cette architecture hérite fortement de la gestion par politique. L'idée du plan d'inférence est très intéressante mais, intégré à l'architecture FOCALE, le tout devient une structure très complexe à mettre en œuvre.

3.3.5 Vue située

Le terme *vue située* (*Situatedness*) est une traduction en français d'un concept qui correspond à plusieurs termes en anglais : *Situatedness*, *Situation-aware*, *Location-aware*. Le plan de connaissance doit fournir la connaissance là ou elle est nécessaire et dans un format compréhensible par le nœud qui la reçoit. Cependant, la gestion de l'hétérogénéité dans une structure comme le plan de connaissance qui gère des éléments réseaux ayant des capacités et des rôles différents est un des grands défis du domaine. La vue située a été introduite avec l'objectif d'adresser ce problème [Bullot *et al.* 2008, Nguengang *et al.* 2008].

Le concept *situated* vient des chercheurs qui ont une vision orientée agents autonomes pour la conception de l'*autonomic* et de la réalisation du plan de connaissance. Importé du monde des systèmes multi-agents, le sens de la vue située a glissé peu à peu pour s'éloigner de ce monde. En effet, dans le monde des systèmes multi-agents, le *situateness* est une propriété des agents autonomes [Florian 2003, Wei 1998]. [Franklin & Graesser 1996] définit un agent autonome comme étant "*un système situé (*situated*) dans un environnement y faisant partie, agissant sur l'environnement au fil du temps ...*". Dans cette définition la notion de *situatedness* est plutôt une position de l'agent dans un environnement lui permettant d'évoluer. [Florian 2003] ajoute que cette notion correspond à la qualité de l'agent qui est d'être dans un environnement, de percevoir les événements qui s'y produisent et d'interagir avec lui.

Dans le domaine du plan de connaissance, la vue située a une connotation légèrement différente. Elle représente les limites au delà desquelles un agent n'a pas besoin d'aller, pour chercher ou diffuser sa connaissance [Nguengang *et al.* 2008] du domaine de l'auto-organisation. C'est donc un paradigme de gestion de la connaissance distribuée qui s'attaque surtout au problème de passage à l'échelle par la détermination des limites de recherche et de partage de la connaissance.

Il y a également une autre différence importante entre ces deux visions. Dans la vision du monde des agents, la *situation* fait partie intégrante de la définition d'un agent autonome. Dans ce domaine, il est possible de faire des raisonnements avec comme hypothèse que cette propriété est acquise. Au contraire, dans le domaine du plan de connaissance, l'optimalité de la *situatedness* face à d'autres paradigmes

comme le peer-to-peer [Bonifacio *et al.* 2002, Brügge *et al.* 2004, Bonifacio *et al.* 2004] reste encore à être prouvée. On peut imaginer, cependant, s'appuyer sur du peer-to-peer pour localiser une vue située. Et, néanmoins, cela n'empêche pas le fait que le paradigme soit très intéressant, c'est peut être la raison pour laquelle la vue située a reçu beaucoup d'attention de la part du monde de la recherche à travers des projets de gestion autonome comme CASCADAS (Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services) par exemple.

3.3.6 Gestion orientée connaissance

La gestion orientée connaissance (*Knowledge-Based Networking*) [Jones *et al.* 2008] est une approche qui est surtout basée sur l'ingénierie des connaissances. Elle cherche, principalement, la mise en place de relations entre les différentes connaissances du réseau. C'est un concept qui ressemble, à beaucoup d'égards, aux réseaux de connaissances. Nous le citons seulement dans le but d'être exhaustif mais le projet porteur de ce paradigme semble être abandonné.

3.4 Plan de connaissance et réseaux autonomes

3.4.1 Positionnement du plan de connaissance vis-à-vis de la gestion autonome

La question de savoir comment le plan de connaissance se positionne par rapport à la gestion autonome s'est très vite posée. Notamment, [Lewis 2004] nous fait remarquer qu'il y a beaucoup de similitudes entre les objectifs de ces deux concepts. En effet, [Clark *et al.* 2003b] décrit le plan de connaissance comme un système fonctionnant en parallèle avec les autres plans de la gestion de réseaux. Cependant, le fonctionnement attendu du plan de connaissance ne doit pas seulement se limiter à la supervision et à l'analyse de la connaissance mais doit utiliser, également, des mécanismes de planning, d'exécution de boucles de contrôle et de tâches de gestion. Or, ces derniers mécanismes rappellent ceux de la gestion autonome.

Ceci amène à se poser la question de savoir s'il faut considérer le plan de connais-

sance comme une architecture de gestion autonome à part ou bien considéré comme une composante des architectures de gestion autonome.

Comme la notion de boucle OODA est au centre de la gestion autonome, et qu'elle existe dans le plan de connaissance qui effectue des tâches de configuration, d'optimisation, de protection et de réparation, on pourrait argumenter pour la deuxième option.

D'ailleurs, [Lewis 2004] conclut par l'idée selon laquelle le plan de connaissance est une étape vers l'autonomie initiée par le monde de l'Internet qui est caractérisée par un pragmatisme d'évolution de proche en proche au lieu de partir sur une vision globale.

Cependant, dans la réalité, l'utilisation du plan de connaissance est un peu différente. Les projets de gestion autonome, et les chercheurs plus généralement, s'approprient le concept de plan de connaissance de manière parcellaire selon leurs objectifs. Ainsi, le plan de connaissance devient assez facilement une fonctionnalité des architectures de gestion autonome au lieu d'être considéré comme une architecture à part entière à laquelle les nouvelles architectures devraient faire référence pour se positionner. Le plan de connaissance est utilisé dans cette vision comme un outils de gestion de la connaissance, de réalisation de systèmes auto-adaptatifs, ...

En marge de ces deux visions, d'autres approches ont abordé [Strassner 2007] la question dans l'autre sens, à savoir ce que pourrait apporter la gestion autonome au plan de connaissance. Cette vision, même si elle est intéressante, reste assez marginale.

Enfin, les deux concepts se sont très vite trouvés corrélés dans la communauté des chercheurs. Ceci a d'ailleurs eu comme conséquence d'avoir très peu de projets spécifiquement sur le thème du plan de connaissance mais plutôt des projets de gestion autonome qui incluent des aspects du plan de connaissance.

3.4.2 Plan de connaissance dans les projets *autonomics*

Au niveau de la première architecture de réseau autonome, celle d'IBM, il n'était pas encore question de fonctionnalités de systèmes cognitifs au centre de la boucle de contrôle. La vision d'IBM s'appuyait en grande partie sur les méthodes d'ingénierie. Certes, on parlait de la connaissance et de l'utilisation de l'apprentissage

mais la connaissance était plate et partagée entre les fonctions MAPE et l'apprentissage peut utilisé. Contrairement à cette architecture, les plus récentes comme AutoI, ANA, CASCADAS, 4WARD, ..., mettent en avant l'utilisation du plan de connaissance comme une entité nécessaire aux réseaux autonomes.

L'architecture FOCALE avec son plan d'inférence se donne clairement comme objectif de marier les deux concepts pour atteindre l'autonomie. C'est d'ailleurs l'un des premiers projets majeurs à mettre en avant la nécessité d'un plan de connaissance dans une architecture autonome.

Cependant, l'utilisation du plan de connaissance dans les projets de réseaux autonomes varie fortement selon les objectifs de recherche. Au lieu d'avoir une structure bien définie qui prend la même forme partout, chacun s'approprie l'aspect du concept qui lui semble intéressant. Les réseaux autonomes ont été le domaine d'expérimentation des concepts du plan de connaissance par excellence. Certains utilisent le plan de connaissance comme un système de gestion de la connaissance en utilisant surtout le raisonnement sur les ontologies alors que d'autres mettent en avant le côté boucle de contrôle.

3.5 Conclusion

Dans ce chapitre, nous avons présenté le plan de connaissance, un concept, qui, certes n'est pas encore totalement défini mais qui est porteur dans la gestion des réseaux du futur. Le but premier du plan de connaissance est l'utilisation des systèmes cognitifs que sont l'apprentissage et le raisonnement pour pouvoir exploiter la connaissance du réseau à des fins de gestion. La connaissance du réseau étant, à notre opinion, les règles, corrélations, qui peuvent être apprises et utilisées pour la gestion du réseau, par opposition aux informations plates qui ne servent qu'à signaler une situation ponctuelle des équipements et/ou ressources. Cette définition du plan de connaissance s'appuyait fortement sur les conditions en termes de ressources, disponibilités des réseaux composant Internet.

Le plan de connaissance a un ensemble de caractéristiques génériques qui permettent une certaine portabilité du concept dans d'autres environnements réseaux. C'est une structure distribuée et hiérarchique pour permettre d'avoir plusieurs niveaux de vue de la connaissance du réseau (granulaire ou synthétique/globale).

Le plan de connaissance est devenu par la suite un concept clé des architectures des réseaux autonomes. Cependant, le fait que le plan de connaissance ne soit pas totalement défini a donné une certaine liberté aux architectes de réseaux autonomes sur leur approche du plan de connaissance. Ces approches peuvent être classées en deux catégories suivant le rôle du plan de connaissance dans l'architecture. Une première catégorie considère surtout le plan de connaissance comme un système de gestion et de distribution de la connaissance, auquel cas le plan de connaissance se charge de la collecte des informations, de la production des connaissances (ou des relations entre les connaissances) et de la distribution aux éléments qui en ont besoin pour effectuer leur adaptation. La seconde approche considère le plan de connaissance comme un système d'auto-adaptation complet, avec la possibilité de prendre des décisions de configuration. Nous avons retenu une approche hybride qui nous parait plus refléter la réalité du plan de connaissance.

La plupart des architectures autonomes modernes incluent plus ou moins un plan de connaissance, parfois même sans le nommer ainsi. Dans ces cas, l'architecture comporte au moins un module d'apprentissage qui sert à fermer la boucle de contrôle. Cependant, peu d'architectures donnent une idée claire sur ce que l'apprentissage peut réellement réaliser comme tâche. Parfois, l'approche parait même "naïve", considérant l'apprentissage comme un outil qui marche dans tous les cas et qui sait tout faire, une boite à outils "*magique*" qui devine l'objectif de sa conception.

Or, dans la plupart des problèmes d'apprentissage, il y a une nécessité de formaliser le problème, faire des suppositions sur les conditions d'apprentissage (biais) pour préciser l'objectif de l'apprentissage et limiter l'espace de recherche. C'est ce travail que nous avons mené et qui sera présentée les chapitre 4 et 5. Nous allons ainsi proposer une formalisation de la problématique de l'auto-adaptation dans les réseaux autonomes, faire un panorama de l'apprentissage et identifier la technique d'apprentissage qui est la mieux adaptée à cette problématique et montrer les limites de cette technique par rapport aux contraintes réseaux, pour ensuite finir par proposer une méthode alternative.

Deuxième partie

Proposition d'une méthode d'apprentissage distribué et collaboratif

CHAPITRE 4

Apprentissage pour l'auto-adaptation

Sommaire

4.1 Introduction

Les chapitres 2 et 3 ont permis de présenter la place de l'apprentissage artificiel dans les architectures autonomes et son utilisation afin d'automatiser l'adaptation et d'améliorer, ainsi, l'utilisation des ressources. Dans ce chapitre, nous allons proposer une formalisation de l'auto-adaptation, dans un premier temps, afin d'identifier les différents paramètres à prendre en compte dans cette problématique. Ensuite, dans un second temps, nous allons faire une étude panoramique des techniques d'apprentissage. Cette étude permettra de faire apparaitre les caractéristiques qu'un technique d'apprentissage doit avoir pour être applicable à l'auto-adaptation dans les réseaux autonomes.

Nous présentons, par la suite l'apprentissage par renforcement qui est une technique d'apprentissage qui a une problématique très proche de celle de l'auto-adaptation. Nous montrons ensuite quelle sont les contraintes de ce type d'apprentissage qu'il n'est pas toujours possible de remplir dans les environnement réseaux.

4.2 L'auto-adaptation : modélisation et formalisation

4.2.1 Motivations : boucle de contrôle

La problématique de l'auto-adaptation consiste à essayer de trouver les *meilleures* actions (s'il y a lieu d'agir) à effectuer sur une ressource [1] à gérer pour assurer un fonctionnement *optimal*. L'auto-adaptation est une combinaison de l'*auto-configuration* et de l'*auto-optimisation* dans le sens où elle cherche à reconfigurer une ressource dans le but de s'adapter à l'environnement mais également optimiser le fonctionnement de l'équipement par rapport aux objectifs des concepteurs. Dans les architectures de réseaux autonomes, cette fonction est modélisée par une boucle de contrôle fermée (figure 4.1). La récupération des informations permet de connaitre la situation de la ressource pour, ensuite, entamer le processus de prise de décisions sous forme d'instructions de configurations.

1. Ici le terme ressource est utilisé au sens général du terme et donc englobe le cas d'un ensemble de ressources.

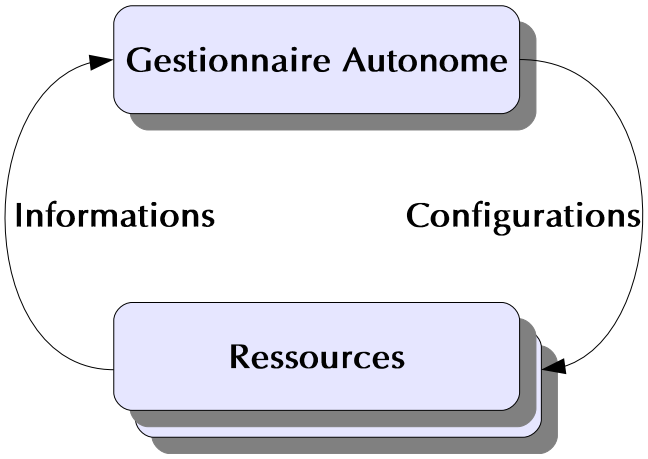

FIGURE 4.1 – Boucle de contrôle pour l'auto-adaptation

L'objectif de ces opérations de configuration peut être multiple :

1. Maintenir le système dans un *état* stable. Par stabilité, nous entendons le fait que le système soit dans les cas extrêmes (dans le cas d'un environnement très perturbé) capable d'assurer la *QoS*, par exemple. Ainsi, le système va essayer de filtrer tout élément perturbateur de l'environnement pour son bon fonctionnement, et de se reconfigurer en fonction de ses objectifs premiers.
2. Optimiser l'utilisation de la ressource à gérer, en adaptant les configurations de la ressource aux changements de l'environnement. L'optimisation est une notion qui dépend fortement des objectifs de haut niveau car c'est l'administrateur (ou les concepteurs du système) qui détermine ce qu'il veut optimiser en priorité quand le système est confronté à un problème de compromis. De manière générale, l'optimisation peut consister à :
   (a) Eviter toute allocation superflue de ressources dans le système. La tâche du système, dans ce cas, est d'éviter tout sur-dimensionnement pour assurer la qualité de services et jouer sur les mécanismes de ré-allocation de ressources pour s'optimiser.
   (b) Maintenir le temps de disponibilité du système. Le système, ici, cherche à maximiser le maintien à un taux d'utilisation dans le temps des ressources qui sont gérées. Ainsi, les opérations de configuration qui permettent de réagir ou de prévenir efficacement toute indisponibilité du

système seront considérées comme optimales.

(c) Respecter les contraintes des applications qui utilisent les ressources. L'optimalité, dans ce cas, est davantage orientée vers le niveau de satisfaction des utilisateurs par rapport aux contrats de services. L'économie de ressources peut, dans certains cas, passer au second plan par rapport aux besoins du client.

Ces objectifs ne sont pas toujours en réalité indépendants et/ou distincts. La stabilité du système, par exemple, peut être déterminée par rapport au critère d'optimalité de l'utilisation des ressources. L'optimisation est, donc, un aspect important dans la définition de l'auto-adaptation. La problématique de l'auto-adaptation peut, donc, être posée en termes d'étude de l'optimalité des opérations de configuration. Définir l'objectif de l'optimisation revient par conséquent à traduire les objectifs de haut niveau en termes d'objectifs techniques interprétables par le système : définir les critères de performance du système. Nous proposons de formaliser le problème de l'auto-adaptation comme suit :

Définition 4.2.1.1 *Soient :*

P_r *Critère de performance défini par le concepteur,*

E_v *Environnement (ressources et éléments externes à un équipement à gérer),*

R *Ressource à optimiser suivant le critère de performance* P_r*,*

Ω *L'ensemble des opérations de configuration possibles sur l'équipement,*

I *L'espace des informations de la situation ou du contexte qui peuvent être obtenues de l'équipement.*

Le problème de l'auto-adaptation consiste à trouver la fonction $h_R : I \rightarrow \Omega^*$ *qui associe à chaque information* $i_0 \in I$ *un ensemble d'actions* $h_R(i_0) = \{\omega_1 \dots \omega_n\}$ *tel que les transitions* $\delta_{E_v}(i_0, \{\omega_1 \dots \omega_n\}) = i'$ *dans l'environnement* E_v *vérifient :*

- $P_r(i_0) < P_r(i')$*, les actions* $(\{\omega_1 \dots \omega_n\}$ *améliorent* i_0 *vers une situation meilleure* $i')$*.*

Autrement dit, il s'agit de trouver dans un environnement donné E_v, suivant les informations $i \in I$ qui remontent de l'équipement, l'ensemble des opérations $\{\omega_1 \dots \omega_n\}$ de configuration qui optimisent la ressource r_i en respectant le critère

de performance P_r. Les informations qui sont remontées peuvent être un ensemble de valeurs de paramètres telles les taux de pertes, les tailles actuelles des files d'attentes, la configuration actuelle de l'équipement, ... Ces informations permettent, donc, une description claire de la situation dans laquelle se trouvent les indicateurs de performances de la ressource à gérer, ceci permettra au processus auto-adaptatif d'identifier les actions à entreprendre avec une certaine précision. Les éléments de l'environnement peuvent être le type de flux qui traverse le réseau en un instant donné, la présence d'un certain nombre de voisins, le caractère fixe ou mobile des voisins, par exemple, ou encore un environnement filaire ou sans-fil. Le comportement de l'environnement E_v est une donnée *a priori* inconnue de l'élément qui s'adapte. Ainsi, on peut considérer l'auto-adaptation comme une estimation de ce modèle de comportement vis-à-vis des opérations de configuration disponibles localement.

Dans la définition 4.2.1.1, Ω^* désigne l'union de l'ensemble des puissances cartésiennes k^{eme} de Ω, en d'autres termes :

$$\Omega^* = \Omega \cup \Omega^2 \cup \Omega^3 \cup \ldots = \bigcup_{k=1}^{\infty} \Omega^k \tag{4.1}$$

Cette grandeur est définie pour permettre la prise en compte des cas où il peut être nécessaire de faire plusieurs opérations de configuration simultanément. Aussi, dans ce cas, pour ne pas avoir des chaines infinies, il faudrait définir un nombre *fini* de répétitions d'une opération sur une chaine d'opérations à faire afin d'éviter une boucle infini.

Le critère de performance permet de rendre *objectif*, quantifiable, l'ordonnancement de l'ensemble des situations $i \in I$ dans lesquelles l'équipement peut se trouver en regroupant (dans des parties) les situations qui sont considérées comme équivalentes. Cela permet à un processus auto-adaptatif de déterminer si les conséquences d'un certain nombre d'actions sont positives ou pas. Cependant, tel qu'il apparait dans la définition 4.2.1.1, le critère de performance devrait ordonner l'ensemble I. Il sert donc de mécanisme de mesure de performance dans l'ensemble I. Nous choisissons de confondre la mesure de performance avec le critère qui en découle. Cependant, si I est infini, comme c'est souvent le cas dans la réalité, alors il y a un nombre infini de situations possibles. Or, un nombre infini de situation pose un risque sérieux de ne jamais pouvoir utiliser l'historique pour améliorer l'adaptation (si chaque situation n'est rencontrée qu'une seule fois). Il vaut mieux, dans ce cas,

regrouper les éléments de I.

Pour définir P_r plus formellement, nous proposons :

Définition 4.2.1.2 *Un critère de performance P_r associe une des valeurs, prise dans un ensemble V, à des éléments d'une partition de I : $P_r(I_k) \in V$.* [2]

Cette définition permet de proposer une expression des critères de performance auxquels doivent se référer les processus auto-adaptatifs pour évaluer l'efficacité des actions à réaliser. Le critère de performance est donc une organisation de l'ensemble I en une famille d'ensembles $\{I_k\}_{k\in 1\ldots n}$ constituant une partition de I telle que toutes les situations dans un même ensemble expriment le même degré de satisfaction du critère de performance. Ce regroupement permet de réduire le nombre de cas à considérer. La difficulté résidant dans l'opération de découpage des ensembles de manière efficace. Les valeurs auxquelles on associe chaque ensemble I_k représentent à quel point il est souhaitable que l'équipement se mette dans cette situation. Il est donc important que cet ensemble de valeurs soit totalement ordonné pour pouvoir décider si une opération a un effet positif, négatif ou est sans effet au niveau du processus adaptatif. Si ces valeurs ne sont pas distinctes on peut arriver à des cas d'ambigüité où l'équipement ne *sait* pas si une situation décrite par i_1 est meilleure que i_2. A titre d'exemple, considérons une information simple constituée du taux de pertes et du délai de bout-en-bout : $i_1 = (10\%, 5ms)$ et $i_2 = (5\%, 10ms)$. Si le concepteur du système donne la même valeur de performance à ces deux situations en les mettant dans des ensembles (des parties) différents, les opérations de configuration qui font passer le système de i_1 à i_2 (ou le contraire) peuvent être considérées comme sans effet car atteignant le même degré de satisfaction de l'objectif du concepteur(même si des applications peuvent privilegier le taux de pertes au délai de bout-en-bout et vice-versa). Cet exemple montre aussi la necessité de définir les valeurs de performance, car dans ce cas précis, le système est confronté

2. Rappelons qu'une partition de l'ensemble I est une famille finie d'ensembles $I_1, \ldots, I_n \subset I$ telle que :
   - $\forall k \neq j, I_k \cap I_j = \emptyset$ *et* $I_k \neq \emptyset$, $I_j \neq \emptyset$ (deux à deux disjoints, aucun n'est vide)
   - $\bigcup_{k=1}^{k=n} I_k = I$ (la réunion est l'ensemble I).

à un choix entre deux paramètres de performance antagonistes et dont l'arbitrage appartient au concepteur suivant ses priorités.

Nous proposons la définition suivante pour limiter les cas d'ambigüité :

Définition 4.2.1.3 *Un critère de performance P_r est dit non ambigu s'il représente une fonction d'une partition de I vers un ensemble de valeurs V avec les propriétés suivantes :*

- *V est totalement ordonnée,*
- *P_r est une fonction de $P_r : I_1, \ldots, I_n \rightarrow V$*
- *$\forall k \neq j, P_r(I_k) \neq P_r(I_j)$*

La première propriété assure que pour toutes informations $i_1 \in I_1$ et $i_2 \in I_2$ qui remontent de l'équipement, il est possible de les comparer en passant, par $P_r(I_1)$ et $P_r(I_2)$, les parties qui les contiennent. Les autres propriétés permettent de lever toute ambiguïté sur la *prévalence* d'une situation sur autre ou bien leur équivalence. En d'autres termes, pour que le problème d'auto-configuration soit efficace, il faudra que le système sache s'il est préférable qu'il soit dans une situation plus que dans une autre. Nous montrerons, par la suite, un autre intérêt au fait de lever l'ambiguïté sur le critère de performance.

Dans un environnement réseau, il n'est pas rare d'avoir plusieurs critères de performance en même temps comme dans l'exemple précédent avec $i_1 = (10\%, 5ms) \in I_1$ et $i_2 = (5\%, 10ms) \in I_2$. Améliorer le taux de pertes constitue un critère de performance aussi bien que celui du délai de bout-en-bout. Considérons P_{r_1} et P_{r_2} representant respectivement les critères de performances consistant à minimiser les taux de pertes et le délai de bout en bout. Nous avons alors $P_{r_1}(I_1) < P_{r_1}(I_2)$ alors que $P_{r_2}(I_1) > P_{r_2}(I_2)$. En des termes simples, l'evaluation des deux situations suivant les deux critères arrive à une situation où le système ne peut pas decider de la valeur d'une transition d'une situation i_1 vers i_2. Plus généralement, si nous considérons un problème d'adaptation avec un ensemble de critères de performance, $P_{r_1}, \ldots, P_{r_n}$, le problème de l'auto-adaptation est non décidable s'il n'existe pas une fonction de performance P_r non ambigüe déduite de l'ensemble $\{P_{r_1}, \ldots, P_{r_n}\}$. En effet, s'il existe des cas où le système ne peut pas décider si une situation est souhaitable par rapport à une autre, tout le processus est bloqué sur cette transition sans décision. L'automatisation étant incomplète on obtient un graphe non connexe de décisions.

Comme corollaire, nous énonçons la proposition suivante :

Définition 4.2.1.4 *Un problème d'auto-configuration d'une ressource est décidable si l'espace des critères* $\{P_{r_1}, \ldots, P_{r_n}\}$ *de performance est non ambigu.*

Un problème d'auto-adaptation, avec un ensemble de critère $\{P_{r_1}, \ldots, P_{r_n}\}$ *est ambigu s'il existe deux informations* $i_1 \in I_1$ *et* $i_1 \in I_2$ *et, deux entiers* $k, j \in [1 \ldots n]$ *tels que :*

1. $k \neq j$
2. $P_{r_k}(I_1) < P_{r_k}(I_2)$ *alors que* $P_{r_j}(I_1) > P_{r_j}(I_2)$.

Une manière de lever l'ambigüité dans le cas d'un système avec plusieurs critères de performance est de mettre des poids sur les critères de performance de manière à rendre déterministe leur ordre d'importance. Si I_1 et I_2 sont comparés par rapport aux critère de performances $P_{r_1} \ldots P_{r_n}$ ayant des poids respectifs $p_1 < \ldots < p_n$: nous avons alors $I_1 > I_2$ ssi il existe $0 < k \leq n$ tel que $\forall j \geq k \;\; P_{r_j}(I_1) > P_{r_j}(I_2)$. Autrement dit, à partir d'un certain indice, les critères de performances de poids supérieurs donnent le même ordre pour deux éléments I_1 et I_2. Il existe aussi le cas particulier où les paramètres de performance sont mutuellement indépendants : l'optimisation de l'un laisse les autres intacts. Ce cas idéal étant assez rare, avant de mettre en place un processus d'auto-adaptation, il est nécessaire de faire des choix en présence de critères de performances multiples.

Pour résumer cette partie, nous dirons qu'il est nécessaire pour automatiser la configuration des équipements dans un objectif d'optimisation, de déterminer un critère de performance sans ambigüité qui permette au système de savoir de manière objective et automatique les objectifs de l'optimisation. Nous rappelons aussi que dans les cas où le problème de l'auto-adaptation est indécidable, aucune automatisation n'est possible et donc l'auto-adaptation efficace est irréalisable. Le but de la démonstration est de montrer une première famille de tâches qui incombe au concepteur des systèmes autonomes, qui est difficilement automatisable et qui peut avoir un impact fort sur le résultat final de l'automatisation de l'adaptation des équipements. La tâche de l'auto-adaptation commence donc en aval de ce travail.

4.2.2 Formalisation du problème

La problématique de l'auto-adaptation peut être posée également en terme de paradigmes *états/actions* ou *évènements/actions*. Pour chaque *état* donné, il s'agit de trouver un ensemble d'actions qui permettent de mettre le système dans le meilleur état souhaitable qu'on puisse espérer atteindre (suivant un critère de performance bien déterminé). En respectant la notation de la section 4.2.1, nous considérons l'ensemble des états comme une partition de I. Un état ou un événement [3] est donc un ensemble d'éléments de I décrivant la même situation (selon le choix du concepteur).

Le regroupement des situations $i \in I$ dans des ensembles décrivant les états est un choix qui permet de ne pas faire exploser l'ensemble des états comme dit précédemment. Ici, également, il y a un compromis à trouver entre un ensemble d'états très grand (donc un découpage fin de I) qui permettrait un *apprentissage* automatique plus fin mais moins rapide/réactif et plus intrusif sur le fonctionnement de la ressource gérée, et un regroupement des ensembles en un petit nombre d'états permettant de faire un apprentissage *peu* riche mais très rapide afin de trouver la fonction de configuration *optimale*. Nous proposons la définition suivante :

Définition 4.2.2.1 *Un état ε est un sous ensemble de I tel que l'ensemble des états Σ est une partition de I.*

Les actions de configuration, elles, sont à déterminer à l'avance, en particulier, en fonction du critère d'optimalité. Elles représentent les opérations de configuration qu'il est possible d'effectuer sur les équipements. Il est nécessaire de faire à l'avance des tests sur le système pour déterminer l'ensemble des actions qui ont une influence sur le critère de performance. Cette étape est primordiale pour que le système ait a sa disposition au moins un certain nombre d'actions susceptibles d'avoir une influence sur l'état du système. En respectant toujours la notation de la section 4.2.1, les actions sont des éléments de Ω^* qui est l'ensemble des puissances k^{eme} de Ω.

Définition 4.2.2.2 *L'ensemble des actions Λ est défini par :*

$$\Lambda = \bigcup_{k=1}^{\infty} \Omega^n = \Omega^*$$

3. Nous utiliserons par la suite indifféremment les termes événement et état

En notant par Σ l'ensemble des états nous pouvons reposer le problème de l'auto-adaptation comme suit :

Définition 4.2.2.3 *Le problème de l'auto-adaptation consiste à trouver la fonction h qui fait correspondre à chaque état ε, un ensemble fini d'actions $\{a_1,\ldots,a_k\}$ qui optimise la configuration suivant un paramètre de performance P_r, dans un environnement E_v.*

Cependant, cette définition cache un certain abus de langage à propos de la notion d'optimisation. En effet, pour parler d'optimisation, il faut avoir un état considéré comme optimal et qu'il faut à tout prix atteindre pour considérer le système comme optimal. Tenter de construire un système avec une telle définition de l'optimalité semble peu réaliste dans un environnement réseaux à moins de faire des choix très forts sur la liberté du système d'agir dans certains cas. Pour illustrer cette irréalisme, nous considérons le cas d'un routeur qui doit s'adapter pour optimiser la quantité de flux en sortie (figure 4.2).

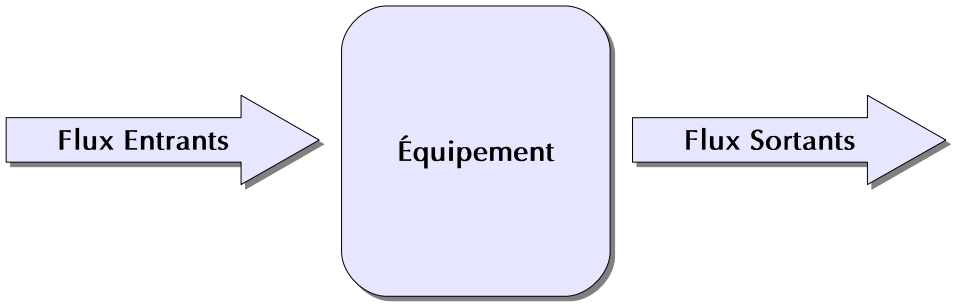

FIGURE 4.2 – Cas de maximisation des flux en sortie

Nous considérons ici l'optimisation au sens d'améliorer au mieux le système et de limiter les temps d'indisponibilité, les rejets au niveau du cœur du réseau, par exemple.

Maintenant que le problème est posé en ces termes, nous allons dans la section suivante présenter brièvement les deux méthodes qui sont souvent citées comme pouvant apporter une solution au problème de l'auto-adaptation.

4.2.3 Outils de réalisation de l'auto-adaptation

Pour réaliser l'auto-adaptation il y a deux grands domaines, qui sont pressentis comme outils : l'ingénierie et l'apprentissage automatique [Sterritt & Bustard 2003].

L'ingénierie est la méthode mise en avant par IBM. Il s'agit de la génération et du déploiement automatique de codes dans les systèmes pour les alimenter [Bradbury *et al.* 2004], [Li 2007]. Cette approche s'inspire très largement des systèmes logiciels adaptatifs du monde du génie logiciel [Bradbury *et al.* 2004], [Oreizy *et al.* 1999]. Le système est automatiquement approvisionné après un travail de test, d'affinement des méthodes en amont par des ingénieurs. Cette méthode a comme principaux avantages de disposer d'un ensemble de cadres théoriques du domaine du génie logiciel et de permettre aux administrateurs d'avoir une vision et un contrôle sur l'ensemble des procédures et actions de configuration sur les équipements. De plus, les contraintes présentées au paragraphe précédent sur les critères de performance sont sans importance puisque les concepteurs/administrateurs auront déjà réalisé de nombreux choix avant l'alimentation du système.

Cependant, dans cette vision, l'exploration de l'espace des configurations possibles pour en déterminer les meilleures reste assez limitée. Ceci, à cause du fait que les tests en amont réalisés par les administrateurs sont souvent limités à un sous-ensemble de configurations et que les tests sont "*manuels*" ou semi-automatiques. Ceci ne garantit pas une exploration exhaustive des situations possibles qui sont susceptible d'être rencontrées durant le fonctionnement de l'équipement. Or, sans cette exploration exhaustive, il est difficile de parler d'optimalité. Aussi, s'il y a une évolution dans l'environnement, le système doit être ré-alimenté pour prendre en compte cette nouvelle situation, ce qui dans une certaine mesure limite l'automatisation.

La méthode d'apprentissage, elle, consiste à laisser le système découvrir son environnement et améliorer sa configuration pour s'adapter de manière automatique. Les tests sont effectués sur l'environnement de manière automatique pour explorer les possibilités intéressantes en les affinant au fur et à mesure que l'historique s'enrichit. Dans les sections suivantes, nous allons présenter le domaine de l'apprentissage et proposer une méthode hybride qui permet de réaliser l'auto-adaptation.

4.3 Présentation de l'apprentissage

4.3.1 Définition

L'apprentissage est un paradigme qui est étudié dans plusieurs domaines scientifiques : l'intelligence artificiel, les théories de la probabilité et des statistiques, la théorie de l'information, l'optimisation numérique, la théorie de la complexité, la psychologie (du développement, cognitive), la neurobiologie, la linguistique, ... Elle couvre aussi diverses processus qu'il n'est pas toujours facile de définir de manière claire.

Cependant, le terme apprentissage symbolique est souvent utilisé pour décrire l'approche issue de l'intelligence artificielle qui se réfère aux méthodes qui représentent les connaissances apprises sous une forme déclarative, symbolique, par opposition aux approches statistiques ou aux méthodes de réseaux de neurones [Mooney 2004].

De manière générale, l'apprentissage artificiel est défini comme *l'étude des systèmes qui effectuent des changements pour améliorer leurs performances sur une tâche avec l'expérience* [Langley 1995, Nilsson 1996, Cornuéjols *et al.* 2003]. Les tâches en question peuvent être la reconnaissance, le diagnostic, le contrôle de robot, la prédiction, etc. Les changements incluent l'acquisition de nouvelles connaissances déclaratives, le développement de moteur et d'aptitudes cognitives à travers l'instruction ou la pratique, l'organisation de nouvelles connaissances plus générales, la représentation efficace des connaissances et la découverte de nouveaux faits et théories à travers l'observation et l'expérimentation [Carbonell *et al.* 1983].

Il y a trois grands domaines de recherche en apprentissage [4] :

- La recherche orientée tâche : il s'agit de l'apprentissage du développement et de l'analyse de systèmes capables d'améliorer leurs performances, dans un ensemble prédéterminé de tâches.
- La simulation cognitive : il s'agit de l'investigation et de la simulation par ordinateur du processus d'apprentissage de l'être humain.
- L'analyse théorique : l'exploration théorique des espaces possibles de méthodes d'apprentissage et d'algorithmes indépendants du domaine d'applica-

4. Nous utiliserons les termes apprentissage, apprentissage automatique et apprentissage artificiel indifféremment

tion.

Pour trouver le domaine le mieux adapté à la problématique de l'auto-adaptation, nous allons utiliser la définition proposée par Mitchell [Mitchell 1997] :

Définition 4.3.1.1 *Un programme est dit apprenant si, à partir d'une expérience Exp en respect d'une classe de tâche T et d'une mesure de performance P, sa tâche dans T mesurée avec P, s'améliore avec Exp.*

De manière générale, pour avoir une bonne définition d'un problème d'apprentissage automatique, il est nécessaire d'identifier trois caractéristiques : la classe de tâches à laquelle appartient le processus d'apprentissage, la métrique de performance à améliorer et une source de l'expérience.

Cette décomposition permet de faire une étude des méthodes d'apprentissage intéressantes en se focalisant sur les caractéristiques de la tâche T et de la forme des expériences. Dans la section 4.3.2, nous allons présenter plusieurs classifications des méthodes d'apprentissage suivant plusieurs critères.

4.3.2 Classification des méthodes d'apprentissage

Les domaines de l'apprentissage peuvent être classés suivant plusieurs dimensions aboutissant à un très grand nombre de types d'apprentissage [Campbell *et al.* 2005].

Suivant la manière d'améliorer l'activité nous avons deux grands types d'apprentissage : l'acquisition de nouvelles connaissances et le développement d'aptitudes. Dans le cas de l'acquisition de la connaissance, l'objectif de l'apprentissage est l'obtention de nouvelles connaissances sur la description et les modèles des systèmes et leurs comportements en utilisant différentes représentations. Ce type d'activité peut être définie comme l'apprentissage de nouvelles informations symboliques couplées avec la possibilité de les appliquer d'une manière efficace. Le domaine du développement d'aptitude exprime le développement graduel d'une aptitude par la pratique. Ce type d'apprentissage nécessite plusieurs répétitions de la pratique. Ce domaine se rapproche des systèmes de contrôle adaptatifs.

Une autre dimension de classification est la disponibilité ou non d'un label (réponse attendue) pour un échantillon d'*exemples*. Nous avons l'apprentissage supervisé, semi-supervisé et non supervisé. Lorsqu'il est fourni au processus d'apprentissage un ensemble de données de la forme (x_i, y_i) qui expriment pour chaque entrée

la vraie réponse que devrait trouver le processus, nous sommes dans le cas de l'apprentissage supervisé. Dans ce cas, il y a un "*oracle*" qui fournit au processus les réponses y_i à l'apparition des entrées x_i. Ici, l'apprentissage doit apprendre une hypothèse h telle qu'au minimum pour toutes les entrées x_i du problème appartenant à l'échantillon, $h(x_i) = y_i$, y_i étant la valeur donnée par l'*oracle*. Le cas de l'apprentissage semi-supervisé correspond à la mise à disposition seulement les réponses pour une partie des exemples de l'échantillon. L'apprentissage non supervisé n'a aucune information sur la fonction à apprendre et cherche des similarités, des régularités pour organiser les entrées.

Une autre classification du domaine de l'apprentissage consiste à séparer les méthodes suivant la stratégie sous-jacente d'apprentissage. Les stratégies d'apprentissage se différencient suivant la quantité d'inférences que l'apprenant (le système) peut faire sur les informations fournies. Il y a deux extrêmes : un cas ou l'apprenant ne fait aucune inférence et le cas ou le système fait un nombre assez substantiel d'inférences. Nous avons plusieurs catégories suivant cette quantité :

- Apprentissage par instruction : apprendre à partir d'une source qui sert d'instructeur et qui fournit des informations au système qui les représente sous une forme utilisable en interne
- Apprentissage par analogie : acquisition de nouveaux faits ou aptitudes par transformation de la connaissance déjà existante qui porte une certaine similarité avec le concept ou aptitude appris. Cette nouvelle connaissance sera utilisée pour de nouvelles situations.
- Apprentissage à partir d'exemples : apprendre à partir d'un ensemble d'exemples et de contre exemples, un concept général qui décrit l'ensemble des exemples et aucun des contre exemples. Il est aussi possible que seuls les exemples positifs soient disponibles. C'est un cas particulier de l'apprentissage inductif(voir section 5.2.1).
- Apprentissage à partir d'observations et de découvertes : C'est de l'apprentissage non supervisé. Le système ne dispose d'aucun ensemble d'exemples décrivant un concept particulier et n'a pas d'oracle pour trouver des réponses. Le système classifie les observations pour sortir des concepts intéressants

Les systèmes d'apprentissage peuvent, également, être classés suivant la représentation de la connaissance acquise qui peut prendre plusieurs formes telles que

par exemple :

- Arbre de décisions : Ce sont des systèmes de classification d'objets en utilisant la structure d'arbre.
- Grammaires formelles : Il s'agit d'apprendre à reconnaitre un langage particulier, les grammaires sont induites de séquences d'expressions dans le langage.
- Règles : Une règle est de la forme condition-action. L'acquisition des règles peut se faire par création de nouvelles règles, de généralisation de règles, de spécialisation ou de composition de règles.
- Logique formelle et formalismes associés : Cette représentation se base sur les expressions de la logique formelle avec comme objets de base des propositions, des prédicats, des variables, des expressions logiques imbriquées, ...
- Graphes et réseaux : C'est l'expression des expressions logiques sous une forme de graphes et de réseaux. Cette représentation permet surtout de profiter des résultats des transformations sur la théorie des graphes.

L'apprentissage peut être aussi réalisé à froid (par opposition à l'apprentissage en ligne) lorsque les données du problème sont disponibles au début du processus.

Ces quelques critères montrent la diversité des types d'apprentissage qui existent et la difficulté d'être exhaustif sur un tel domaine. De plus, il n'est pas toujours simple de classifier un problème tel que l'auto-adaptation dans une de ces classes d'apprentissage. Cependant, nous avons un certain nombres de propriétés du problème qui sont transverses à plusieurs techniques :

- Incrémentalité : Une technique d'apprentissage pour l'auto-adaptation doit avoir le caractère incrémental du fait même de la nature des informations sur le réseau. Comme nous l'avons montré sur la figure 4.1, les informations sont disponibles avec les changements qui s'opèrent au niveau de l'environnement et au niveau interne.
- Découverte de nouvelles connaissances : La découvertes de nouvelles connaissances pour s'adapter est nécessaire pour que le système auto-adaptatif puisse construire l'environnement au fur et à mesure qu'il rencontre les différentes situations
- Développement d'aptitude : Ceci montre la capacité du système d'améliorer sa tâche en construisant des aptitudes à partir de l'ensemble d'opérations disponibles. Le développement de ces aptitude va permettre au système d'améliorer

sa manière de gérer les situations jamais rencontrées auparavant.

– Gestion des données incomplètes : Une méthode d'apprentissage pour l'auto-adaptation devrait pouvoir prendre en compte une incomplétude des données et un environnement fortement dynamique. Dans l'environnement réseaux en fonction du type de trafic qu'il y a sur le réseau, les données peuvent fournir des données qui ne permettent qu'une adaptation efficace sur une partie des situations du système.

Au vu de ces critères nous pouvons définir l'apprentissage pour l'auto-adaptation comme suit :

Définition 4.3.2.1 *L'apprentissage pour l'auto-adaptation consiste à apprendre pour chaque état* $\varepsilon \in \Sigma$ *un ensemble d'actions* $\{\omega_1, \ldots, \omega_n\}$ *qui permettent de l'améliorer suivant un groupe de critères de performance* $\{P_{r_1}, \ldots, P_{r_m}\}$*, dans un environnement* E_v*.*

Par rapport à cette définition l'apprentissage symbolique offre beaucoup plus de souplesse et de possibilités pour cette problématique permettant de prendre en compte le plus de variables possibles sous une représentation encore compréhensible par l'homme. Les explications peuvent être comprises alors par l'administrateur qui pourra voir d'un d'œil bienveillant les adaptations réalisées par le système d'une manière autonome.

Défini comme tel, l'apprentissage pour l'auto-adaptation ressemble à beaucoup d'égards à la problématique de l'apprentissage par renforcement. Ils ont des objectifs très proches, et il a la propriété de l'incrémentalité et de l'apprentissage de couples état-actions. Il a, cependant, des contraintes qui sont difficiles à respecter dans un environnement réseau. Pour détailler ces ressemblances et ces différences, nous allons présenter l'apprentissage par renforcement à travers sa problématique, ses contraintes afin de montrer jusqu'à quel point ce type d'apprentissage peut permettre de réaliser l'auto-adaptation dans les réseaux autonomes.

4.4 Apprentissage par renforcement

4.4.1 Principe

L'apprentissage par renforcement a comme objectif de permettre à un apprenant/agent qui capture des informations dans un environnement et agit sur ce dernier de choisir un ensemble d'actions optimales pour atteindre un but.

Autrement dit, l'apprentissage par renforcement est le moyen d'apprendre à faire correspondre les situations de l'agent/apprenti dans son environnement (ses états) aux actions qu'il a la possibilité de faire de sorte à maximiser une récompense numérique lui indiquant le degré de pertinence de ses actions [Sutton & Barto 1998, Mitchell 1997]. L'apprenti n'est pas informé des actions à réaliser mais découvre les actions les plus intéressantes par rapport à la récompense qu'il reçoit de l'environnement en les essayant au cours de son fonctionnement. Suivant un état donné, les actions n'influent pas seulement sur la récompense directe mais aussi sur les récompenses liées aux états suivants et ainsi l'ensemble des séquences de récompenses.

Les deux caractéristiques de l'apprentissage par renforcement apparaissent dès lors : la recherche essai-erreur et la récompense différée. Il inclut donc une certaine liberté accordée à l'agent sur la possibilité de tester des actions sur certains états pour découvrir la correspondance, mais il considère également que la récompense suite à l'action qu'il vient de faire n'est pas seulement due à l'action elle seule mais à toutes les séquences d'actions qui ont été effectuées dans le passé et celles du futur.

Vis-à-vis du problème, il y a trois considérations très importantes à prendre en compte : l'agent doit avoir la capacité de capturer des informations de son environnement pour en avoir l'état, il doit avoir aussi la possibilité d'effectuer des actions qui affectent l'environnement et surtout il doit avoir un objectif relatif aux états du système. Cette dernière considération détermine la fin de l'apprentissage. La figure 4.3 modélise la problématique de l'apprentissage par renforcement.

L'agent interagit avec l'environnement décrit par un ensemble d'états possibles Σ. Il peut réaliser n'importe quelle action dans un ensemble fini d'actions Λ. A chaque fois qu'il fait une action a_t dans un état ε_t, il reçoit une valeur réelle ρ_t en guise de récompense indiquant la valeur immédiate de la transition du couple état-action. Ensuite, cela va produire une séquence d'états ε_i actions a_i et de récompenses

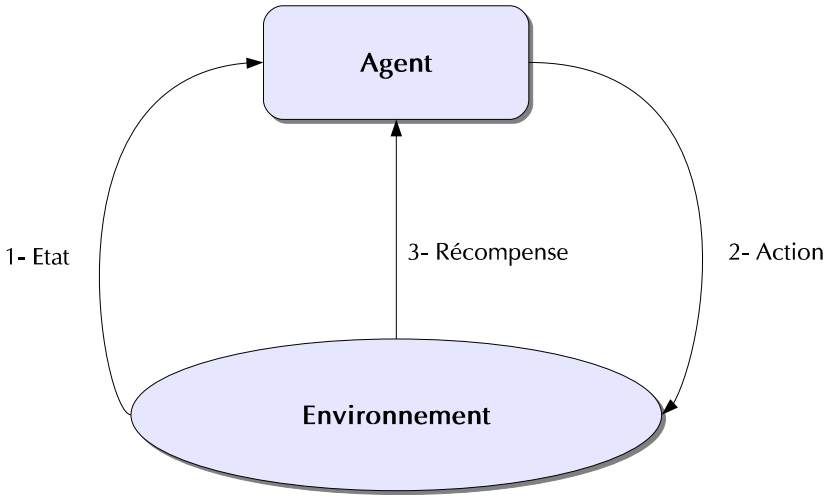

FIGURE 4.3 – Fonctionnement de l'apprentissage par renforcement

immédiates ρ_i comme le montre la figure 4.3. Le rôle de l'agent est d'apprendre une politique $\pi : \Sigma \rightarrow \Lambda$ qui maximise la somme attendue des récompenses avec les récompenses futures décomptées de manière exponentielle suivant leur éloignement dans la séquence.

L'apprentissage par renforcement se différencie des autres types d'apprentissage de fonctions classiques de par plusieurs aspects [Mitchell 1997]. Ainsi, dans les autres types d'apprentissage de fonction cible, il y a un ensemble d'exemples de la forme $(x,\ h(x))$ qui est fourni pour l'apprentissage alors que dans le cas de l'apprentissage par renforcement les exemples ne sont pas disponibles sur cette forme mais déduits de la séquence de récompenses.

4.4.2 Apprentissage de tâche

Un des défis les plus importants de l'apprentissage par renforcement est le compromis entre exploration et exploitation. L'exploration est l'exercice qui consiste à tester le maximum de couples état-action pour établir l'optimalité d'une action en particulier alors que l'exploitation consiste à utiliser les actions déjà connues comme étant performantes (sans assurance qu'elles soient optimales). Pour avoir davantage de récompense, l'agent apprenant doit préférer les actions qu'il a déjà testées dans le passé et retenues comme étant efficaces. Cependant, pour découvrir de telles actions, il doit tester des actions, qu'il n'a jamais sélectionnées auparavant.

L'agent doit, entre autre, exploiter ce qu'il sait déjà pour avoir plus de récompense et explorer la partie inconnue pour améliorer sa sélection d'actions dans le futur. Le dilemme vient du fait que ni l'exploitation ni l'exploration ne peuvent être exclusivement appliquées avec la garantie de ne pas tomber sur des cas d'échec du système. L'optimalité de la solution est obtenue avec une exploration exhaustive, et l'efficacité immédiate des actions est obtenue avec l'exploitation.

Sur une tâche stochastique, certains couples état-action doivent être testés plusieurs fois pour avoir une estimation correcte de la récompense attendue. Ceci complique davantage le problème (sachant que vraisemblablement, l'auto-adaptation dans les réseaux est dans ce cas de figure).

Le problème de l'apprentissage par renforcement est fortement lié au processus de décision markovien (PDM) [Mitchell 1997]. En effet, le modèle d'un processus de prise de décision de Markov est décrit par un quadruplet $< \Sigma, \Lambda, \rho, \delta >$ où Σ est l'ensemble des états possibles du système, Λ l'ensemble des actions possibles, ρ la fonction de récompense $\rho(\varepsilon, a)$ et δ une description de l'effet de chaque action sur chaque état (le modèle de l'environnement) qui représente les probabilités de transition. A chaque étape discrète du temps [5] t, le système se trouvant dans un état ε_t, l'agent choisit une action a_t et l'applique. Le système répond par la récompense immédiate $\rho_t = \rho(\varepsilon_t, a_t)$ et passe à l'état suivant $\varepsilon_{t+1} = \delta(\varepsilon_t, a_t)$. La particularité du processus de décision markovien est que $\delta(\varepsilon_t, a_t)$ et $\rho(\varepsilon_t, a_t)$ ne dépendent que de l'état courant et de l'action courante et pas des états et actions antérieures.

La tâche de l'agent est d'apprendre la politique $\pi : \Sigma \to \Lambda$ pour sélectionner à chaque état courant ε_t, l'action a_t à effectuer : $\pi(\varepsilon_t) = a_t$. La meilleure politique est celle qui maximise la valeur cumulée à chaque étape. Si nous considérons à partir d'un étape de temps t que l'agent a reçu les récompenses $\rho_t, \rho_{t+1}, \rho_{t+2}, \ldots$ la valeur cumulée de ses récompenses doit être exprimée avec les récompenses reçues. La valeur cumulée d'une politique π quelconque à partir d'un état quelconque est définie comme suit :

$$\begin{aligned} V^{\pi}(\varepsilon_t) \equiv & \quad \rho_t + \gamma\rho_{t+1} + \gamma^2\rho_{t+2} + \ldots \\ \equiv & \quad \sum_{i=0}^{\infty} \gamma^i \rho_{t+i} \end{aligned} \tag{4.2}$$

5. Ici le temps est utilisé au sens large du terme pouvant être un événement par exemple

Où les séquences de récompenses ρ_{t+1} sont générées en commençant par l'état ε_t en utilisant la politique π de manière répétée pour sélectionner les actions $(\pi(\varepsilon_{t+i}) = a_{t+i}))$. γ, appelé aussi taux d'escompte, est une constante telle que $0 \leq \gamma < 1$ qui détermine la valeur relative de l'état (par opposition à la récompense immédiate). Le taux d'escompte détermine la valeur présente des futures récompenses : une récompense reçue après k étapes de temps dans le futur à partir de l'étape présente n'est importante que γ^{k-1} fois que si elle était reçu à l'état présent.

L'utilisation de γ, au lieu d'une simple somme [6] $\sum_{i=0}^{\infty} \rho_{t+i}$ est qu'elle garantit que le cumul des récompenses est borné lorsque la suite $\{\rho_k\}_{k\geq 1}$ est bornée. Lorsque $\gamma < 1$, la somme infinie de l'équation 4.2 est bornée si la suite $\{\rho_k\}_{k\geq 1}$ est bornée. Si $\gamma = 0$, l'agent est *myope* ne cherchant qu'à maximiser la récompense immédiate. Si les actions de l'agent n'influencent que les récompenses immédiates, l'agent myope maximise alors la valeur de $V^{\pi}(\varepsilon_t)$ en maximisant les récompenses successives séparément. Cependant, plus γ est proche de 1 plus l'agent a une vision à long terme.

$V^{\pi}(\varepsilon_t)$ est appelé la récompense différée cumulative réalisée par la politique π à partir de l'état initial ε.

L'apprentissage d'un agent par renforcement consiste donc à apprendre une politique π qui maximise $V^{\pi}(\varepsilon)$ pour tout état ε. Notons π^* une telle politique. Nous avons :

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ V^{\pi}(\varepsilon), (\forall \varepsilon) \tag{4.3}$$

Par soucis de simplification la notation $V^{\pi^*}(\varepsilon)$ est noté $V^*(\varepsilon)$. $V^*(\varepsilon)$ donne la récompense différée cumulée maximum que l'agent obtient à partir de l'état ε en appliquant la politique π.

4.4.3 Eléments de l'apprentissage par renforcement

Dans un problème d'apprentissage par renforcement, il y a, entre autre (en plus de l'agent lui-même et de l'environnement), quatre éléments importants dans la formulation du problème : une politique, une fonction de récompense, une fonction de valeur, et, éventuellement un modèle de l'environnement.

6. Il est possible aussi de définir $V^{\pi}(\varepsilon_t)$ par une moyenne infinie des valeurs

La politique détermine le comportement de l'agent à un instant donné. Il s'agit d'une forme de *mapping* entre les états et les actions qui déterminent les choix de l'agent apprenant en fonction de chaque état où il se trouve. La politique peut être une simple fonction (à chaque état correspond au plus une action), une table de correspondances, ou bien être plus complexe comme dans le cas d'un processus de recherche.

La fonction de récompense, définit l'objectif dans un problème d'apprentissage par renforcement. Grossièrement, il effectue une correspondance entre un couple état/action de l'environnement et un nombre unique une récompense, qui indique implicitement la *désirabilité* de l'état sur lequel il se trouve par rapport à l'état précédent. Il informe ainsi l'agent des "*bons*" et mauvais événements. La contrainte, par rapport à cette fonction de récompense, est que l'agent n'a pas la possibilité de l'altérer mais constitue plutôt un moyen pour lui de modifier sa politique. Par exemple, une mauvaise action définie par la politique obtient une récompense "*négative*", ce qui peut donner lieu à une mise à jour de la politique. De plus, il faut choisir une valeur particulière de récompense qui indique à l'agent que le but est atteint. La fonction de récompense doit représenter très précisément ce que le concepteur de l'agent attend de sa part en terme de réalisation de tâche.

La fonction de valeur, contrairement à la fonction de récompense qui mesure l'efficacité immédiate d'une action sur l'état de l'environnement, indique les actions qui seront bonnes à long terme. Cela permet d'avoir une certaine vision du long terme. Elle permet, entre autre, de détecter un cas où il est nécessaire d'accepter quelques échecs afin d'atteindre l'optimum. La valeur d'un état est le cumul total de l'ensemble des récompenses espérées en partant de l'état présent. Alors que la fonction de récompense indique la désirabilité immédiate d'un état, celui de la fonction de valeur indique la désirabilité sur le long terme.

Au niveau de l'apprentissage par renforcement, on cherche les actions qui maximisent la fonction de valeur d'un état plutôt que celle qui maximisent la récompense immédiate. Cependant, il est plus difficile de déterminer la fonction de valeur que la fonction de récompense qui, elle, est a *priori* donnée à l'agent apprenant.

Le dernier élément non obligatoire lié au problème de l'apprentissage par renforcement est un modèle de l'environnement. Le modèle représente une estimation de l'effet des actions sur les états. Cet élément n'est pas explicitement obligatoire car il

"*réduit*" l'apprentissage à la recherche d'optimum par programmation dynamique. Dans ce cas, les tâches de l'agent sont essentiellement de la planification en utilisant une estimation des valeurs données par le modèle avant même d'avoir expérimenté les actions.

4.4.4 La fonction Q

Une fonction d'évaluation que pourrait apprendre l'agent pour optimiser sa politique est V^*. Il va préférer un état ε_1 à un état ε_2 si $V^*(\varepsilon_1) > V^*(\varepsilon_2)$ car la récompense cumulée future de ε_1 sera plus grande que celle de ε_2.

L'action optimale dans un état ε est celle qui maximise la somme de la récompense immédiate $R(\varepsilon, a)$ plus la valeur de V^* de l'état suivant pondéré par γ :

$$\pi^*(\varepsilon) = \underset{a}{\operatorname{argmax}} \ [\rho(\varepsilon, a) + \gamma V^*(\delta(\varepsilon, a))] \tag{4.4}$$

Nous rappelons que $\delta(\varepsilon, a)$ représente la fonction de transition qui retourne l'état suivant en appliquant l'action a à l'état ε. L'agent peut déduire la politique optimale en apprenant V^* à condition qu'il ait une bonne connaissance de la fonction de récompense ρ et de la fonction de transition δ. La connaissance de ρ et δ lui permet d'utiliser l'équation 4.4 pour trouver les actions à réaliser à chaque étape. Lorsque l'agent n'a pas une connaissance parfaite de ces fonctions, l'apprentissage de V^* ne suffit pas pour trouver la politique optimale. Or, dans la réalité, on se trouve bien plus souvent dans ce cas de figure (sans connaissance parfaite de ρ et δ). La question est donc de savoir comment l'agent peut trouver la politique optimale en ignorant les fonctions ρ et δ et en n'ayant à sa disposition que les récompenses immédiates étalées dans le temps (après chaque étape).

La fonction Q est une des réponses proposées à cet effet. Elle représente la valeur entre crochets dans l'équation 4.4. En d'autre terme, elle est définie comme suit :

$$Q(\varepsilon, a) \equiv \rho(\varepsilon, a) + \gamma V^*(\delta(\varepsilon, a)) \tag{4.5}$$

$Q(\varepsilon, a)$ représente donc la quantité à maximiser dans l'équation 4.4 pour pouvoir choisir l'action optimale a dans un état ε. La réécriture de cette équation donne :

$$\pi^*(\varepsilon) = \underset{a}{\operatorname{argmax}} \ Q(\varepsilon, a) \tag{4.6}$$

Cette nouvelle réécriture permet de voir que l'agent peut choisir les actions optimales en apprenant seulement la fonction Q sans même avoir une connaissance explicite et parfaite de ρ et δ. En effet, les actions qui maximisent localement la valeur de Q vont maximiser la récompense cumulée. L'agent peut ainsi choisir les actions directement à partir de cette valeur sans avoir une vision explicite de leur conséquence. Cette propriété constitue une des forces de l'apprentissage par renforcement Q-Learning.

Le problème, maintenant, consiste à trouver une manière d'estimer la valeur de Q pour la récompense immédiate. Cette opération peut être faite en utilisant une méthode itérative grâce à la relation entre V^* et Q

$$V^*(\varepsilon) = \max_{a'} Q(\varepsilon, a') \tag{4.7}$$

Ce qui permet de réécrire l'équation 4.5 en :

$$Q(\varepsilon, a) \equiv r(\varepsilon, a) + \gamma \max_{a'} Q(\delta(\varepsilon, a), a') \tag{4.8}$$

Cette réécriture récursive donne une base pour un algorithme permettant d'approximer la fonction Q [WATKINS 1989, WATKINS & DAYAN 1992].

4.4.5 Algorithme Q-Learning

Le processus d'apprentissage Q-Learning est décrit par l'algorithme 1. L'agent représente l'estimation $\hat{Q}$ qu'il a de la fonction Q à travers une table, avec une entrée pour chaque couple état-action. Dans chaque case de cette table, il stocke la valeur courante estimée $\hat{Q}(\varepsilon, a)$ de la valeur $Q(\varepsilon, a)$.

La table est initialement à zéros (mais peut également être remplie de valeurs aléatoires). L'agent répète de manière continue l'observation de l'état courant ε, choisit une action a qu'il applique et observe ensuite la récompense $\rho = \rho(\varepsilon, a)$ et le nouvel état $\varepsilon' = \gamma(\varepsilon, a)$. Après chaque transition, l'agent met à jour la case de la table correspondant à $\hat{Q}(\varepsilon, a)$ avec la règle suivante :

$$\hat{Q}(\varepsilon, a) \leftarrow \rho + \gamma \max_{a'} \hat{Q}(\varepsilon', a') \tag{4.9}$$

L'estimation de l'agent $\hat{Q}$ converge en terme de limites vers Q si le système peut être modélisé comme un processus de décision markovien déterministe si la fonction

Algorithme 1 : Algorithme Q-Learning

début

  pour tous les *couples* (ε, a) **faire**

    $\hat{Q}(\varepsilon, a) \leftarrow 0$

  Observer l'état courant ε

  répéter

    Sélectionner une action a et l'exécuter

    Recevoir la récompense immédiate ρ

    Observer un nouvel état ε'

    Mettre à jour la case de la table $\hat{Q}(\varepsilon, a)$ comme suit :

$$\hat{Q}(\varepsilon, a) \leftarrow \rho + \gamma \max_{a'} \hat{Q}(\varepsilon', a')$$

  jusqu'à *jamais*

fin

de récompense ρ est bornée et, enfin si les actions sont choisies de telle sorte que tout couple état-action est visité un nombre infini de fois.

Nous rappelons qu'un processus de décision markovien est dit déterministe lorsque la fonction de transition est déterministe. A chaque couple (ε, a) est associé un seul état (la même action sur le même état produit le même résultat).

4.4.6 Contraintes et difficultés de l'apprentissage par renforcement

L'apprentissage par renforcement est bien adapté, dans le cas de processus markovien déterministe. L'algorithme Q-Learning, dans ce cas, converge à l'infini vers la politique optimale [WATKINS & DAYAN 1992]. L'algorithme Q-Learning, cependant, ne converge, dans le cas de processus markovien indéterminisme, que dans un certain nombre de conditions particulières [WATKINS & DAYAN 1992].

Or, dans le problème d'auto-adaptation purement réseaux, il est difficile de se prononcer sur le caractère markovien du processus surtout quand l'environnement est constitué de beaucoup d'éléments (réseaux) qui agissent de manière distribuée. Dans ces conditions, nous avons un environnement stochastique où on n'a pas forcé-

ment une relation claire et déterministe entre les actions et les changements observés sur l'environnement. En d'autres termes, pour un état ε donné et une action a, il n'est pas toujours possible de dire si l'action est la cause précise d'un changement observé dans l'environnement. On ne sait pas jusqu'à quelle durée dans le passé, les actions précédemment effectuées influent sur la transition observée à l'étape présente. Autrement dit, la longueur de la séquence est inconnue au cas où elle existe. De plus, le fait qu'un élément réseau agisse ne lui garantit pas que c'est son action qui a eu l'impact observé sur l'environnement. Il est possible qu'en amont le type de flux ait changé, par exemple, ou un nœud se soit occupé du problème... La première conséquence directe est la difficulté de créer une fonction de récompense objective, absente de la formalisation du problème de l'auto-adaptation, qui permette de récompenser une action de manière efficace. On pourrait imaginer s'en extraire en utiliser une fonction similaire à Q, mais elle ne garantit pas une convergence ni la correction des résultats pour tous les cas de figures. D'un autre coté, il est possible de choisir la fonction de récompense pour une problématique d'auto-adaptation bien particulière, mais il faudra la redéfinir à chaque fois que le contexte d'application de l'apprentissage par renforcement change. L'idéal serait d'avoir un type d'apprentissage qui rende abstrait cette fonction de récompense ou bien la découvre automatiquement.

Une seconde conséquence est qu'on n'a pas un état objectif clair, une fonction de récompense objective dans le cas de l'auto-adaptation. Dans l'apprentissage par renforcement, nous avons un état qui permet d'arrêter l'exploration pour en déduire la séquence d'actions à réaliser. Dans le cas de l'auto-adaptation, l'état idéal n'a pas forcement de sens car pour le même état de l'équipement et des états de l'environnement différent l'état idéal peut changer au cours du temps.

Le meilleur état qu'on peut atteindre dépend, aussi, de l'activité du réseau dans le temps. La notion de temps rend encore le problème plus compliqué puisque que cela correspondrait au niveau d'un apprentissage par renforcement à un environnement versatile, avec une période de stabilité plus ou moins variable. Enfin, lorsque les séquences d'actions à effectuer sont trop longues cela peut avoir une influence sur la réactivité du système. Toutes ces différences montrent qu'il ne semble très peu réaliste d'appliquer l'apprentissage par renforcement tel qu'il est défini.

L'apprentissage par renforcement implique un certain temps d'exploration des

états pour déterminer les meilleures actions. Dans le cas d'un processus markovien déterministe, une seule exploration d'un couple état-action suffit pour connaitre la valeur d'une action pour un état donné. Cependant, dans le cas des réseaux nous avons affaire à un processus stochastique où il est nécessaire de faire plusieurs tests pour avoir une probabilité représentative de l'efficacité d'une action sur un état donné. Dans le cas idéal de convergence, tous les couples état-actions sont exécutés de manière infinie. Cette considération constitue un problème assez important et ne peut être utilisée en réseau car il y a une contrainte de QoS qui fait que l'auto-adaptation doit se stabiliser le plus rapidement possible.

4.5 Conclusion

Dans ce chapitre, nous avons posé le problème de l'auto-adaptation en identifiant les éléments qui sont indispensables pour que le problème soit complet. En effet, comme nous l'avons montré, il faut la définition d'un critère de performance (P_r) pour que le système, durant l'automatisation, puisse identifier l'objet de son amélioration. Il faut aussi, identifier les paramètres importants de l'environnement (le nombres de voisins par exemple, la bande passante des liens sur chaque interface, ...), définir l'espace des paramètres qu'il faut prendre en compte durant l'auto-adaptation I, et les ressources qu'il faut adapter R en fonction du critère de performance P_r. Le système aura donc comme tâche de trouver pour chaque situation, définie par un ensemble de valeurs de paramètres de I et de E_v, les actions de configurations qu'il faut effectuer pour améliorer son état.

Après cette formalisation, nous avons fait un tour synthétique des différents types d'apprentissage qui existent et puis nous avons identifié un certain nombre de caractéristiques de l'auto-adaptation que prennent en charge le outils d'apprentissage tels que l'incrémentalité, l'acquisition de nouvelles connaissances sur l'environnement. Nous avons, par la suite fait souligner que, par sa conception et sa problématique, l'apprentissage par renforcement est la méthode d'apprentissage qui se rapproche le plus de la problématique de l'auto-adaptation. Cependant, cette méthode d'apprentissage exige un paramètre important qui est la fonction de récompense, chose qui n'est pas forcément disponible ni facile à trouver pour l'environnement réseaux. Après avoir présenté en détail cette méthode, nous avons montré quels

sont les problèmes liés à son utilisation sur la problématique de l'auto-adaptation dans les réseaux autonomes. Entre autre, la sensibilité de l'apprentissage au fait que l'environnement de l'auto-adaptation soit stochastique, la non disponibilité de la fonction de récompense ... Pour ne pas avoir à traiter ces limites et aboutir à une méthode plus complexe que l'apprentissage par renforcement sans avoir ses garanties, nous proposons dans ce cas d'apprendre à s'adapter au mieux, c'est à dire s'améliorer suivant un critère de performance jusqu'à ne plus pouvoir être dans un meilleur état. Cette nouvelle démarche est plus réaliste que celle qui consiste à vouloir atteindre un état optimal bien prédéterminé.

Cette approche est présentée dans le chapitre 5.

CHAPITRE 5

Module d'apprentissage pour l'auto-adaptation

Sommaire

5.1 Introduction

Nous avons montré, dans le chapitre 4, les difficultés et contraintes qui rendent l'application de l'apprentissage par renforcement comme solution générique au problème de l'auto-adaptation. Ces contraintes sont principalement :

- l'utilisation de la fonction de récompense qu'il est nécessaire de définir de manière rigoureuse et générale dans les environnement réseau ;
- le caractère stochastique de l'auto-adaptation avec la contrainte sur le nombre de tests que le système peut faire ;
- la contrainte pour l'auto-adaptation de s'orienter assez rapidement vers les actions efficaces au début.

Notre proposition est de faire une estimation de manière stochastique des probabilités de réussite des actions et d'utiliser la programmation logique inductive pour découvrir de nouvelle règles. Ici, l'utilisation de la programmation logique inductive va permettre d'apprendre à partir des couples état-actions des règles générales qui permettrons de découvrir de nouvelles règles dans l'espace non encore exploré. Cette orientation va permettre d'orienter le choix de l'apprentissage vers les couples qui sont les plus susceptibles d'être intéressants.

Le choix de la programmation logique inductive est motivé par le fait qu'elle fait partie des techniques d'apprentissages de règles. La forme des règles est très proche de celle des règles de configuration pour l'auto-adaptation. Le caractère inductif de la PLI aussi est très intéressante et permet d'enrichir la base de connaissances sur les couples état-actions.

Ensuite, pour accélérer le processus nous proposons des échanges de stratégies (couples état-actions) entre les équipements s'auto-adaptant, s'inscrivant dans une approche collaborative de l'auto-adaptation. Nous profitons aussi d'idées de l'apprentissage par renforcement (apprentissage de couples état-meilleure action) sans nous imposer toutes ses contraintes.

Nous allons d'abord, présenter la programmation logique inductive pour introduire les concepts qui seront par la suite utilisés.. Ensuite, nous détaillerons l'approche d'adaptation. Enfin, nous finirons par présenter l'aspect distribué de l'approche par échange de connaissances.

5.2 Programmation logique inductive

La programmation logique inductive (PLI) est un domaine de la recherche qui est à l'intersection de la programmation logique et de l'apprentissage inductif. Il a pour objectif de fournir un cadre et des algorithmes pour l'apprentissage inductif de programmes logiques à partir d'exemples.

5.2.1 Apprentissage inductif de concepts

L'induction est définie comme le type de raisonnement consistant à remonter, par une suite d'opérations cognitives, de données particulières (faits, expériences, énoncés) à des propositions plus générales, de cas particuliers à la loi qui les régit, des effets à la cause, des conséquences au principe, de l'expérience à la théorie. En apprentissage inductif, l'apprenant dispose d'un ensemble d'exemples à partir duquel il dérive des règles générales et une théorie sous-tendant les exemples.

Définition 5.2.1.1 *Soit un ensemble universel U d'objets (ou d'observations), un concept C peut être formalisé comme un sous ensemble de U : $C \in U$. Apprendre un concept revient à reconnaitre l'ensemble des objets de U qui appartiennent à C en partant d'une hypothèse générale qui décrit l'ensemble.*

En d'autre terme, l'apprentissage permet d'avoir une hypothèse décrivant un concept C de telle sorte qu'on puisse décider si la description d'un objet x de U satisfait ou non à la description de C. Dans ce cas, on dit que la description du concept C couvre celle de l'objet x.

Un fait peut être défini comme la description d'un objet particulier et une hypothèse comme la description en intention d'un concept à apprendre.

Un exemple e pour l'apprentissage d'un concept C est un fait avec un label. Le label est noté $\oplus$ si e est une instance de C, et $\ominus$ sinon (c'est-à-dire s'il n'est pas une instance de C). Notons E l'ensemble des exemples. Les éléments de E avec le label $\oplus$ forment l'ensemble des exemples positifs appelés exemples positifs E^+ et les exemples avec le label $\ominus$ forment l'ensemble des exemples négatifs E^- pour le concept C. Au cours d'un processus d'apprentissage inductif, le label d'un exemple n'est pas toujours explicitement énoncé, mais est plutôt implicite en fonction de son appartenance à E^+ ou E^-.

Nous pouvons, dès à présent, définir le concept d'apprentissage inductif de concept :

Définition 5.2.1.2 *Soit E un ensemble d'exemples positifs et négatifs pour un concept C, l'apprentissage inductif de concept consiste à trouver une hypothèse H telle que :*

- *Tout exemple positif $e \in E^+$ est couvert par H*
- *Aucun exemple négatif $e \in E^-$ n'est couvert par H*

Le test de couverture est fait à l'aide de la fonction $covers(H,e)$:

$$covers(H,e) \tag{5.1}$$

Cette fonction retourne vrai si e est couvert pas H et faux sinon. La fonction peut être étendue aux ensembles (pour éventuellement exprimer la couverture d'un ensemble d'exemples) :

$$covers(H,E) = \{e \in E | covers(H,e) = true\} \tag{5.2}$$

Cette fonction retourne le sous-ensemble de E couvert par H.

La problématique de l'apprentissage inductif de concept exige donc que l'hypothèse H trouvée couvre l'ensemble des exemples positifs et aucun des exemples négatifs. Dans ce cas, on dit que l'hypothèse H est consistante et complète.

Définition 5.2.1.3 *Soit H une hypothèse pour un concept à apprendre et E un ensemble d'exemples.*

1. *Une hypothèse H est complet en respect de l'ensemble d'exemples E si $covers(H,E^+) = E^+$, c'est-à-dire si H couvre l'ensemble des exemples positifs.*
2. *Une hypothèse H est consistant par rapport à l'ensemble E si $covers(H,E^-) = \emptyset$, c'est-à-dire que H ne couvre aucun exemple négatif.*

La figure 5.1 [Muggleton & Raedt 1994] représente l'ensemble des combinaisons possibles, les exemples positifs étant représentés par des signes + et les exemples négatifs par des signes −.

Etant donné un concept C et un ensemble d'exemples E, l'existence d'une hypothèse H qui soit consistant et complet il n'est pas toujours garantie. Si la contrainte

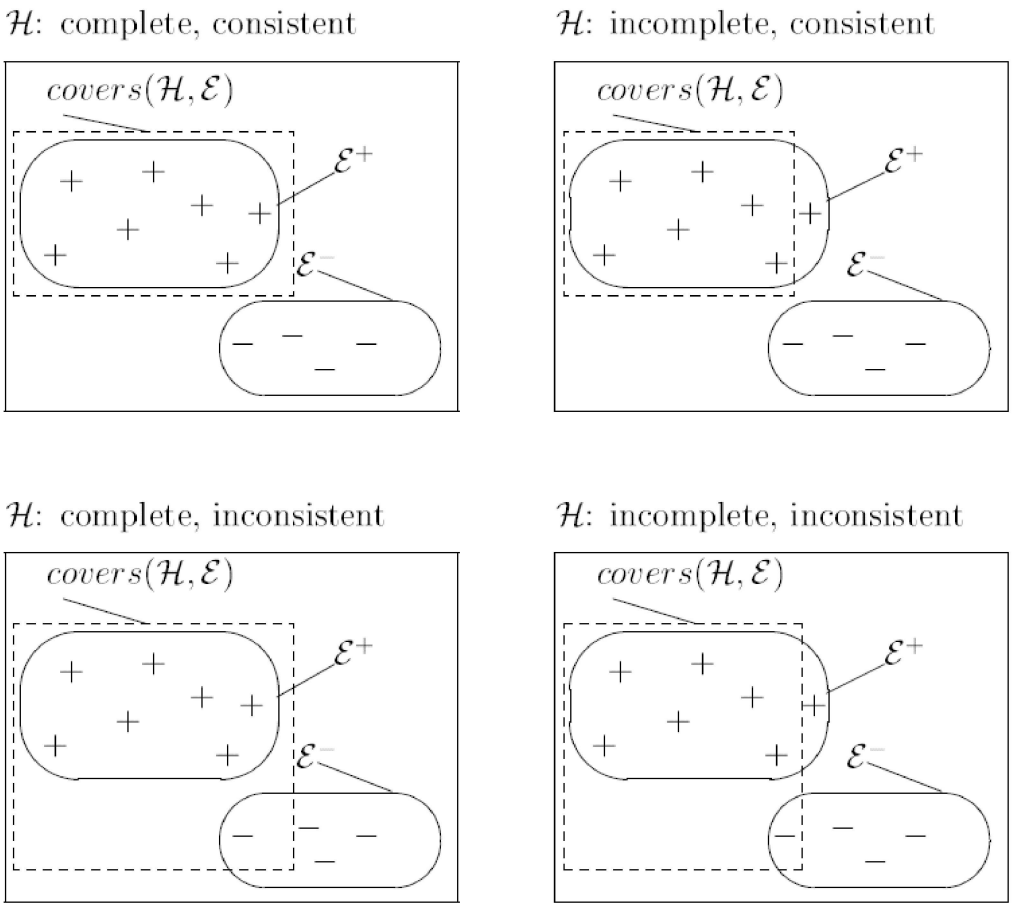

FIGURE 5.1 – Consistance et complétude d'une hypothèse

de la complétude est relâchée, on peut introduire un nouveau critère de précision indiquant le nombre d'exemples positifs couverts. Nous reviendrons sur cette notion.

5.2.2 Connaissance de base

L'apprentissage inductif de concept, comme nous l'avons dit, consiste à trouver une hypothèse décrivant le concept en question. Implicitement, cela correspond à une recherche dans l'espace de l'ensemble des hypothèses pouvant décrire le concept à apprendre. Si l'apprenant ne dispose pas de connaissance a priori, il fera son apprentissage exclusivement avec les exemples. Cependant, dans la plupart des problématiques d'apprentissage, il y a une connaissance disponible à *priori*, qui est appelée connaissance de base. L'utilisation de cette connaissance de base permet à l'apprenant d'exprimer la généralisation des exemples d'une manière plus concise par rapport à la nature du problème. La connaissance de base détermine également de manière indirecte l'espace de recherche des descriptions possibles du concept appelé : l'espace des hypothèses.

L'apprentissage inductif de concept avec connaissance de base peut, ainsi, être défini comme suit :

Définition 5.2.2.1 *Soit E un ensemble d'exemples, et une théorie du domaine B, que l'on nommera connaissance de base par la suite, l'apprentissage inductif de concept avec connaissance de base consiste à trouver une hypothèse H qui soit consistante et complète par rapport à E et B.*

Il semble nécessaire d'étendre la fonction *covers*, qui ne prenait que deux paramètres, pour prendre en compte B.

Les notions de complétude et de consistance sont étendues par la même occasion :

Définition 5.2.2.2 *Une hypothèse H est complète par rapport à une connaissance de base B et un ensemble d'exemples E si elle couvre tous les exemples positifs, c'est-à-dire si $covers(B,H,E^+) = E^+$.*

Une hypothèse H est consistante par rapport à une connaissance de base B et un ensemble d'exemples E si elle ne couvre aucun exemple négatif c'est-à-dire si $covers(B,H,E^-) = \emptyset$.

5.2.3 Programmation logique

La programmation logique prend ses racines dans la logique combinatoire. Dans cette partie, nous allons présenter la terminologie de ce concept pour mieux comprendre comment l'apprentissage inductif est appliqué pour donner un nouveau paradigme très intéressant pour l'apprentissage de règles.

Nous allons présenter quelques définitions de base de la logique du premier ordre en utilisant la syntaxe Prolog comme exemple.

Un *alphabet* du premier-ordre consiste en un ensemble de variables, de prédicats et de symboles de fonctions (constantes inclues).

Une *variable* est représentée par un caractère en majuscule suivi de minuscules combinés à des chiffres éventuellement.

Un *symbole* de fonction est un caractère en minuscule suivi par une suite de caractères combinés à des chiffres éventuellement.

Un *prédicat* (ou symbole de prédicat) est une suite de chiffres et de lettres commençant par une lettre en minuscule.

Un *terme* est une variable ou une fonction suivie d'un ensemble de termes entre parenthèses. Le nombre de termes entre parenthèses est appelé *arité*.

Exemple : si f, g, h sont des fonctions et X une variable, $f(g(X),h)$.

Une *constante* est une fonction d'*arité* 0. Un prédicat suivi d'un $n-uplet$ de termes entre parenthèses est appelé un atome ou formule atomique.

L et $\bar{L}$ sont appelés des littéraux si L est un atome. Dans ce cas, L est appelé un littéral positif et $\bar{L}$ un littéral négatif.

Une *clause* est une formule ayant la forme suivante :

$$\forall X_1, \forall X_2, \ldots, \forall X_n (L_1 \vee L_2 \vee \ldots L_m) \tag{5.3}$$

Où les L_i sont des littéraux et les X_j sont les variables qui peuvent apparaitre dans l'expression $L_1 \vee L_2 \vee \ldots L_m$

Une clause peut être ainsi représentée comme un ensemble fini et éventuellement vide de littéraux. L'ensemble $\{L_1, L_2, \ldots, L_i, L_{i+1}, \ldots\}$ représente la clause $(L_1 \vee L_2 \vee \ldots \overline{L_i} \vee \overline{L_{i+1}} \vee \ldots)$ qui peut être représentée rigoureusement par la formule :

$L_1, L_2, \ldots \leftarrow L_i, L_{i+1}, \ldots$

Où les virgules à gauche du signe $\leftarrow$ représentent des disjonctions et celles à droite des conjonctions. Un ensemble de clauses représente une *théorie clausale*. Les littéraux, les clauses et les théories clausales sont appelés des *formules bien formées*.

Définition 5.2.3.1 *Une clause de Horn est une clause ne contenant qu'au plus un littéral positif.*

Une ***clause définie*** *est une clause de Horn avec exactement un littéral positif.*

$T \leftarrow L_1, \ldots L_m$

Où $T, L_1, \ldots L_m$ sont des atomes.

Le littéral positif dans un *clause définie* est appelé la tête de la clause. La conjonction de littéraux négatifs est appelée le corps de la clause. Dans la terminologie de Prolog, une clause de Horn avec un corps vide est appelé un fait.

Définition 5.2.3.2 *La définition d'un prédicat est un ensemble de clauses avec le même symbole (et arité) dans la partie tête.*

5.2.4 Programmation logique inductive

Le fonctionnement de la programmation logique inductive peut maintenant être présenté de manière générale ainsi que les différentes implémentations existantes.

En programmation logique inductive, $\mathscr{E}$ est un ensemble de faits de telle sorte que les exemples positifs $\mathscr{E}^+$ sont considérés comme vrais et les exemples négatifs $\mathscr{E}^-$ comme faux. Il reste donc à redéfinir la notion de couverture au sens de la programmation logique.

Définition 5.2.4.1 *Etant donnés une connaissance de base $\mathscr{B}$, une hypothèse $\mathscr{H}$ et un ensemble d'exemples $\mathscr{E}$. L'hypothèses $\mathscr{H}$ couvre un exemple $e \in \mathscr{E}$ si $\mathscr{B} \cup \mathscr{H} \models e$,*

$$covers(\mathscr{B}, \mathscr{H}, e) = vrai \quad si \quad \mathscr{B} \cup \mathscr{H} \models e \tag{5.4}$$

Par conséquent la fonction $covers(\mathscr{B}, \mathscr{H}, \mathscr{E})$ peut être définie comme suit :

$$covers(\mathscr{B}, \mathscr{H}, \mathscr{E}) = \{e \in \mathscr{E} \| \mathscr{B} \cup \mathscr{H} \models e\} \tag{5.5}$$

La complétude et la consistance signifient respectivement $covers(\mathscr{B},\mathscr{H},\mathscr{E}^+) = \mathscr{E}^+$ et $covers(\mathscr{B},\mathscr{H},\mathscr{E}^-) = \emptyset$.

Sémantiquement, cette notion de couverture en PLI a une signification en relation avec la logique. En d'autres termes, un exemple e est couvert par l'hypothèse $\mathscr{H}$, étant donné un ensemble de connaissances de base $\mathscr{H}$, si e est une conséquence logique de $\mathscr{B} \cup \mathscr{H}$. Cette dernière notion de conséquence logique est notée à l'aide de l'expression : $\mathscr{B} \cup \mathscr{H} \models e$.

Une définition plus formelle, précisément il s'agit ici d'apprentissage par dérivation(Learning by entailment), est donc :

Etant donné :

- Un ensemble d'apprentissage $\mathscr{E}$, constitué de faits vrais ($\mathscr{E}^+$) et faux ($\mathscr{E}^-$), d'un prédicat p inconnu ;
- Une connaissance de base $\mathscr{B}$ définissant un ensemble de prédicats q_i (autre que p donc $\mathscr{B} \not\models E^+$) qui peuvent être utilisés dans la définition de p.

La PLI consiste à trouver une définition $\mathscr{H}$ pour p de sorte que :

- $\forall e \in \mathscr{E}^+$, $\mathscr{B} \cup \mathscr{H} \models e : \mathscr{H}$ est complet par rapport à $\mathscr{B}$ et $\mathscr{E}$;
- $\forall e \in \mathscr{E}^-, \mathscr{B} \cup \mathscr{H} \not\models e \ : \mathscr{H}$ est consistant.

Rappelons que l'induction consiste à partir du particulier pour arriver à une hypothèse générale. Il existe deux approches pour explorer l'espace des hypothèses :

- Descendante (ou Top-down) qui consiste à partir d'une hypothèse la plus générale (qui va donc être complète) et la spécialiser jusqu'à ce qu'elle soit consistante ;
- Ascendante (ou Bottom-up) qui consiste à partir d'une hypothèse la plus spécifique qui soit consistante et la généraliser jusqu'à ce qu'elle soit complète.

Etant donné un prédicat p, le programme le plus général (couvrant l'ensemble des entrées), qui est certainement complet (couvre l'ensemble des exemples positifs) est le programme avec un corps vide (un fait en Prolog) `[p:-]`. Le programme le plus spécifique contenant p (qui est certainement consistant) est celui renvoyant faux (F) pour toutes les entrées `[p:-F]`. Les deux méthodes citées plus haut peuvent être réalisées respectivement en utilisant des opérations de généralisation et de spécialisation :

- Commencer par `[p:-]` et spécialiser jusqu'à ce que le programme devienne consistant ;

– Partir de `[p:-F]` et généraliser jusqu'à ce que le programme devienne complet.

Une clause C peut être généralisée principalement de trois manières :

– Remplacement de certains termes dans le *programme clausale* par des variables. Exemple `p(x) :- g(y,z), h(m)` peut être généralisé en `p(x) :- g(y,z), h(M)`. Ce procédé est l'inverse de l'opération de substitution ;

– Suppression de littéraux dans le corps de la clause ;

– Ajout d'une clause au programme

Par analogie, pour spécialiser le programme clausale C il y a trois possibilités :

– Remplacement de variables par des termes (substitution) ;

– Ajout de littéraux dans le corps d'une clause ;

– Retrait d'une clause au programme.

L'ensemble des clauses peut ainsi être partiellement ordonné avec la relation de spécialisation. De manière générale, une clause c_1 est plus spécifique que c_2 si $c_2 \models c_1$ (toute interprétation qui rend c_2 vraie rendra c_1 automatiquement vrai). Un cas particulier où c_1 est plus spécial que c_2 est celui où les termes dans le corps de la clause c_1 sont un sous ensemble des termes dans le corps de c_2. Cette relation d'ordre peut être utilisée pour organiser l'ensemble des clauses dans une structure de clauses partiellement ordonnée, appelée *graphe de raffinement*. Une clause c_1 est le successeur directe de c_2 dans ce graphe si et seulement si c_1 peut être obtenu en ajoutant un littéral dans le corps de la clause c_2. Le graphe de raffinement permet donc de déterminer les possibilités de raffinement en ajoutant des littéraux dans le corps d'une clause. Ces possibilités étant trop grandes pour ne pas dire infinies, dans les systèmes ILP de base on réduit l'espace de raffinement suivant certains critères :

– Restriction sur les littéraux de la connaissance de base utilisable pour la spécialisation. En d'autres termes, il s'agit d'énumérer les littéraux qui sont à utiliser lors de la recherche ;

– Les littéraux dont les arguments sont des sous ensembles des arguments du littéral en tête de clause ;

– Les littéraux qui apportent de nouvelles variables différentes de celles des littéraux de tête ;

– ...

Exemple : $H = \{vole(X) \leftarrow oiseau(X) \wedge migrateur(X); vole(X) \leftarrow avion(X)\}$

peut-être induite de la théorie du domaine
$B = \{nom(a,titi), oiseau(a), migrateur(a), avion(b),$
$nom(c,roussette), poule(c), oiseau(X) \leftarrow poule(X)\}$
et des exemples
$E^+ = \{vole(a), vole(b)\}$ et $E^- = \{vole(c)\}$.

Maintenant, nous pouvons présenter l'algorithme classique descendant de la programmation logique inductive dans lequel les clauses sont construites une à une.

Algorithme 2 : Algorithme générique top-down de la PLI

Données :

Entrées :

$\mathscr{E}_{cur} \leftarrow \mathscr{E}$;

$\mathscr{H} \leftarrow \emptyset$;

Sorties :

Hypothèse $\mathscr{H}$

```
1  début
2      répéter
3          Initialiser c ←  T :− ;
4          répéter
5              Trouver le meilleur raffinement c_meilleur ∈ p(c) ;
6              Assigner c ← c_meilleur ;
7          jusqu'à ce que critère de consistance soit satisfait ;
8          Ajouter c à ℋ pour avoir une nouvelle hypothèse ℋ' = ℋ ∪ c;
9          Enlever les exemples positifs de ℰ_cur couverts par c pour avoir un
           nouvel ensemble d'exemples ℰ'_cur = ℰ_cur − covers(ℬ, ℋ', ℰ+_cur);
10         Assigner ℰ_cur = ℰ'_cur ;
11         ℋ = ℋ'
12     jusqu'à ce que le critère de complétude soit satisfait ;
13 fin
```

Dans l'algorithme 2 le processus de construction des hypothèses se fait en fonction d'un ensemble courant d'exemples $\mathscr{E}_{cur}$ non encore explorés (initialisé à $\mathscr{E}_{cur}$, puis

mis à jour) et une hypothèse courante (contenant au début l'ensemble vide). L'algorithme est constitué de deux boucles "*répéter*" de couverture et de spécialisation. La boucle de couverture construit l'hypothèse alors que la boucle de spécialisation construit les clauses de l'hypothèse.

Les deux boucles sont contrôlées par un critère d'arrêt, appelé critère de qualité.

- Dans la boucle de construction de la clause (spécialisation), le critère de consistance (ne couvre aucun exemple négatif) permet d'arrêter l'ajout de littéraux sur le corps de la clause.
- Dans la boucle de construction de l'hypothèse (couverture) le critère de complétude (couvre tous les exemples positifs) permet d'arrêter l'ajout de clauses dans l'hypothèse.

Les systèmes PLI Parmi les systèmes de programmation logique inductive [Lavrac & Dzeroski 1994], nous pouvons citer FOIL [Quinlan & Cameron-jones 1993], mFOIL [Lavrač *et al.* 1996], GOLEM [Muggleton & Feng 1990], PROGOL [Muggleton 1995] [Muggleton 1997, Muggleton 2002], LINUS [Dzeroski & Lavrac. 1991], ...

5.3 Apprentissage auto-adaptatif

5.3.1 Estimation des probabilités de réussite

Comme nous l'avons dit précédemment, nous sommes en présence d'un problème stochastique et nous n'avons à notre disposition que l'historique des actions pour faire notre apprentissage. Cet historique est essentiellement constitué d'informations recueillies sur l'équipement et d'actions qui ont été appliquées et de leurs conséquences (de nouvelles informations). Il nous sera nécessaire de mettre en place un mécanisme pour combler l'absence de matrice de transition sans pour autant être obligé d'attendre trop longtemps pour avoir un historique suffisant.

Nous représentons l'historique des transitions à travers les observations qui sont tirées du système. Une observation est un quadruplet $o = (t, \varepsilon_1, a, \varepsilon_2)$ signifiant qu'à l'instant t l'équipement était à l'état ε_1, l'action a a été exécutée et l'équipement s'est retrouvé à l'état ε_2. Ici, les observations ont la forme la plus basique possible pour évaluer l'efficacité individuelle des actions sur un état donné. Comme le montre

la figure 5.2, les trois possibilités qui en découlent seront alors soit une transition vers un état meilleur, soit un état qui reste inchangé, soit une transition vers un état moins bon. Ici, le terme moins bon fait référence à la relation d'ordre qu'induit le critère de performance P_r dans l'ensemble des états E.

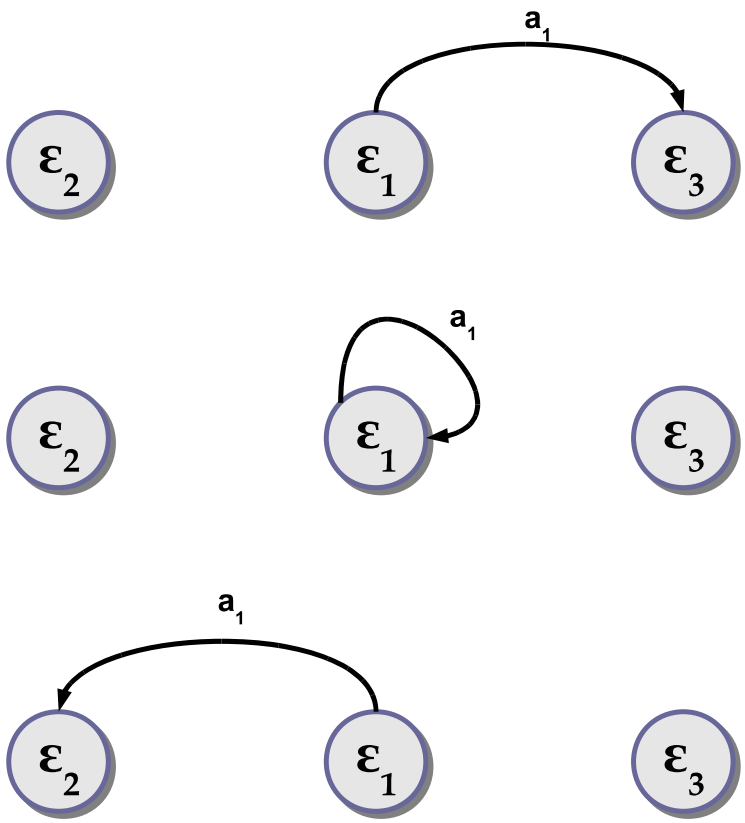

FIGURE 5.2 – Transitions envisageables

A partir de ces cas de figures, il est possible de diviser les observations en deux catégories : les observations positives et les observations négatives. Les observations positives sont les transitions d'un état vers un état meilleur, suite à l'application d'une action, alors que les observations négatives sont des transitions vers un état moins bon suite à l'application d'une action (des détériorations de performances) ou bien lorsque l'état reste inchangé. Les observations dans lesquelles les états de départ et d'arrivée coïncident sont rangées parmi les observations négatives car nous les considérons comme des transitions avec des actions potentiellement sans effet sur l'état considéré. Notons O^+ (resp. O^-) l'ensemble des observations positives

(resp. négatives) et O l'ensemble des observations. Contrairement à la méthode d'apprentissage par renforcement qui a besoin d'une fonction de récompense et un but pour évoluer, ici l'observation de l'état du système à l'étape suivante permet d'évaluer la pertinence de l'action qui a été réalisée. On ne peut pas supposer qu'il a un effet directe mais la répétition permet d'avoir une idée de l'effet de l'action sur le système.

L'exploitation des observations constitue la première étape permettant d'avoir une estimation de la probabilité de réussite d'une actions a sur un état ε. En effet, l'ensemble des transitions représente des statistiques sur le taux d'efficacité de l'action. Il y a deux manières de considérer ce taux de réussite :

- Soit on considère l'ensemble des triplets $(\varepsilon_1, a_1, \varepsilon_2)$: dans ce cas on a une connaissance assez détaillée pour pouvoir construire une fonction de prédiction δ des états dans lesquels le système va se trouver en appliquant les actions. Cependant, pour que cette estimation de la fonction de transition soit représentative, un grand nombre d'observations est nécessaire. En d'autres termes, il faut avoir observé un nombre significatif de fois le même triplet pour déterminer si cette transition est crédible ou pas. Sachant que le nombre de transitions possibles peut être très grand, le nombre de données générées peut facilement exploser ;
- Soit on considère l'ensemble des couples (ε, a) où le calcul du taux de réussite se fait sur les couples état-action en regroupant toutes les observations qui ont la même entête. Dans ce cas on "perd" la fonction de prédiction des transitions mais on a besoin de moins d'exemples pour la représentativité. C'est à dire qu'on regroupe l'ensemble des observations qui une à une ne sont pas représentatives en une entité moins fine mais qui permet de prendre une décision avec beaucoup moins d'exemples.

Nous pouvons noter que chaque nouvelle observation dont a besoin le système est potentiellement une erreur que le système fait volontairement pour avoir un nombre d'exemples suffisant pour établir la probabilité de réussite. Le système teste à plusieurs reprises une action pour s'assurer qu'elle est bonne ou mauvaise, il faut donc un moyen pour diminuer ces tests sans perdre la représentativité des probabilités de réussite. Il apparait que la seconde méthode est mieux adaptée car on ne peut pas envisager de laisser le système tester un trop grand nombre de

triplets.

On cherchera par la suite à trouver un compromis entre l'utilisation de la connaissance du domaine fournie par l'administrateur et l'exploration des couples état-action.

Si nous notons $||(\varepsilon,a)||^+$ le nombre d'observations positives, $||(\varepsilon,a)||^-$ le nombre d'observations négatives, et $||(\varepsilon,a)|| = ||(\varepsilon,a)||^+ + ||(\varepsilon,a)||^-$ le nombre total d'observations de (ε,a). Le rapport $\frac{||(\varepsilon,a)||^+}{||(\varepsilon,a)||}$ est une estimation naïve de la probabilité de réussite de l'action a sur l'état e. En effet, pour un petit nombre d'observations, la probabilité sera largement sur-estimée et l'exploration en sera plus ou moins réduite. Cette estimation étant mauvaise lorsqu'il y a peu d'observations, nous utilisons un *régulateur de Laplace*, c'est-à-dire l'initialisation des probabilités avec des valeurs *à priori* qui restent prédominantes tant que le nombre d'observations est faible. Ceci permet également d'affiner l'exploration des états. Par exemple, en surestimant initialement les probabilités de réussite *à priori* on favorisera une plus large exploration des actions : des actions a priori peu pertinentes vont toutefois être tentées, aux risque de rendre le fonctionnement de l'équipement peu pertinent en début d'apprentissage, ou lorsque l'environnement change.

Définition 5.3.1.1 *Pour toute stratégie, nous définissons son indice de réussite* θ_t *comme suit :*

- $\theta_t(\varepsilon,a) = \frac{||(\varepsilon,a)||^+ + n_{(\varepsilon,a)}}{||(\varepsilon,a)|| + m_e}$ *pour tout couple* (ε,a)
- $\theta_t(\varepsilon,[a_1,a_2,\ldots,a_n]) = \frac{1}{n}\sum_{i=1}^{n}\theta_t(\varepsilon,a_i)$ *pour tout couple* $(\varepsilon,[a_1,a_2,\ldots,a_n])$
- m_e *et* $n_{(\varepsilon,a)}$ *représentent les cœfficients régulateurs de Laplace :*
  $m_e = \sum_{(\varepsilon,a)\in\Sigma\times\Lambda}(n_{(\varepsilon,a)})$

Le choix du régulateur dépend de la liberté que le concepteur veut donner au niveau de l'exploration des état-actions (c'est-à-dire la connaissance du domaine qu'il souhaite introduire dans le système). Plus m_ε et $n_{(\varepsilon,a)}$ sont grands, plus le système va explorer les couples état-actions avant de les classer comme étant pertinents ou pas. Il faut, cependant, que ces valeurs ne soient pas trop grandes car le but est qu'elles soient non significatives pour un nombre représentatif d'observations. Comme on peut le remarquer, θ_t est une moyenne mathématique.

La valeur de $\theta_t(\varepsilon,a)$ comme le sous entend son indice dépend du temps, elle est mise à jour de manière régulière par rapport à l'apparition de chaque observation

$(\varepsilon, a, \varepsilon')$. Dans un environnement réseau traversée par un modèle de flux, et disposant d'une loi de probabilité $\mathbb{P}$ qui définit la réussite d'une action, il est intéressant de voir dans quelle mesure θ_t est lié à cette loi de probabilité.

Nous énonçons la proposition suivante :

Proposition 5.3.1.1 *Dans un environnement stable (suivant un modèle traversé par un type de flux bien déterminé) suffisamment longtemps,* $\mathbb{P}(\varepsilon, a)$ *et* $\mathbb{P}(\theta_t(\varepsilon, a) \geq \frac{1}{2})$ *sont du même coté par rapport à* $\frac{1}{2}$ *:*

- *Si* $\mathbb{P}(\varepsilon, a) \geq \frac{1}{2}$ *alors* $\mathbb{P}(\theta_t(\varepsilon, a) \geq \frac{1}{2}) \geq \frac{1}{2}$
- *Si* $\mathbb{P}(\varepsilon, a) \leq \frac{1}{2}$ *alors* $\mathbb{P}(\theta_t(\varepsilon, a) \leq \frac{1}{2}) \leq \frac{1}{2}$

Preuve : Si $\mathbb{P}(\varepsilon, a) \geq \frac{1}{2}$ alors la probabilité pour que $\|(\varepsilon, a)\|^+ \geq \frac{1}{2}\|(\varepsilon, a)\|$ est supérieure à $\frac{1}{2}$. En effet, $\mathbb{P}(\varepsilon, a) \geq \frac{1}{2}$ signifie que le nombre de réussites en appliquant l'action a à l'état e devrait dépasser la moitié du nombre total d'essais de l'action sur cet état.

Or $\mathbb{P}(\|(\varepsilon, a)\|^+ \geq \frac{1}{2}\|(\varepsilon, a)\|) = \mathbb{P}(\frac{\|(\varepsilon, a)\|^+}{\|(\varepsilon, a)\|} \geq \frac{1}{2}) = \mathbb{P}(\theta_t(\varepsilon, a) \geq \frac{1}{2})$ pour un nombre non nul d'observations. D'où le premier résultat. On montre de la même manière la seconde affirmation de la proposition.

Ainsi la valeur de $\theta_t(\varepsilon, a)$ nous informe bien sur l'efficacité de l'action a à l'état e si le système suit un modèle assez longtemps. Cette propriété nous dispense de la connaissance exhaustive de la loi de probabilité.

En ce qui concerne la convergence de la suite $\{\theta_t(\varepsilon, a)\}_{t \in \mathbb{R}}$, elle dépend fortement de la distribution de probabilité. Si elle est telle que la suite est monotone, alors elle converge vers une valeur pour laquelle nous n'avons pas la garantie que c'est la vraie probabilité mais qui est placée du bon coté pour nous indiquer la vraie efficacité de l'action. Nous allons assimiler dans le reste du rapport les deux valeurs, et nous pouvons dès lors dire qu'en apprenant au fur et à mesure les valeurs de θ_t, nous pouvons avoir de bonnes estimations de l'efficacité des actions.

5.3.2 Apprentissage auto-adaptatif de stratégies

L'objectif de l'algorithme d'auto-adaptation, que nous allons décrire à présent, est d'utiliser les propriétés que nous avons décrites dans la section précédente pour

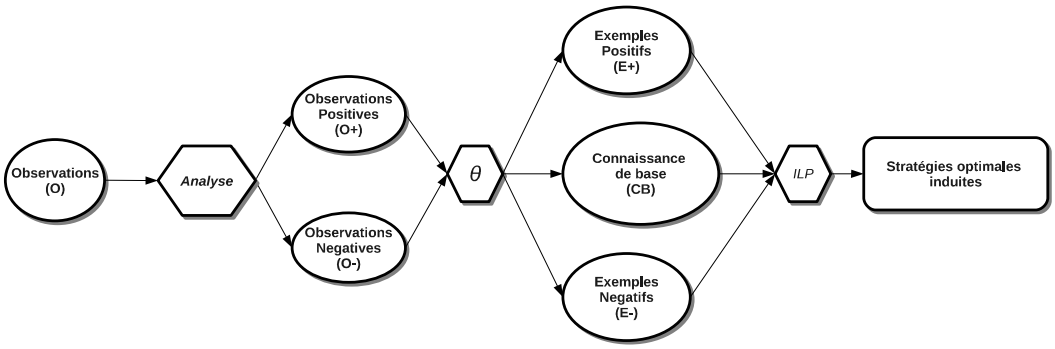

FIGURE 5.3 – Traitement des données

en déduire des stratégies de configuration en utilisant la PLI et l'échange de connaissance pour éviter une exploration aveugle de l'ensemble des couples états-actions.

La figure 5.3 présente les différents traitements qui sont effectués sur les données pour en arriver aux stratégies d'adaptation.

Les observations sont constituées à partir de l'historique des transitions qui sont survenues sur l'équipement. Cet ensemble d'observations est ensuite analysé en utilisant la relation d'ordre implicite qu'induit le critère de performance sur l'ensemble des états. Il va découler de cette analyse deux catégories d'observations : positives et négatives. Ici, le critère de performance n'intervient pas directement dans le processus d'adaptation mais permet seulement de déterminer la préférence d'un état par rapport à un autre, et donc de construire une comparaison qualitative entre les états. Ensuite, les observations sont organisées en ensembles de couples (ε, a) regroupant l'ensemble des observations $(\varepsilon, a, *)$. L'indice de réussite θ_t est calculé pour chaque couple de manière incrémentale en fonction de l'arrivée des observations. La valeur de θ_t permettra de distinguer les actions efficaces sur un état donné et de construire les exemples nécessaires au processus PLI. Les sections suivantes vont permettre de détailler l'ensemble de ces éléments ainsi que l'algorithme.

5.3.2.1 Base de connaissance B_C

L'utilisation de la programmation logique, comme nous l'avons vu dans la section 5.2.4, nécessite deux concepts en entrée : les exemples qui sont des faits (au sens Prolog du terme) décrivant le concept qui est appris et la connaissance du do-

maine qui décrit les objets (clauses et types) qui peuvent être utilisés pour décrire le concept.

Nous utilisons comme environnement de programmation Aleph [Srinivasan 2002] dans nos développements et nous faisons appel à la notation $Aleph.induce(B, E^+, E^-)$ pour exprimer la sortie du programme dans les algorithmes que nous définirons.

Nous notons la base de connaissance qui contient ces données B_C. Cette section est entièrement dédiée à la description de son contenu.

La base B_C contient les connaissances B du domaine ainsi que les exemples E^+ et E^- nécessaires à l'apprentissage. Elle contient la description des états, des actions et les règles ordonnant (au moins partiellement) l'ensemble des états. Elle contient l'ensemble des informations que possède un équipement du réseau et qui ne résulte pas d'un processus d'apprentissage.

La définition des états peut se faire en utilisant un seul paramètre (comme le taux de perte), ou encore plusieurs paramètres pour trouver un compromis de configuration lors de l'adaptation (ε utilisant une règle par exemple). L'utilisation d'un paramètre pour définir les états a l'avantage d'être simple à gérer et consiste en un découpage d'un ensemble linéaire. Le paramètre de performance sera, dans ce cas, simplement la relation d'ordre dans l'ensemble des valeurs du paramètre (des nombres réels en général). Cependant, une description linéaire suppose qu'on puisse exhiber un paramètre qui doit être optimisé en priorité. La seconde manière de définir les états est l'utilisation de plusieurs paramètres (taux de pertes, délai, taille des files d'attentes, taux d'utilisation de la bande passante, ...), ce qui permet de prendre en compte plusieurs aspects de la ressource à gérer. Cependant, dans ce cas de figure il y a un risque important d'ambiguïté et le découpage de l'espace des états est un peu plus compliqué. En effet, il faudra choisir en priorité les paramètres à utiliser pour ordonner les états. Dans tous les cas, il faudra s'assurer que le processus d'apprentissage pourra toujours déterminer l'état courant. Pour simplifier la présentation de l'algorithme, nous continuerons à utiliser la notation ε pour les états et supposons qu'il existe un mécanisme de traduction entre les symboles et la définition des états. Nous utilisons également la notation Σ pour designer l'ensemble des états et $etat/1$ désigne le prédicat d'*arité* (nombre d'arguments) 1 qui définit les états de manière symbolique dans le système de programmation logique inductive.

Exemple : Un exemple de définition d'état est $\{\varepsilon_0 =$ pertes $< 10\%$, $\varepsilon_1 =$ pertes

entre 10% et 15%,... }.

Cet exemple permet d'illustrer l'importance de la relation d'ordre entre les états dont nous avons déjà parlé. Un ISP [1] pourrait considérer qu'un état $e_1 = (15\%$ de pertes, $2ms$ de délai) est meilleur qu'un autre état $e_2 = 10\%$ de pertes, avec $5ms$ de délai) alors que cela pourrait être le contraire pour autre ISP. Cependant, pour qu'un apprentissage soit possible, il est nécessaire de pouvoir déterminer si une stratégie produit un bon résultat ou pas, en comparant les états de l'equipement avant et après son application. La relation d'ordre dans l'ensemble des états doit donc être définie préalablement à l'apprentissage. Nous appelons cette relation d'ordre par le prédicat $Meilleur/2$ qui permet de comparer deux états, $Meilleur(\varepsilon_1, \varepsilon_2)$ signifie que ε_1 est meilleur que ε_2.

Enfin, il faut que l'apprentissage dispose d'un ensemble d'actions qu'il pourra utiliser pour construire des stratégies. Un exemple d'action pourrait être la suppression d'un ensemble d'entrées dans une table de routage pour redécouvrir un chemin (après un échec) ou la décision de préférer un chemin plutôt qu'un autre. Pour les actions, nous continuons à donner une notation symbolique a, l'ensemble des actions est noté Λ et le prédicat $action/1$ décrit les actions dans l'ensemble des actions élémentaires qui peuvent être exécutées sur l'équipement ou sur la configuration de la ressource gérée.

Une stratégie est une correspondance entre un état et une ou plusieurs actions. L'apprentissage pour l'auto-adaptation consistera à apprendre à construire ces correspondances au fil de l'activité. Nous définissons le prédicat $Strategie/2$ pour les deux types de stratégies :

- $Strategie(\varepsilon, a)$ pour les stratégies élémentaires (élément de $\Sigma \times \Lambda$) ;
- $Strategie(\varepsilon, [a_1, a_2, \ldots, a_n])$ pour les stratégies complexes (élément de $\Sigma \times \Lambda^n$).

Ce prédicat dans sa forme élémentaire correspond au couple état-actions que nous avions utilisé pour définir l'indice de réussite θ_t. Ainsi la valeur de réussite d'une $Strategie(\varepsilon, a) = \theta_t(\varepsilon, a)$ et $Strategie(\varepsilon, [a_1, \ldots, a_n]) = \theta_t(\varepsilon, [a_1, \ldots, a_n])$. Ce prédicat signifie pour un état donné exécuter le ou les actions.

Nous utilisons ce prédicat, également, pour définir les exemples positifs et négatifs pour le processus de programmation logique inductive.

1. Internet Service Provider

Définition 5.3.2.1 *Une stratégie* $Strategie(\varepsilon, a)$ *(resp.* $Strategie(\varepsilon, [a_1, \ldots, a_n])$*) est un exemple positif ssi* $\theta_t(\varepsilon, a) > \frac{1}{2}$ *(resp.* $\theta_t(\varepsilon, [a_1, \ldots, a_n]) > \frac{1}{2}$*)) et un exemple négatif sinon.*

Une stratégie est donc un exemple positif, s'il a plus de chance de produire une amélioration de l'état du système, un exemple négatif sinon. Une amélioration est une transition vers un état meilleur. Une stratégie complexe signifie que les actions dans la liste sont à exécuter de manière combinée.

Les stratégies peuvent passer de E^+ à E^- ou l'inverse suivant l'évolution des observations.

Enfin, nous introduisons le prédicat $Transition(\varepsilon, a, \varepsilon')$ qui signifie que le module a observé au moins une fois une transition de ε à ε' en appliquant l'action a. Ce prédicat est présent pour enrichir la description du concept. Il est présent en tant qu'ensemble de faits. Par contre, pour ne pas se trouver en présence de transitions contradictoires, nous avons fait la sélection suivante :

- Pour tout couple (ε, a) telle que $Strategie(\varepsilon, a) \in E^+$, inclure dans la base toutes les transitions $Transition(\varepsilon, a, \varepsilon')$ telle que $\varepsilon' > \varepsilon$ (inclure les observations positives) ;
- Pour tout couple (ε, a) telle que $Strategie(\varepsilon, a) \in E^-$, inclure dans la base toutes les transitions $Transition(\varepsilon, a, \varepsilon')$ telle que $\varepsilon' \leq \varepsilon$ (inclure les observations négatives).

5.3.2.2 Algorithme d'auto-adaptation

L'algorithme d'auto-adaptation s'exécute à travers un module d'apprentissage comme décrit sur la figure 5.4. Le module est composé d'une base de connaissance contenant toutes les stratégies et les connaissances de base, un moteur de raisonnement qui va servir à extraire les nouvelles règles induites par la programmation logique inductive et enfin le module d'échange de connaissances que nous détaillons par la suite(section 5.4).

La fonction de monitoring, comme dans le cas de l'architecture IBM, permet de récupérer les informations sur la situation de la ressource gérée à travers le plan de contrôle et le plan de données. Ces informations sont traduit ensuite par l'analyseur qui s'occupe de le mettre sous un format (prédicats, observation) com-

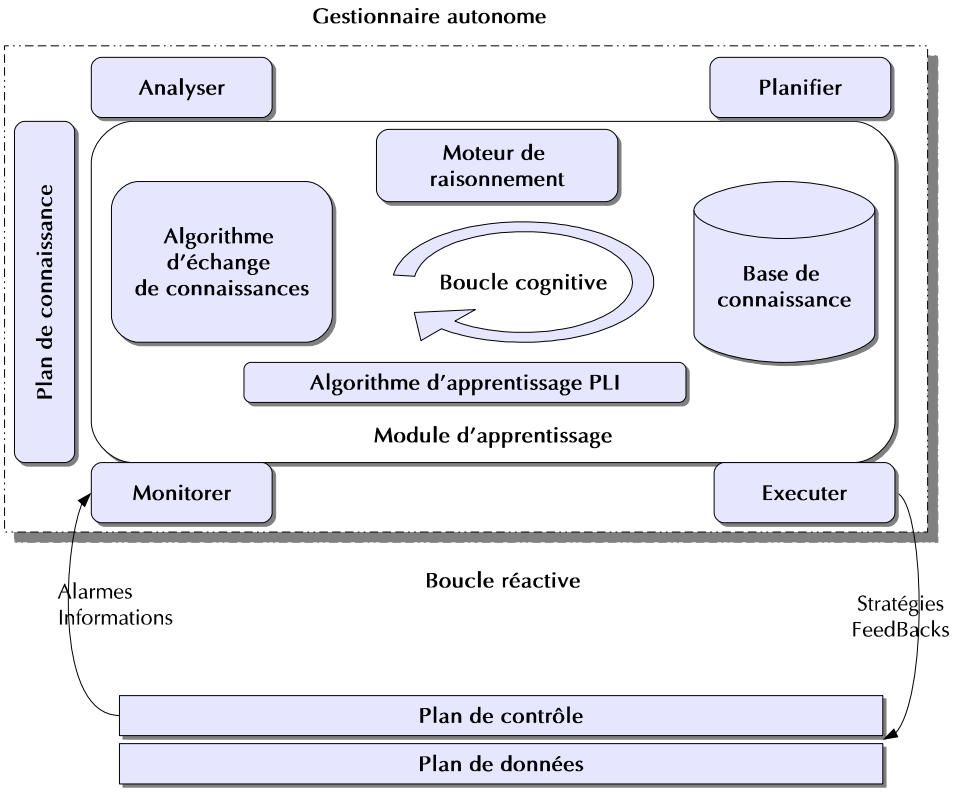

FIGURE 5.4 – Architecture du module d'apprentissage

préhensible (état) par le module d'apprentissage. Le module de raisonnement prend le relai pour choisir dans la base de connaissance les actions appropriées (selon le contenu de la base). Ensuite, ces actions sont passées au *planifier* qui fait appel à la fonction d'exécution pour traduire les actions en instructions de configuration sur la ressource. Au moment de l'arrivé de la *feedback*, une observation est créée, et, ensuite l'algorithme d'apprentissage PLI prend le relai pour déduire des règles générique et par déduction identifier de nouvelles stratégies pour mettre à jour la base de connaissance. Il y a donc une boucle réactive qui passe par le moteur de raisonnement pour le choix des actions réactives et une boucle cognitive qui met à jour la base de connaissance en fonction de l'historique.

L'algorithme 3 décrit le comportement du processus d'apprentissage d'auto-adaptation d'un élément. Le processus fonctionne en réaction à des événements qui sont les changements d'états du système. Au démarrage, ne disposant pas d'un nombre représentatif d'observations le système va choisir les actions à exécuter

en fonction de la valeur donnée aux *régulateurs de Laplace*. Ainsi, il se construit des observations en testant les actions qui sont considérées comme fonctionnant mieux *à priori* selon ces valeurs. En ayant, au début, à disposition ces valeurs fournies par le concepteur, il n'est pas certain que les actions qui seront retenues à priori par le processus d'apprentissage auto-adaptatif seront les plus efficaces pour chaque situation. L'astuce que nous proposons pour orienter l'apprentissage est de trouver des règles génériques en utilisant la programmation logique inductive qui découlent des règles déjà connus de par l'historique pour déduire des actions à explorer en priorité. L'avantage est la possibilité d'enrichir la base de connaissance par les actions qu'il est possible d'effectuer et aussi de s'appuyer sur l'expertise du concepteur au début.

Lorsque le système acquiert un nombre représentatif d'observations pour un couple (ε, a) il va calculer la valeur réelle de $\theta_t(\varepsilon, a)$, et à chaque changement d'état vers e au niveau de l'équipement, le nouvel état est notifié au module d'apprentissage qui mettra la valeur de $\theta_t(\varepsilon, a)$ à jour . Ensuite, le processus en réaction au changement d'état, se chargera d'extraire la stratégie optimale (s_0), parmi l'ensemble des stratégies dont l'état initial est l'état courant ε.

L'action optimale est déterminée par rapport à la valeur de $\theta_t(\varepsilon, *)$ dans un premier temps $s_0 = max\{\theta_t(\varepsilon, A)_{A \in \Lambda}\}$, Λ pouvant être une ou plusieurs actions.

Pour orienter l'exploration il sera nécessaire de faire un choix parmi les actions à explorer par l'apprentissage. La figure 5.5 décrit le cas type qui nécessite un choix du concepteur du système. Supposons qu'il existe un état ε_i tel qu'en appliquant respectivement les actions $a_{i_1}, a_{i_2}, \ldots, a_{i_n}$ on passe dans les états $\varepsilon_{i_1}, \varepsilon_{i_2}, \ldots, \varepsilon_{i_n}$ avec des probabilités de transition $P_{i_1}, P_{i_2}, \ldots, P_{i_n}$ tels que ε_i moins bon que ε_{i_1} qui est moins bon que ε_{i_2} ... qui est moins bon que ε_{i_n} et $P_{i_1} \geq P_{i_2} \geq \ldots \geq P_{i_n}$. Dans ce cas, faut-il choisir a_{i_1} qui mène au plus petit état atteignable qui est meilleur que ε_i mais avec une forte probabilité ou bien a_{i_n} qui atteint un meilleur état mais avec une probabilité plus faible de réussir à l'atteindre ? La première approche paraissant moins risquée, c'est le choix que nous avons fait.

Après le choix de la stratégie optimale, le(s) action(s) de cette stratégie sont ensuite appliquées pour adapter la configuration, et l'équipement se met en attente d'un *FeedBack* qui lui permettra d'observer l'effet de l'action sur la ressource.

Suite à ce retour, le module construit une observation qui permettra de mettre

Algorithme 3 : Module d'apprentissage

Données :

B_C : Connaissances de base;

B_L : Connaissances locales;

B_S : Connaissances reçues;

ε : Etat ;

s_0 : Stratégie;

$FeedBack \leftarrow FAUX$: Booléen

```
1  début
2      si FeedBack = FAUX alors
3          /*A la notification d'un nouvel état */;
4          ε ← EtatCourant();
5          s0 ← BL.extraireStratOptimale(ε);
6          si s0 ≠ NULL) ou (s0 ← BS.extraireStratOptimale(ε)) ≠ NULL
           alors
7              Appliquer(s0.Actions());
8              FeedBack ← VRAI;
9          sinon
10             SORTIR avec ECHEC;
11     sinon
12         /* A La réception d'un retour */;
13         ε ← EtatCourant();
14         ContruireObservation(t, s0.Actions(), ε);
15         MettreAjour(theta(s0));
16         MettreAjour(BL);
17         H ← Aleph.induce(BL.BC, BL.E+, BL.E−);
18         si H ≠ NULL alors
19             BL ← BL ∪ H;
20             EchangerConnaissance(H)
21 fin
```

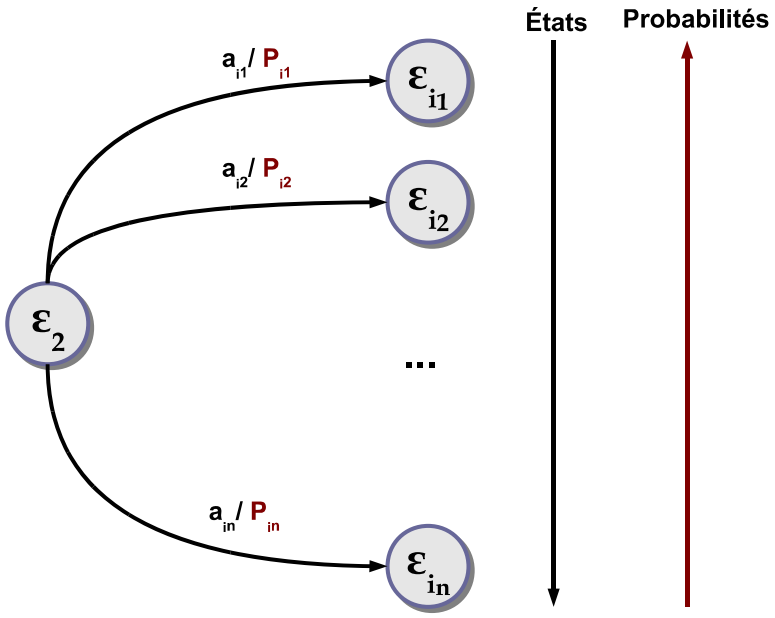

FIGURE 5.5 – Cas problématique

à jour la valeur de θ_t pour la stratégie s_0 qui vient d'être choisie. Cette mise à jour conduit à la mise à jour également de la base locale B_C en recalculant les exemples positifs et négatifs. Si un exemple change de label en passant d'exemple positif à un négatif, toutes ses observations positives sont retirées de la base de connaissance et ses observations négatives vont être introduites à la place avec le prédicat *Transition*/3. Si le changement de label se fait dans le sens inverse, les observations négatives seront retirées et les positives introduites avant de lancer le processus de PLI.

Ces nouvelles données vont être fournies au module de programmation logique inductive qui se chargera de trouver des règles génériques qui sous-tendent les stratégies efficaces déjà trouvées. Notons que le fait de considérer les stratégies non efficaces comme des exemples négatifs, nous assure que les règles apprises ne couvrent pas de stratégies inefficaces. Le module de d'apprentissage permet, ensuite, à partir de la règle plus générale apprise de déduire des stratégies qui ne sont pas encore découvertes. Ces nouvelles stratégies n'ont pas encore de Labels (positifs ou négatifs). Ces stratégies seront testées en priorité sur les stratégies qui ne disposent que de l'indicateur de Laplace et qui ne sont pas encore déductibles à partir du résultat de la programmation logique inductive.

Les base B_L et B_S seront présentées plus loin, à la section 5.4.

5.3.3 Exemple d'exécution

Nous allons dans cette partie présenter un exemple d'exécution de l'algorithme d'adaptation pour montrer l'apport de la PLI. Considérons les listings suivants qui sont un exemples de l'ensemble de la connaissances de base (listing 5.1) et les exemples positifs et négatifs (listing 5.2) sous une forme de représentation Prolog.

Listing 5.1 – Connaissance de base

```
1  /*Connaissance de base */
2  ...
3  /*Etats*/
4  etat(ε1). etat(ε2). etat(ε3). etat(ε4).
5
6  /* Actions */
7  action(a1). action(a2). action(a3). action(a4). action(a5).
8  action(a6). action(a7). action(a8). action(a9). action(a10).
9  action(a11). action(a12). action(a13). action(a14). action(a15).
10
11 /* ε4 meilleur que ε3 meilleur que ε2 meilleur que ε1 */
12 meilleur(ε2, ε1). meilleur(ε3, ε1). meilleur(ε3, ε2).
13 meilleur(ε4, ε1). meilleur(ε4, ε2). meilleur(ε4, ε3).
14
15 /* Les transitions incluses */
16 transition(ε1, a1, ε2). transition(ε1, a2, ε4).
17 transition(ε1, a3, ε3). transition(ε1, a5, ε3).
18 transition(ε1, a6, ε4). transition(ε1, a7, ε3).
19 transition(ε1, a4, ε1).
20
21 transition(ε2, a3, ε3). transition(ε2, a1, ε1).
22 transition(ε2, a2, ε4). transition(ε2, a4, ε1).
23 transition(ε3, a4, ε4).
```

La connaissance de base est composée, entre autre, des prédicats que nous avions définies dans la section 5.3, à savoir, *etat/1*, *action/1*, *transition/3* et *meilleur/2*. Pour les trois premiers prédicats, il s'agit seulement de déclarations de types alors que pour *transition/3* il s'agit du résultat de la sélection dont nous avons déjà parlé. Cette sélection, nous le rappelons, consiste à choisir, pour les exemples positifs, la transition positive (dérivée d'une observation positive) la plus fréquente, et pour les exemples négatifs, la transition négative la plus fréquente. Donc les transitions que nous avons sur le listing 5.1 ne sont qu'une capture de l'état de cette base en un instant donné. Par exemple : la présence de `transition(`ε_1`,` a_1`,` ε_2`)` veut dire que le couple (ε_1,a_1) est un exemple positif et que parmi toutes les transitions observées pour ce couple, celle qui va vers ε_2 est la plus fréquente. Ces prédicats servent à décrire le concept qui est apprise qui sont les associations états-actions, les stratégies qui sont représentée par le prédicat *strategie/2*.

Listing 5.2 – Exemples positifs et négatifs

```
1  /* Exemples positifs */
2
3  strategie(ε1, a1). strategie(ε1,a2).
4  strategie(ε1, a3). strategie(ε1,a5).
5  strategie(ε1, a6). strategie(ε1,a7).
6
7  strategie(ε2, a2). strategie(ε2,a3).
8
9  strategie(ε3, a4).
10
11 /* Exemples negatifs*/
12
13 strategie(ε1, a4).
14
15 strategie(ε2, a4). strategie(ε2, a1).
```

La situation de représentée par les deux listings 5.1 et 5.2 peut être illustrée par les graphes de transitions de la figure 5.6. Cette représentations nous permettra de faire apparaitre plus facilement l'apport de la PLI dans cet exemple. Ces graphes permettent de faire voir, par exemple, le fait que pour l'état ε_2, les actions a_1 et a_2 sont efficaces et que a_3" et a_4 ne le sont pas. Sur la quinzaine d'actions définie, le système ne peut répondre a propos de l'efficacité que de quatre actions sur cet état.

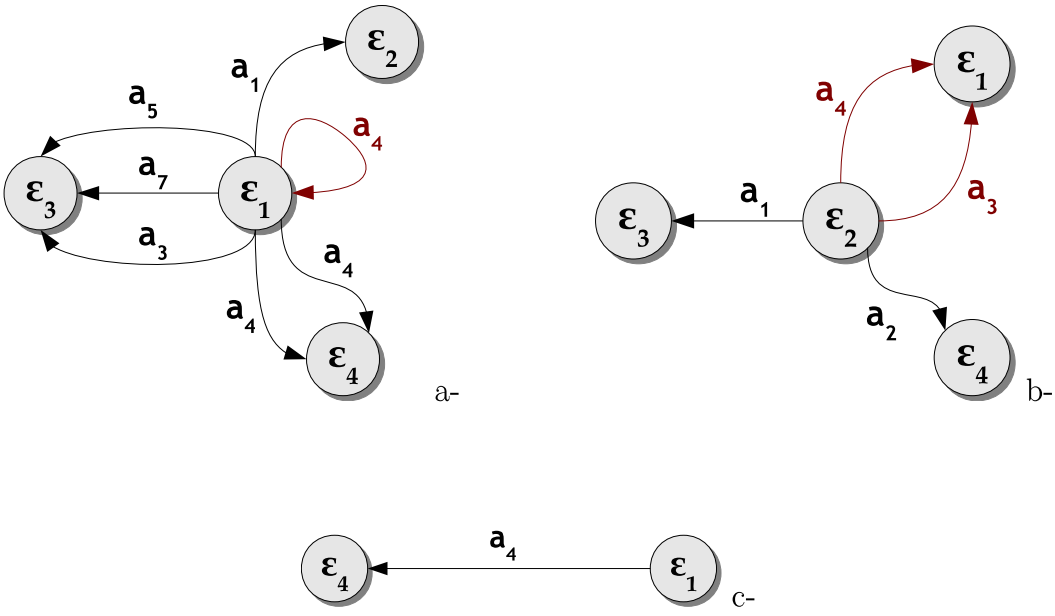

FIGURE 5.6 – Graphes de transitions : a- pour l'état ε_1, b- pour l'état ε_2, c- pour l'état ε_3. Les arêtes noires sont les stratégies positives et les arêtes rouges les stratégies négatives.

En cas d'échec des deux actions qui sont connues comme efficaces, le système devra tester parmi le reste des actions non encore exploré. Nous utilisons la PLI, pour guider ce choix en apprenant des stratégies à partir de celle déjà connues. Avec les données des deux listing, le moteur PLI Aleph, produit comme résultat les règles décrites dans le listing 5.3.

Listing 5.3 – Règles apprises

```
1
2 strategie(ε1, A) :-
3     transition(ε1, A, B), meilleur(B, ε1).
4
5
6 strategie(ε2, A) :-
7     transition(ε1, A, B), meilleur(B, ε2).
8
9 strategie(ε2, a4).
```

La première règle veut dire que les actions efficaces sur l'état ε_1 sont celle qui contiennent les transitions vers un meilleur état que ε_1 dans la connaissance de base. Nous rappelons que pour lire cette règle il faut considérer les majuscules comme des variables quantifiée universellement. En d'autres termes, la première règle pourrait être traduite en écrivant : $\forall A$ *et* B , si $meilleur(B,A)$ et $transition(\varepsilon_1,A,B$ sont vraies alors la stratégie $strategie(\varepsilon_1,A)$ est efficace. Cette règle ne nous permet pas d'apprendre de nouvelles stratégies mais elle donne une information l'efficacité de l'apprentissage qui respecte le choix qui avait été fait de mettre les transitions positives pour les exemples positifs.

Cependant, la seconde règle est plus intéressante. Elle signifie que toute action A qui permet de passer de l'état ε_1 à un état qui est meilleur que ε_2 devrait permettre d'améliorer l'état ε_2. Cette règle peut sembler aller de soi pour un être humain (administrateur) mais elle n'est pas évident. En réalité, elle n'est vrai que pour cet état de la base et peut évoluer mais elle permet au système de compléter une partie des actions qui ne sont pas encore testés sur l'état ε_2. Ainsi, par déduction nous avons les stratégies supplémentaires $strategie(\varepsilon_2,a_5)$, $strategie(\varepsilon_2,a_6)$ et $strategie(\varepsilon_2,a_7)$. Ces stratégies ne sont pas prioritaires sur les exemples positifs car ils n'ont pas encore de label (positif ou négatif). Cependant, elles ont une position privilégiée dans l'ensemble des actions $\{a_6,a_{15}\}$. Ainsi, de proche en proche,

le module d'apprentissage découvre les stratégies susceptibles d'être efficaces dans l'ensemble des actions sur un état données et ainsi avoir une recherche orientée par ce qui est déja connue.

5.4 Collaboration par échange de connaissances

5.4.1 Considérations architecturales

Dans notre approche, nous considérons deux boucles pour effectuer les tâches du plan de connaissance (Figure 5.7). La *boucle d'adaptation* qui représente les échanges entre le plan de connaissance et la ressource locale, et la *boucle d'auto-organisation* qui représente les échanges entre les différents éléments du plan de connaissance. Le plan de connaissance est composé d'éléments distribués dans le réseau, chacun gérant un groupe de ressources.

La boucle d'auto-adaptation sert donc à une adaptation locale alors que la boucle d'auto-organisation sert à des objectifs de collaboration.

Cette collaboration présente un certains nombre d'avantages :

- L'initialisation des éléments qui arrivent dans le réseau : Les nouveaux éléments du réseau vont bénéficier de l'expérience des éléments qui étaient déjà dans le réseau. Cette initialisation peut s'avérer plus efficace que l'utilisation des *régulateurs de Laplace* bien que ces derniers soient très utiles ;
- Le rééquilibrage de la distribution des exemples entre les différentes parties du réseau : Les modules d'apprentissage situés à différents endroits du réseau peuvent ne pas apprendre au même rythme. Les éléments qui sont dans des zones de forte activité vont apprendre très rapidement car ils obtiennent beaucoup d'observations alors que les éléments situés dans les zones "calmes" vont apprendre plus lentement. L'échange d'observations et de stratégies peut permettre de rééquilibrer cela ;
- La validation distribuée de la connaissance : les éléments qui collaborent vont critiquer mutuellement les connaissances apprises en comparant des connaissances qui se trouvent sur les différents nœuds grâce au processus d'échange de la connaissance ;
- La gestion distribuée de la connaissance : Chaque élément garde au niveau lo-

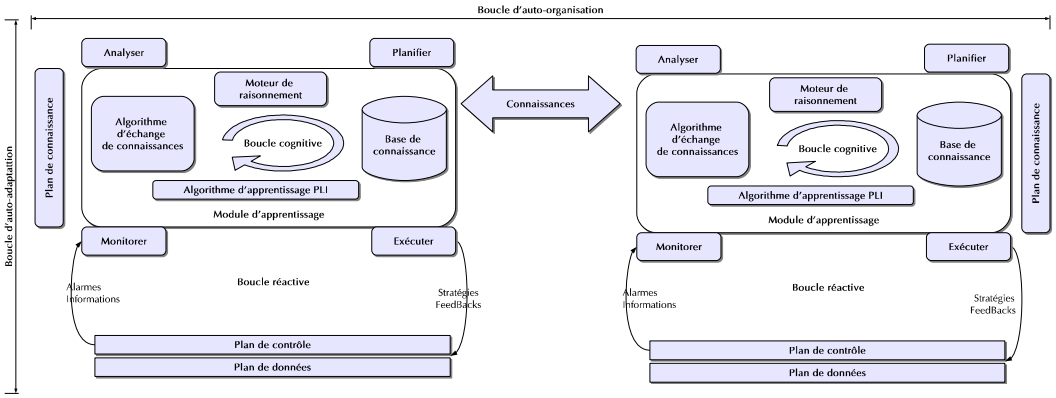

FIGURE 5.7 – Boucles d'auto-adaptation et d'auto-organisation

cal uniquement les connaissances nécessaires à son adaptation, tout en échangeant avec ses voisins. Nous obtenons, ici, le caractère distribué du plan de connaissance.

Nous pouvons noter que dans notre cas, l'adaptation jusqu'ici prend assez peu en compte les voisins se contentant des observations produites localement. La raison est double :

- Nous voulons avoir un système qui puisse fonctionner sans trop dépendre des voisins. On pourrait dire qu'un élément autonome devrait pouvoir se débrouiller et encore mieux s'il a des voisins ;
- La question de savoir jusqu'à quel point un élément dans le réseau doit prendre en compte ses voisins est assez difficile à répondre et finit souvent par des choix de conception.

La figure 5.7 présente les différents niveaux de conception du plan de connaissance :

- Niveau cognitif où sont déployés les modules d'apprentissage ;
- Niveau productif où se trouvent les ressources à gérer.

La relation du niveau cognitif au niveau productif est une relation de gestionnaire à agent. Le niveau cognitif ayant une vision à plus long terme concentre une partie non négligeable de l'intelligence du système. La relation de coopération entre les différents niveaux cognitifs, par contre, peut suivre plusieurs patterns :

- Diffusion/Inondation (Broadcast) : Il consiste à diffuser aveuglément l'information de sorte à ce qu'elle arrive à un nombre maximal de nœuds dans le

réseau. Ce type de communication n'est pas souhaitable mais il a l'avantage d'assurer que tout les nœuds qui sont susceptibles d'être intéressés par la connaissance seront atteints tant que le réseau reste connexe.

- Souscription/Notification (Publish/Subscribe) : C'est le système de fonctionnement des annuaires. Chaque élément souscrit à tous ses voisins possédant des connaissances susceptibles de l'intéresser. Chaque élément notifie à ses abonnés les nouveautés de sa base. Ce paradigme inclut un problème de passage à l'échelle en terme de complexité en mémoire : chaque élément doit maintenir une base des abonnés à ses connaissances. Cependant, il ressemble beaucoup à la communication *multicast* et il est moins intrusif que la diffusion. Un autre défaut de ce paradigme est que pour l'automatiser, il peut nécessiter une phase d'identification des sources intéressantes pour chaque nœud.
- Client/Serveur où connaissance à la demande : Les nœuds demandent les connaissances aux voisins que les intéressent. Ce paradigme génère moins de messages que les deux premiers mais il reste le problème de la détection du moment où il faut faire une requête et de l'identification des destinataires les plus pertinents.
- Pair-à-Pair : Dans ce paradigme, tous les nœuds sont en même temps clients et serveurs. Ce système est très robuste face aux pannes et son approche totalement distribuée permet de diminuer les dépendances entre les nœuds. C'est l'approche qui nous parait la plus intéressante.

Nous proposons un algorithme de propagation dont le but est d'accélérer le processus d'apprentissage. Son principe général est de construire des vues situées sous forme d'arbres recouvrants minimisant ainsi le nombre de messages envoyés. Il a des propriétés des réseaux Pair-a-Pair dans la mesure ou chaque nœud peut être un relai pour les autres nœuds. La section suivante détaille le processus.

5.4.2 Processus de propagation de connaissances

Les différents modules d'apprentissage du plan de connaissance collaborent en partageant des stratégies. Le but est d'accélérer l'apprentissage globalement en faisant dériver les connaissances des nœuds qui sont dans des zones très dynamiques (produisent plus d'observations variées) vers les zones les moins dynamiques. Le

processus de l'échange est décrit par l'algorithme 4, proche dans son principe de celui proposé par [Bourgne *et al.* 2007].

Du fait que les nœuds collaborent, il y a une difficulté supplémentaire de gestion de la cohérence au niveau local c'est pourquoi nous proposons de séparer la connaissance d'un nœud en deux bases :

- B_L qui contient les stratégies apprises localement et donc inclue B_C et les exemples positifs et négatifs ;
- B_S qui est la base des stratégies reçues des voisins et en attente de label.

Ainsi, le nœud donc, au moment de choisir une action, pourra utiliser les stratégies dans la base B_S pour les états pour lesquels il n'a pas encore de stratégie efficace.

La collaboration au niveau de l'apprentissage repose sur trois procédures correspondant aux processus exécutés après différents types d'évènements.

Tout nœud qui apprend une nouvelle stratégie, la propage à ses voisins proches (PROCEDURE 1). Les nœuds qui reçoivent cette stratégie l'acceptent ou la rejettent après avoir vérifié un certain nombre de conditions (PROCEDURE 2).

Un nœud accepte une connaissance lorsque :

- La connaissance n'existe pas déjà dans l'une des bases de connaissances locales (B_L et B_S). Si elle est présente, la connaissance est ignorée (ligne 11) pour assurer le fait qu'une connaissance ne soit traitée qu'une seule fois durant l'échange et donc éviter les boucles.
- Elle n'est pas en contradiction (ligne 13) avec la connaissance apprise localement (connaissance dans B_L). Il y a contradiction lorsque la stratégie fournie contient une action apprise localement comme étant inefficace (élément de E^-). Dans ce cas la connaissance locale est préférée à la connaissance reçue qui a plus de chance de correspondre au contexte local. La stratégie est alors rejetée sans qu'il y ait de mises à jour locales. Dans le cas d'un rejet, les nœuds critiques à distance 1 (les voisins directs) envoient une observation négative o^- qui modifiera la valeur θ_t d'un exemple du nœud initiateur comme s'il avait lui-même fait cette observation.
- La connaissance est compatible (ligne 14) avec le nœud local, c'est-à-dire que toutes les actions dans la stratégie peuvent être exécutées sur l'équipement géré. En cas de contradiction, la connaissance est rejetée car ne pouvant déci-

Algorithme 4 : Algorithme d'échange de la connaissance

Données :

id_i : Identifiant du nœud initiateur de l'échange

id_{local} : Identifiant du nœud local

S : Stratégie

PROCEDURE 1 Initiation de la propagation
début
  A l'induction d'une stratégie S
  Envoyer $< id_i, S >$ à tous les voisins
fin
PROCEDURE 2 Traitement à la réception d'une stratégie
début
  A la réception de $< id_i, S >$
  si $id_i \neq id_{local}$ **alors**
    si $S \in (B_S \cup B_L)$ **alors**
      Ignorer S
    sinon
      si $S \cup B_L$ *est cohérent* **alors**
        si S *est compatible avec* B_L **alors**
          $B_S \leftarrow B_S \cup S$
          Envoyer $<id_i, S>$ à tous les voisins sauf à celui d'où vient le message
        fin
      sinon
        si id_i *est un voisin direct* **alors**
          /* o^- *est une observation négative* */
          Envoyer $<o^-, id_i, S>$ au voisin id_i
        fin
      fin
    fin
  fin
fin
PROCEDURE 3 Réception d'une critique (observation négative)
début
  A la réception de $<o^-, id_i,$ S> recalculer les θ_t en incluant o^-
fin

der si la pertinence de la stratégie ne provient pas en grande partie de l'action impossible à exécuter localement.

La stratégie est ainsi propagée, de proche en proche, dans le réseau tant qu'elle est considérée comme acceptable par les nœuds intermédiaires. L'intérêt pour le nœud initiateur est de recevoir des critiques (observations négatives) lorsque sa connaissance n'est pas acceptée par des voisins (PROCEDURE 3). Nous avons choisi de limiter les nœuds pouvant émettre des critiques aux voisins directs (les nœuds à distance 1, où la distance entre les nœuds est la longueur du chemin minimal de l'un à l'autre). Cependant la portée, c'est-à-dire la distance maximale entre un initiateur, et un voisin critique, pourrait bien constituer un paramètre de la méthode, même si nous considérons que plus on s'éloigne, moins les critiques deviennent pertinentes, le contexte ayant de fortes chances d'être différent.

La figure 5.8 représente une simulation de l'algorithme avec *Visidia* [VISIDIA 2008]. *Visidia* est un outil de visualisation d'algorithmes distribués. Il permet ainsi d'avoir un support visuel du comportement de l'algorithme. Le cas que nous avons testé consiste en un ensemble de nœud qui tirent chacun, au hasard un booléen pour représenter la fonction d'acceptation. Cette simulation a permis d'expliciter le fonctionnement de l'algorithme. L'arbre recouvrant à partir du point de départ de la connaissance est retrouvée de manière dynamique et n'est pas maintenue car il se forme au fil de l'eau.

L'algorithme de propagation se termine au bout d'un temps fini en considérant que le nombre de nœud est fini. Lors de la propagation d'une stratégie, chaque nœud qui la reçoit ne la retransmet qu'au plus une seule fois car il y a une construction à la volée d'un arbre recouvrant. En effet, si une stratégie a déjà été traitée par un nœud, soit la stratégie a été retransmise auquel cas il y a une copie gardée localement, soit elle est refusée et donc il n'y a pas de retransmission de la connaissance. Dans le premier cas (copie locale) le test à la ligne 10 de l'algorithme 4 empêche un nouveau traitement.

Sachant cette propriété, le pire des cas est d'avoir la connaissance acceptée par tous les nœuds. Deux cas peuvent se présenter alors :

– Le graphe des nœuds ne contient pas de circuit, alors nous sommes en présence d'un arbre sur lequel chaque arc est traversé par la stratégie exactement une fois : Donc nous avons $n-1$ messages envoyés, n étant la taille du réseau.

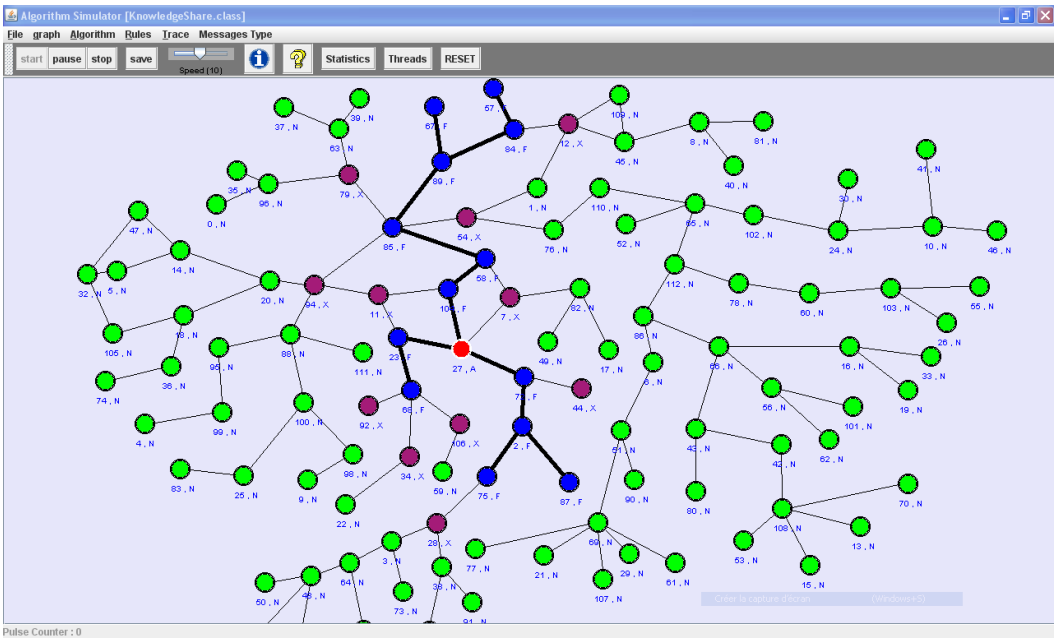

FIGURE 5.8 – Exécution de l'algorithme d'échange de connaissance avec Visidia

Cette complexité en nombre de message correspond à celle de l'inondation dans un arbre recouvrant.

– Si le graphe contient au moins un circuit alors un message traverse au plus deux fois la même arrête. Une situation simplifiée avec un circuit de longueur 3 est décrite par la figure 5.9 pour illustrer le cas avec présence de circuit. Le premier nœud qui reçoit la stratégie dans le circuit l'envoie aux autres qui lorsqu'il l'ont reçu, se le renvoient les uns aux autres. Chaque nœud ayant déjà traité la stratégie, elle n'est pas retransmise une deuxième fois. Un circuit plus long ne change pas cette valeur. Le pire des cas est celui d'un graphe complet, en utilisant la relation entre le nombre de liens et le nombre de nœuds dans un graphe complet, nous pouvons en déduire que la propagation coûte au pire $n(n-1)$) messages, ce qui donne une complexité de $O(n^2)$, n étant le nombre de nœuds S dans le graphe.

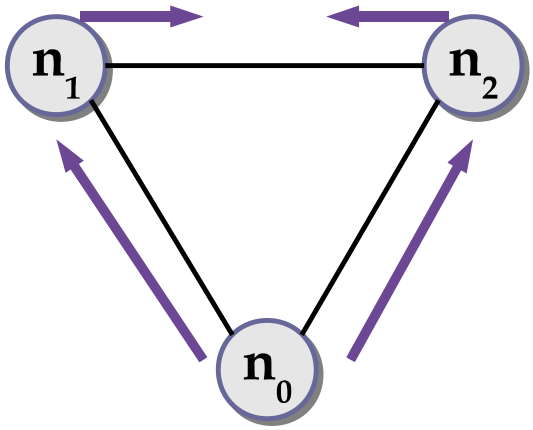

FIGURE 5.9 – Propagation dans un circuit de longueur 3

5.4.3 Voisinage et gestion de l'hétérogénéité

La notion de voisinage occupe une importance de premier plan sur l'algorithme de propagation de la connaissance. Cependant, il y a deux types de voisinage qui ont un impact direct sur le nombre de messages nécessaires pour la propagation d'une connaissance : voisinage logique et voisinage physique.

La notion de voisinage physique est celle qui est connue traditionnellement dans le monde des réseaux, avec comme critère, l'accessibilité directe par rapport au lien. Les algorithmes de routage traditionnel tels que RIP ou OSPF exploitent la topologie avec ce critère. Ce type de voisinage a un avantage considérable qui est de correspondre à la topologie réelle du réseau. Si les éléments qui se partagent la connaissance sont des voisins directs, alors nous avons la garantie que la propagation a une complexité en nombre de messages de l'ordre de $O(n)$, n étant la taille du réseau.

Cependant, par extension, la notion de voisinage logique commence à se développer avec les réseaux *overlay*. Dans ce cas, le critère de voisinage n'est pas seulement une accessibilité par rapport au lien mais un critère partagé par les différents éléments comme le type de nœud, une fonctionnalité disponible. La complexité en nombre de message de la propagation inclus aussi la complexité en messages du routage dans le réseau réel. Il est intéressant de voir que le voisinage physique est un cas particulier du voisinage logique.

La figure 5.10 résume ces deux visions. Le nœud a quatre voisins (gris) physiques

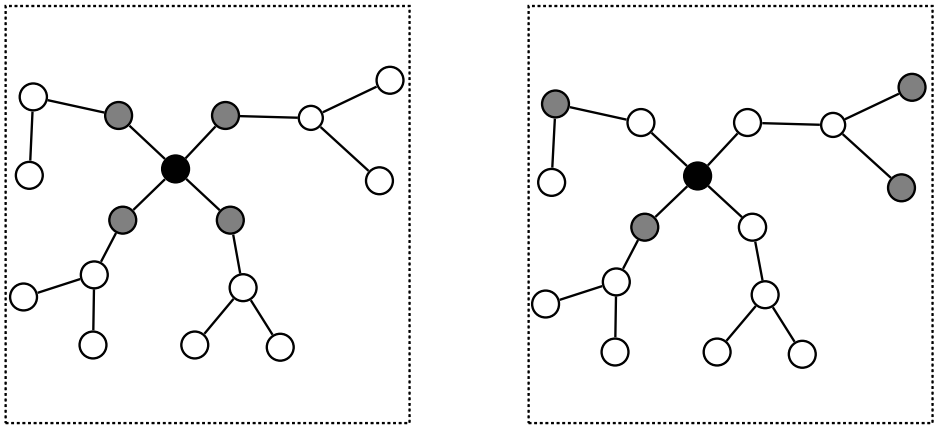

FIGURE 5.10 – Deux types de voisinage : a- Voisinage physique, b- Voisinage logique (Overlay)

dans le cas a) logiques dans le cas b).

5.5 Conclusion

Dans ce chapitre, nous avons détaillé notre approche de module d'apprentissage pour l'auto-adaptation.

Cette approche se base sur l'utilisation des transitions observées pour calculer l'efficacité des actions sur les états du système. Nous avons montrés qu'une telle utilisation pouvait se justifier analytiquement grâce à la relation qu'il y a entre les probabilités de réussite des actions et l'historique des transitions observée. Les stratégies efficaces sont ensuite pour compléter ou trouver des alternatives pour le couples états-actions non encore testé en prenant soin d'écarter les stratégies inefficaces. Cette opération est effectuée à l'aide de la programmation logique inductive.

Ceci justifie une introduction de la programmation logique inductive, un paradigme entre la programmation logique et l'apprentissage inductif de concept, afin de montrer comment il nous aide à avoir cette caractéristique. Cette apprentissage par renforcement a trois données en entrées : une connaissance de base B qui décrit la théorie du domaine (ce qui ne bouge pas), un ensemble d'exemples positifs décrivant le concept appris (ici les stratégies efficaces), d'exemples négatifs ne décrivant pas le concepts. La PLI apprends des règles générales régissant les stratégies. Cela veut dire que dans le cadre de l'approche que nous proposons, il s'agit de trouver un ensemble de règles qui décrivent les stratégies déjà connus comme étant efficace tout en évitant les stratégies qui sont inefficaces. Cela permet d'avoir un traitement des nouvelles situations en se basant sur les résultats actuels de l'apprentissage, les théories qui les sous-tendent et la déduction automatique.

Pour accélérer le processus, nous introduisons la collaboration entre les éléments réseaux effectuant l'auto-adaptation. Ils échangent leurs connaissances dans le but d'accélérer chacun de son coté l'apprentissage local, mais aussi effectuer une validation distribuée de la connaissance à travers les critiques mutuelles et des échanges d'observations. Cette collaboration se fait en utilisant la notion de "vue située", sous un arbre recouvrant minimisant le nombre de paquets.

Le prochain chapitre 6 présente un certain nombre de cas d'utilisation permettant de mettre en œuvre et de tester la performance de la méthode proposée.

Troisième partie

Implémentation et évaluation

CHAPITRE 6

Application dans le cadre des réseaux IP nouvelle génération

Sommaire

6.1 Introduction

Les chapitres 4 et 5 ont permis de présenter l'approche que nous avons proposée pour l'auto-adaptation. L'approche se base principalement sur l'utilisation de l'historique des transitions pour trouver un ensemble d'actions qui permettent de faire une adaptation efficace suite aux changements de l'environnement. Cette approche, présentée dans un cadre général, permet d'avoir des domaines d'applications variés tant qu'il est possible de ramener le problème à une modélisation état/action.

Dans ce chapitre, nous allons présenter des applications de notre approche dans les environnements IP de nouvelle génération.

Dans un premier temps, nous présentons un cas d'application sur *DiffServ* qui est une architecture de QoS dans l'environnement IP. Cette application est détaillée pour mettre en évidence les aspects de l'approche. Ensuite, nous allons montrer dans le cadre des applications multimédia sur les réseaux sans-fils 802.11, comment l'approche pourrait être transposée pour optimiser un ensemble d'adaptations définies pour améliorer la qualité du contenu multimédia transmis. Enfin, nous allons montrer toujours dans le cadre des applications multimédia, comment un mécanisme de prédiction permet d'améliorer l'*overhead* de certains mécanismes d'adaptations comme la FEC (Forward Error Correction).

Le premier cas d'étude nous a permis de valider notre approche par simulations.

6.2 Simulation de la fonction d'auto-adaptation

La simulation de la fonction d'auto-adaptation dans les environnements réseaux est également encore un défi car les simulateurs existants ne fournissent pas toutes

les fonctionnalités qu'il faudrait pour pouvoir faire certaines manipulations de reconfiguration durant la simulation.

Nous allons discuter ce point avant de présenter les cadres d'applicatifs.

6.2.1 Quelques Simulateurs

Les simulateurs actuellement les plus importants pour l'environnement réseau sont les suivants :

JSIM [J-SIM 2006] est un simulateur dont la conception est basée sur le modèle composant et qui est codé en JAVA. Un élément autonome est l'entité de base sur laquelle les composants de JSIM sont construits. Ce simulateur offre des fonctionnalités très intéressantes pour la simulation réseau mais aussi de par sa conception facilement extensible.

NS2 [NS2 2009] est un simulateur à événement discret écrit en C++. Il est sans doute l'un des simulateurs les plus utilisés dans le monde de la recherche car il offre un très large support de protocoles réseaux, sur réseaux filaires et/ou sans fils. Le projet JNS a été réalisé pour faire une version Java de NS2 mais le projet semble être arrêté par manque de membres actifs suffisants.

OMNET++ [OMNETPP 2009] est un environnement de simulation écrit en C++ avec un kernel facilement extensible pouvant être embarqué dans d'autres applications. Il fournit aussi un environnement de simulation réseau très riche. Il peut, même, être utilisé dans des environnements autres que les réseaux, tel que les architectures matérielles, les *business process* ...

OPNET [OPNET 2009] est un outil professionnel très complet mais, malheureusement, il n'est ni gratuit ni *opensource*. Pour cette raison, nous l'écarterons de cette comparaison.

Ces simulateurs sont connus traditionnellement pour la simulation à l'aide de langages de script pour décrire les scénarii. Une nouvelle génération de simulateurs a fait cependant son apparition en définissant les scénarii par programmation. Ces simulateurs s'inspirent en partie d'OPNET.

GTNets [GTNetS 2008] est un simulateur développé à l'université de Geogia Tech. Il est écrit en C++ et sa conception s'appuie sur un découpage par niveau

entre les différentes couches du modèle TCP/IP. Une déclinaison de ce simulateur est dédiée aux réseaux de capteurs : GTsNets [1]. Cette différentiation permet d'avoir une flexibilité accrue du simulateur. Son point faible est qu'il n'a pas une bibliothèque aussi fournie que celle de NS2

NS3 [NS3 2009] est une tentative de refaire NS2 avec davantage de réalisme, la possibilité d'utiliser les socket Bekerley dans la simulation (ce qui facilite l'interfaçage avec un autre outil, et la simulation distribuée). Il souffre cependant de sa jeunesse qui fait que sa bibliothèque (qui ne cesse de s'enrichir) reste encore limitée. Ce simulateur nous parait particulièrement prometteur, vu son roadmap.

Le choix du simulateur peut paraitre assez compliqué et surtout dépendre de ce que l'on cherche à faire précisément. Nous avons identifié un certain nombre de critères qui nous ont parus importants pour la simulation de la fonction d'auto-adaptation. Le tableau 6.1 résume les aptitudes des simulateurs par rapport aux critères que nous avons retenus.

1. Les deux simulateurs ont le même site officiel

Critère	possibilité de fermer la boucle	disponibilité des données de performances	extensibilité	popularité
NS2	*	*	***	*****
NS3	**	*	***	*
JSIM	****	***	****	**
GTNets	***	**	***	*
OMNet++	?	?	**	**

TABLE 6.1 – Récapitulatif des simulateurs

6.2.2 Critères de choix

Le choix entre ces logiciels pour simuler un comportement autonome n'est pas une mince affaire. Chacun offre une partie des éléments dont nous avons besoin pour réaliser ce comportement. Cependant, nous pouvons poser un certain nombre de critères pour justifier notre choix.

- *La possibilité de fermer la boucle* : C'est le critère le plus important pour la sélection finale. En effet, il faut que le simulateur donne la possibilité d'obtenir un maximum d'informations de performance et permettre d'effectuer les configurations durant la simulation. Ceci n'est pas très courant car, dans la plupart des cas, ces informations sont calculées après simulation. La possibilité de faire des configurations durant la simulation rend la simulation totalement aléatoire dans certains cas que nous avons testés. Il faudrait, alors, qu'un simulateur avec un *scheduler* d'événements (dans le cas des simulateurs à événement discret comme NS2 et JSIM) puisse prendre en compte les reconfigurations durant la simulation sans donner de résultats erronés.
  Ce critère défavorise NS2 car il est codé avec deux langages OTCL/C++ avec des notions d'agent, nœud, liens, ... qui sont codés en dur et la topologie est gérée par le scheduler d'événements. Si on veut avoir la possibilité de modifier la topologie à la volée ou changer certaines valeurs de paramètres de configuration, les modifications à faire sur le simulateur sont assez profondes. D'ailleurs, c'est pour cette raison que NS3 s'est défait de la contrainte d'avoir deux langages. Au contraire, JSIM est avantagé sur ce point par rapport aux autres car tous les éléments ont la même structure : composant. Ils communiquent à travers des ports. L'environnement de simulation peut se construire à la volée ou avant le début des simulations. Cependant, quelques incohérences peuvent apparaitre lors d'opérations de configuration.
- *La disponibilité d'un grand nombre de données de performances* (telle que la bande passante restante, les taux de perte, ...) durant la simulation. Suivant le problème d'auto-configuration, il peut y avoir un besoin d'une valeur en particulier pour la définition des états du système et la prise de décision des configurations sans avoir à tout recoder à la main. Avec NS2, presque toutes les informations de performance sont disponibles à la fin de la simulation ou

sont parfois à calculer à la main. JSIM dispose, quant à lui, de composants qui calculent certaines informations telles que le taux de perte instantané et qui peuvent être récupérées durant la simulation en se connectant seulement sur un port.

- *L'extensibilité* : La plupart des simulateurs cités plus haut sont assez extensibles mais la différence va se faire en fonction du niveau de difficulté d'étendre le système. JSIM dans ce cas précis, nous semble nettement plus facile à modifier, l'ajout de composants pouvant se faire rapidement sans avoir à connaitre l'organisation interne dans les détails. Il suffit juste de connaitre la structure d'un composant. Pour OMNET++, l'ajout de nouveaux composants est simple également mais la prise en main de la structure est assez complexe. Pour NS2, la tâche est un peu plus ardue à cause des deux langages. L'extensibilité aussi inclut la possibilité de faire inter-agir le simulateur avec un autre système et de prendre en compte les temps de calcul induit par cette interaction. Nous sommes dans ce cas car nous avons un système de programmation logique inductive qui doit inter-agir avec le simulateur.
- *La popularité* : L'intérêt de ce critère est uniquement de garantir la maintenance et l'enrichissement de la communauté. Ce critère, d'ailleurs, fait la force des simulateurs opensource par rapport aux simulateurs non libres. NS2 est le plus populaire de tous, mais commence à montrer certaines de ces limites. GTNetS souffre de ce critère, il n'a pas été très souvent mis à jour depuis sa première distribution.

6.2.3 Contraintes d'implémentation

A coté de ces critères nous avons un certain nombre de contraintes à prendre en compte qui sont propres à notre approche. Le module d'apprentissage que nous proposons contient un module tiers qui prend en charge le processus de programmation logique inductive. Il faudrait un simulateur qui supporte bien l'interfacage avec un logiciel tiers. Comme indiqué plus loin, nous avons choisi Aleph [Srinivasan 2002] comme environnement de développement. Or, Aleph est un environnement de programmation PLI exécuté sur un interpréteur Prolog. Nous utilisons swi-prolog qui offre une interface de programmation C++ facile à utiliser. Quasiment tous les si-

mulateurs peuvent l'exploiter mais il reste à ajouter la prise en compte de cette interaction dans le temps de simulation. En effet, dans la plupart des cas de simulations à temps discret, les temps de simulation sont calculés avant le lancement par un scheduler. Surtout lorsque le simulateur utilise un compteur pour exécuter les événements. NS2 s'est révélé particulièrement sensible à la modification d'un tel élément. Le problème a été plus facilement contourné sur JSIM, par l'ajout de nouveaux événements dans le vecteur d'évènements sans que cela ne perturbe le fonctionnement du simulateur.

Un autre aspect qu'il est nécessaire de considérer pour simuler l'approche que nous proposons est la possibilité de reconfigurer comme nous l'avons dit sans que le simulateur ne soit déstabilisé. Aussi, faut-il avoir la possibilité de reconfigurer durant la simulation des éléments. Par exemple, sur NS2 une reconfiguration des routeurs de cœur dans l'environnement DiffServ (voir section 6.3) durant la simulation est assez complexe à faire et la cohérence des résultats n'est pas garantie.

Nous avons choisi, sans prétentions d'avoir testé à fond tous les simulateurs, JSIM après l'évaluation de ces critères bien qu'il soit moins populaire que NS2. JSIM utilise la notion de composants et de ports. La méthode la plus naturelle pour faire de l'auto-adaptation est de mettre tous les composants et de faire des compositions à la volée en faisant des branchements sur les ports. Cependant, il nous a fallut faire des modifications (2300 lignes de code environ) qui permettent de stabiliser le comportement du simulateur dans le cadre de l'auto-adaptation, au moins pour les simulations DiffServ.

Nous sommes aussi passé par JNI (Java Native Interface) pour pouvoir utiliser l'API de Prolog (et donc Aleph) sur JSIM.

6.3 DiffServ

Le fonctionnement de base du protocole IP est le Best-Effort, c'est-à-dire la transmission au mieux des paquets. En cas de compétition pour l'accès aux ressources du réseau, le traitement des paquets est effectué sans "*distinction*" ou favoritisme.

Avec l'apparition de nouvelles applications TCP/IP (vidéo haute qualité, téléphonie sur IP, ...) plus ou moins sensibles aux caractéristiques de transmission du

réseau, une nette différenciation entre les applications réseaux en terme de besoins critiques de bande passante, de latence, de délai de propagation, c'est-à-dire de manière générale, de qualité de service est très vite apparue.

Ce besoin a motivé l'enrichissement de l'infrastructure TCP/IP avec l'ajout de mécanismes permettant de fournir la qualité de service (QoS). Deux approches ont été proposées : *IntSerV* et *DiffServ*. La première utilise le protocole RSVP (ReSerVation Protocol) pour la réservation de ressources. Cependant, *IntServ* souffre d'un problème de passage à l'échelle. Au contraire, la seconde approche, *DiffServ*, plus simple et offrant la possibilité de passer à l'échelle semble être la solution la mieux adaptée pour fournir de la QoS dans les réseaux IP.

En plus de doter les réseaux IP de mécanismes de QoS, il est aussi indispensable de bien les gérer pour une utilisation optimale des ressources.

Dans cette partie nous allons présenter l'architecture DiffServ telle qu'elle a été définie par l'IETF ainsi que la gestion par politique qui est la méthode de configuration privilégiée dans ce contexte. Par la suite, nous montrerons comment notre approche pourrait être appliquée dans ce contexte pour améliorer le fonctionnement d'un équipement *DiffServ*.

6.3.1 Architecture

L'architecture *DiffServ* a été proposée par le groupe de travail du même nom qui s'est constitué en 1998. Ce groupe de travail a été formé avec comme objectif de produire une architecture de QoS simple qui peut être applicable à IPV4 et IPV6. Dans cette vision, l'identification des micro-flux et l'utilisation de mécanisme de signalisation sont exclues pour éviter les problème de *IntServ*. Nous rappelons qu'un micro-flux signifie ici un trafic IP avec un même quintuplet : adresses source et destination, protocole, ports source et destination.

Le modèle *DiffServ* est basé sur la définition de classes de trafic avec différents niveaux de services garantis. Le regroupement des flux en classes en s'appuyant sur l'information qui est codée sur l'octet du champ ToS (*Type of Service* [Institute 1981] de l'entête IPv4. Cette information est appelée *DiffServ Codepoint* ou *DSCP* [Nichols *et al.* 1998] et permet d'identifier le type de service.

La spécification d'IPv4 (RFC 791) a défini le champ ToS avec la structure sui-

vante :

- Les trois bits les plus significatifs représentent la *précédence* (une indication à une priorité relative du paquet)
- Les trois bits suivants définissent les besoins en terme de délai, de débit et de fiabilité des paquets
- Les deux derniers bits sont mis à zéro et leur rôle n'a pas été déterminé

Dans la spécification de la version 6 du protocole IP (RFC 2460), ce même champs est redéfini comme le champs *Traffic Class* mais sa structure n'est pas déterminée.

L'architecture *DiffServ* a redéfinit, à son tour, ce champs en prenant les six bits les plus significatifs pour représenter le DSCP. Un nœud *DiffServ* détermine le service associé à un paquet en fonction de la valeur de ce nouveau champs. Un ensemble de paquets partageant le même DSCP et traversant le réseau dans le même sens constituent un *Behavior Aggregate* ou BA. Une classe de trafic est un ensemble de BA, elle peut aussi être éventuellement composée d'un seul BA.

DiffServ définit une architecture qui va d'un équipement seul, à un réseau, jusqu'à un groupe de réseaux. Un ensemble de nœuds avec une implémentation commune de *DiffServ* forment un domaine *DiffServ*. Ces nœuds sont typiquement sous le même contrôle administratif. Un ensemble de domaines *DiffServ* contigüs forment une région *DiffServ*. Les domaines dans une région peuvent avoir des politiques et des mécanismes de marquage différents. Dans ce cas, les *domaines* doivent se mettre d'accord pour spécifier comment les trafics sont pris en charge quand ils traversent les domaines.

Un domaine *DiffServ* est constitué de deux types de nœuds [Blake *et al.* 1998] (figure 6.1) :

- les nœuds aux extrémités (*edge routers ou border nodes*) qui relient le domaine aux autres domaines *DiffServ* ou aux autres domaines non *DiffServ*. Ils s'assurent que les trafics entrant dans le domaine sont convenablement marqués, classés dans les limites du contrat de service. Ces opérations sont appelées classification de trafic (T*raffic Classification*) et conditionnement (*Traffic conditionning*).
- les nœuds à l'intérieur (*core routers ou Interior nodes*) du réseau *DiffServ* dont le travail se résume essentiellement à la retransmission des paquets sui-

vant le *Peer-Hop Behaviour* (PHB) qui est associé au DSCP du paquet. Le PHB définit le traitement à effectuer sur un paquet avant de le retransmettre.

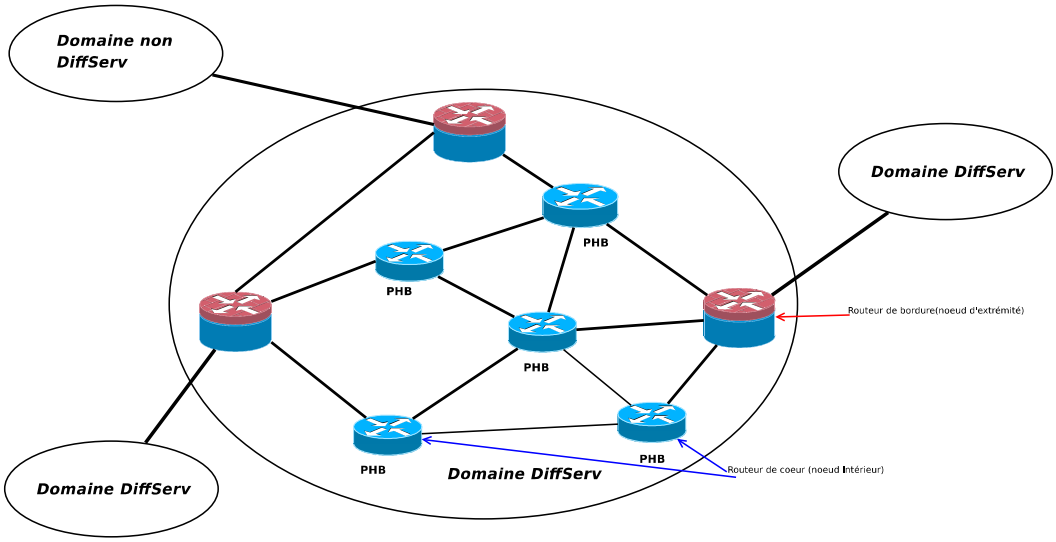

FIGURE 6.1 – Modèle de réseau DiffServ

Les nœuds de bordure et les nœuds du cœur du réseau implémentent des politiques locales qui permettent d'offrir différents niveaux de services à chaque BA : ces politiques décrivent le *Per-Hop behavior*.

L'IETF a défini un certain nombre de PHB standards :

- *Class-Selector* (CS-PHB) [Nichols *et al.* 1998] : Ce PHB a été défini pour une compatibilité avec la signification passée de la Précédence du champs ToS de l'entête IP. Il maintient le même ordre relatif que le champ précédence. Une haute valeur du Class-Selector représente une forte probabilité de retransmission. Pour ce PHB, il n'y a pas de caractérisation selon la latence, la gigue, ou le taux de pertes.
- *Expected Forwarding* (EF-DS) [Jacobson *et al.* 1999] : Ce PHB est aussi connu sous le nom de *Service Premium* et a la priorité maximale. Il définit une faible latence, une faible gigue et un faible taux de perte que le nœud *DiffServ* doit implémenter. Ce PHB est adapté pour le transport de trafic temps réel à travers un domaine *DiffServ*. Un nœud ne devrait pratiquer aucune opération de ré-ordonnancement sur un flux EF. Les RFC 3246 et RFC 3247 spécifient les contraintes et méthodes de réalisation d'un tel service.

- *Assured Forwarding* (AF-PHB) [Heinanen *et al.* 1999] : La famille de groupes de PHB AF définit quatre niveaux de garantie de retransmission. Ce type de PHB permet à un nœud de supporter différents niveaux de garantie de taux de pertes. Les groupes sont AF1, AF2, AF3, AF4. Chaque groupe supporte trois niveaux de "drop-precedence". Plus la valeur du drop-precedence est grande, plus la probabilité que le paquet soit supprimé sera grande lorsque le groupe aura atteint les limites de la ressource accordée. Il n'y a pas de caractérisation implicite temporelle pour ce type de PHB (par rapport à la latence ou la gigue). Ce PHB est dédié aux clients qui ont besoin d'un service de retransmission plus fiable que le Best-Effort.
- *Default* (DS-FIELD) [Nichols *et al.* 1998] : Ce PHB correspond au Best-effort classique. Les paquets seront envoyés le plus vite possible et avec la plus grande quantité possible. Il n'y a aucune caractérisation par rapport à la latence.

Les trois derniers sont les plus utilisés dans le domaine de la qualité de service dans les réseaux IP. La figure 6.1 résume l'architecture d'un réseau *DiffServ*.

Une implémentation d'un PHB sur un nœud supportant *DiffServ* peut comporter un ensemble d'éléments. Ces éléments font l'objet d'un modèle proposé par l'IETF : DS-MODEL [Blake *et al.* 1998]. Le DS-MODEL présente ces éléments comme des blocs fonctionnels. Les principaux blocs sont, appelés [Blake *et al.* 1998] : Traffic Classification Elements (TCE) , Metering Functions, Actions of Marking, Dropper, Counting & Multiplexing, Queing. Dans chacun de ces blocs, un élément se charge de réaliser la tâche principale ; par exemple le Classifier pour le TCE, l'élément l'Absolute dropping pour le Dropper etc.

Des combinaisons forment un bloc de haut niveau qui définit la manière de traiter les flux, et le parcours des paquets dans les différents éléments. Ce bloc de haut niveau est connu sous l'appellation de Traffic Conditioning Block (TCB) ou bien Traffic Conditioner. La figure 6.2 donne un exemple de TCB.

Le DS-MODEL définit aussi la constitution logique d'un matériel supportant *DiffServ* : un ensemble de TCB, un composant de routage et un module de configuration et de gestion. Ces composants sont interconnectés comme le montre la figure 6.3.

Dans cet exemple, le classifier sélectionne les paquets suivant la valeur d'une par-

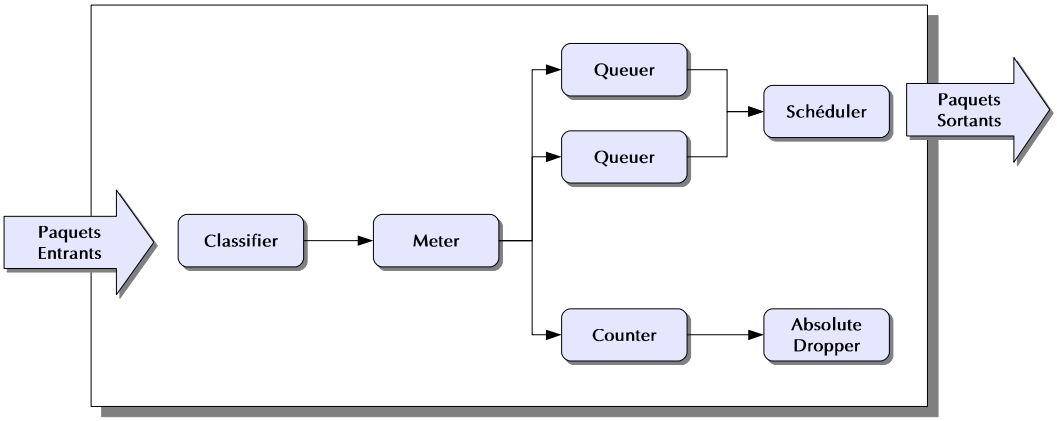

FIGURE 6.2 – Exemple de Traffic Conditioning Block

tie de l'entête IP. Deux types de classifiers ont été définis par l'IETF [Blake *et al.* 1998] : le BA (qui classe les paquets suivant le DSCP) et le MF (MultiField) qui utilise plusieurs champs comme critère de classification (e.g. Adresses source/destination).

Le Meter prend ensuite le relai pour mesurer les propriétés temporelles de la classe auquel appartient le paquet. Ceci afin de voir sa conformité au Traffic Conditioning Agreement. Les différents types de Meters sont : Average meter, Exponential Weighted Moving Average (EWMA) Meter,Two-Parameter Token Bucket Meter, Multi-Stage Token Bucket Meter, Null Meter [Blake *et al.* 1998]. Si le paquet est profil, il est envoyé à l'Absolute Dropper qui supprime les paquets lui arrivant en entrée. Le Counter juste avant la suppression permet de générer des statistiques sur les données supprimées par exemple.

Le paquet en conformité avec le profil est mis dans le Queuer qui correspond à sa classe. L'élément Queue est une structure de données qui permet de gérer la mise en tampon des données avant leur retransmission. Le Queuer est couplé avec un algorithme d'ordonnancement qui sélectionne les paquets qui doivent être envoyés à la sortie : Scheduler. Plusieurs types de Queuer sont utilisés, chacun ayant un objectif particulier comme FIFO (First In First Out), RED (Random Early Detection), SFQ (Stochastic Fair Queue).

La gestion et la configuration des éléments et des TCB d'un équipement supportant *DiffServ* se fait à l'aide d'une interface de configuration. Un protocole (COPS a été proposé à cet effet) est aussi utilisé pour cette tâche de configuration (voir figure 6.3).

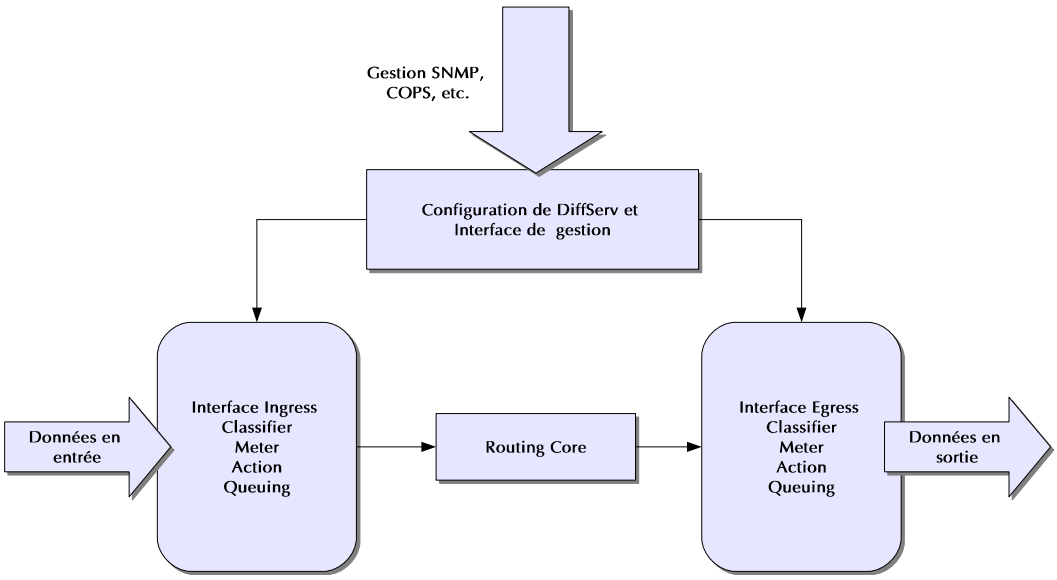

FIGURE 6.3 – Composants d'un équipement supportant DiffServ

6.3.2 Gestion des réseaux DiffServ

La gestion des équipements est un problème crucial dans le fonctionnement d'un réseau. En effet, le comportement d'un système *DiffServ* est étroitement lié à la manière avec laquelle les routeurs de cœur et de bordure sont configurés.

Les éléments sont gérés grâce à des règles du type "si condition alors action". Ces types de règles sont connus sous l'appellation de règles de politique. Un paradigme regroupe les notions de règles de politique et de gestionnaire de politique : la gestion par politique (Policy-based Management ou PBM).

De nombreuses recherches ont été menées pour simplifier et automatiser (exemple [Lymberopoulos 2002] et [N. Samaan 2005]) le processus de gestion des réseaux *DiffServ* grâce aux politiques. Un certain nombre de caractéristiques ont été identifiées pour un gestionnaire par politique :

- Il doit avoir assez de flexibilité pour faire abstraction (au maximum) des ressources matérielles qu'il gère. Ceci aura comme avantage de permettre un ajout ou un retrait d'un équipement du système avec un minimum de modifications du système global. Des frameworks ont ainsi été proposés.
- Il doit supporter le passage à l'échelle et pouvoir appliquer les règles de politique à un ensemble d'équipements plutôt que de manière individuelle.

- Le système doit pouvoir s'adapter à des événements tels que l'échec, la violation de garanties de qualité de service. Il adapte son comportement ou celui des équipements qu'il gère aux événements qui surviennent.
- Il doit permettre la spécification des politiques à plusieurs niveaux : application et réseau. Cette caractéristique est nécessaire car les éléments gérés peuvent être aussi bien des applications que des équipements réseau.

6.3.3 Vers un réseau DiffServ autonome

Les réseaux actuels constituant des environnements de plus en plus dynamiques, du point de vue de la gestion une caractéristique très importante est la capacité d'adaptation. En effet, une utilisation optimale passe par une adaptation du comportement des équipements face aux événements qui surviennent sur le réseau sous-jacent.

Trois méthodes sont utilisées pour atteindre cet objectif [N. Samaan 2005] : le gestionnaire par politique fait une adaptation en agissant sur les équipements par l'une des manières suivantes :

- Une adaptation par activation/désactivation de politiques prises dans un ensemble de règles de politique prédéfinies : cette méthode souffre d'un manque de passage à l'échelle.
- Une adaptation par changement dynamique des paramètres des politiques de QoS : cette méthode a le défaut de ne pas prendre en compte toutes les situations possibles dans un environnement fortement dynamique.
- l'adaptation par sélection ou assemblage de nouvelles politiques : c'est un assemblage basé sur l'apprentissage du comportement du système pour une meilleure configuration.

Dans un environnement *DiffServ* on peut voir, dans un premier temps que l'adaptation par l'apprentissage est très intéressant car elle peut permettre d'assembler des politiques pour configurer les équipements tout en respectant les contrats de services (SLA). Les configurations statiques ont un fort risque d'aboutir à des situations de suppression de paquets qui pourraient être évités par exemple. Il ne s'agit pas de dépassement de SLA, mais entre les flux AFxy, par exemple, une sur-allocation peut entrainer la suppression des paquets pour des trafics de drop-

précédence inférieure.

La gestion des équipements *DiffServ* implique la transformation d'un SLA en SLS qui se traduit ensuite sur le réseau en choix des éléments des TCB et des mécanismes qui implémentent les PHB. Une mauvaise traduction de ces éléments peut avoir comme effet des pertes qui pouvaient être évitées [Serban 2003] et ainsi une efficacité limitée de l'ensemble des règles. Nous proposons dans ce cadre l'utilisation de l'approche d'auto-adaptation que nous avons présenté dans le chapitre 5.

L'architecture *DiffServ* va ainsi se transformer à celle décrite sur la figure 6.4. Le gestionnaire autonome comporte ainsi tous les éléments qui permettent d'assembler les blocs fonctionnels et/ou de les configurer en fonction de ce qui se passe sur le réseau.

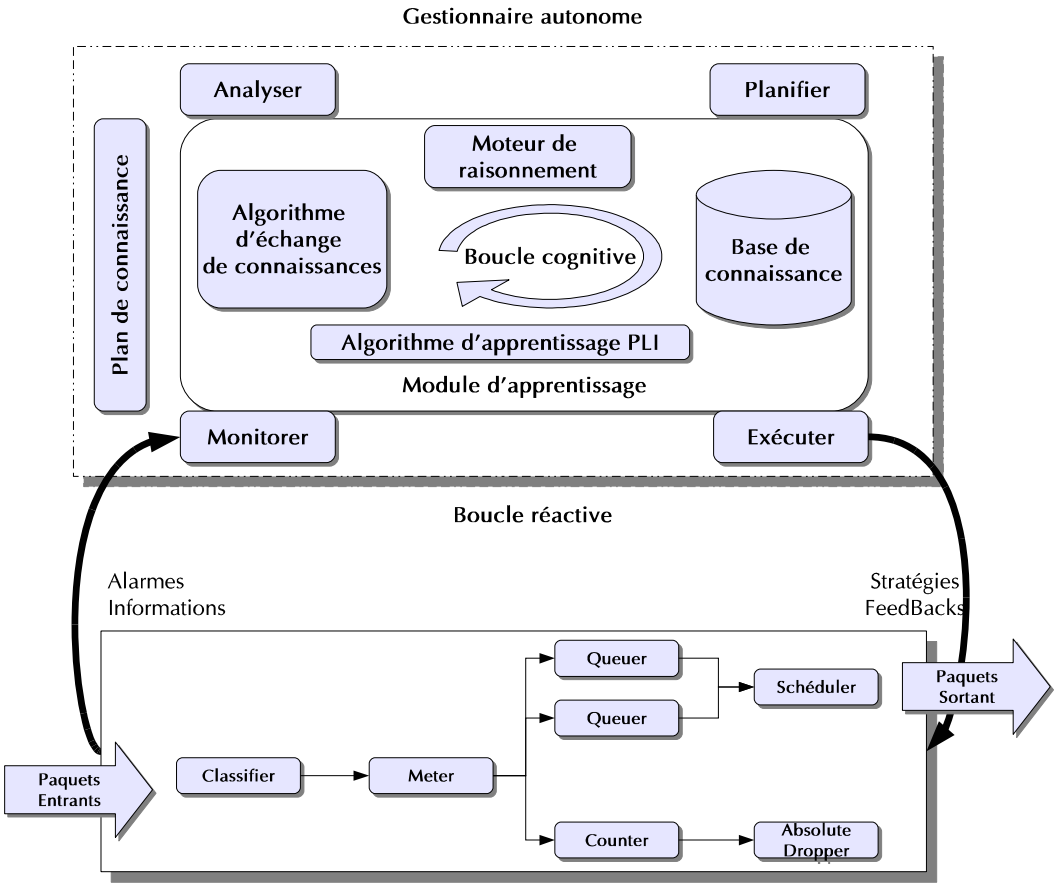

FIGURE 6.4 – Architecture DiffServ autonomique

Nous allons par la suite présenter un scénario de test qui permettra d'évaluer

les performances de notre approche dans ce cas bien précis.

6.3.4 Scénario et tests de performance

Nous avons retenu le contexte des réseaux DiffServ [Blake *et al.* 1998] comme cas d'étude pour l'implémentation et le test du plan de connaissance que nous proposons.

Pour transposer les données de la méthode que nous avons définie dans le chapitre 5, nous avons besoin de déterminer l'ensemble des états, celui des actions et le critère de performance.

6.3.4.1 Critère de performance

Le critère de performance est de répondre aux besoins des flux AFxy en minimisant leur taux de pertes dans le temps. En effet, les garanties apportées à cette classe sont faites en s'appuyant sur cette métrique. Cependant, minimiser les taux pertes ou le taux de suppression des paquets sur les nœuds DiffServ reviendrait à augmenter le taux de paquets arrivés à destination. Cette grandeur dépend en grande partie de la configuration des éléments du Trafic Control Block. Les possibilités de configuration sont assez important et il est difficile d'en extraire une qui soit adaptée quelque soit le type et la charge du trafic sur le réseau.

L'objectif, ici, sera donc de faire apprendre au module d'apprentissage les combinaisons du Trafic Conditionning Block et des Per-Hop-Behavior pour maximiser le débit réel de bout en bout sur les flux AFxy. Aucune opération ne sera effectuée sur les flux EF à cause de leurs garanties fortes, on se cantonnera au policing, c'est-à-dire à supprimer tout ce qui est hors profil.

6.3.4.2 Définition des états

Ne traitant que les flux AFxy, les taux de perte représentent un paramètre dominant sur les autres caractéristiques (délai, ...). Ceci nous permet, ici, de définir l'ensemble des états de manière linéaire.

Le tableau 6.5 décrite l'ensemble des états qui sont considérée

Le critère de performance qui en découle est celui de minimiser les taux de pertes (et implicitement d'augmenter le trafic transmis).

Etat	Condition
SousUtilseBW	RBW [a] (AF) $\leq$ 20% BW [b] (AF)
BesoinBW	RBW (AF)$\geq$ 80% BW (AF)
LegerCongestion	10% $<$ Perte [c](AF)$\leq$ 20%
Congestion	20%$<$Perte(AF)$\leq$ 60%
SevereCongestion	60% $<$ Perte(AF)

a. Bande passante réelle utilisée
b. Bande passante reservée
c. Pertes actuelles

FIGURE 6.5 – Etats du réseau.

6.3.4.3 Définition des actions

Pour la définition des actions, nous avons fait quelques tests pour déterminer les actions qui ont une influence sur les taux de perte des flux. Nous précisons, ici, que le taux de pertes que l'on essaie de minimiser concerne les paquets supprimés pour des raisons de précédence.

Le premier choix se tourne sur les éléments du TCB : les schedulers, les gestionnaires de file d'attente, ... Les actions locales que peut faire un nœud et qui ont véritablement une influence sur les taux de perte sont le changement de queuer (FIFO, DT et RED) et la manipulation de la taille (CTB) des buffers dans les différents nœuds intermédiaires.

Le gestionnaire de queuer FIFO est connu pour sa rapidité mais n'est pas très optimisé quand le trafic commence à congestionner le réseau.

Le DT ou DropTail a un comportement similaire il constitue une petite variante de FIFO dans notre environnement de simulation.

RED est un gestionnaire de Queue qui fonctionne avec des niveaux de seuils pour prévenir et contenir les gestions.

Pour ce qui est de la modification de la taille des files d'attente, nous avons réalisé un test pour montrer comment cette action influe sur les taux de pertes.

Pour mesurer l'impact de la manipulation des files d'attente sur les pertes sur un nœud donné, nous mettons en entrée un flux *Poissonien* décrit par la figure 6.6 qui est celui que nous allons utiliser dans les tests. Nous effectuons les tests avec

plusieurs tailles de buffers entre $5Ko$ et $3Mo$.

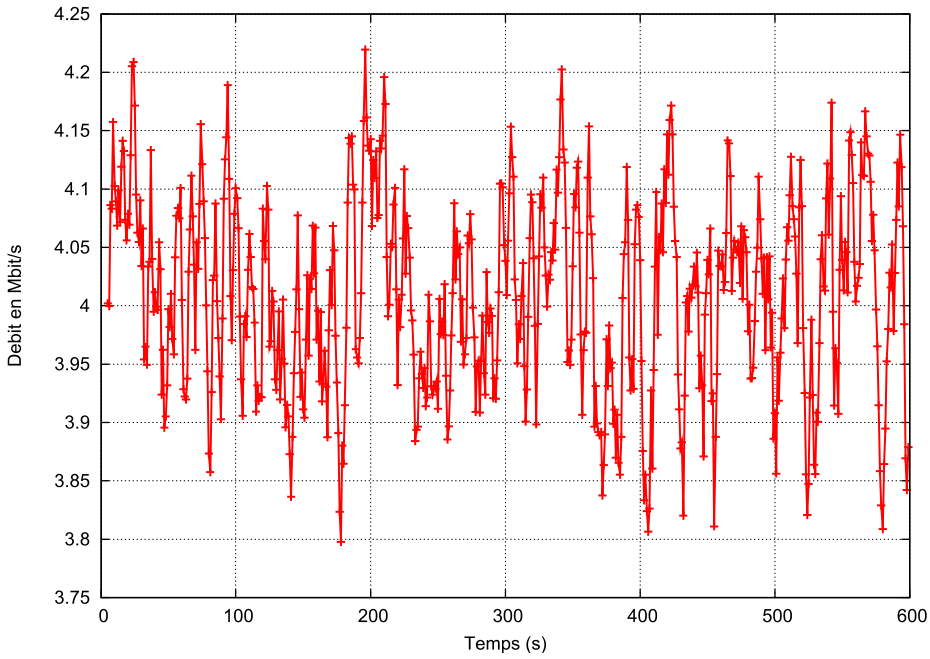

FIGURE 6.6 – Trafic Poissonien en entrée avec taille variable de paquets. Taille moyenne des paquets = 512o, Débit moyen 4Mbps

La figure 6.7 montre l'impact de la manipulation de la taille des files d'attente sur les pertes cumulées [2] du nœud local et le délai supplémentaire de traversée du nœud. Elle permettent de montrer que plus la longueur des files d'attente est importante, plus elle permet de retarder l'apparition des pertes, ce à quoi on pouvait s'attendre intuitivement. Cependant, un allongement de la longueur a un coût comme le montre la figure 6.8 décrivant le délai supplémentaire introduit pas une telle action. Cette même figure montre aussi que pour un modèle de trafic bien déterminé, tel que le trafic Poissonien, il est possible de trouver une valeur qui offre un compromis entre l'allocation de ressource et la diminution des paquets supprimés. De plus, dans ce cas, la valeur du délai supplémentaire qu'implique les opérations de reconfiguration est majorée par une valeur (4,1ms sur les conditions de simulation). La valeur qui réalise le meilleur compromis (minimise les pertes pour une taille fixe, sans surplus de mémoire) se situe autour de $2Mo$, toujours pour les conditions de simulation. En

2. Le nombre de paquets perdus depuis le début de la simulation jusqu'à un instant t

effet, en ce point de la figure 6.8, le délai ne varie presque plus et allouer plus de mémoire ne rapporte plus grand chose car les pertes s'annulent.

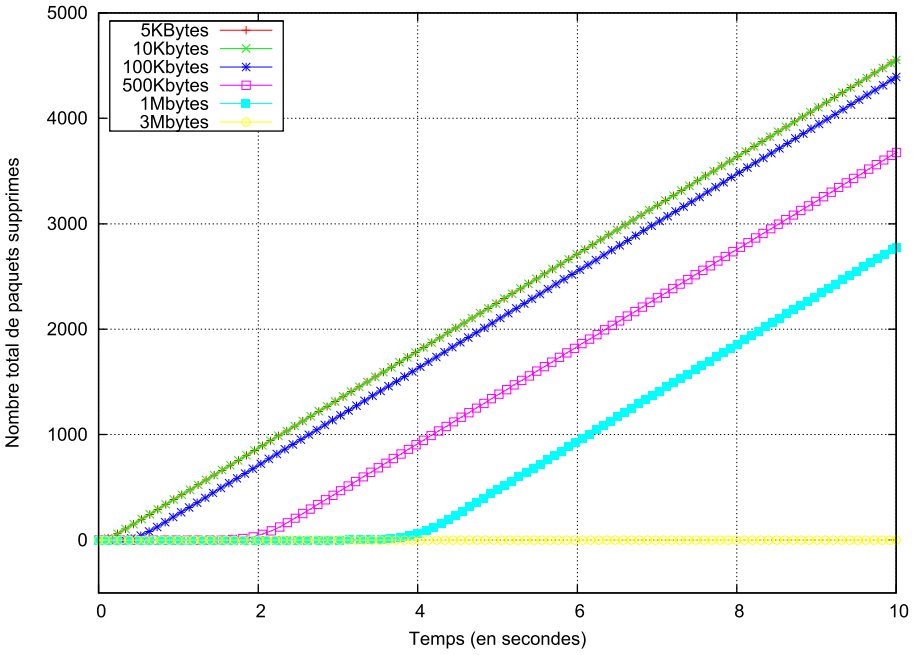

FIGURE 6.7 – Evolution des pertes cumulées en fonction de la taille des buffers

Nous avons retenu quatre actions `ChangerQueurDT`, `ChangerQueurFIFO`, `ChangerQueurRED` et `ChangerQueurTailleFileAttente`.

De plus, quelques tests sur une machine réelle, nous ont montré qu'une opération de configuration sur une machine réelle sous Linux avec l'infrastructure *netfilter* coûte $90\mu s$ quand l'arbre est totalement reconstruit et se situe autour de $30\mu s$ si l'opération consiste seulement à réajuster la taille des buffers. Ceci montre l'importance d'un paramètre qui est celui de la fréquence de reconfiguration car dans les systèmes réels il implique des pertes (paquets rejetés pour cause de non disponibilité) même très négligeables. Ce paramètre est nécessaire également pour l'observation des effets des actions sur les états.

Pour réaliser nos tests nous utilisons l'environnement de simulation JSIM [Hung-Ying] auquel nous avons ajouté l'autonomic manager qui implémente l'auto-adaptation ainsi qu'une interface avec Aleph[Srinivasan 2002] comme environnement de programmation logique inductive.

Le banc de test est décrit par la figure 6.9 avec des routeurs de bordure et de

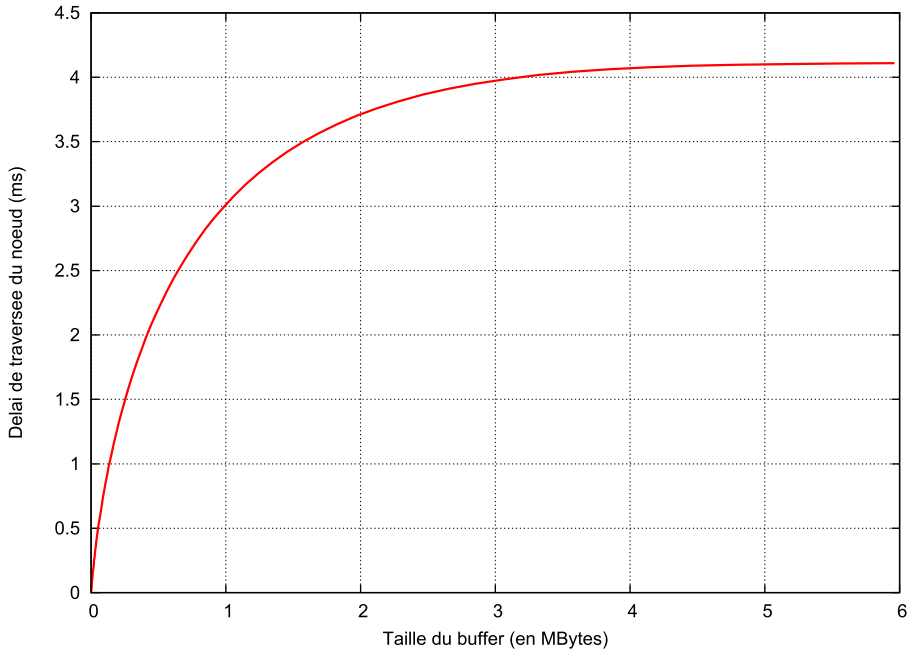

FIGURE 6.8 – Délai de traversée d'un nœud en fonction de la taille du buffer

cœurs ayant des autonomic manager (AM) implémentés et terminaux qui communiquent. Les routeurs dans le domaine DiffServ sont reliés par des liaisons de $5Mbps$ et les routeurs d'extrémités sont reliés aux terminaux par des liaisons Ethernet de $10Mbps$. Les deux types de liens ont des délais de propagation de $20ms$. Chaque terminal communique avec sont vis-a-vis : AF11-AF11, AF21-AF21, *ldots*

Le modèle de trafic de Poisson permet de simuler un trafic très perturbé et très aléatoire qui consiste un cas intéressant de simulation. A notre avis, si une méthode peut se permettre d'améliorer une telle situation, alors elle permettra d'améliorer des modèles moins accidentés.

D'autre part, se limiter seulement à la simulation des modèles de trafics réaliste, ne nous parait pas une bonne idée pour tester un système d'auto-adaptation car, demain si le comportement des utilisateurs changent les méthodes d'adaptation devront être alors prouvées dans ce contexte. Par opposition avec les flux de Poisson avec des tailles variables de paquets simulent bien les communications de plusieurs applications simultanées. Enfin, des flux réels tels que les flux VBR pourraient être considérés comme des cas particuliers de trafics *Poissoniens*.

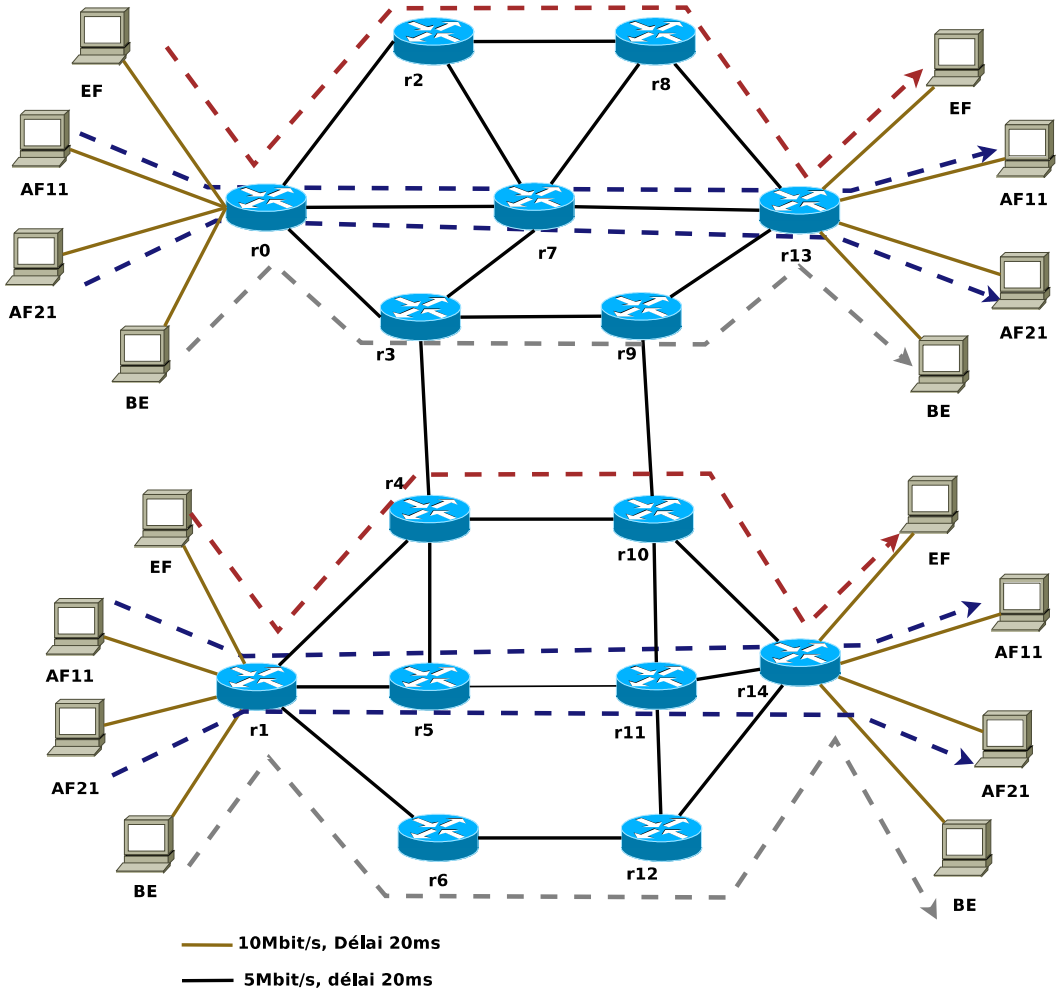

FIGURE 6.9 – Banc de test avec onze routeurs de cœur et quatre routeurs de bordure

Le trafic utilisé dans nos simulations est donc toujours de type *Poissonnien* avec une taille de paquets variable afin de simuler un trafic très instable. Les trafics AFxy et BE ont une taille moyenne des paquets est de 512 *octets* et le débit moyen est de 4 Mo/s. La bande passante réservée aux flux AF est de 3 Mo/s et les trafics EF sont de type CBR de 512 $kbit/s$. L'apprentissage doit ajuster la configuration afin de permettre aux flux AFxy et BE de maximiser leurs débits utiles en donnant la priorité aux flux AFxy.

Les actions d'augmentation ou de diminution de la taille des files d'attentes se font en ajoutant ou retranchant des tailles mémoire de 1 $Kbit/s$. Les changements

de Queuers se font entre FIFO, RED et une variante DropTail de FIFO qui retarde les paquets un petit moment avec un tampon secondaire au lieu de les rejeter.

6.3.5 Apport de l'apprentissage local

Les figures 6.10 et 6.11 représentent les résultats de la simulation du réseau *DiffServ* avec l'autonomic manager, comparés à une configuration classique. Pour obtenir ces résultats, nous avons lancé des flux (EF, AFxy et BE) en compétition sans activer le gestionnaire autonome pour avoir le débit sans auto-adaptation. Nous avons, ensuite, comparé le résultat obtenu avec celui en présence d'un gestionnaire autonome(AM) sur les nœuds de cœur.

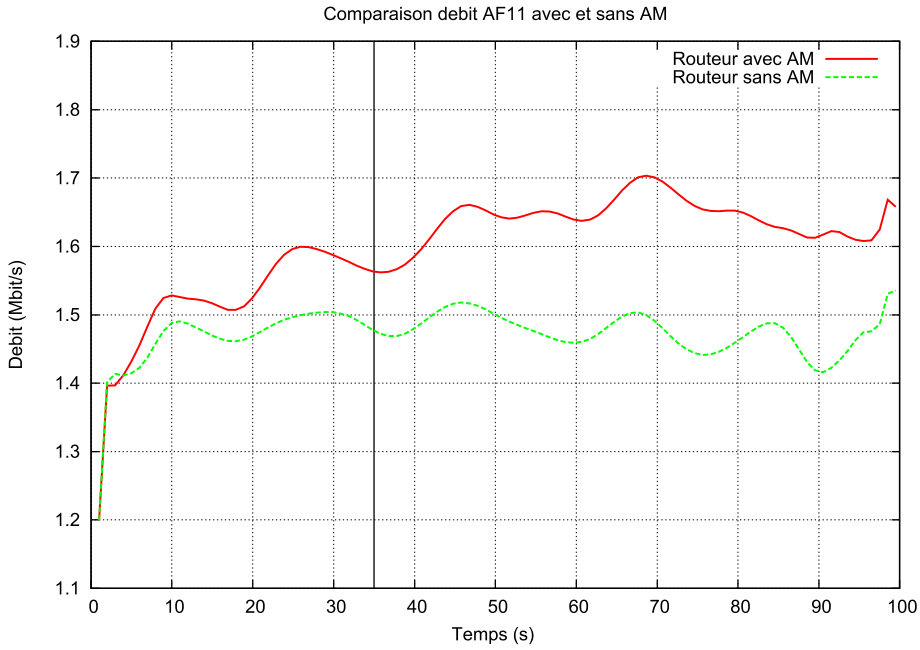

FIGURE 6.10 – Comparaison des débits avec et sans gestionnaire autonome (AM) pour AF11

Le résultat obtenu montre qu'il y a une amélioration du débit utile des flux AF avec le gestionnaire autonome. Ce résultat est confirmé par la figure 6.12 qui montre l'évolution du nombre de paquets supprimés dans le temps. Le module d'apprentissage permet de perdre beaucoup moins de paquets indifféremment de l'initialisation de la taille des buffers qui est effectuée dans le cas sans AM.

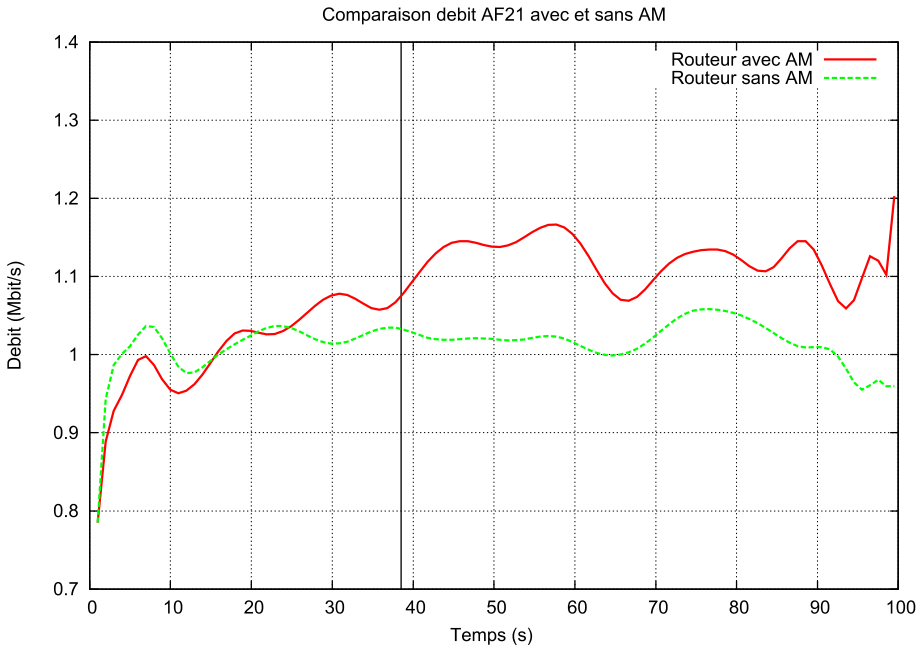

FIGURE 6.11 – Comparaison des débits avec et sans gestionnaire autonome (AM) pour AF21

Les figures 6.13 et 6.14 donnent l'évolution de l'indice de réussite des actions θ_t durant la simulation. Ces courbes montrent, qu'une fois l'apprentissage réalisé, pour de petites valeurs du taux de pertes utiliser FIFO permet de perdre moins de paquets que d'utiliser RED. Les résultats de la figure 6.14 nous font dire également que FIFO est parfois meilleur que RED lorsque la taille de la file d'attente est adaptative. Enfin, la manipulation de la taille des files d'attentes a beaucoup plus d'impact sur les taux de perte que le fait de changer de file d'attente. Dans tous les cas, la stratégie complexe consistant à utiliser le *queuer* FIFO et à manipuler sa taille est la plus intéressante.

6.3.6 Optimisation des actions

Dans les tests précédents, nous avons modifié la taille des files d'attente selon une suite arithmétique (ajout de 1 *Koctets*). L'adaptation est lente et beaucoup de paquets sont encore perdus en dépit de l'amélioration. Nous avons, par la suite, testé une adaptation de la taille des files d'attente qui évolue selon une suite géo-

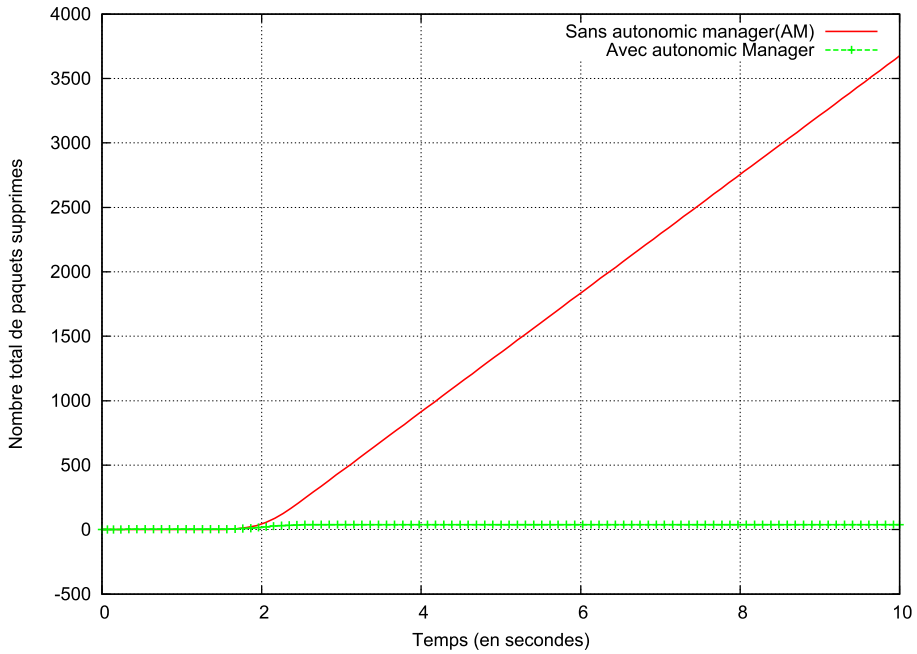

FIGURE 6.12 – Comparaison des pertes avec /sans apprentissage

métrique qui croît plus rapidement (pour dépasser rapidement l'optimum et ensuite se réajuster). Le tableau 6.2 présente la taille de la mémoire allouée après la stabilisation du système. Il montre que, en considérant que la taille optimale est de $2Mo$ pour ce cas précis (voir section 6.3.4.3), la raison de la suite, nous avons testé les valeurs $q = 1 + \frac{1}{n}$ avec, $n \in [1 \dots 10]$.

La figure 6.15 montre l'évolution des taux de pertes cumulés pour les valeurs de la raison. On remarque que l'algorithme se stabilise quelque soit la valeur de la raison de la suite, mais aussi quelque soit la nature de la suite (arithmétique, géométrique). La différence se fait sur le temps de stabilisation de l'apprentissage. Plus la raison de la suite géométrique est grande, moins le délai de stabilisation est long et plus le nombre de paquets perdus est faible.

Il est intéressant aussi de voir l'évolution du nombre d'opérations de configuration. En effet, comme nous l'avons déjà dit, ce paramètre a une influence sur les taux de perte mais permet aussi de mesurer la stabilité du système. La figure 6.16 montre l'évolution du nombre *d'échecs* en fonction de la raison. Un échec est une décision de configuration proposée par le module d'apprentissage qui n'a pas permis

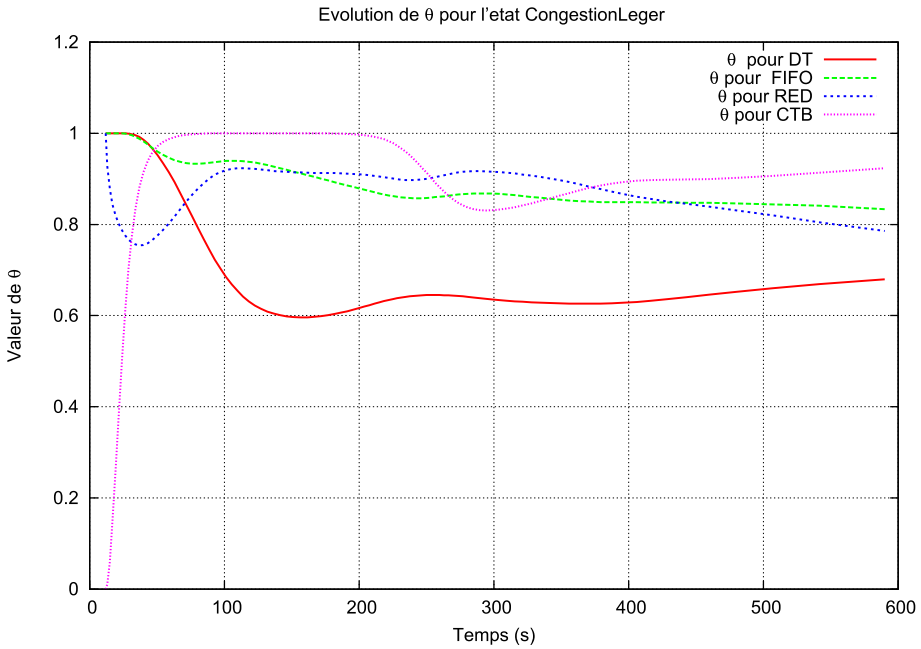

FIGURE 6.13 – Évolution de l'indice de réussite des actions (θ) pour l'état *CongestionLeger*

d'améliorer l'état de l'équipement. Toutes les figures ont la même allure avec une phase où le module choisit des stratégies peu ou pas efficaces. Ensuite, lorsque le nombre d'observations devient représentatif, il commence à découvrir des stratégies efficaces. Pour de très grandes valeurs de la raison, l'algorithme se stabilise avec des tailles de file d'attente qui ne sont pas forcement optimales (largement au dessus). Le compromis entre rapidité et valeur optimale de la raison est obtenu pour plusieurs valeurs : $\frac{6}{5}$, $\frac{7}{6}$ et $\frac{8}{7}$.

En résumé, l'algorithme se stabilise quelque soit la période de la suite considérée et quelque soit le type de suite (arithmétique, géométrique) dans les conditions de simulation, même dans un cas aussi extrême qu'un trafic Poissonien.

6.3.7 Apport de l'apprentissage collaboratif

A présent, nous nous intéressons à l'apport de l'apprentissage collaboratif au niveau de chaque nœud. Pour cela, dans un premier temps, nous faisons une simulation dans laquelle les nœuds rattachés au routeur de bordure $r0$ envoient du

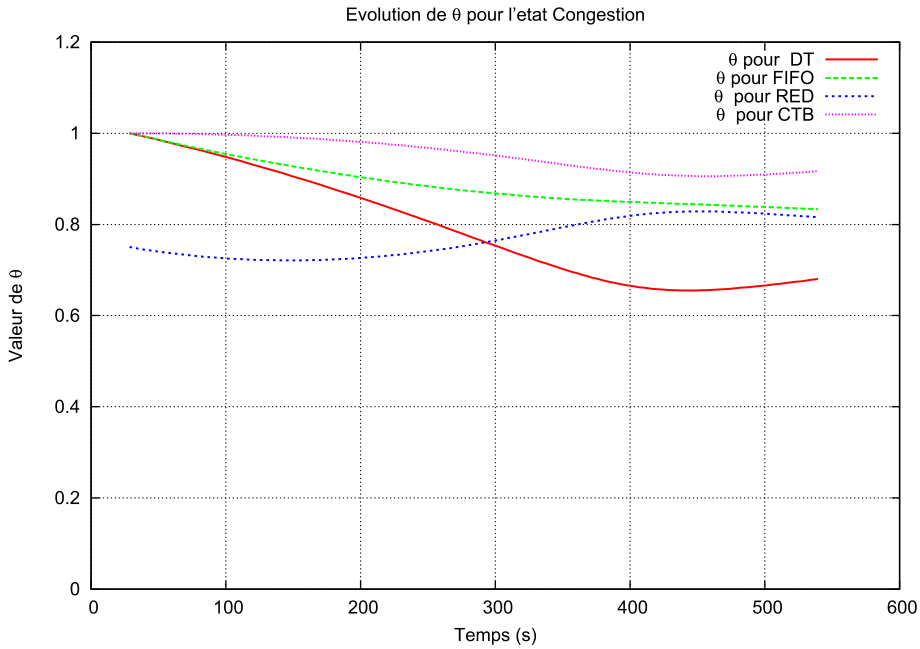

FIGURE 6.14 – Évolution de l'indice de réussite des actions (θ) pour l'etat *Congestion*

Raison de la suite	Taille mémoire allouée à la fin (Octets)
$\frac{3}{2}$	2560000
$\frac{4}{3}$	3284127
$\frac{6}{5}$	2034307
$\frac{7}{6}$	2040241
$\frac{8}{7}$	2050438
$\frac{9}{8}$	2584363
$\frac{10}{9}$	2802321
$\frac{11}{10}$	1804504
$\frac{12}{11}$	1641420
$\frac{13}{12}$	1521451
...	...

TABLE 6.2 – Taille mémoire à la stabilisation

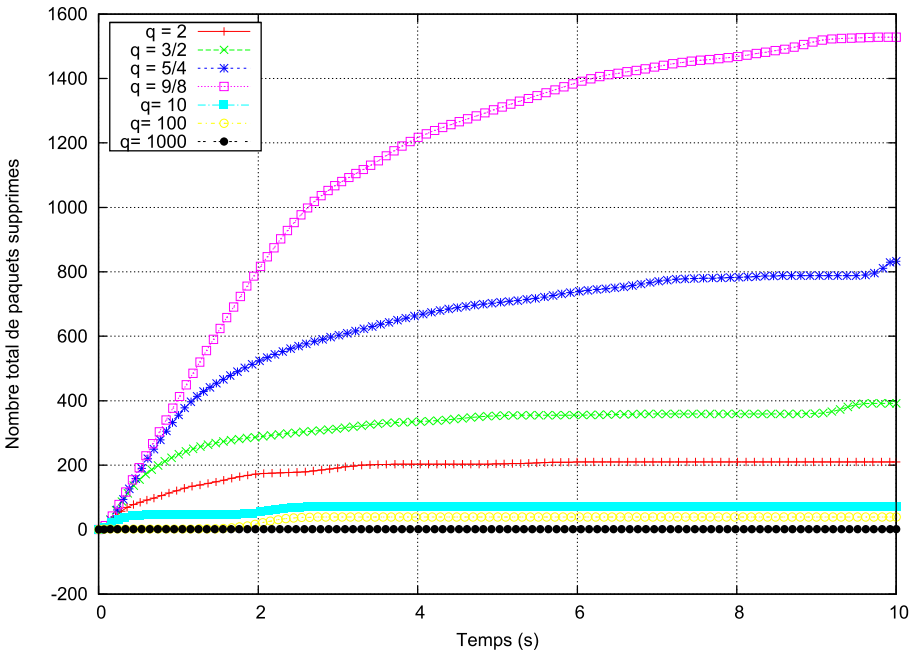

FIGURE 6.15 – Évolution du nombre de pertes cumulées pour différentes valeurs de période.

trafic pendant $10s$ puis les nœuds rattachés au routeur de bordure $r1$ commencent à envoyer des flux sur $r1$. L'opération est effectuée deux fois, une première fois sans collaboration et une seconde fois avec collaboration. La figure 6.17 présente le résultat de cette expérience. Sans collaboration, les processus d'apprentissage sur $r0$ (courbe en rouge) et $r1$ (courbe en vert) se comportent de la même manière et prennent le même temps ce qui était prévisible. Cependant, lorsqu'on active la collaboration, le temps de convergence de $r1$ est divisé par 10!. En effet, le point culminant de la courbe verte représentant le nombre d'echecs par unité de temps est de 48 pour r_1 sans échange de la connaissance alors qu'elle tombe à 5 avec la collaboration(courbe bleue). Et ses pertes réduites par un facteur 20, en comparant les pertes cumulée nous passons de 1232 paquets perdus dans le cas sans collaboration à 63 paquets perdus ce qui est un gain non négligeable. Le partage ici a permis une accélération considérable au niveau du routeur $r1$.

Nous avons fait un second test pour approfondir ce premier résultat. Cette fois-ci, nous distribuons de manière aléatoire 100 nœuds dans un réseau. Ces nœuds

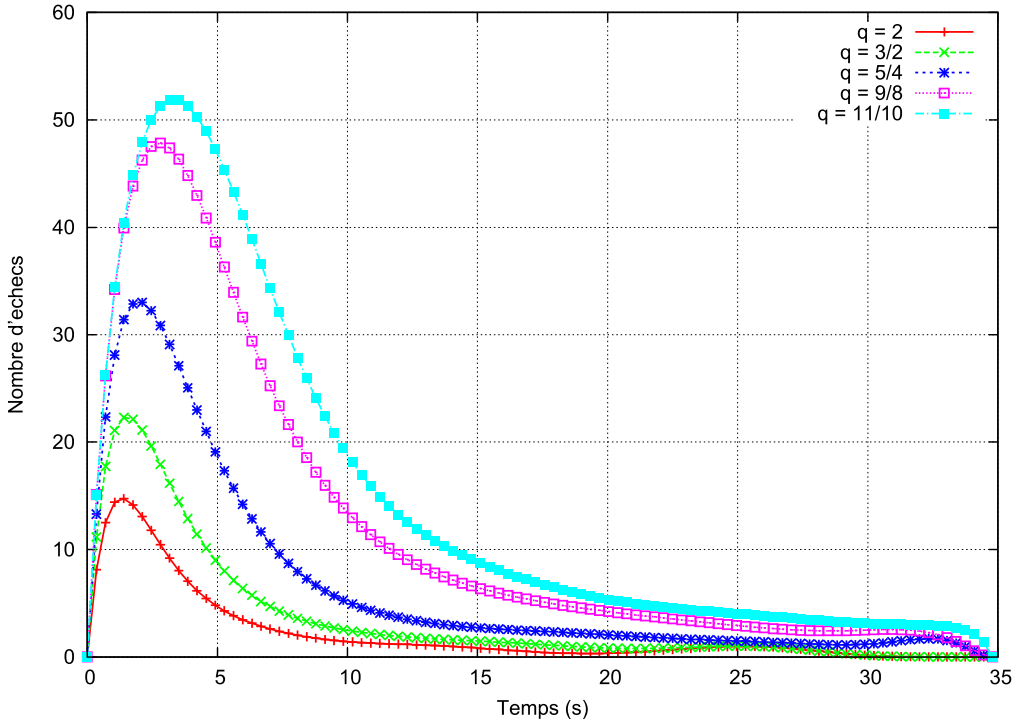

FIGURE 6.16 – Évolution du nombre d'échecs en fonction du temps avec différentes valeurs de la période.

communiquent avec des débits qui varient de manière aléatoire toutes les $10s$. On observe alors le facteur d'amélioration du temps de convergence en fonction de la distance par rapport au nœud de bordure. Le résultat est donné par la figure 6.18. En résumé, l'amélioration observée sur le nœud $r1$ décroît quand on s'éloigne de $r1$ qui est plus proche de la source du problème.

L'échange de connaissance permet d'améliorer l'apprentissage en diminuant les pertes en moyenne de près de 60% et en accélérant le temps de stabilisation du système jusqu'à 73%. Cependant, plus on s'éloigne de la source véritable du problème, moins la collaboration a un impact fort sur la vitesse de l'apprentissage. Cette diminution est peut être due à l'attenuation du problème à chaque étape en amont par les noeuds les plus proches de la cause du problème. Ceci peut justifier le choix que nous avions fait de ne faire la collaboration que de proche en proche et de n'echanger les critiques qu'entre les élements directement voisins.

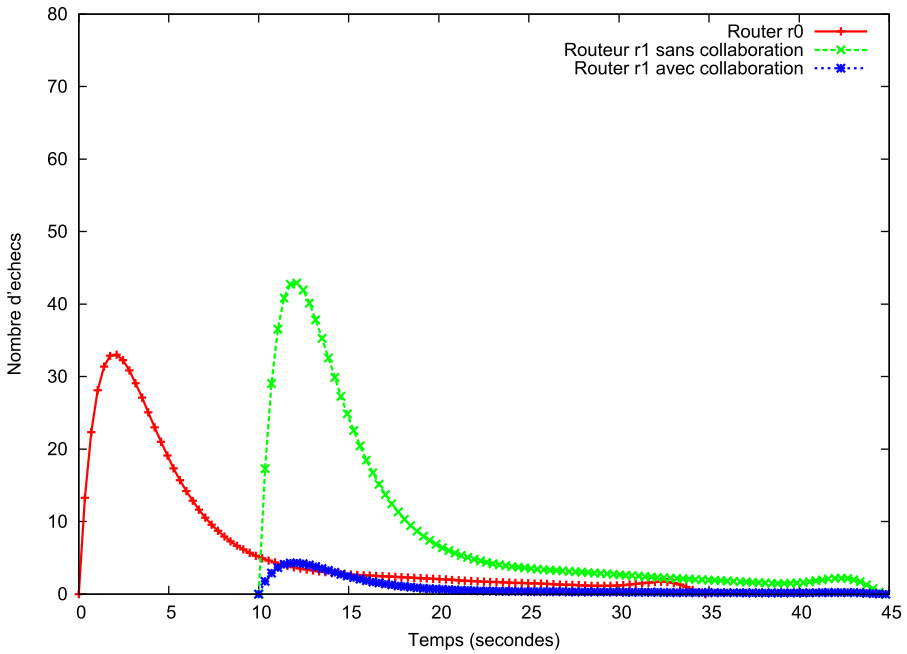

FIGURE 6.17 – Évolution du nombre d'échecs, avec et sans collaboration, pour le routeur $r1$

6.4 Application multimédia dans les réseaux sans-fil

Dans cette section, nous proposons une transposition de notre méthode d'apprentissage dans le cadre de la problématique de la transmission flux IPTV sur les réseaux sans-fil 802.11.

Ce type d'applications est souvent face au problème de la non fiabilité du canal. En effet, les taux de pertes sur le canal ne sont pas contrôlables. On ne peut pas remplacer le média air comme il saurait de le faire dans le cas d'une paire torsadée dégradée. Dans ce contexte, [Djama 2008] proposent un système pour la transmission des flux audio/vidéo sur les réseaux 802.11 basé sur une approche Cross-layer qu'il a implémenté sur une plateforme IPTV. Quatre adaptations sont proposées pour améliorer la robustesse de la plateforme face aux pertes : l'adaptation dynamique du débit vidéo en fonction du débit physique, l'adaptation conjointe de la FEC et du débit vidéo en fonction de la puissance du signal et des taux de pertes,

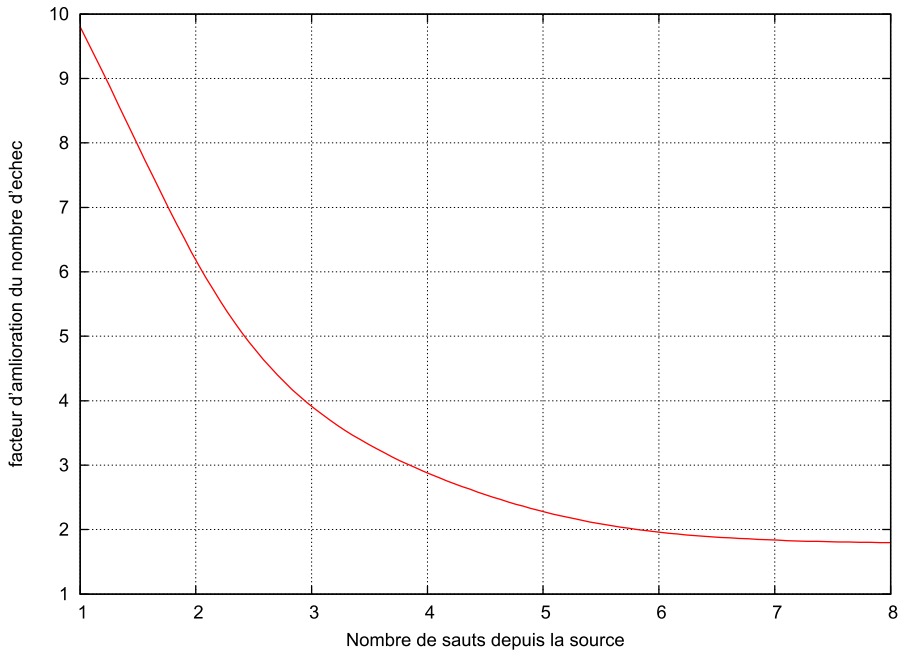

FIGURE 6.18 – Facteur d'amélioration du temps de convergence dans le cas collaboratif, en fonction de la distance (longueur du chemin) à la source d'une connaissance transmise.

la fragmentation 802.11 adaptative pour trouver un compromis entre les pertes de paquets et l'*overhead* introduit par les couches 802.11 et, enfin, le groupage des images vidéo au niveau MAC pour permettre au flux vidéo d'avoir un accès au canal 802.11 proportionnel à son débit. Cependant, ces adaptations sont testées de manière indépendante et il n'est pas encore possible de déterminer ce que pourrait donner leur comparaison ou bien leurs interactions lorsqu'elles sont appliquées simultanément.

Nous allons présenter ces différentes adaptations avant de détailler comment notre approche pourrait être transposée dans ce cas pour trouver des stratégies d'adaptation efficaces.

Les différentes adaptations implémentées sur la plateforme IPTV ont pour objectif de maintenir la QoS lors de la transmission des flux vidéo aux clients et peuvent être réparties selon deux approches globales : les adaptations ascendantes exécutées au niveau applicatif et les adaptations descendantes exécutées au niveau MAC 802.11.

6.4.1 Adaptations ascendantes

Dans le cadre de cette approche, deux adaptations ont été mises en œuvre. Ces dernières sont exécutées au niveau applicatif, et permettent aux couches supérieures de s'adapter dynamiquement aux variations des couches inférieures.

Adaptation dynamique du débit vidéo en fonction du débit physique : En effet, les conditions de transmission dans les réseaux 802.11 varient suivant plusieurs paramètres. Par exemple, lorsque la distance entre la station émettrice des flux vidéo et la station cliente devient importante. Il y a alors une contrainte de réduction du débit physique utilisé pour transmettre les flux IPTV. Ceci peut ne pas satisfaire le débit vidéo du flux et donc, présente un risque de ne transmettre qu'une partie des données.

Néanmoins, la mise en œuvre d'une telle adaptation nécessite d'abord de déterminer dynamiquement le débit physique en fonction duquel le flux à transmettre sera adapté. Ceci se fait à l'aide d'algorithmes de contrôle du débit physique 802.11, appelés RCA (Rate Control Algorithm).

Ces RCA sont regroupés en trois principales catégories selon les informations utilisées pour juger de l'état du canal. D'une part, il y a des algorithmes basés sur des statistiques permettant une adaptation du débit à long terme. D'autre part, il y a les RCA basés sur le SNR (Signal to Noise Ratio) qui réagissent rapidement aux changements de l'état du canal, mais ils manquent de fiabilité à cause de l'estimation approximative du SNR. La troisième catégorie regroupe les RCA hybrides qui profitent des avantages et inconvénients des deux catégories précédentes.

Une fois le débit physique déterminé, le module XLDP décide du débit physique effectif utilisé pour chaque station cliente. En effet, le débit physique est supérieur au débit physique effectif puisqu'il considère les délais supplémentaires introduits par les entêtes des couches réseaux et le mécanisme d'accès 802.11 MAC. Pour cela, l'équation ci-dessous est utilisée pour calculer le débit physique effectif :

$$[ht] \sum_{i=1}^{nb} \frac{1}{r_i} \tag{6.1}$$

Avec, nb : Le nombre de trames transmises, r_i : Le débit physique théorique utilisé pour transmettre les trames à la station i.

Par la suite, le module XLDP détermine le débit vidéo en permettant au flux IPTV d'occuper un pourcentage du débit physique effectif. Ce pourcentage dépend des politiques fixées par l'administrateur du réseau.

Les interactions entre les différents modules de cette adaptation au niveau XLAG sont illustrées sur la figure 6.19 ([Djama 2008]) :

Enfin, Le module **Transcoder** se charge d'adapter le débit vidéo à la nouvelle valeur calculée. Pour cela, deux techniques sont utilisées dans le cas d'un flux vidéo MPEG1 :

- La variation de la résolution temporelle : elle consiste à varier le nombre d'images par seconde d'une séquence vidéo, en supprimant les images suivant par exemple la hiérarchie temporelle introduite par le codage MPEG. Le débit vidéo varie alors en fonction du nombre d'images par seconde.
- La variation de la résolution SNR : la résolution SNR correspond au degré de finesse des images vidéo définies par les fréquences élevées d'une matrice DCT. Ces fréquences sont réduites durant le codage spatial par la procédure de quantification en utilisant le facteur de quantification. Ainsi, le nombre de fréquences nulles dans une matrice DCT augmente lorsque le facteur devient plus élevé. Ceci réduit la finesse de l'image, mais réduit aussi le débit de la vidéo. Par contre, lorsque le facteur de quantification est faible, la matrice DCT contiendra plus de fréquence qui font augmenter la finesse des images, et par la même, le débit vidéo.

Néanmoins, la deuxième technique offre un meilleur contrôle du débit et une meilleure qualité vidéo.

Adaptation conjointe de la FEC et du débit vidéo en fonction de la puissance du signal et des taux de pertes : L'objectif de cette adaptation est de permettre aux flux vidéo de devenir plus résistants aux pertes tout en gardant un débit stable. Pour atteindre cet objectif, plusieurs mécanismes capables de gérer les pertes de paquets peuvent être implémentés. Parmi ces mécanismes, la FEC (Forward Error Correction) représente une des solutions les plus utilisées pour la transmission des données en temps réel.

La FEC est un système de protection contre les erreurs lors de la transmission de données. Ce mécanisme se base sur des codes correcteurs afin de permettre au

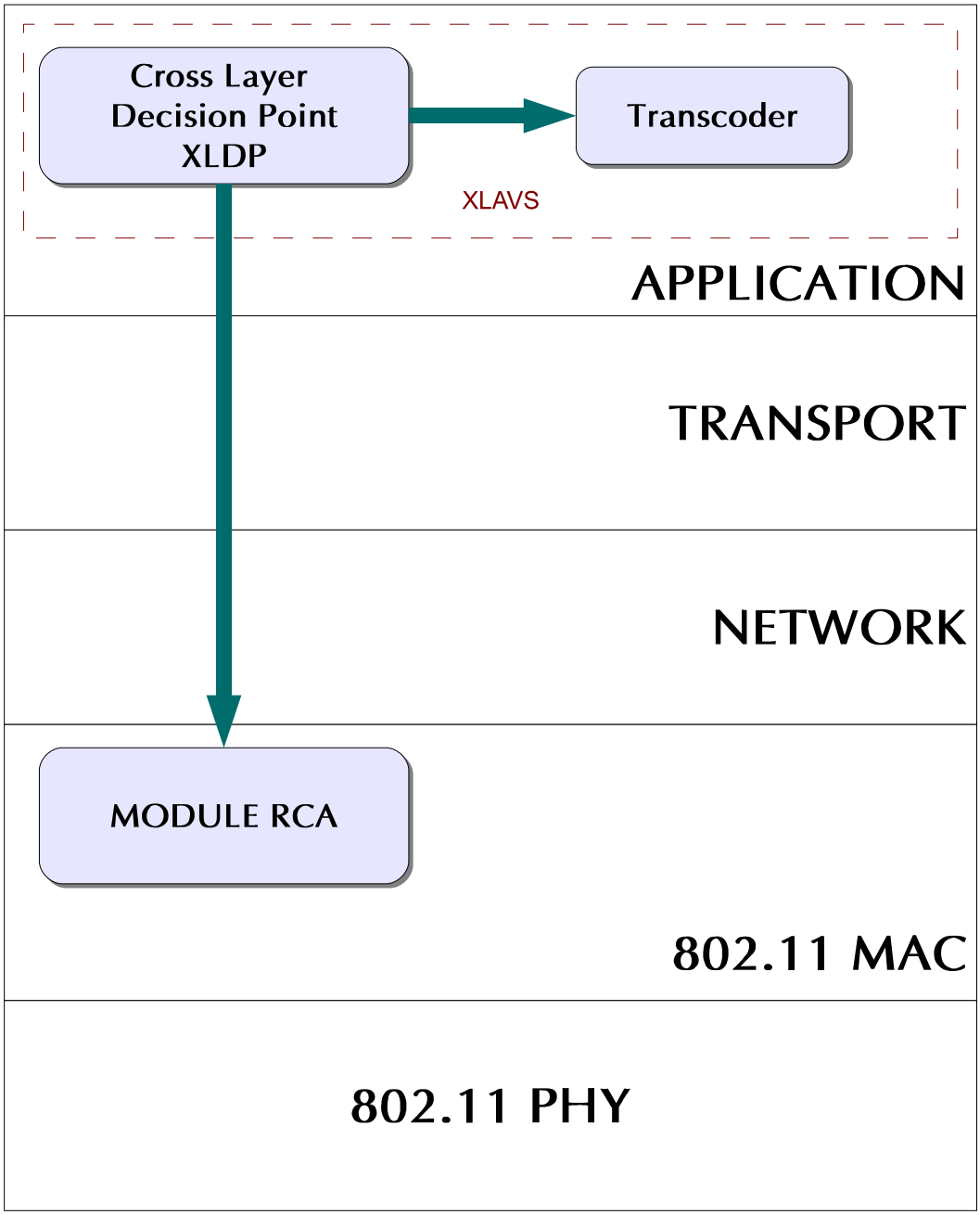

FIGURE 6.19 – Interactions durant l'adaptation du débit de la vidéo

récepteur de générer les paquets perdus. En effet, il ajoute de la redondance afin de permettre au destinataire de détecter et de corriger une partie des erreurs. Cela permet d'éviter la retransmission, et donc de faire des économies de bande passante, voire d'assurer la transmission dans certaines situations où il n'y a pas de voie de

retour.

Différents codes peuvent être utilisés par la FEC (code XOR, code LDPC, ...). Cependant, le code RS (Reed Salomon) reste largement déployé pour les communications DVB-S. La redondance de ce sur-échantillonnage permet au récepteur du message encodé de reconstruire le polynôme même s'il y a eu des erreurs pendant la transmission.

Les performances de ce mécanisme restent très limitées puisqu'il réagit une fois que les pertes se sont produites. De plus, les paquets redondants entrainent une augmentation du débit qui représente un inconvénient réel dans les réseaux 802.11, et qui ne sont d'aucune utilité lorsque les pertes sont minimes.

6.4.2 L'approche descendante

A l'instar de la première partie, deux adaptations ont été mises en œuvre dans le cadre de cette approche. Ces dernières permettent aux couches supérieures de configurer dynamiquement les couches inférieures.

La fragmentation adaptative au niveau MAC 802.11 : La transmission du signal dans les réseaux 802.11 est soumise à de nombreuses contraintes rendant les communications plus difficiles. Parmi ces contraintes, il y a notamment l'évanouissement rapide (Fast Fading) qui provoque la corruption des trames lors de la transmission. Ces trames corrompues sont supprimées directement par la couche MAC après vérification du champ de contrôle des erreurs nécessitant alors leur retransmission. D'autre part, la probabilité de corruption de ces trames devient plus grande lorsque celles-ci sont exposées aux perturbations du canal pendant une durée plus longue. Autrement dit, lorsqu'elles ont une taille plus grande.

Néanmoins, le choix de la taille de la trame doit répondre à un compromis entre la durée de transmission et l'*overhead* introduit par les entêtes des différentes couches de transmission. En effet, la transmission de trames de petites tailles nécessite moins de temps mais augmente en contre partie le pourcentage de l'*overhead*.

C'est dans ce sens que cette première adaptation appartenant à l'approche descendante a été implémentée. La fragmentation au niveau MAC des réseaux 802.11 permet donc de réduire les pertes de paquets durant les périodes de fortes interfé-

rences.

Afin de mettre en œuvre cette adaptation, il faut d'abord déterminer la taille idéale de la trame à émettre. Cependant, les problèmes liés à l'estimation des taux de pertes en temps réel rendent cela impossible. D'où l'idée de trouver une limite inférieure pour la taille d'une trame qui respecte un certain taux d'overhead. Ceci se fait à l'aide de l'équation suivante :

$$fs_i = \frac{hs_{max} + hs_{phy}}{1 - Ri_{llc} \times \sum_{i=1}^{nb} \frac{1}{r_i}} \tag{6.2}$$

Avec,

fs_i : la taille minimum d'une trame pour une station i.

Ri_{llc} : le débit du flux au niveau de la couche LLC destiné à la station i.

nb : Le nombre de trames transmises.

r_i : Le débit physique utilisé pour transmettre les trames à la station i.

hs_{mac} **et** hs_{phy} : la taille respectivement de l'entête de la couche mac et physique.

Une fois la taille limite déterminée, il est possible d'adapter dynamiquement la taille des trames en utilisant la fragmentation MAC 802.11. En effet, la norme 802.11 offre un mécanisme de fragmentation, permettant de découper une trame en plusieurs morceaux. L'interaction entre les différents modules qui composent cette adaptation au niveau du XLAVS sont décrit sur la figure 6.20 ([Djama 2008]) :

En effet, le module XLDP récupère les informations nécessaires pour calculer la limite inférieure de la taille de la trame à envoyer pour chaque station. Ces paramètres sont :

- Le nombre de stations qui sont connectées au XLAG.
- Le débit physique de chaque station.
- Le débit de chaque flux vidéo au niveau de la couche LLC.
- Le taux de perte RTCP de chaque session IPTV ouverte au niveau du XLDP.

Ensuite, cette nouvelle valeur est signalée à la couche mac grâce à l'appel de fonction interne `iwconfig`. Ce calcul est enfin répété après la réception de chaque nouveau rapport RTCP.

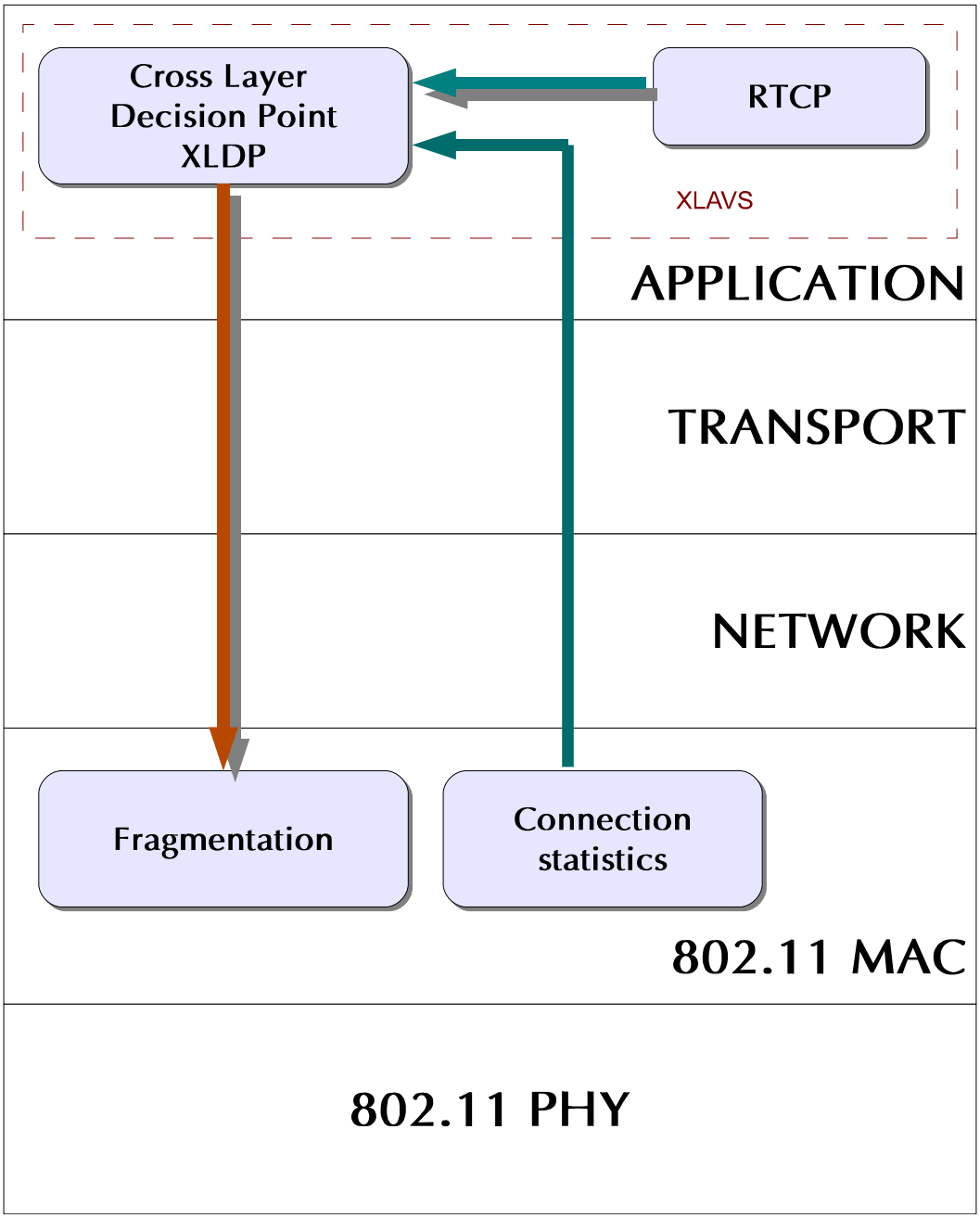

FIGURE 6.20 – Iinteractions entre les différents modules pour la fragmentation MAC 802.11

Le groupage de trames basé sur l'image vidéo : La dernière adaptation implémentée est motivée par le besoin de garantir plus de bande passante au niveau du lien d'accès 802.11 pour les flux vidéo, lorsque ces derniers sont concurrencés

par d'autres flux asynchrones UDP ou TCP. En effet, le débit du flux dans ce cas est sérieusement limité à cause du partage équitable du lien d'accès garanti par le mécanisme d'accès DCF (Distributed Coordination Function).

L'objectif est de réduire le temps supplémentaire introduit par la couche MAC, et qui correspond à la durée de transmission de l'en-tête et du préambule physique (PLCP préambule et PLCP header), et aussi, les intervalles de temps IFS définit par le DCF afin d'assurer un accès équitable au canal.

Ainsi, le principe de fonctionnement de cette adaptation consiste à partager le temps d'accès au canal entre plusieurs trames possédant la même destination. Cela se fait en transmettant ces trames en rafale sans relâcher le canal de transmission. Chaque trame est alors acquittée par le récepteur. Ce mécanisme peut être très utile pour assurer aux flux vidéo plus de bande passante sur le lien d'accès par rapport à d'autres flux.

La mise en œuvre du groupage des trames au niveau du XLAG, nécessite d'abord de délimiter les images qui vont être regroupées et transmises en rafale. Une des solution consiste à introduire une taille de groupe dynamique pour chaque flux vidéo. Cette dernière correspond à une image vidéo qui est déterminée à partir du débit vidéo et du type d'image pour un codage MPEG (images I, P ou B).

6.4.3 Transmission auto-adaptative

Toutes ces adaptations se font en fonction d'un certain nombre de paramètres que sont les taux de pertes, le niveau du signal, la qualité du canal. Et les métriques des adaptations peuvent également être exprimées en fonctions de ces mêmes paramètres.

La problématique qui consiste à déterminer les adaptations les plus efficaces en fonction des valeurs des paramètres que nous avons cités plutôt, peut être posée en termes d'auto-adaptation telle que nous l'avons posé dans le chapitre 5.

En effet, les états du système peuvent être définis avec la combinaison des taux de pertes et du niveau du signal. L'espace des états est un plan et il va donc nécessiter des tests pour déterminer un bon découpage. Le critère de performance peut être déduite du taux de pertes à la réception de la vidéo. Ce critère est non ambigüe car il s'agit d'un ensemble totalement ordonné. Les actions vont être les

différentes adaptation proposées par [Djama 2008] et décrite précédemment.

Le module a été implémenté sur la plateforme mais, malheureusement, nous n'avons pu tester l'efficacité de l'autonomic manager dans ce contexte à cause de l'indisponibilité du module de la plateforme IPTV nous permettant de récupérer les taux de pertes indispensable à la prise de décision.

6.5 Adaptation par prédiction

Les cas d'applications présentés précédemment se situent dans un cadre où le mécanisme autonome est réactif face aux changements de l'environnement. Elle est adaptée dans le cas où l'application des actions a un impact direct sur le comportement de l'environnement. En effet, dans le cas des réseaux DiffServ, la configuration peut avoir un impact direct sur les taux de pertes, les liens filaires étant assez fiables. Cependant, dans le cas d'un réseau sans-fils, le système ne peut pas avoir une influence sur les pertes dues à la non fiabilité du lien air. Dans [Djama 2008], il est proposé de faire une FEC (Forward Error Corrector) adaptative dans le cadre d'un environnement de transmission de flux vidéo sur les réseaux 802.11 qui se base sur des règles de politiques. Ce mécanisme est très efficace et donne de bons résultats. Cependant avec des politiques fixes, l'*overhead* n'est pas optimal. Dans cette section nous allons présenter un mécanisme de prédiction des taux de pertes qui permet d'améliorer l'*overhead* tout en gardant son efficacité.

6.5.1 Introduction à la FEC

Il existe plusieurs mécanismes de QoS de niveau applicatif qui permettent aux applications multimédia de gérer les pertes de paquets. Parmi ces mécanismes, la FEC est une technique très utilisée pour les transmissions temps réel. Le principe de base de la FEC est la génération de paquets vidéo redondants à partir du flux vidéo original en utilisant un code correcteur particulier. Les paquets originaux et les paquets redondants sont transmis simultanément afin de permettre au récepteur de générer les paquets perdus en temps réel durant la réception du flux. Ainsi, la FEC représente un mécanisme parfait pour la transmission de flux vidéo sur les réseaux sans-fil caractérisés par leur manque de fiabilité et leur dépendance vis-a-vis

de l'état du canal.

Cependant, deux inconvénients majeurs sont engendrés par l'utilisation de la FEC :

- L'augmentation du débit du flux vidéo causée par l'ajout des paquets redondants. Cette augmentation est variable et dépend principalement du taux de redondance utilisé au niveau du code correcteur.
- L'augmentation du délai de transmission de bout-en-bout due à l'introduction de délais supplémentaires. Ces derniers sont engendrés par le codage des paquets redondants au niveau de l'émetteur et le décodage des paquets perdus au niveau du récepteur.

Actuellement, il existe plusieurs codes correcteurs qui peuvent être exploités par un mécanisme FEC. Chaque code possède des caractéristiques spécifiques comme sa complexité algorithmique et son taux de redondance maximal. Les codes correcteurs sont définis théoriquement sur des symboles qui peuvent avoir différentes tailles. La génération des symboles redondants est appelée codage et la génération des symboles perdus à partir des symboles redondants est appelée décodage.

Pour la transmission de données, ces codes peuvent être appliqués sur deux niveaux distincts : au niveau bit et au niveau paquet. Dans le premier niveau, le symbole est un bit et le code correcteur génère des bits redondants. Par contre, dans le deuxième niveau, un paquet est considéré comme un symbole et des paquets redondants sont codés à partir des paquets originaux.

Lorsque le mécanisme FEC est utilisé au niveau applicatif, la FEC au niveau paquet est plus avantageuse pour différentes raisons listées ci-dessous comme le rapporte [Djama 2008] :

- Les pertes dans le réseau sont considérées à un niveau de granularité *paquet*, donc la FEC au niveau bit ne peut pas décoder un paquet perdu entièrement.
- La FEC avec un niveau de granularité paquet offre une capacité de décodage supérieure car un paquet redondant, qui appartient à un groupe de paquets, peut décoder n'importe quel paquet perdu dans ce groupe.
- La FEC au niveau bit utilisée avec des paquets corrompus durant la transmission n'atteint jamais la couche applicative. Ces paquets sont supprimés automatiquement au niveau MAC avec la vérification du CRC.

Pour la transmission de flux multimédia, le mécanisme FEC est implémenté au

niveau du protocole RTP afin d'exploiter la numérotation séquentiel des paquets présente dans l'en-tête RTP comme souligné dans [Djama 2008]

Plusieurs mécanismes de code correcteurs ont rencontré un grand succès, nous renvoyons le lecteur à [Djama 2008] pour une présentation détaillées. Nous présentons uniquement celle qui nous intéresse, à savoir le code correcteur Reed Salomon [Djama 2008].

Le code RS forme une classe spéciale de code linéaire cyclique non binaire très puissant capable de corriger plusieurs erreurs. Il est construit sur un ensemble fini de symboles, appelé corps ou champ de Galois $CG(q)$. Un code $RS(n,k)$ génère n paquets (avec $n = q-1$) à partir de k paquets sources en ajoutant h paquets redondants (avec $h = n-k$) comme illustré par la figure 6.21 ([Djama 2008])

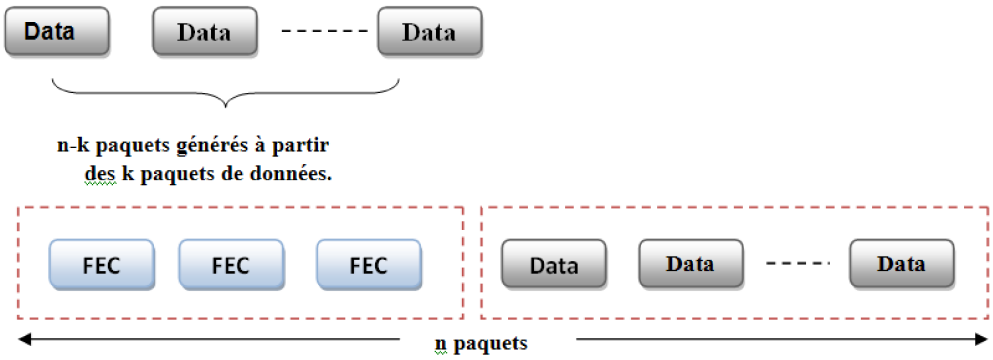

FIGURE 6.21 – Principe du code correcteur RS

Ce type de correcteur a une capacité de décodage assez important ce qui peut constituer un avantage important pour son utilisation dans le cadre des flux multimédia. Par contre, sa complexité algorithmique est aussi très importante, ce qui peut être un inconvénient.

Avec la FEC $RS(n,k)$ entre deux stations communicantes, le récepteur peut décoder jusqu'à $h = n-k$ paquets lorsqu'il a une telle quantité de pertes. Ce mécanisme permet donc d'augmenter la résistance du récepteur aux pertes du réseau avec une probabilité maximale p_{max} telle que :

$$p_{max} = \frac{h}{n} \tag{6.3}$$

En se basant sur cette équation, le taux de redondance utilisé par le mécanisme

Taux de perte(%)	FEC (n, k)	Overhead
= 0%	Pas de FEC	0%
< 5%	FEC (30, 24)	+25%
> 5%	FEC (30, 21)	+42%

TABLE 6.3 – Politiques d'adaptation de la FEC

FEC au niveau de l'émetteur peut être calculé à partir des taux de pertes (p) que subisse le flux durant la transmission. Ces taux de perte peuvent être déterminés au niveau du récepteur et rapportés périodiquement à l'émetteur en utilisant, par exemple, les rapports du récepteur (RR : Receiver Report) dans le protocole RTCP.

$$h = \frac{p \times k}{1 - p} \tag{6.4}$$

L'équation 6.4 permet de déterminer le taux de redondance dynamiquement en se basant sur les taux de pertes instantanés. Plusieurs travaux de recherche utilisent ce type d'approche pour adapter le mécanisme FEC en fonction des conditions de transmission réseau et réduire, par la même, l'*overhead* global (débit supplémentaire) introduit par le mécanisme FEC. Le taux d'*overhead* est donné dans l'équation suivante :

$$Ov_{FEC} = \frac{h}{k} \tag{6.5}$$

Afin d'avoir un mécanisme FEC adaptatif qui anticipe l'apparition des pertes, il faut utiliser d'autres métriques que seulement les rapports des pertes qui informent sur les pertes de la période précédente. Entre autre, il faudrait avoir à disposition au niveau de l'émetteur la probabilité d'apparition de ces pertes et leur valeur dans l'étape suivante (future). Des travaux qui modélisent ce problème par un problème de processus markovien, mais ces modélisations ne détectent pas les cas où la FEC est inutile.

[Djama 2008] propose d'avoir une FEC adaptative qui est couplée avec l'adaptation du débit de la vidéo pour diminuer le débit de la vidéo. Cependant, l'adaptation est effectuée suivant un ensemble de politiques fixes, donc une prédiction fixe des pertes (Table 6.3).

Nous proposons à la place, une approche stochastique qui permet de prédire les taux de pertes sur le canal.

6.5.2 Adaptation de la FEC par prédiction des taux de pertes

Le principe général de notre approche consiste à de considérer un découpage de l'ensemble des taux de pertes en segments de 10% qui representent les états du système. Il s'agit manitenant de prédire de manière stochastique les transitions similaires pour determiner les prochains taux de pertes. Chaque transition a une certaine probabilité de se réaliser.

Considérons les états $e_1 \dots e_{10}$, chaque état e_i étant décrit par l'intervalle de taille 10% qui contient i en commencant par 0% ($e_1 = [0\%, 10\%[$). Une transition $e_i \rightarrow e_j$ a une frequence notée $\tau_{i,j}$.

Pour chaque état e_i, les transitions observées se font sur un ensemble $\{e_{i_j}\}_{j \leq 10}$. Nous utilisons la matrice des fréquences τ_{i,i_j} pour faire la prédiction des taux de perte à la prochaine étape sachant qu'on est à l'état e_i. Il est possible de calculer cette valeurs de deux manières :

- En considérant l'état e_{i_j} qui maximise la probabilité des transitions depuis l'état e_i. La probabilité d'une transition étant calculé, en faisant le rapport entre $\tau_{i,j}$ et le nombre d'apparitions de e_i :

$$ProchainePertes(e_i) = \frac{1}{n} \sum_{j=1}^{n} \tau_{i,i_j} \times \frac{(sup(e_j) - inf(e_j))}{2} \tag{6.6}$$

  $sup(e_j)$ et $inf(e_j)$ étant les bornes de l'intervalle des pertes que représente e_j.
- En considérant que chaque état est défini par la valeur du milieu du segment, et en calculant une moyenne barycentrique avec une pondération de chaque transition $e_i \rightarrow e_j$ par $\tau_{i,j}$.

Nous avons testé l'efficacité de la prédiction sur la plateforme IPTV développé par [Djama 2008]. Pour cela nous avons commencé par capturer des pertes dans le réseau sans-fil de l'ENSEIRB (Ecole Nationale Supérieure d'Electronique d'Informatique et de Radiocommunications de Bordeaux). Cette capture a été effectuée sur différentes périodes d'affluence avec l'outils Iperf [IPERF 2008] avec une fréquence

d'une capture toutes les 5 secondes.

Les pertes sont représentées sur la figure 6.22.

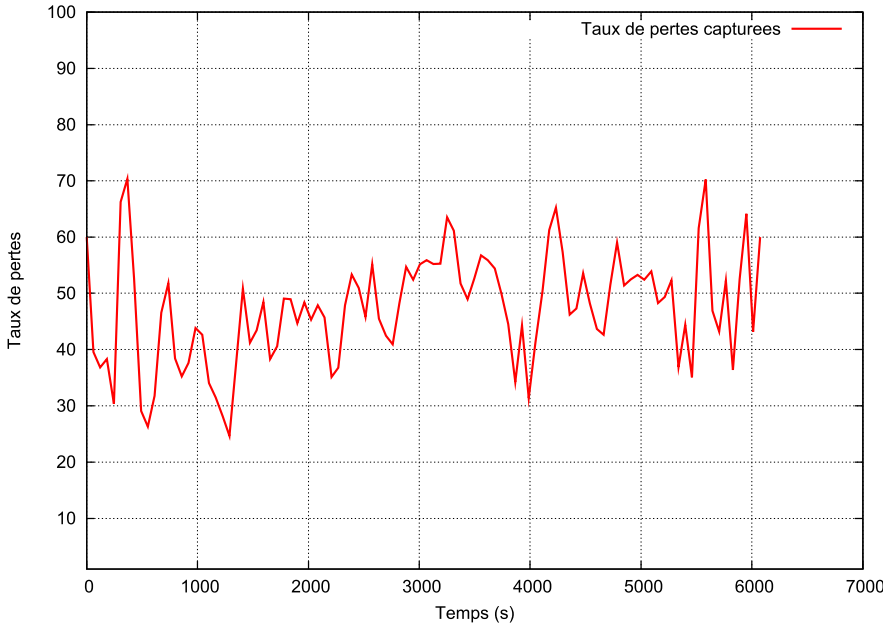

FIGURE 6.22 – Pertes capturées sur le réseaux de L'ENSEIRB

Ces traces ont été ensuite injectées sur la plateforme IPTV en utilisant l'outil *netem* d'une machine Linux. Les résultats de la prédiction sont données par la figure 6.23.

Il se dégage deux résultats essentiels de ces courbes :

- L'utilisation de la moyenne barycentrique et de la plus grande fréquence donne des résultats presque superposables, l'utilisation de la fréquence donnant une prédiction des pertes supérieures à celle du barycentre. Cette similarité est cependant moins parfaite que ce que présage la figure 6.23. En effet, c'est le mécanisme de lissage de *gnuplot* sur un aussi grand nombre de points (6000) qui fait que les petites différences sont noyées dans la masse.
- La prédiction dans tous les cas ne s'éloigne pas de l'allure des pertes. L'utilisation de ces prédictions, en transformant un taux de perte de $x\%$ en des valeurs de redondance $\frac{x}{10}$ paquets de redondances sur 10 paquets, permet de reduire l'*overhead* tout en gardant l'efficacité de l'adaptation proposée par [Djama 2008].

En faisant la moyenne mathématique des différences, en valeurs absolues, entre

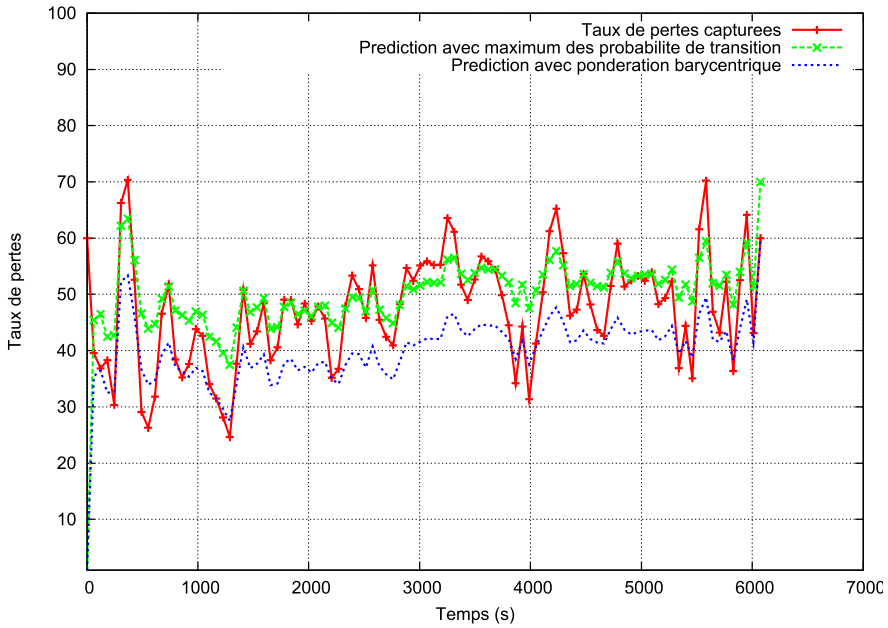

FIGURE 6.23 – Résultat de la prédiction

les pertes prédites et les pertes réelles, nous obtenons une valeur de 23,55 pour l'utilisation de la fréquence et 24,41 pour l'utilisation des moyennes barycentriques alors que l'adaptation basée sur les politiques effectue un excédent de 40 lorsqu'on injecte ces pertes. Cela signifie l'estimation du système de prédiction permet d'avoir une estimation correcte des taux de pertes futures, et donc de choisir, dans ce cas précis, des taux de redondances correctes à ajouter 2 paquets prêts sur 10 paquets contre 4 paquets prêts sur 10 paquets avec l'adaptation basée sur les politiques. Ceci, améliore, aussi l'adaptation car il permet de détecter les cas ou la FEC est inutile, c'est-à-dire, après des tests, une prédiction des pertes comprises entre $[0\%, 10\%[$ et $[60\%, 100\%[$. On serait tenté d'utiliser cette valeur connue à froid pour améliorer le processus, mais nous préférons laisser la méthode telle qu'elle, car cette valeur peut être étroitement liée aux cas des pertes que nous avons capturées (même si elles répresentent une réalité). Mais la simplicité de la méthode constitue un avantage non négligeable.

Cette section montre l'intérêt d'avoir, si possible, un mécanisme de prédiction du comportement de certains paramètres de performance car les décisions peuvent ainsi être affinées. Ces valeurs prédites pourraient, ainsi, être ajoutées à la base de connaissance et utilisés pour effectuer les actions.

6.6 Conclusion

Dans ce chapitre, nous avons présenté des cadres d'applications de l'approche que nous proposons. Une première discussion a montré la difficulté de simuler l'auto-adaptation à cause de la modification des configurations durant la simulation. En effet, les données fournies durant la description du scénario ne sont pas altérables durant la simulation au risque d'avoir une simulation instable. Après une comparaison sur un certain nombre de critères qui nous ont semblés importants, nous avons fait un choix par rapport à leurs caractéristiques actuelles. Nous avons étendu ce simulateur pour qu'il puisse prendre en charge le module de programmation logique inductive.

Nous avons aussi présenté un premier cas de test où l'application de la méthode d'auto-adaptation permet d'avoir des améliorations : *DiffServ*. Dans ce contexte, la mise en œuvre du module d'apprentissage pour l'auto-adaptation donne des résultats intéressants en terme d'amélioration du débit utile au sein d'un domaine *DiffServ*. Nous proposons ainsi un nouveau cadre de configuration des architectures DiffServ en automatisant l'adaptation des équipements. Les simulations dans cet environnement permettent aussi de mettre en évidence l'apport du processus de collaboration par partage de connaissance entre les éléments qui apprennent. Cette collaboration permet d'accélérer la vitesse de stabilisation des processus d'auto-adaptation avec un facteur qui décroit suivant la distance par rapport au nombre de sauts qui sépare le nœud considéré de la source du problème. Ainsi, elle montre que la vue située n'est intéressante que sur un nombre de sauts assez restreint.

Nous avons, ensuite, transposé notre approche dans le cadre de transmission de flux vidéo sur les réseaux sans-fils 802.11. Cette transposition pourrait permettre de mettre en place un système auto-adaptatif qui évalue de manière automatique l'efficacité des différents types d'adaptations proposées par [Djama 2008]. Cette transposition a été implémentée sur la plateforme IPTV mais n'a malheureusement pu être testé faute de pouvoir disposer des taux de pertes en retour.

Enfin, nous avons montré un cas où l'adaptation, enrichi d'un mécanisme de prédiction d'un des paramètres clé définissant les états du système. Plus précisément, nous avons montré comment la prédiction des taux de pertes permettrait d'améliorer un mécanisme d'adaptation basée sur la FEC (Forward Error Corrector) couplée

à une adaptation du débit d'envoi d'une vidéo. Cette amélioration permet de d'avoir un meilleur overhead tout en ayant la possibilité d'anticiper sur les pertes.

CHAPITRE 7

Conclusion et perspectives

La gestion autonome est de nos jours vue par le monde de la recherche et une partie de l'industrie comme la seule possibilité viable pour la gestion des réseaux du futur. Ceci parce que les réseaux ne cessent de se complexifier, de par la diversité des équipements terminaux qui s'y connectent, l'hétérogénéité des technologies de communication (filaire, sans fils, mobile, fixe), et, enfin, la multiplication des services qui sont de plus en plus exigeants en terme de qualité de service. Le concept de réseaux autonomes a pour objectif de résoudre le problème de la complexité de la gestion en rendant les équipements autonomes sous contrôle de politiques de haut niveau exprimant les objectifs des concepteurs.

Cette autonomie est réalisée à l'aide d'une boucle de contrôle adaptative qui permet aux équipements d'adapter leurs configurations aux changements de l'environnement. L'utilisation des systèmes cognitifs (apprentissage artificiel, raisonnement) pour réaliser cette boucle de contrôle fermée a reçu beaucoup d'attention de la part du monde de la recherche et de l'industrie. Beaucoup d'architectures de réseaux autonomes ont été proposées autour de cette idée qui porte le nom de plan de connaissance. Le plan de connaissance est une structure distribuée, hiérarchique et collaborative qui permet l'utilisation des systèmes cognitifs pour la gestion autonome des réseaux. Le concept de plan de connaissance est très important pour le domaine des réseaux autonomes car, dans la vision dominante, il se veut concentrer toute l'intelligence des architectures. De nos jours, plan de connaissance et réseaux autonomes sont des domaines inséparables. La plupart des architectures autonomes intègrent plus ou moins un plan de connaissance pour réaliser l'auto-adaptation des équipements réseaux. Cependant, peu d'études ont été effectuées pour déterminer le

rôle réel que pourrait jouer l'apprentissage. L'objectif de cette thèse était d'identifier et de proposer une solution orientée apprentissage qui réponde à la problématique de l'auto-adaptation dans le cadre du plan de connaissance, afin de réaliser l'autonomie.

Contributions Les principales contributions de cette thèse peuvent être regroupées en six points :

1- Formalisation du problème d'auto-adaptation Nous avons proposé une formalisation abstraite du problème de l'auto-adaptation, qui s'appuie sur des réalités réseaux. Cette formalisation implique un effort pour identifier les différents paramètres qui interviennent dans la problématique. Entre autre, nous avons identifié : un critère de performance P_r qui est nécessaire pour que le système soit capable de s'auto-évaluer. Le critère de performance, notamment, définit l'ensemble des paramètres qui doivent être évalués et améliorés par l'adaptation. Nous avons aussi introduit la notion d'ambiguïté sur un critère P_r. L'ambiguïté d'un critère de performance consiste à ne pas décrire les préférences entre les états de l'équipement de manière exhaustive. Elle empêche l'équipement d'apprécier l'efficacité de certaines actions sur les états. Le problème de l'ambiguïté est important à régler car il influe sur la décidabilité du problème et donc sur la faisabilité de l'automatisation de l'adaptation. Nous avons proposé également une définition des états du système comme étant des partitions de l'espace des informations qui peuvent être remontées. Ce partitionnement permet de se ramener à un ensemble d'états fini pour que l'apprentissage puisse se stabiliser en un temps fini. Nous avons ainsi défini le problème de l'auto-adaptation comme le processus qui consiste à trouver pour l'ensemble des états du système un ensemble d'actions qui permettent d'améliorer, les états suivant le critère de performance P_r à définir par le concepteur dans un environnement donné E_v.

Cette définition de haut niveau permet de prendre en compte toute la problématique des réseaux qui peut être modélisée en terme d'état/action ou événement/action. Ainsi, la méthode que nous avons proposée aura une large portée en termes d'applications.

2- Identification des contraintes de la problématique et étude des outils d'apprentissage Cette formalisation simple permet de faire apparaitre les différentes contraintes du problème :

- L'incrémentalité, les exemples sont disponibles au fur et à mesure ;
- La nécessité que l'outils d'apprentissage découvre l'environnement ;
- La non disponibilité à priori d'une fonction de transition qui permettrait de faire de la programmation dynamique classique ;
- Le non déterminisme des environnements auxquel s'adaptent les équipements ;
- Le caractère plus ou moins collaboratif qui permettrait de profiter de la distribution des équipements pour optimiser les ressources.

Ceci constitue une base intéressante pour faire une étude de l'ensemble des outils d'apprentissage qui peuvent répondre à la question de l'auto-adaptation. Notre étude des outils d'apprentissage nous a conduit à l'apprentissage par renforcement dont la problématique est la plus proche de celle de l'auto-adaptation. Cependant, nous avons présenté un certain nombre de contraintes liées à l'utilisation de ce type d'apprentissage. De manière résumée :

- L'apprentissage par renforcement dépend fortement d'une fonction de récompense. Si la fonction de récompense ne reflète pas exactement ce que veut le concepteur du système, l'apprentissage maximisera la récompense mais la tâche sera mauvaise. Or, dans le domaine des réseaux, une telle fonction n'existe pas. On peut faire le choix de fournir la fonction a priori ($+1$ ou -1), mais on perd la garantie d'aboutir à un apprentissage efficace. De même, il faudrait définir une fonction de récompense pour chaque environnement.
- L'apprentissage par renforcement n'est rapide en terme de convergence que dans des cas très précis (processus de décision markovien déterministe notamment). Et même pour ces cas, la convergence se fait à l'infini. Dans le cas de processus stochastique et/ou de mécanisme de récompense non déterministe le nombre de tests nécessaire peut être assez grand. Cette situation présente un risque majeur de faire échouer ce type d'apprentissage.

Par conséquent, la première conclusion est que la ressemblance de la problématique d'une technique d'apprentissage ne garantit pas forcement la possibilité de l'appliquer. Il serait difficilement imaginable d'avoir un apprentissage par renforcement qui convienne à toutes les situations. Comme alternative à cette approche,

nous proposons une approche différente basée sur la programmation logique inductive.

3- Relation entre historique et probabilité de réussite Dans la problématique de l'auto-adaptation, nous n'avons à disposition, et de manière incrémentale, que les transitions comme information sur l'efficacité des actions sur les états. Or, dans le cas idéal, il faudrait avoir une bonne estimation des probabilités de réussite des couples état-actions. A priori, cette probabilité est inconnue, mais nous avons montré sa relation avec les statistiques sur les transitions observées. S'il existe une loi de probabilité qui définit l'efficacité d'une action sur un état donné, alors le rapport entre le nombre d'observations positives et le nombre total d'observations relatives à un couple état-action, est une bonne estimation de cette efficacité pour un nombre représentatif de transitions positives. En d'autres termes, l'utilisation de l'historique des transitions donne une bonne idée sur l'efficacité des actions sur les états si nous avons un grand nombre d'actions. Cet historique nous permet de nous passer de la notion de récompense liée à l'apprentissage par renforcement. Cependant, pour avoir une valeur correcte nous avons besoin d'un nombre minimum d'observations. Pour nous assurer de cela, nous utilisons le régulateur de Laplace.

4- Apprentissage adaptatif avec programmation logique inductive Dans un environnement variant très peu c'est-à-dire où le système ne fait que des transitions vers un ensemble d'états très limité, il peut exister un grand nombre d'états pour lesquels le système n'a pas de stratégies pour s'adapter a l'ensemble des états qui ne sont presque jamais visités. En cas d'apparition de ces états, le choix des actions peut se faire de manière complètement arbitraire, ce qui peut ralentir l'apprentissage à cause des tests supplémentaires. Nous introduisons l'utilisation des régulateurs ou coefficients de Laplace qui garantissent l'utilisation de probabilités à priori jusqu'à ce que l'historique atteigne une certaine taille.

En plus, nous proposons l'utilisation de la programmation logique inductive pour apprendre des stratégies de configurations à partir de stratégies existantes. Nous rappelons que la programmation logique inductive , à partir d'une connaissance de base B, d'un ensemble d'exemples positifs E^+ et négatifs E^-, trouve une hypothèse H qui couvre tous les exemples positifs et aucun des exemples négatifs. L'utilisation que

nous proposons de la PLI est celle de présenter le concept à apprendre comme celui d'ensemble de couples état/actions. L'ensemble des exemples positifs est l'ensemble des couples état-actions qui ont été appris comme efficaces et les exemples négatifs sont des couples état-actions apprises comme étant inefficaces. La PLI permettra, d'apprendre une hypothèse, une règle plus générale qui décrit les exemples positifs (stratégies efficaces) et ne couvre aucun exemple négatif (stratégies inefficaces). L'hypothèse produite est sous forme de programme logique, ce qui va permettre par déduction, de trouver un ensemble de règles qui peuvent être induites de règles déjà établies. Ceci va permettre d'orienter le choix de l'adaptation vers des actions qui ont une plus grande chance de réussir du fait de l'induction à partir de stratégies déjà éprouvées.

5- Mécanisme d'échange de connaissance et collaboration L'utilisation de la PLI pour apprendre de nouvelles stratégies va faire apparaitre un ensemble de nouvelles stratégies qui ne sont pas labellisées (n'ayant pas d'appréciation efficace ou non)

Nous introduisons en plus dans le processus d'apprentissage une collaboration dont l'objectif est multiple :

- L'initialisation des nouveaux éléments qui arrivent dans le réseau. Cette initialisation peut s'avérer plus efficace que l'utilisation des indicateurs de Laplace bien que très utiles ;
- Le rééquilibrage de la distribution des exemples entre les différentes parties du réseau. Les éléments qui sont dans des zones très dynamiques vont apprendre très rapidement alors que les éléments situés dans les zones "calmes" vont apprendre plus lentement. L'échange d'observations et de stratégies peut permettre de rééquilibrer cela et ainsi permettre d'accélérer le processus d'apprentissage dans les zones calmes ;
- La validation distribuée de la connaissance. Les éléments qui collaborent vont critiquer mutuellement les connaissances apprises par les voisins à travers l'échange de connaissance et d'observations négatives.
- La gestion distribuée de la connaissance. Chaque élément garde au niveau local uniquement les connaissances nécessaires à son adaptation, tout en échangeant avec ses voisins. La connaissance est ainsi distribuée dans le réseau

uniquement dans les zones où elles seront utiles ;

Nous proposons un algorithme de propagation, qui sans faire exploser le nombre de messages dans le réseau permet cet échange de la connaissance. L'algorithme réalise une vue située sous forme de diffusions sur un arbre recouvrant.

Applications dans les environnements IP de nouvelle génération : Nous avons réalisé une application de la méthode proposée dans le cadre des réseaux IP de nouvelle génération. Nous avons montré l'efficacité de l'utilisation d'un module d'apprentissage dans le cadre d'un réseau *DiffServ*. Elle permet entre autre d'améliorer le débit au sein d'un domaine *DiffServ* en permettant aux routeurs de s'adapter aux variations des taux de pertes. L'algorithme, dans les conditions de simulations, se stabilise en trouvant assez rapidement les stratégies les mieux adaptées pour configurer les routeurs. Ces simulations ont montré l'intérêt que peut avoir la collaboration entre plusieurs éléments en échangeant leurs connaissances. La collaboration apporte une amélioration tant qu'elle ne propage pas une stratégie à plus de 6 sauts du point le plus proche de l'élément qui a généré le problème. Nous avons ensuite montré comment cette approche pouvait être transposée dans le cadre du transport de flux multimédia sur des réseaux sans-fils 802.11. Cette transposition est la continuité d'un travail effectué durant sa thèse par [Djama 2008]. Cette thèse présente plusieurs adaptations qui permettent d'améliorer la qualité de la vidéo reçue. La plupart des adaptations ont été testées indépendamment. Notre approche peut constituer un bon moyen d'évaluer la coexistence des différentes adaptations.

Enfin, nous avons montré un cas de figure où la mise en œuvre d'un mécanisme de prédiction simple peut permettre d'améliorer l'*overhead* d'une adaptation en utilisant la prédiction d'un paramètre de performance pour affiner le choix des actions à effectuer.

Perspectives La solution présentée peut encore être améliorée sur plusieurs aspects :

Cohérence locale et cohérence globale Dans l'état actuel, le système gère la cohérence locale de l'équipement. Il y a deux types de cohérence : la cohérence logique qui est en relation avec la programmation logique et la cohérence

sémantique qui est liée à l'efficacité réelle de l'application. Cependant, nous avons implicitement fait l'hypothèse selon laquelle la cohérence locale sur chaque nœud pris individuellement permet d'avoir un comportement globalement efficace. Cette hypothèse est sous-tendue par le fait qu'on ne peut pas forcer un ensemble d'équipements dont on ignore, parfois, la variété des différents contextes dans lesquels il se trouvent, à apprendre les mêmes stratégies d'adaptation. Une amélioration importante serait celle d'étendre le système pour qu'il apprenne automatiquement à détecter des régions uniformes en terme de contexte et à y imposer une cohérence globale.

Dépendance par rapport au nombre de transitions La solution proposée se base sur le nombre de transitions. Ceci implique qu'il n'y a pas de vraie corrélation entre le temps de vie du système et sa stabilisation. Si les transitons sont très éloignées, c'est-à-dire un environnement est assez monotone, l'apprentissage sera lent et pauvre. Et il peut se passer un temps assez long avant que le système ne détecte que l'environnement est monotone. Une amélioration pourrait être d'avoir un mécanisme qui détecte dès le début quand un environnement est monotone afin de prédire son comportement pour anticiper sur les événements de l'environnement.

Stabilité du système L'adaptation que nous proposons, même si nous avons observé une stabilisation du système dans le cas *DiffServ*, ne garantit pas une stabilité dans tous les environnements. Pour le moment, nous ne sommes pas en mesure de nous prononcer sur sa stabilité dans un cadre général. Tout ce qui peut être énoncé c'est que si le type d'activités sur le réseau produit assez d'observations, le système va trouver la loi qui le sous-tend et se stabiliser. Cette difficulté vient du fait que dans la stabilité du système intervient l'activité qu'il y a dans le réseau. Or, il aurait fallut étudier tous les types d'activités possibles et toutes les configurations et contextes. Il pourrait être intéressant de trouver une approche analytique qui permette d'établir sa stabilité ou non. D'autre part, nous n'avons pas pu tester la sensibilité de l'approche proposée en présence d'un très grand nombre d'états. Dans l'environnement *DiffServ* testé, en tentant de découper l'ensemble des pertes en intervalles plus petites, les transitions étant peu diversifiées, nous avons un très grand nombre d'états qui ne sont jamais rencontrés, et on se retrouve presque dans

le cas présenté. Il serait intéressant de trouver un cas où la multiplication des états prend tout son sens pour évaluer les performances de l'approche par rapport à la taille de l'ensemble des états (Σ).

Apprentissage pour l'anticipation Le système que nous avons proposé est un système réactif par rapport aux changements de l'environnement. Il serait intéressant de regarder le cas d'un système qui pourrait identifier les types de flux à la volée pour pouvoir prédire leur comportement et ainsi mettre en place des déclencheurs d'opérations de configuration. Ceci peut être particulièrement intéressant dans les flux qui présentent une certaine régularité. Comme exemple nous pouvons citer le cas des trafics *self-simular* qui est le modèle sur Internet, si le système est capable d'identifier le motif qui se répète à petite échelle et le facteur d'agrandissement pour une échelle supérieure, il sera capable d'avoir l'ensemble du graphe de transition dans le futur. Dans ce cas, un calcul à froid par programmation dynamique permet d'avoir un ensemble de stratégies avec la garantie de l'optimalité.

Programmation logique inductive et auto-protection La programmation logique inductive pourrait également servir à réaliser d'autres auto-fonctions comme l'auto-protection en particulier des systèmes de détection d'intrusions (IDS). En effet, le domaine des systèmes de détection d'intrusions peut être divisé en deux : les systèmes basés sur les signatures et ceux basés sur la détection d'anomalies. Dans le cas du système de détection par signature, il y a une base de données d'intrusions qui est mise à jour en fonction de l'identification des signatures. Une signature est un ensemble de règles définissant les prémisses d'une attaque particulière. Avec une formulation bien faite, la programmation logique inductive pourrait permettre après l'identification d'une nouvelle intrusion de générer automatiquement sa signature en trouvant une règle générale définissant l'ensemble des traces observées. Cela ne permettrait certes pas encore de détecter les intrusions la première fois qu'elle surviennent, mais d'automatiser la génération de sa signature.

Raisonnement collaboratif dans les réseaux de capteurs Durant tout ce rapport, nous avons surtout insisté sur l'aspect apprentissage des systèmes cognitifs et moins sur le rôle du raisonnement, même s'il est quasi impossible de

dissocier les deux [Dietterich 2004]. Le raisonnement est la capacité à faire des inférences [of Philosophy SEP 2005] (raisonnement déductif). Elle peut permettre de mettre en place des systèmes de prise de décision pour orienter un processus. Or, dans les réseaux de capteurs, l'économie d'énergie et l'auto-organisation sont très importantes. Il serait intéressant de voir comment profiter des outils de raisonnement et du mécanisme de collaboration pour faire prendre aux équipements des décisions locales et distribuées pour ne remonter vers les puits que les données réellement pertinente. Ceci, aurait comme avantage de diminuer le nombre de messages dans le réseau car les équipement ne décideront de communiquer ou collaborer que lorsque c'est nécessaire. Ceci facilitera également le déploiement, les capteurs s'auto-organisant. Nous sommes entrain de travailler sur cette approche dans le cadre du projet ANR Diaforus.

Bibliographie

[4WARD 2009] 4WARD. *The Network of the Future : 4WARD - Architecture and Design for the Future Internet*, 05 2009. http ://www.4ward-project.eu/index.php ?s=Deliverables&c=project dernier accès 10 octobre 2009. 33

[Abid *et al.* 2008] M. Abid, A. Berl et al. *Autonomic Internet Initial framework.* Deliverable d6.1, Autonomic Internet (autoi) project, August 2008. 33

[Agoulmine *et al.* 2006] N. Agoulmine, S. Balasubramaniam et Al. *Challenges for Autonomic Network Management.* In 1st IEEE International Workshop on Modeling Autonomic Communications Environments (MACE), 2006. 22

[Anderson 2003] Michael L. Anderson. *Embodied Cognition : A field guide.* Artificial Intelligence, vol. 149, no. 1, pages 91–130, September 2003. 47

[AutoI 2008] AutoI. *AutoI Project*, 2008. site officiel http ://ist-autoi.eu/ , flyer du projet recuperable à l'adresse ftp ://ftp.cordis.europa.eu/pub/fp7/ict/docs/future-networks/projects-autoi-factsheet/_en.pdf dernier accès 23 juin 2009. 33

[Autonomia 2007] Project Autonomia. *Autonomia : Autonomic Control and Management Environment*, 2007. page du projet à http ://acl.ece.arizona.edu/projects/Autonomia_Programmable/ dernier accès le 23 Aout 2009. 34

[Ballani & Francis 2006] Hitesh Ballani et Paul Francis. *CONMan : taking the complexity out of network management.* In INM '06 : Proceedings of the 2006 SIGCOMM workshop on Internet network management, pages 41–46, New York, NY, USA, 2006. ACM. 56

[Baumgarten *et al.* 2007] Matthias Baumgarten, Nicola Bicocchi, Rico Kusber, Maurice D. Mulvenna et Franco Zambonelli. *Self-organizing knowledge networks for pervasive situation-aware services.* In SMC, pages 1–6. IEEE, 2007. 68

[Bieber & Carpenter 2003] Guy Bieber et Jeff Carpenter. *OpenWings : A Service-Oriented Component Architecture for Self-Forming, Self-Healing,*

Network-Centric Systems (Rev 2.0), 2003. accessible à l'adresse http ://www.openwings.org/download/specs/openwingswp.pdf dernier accès 24 juin 2009. 34

[BIO-NETWORKING 2007] Project BIO-NETWORKING. *THE BIO-NETWORKING ARCHITECTURE*, 2007. accessible à l'adresse http ://-netresearch.ics.uci.edu/bionet/ dernier accès 23 juin 2009. 34

[Blake *et al.* 1998] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang et W. Weiss. *RFC 2475 : An architecture for differentiated services.* 1998. http ://www.ietf.org/rfc/rfc2475.txt. 158, 160, 161, 165

[Bonifacio *et al.* 2002] M. Bonifacio, P. Bouquet, G. Mameli et M. Nori. *Kex : a peer-to-peer solution for distributed knowledge management.* In Fourth International Conference on Practical Aspects of Knowledge Management (PAKM-2002, pages 490–500, 2002. 74

[Bonifacio *et al.* 2004] Matteo Bonifacio, Paolo Bouquet, Alberto Danieli, Antonia Donà, Gianluca Mameli et Michele Nori. *KEEx : A Peer-to-Peer Solution for Distributed Knowledge Management.* In Proceedings of I-KNOW '04, Graz, Austria, June 30 - July 2, 2004. 74

[Boscovic 2005] Dragan Boscovic. *Cognitive Networks, MOTOROLA TECHNOLOGY POSITION PAPER.* Rapport technique, MOTOROLA, dernier accès 16 juin 2009, 2005. 68, 69

[Bourgne *et al.* 2007] Gauvain Bourgne, Amal El Fallah Seghrouchni et Henry Soldano. *SMILE : Sound Multi-agent Incremental LEarning ;-).* In International Conference on Autonomous Agents and Multiagent Systems, (AAMAS), pages 164–171, Honolulu, Hawaï, 5 2007. 140

[BOUTABA *et al.* 2001] Raouf BOUTABA, Salima OMARI et S. Virk. *SELFCON : An architecture for self-configuration of networks.* Journal of communication and networks, vol. 3, pages 317–323, 2001. 34

[Brachman & Levesque 2004] Ron Brachman et Hector Levesque. Knowledge representation and reasoning. Elsevier, 2004. 58

[Brachman 2002] Ronald J. Brachman. *Systems That Know What They're Doing.* IEEE Intelligent Systems, vol. 17, no. 6, pages 67–71, 2002. 48

[Bradbury *et al.* 2004] Jeremy S. Bradbury, James R. Cordy, Juergen Dingel et Michel Wermelinger. *A survey of self-management in dynamic software architecture specifications.* In WOSS '04 : Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems, pages 28–33, New York, NY, USA, 2004. ACM. 91

[Brown & Patterson 2001] A. Brown et D. Patterson. *To err is Human.* In Proceedings of First workshop of Evaluation and Architecture of dependanbility EASY'01, 2001. 2, 13

[Brügge *et al.* 2004] B. Brügge, P. Renner, M. Strassberger et M. Adamski. *MEDUSA - Framework for the Secure Peer-To-Peer Sharing of Topic Map based Knowledge.* In Conference on knowledge sharing and collaborative engineering (KSCE 2004), 2004. 74

[Bullot *et al.* 2008] Thomas Bullot, Rida Khatoun, Louis Hugues, Dominique Gaïti et Leila Merghem-Boulahia. *A situatedness-based knowledge plane for autonomic networking.* Int. J. Netw. Manag., vol. 18, no. 2, pages 171–193, 2008. 73

[Campbell *et al.* 2005] Jason Campbell, Phillip B. Gibbons, Suman Nath, Padmanabhan Pillai, Srinivasan Seshan et Rahul Sukthankar. *IrisNet : an internet-scale architecture for multimedia sensors.* In MULTIMEDIA '05 : Proceedings of the 13th annual ACM international conference on Multimedia, pages 81–88, New York, NY, USA, 2005. ACM. 65, 93

[Carbonell *et al.* 1983] J. Carbonell, R.S. Michalski et T.M. Mitchell. *An Overview Of Machine Lerning.* Machine Learning, An artificial Intelligence approach, vol. III, pages 3–23, 1983. 92

[CASCADAS 2006] Project CASCADAS. *Component-ware for Autonomic Situation-aware Communications, and Dynamically Adaptable Services*, 2006. page officielle http ://www.cascadas-project.org/ dernier accès 23 juin 2009. 32

[Christensen & Nagel 2006] Henrik I. Christensen et Hans-Hellmut Nagel. *Cognitive Vision Systems, Sampling the Spectrum of Approaches [based on a Dagstuhl seminar].* In Cognitive Vision Systems, volume 3948 of *Lecture Notes in Computer Science*. Springer, 2006. 47

[Chun *et al.* 2004] Brent Chun, Joseph M. Hellerstein, Ryan Huebsch, Petros Maniatis et Timothy Roscoe. *Design Considerations for Information Planes.* In IN WORLDS'04, 2004. 63

[Clark *et al.* 2003a] David Clark, Karen Sollins, Robert Braden, Ted Faber et Mark Handley. *NewArch Final Technical Report.* In MIT LCS et ICSI) (DARPA USC/ISI, editeur, NewArch Project : Future-Generation Internet Architecture, 2003. http ://www.isi.edu/newarch/. 40, 41

[Clark *et al.* 2003b] David D. Clark, Craig Partridge, J. Christopher Ramming et John T. Wroclawski. *A knowledge plane for the internet.* In SIGCOMM '03 : Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pages 3–10, New York, NY, USA, 2003. ACM. 4, 42, 44, 50, 52, 54, 68, 74

[Cornuéjols *et al.* 2003] Antoine Cornuéjols, Laurent Miclet et Yves Kodratoff. Apprentissage artificiel, concepts et algorithmes. Eyrolles collection Algorithmes, 2003. 92

[Curran *et al.* 2007] K. Curran, M. Mulvenna, C. Nugent et A. Galis. *Challenges and research directions in autonomic communications.* volume 2, pages 3–17, Inderscience Publishers, Geneva, SWITZERLAND, 2007. Inderscience Publishers. 34

[Davis *et al.* 1993] Randall Davis, Howard E. Shrobe et Peter Szolovits. *What Is a Knowledge Representation ?* AI Magazine, vol. 14, no. 1, pages 17–33, 1993. 58

[Dietterich & Langley 2003] Tomas Dietterich et Pat Langley. *Machine Learning for Cognitive networks : Technology Assessment and Research Challenge.* Rapport technique, Oregon State University and Institue of Study Learning and Expertise, 2003. 61

[Dietterich 2004] T. G. Dietterich. *Learning and Reasoning.* Rapport technique, School of Electrical Engineering and Computer Science, Oregon State University, 2004. 205

[Djama 2008] Ismail Adel Djama. *Adaptations inter-couches pour la diffusion des services vidéo sans fil.* PhD thesis, UNIVERSITÉ BORDEAUX I, 2008. 178, 181, 184, 187, 188, 189, 190, 191, 192, 194, 202

[DMTF 1999] DMTF. *COMMON INFORMATION MODEL (CIM) SPECIFICATION version 2.2*, 1999. 59

[Dobson *et al.* 2006] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt et Franco Zambonelli. *A survey of autonomic communications.* ACM Trans. Auton. Adapt. Syst., vol. 1, no. 2, pages 223–259, 2006. 34

[Dzeroski & Lavrac. 1991] 6. S. Dzeroski et N. Lavrac. *Learning relations from noisy examples : An empirical comparison of LINUS and FOIL.* In In L. Birnbaum et editors G. Collins, editeurs, Proceedings of the 8th International Workshop on Machine Learning, ., pages 399–402. Morgan Kaufmann, 1991. 120

[Fahy *et al.* 2008] C. Fahy, S. Davy, Z. Boudjemil et al. *Towards an Information Model That Supports Service-Aware, Self-managing Virtual Resources.* In MACE'08 : Proceedings of the 3rd IEEE international workshop on Modelling Autonomic Communications Environments, pages 102–107, Berlin, Heidelberg, 2008. Springer-Verlag. 56

[Florian 2003] R. V. Florian. *Autonomous artificial intelligent agents.* Rapport technique, Technical Report Coneural-03-01, 2003. 73

[Franklin & Graesser 1996] Stan Franklin et Art Graesser. *Is it an Agent, or just a Program ? : A Taxonomy for Autonomous Agents.* In Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996. 73

[Friend *et al.* 2007] D. H. Friend, R. W. Thomas, A. B. MacKenzie et L. A. DaSilva. *Distributed Learning and Reasoning in Cognitive Networks(Book chapter).* Cognitive Networks - Towards Self-Aware Networks by Qusay H. Mahmoud, ed. , John Wiley & Sons, vol. 1, pages 224–246, 2007. 61, 70

[Fujitsu-Siemens 2003] Fujitsu-Siemens. *Autonomic Systems : Concept for self-managing IT infrastructures*, 2003. White Paper accessible à l'adresse http ://sysdoc.doors.ch/FUJITSUSIEMENS/autkonz-3030e.pdf dernier accès 23 juin 2009. 33

[Garfield & Weisler 1987] J. L. Garfield et S. E. Weisler. *Semantics.* In N. A. Stillings, M. H. Feinstein, J. L. Garfield, E. L. Rissland, D. A. Rosenbaum, S. E. Weisler et L. Baker-Ward, editeurs, Cognitive Science : An Introduction, pages 389–421. MIT Press, Cambridge, MA, 1987. 47

[Giordano & Puiatti 2006] Silvia Giordano et Alessandro Puiatti. *Specification of first Haggle application and INFANT-Haggle*, 2006. accessible à l'adresse http ://www.haggleproject.org/images/b/b0/D11_final.pdf dernier accès 23 juin 2009. 33

[Greene & Lancaster 2007] Wedge Greene et Barbara Lancaster. *Autonomic Networks - Autonomic Communication.* Pipelinepub.com book Automation, Volume 4 issue 3, Aout 2007. 34

[Greer *et al.* 2008] Kieran Greer, Matthias Baumgarten, Chris Nugent, Maurice Mulvenna et Kevin Curran. *Autonomous Querying for Knowledge Networks.* In ATC '08 : Proceedings of the 5th international conference on Autonomic and Trusted Computing, pages 249–263, Berlin, Heidelberg, 2008. Springer-Verlag. 67

[GTNetS 2008] GTNetS. *Georgia Tech Network Simulator (GTNetS) website*, 2008. http ://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/. 151

[Heinanen *et al.* 1999] J. Heinanen, F. Baker, W. Weiss et J. Wroclawski. Rfc 2597 : Assured forwarding phb group. http ://rfc.dotsrc.org/rfc/rfc2597.html, 1999. 160

[Hitachi 2007] Hitachi. *Harmonous Computing*, 2007. accessible à la page http ://www.hitachi.co.jp/products/harmonious/center/gl/main/information/concept/index.html. 33

[Hollnagel & Woods 1999] E. Hollnagel et D. D. Woods. *Cognitive systems engineering : New wind in new bottles.* International Journal of Human-Computer Studies, vol. 51, no. 1, pages 339–356, September 1999. 48

[Horn 2001] Paul Horn. *Autonomic computing : IBM's Perspective on the State of Information Technology*, 2001. 14, 17

[Huebsch *et al.* 2003] Ryan Huebsch, Joseph M. Hellerstein, Nick Lanham, Boon Thau Loo, Scott Shenker, Loo Scott Shenker et Ion Stoica. *Querying the Internet with PIER.* In In VLDB, pages 321–332, 2003. 65

[Hung-Ying] T. Hung-Ying. *J-SIM home page.* please see home Page http ://www.j-sim.org/. 168

[IEEE 2009] IEEE. *Standard Upper Ontology Working Group (SUO WG)*, 2009. Site officielle http ://suo.ieee.org/. 59

[Institute 1981] Information Sciences Institute. *RFC 791 INTERNET DARPA INTERNET PROTOCOL SPECIFICATION*, September 1981. University of Southern California http ://www.ietf.org/rfc/rfc791.txt. 157

[IPERF 2008] IPERF. *Site officiel d'Iperf*, 2008. http ://iperf.sourceforge.net/. 191

[J-SIM 2006] J-SIM. *J-Sim Official Website*, 2006. http ://-sites.google.com/site/jsimofficial/. 151

[Jacob *et al.* 2002] B. Jacob, R. Lanyon-Hogg, D. K. Nadgir et A. F. Yassin. *A practical guide to the ibm autonomic computing toolkit.* In Proceedings of 18th International Conference on Logic Programming (ICLP 2002). IBM, 2002. 17, 22, 30

[Jacobson *et al.* 1999] V. Jacobson, K. Nichols et K. Poduri. *RFC 2598 : An Expedited Forwarding PHB.* In RFC editor, 1999. http ://www.ietf.org/rfc/rfc2598.txt. 159

[Jones *et al.* 2008] Dominic Jones, John Keeney, David Lewis et Declan O'Sullivan. *Knowledge-based networking.* In Proceedings of the Second International Conference on Distributed Event-Based Systems, DEBS 2008, Rome, Italy, July 1-4, 2008, ACM International Conference Proceeding Series, pages 329–332. ACM, 2008. 74

[Kephart & Chess 2003] J. O. Kephart et D. M. Chess. *The Vision of Autonomic Computing.* volume 36, pages 41–50, Los Alamitos, CA, USA, January 2003. IEEE Computer Society Press. 17, 19, 22, 28, 30

[Konstantinou & Yemini 2003] Alexander V. Konstantinou et Yechiam Yemini. *Programming Systems for Autonomy.* In amsw, pp.186, Autonomic Computing Workshop Fifth Annual International Workshop on Active Middleware Services (AMS'03), 2003. 34

[Kopena & Loo 2008] Joseph B. Kopena et Boon Thau Loo. *OntoNet : Scalable knowledge-based networking.* In Proceedings of the 24th International

Conference on Data Engineering Workshops, ICDE 2008, April 7-12, 2008, Cancún, México, pages 170–175. IEEE Computer Society, 2008. 56

[Langley 1995] Pat Langley. Elements of machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995. 92

[Latré *et al.* 2008] Steven Latré, Pieter Simoens, Wim Van de Meerssche, Bart De Vleeschauwer, Filip De Turck, Bart Dhoedt, Piet Demeester, Steven Van den Berghe et Edith Gilon de Lumley. *Automated Generation of Knowledge Plane Components for Multimedia Access Networks.* In Modelling Autonomic Communications Environments, Third IEEE International Workshop, MACE 2008, Samos Island, Greece, September 22-26, 2008. Proceedings, Lecture Notes in Computer Science, pages 50–61. Springer, 2008. 57

[Lavrac & Dzeroski 1994] N. Lavrac et S. Dzeroski. Inductive logic programming : Techniques and applications. Ellis Horwood, New York, 1994. 120

[Lavrač *et al.* 1996] N. Lavrač, S. Džeroski et I. Bratko. *Handling Imperfect Data in Inductive Logic Programming.* In L. De Raedt, editeur, Advances in Inductive Logic Programming, pages 48–64. IOS, 1996. 120

[Lewis 2004] David Lewis. *Panel Report : "How the Autonomic Network Interacts with the Knowledge Plane ?".* In Autonomic Communication, First International IFIP Workshop, WAC 2004, Berlin, Germany, October 18-19, 2004, Revised Selected Papers, pages 275–278, 2004. 74, 75

[Li 2007] J. Li. *Agent Organization and Request Propagation in the Knowledge Plane.* Rapport technique, CSAIL Technical Reports., 2007. 91

[Lymberopoulos 2002] Leonidas A. Lymberopoulos. *An Adaptive Policy Based Management Framework for Network Services Management.* PhD thesis, Imperial College London, 2002. http ://www.doc.ic.ac.uk/ llymber/downloads/PhD-Tranfer-llymber.pdf. 162

[Macedo *et al.* 2007] D. R. Macedo, A. L. dos Santos, G. Pujolle et J. M. S. Nogueira. *Optimizing wireless ad hoc communication resources with a knowledge plane.* In Proc. 9th IFIP International Conference on Mobile Wireless Communications Networks, pages 86–90, 19–21 Sept. 2007. 56

[Macedo *et al.* 2008] D. F. Macedo, A. L. dos Santos, G. Pujolle et J. M. S. Nogueira. *MANKOP : A Knowledge Plane for wireless ad hoc networks.* In

Proc. IEEE Network Operations and Management Symposium NOMS 2008, pages 706–709, 7–11 April 2008. 56

[Madhyastha *et al.* 2006] Harsha V. Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy et Arun Venkataramani. *iPlane : an information plane for distributed services.* In OSDI '06 : Proceedings of the 7th symposium on Operating systems design and implementation, pages 367–380, Berkeley, CA, USA, 2006. USENIX Association. 64, 65

[Marbukh 2004] Vladimir Marbukh. *A knowledge plane as a pricing mechanism for aggregate, user-centric utility maximization.* SIGMETRICS Perform. Eval. Rev., vol. 32, no. 2, pages 22–24, 2004. 57

[McCloghrie & Rose 1991] K. McCloghrie et M. Rose. *Management Information Base for Network Management of TCP/IP-based internets : MIB-II.* Request for Comments : 1213, 1991. 59

[McHenry & Steadman 2005] Mark A. McHenry et Karl Steadman. *Spectrum Occupancy Measurements, Location 1 of 6 : Riverbend Park, Great Falls, Virginia.* Rapport technique, Shared Spectrum Company, 2005. http ://www.sharedspectrum.com/inc/content/measurements/, Dernier accès 19 juin 2009. 70

[Menascé *et al.* 2001] Daniel A. Menascé, Daniel Barbará et Ronald Dodge. *Preserving QoS of e-commerce sites through self-tuning : a performance model approach.* In EC '01 : Proceedings of the 3rd ACM conference on Electronic Commerce, pages 224–234, New York, NY, USA, 2001. ACM. 34

[Microsoft 07] Microsoft. *Dynamic Systems Initiative (DSI)*, 07. page officielle http ://www.microsoft.com/business/dsi/default.mspx dernier accès 23 juin 2009. 33

[Mitchell 1997] Tom M. Mitchell. Machine learning. McGraw-Hill, New York, 1997. 93, 97, 98, 99

[Mitola III 2000] Joseph Mitola III. *Cognitive Radio : An Integrated Agent Architecture for Software Defined Radio.* PhD thesis, Royal Institute of Technology, Swenden, 2000. Available at http ://web.it.kth.se/ maguire/jmitola/ last visit 16th june 2009. 70

[Mooney 2004] Raymond J. Mooney. *Machine Learning*. The Oxford handbook of computational linguistics, vol. 1, pages 376–394, 2004. 92

[Muggleton & Feng 1990] S. Muggleton et C. Feng. *Efficient induction of logic programs*. In Proceedings of the 1st Conference on Algorithmic Learning Theory, pages 368–381. Ohmsma, Tokyo, Japan, 1990. 120

[Muggleton & Raedt 1994] S. Muggleton et L. D. Raedt. *Inductive Logic programming : theory and application*. In Journal of logic programming, 1994. 112

[Muggleton 1995] S. Muggleton. *Inverse Entailment and Progol*. New Generation Computing, Special issue on Inductive Logic Programming, vol. 13, no. 3-4, pages 245–286, 1995. 120

[Muggleton 1997] Stephen Muggleton. *Learning from Positive Data*. In ILP '96 : Selected Papers from the 6th International Workshop on Inductive Logic Programming, pages 358–376, London, UK, 1997. Springer-Verlag. 120

[Muggleton 2002] S. Muggleton. Inductive machine learning new generation computing. 8(4) :295-318., July 29-August 01 2002. 120

[Mulvenna *et al.* 2006] Maurice Mulvenna, Franco Zambonelli, Kevin Curran et Chris Nugent. *Knowledge Networks*. book chapter in LNCS Autonomic Communication, Subject Collection Computer Science, Springer, vol. Volume 3854/2006, pages 99–114, 2006. 67, 68

[N. Samaan 2005] A. Karmouch N. Samaan. *An automated Policy-Based Management Framework for Differentiated Comminication Systems*. In IEEE journal on selected areas in Communications, volume 23, December 2005. 162, 163

[Nguengang *et al.* 2008] Gérard Nguengang, Thomas Bullot, Dominique Gaiti, Louis Hugues et Guy Pujolle. *Autonomic Resource Regulation in IP Military Networks : A Situatedness Based Knowledge Plane*. In Advanced Autonomic Networking and Communication, pages 81–100, 2008. 73

[Nichols *et al.* 1998] K. Nichols, S. Blake, F. Baker et D. Black. *RFC 2474 : Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, December 1998. http ://www.ietf.org/rfc/rfc2474.txt. 157, 159, 160

[Nilsson 1996] Nils J. Nilsson. Introduction to machine learning. Stanford University Stanford, 1996. 92

[Nolan & Doyle 2007] K.E. Nolan et L.E. Doyle. *Teamwork and Collaboration in Cognitive Wireless Networks.* IEEE WIRELESS COMMUNICATIONS, vol. 14 N° 4, pages 22–27, 2007. 69

[NS2 2009] NS2. *The Network Simulator website*, 2009. http ://ns-nam.isi.edu/nsnam/index.php/Main_Page. 151

[NS3 2009] NS3. *The ns-3 network simulator*, 2009. http ://www.nsnam.org/. 152

[of Philosophy SEP 2005] Stanford Encyclopedia of Philosophy SEP. *Automated Reasoning*, 2005. disponible à l'adresse http ://plato.stanford.edu/entries/reasoning-automated/. 205

[Omar & Taleb-Bendiab 2006] W. M. Omar et A. Taleb-Bendiab. *Service Oriented Architecture for E-health Support Services Based on Grid Computing Over.* In Proc. IEEE International Conference on Services Computing SCC '06, pages 135–142, 18–22 Sept. 2006. 57

[OMNETPP 2009] OMNETPP. *OMNeT++ Community Site*, 2009. http ://www.omnetpp.org/. 151

[OPNET 2009] OPNET. *OPNET official Website*, 2009. http ://www.opnet.com/. 151

[Oreizy *et al.* 1999] Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S. Rosenblum et Alexander L. Wolf. *An Architecture-Based Approach to Self-Adaptive Software.* IEEE Intelligent Systems, vol. 14, no. 3, pages 54–62, 1999. 91

[Patterson 2002] D. Patterson. *Availability and Maintainability Performance : New focus for a New Century.* In USENIX conference on File and Storage Technologies FAST 2003, Monterey, 2002. 2, 13

[Pellegrini *et al.* 2006] Francesco De Pellegrini, Iacopo Carreras, Giuseppa Alfano et Al. *Application Scenario Analysis, Network Architecture Requirements and High-Level Specifications*, 2006. Bionets deliverable Deliverable accessible à l'adresse http ://www.bionets.eu/docs/BIONETS_D1_1_1.pdf dernier accès 23 juin 2009. 32

[Prehofer & Bettstetter 2005] C. Prehofer et C. Bettstetter. *Self-organization in communication networks : principles and design paradigms*. volume 43, pages 78–85, 2005. 22

[project SELF-STAR 2007] project SELF-STAR. *Self-Star : Agents, Composants et Services Autonomiques*, 2007. accessible à l'adresse http ://liuppa.univ-pau.fr/spip/article.php3 ?id_article=4 dernier accès 23 juin 2009. 34

[Quinlan & Cameron-jones 1993] J. R. Quinlan et R. M. Cameron-jones. *FOIL : A Midterm Report*. In In Proceedings of the European Conference on Machine Learning, pages 3–20. Springer-Verlag, 1993. 120

[Quirolgico *et al.* 2003] S. Quirolgico, K. Mills et D. Montgomery. *Deriving Knowledge for the Knowledge Plane*. June 2003. 58, 59

[Qusay 2007] H. Mahmoud Qusay. Cognitive networks - towards self-aware networks : Introduction, volume 1. Editions John Wiley & Sons, 2007. 70

[Ramming 2004] Christopher Ramming. *Cognitive Networks*. In DARPATech 2004 Symposium March 9-11, in Anaheim, California, 2004. Available at http ://www.darpa.mil/DARPATech2004/pdf/scripts/RammingScript.pdf. 68, 70

[Sawma *et al.* 2008a] G. Sawma, I. Aib, K. Barbar et G. Pujolle. *A Recursive Load Balancing technique for VoIP-dedicated WLANs*. In Proc. IEEE/ACS International Conference on Computer Systems and Applications AICCSA 2008, pages 830–833, March 31 2008–April 4 2008. 57

[Sawma *et al.* 2008b] G. Sawma, I. Aib, R. Ben-El-Kezadri et G. Pujolle. *ALBA : An autonomic load balancing algorithm for IEEE 802.11 wireless networks*. In Proc. IEEE Network Operations and Management Symposium NOMS 2008, pages 891–894, 7–11 April 2008. 57

[Serban 2003] Rares Serban. *La gestion dynamique de la Qualité de Service dans l'Internet*. PhD thesis, Université de Nice Sophia Antipolis, 09 2003. 164

[Serrano *et al.* 2007] J. M. Serrano, J. Serrat et J. Strassner. *Ontology-Based Reasoning for Supporting Context-Aware Services on Autonomic Networks*. In Proc. IEEE International Conference on Communications ICC '07, pages 2097–2102, 24–28 June 2007. 56

[Shen *et al.* 2004] B. Shen, Z. Xu, S. Wee et J. Apostolopoulos. *Semantic-enhanced distribution &adaptation networks.* In Proc. IEEE International Conference on Multimedia and Expo ICME '04, volume 2, pages 1383–1386, 30–30 June 2004. 57

[Sifalakis *et al.* 2004] Manolis Sifalakis, Manolis Mavrikis et Georgios Maistros. *Adding Reasoning and Cognition to the Internet.* In 3rd Hellenic Conference on Artificial Intelligence (EETN), pages 365–375, 2004. 70

[Srinivasan 2002] Ashwin Srinivasan. *The Aleph Manual. http ://www.comlab.ox.ac.uk/ oucl/research/areas/machlearn/Aleph/, Last update : Nov 21 (2002)*, 2002. 126, 155, 168

[Sterritt & Bustard 2003] R. Sterritt et DW. Bustard. *Autonomic Computing : a mean of achieving dependanbility.* In Proceedings of IEEE International Conference on the engeneering of computer based systems ECBS'03, pages 247–251, 2003. 17, 91

[Stillings 1987] N. A. Stillings. *What is Cognitive Science ?* In N. A. Stillings, M. H. Feinstein, J. L. Garfield, E. L. Rissland, D. A. Rosenbaum, S. E. Weisler et L. Baker-Ward, editeurs, Cognitive Science : An Introduction, pages 1–16. MIT Press, Cambridge, MA, 1987. 47

[Strassner *et al.* 2006] John Strassner, N. Agoulmine et E. Lehtihet. *FOCALE : A Novel Autonomic Networking Architecture.* In Latin American Autonomic Computing Symposium (LAACS), 2006. 72

[Strassner *et al.* 2007] J. Strassner, N. Agoulmine et E. Lehtihet. *FOCALE A Novel Autonomic Networking Architecture.* In International Transactions on Systems, Science, and Applications (ITSSA) Journal, volume Vol. 3, No 1 of *ISSN 1751-1461*, pages 64–79, 2007. 71

[Strassner 2004] John S. Strassner. Policy-based network management : solutions for the next generation. The Morgan Kaufmann series in interactive technologies, 2004. 71

[Strassner 2007] John Strassner. *The role of Autonomic Networking in Cognitive Networks(Book Chapter).* Cognitive Networks - Towards Self-Aware Networks by Qusay H. Mahmoud, ed. , John Wiley & Sons, vol. 1, pages 23–52, 2007. 70, 75

[SUNMicrosystems 2007] SUNMicrosystems. *Sun N1 Service Provisioning System*, 2007. available at http ://www.sun.com/software/products/service_provisioning/index.xml. 33

[Sutton & Barto 1998] R. S. Sutton et A. G. Barto. *Reinforcement Learning : An Introduction*. Neural Networks, IEEE Transactions on, vol. 9, no. 5, page 1054, 1998. 97

[Thomas *et al.* 2005] R. W. Thomas, L. A. Dasilva et A. B. Mackenzie. *Cognitive networks*. In New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on, pages 352–360, 2005. 68, 71

[Tschudin *et al.* 2008] Christian Tschudin, Christophe Jelger et al. *of the ANA Blueprint - final version*, 2008. deliverable of ANA project accessible at http ://www.ana-project.org/web/publications/deliverables2008/start last acces 23th june 2009. 32

[Urra *et al.* 2004] A. Urra, E. Calle et J. L. Marzo. *Adding new components to the knowledge plane GMPLS over WDM networks*. In Proc. IEEE Workshop on IP Operations and Management, pages 82–86, 11–13 Oct. 2004. 57

[Vernon *et al.* 2007] David Vernon, Giorgio Metta et Giulio Sandini. *A Survey of Artificial Cognitive Systems : Implications for the Autonomous Development of Mental Capabilities in Computational Agents*. IEEE Trans. Evolutionary Computation, vol. 11, no. 2, pages 151–180, 2007. 48, 49

[VISIDIA 2008] VISIDIA. *The ViSiDiA Project : Visualization and Simulation of Distributed Algorithms*. In Please see http ://www.labri.fr/projet/visidia/, 2008. 142

[WATKINS & DAYAN 1992] CHRISTOPHER J.C.H. WATKINS et PETER DAYAN. *Technical Note Q,-Learning*. Machine Learning, Kluwer Academic Publishers, vol. 8, pages 279–292, 1992. 103, 104

[WATKINS 1989] CHRISTOPHER J.C.H. WATKINS. *Learning from delayed rewards*. PhD thesis, University of Cambridge,, 1989. 103

[Watts *et al.* 2007] E. C. L. Watts, M. Merabti et A. Taleb-Bendiab. *A Control Plane Architecture to Enhance Network Appliance Agility through Autono-*

mic Functionality. In Proc. 21st International Conference on Advanced Information Networking and Applications Workshops AINAW '07, volume 1, pages 923–928, 21–23 May 2007. 57

[Wawrzoniak *et al.* 2003] M. Wawrzoniak, L. Peterson et T. Roscoe. *Sophia : An information plane for networked systems.* In In HotNets-II, 2003. 63

[Wawrzoniak *et al.* 2004] Mike Wawrzoniak, Larry Peterson et Timothy Roscoe. *Sophia : an Information Plane for networked systems.* SIGCOMM Comput. Commun. Rev., vol. 34, no. 1, pages 15–20, 2004. 63

[Wei 1998] G. Wei. *A Multiagent Perspective of Parallel and Distributed.* In Machine Learning&quot ;, Proceedings of the 2nd International Conference on Autonomous Agents, pages 226–230, 1998. 73