HAL
open science

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

▶ **To cite this version:**

Maïssa Mbaye, Francine Krief. Un module d'apprentissage pour l'auto-configuration et l'auto-optimisation des réseaux IP offrant des garanties de qualité de service. Huitièmes Journées Doctorales en Informatique et Réseaux (JDIR'07), Jan 2007, Marne-la-Vallée, France. pp.121-130. hal-01116626

HAL Id: hal-01116626

https://hal.science/hal-01116626v1

Submitted on 16 Feb 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Un module d'apprentissage pour l'auto-configuration et l'auto-optimisation des réseaux IP offrant des garanties de qualité de service

Maïssa Mbaye
LaBRI
Bordeaux, France
Email : maissa.mbaye@labri.fr

Francine Krief
LaBRI, ENSEIRB
Bordeaux, France
Email : francine.krief@labri.fr

***Résumé*— L'automatisation des fonctions de gestion des réseaux IP offrant des garanties de qualité de service est de plus en plus une nécessité vu la taille et la complexité des réseaux actuels. La gestion "autonomique" veut apporter une réponse à ce problème en donnant aux équipements la capacité de s'autogérer en respectant des politiques de haut niveau fournies par l'administrateur. Cet article propose une méthode basée sur les outils de l'intelligence artificielle pour réaliser les fonctions d'auto-configuration et d'auto-optimisation d'un réseau DiffServ. Cette approche s'appuie sur un module d'apprentissage qui alimente une base de connaissances afin d'apprendre à adapter la configuration de l'équipement réseau aux changements de son environnement.**

I. Introduction

De nos jours, nous assistons à une explosion des équipements qui utilisent le réseau Internet (Téléphones portables, PDA, ordinateurs ...) et des services disponibles auprès des ISP (Internet Service Providers). Les nouvelles applications sur Internet sont demandeuses de garanties en termes de bande passante, de délai de transmission...

Loin de suivre un effet de mode, les opérateurs sont obligés de fournir de la qualité de service sur IP pour suivre la mutation des équipements terminaux et l'évolution des besoins utilisateurs. Cependant cette tâche n'est pas triviale car la pile TCP/IP n'était pas prévue pour faire de la qualité de service.

L'IETF (Internet Engineering Task Force) à proposé deux solutions pour pallier à ce manquement : IntServ (Integrated Services) et DiffServ (Differentiated Services). Cette avancée qui consiste à enrichir IP d'un mécanisme de gestion de Qualité de Service a eu comme effet de complexifier la gestion de ce type de réseau. En effet, la taille des réseaux et l'hétérogénéité des équipements qui les composent compliquent grandement la configuration et l'optimisation d'un réseau IP avec garantie de qualité de service. En particulier, peu de personnes sont capables de configurer de manière optimale des routeurs IP DiffServ. Et pour de grands réseaux, l'opération devient quasi impossible à faire de manière rigoureuse à l'échelle humaine.

Un nouveau concept est né pour aider les administrateurs, celui de "gestion autonomique". Il a pour objectif de rendre les équipements autonomes en réduisant leur dépendance par rapport à l'être humain. Ainsi ces équipements sont capables de s'autogérer en fonction de leur état et de celui de leur environnement tout en respectant des directives de haut niveau fournies par l'administrateur.

Le présent document expose une méthode d'application de l'apprentissage artificiel aux réseaux IP DiffServ afin d'auto-optimiser la configuration des équipements, en accord avec la vision de la gestion autonomique.

La première partie de ce document présente DiffServ et la gestion par politique qui permettent d'appréhender la notion de qualité de service sur Internet. La seconde partie présente la gestion autonomique, ceci permettra de bien situer le problème de l'auto-optimisation. La troisième partie introduit le paradigme de l'apprentissage artificiel en justifiant au passage les choix qui sont faits au niveau du module d'apprentissage. Enfin la dernière partie présente le module d'apprentissage que nous proposons.

II. Infrastructure DiffServ

Le fonctionnement basique du protocole IP est le Best-effort, c'est-à-dire la transmission au mieux des paquets. Il n'y a aucun favoritisme entre les paquets en cas de compétition pour l'accès aux ressources du réseau. Avec l'apparition de nouvelles applications TCP/IP (vidéo haute qualité, téléphonie sur IP, ...), une nette différenciation entre les applications réseaux en terme de besoins critiques de bande passante, de latence, de délai, c'est à dire de qualité de service est très vite apparue.

Ce besoin a motivé l'enrichissement de l'infrastructure TCP/IP de mécanismes permettant de fournir la qualité de service (QoS) tel que DiffServ. 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 (Internet Engineering Task Force) et la gestion par politique qui est la méthode de configuration privilégiée dans ce contexte.

A. *Le modèle DiffServ*

Dans le modèle DiffServ, l'information de qualité de service est codée sur l'octet du champ ToS (Type of Service

[ISI81]) de l'entête IP. Elle est appelée DiffServ Codepoint ou DSCP [NBB98] et permet d'identifier le type de service.

Le trafic entrant dans le réseau est classé et (peut être) conditionné par les noeuds de bordure du réseau pour être, ensuite assigné à différents BA (Behaviour Aggregates). Un BA (ou DS Behaviour Aggregate) est une collection de paquets qui ont le même DSCP et qui traverse un lien dans une direction particulière. Le DSCP représente ainsi l'identifiant du BA.

Le travail des noeuds dans le coeur du réseau se résume essentiellement à la retransmission des paquets suivant le Per-Hop Behaviour (PHB) qui est associé au DSCP du paquet [BCD98]. Le PHB définit le traitement à effectuer sur un paquet avant de le retransmettre. L'IETF a défini un certain nombre de PHB standards tels que Assured Forwarding (AF-PHB) [HBW99], Expected Forwarding (EF-DS) [JNP99], et Default ou Best-effort (DS-FIELD) [NBB98].

Une implémentation d'un PHB sur un noeud supportant DiffServ peut comporter un ensemble d'éléments. Ces éléments sont l'objet d'un modèle proposé par l'IETF : DS-MODEL [BCD98]. Le DS-MODEL présente ces éléments comme des blocs fonctionnels. Les principaux blocs sont entre autres[BCD98] : Traffic Classification Elements (TCE) , Metering Functions, Actions of Marking, Absolute Dropping, Counting et Multiplexing et Queing des éléments.

Des combinaisons de ces éléments 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 Fig. 1 donne un exemple de TCB.

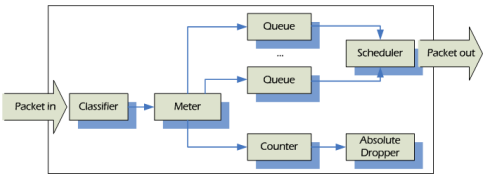


Fig. 1. Exemple de Traffic Conditioning Block

La gestion et la configuration de ces éléments et des TCB d'un équipement supportant DiffServ se fait à l'aide d'une interface de configuration.

La gestion des réseaux est présentée dans la section suivante.

B. La gestion des réseaux DiffServ

La gestion des équipements est un problème crucial dans le fonctionnement d'un système. En effet, le comportement d'un système DiffServ est étroitement lié à la manière avec laquelle les routeurs de coeur 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 règle 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 dans le sens de simplifier et d'automatiser (exemple [LYM02] et [SAK05]) le processus de gestion des réseaux DiffServ par des politiques.

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 aux événements qui surviennent sur le réseau sous-jacent.

Trois méthodes sont utilisées pour atteindre cet objectif : 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 changements dynamiques 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.

La méthode qui semble être la plus intéressante pour un environnement IP est évidement la troisième Il serait souhaitable d'avoir des équipements qui sachent s'auto-optimiser et interagir avec leur environnement pour adopter la meilleure stratégie globale. Des recherches sont réalisées dans ce sens et ont abouti au concept de "gestion autonomique".

III. Gestion autonomique

L'idée de la gestion autonomique est de développer des systèmes qui soient autonomes, capables de s'auto-organiser, s'auto-gouverner sans intervention humaine sauf pour la spécification des directives et objectifs de haut niveau. Ceci permet de cacher les détails de gestion et de contrôle des équipements logiciels et matériels aux administrateurs.

Le réseau devient alors autonome ou "autonomique", terme plus proche de la traduction anglaise "autonomic networking" et qui permet de souligner le fait que le comportement du réseau est guidé par des directives (ou politiques) de haut niveau émanant de l'administrateur. En ce sens, il n'est pas seul. Par la suite nous utiliserons indifféremment les termes autonome et autonomique.

Les laboratoires d'IBM [HOR01] ont lancé en 2001 l'ACI : Autonomic Computing Initiative.

Cette initiative a un ultime objectif, celui de rendre les systèmes des technologies de l'information complètement autonomes grâce une gestion autonome en accord avec des politiques de haut niveau. L'AC est considéré aujourd'hui comme un modèle de référence pour évoluer vers la gestion autonome des réseaux

les objectifs de la gestion autonome au nombre de quatre résument les principales fonctions d'un système autonome : l'auto-configuration (self configuration), l'auto-optimisation

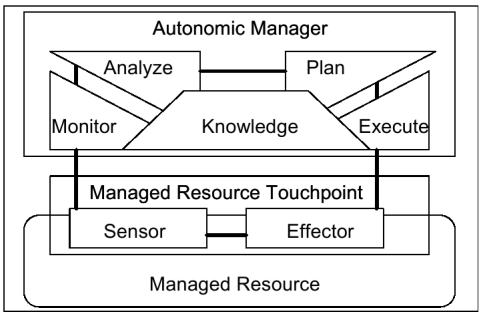


Fig. 2. Elément autonome

(self optimisation), l'auto-restauration (self healing) et enfin l'auto-protection (self protecting).

Nous nous intérressons dans l'article à l'aut-optimisation.Les systèmes autonomes vont continuellement chercher à améliorer leurs performances. Ils surveillent leur environnement afin d'adapter leur fonctionnement pour assurer une utilisation optimale des ressources par rapport aux besoins. Aussi ils visent à acquérir une certaine expérience afin faire les meilleurs choix. La réalisation de cet objectif vient en aide aux administrateurs qui sont souvent confrontés à des disfonctionnements, suite à des essais d'optimisation.

IV. L'ARCHITECTURE DE L'AUTONOMIC COMPUTING

L'architecture de l'Autonomic Computing est basée sur une architecture en couches. Chaque couche (composant) offre des services et est demandeuse de services proposés par les autres couches. Les systèmes autonomes ainsi constitués disposent d'éléments autogéres qui interagissent tout en respectant les politiques de haut niveau.

L'élément central dans cette architecture est sans nul doute l'entité autonome qui est une combinaison de gestionnaire autonome, de ressources gérées et des points de contact qui les relient.

Comme le montre la figure Fig. 2, un élément autonome est essentiellement composé d'une ou de plusieurs ressources gérées, de capteurs, d'actionneurs, d'un gestionnaire autonome mais aussi d'une boucle de contrôle qui lui permettra d'atteindre les objectifs de la gestion autonome.

Les ressources gérées sont des composantes logicielles ou matérielles qui peuvent être contrôlées. Contrairement aux ressources traditionnelles, les ressources gérées de l'AC sont supervisées au moyen d'interfaces qui sont implémentées par les capteurs et les actionneurs. Une ressource gérée peut être une composante unique telle qu'un serveur, une base de données ...ou un ensemble de ressources. Une entité autonome peut être lui même un élément géré.

L'interface de gestion pour le contrôle des éléments gérés est implémentée par les points de contact que sont les capteurs et les actionneurs. Ces points de contact permettent de réduire la complexité des technologies utilisées par les éléments gérés.

Les capteurs représentent une agrégation de tous les mécanismes capables de collecter des informations sur l'élément géré.

Les actionneurs sont l'agrégation de tous les mécanismes qui permettent de changer le comportement de l'élément gérée. Les actionneurs et les capteurs sont liés entre eux. En effet, un changement de configuration effectuée par un actionneur doit être accompagnée par une notification de ce changement grâce aux capteurs. La supervision et l'adaptation passent par ces interfaces de gestion : les capteurs permettent de collecter les informations sur l'état des éléments gérés et les actionneurs permettent d'entreprendre les actions adéquates.

Le gestionnaire autonome est l'élément central car il implémente la boucle de contrôle qui ne fait plus appel à une intervention humaine. En effet, un système capable de s'auto-gérer doit disposer d'une méthode automatique lui permettant de collecter les informations dont il a besoin et de les analyser pour savoir s'il doit ou non changer un des attributs du système qu'il gère. Le cas échéant, il doit mettre en place un plan des actions à effectuer pour atteindre le nouvel état souhaité. La boucle de contrôle fait référence à la réalisation de cette tâche de façon autonome par le gestionnaire autonome. L'exécution de la boucle de contrôle passe par les quatre fonctions qui sont représentées sur la Fig. 2.

Le monitoring : cette fonction offre des mécanismes qui réalisent la collecte, l'agrégation, la corrélation et le filtrage des données portant sur la topologie, les paramètres de configuration, l'état fonctionnel, la capacité offerte ou encore des mesures qui proviennent des ressources gérées.

L'analyse : Cette fonctionnalité offre les mécanismes pour observer et analyser des situations afin déterminer si des changements doivent être effectués ou non. Cette fonction d'analyse permettra la détection des symptômes et l'enclenchement des traitements appropriés. Cette fonctionnalité est influencée par le plan de connaissance (voir plus bas).

La planification offre les mécanismes pour planifier les actions nécessaires afin d'atteindre les objectifs et buts fixés. Elle consiste à sélectionner des procédures capables de mettre en œuvre les changements souhaités.

L'exécution : c'est la fonctionnalité de contrôle de l'exécution du plan d'action. Elle est réalisée au moyen des actionneurs.

Plan de connaissance

Dans la gestion autonome la connaissance que l'élément autonome a à sa disposition est un élément très important pour son fonctionnement. La manière d'acquérir cette connaissance et l'usage qu'il en fait n'en sont pas moins importants. Une méthode d'acquisition de la connaissance et de raisonnement sur ces connaissances qui nous vient du monde de l'intelligence artificielle est nommée l'apprentissage artificiel.

Dans le cadre de notre travail, nous ferons appel à l'apprentissage artificiel afin de réaliser la fonction d'auto-optimisation. En effet cette approche permettra de réaliser une auto-optimisation par des reconfigurations en fonction de la connaissance acquise par l'équipement et de l'état du réseau.

Dans la partie suivante nous présentons l'apprentissage artificiel sans toutefois rentrer dans ses formalismes.

V. L'APPRENTISSAGE ARTIFICIEL

L'apprentissage est une discipline de l'intelligence artificielle qui occupe une place prépondérante dans ce domaine. Les méthodes d'apprentissage artificiel (Machine Learning en anglais) sont à la base de la reconnaissance de formes, de parole, de l'extraction de connaissances à partir de données, etc.

Une définition classique qui pourrait être donnée à l'apprentissage du point de vue des sciences cognitives est la suivante : "C'est la capacité à améliorer les performances au fur et à mesure de l'exercice d'une activité [COM02]".

Certaines facultés peuvent être liées à cette notion : l'entraînement, la reconnaissance, la généralisation, l'adaptation, l'amélioration et l'intelligibilité. Loin de faire une présentation exhaustive et formelle de l'apprentissage, nous allons dans cette partie simplement présenter les différents types d'approches puis l'environnement méthodologique général pour la mise en place d'une méthode d'apprentissage. Enfin, nous discuterons des approches qui nous paraissent correspondre le mieux à notre cas ce qui permettra de poser les bases de l'algorithme d'apprentissage présenté dans la partie VI. Nous insisterons en particulier sur les types d'apprentissage et les algorithmes répondant le mieux à la problématique de l'auto-optimisation des équipements réseaux dans une vision gestion autonome.

A. Les types d'apprentissage

Selon le type de connaissances découvertes, l'apprentissage artificiel peut se découper en deux grands domaines : l'apprentissage numérique et l'apprentissage symbolique.
Le premièr est issu du monde des mathématiques. Ce type d'apprentissage manipule en général des données numériques et tente de ramener tous les problèmes à des problèmes numériques. Comme exemple, nous pouvons citer l'apprentissage de réseaux connexionnistes. Son défaut principal par rapport à la seconde tendance est une perte d'explications sur la connaissance acquise.

L'apprentissage symbolique, quand à lui, laisse les exemples sous une forme plus naturelle (ne ramène pas les descriptions à des données purement numériques) et donne une meilleure explication. L"apprentissage peut également être ou non supervisé.

Un panorama sur les types d'apprentissage peut être trouvée dans [MAM03]. Dans les sections qui suivent nous traiterons uniquement de l'apprentissage supervisé, qui répond mieux au contexte de notre étude(cf partie VI).

B. Environnement méthodologique de l'apprentissage supervisé

Dans un scénario classique d'apprentissage, le système apprenant reçoit des données de l'univers dans lequel il est placé. En apprentissage supervisé chacune de ces données prend la forme d'un couple (u,v) dont les membres représentent successivement la description d'une situation (encore appelée observation) et une réponse (ou encore réponse désirée) qui est supposée fournie par un Oracle (Fig. 3).

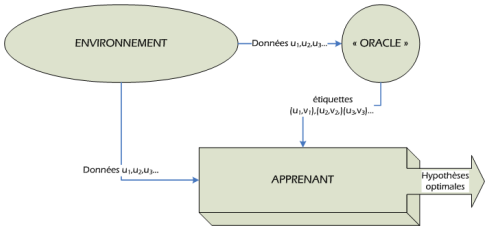


Fig. 3. Scénario d'induction supervisée

Cette situation présente plus précisément la notion d'induction supervisée. De manière plus formelle, l'induction est le processus par lequel on tire des lois de portée générale en partant de l'observation de cas particuliers. D'un point de vue conceptuel, l'apprentissage se joue entre un espace de description des objets en entrée et un espace des hypothèses. Le choix d'un principe inductif permet d'évaluer à partir d'exemples la qualité des hypothèses. Pour qu'une méthode d'apprentissage soit effective, il faut spécifier un algorithme de recherche dans l'espace des hypothèses qui tentera d'identifier une hypothèse optimale ou plus ou moins de s'en approcher [COM02]. La mise en place d'une méthode d'apprentissage requiert donc un passage forcé par une définition d'un certain nombre de notions : l'espace des représentation des données, l'espace des hypothèses et un algorithme d'apprentissage qui cherchera la meilleure hypothèse dans l'ensemble des hypothèses.

Espace de description des objets

L'apprentissage travaille sur un ensemble de données (objets) qu'il faut représenter avant toute chose. La manière de représenter les données dépend principalement de la nature des objets. La description se fait à deux niveaux : la représentation des exemples et celle de la connaissance acquise. La description des objets est souvent faite au moyen d'un vecteur d'attributs qui regroupe les informations atomiques. La nature de ces attributs peut varier suivant le cas considéré.

De plus amples développements sur la nature des attributs sont disponibles dans [COM02].

Un bon choix à ce niveau a une influence déterminante sur l'apprentissage et peut nécessiter ou non un prétraitement pour élaguer les données bruitées afin d'avoir de meilleurs résultats.

L'espace des hypothèses nous permet de ne pas avoir à définir les concepts décrivant les objets par des descriptions en extension (i.e par la liste des exemples). Une fois les objets représentés, il est nécessaire de représenter la connaissance au moyen de représentation appropriée au contexte et à la tâche. Il faut également définir une manière de faire la liaison entre les hypothèses et les données.

Algorithme d'apprentissage

Au niveau algorithmique, nous présentons deux méthodes qui semblent être particulièrement adaptée à notre cadre de recherche : l'apprentissage par renforcement et l'apprentissage incrémental.

L'apprentissage par renforcement peut être défini comme le problème d'apprendre, à partir d'expériences, l'action à effectuer en chaque situation d'une manière à maximiser

une récompense numérique au cours du temps [GAR04]. L'objectif étant alors de générer à partir d'expériences (état courant, action, état suivant, récompense) une stratégie

maximisant en moyenne la somme des récompenses au cours du temps. Beaucoup d'algorithmes de cette famille sont présentés dans [GAR04] et [COM02] comme le Q-learning, R-learning...

l'Apprentissage incrémental est réalisé au moyen d'un algorithme incrémental dédié à une tâche incrémentale. Une tâche d'apprentissage est incrémentale si les exemples d'apprentissage ne sont disponibles qu'au fil du temps et le plus souvent un par un. Une tâche incrémentale a deux caractéristiques :

- les exemples sont disponibles au fil du temps, généralement un à la fois, et non à priori.
- L'apprentissage peut durer "indéfiniment"

Formellement, un algorithme d'apprentissage est incrémental sans mémoire si pour tout ensemble d'exemples $e_1, \ldots, e_n$, il produit une séquence d'hypothèses $h_0, \ldots, h_n$ telle que h_{i+1} dépend seulement de h_i et de l'exemple courant [CAR00]. Ces algorithmes ont également deux caractéristiques principales :

- Un exemple n'est traité qu'une seule fois durant l'apprentissage.
- La qualité de la réponse fournie par l'apprentissage par rapport à l'objectif, est croissante au fil du temps. Plus l'apprentissage a d'exemples, plus sa connaissance sera fiable.

C. Discussion

Il apparaît assez clairement que l'apprentissage pourrait nous apporter au moins l'auto-optimisation définie dans l'architecture de l'Autonomic Computing et l'auto-adaptation.

Si nous nous replaçons dans le contexte de l'apprentissage afin d'optimiser le fonctionnement d'un équipement, l'intérêt de l'apprentissage symbolique supervisé paraît assez clair. En effet, les objets (voir chapitre IV) ici ne sont pas numériques et leur représentation nous est utile. L'ensemble des états du réseau est difficile à caractériser mais le critère qui revient toujours est le taux de pertes. L'ensemble des actions pourrait correspondre aux classes possibles et les états du réseau combinés à d'autres informations aux objets à classifier. De plus nous avons la possibilité de connaître le résultat (succès ou échec) d'une action entreprise presque automatiquement grâce aux capteurs et au monitoring. Cette réponse pourrait être vue comme celles fournies par l'"oracle" dans un mécanisme d'apprentissage supervisé.

L'apprentissage par renforcement permet de choisir des actions qui permettent de maximiser un gain. Cette définition ressemble au mode de fonctionnement du module d'apprentissage car sa fonction principale consiste en quelque sorte à apprendre à choisir une action (ou plusieurs) parmi un ensemble en fonction de l'état du réseau en utilisant sont expérience. Cependant deux raisons principales écartent l'apprentissage par renforcement comme mécanisme d'apprentissage dans notre cas : D'une part, l'emploi de simulation dynamique du processus à contrôler afin d'orienter l'exploration des fonctions de valeurs ou des politiques. De plus, nous ne disposons pas de loi probabiliste d'estimation du coût d'une action sur le réseau[1]. D'autre part dans le contexte réseau de routeurs, le facteur temps est une contrainte majeure. Donc nous n'avons pas le temps d'explorer[2] tout l'espace disponible à cause d'une demande de réactivité prioritaire sur le processus d'apprentissage lui même. En revanche cette méthode correspond à notre situation d'adaptation sur plusieurs points d'où l'intérêt de conserver certains avantages pour notre module final.

L'apprentissage incrémental quand a lui semble très adapté au problème du fait que l'état du réseau change au fil du temps, les données sont donc disponibles au fil du temps et la période d'apprentissage n'est pas limitée.

Apres avoir passé en revu ces deux types d'apprentissage il est naturel de se demander s'il n'existe pas quelque chose de déjà fait qui soit adaptée à notre problème. Malheureusement non, car notre problème est transversal aux deux méthodes décrites. La manière de procéder est proche de l'apprentissage par renforcement mais la validation de la connaissance est incrémentale.

Dans la partie suivante nous allons présenter l'approche que nous proposons qui est inspirée des algorithmes Q-learning [COM02] et ILA [CAM00] qui sont respectivement un algorithme générique d'apprentissage par renforcement et un système d'apprentissage supervisé incrémental.

VI. Le module d'apprentissage

Le fonctionnement du module d'apprentissage est représenté par la figure Fig. 4. Nous rappelons que le but du module d'apprentissage dans ce travail est d'arriver à l'auto-optimisation du réseau par configuration et reconfigurations successives. Nous avons l'élément réseau ("équipement" sur le schéma) qui a une certaine configuration de départ. Nous avons l'apprenti ("module d'apprentissage") et une base de connaissances qui représente les connaissance et l'expérience acquise au fur et à mesure. A chaque changement de l'état de l'équipement, une alarme est remontée vers le module d'apprentissage qui réagira (ou non) par une action de reconfiguration en utilisant le guide (la base de connaissance) qu'il a construit. Une fois l'action terminée, la base de connaissances (si c'est nécessaire) est mise à jour. Le module d'apprentissage essaie de trouver dans un cas problématique ou "optimisable" la meilleure action à effectuer et le meilleur paramètre à considérer. Par exemple si l'action est ChangerScheduler alors il va essayer de trouver le Scheduler le mieux adapté à l'état actuel du réseau.

Le module d'apprentissage peut être vu comme une boite noire qui prend en entrée des alarmes et une base de règles. En réponse aux alarmes, le module fournit l'action et le paramètre qui optimise la configuration en utilisant la connaissance qu'elle a accumulée au fil du temps en fonction de l'état actuel du réseau sous-jacent. Cette action s'applique sur la base des

[1] La logique floue serait une piste pour pallier à ce manquement

[2] Bien qu'il existe des méthodes de prédiction pour avoir une valeur dans un cours délai

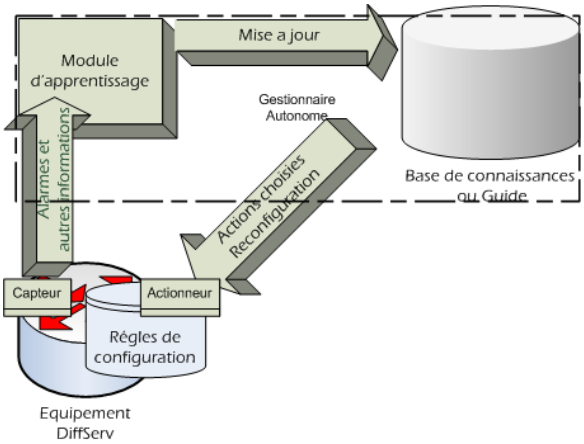


Fig. 4. Fonctionnement du Module d'apprentissage

règles et son résultat permettra de mettre à jour sa base de connaissances. En sortie nous aurons donc une nouvelle base de règles de configuration et une base de connaissances mise à jour.

A. Etats du réseau

Comme indiqué plus haut, le module fonctionne sur la base des alarmes qui lui sont remontées depuis l'équipement à chaque changement d'état. Il paraît donc indispensable de définir ce qui va représenter l'état du réseau. Il est assez difficile de définir l'état du réseau sachant que les caractéristiques candidats ne manquent pas. Mais dans le cas de ce travail nous allons le définir en fonction de ce qui est attendu de l'apprentissage. Les états du réseau peuvent être divisés en deux catégories : les états de crise et les états d'optimisation. Les états de crise ou problématiques sont liés à la congestion : légèrement congestionné, congestionné ou sévèrement congestionné. Nous définissons des seuils pour les différencier car ils n'ont pas le même niveau de gravité. L'équipement est légèrement congestionné si le taux de pertes de AF est compris entre 0% et 20%, il est congestionné si ce taux est compris entre 20% et 80% et l'équipement est très congestionné si les pertes dépassent strictement les 80% pour AF. La définition de l'état par rapport à AF est motivée par le fait que EF a des garanties strictes et que tout ce qui dépasse de son trafic est supprimé tout simplement et que BE n'a aucune garantie. D'un autre coté, nous avons les états d'optimisation qui sont ceux où le fonctionnement de l'équipement n'utilise pas les ressources de manière optimisée. Deux états sont considérés :

- une demande de bande passante de AF quand il utilise 80% la bande passante qui lui a été allouée.
- Une sous utilisation de la bande passante si AF utilise moins de 50% de la bande passante qui lui est allouée.

Dans ces deux derniers états si AF a besoin de bande passante alors l'équipement lui fournit une partie de celle de BE. Et dans le cas contraire, s'il n'utilise pas toute la bande passante qui lui a été allouée alors une partie est cédée à BE.

La figure Fig. 5 donne un résumé de ces états.

Numéro	Etat	Condition
1	SousUtilseBW	RBW[a] (AF) $\leq 20\%$ BW[b] (AF)
2	BesoinBW[c]	BW(AF)$\geq$RBW (AF)$\geq 80\%$ BW (AF)
3	LegerCongestion	Perte[d](AF)$\leq 20\%$
4	Congestion	20%<Perte(AF)$\leq 50\%$
5	SevereCongestion	50% < Perte(AF)$\leq 20\%$

[a]Bande passante réelle utilisée
[b]Bande passante reservée
[c]pas encore de pèrtes
[d]Pertes actuelles

Fig. 5. Etats du réseau.

Nous sommes assurés que ces états sont disjoints c'est-à-dire que le réseau ne peut être dans deux de ces états en même temps. Les deux premiers états surviennent avant que les pertes n'adviennent alors que les autres états surviennent lorsque le réseau commence à contracter des pertes.

Les choix de ces valeurs de seuil (20%, 80% ...) ne sont pas basés sur des tests mais son plutôt sur l'"intuition". Des travaux futurs pourraient permettre de calculer les valeurs optimales de manière dynamique en fonction de l'experience. L'ensemble des états du réseau représente l'espace des hypothèses de notre futur algorithme d'apprentissage.

B. Alarmes

Les alarmes sont des moyens de notifier au module ce qui se passe sur le réseau, les changements qui y surviennent et qui lui permettront de réagir au bon moment et de la meilleure manière. Deux aspects importants sont à définir : leur nature et leur fréquence.

Pour la nature des alarmes ce sont des tuples qui contiennent les informations dont le module d'apprentissage a besoin pour déclencher des actions. Ce sont des éléments de $CL \times EVT^2 \times R$ où

- CL est l'ensemble des classes AF1.1 ,AF1.2 et BE. Cette information dans l'alarme représente la classe de service qui est concernée par l'alarme.
- EVT est un événement qui correspond à un état du réseau parmi ceux définis précédemment. Un élément (e_1, e_2) de EVT^2 représente le passage de l'équipement de l'état e_1 à l'état e_2.
- R est l'ensemble des règles de configuration qui sont sur l'équipement. Cette composante de l'alarme nous renseigne sur la règle en question qui est concernée. Un numéro identifie chaque règle. (nous supposons qu'une alarme ne concerne qu'une règle).

Dans le cadre de l'apprentissage, l'ensemble des alarmes représente l'espace de représentation des données : l'espace sur lequel nous apprenons.

La fréquence de déclenchement des alarmes est aussi un facteur important à prendre en compte. Une alarme est envoyée au module d'apprentissage à chaque fois qu'il y a un changement d'état de l'équipement. Cela permet en même temps de réagir et de vérifier le résultat de l'action déclenchée. Par contre, toutes les réactions aux alarmes ne sont pas toujours des reconfigurations de l'équipement. En effet, une

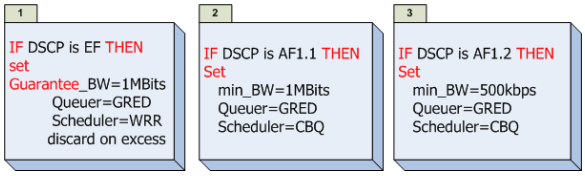


Fig. 6. Exemple de règles de configuration numérotées.

reconfiguration n'est déclenchée que si une alarme notifie un changement d'état du réseau dans le sens négatif. Cela suppose qu'il existe un moyen de comparer les états, nous en parlerons plus loin. Les alarmes ne sont pas modifiées par le module d'apprentissage il s'en sert seulement comme ensemble d'exemples au cours de l'entraînement.

C. Base de règles

Le module d'apprentissage agit aussi en prenant acte du contenu actuel de la base de règles de politique de configuration. Les règles de configuration concernent chacune une et une seule classe de service parmi AF, EF et BE comme nous l'avons précisé plutôt. Le but est de simplifier la modification d'une partie de ces règles (condition ou actions). La figure 6 donne un exemple de règles de configuration.

Optimiser un réseau est souvent assimilé au fait d'essayer d'utiliser les ressources (la bande passante en premier) au maximum, de prévenir la congestion et de bien la gérer lorsqu'elle devient inévitable. Dans le cadre d'un réseau avec qualité de service, nous pouvons y ajouter une amélioration du délai et de la gigue. Pour y parvenir le module dispose de deux bases sur lesquelles il peut agir : la base de règles de configuration et la base de connaissance.

Le module d'apprentissage agit sur les règles de configuration au moyen d'un certain nombre d'opérateurs qui sont applicables :

- Sur la base en tant qu'ensemble de règles
  - AjouterRegle (Condition, Actions)
  - InhiberRegle (NumeroRegle)
  - SupprimerRegle (NumeroRegle)
- Sur une règle particulière en la considérant comme un ensemble de conditions et d'actions.
  - ChangerScheduler (NouveauSched)
  - ChangerQueuer (NouveauQueuer)
  - MajShaper (NouvellesParams)
  - ChangerCondition (Anciennes, Nouvelles) : Changement de la partie condition d'une règle.
  - RedistributionBW () : déclanche une redistribution de la bande passante.

Le présent travail se limite à la prise en compte de la dernière catégorie, la première faisant appel à des méthodes de gestion de cohérence de base règles qui dépassent le cadre de ce travail[3]. Les opérateurs de la deuxième catégorie permettent de manipuler les éléments des Per Hop Behaviour qui sont les mécanismes de gestion des files d'attentes (Queuing) et

[3]Les evolutions futures prendront en compte cet aspect à l'aide des systèmes experts

Procédure RedistribuerBW(Etat,Cl)
[Calcul de la bande passante à redistribuer]
FW $\leftarrow$ 0 ; **Si** (Etat = BesoinBW) **Alors**
FW $\leftarrow$ Bandwidth (BE) ;
Sinon
Si (Etat = SousUtilseBW) **Alors**
FW $\leftarrow$ BW (BE)+25% BW (Cl) ;
Fin Si
Fin Si
[Restribution de la bande passante]
Pour $C_i \in$ **CL faire**
BW(C_i)$\leftarrow$BW(C_i)+FW/N ;
[N étant le nombre de classes]
Fin Pour

Fin
[FW Bande passante libre à redistribuer]
[Etat : état actuel du réseau]
[Cl : la classe de la règle qui a généré l'alarme]
[BW () : renvoie la bande passante réservée pour une classe donnée]

Fig. 7. Algorithme de redistribution de la bande passante

les mécanismes d'ordonnancement (Scheduling). En effet, les caractéristiques de qualité de service sont très sensibles à ces deux éléments.

Le module d'apprentissage va essayer d'adapter la configuration de l'équipement à l'état du réseau. Le choix d'une action (opérateur) plutôt que d'une autre, et du paramètre à passer à une action suivant l'état du réseau se fera à l'aide de poids et de priorité qui leur seront attribués. La pondération et la "priorisation" sont décrites plus loin. Le principe de fonctionnement de l'action $RedistributionBW()$ n'est pas trivial et dépend du type d'événement qui l'a déclenché et de la classe de la règle concernée. La figure 7 est une description en pseudo code de son fonctionnement

Cette proocedure prend en argument la classe de service qui a généré l'alarme(Cl) et l'Etat actuel de l'equipement(Etat). Dans un premier temps, l'algorithme calcule la bande passante redistribuable(FW). Si la classe est en cas de besoin de bande passante, la valeur de FW est toute la bande passante utilisable par la classe BE. Sinon, si nous sommes dans le cas d'une sous utilisation de la bande passante (Cl utilise moins de 20% de la bande passante qui lui a été allouée), alors l'algorithme ajoute à BW(BE), 25% de la bande passante allouée à Cl pour avoir la valeur de FW. Après ce calcul la redistribution (de FW) est faite de manière equitable (FW/N).

Ainsi en cas de manque de bande passante des classes protégées (AF, Etat=BesoinBW), la bande passante utilisée par BE est redistribuée aux autres. Si une classe protégée

n'utilise pas assez sa bande passante (Etat=SousUtliseBW), la redistribution se fait dans le sens inverse mais seule une partie de la bande passante de la classe est redistribuée.

D. Algorithme d'apprentissage

Maintenant que toutes les entrées du module sont décrites nous pouvons présenter l'algorithme d'apprentissage et la forme de représentation de la connaissance en sortie.

Le principe général de l'algorithme est de construire au fur et à mesure un ordre de préférence des actions à effectuer suivant l'état de l'équipement. Cet ordre permettra de mieux choisir les actions à planifier à chaque fois que le besoin s'en fait sentir et d'optimiser ainsi le fonctionnement de l'équipement.

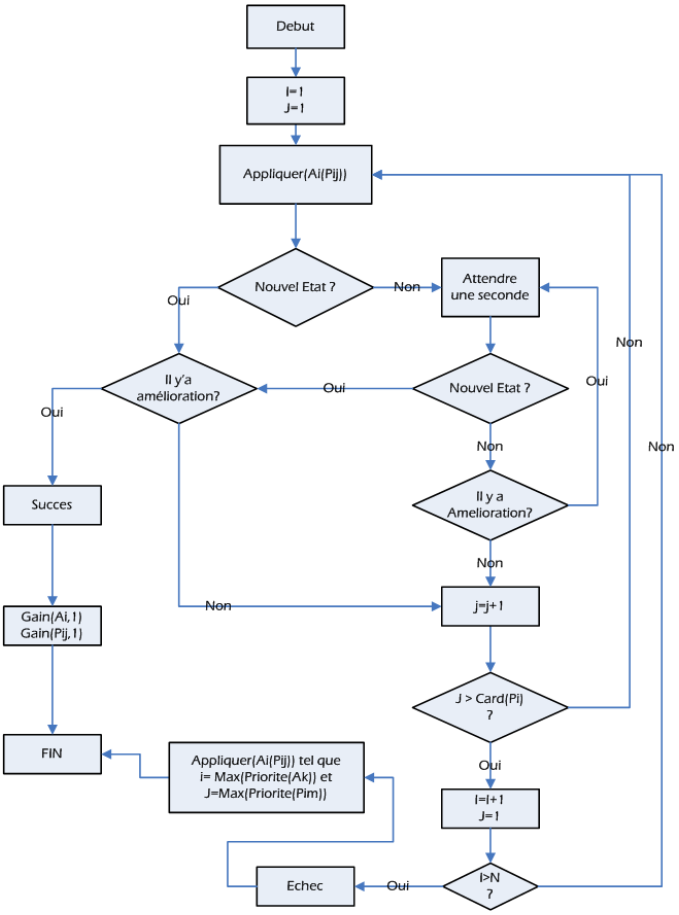
Toute règle est considérée comme étant composée d'un ensemble de conditions et d'actions. Pour chaque action un ensemble de paramètres est possible. Nous supposons que nous avons à notre portée toutes les actions possibles ainsi que tous les paramètres possibles des actions (Cette considération étant tout à fait conforme avec la vision gestion de la gestion par politique). Par la suite lorsque nous parlerons d'action nous ferons référence aux actions listées dans la section IV.2.3 (les opérateurs).

Les actions sont classées suivant les états et suivant un poids ou pondération qui dépendra de leur fréquence de réussite et d'une priorité. La priorité sera le moyen pour les experts du domaine d'influencer les choix du module et permettra au module de faire son choix parmi des actions qui ont la même pondération. La priorité 0 a une signification particulière, elle dénote une action qui n'est pas applicable lorsque le réseau est dans un état donné. C'est un moyen d'éviter que l'algorithme essaie toutes les actions.

Cependant, la pondération sera prise en compte en premier car elle est supposée être basée sur une expérience de l'apprenti concernant le fonctionnement du réseau et est ainsi plus adaptée à la réalité.

Une fois l'action désignée, il est nécessaire de lui donner le bon argument, le bon paramètre. Les paramètres d'une action sont aussi ordonnés suivant une priorité et une pondération de façon similaire à ce qui a été fait pour les actions. la figure 8 représente l'algorithme qui est exécuté par le module d'apprentissage.

L'algorithme se compose de deux phases : une phase d'essai et une phase de pondération, correspondant à l'apprentissage proprement dit. La phase d'essai travaille à partir d'un état. Il essaie les différentes actions disponibles en fonction de la pondération et de la priorité (priorité différente de 0). Et pour chaque action, il essaie les valeurs de paramètre qui peuvent être passées en argument en fonction du poids et de la priorité. Une action est considérée comme ayant réussie si elle change l'état du réseau dans le sens positif. Elle se verra attribuer un gain sur sa pondération ainsi que sur la valeur de paramètre qui lui a été appliquée. Sinon, si l'action ne change pas l'état du système mais l'améliore au bout d'une seconde l'algorithme attends à nouveau une seconde.



A=$A_1, A_2 \ldots A_n$: l'ensemble des actions disponibles de priorité $\neq 0$
P_i est l'ensemble des valeurs prises par le paramètre de l'action A_i.
P_{ij} est une valeur particulière du paramètre

Fig. 8. Algorithme d'apprentissage du module.

Deux types d'améliorations apparaissent dès lors : améliorer en changeant d'état et améliorer sans changer d'état. Le premier cas est caractérisé pas la réception d'une alarme alors que le second cas, le système récupère des informations sur l'état actuel du réseau telles que les pertes et la bande passante de la classe. Ces informations serviront ensuite de référence pour la prochaine mesure de l'amélioration de l'action choisie. Cette manière de faire évite qu'une action fasse osciller l'état du réseau dans un même état. Enfin si aucune des actions ne réussit, alors on revient à l'action la plus prioritaire avec son paramètre le plus prioritaire et on ne distribue pas de point. Cela signifie que l'on privilégie, dans ce cas, le choix de l'expert.

La définition de la réussite d'une action demande q'une relation d'ordre soit définie dans l'ensemble des états. Cette relation d'ordre est représentée sur la figure 9 (ordre décroissant) :

Cette la relation est définie de manière extensive car le nombre d'états est limité. L'attente d'une seconde dans l'algorithme est justifiée par le fait qu'il faut du temps pour que les effets d'une action soient perceptibles. Par contre, la valeur de durée est choisie de manière ampirique car une connaissance

SousUtilseBW
BesoinBW
LegerCongestion
Congestion
SevereCongestion

Fig. 9. Relation d'ordre dans l'ensemble de états

	$Etat_1$	...	$Etat_i$	...	$Etat_m$
A_1	$(0,0)$	...	$(pr^a\ (i,A_1), pd\ ^b\ (i,A_1))$	...	$(pr(m,A_1), pd(m,A_1))$
...	...	...	...	...	...
A_i	$(pr(1,A_i), pd(1,A_i))$	...	$(pr(i,A_i), pd(i,A_i))$	...	$(pr(m,A_i), pd(m,A_i))$
...	...	...	...	...	...
A_n	$(pr(1,A_n), pd(1,A_n))$	...	$(pr(i,A_n), pd(i,A_n))$	...	$(pr(m,A_n), pd(m,A_n))$

[a]Priorité de l'action A_1 pour l'etat i
[b]Poids de l'action A_1 pour l'etat i

Fig. 10. Matrice de description des actions

de la durée moyenne pour qu'une action prenne effet n'est pas connue et dépend de l'action, du paramètre et de l'état du réseau et est donc assez complexe pour être définie de manière rigoureuse.

La phase de pondération entre en jeu lorsque nous arrivons à une situation de réussite ou d'amélioration de l'état initial. Il y a deux niveaux de pondération suivant un état donné :

- Au niveau des actions à appliquer sur la règle
- Au niveau des valeurs à donner aux paramètres de l'action.

A partir d'un état donné, le module d'apprentissage choisira la meilleure action à entreprendre avec le bon paramètre et construira ensuite sa connaissance avec les résultats que donnent ces choix.

Pour représenter la connaissance qui est tirée de l'apprentissage dans un premier temps, est associé à chaque action, une ligne d'une matrice de taille (N, M) (N étant le nombre d'Actions, M étant le nombre d'états existants dans le système). Elle est mise à jour par le module d'apprentissage. Cette matrice donnera le poids et la priorité des actions pour chaque état considéré (voir Fig. 10).

Pour déterminer l'action à entreprendre lorsque le système est dans l'état $Etat_i$, les actions dont la priorité (prise à la colonne i) n'est pas nulle sont extraites et ordonnées suivant le poids puis la priorité de manière décroissante, de manière à ce que progressivement l'expérience l'emporte ou vienne conforter l'expertise. Ce prétraitement peu cependant être non adapté si nous définissons un nombre trop important d'états pour le réseau. Une fois une action choisie, l'ordre d'essai des valeurs de paramètres P_{ij} est calculé à partir d'un arbre construit au fur et à mesure (voir figure 11) de l'entraînement du module.

Ici nous ne considérons que les actions dont l'ensemble Pi des valeurs de paramètre est discret et fini. Par exemple ChagerScheduler aura comme ensemble de paramètres l'ensemble des mécanismes de Scheduling disponibles sur l'équipement.

E. Les priorités

La configuration des priorités n'est pas une chose triviale et nécessite généralement l'avis d'un expert du domaine des

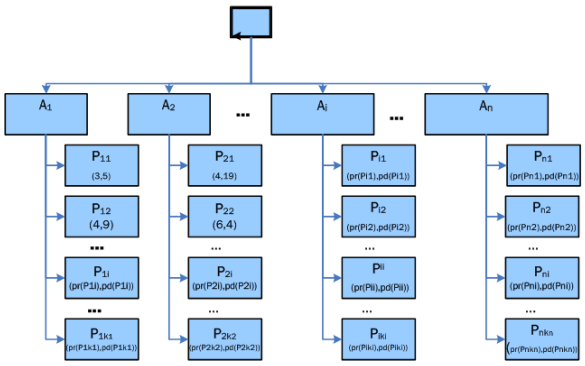


Fig. 11. Arbre de pondération des Paramètres

	ChangerScheduler	ChangerQueuer	RedistributionBW
SousUtilseBW	0	1	3
BesoinBW	1	2	3
LegerCongestionn	1	3	0
Congestion	1	3	0
SevereCongestion	2	3	0

Fig. 12. Tableau indicatif sur la configuration des priorités

réseaux. Cependant un certain nombre de résultats issus du monde de la recherche (voir plus bas) et de tests de performances sont disponibles. Le tableau Fig. 12 donne des indications sur la manière de donner les priorités en fonctions des états et actions définies.

Dans [FUI04] nous avons une indication sur les manipulations mécanismes de gestion de queues et leur utilisation en fonction de l'état du réseau (en fonction du niveau de congestion). [NIS03] donne un benchmark sur la manipulation des paramètres de QoS pour montrer leurs influences sur la gigue, la bande passante et les pertes de données. [ISA02] propose une manière de manipuler la bande passante pour une utilisation optimale des ressources.

VII. L'IMPLÉMENTATION

Une implémentation en JAVA du module d'apprentissage a permis de tester le fonctionnement de l'algorithme.

Le banc de test est composé de deux éléments principaux :

- Un générateur d'alarmes : Il simule un équipement qui envoie régulièrement des alarmes et reçoit si nécessaire des opérations (actions) à effectuer sur sa base de règles de politique de configuration.
- Un apprenti qui déroule l'algorithme proprement dit. Il reçoit les alarmes et envoie (si nécessaire) une action qu'il choisit parmi celles disponibles en fonction de leur poids et priorité.

Ces deux elements communiquent à travers des sockets JAVA. Pour le stockage des données (base de connaissances et base de règles) nous avons utilisé le langage XML [BIR01]. XML a l'avantage d'avoir des langages de définition de prototype (XML Schema [THO01], [BIR01] par exemple), de formatage (XSLT [JAM99]) et d'interrogation (XPATH [THB99], XQuery [SCF05]) de ses données, ce qui facilite considérablement la manipulation du contenu de la base.

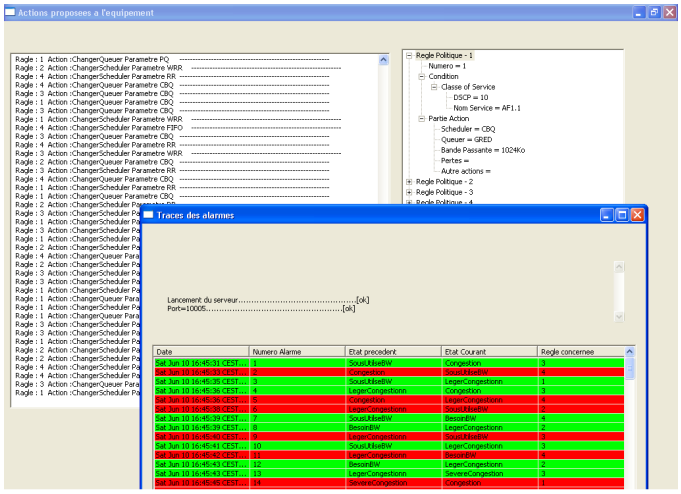


Fig. 13. Implémentation en fonctionnement

Pour traiter ces fichiers à partir de JAVA, nous avons utilisé l'API JDOM [JDOM], une API de manipulation de documents XML. La figure 13 est une capture du module en fonctionnement.

Des tests de fonctionnement qui sont basés sur des captures de la base de connaissances et en faisant varier les priorités ont été faits. Ces tests montrent que le comportement de l'algorithme dépend assez fortement de la configuration initiale des priorités. En effet, les premières actions qui sont proposées au début de l'exécution sont celles qui ont la plus grande priorité. Et si elles réussissent à faire changer d'état le réseau assez souvent les autres actions auront moins de chance d'être proposées durant l'exécution de l'algorithme.

VIII. Conclusion

Dans ce travail nous avons présenté les problèmes de gestion des réseaux IP avec qualité de service. En effet, de nombreux équipements utilisant TCP/IP ont maintenant des besoins en qualité de service qu'il faut garantir. Le modèle de QoS DiffServ nous a servi de base de test. Un champ déjà existant dans le protocole IP est utilisé pour offrir la qualité de service.

Nous avons proposé une solution au problème de l'auto-optimisation dans le cadre de l'Autonomic Computing. Cette solution implique une certaine "intelligibilité" de l'équipement. Ce travail s'est focalisé sur l'apprentissage symbolique. Nous avons proposé un algorithme d'apprentissage inspiré de Q-learning [COM02] et ILA [CAM00]. Le premier est issu de l'apprentissage par renforcement alors que le second provient de l'apprentissage incrémentale. L'algorithme permet en tant que tel d'adapter les paramètres de TCB en fonction de l'état du réseau. Il fonctionne sur la base d'alarmes qui lui sont remontées. Une implémentation a permis de valider son l'aspect comportemental(de la version relativement locale de l'algorithme).

Par la suite nous nous intérressons à l'apprentissage distribué. En effet, la portée et les paramètres des prises de décisions du module reste limitée au niveau local. Il est intéressant de l'étendre à un niveau plus global avec un dialogue avec les équipements voisins (gestion distribuée). Mais aussi, faire des tests de perfomance pour une validation définitive.

Remerciements

Nous tenons à remercier tout particulièrement M. Henry Soldano pour son aide sur la partie apprentissage artificiel.

References

[BCD98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss. *"RFC 2475 : An Architecture for Differentiated Services."* IETF December 1998. http ://www.ietf.org/rfc/rfc2475.txt

[BIR01] BIRON P.V., ET AL., ... *"XML Schema Part 2 : Datatypes."*, W3C Recommendation, mai 2001.

[CAM00] Christophe Giraud-Carrier and Tony Martinez. *"ILA : Combining Inductive Learning with Prior Knowledge and Reasoning."* 2000

[CAR00] Christophe Giraud-Carrier. *"A Note on the Utility of Incremental Learning."* AI Communications,volume 13 (4) : 215–223, December 2000.

[COM02] A. Cornuéjols-L. Miclet. *"Apprentissage artificiel, Concepts et algorithmes."* Eyrolles 2002.

[FUI04] David Fuin, thèse de doctorat. *"Qualité de service : Des réseaux IP à l'intégration dans les réseaux actifs."* Université de Franche Comité chapitre 2 sur la qualité de service. Décembre 2004. http ://lifc.univ-fcomte.fr/ fuin/Files/TheseDavidFUIN.pdf

[GAR04] Frederic Garcia. *"Apprentissage et contrôle : Apprentissage par renforcement."* INRA-URBIA, 2004.

[HBW99] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski. *"RFC 2597 : Assured Forwarding PHB Group."* IETF June 1999. http ://www.ietf.org/rfc/rfc2597.txt

[HOR01] HORN P., *"Autonomic computing : IBM perspective on the state of information technology."* IBM T.J.Watson Labs, NY, Presented at AGENDA 2001, Scottsdale, octobre 2001.

[ISA02] INRIA-Sophia Antipolis. *"ARchitecture de Contrôle Adaptative des Envoronnement IP(ARCADE) : Algorithmes et politiques de contrôle de la QoS."* Année 2002

[ISI81] Information Sciences Institute University of Southern California. *"RFC 791 INTERNET DARPA INTERNET PROTOCOL SPECIFICATION."* September 1981 http ://www.ietf.org/rfc/rfc791.txt

[JAM99] James Clark. *"XSL Transformations (XSLT) Version 1.0."* W3C .Recommendation 16 November 1999.

[JDOM] API JDOM. http ://www.jdom.org/

[JNP99] V. Jacobson, K. Nichols, K. Poduri. *"RFC 2598 : An Expedited Forwarding PHB."* June 1999. http ://www.ietf.org/rfc/rfc2598.txt

[LYM02] Leonidas A. Lymberopoulos. *"An Adaptive Policy Based Management Framework for Network Services Management."* PhD Transfer Report, Imperial College London, 29 May 2002. http ://www.doc.ic.ac.uk/ llymber/downloads/PhD-Tranfer-llymber.pdf

[MAM03] M.A. Maloof, R. S. Michalski *"Incremental learning with partial instance memory."* ElSEVIER 2003

[NIS03] Ibrahima Niang - Dominique Seret. *"Dimensionnement de DiffServ basé sur des métriques de performance ."* DNAC 2003 Congres Paris.

[SAK05] N. Samaan, A. Karmouch. *"An automated Policy-Based Management Framework for Differentiated Comminication Systems."* IEEE journal on selected areas in Communications, vol. 23 No 12, December 2005.

[SCF05] Scott Boag, Don Chamberlin, Mary F. Fernández... *"XQuery 1.0 : An XML Query Language."* W3C Candidate Recommendation, November 2005 http ://www.w3.org/TR/xquery/

[THB99] Jean-Jacques Thomasson ,Yves Bazin. *"XML Path Language (XPath) 1.0."* W3C recommendation, Novembre 1999. http ://xmlfr.org/w3c/TR/xpath/

[THO01] THOMPSON H.S., ... *"XML Schema Part 1 : Structures."* W3C Recommendation, Mai 2001. http ://www.w3.org/TR/xmlschema-1/