Une approche adaptative pour la surveillance d'informations agrégées dans les réseaux complexes

Rafik Makhloufi, Guillaume Doyen, Grégory Bonnet, Dominique Gaïti

Une approche adaptative pour la surveillance d'informations agrégées dans les réseaux complexes

Rafik Makhloufi*, Guillaume Doyen*, Grégory Bonnet† and Dominique Gaïti*
*ICD/ERA, Université de Technologie de Troyes ; †GREYC/MAD, Université de Caen
Email : {rafik.makhloufi, guillaume.doyen, dominique.gaiti}@utt.fr ; gregory.bonnet@unicaen.fr

***Résumé*—Dans le domaine de la gestion autonome, être capable de concevoir des infrastructures efficaces qui opèrent à un coût moindre est une problématique importante. Dans ce contexte, nous proposons une architecture auto-adaptative pour la surveillance de données agrégées. Cette architecture surveille l'état opérationnel du réseau sur lequel elle est déployée et met en œuvre dynamiquement le protocole d'agrégation de données le plus utile en fonction du contexte courant. Pour ce faire, elle repose sur la logique floue et une prise de décision multi-critère. Nous validons cette proposition par simulation, en montrant la précision de nos composants ainsi que la limite du coût induit.**

***Mots-Clés*—Gestion autonome, agrégation décentralisée, surveillance, vue située, prise de décision, logique floue.**

I. Introduction

Depuis quelques années, les infrastructures de gestion autonome se présentent comme une alternative forte aux architectures standards pour faire face à la complexité, la taille et la dynamique et des réseaux et services d'aujourd'hui. Si les solutions existantes offrent un premier degré d'autonomie et de décentralisation des fonctions de gestion et contrôle, elles restent non-adaptatives. En effet, elles n'utilisent qu'une seule approche de gestion quel que soit le contexte opérationnel du réseau géré. À l'inverse, les réseaux sur lesquels elles sont déployées sont dynamiques en termes, par exemple, de taille, de mobilité et d'usage des services déployés. Ainsi, adapter ces infrastructures de gestion autonome au contexte opérationnel des réseaux qui les supportent est essentiel afin de pouvoir garantir leur efficacité et minimiser leur coût.

Dans ce contexte, nous nous intéressons en particulier à la fonction de surveillance qui est nécessaire aux gestionnaires autonomes dans leur processus de décision. Plus spécifiquement, dans le cas de la surveillance d'informations agrégées, les systèmes de gestion autonome proposent de définir d'une manière statique (ou non-adaptative) le protocole d'agrégation à utiliser. Ainsi, un seul protocole d'agrégation est utilisé pour la collecte et le traitement des informations de gestion quelle que soit la nature du réseau géré et celle de ses informations. Cependant, il est établi dans la littérature que chacun des protocoles représentant une catégorie d'agrégation se comporte différemment et surpasse les autres protocoles en fonction du contexte dans lequel il opère. Il est donc nécessaire de prendre en compte le contexte opérationnel pour garantir l'efficacité du service de surveillance et ainsi celle du système de gestion autonome.

Dans ce but, nous proposons dans cet article une infrastructure de gestion autonome qui s'auto-adapte à l'état opérationnel du réseau qu'elle contrôle en sélectionnant dynamiquement l'approche de gestion adaptée au contexte opérationnel courant. Nous instancions cette infrastructure dans le cas de la fonction de surveillance décentralisée en proposant un nouveau mécanisme d'agrégation auto-adaptatif que nous nommons SAAM (*Self-Adaptive Aggregation Mechanism*). SAAM choisit parmi les protocoles d'agrégation typiques celui qui maximise sa performance et minimise son coût dans le contexte courant. Ce dernier est régi par des propriétés comme la dynamique du réseau, sa taille et également la dynamique des informations surveillées. Pour obtenir une vue de ce contexte et quantifier le coût et la performance de chacun de ces protocoles, nous nous fondons sur les résultats d'une étude de performance que nous avons menée dans nos travaux précédents. L'architecture de SAAM est elle-même autonome et repose sur (1) la logique floue, avec deux contrôleurs qui estiment respectivement le coût et la performance de chacun des protocoles d'agrégation implémentés, et (2) le mécanisme de prise de décision multi-attribut SAW pour calculer l'utilité de chaque protocole.

Nous proposons ici de valider SAAM en évaluant d'abord la validité de son modèle flou, puis son coût de communication en termes de messages d'adaptation et son temps de latence causé par l'attente des nouveaux AM avant de commencer l'agrégation. Ensuite, nous comparons SAAM aux quatre approches d'agrégation dans différents scénarios de test, en considérant la maximisation de l'utilité qui intègre leur coût et performance.

La suite de cet article est organisée comme suit. La section II mentionne les travaux connexes à notre proposition. La section III présente l'architecture de SAAM ainsi que chacun de ses composants. La section IV présente les résultats de validation que nous avons obtenus par simulation. Enfin, la section V conclut ce travail et présente les travaux à venir.

II. Travaux existants

A. Approches adaptatives de gestion

Notre contribution s'inscrit dans le cadre de la gestion autonome, où divers systèmes ont été récemment proposés afin de réduire la complexité de la gestion pour l'administrateur humain. Les systèmes de gestion existants, qu'ils soient fondés sur les plans de connaissance comme MANKOP [1] et Sophia [2] ou sur le modèle P2P comme Astrolabe [3] ou l'approche de Leitao *et al.* [4], offrent un premier degré d'autonomie mais ne sont pas auto-adaptatifs.

Pour résoudre cette problématique, des travaux plus récents proposent de combiner différentes techniques en fonction du contexte. Par exemple, dans l'un des travaux les plus récents et les plus proches du notre qu'est celui de Maurel *et al.* [5], [6], les auteurs ont défini une architecture de gestion de tâches d'administration. Le contrôleur d'adaptation de cette architecture observe et analyse son environnement et décide d'une manière dynamique de créer, observer, mettre à jour ou supprimer une tâche. Bien que le principe général de cette étude se rapproche du notre, nous ne nous intéressons pas à l'ordonnancement et adaptation de tâches élémentaires, mais à l'adaptation de toute une fonctionnalité qu'est la reconfiguration du plan de gestion.

Dans [7], les auteurs présentent une approche de gestion adaptative basée sur le concept de *strategy-tree* pour la gestion d'un centre de données. Cette approche vise à maximiser le profit cumulatif net hebdomadaire engendré par ce centre suite au traitement de requêtes, tout en limitant les violations du contrat de niveau de service (SLA pour *Service Level Agreement*). Trois stratégies sont définies pour le traitement des requêtes, permettant de minimiser les violations du SLA, de maximiser le profit ou un compromis entre les deux. En fonction de leur performance, le système décide périodiquement et dynamiquement s'il faut changer de stratégie au prochain cycle. Les premiers résultats de validation ont montré que, comparée aux stratégies exécutées individuellement, l'approche proposée offre globalement une performance comparable mais n'améliore pas toujours le profit.

Le travail présenté dans [8] propose un mécanisme de surveillance auto-adaptatif et une ébauche de son architecture. Ce travail reprend et étend [9] qui vise à gérer l'adaptabilité de la tâche de surveillance active (*polling*) dans une infrastructure CIM/WBEM. Les auteurs de [8] ont défini également trois critères que doit prendre en compte tout mécanisme de surveillance auto-adaptatif : (1) la gouvernabilité qui est la capacité à détecter le besoin d'adapter la surveillance, (2) l'adaptabilité qui est la capacité à exécuter les opérations permettant de modifier l'approche de surveillance courante et, finalement, (3) la configurabilité qui est la capacité d'adapter dynamiquement le comportement des mécanismes de surveillance utilisés sans interruption du service proposé. La conception et la validation du mécanisme sont présentées comme étant des perspectives.

B. Protocoles d'agrégation décentralisés

Dans cette étude, nous nous fondons sur des approches d'agrégation représentatives de chacune des catégories que nous avons développées et comparées dans [10]. Les approches considérées sont : un protocole fondé sur la rumeur, un protocole fondé sur un arbre et une vue située.

1) Arbre: il s'agit d'un protocole proactif qui utilise le processus d'agrégation de GAP [11] sur un *overlay* P2P structuré [12]. Le fonctionnement de ce protocole d'agrégation est illustré dans l'algorithme 1. Deux threads sont exécutés sur chaque gestionnaire autonome (AM) : un thread actif qui démarre le processus d'agrégation en envoyant la valeur de i à son parent dans l'arbre et un thread passif qui attend passivement la réception de messages venant des autres AM pour pouvoir les traiter. Initialement, chaque AM i sélectionne son parent j dans l'arbre grâce à la méthode $getParent()$. Puis, il lui envoie un message <agrégat, poids> où le poids représente le nombre de valeurs utilisées dans le calcul de l'agrégat (a.1 et a.2). Ensuite, chaque AM i qui reçoit un message de son fils j calcule un nouvel agrégat partiel en utilisant une fonction d'agrégation donnée, met à jour son état local $state_i$, puis le transfère à son parent p dans l'arbre (b.2 à b.5). Si l'AM i est la racine de l'arbre, il attend jusqu'à la réception des agrégats partiels de tous ses fils puis le diffuse avec le système de multicast applicatif sur tous les AM ayant participé au calcul de cet agrégat (b.6 à b.9).

Algorithme 1 : Agrégation fondée sur un arbre

(a) Thread actif

```
1: p←getParent()
2: send (< X_i, 1 >, p)
3: receive(X_j, j)
4: state_i←update(X_j)
```

(b) Thread passif

```
1: loop
2:     receive (< X_j, w_j >, j)
3:     state_i←update(X_j, w_j)
4:     p←getParent()
5:     send (< X_i, w_i >, p)
6:     if (getParent()=null) then
7:         attendre jusqu'à la réception de tous
           les agrégats
8:         diffuse (X_i)
9:     end if
10: end loop
```

2) Rumeur: c'est un protocole d'agrégation proactif fondé sur la rumeur dont le fonctionnement est similaire au protocole basique de Jelasity *et al.* [13]. Ce protocole est illustré dans l'algorithme 2. À l'aide de la méthode $getNeighbors(1)$ qui permet de sélectionner d'une manière aléatoire et uniforme un voisin direct j, chaque AM i envoie à son voisin j son agrégat partiel X_i (a.2 et a.3). Quand un AM i reçoit d'un AM j un message de réponse (X_j, j), il calcule un agrégat partiel selon une fonction d'agrégation donnée et met à jour son état local (a.4 et a.5). Ensuite, l'AM attend le début du prochain cycle pour répéter le même processus (ligne a.6). Ce processus continue jusqu'à ce que tous les AM convergent vers un même agrégat global.

Algorithme 2 : Agrégation fondée sur la rumeur

(a) Thread actif

```
1: loop
2:     j←getNeighbors(1)
3:     send (X_i, j)
4:     receive(X_j, j)
5:     state_i←update(X_j)
6:     wait(round_duration)
7: end loop
```

(b) Thread passif

```
1: loop
2:     receive(X_j, j)
3:     send (X_i, j)
4:     state_i←update(X_j)
5: end loop
```

3) Vue située: un protocole d'agrégation typique de vue située se fonde sur le protocole d'adhésion HyParView utilisé dans l'approche de gestion P2P de Leitao *et al.* [4] et sur le protocole Propagate2All [14]. Chaque AM exécute l'algorithme 3 pour collecter les informations de ses voisins à h-sauts et calculer un agrégat partiel représentant sa vue.

Un AM i initialise le processus d'agrégation en envoyant un message de requête vers tous ses voisins directs connus par la méthode $getNeighbors(all)$ (a.1 et a.2). Ce message contient un paramètre h qui représente le nombre maximal de sauts qu'une requête peut effectuer. Un AM i qui reçoit une requête venant d'un autre AM j vérifie s'il n'a pas répondu auparavant à cette requête (b.2). Si cette condition est vérifiée, i répond directement à j en lui envoyant sa valeur locale brute (X_{raw_i}, j) (b.3). Ensuite, l'AM i décrémente la valeur de h, vérifie si le nombre maximal de sauts n'est pas atteint ($h \neq 0$), puis transmet à son tour la requête reçue à tous ses voisins directs (b.4 à b.8). Quand l'AM i demandeur d'informations reçoit une réponse (X_{raw_j}, j) d'un autre AM j, il calcule un nouvel agrégat partiel, puis met à jour sa vue locale (a.3 et a.4).

Algorithme 3 : Agrégation fondée sur une vue située

(a) Thread actif

1: $\mathbb{D}_i \leftarrow$getNeighbors(*all*)
2: send $(h, \mathbb{D}_i)$
3: receive(X_{raw_j}, j)
4: $state_i \leftarrow$update(X_{raw_j})

(b) Thread passif

1: **loop**
2: receive(h, j)
3: send (X_{raw_i}, j)
4: $h \leftarrow h - 1$
5: **if** $(h > 0)$ **then**
6: $\mathbb{D}_i \leftarrow$getNeighbors(*all*)-{j}
7: send($h, \mathbb{D}_i$)
8: **end if**
9: **end loop**

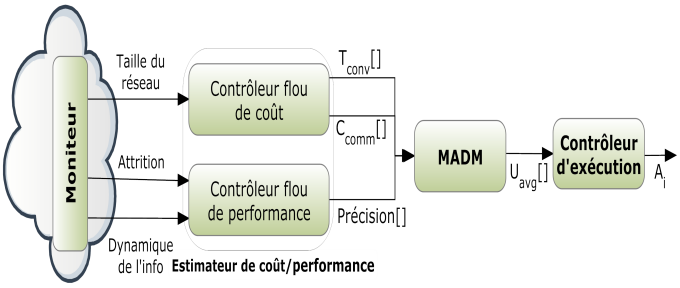

FIGURE 1. Architecture générale de SAAM

Algorithme 4 : Algorithme d'adaptation SAAM

1: diffuse("reqNeighInfo")
2: $<A_{curr}$,remainingAdaptTime$>$ $\leftarrow$ getNeighInfo()
3: executeApproach(A_{curr})
4: wait($remainingAdaptTime$)
5: **loop**
6: $State \leftarrow$ getState()
7: **for** each approach i **do**
8: $D_i \leftarrow$ costPerfEstimator($State$)
9: $U_i \leftarrow$ getUtility(D_i,W)
10: **end for**
11: diffuse("reqUtilities")
12: $U_{avg}[\,] \leftarrow$ getAverageUtilities()
13: $U_{max} \leftarrow max(U_{avg}[\,])$
14: **if** $(A_{max} \neq A_{curr}$ and $U_{max} - U_{curr} > GainT)$ **then**
15: stopApproach(A_{curr})
16: executeApproach(A_{max})
17: **end if**
18: wait($adaptationRound$)
19: **end loop**

III. UNE INFRASTRUCTURE DE GESTION ADAPTATIVE

Notre approche, appelée SAAM (*Self-Adaptive Aggregation Mechanism*) est un mécanisme décentralisé et adaptatif pour l'agrégation de données. Il permet d'exécuter quatre protocoles d'agrégation différents (un protocole d'arbre, un protocole de rumeur et deux protocoles de vue située, à un saut (SV1) et à deux sauts (SV2)) et sélectionne dynamiquement le plus performant en fonction de l'état courant du réseau et des informations à gérer.

A. Architecture et algorithme d'adaptation

L'architecture de SAAM, illustrée sur la figure 1, se fonde sur une boucle de contrôle MAPE (*Monitor, Analyse, Plan, Execute*). Il s'agit d'une architecture décentralisée qui est exécutée sur chaque gestionnaire autonome. Elle infère périodiquement l'état du réseau, évalue la performance de chacun des protocoles d'agrégation considérés et choisit le meilleur d'entre eux via les quatre composants suivants : (1) un moniteur décentralisé qui collecte les données de surveillance puis estime la taille du réseau, sa dynamique et celle de ses informations ; (2) un estimateur de coût et de performance qui utilise deux contrôleurs flous pour évaluer chacun des protocoles d'agrégation pris en compte ; (3) un moteur de prise de décision qui utilise une somme pondérée afin de calculer localement l'utilité des protocoles ; (4) un contrôleur d'exécution qui vérifie si le gain en termes d'utilité est assez important pour interrompre l'exécution du protocole courant et en changer.

Les principales étapes du processus d'adaptation permettant de réaliser les tâches des blocs fonctionnels de notre infrastructure sont données dans l'algorithme III-A. Quand un nouveau nœud arrive dans le système, son AM réalise d'abord une étape d'initialisation (lignes 1 à 4) qui lui permet de se synchroniser avec les autres AM de son voisinage : il communique avec son voisinage pour déterminer le protocole d'agrégation courant et synchroniser sa prochaine prise de décision avec eux. Ensuite, il exécute une phase d'adaptation (lignes 5 à 19) durant laquelle il calcule l'utilité de chaque protocole d'agrégation à disposition (lignes 6 à 10) puis communique avec son voisinage (ligne 11 à 13) pour calculer l'utilité moyenne de chaque approche. Enfin, si l'utilité moyenne d'un protocole surpasse d'un seuil donné l'utilité du protocole courant, l'AM change sa stratégie d'agrégation (ligne 14 à 17).

B. Estimateur de coût et de performance

En se fondant sur les informations d'état collectées par le moniteur, ce composant utilise deux contrôleurs flous (FLC) décrits dans la figure 2 pour estimer le coût et la performance des protocoles d'agrégation arbre, rumeur, SV1 et SV2. Un FLC utilise la logique floue [15] qui est un formalisme servant à représenter et manipuler une connaissance heuristique humaine. Elle permet une prise de décision simple et à faible coût en se fondant sur des informations incertaines ou imprécises.

Le premier FLC considère comme entrée la taille du réseau en termes de **nombre de nœuds** (NS) et comme sortie le coût estimé de chaque protocole en termes de **temps de conver-**

gence (CT) et de **nombre de messages** (CC) d'agrégation. Quant au deuxième, il considère comme entrées le niveau de dynamique du réseau en termes de **taux d'attrition** (ND) et de ses informations de gestion en termes de **fréquence de changement de leur valeur** (ID), et comme sortie la performance du protocole en termes de **précision des agrégats** (AA). Chaque FLC utilise trois opérations : la fuzzification qui consiste à traduire les informations d'état en variables floues, l'inférence qui consiste à qualifier l'utilité d'un protocole au regard de ces variables et la défuzzification qui consiste à donner une valeur d'utilité rélle à chaque protocole. Une première version de la conception de cet estimateur est présentée dans [16].

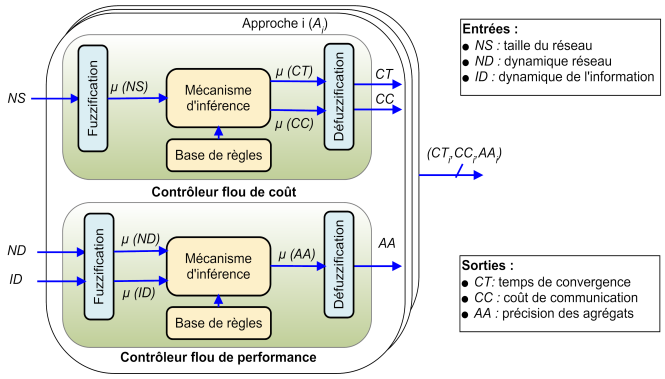

FIGURE 2. Architecture de l'estimateur de coût et de performance

La fuzzification des entrées/sorties des deux FLC est donnée dans la figure 3. Nous avons défini les sous-ensembles flous de chaque variable en nous fondant sur les résultats de simulation présentés dans [10]. La précision et la taille du réseau sont définies par quatre sous-ensembles flous, du plus petit au plus grand. Cette partition est faite de manière à mettre en évidence les différences de coût et de performance entre un protocole d'agrégation et un autre. Concernant la dynamique du réseau, nous proposons d'abstraire cette métrique par la somme des taux d'arrivée et de départ de nœuds normalisée sur un intervalle $[0, 100]$.

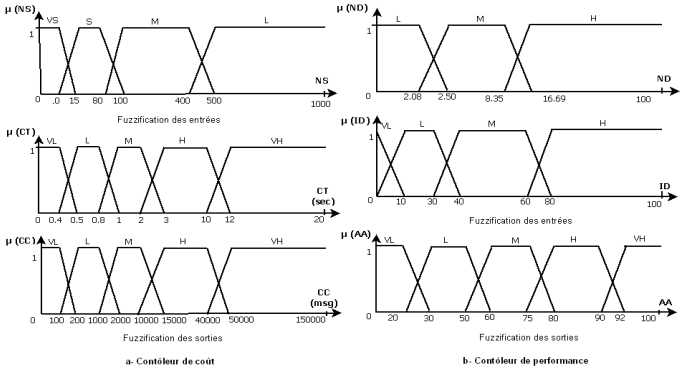

FIGURE 3. Fuzzification des entrées/sorties des contrôleurs flous [17]

Concernant le moteur d'inférence, nous utilisons la méthode de Mamdani [18] qui fournit de meilleurs résultats et qui est également la méthode la plus utilisée. Nous avons établi pour chaque contexte d'exécution une règle d'inférence précise en nous fondant sur les résultats de simulation présentés dans [10]. Ainsi, en fonction des sous-ensembles flous en entrée des FLC, nous pouvons déduire directement ceux correspondant à la sortie. Ces règles sont données par les tableaux I et II. Par exemple, dans le cas d'une faible dynamique des informations de gestion et d'une forte dynamique des nœuds, la précision des agrégats calculés par l'arbre est très faible. Ce constat correspond à la règle suivante : `si ID = L et ND = H alors AA = VL`.

TABLE I
RÈGLES D'INFÉRENCE DU CONTRÔLEUR FLOU DE COÛT [17]

		Arbre		Rumeur		SV1		SV2	
		CT	CC	CT	CC	CT	CC	CT	CC
NS	VS	L	VL	M	VL	VL	VL	VL	VL
	S	M	VL	H	M	L	L	L	M
	M	M	L	VH	M	L	M	L	H
	H	M	M	VH	H	L	H	L	VH

TABLE II
RÈGLES D'INFÉRENCE DU CONTRÔLEUR FLOU DE PERFORMANCE [17]

		Arbre			Rumeur			SV1			SV2		
		ND			ND			ND			ND		
		L	M	H	L	M	H	L	M	H	L	M	H
ID	VL	VH	VL	VL	VH	H	M	M	M	M	H	M	M
	L	VH	VL	VL	VH	H	M	L	L	L	M	M	M
	M	VH	VL	VL	H	H	M	L	L	L	M	M	L
	H	H	VL	VL	H	M	L	L	L	L	M	M	L

Concernant la défuzzification, nous utilisons la méthode du centre de gravité [19] qui calcule la valeur numérique exacte de chaque sortie d'un contrôleur. Cette méthode est la plus coûteuse en termes de calcul mais donne la meilleure estimation pour les sorties des FLC.

C. Moteur de prise de décision multi-attributs

La prise de décision multi-attribut (MADM pour *Multiple Attribute Decision Making*) [20] est une technique qui sélectionne une action parmi différentes alternatives selon divers attributs. Cela permet de considérer plusieurs dimensions lors de l'évaluation des différentes alternatives qui se présentent. Le moteur que nous proposons utilise une méthode MADM où les actions sont l'utilisation d'un protocole d'agrégation donné et les attributs les coûts et les utilités de chaque protocole. Ce moteur est constitué d'un composant de calcul d'utilité et d'un processus d'agrégation collaboratif, illustrés sur la figure 4.

Le composant de calcul d'utilité utilise la méthode standard de pondération par simple addition (SAW) pour la prise de décision [21]. Cette méthode répond à nos besoins car elle intègre les critères positifs (gain) et négatifs (coût) de chaque approche. Le processus de calcul de SAW se divise en deux étapes que sont la normalisation des valeurs d'attributs et le calcul de l'utilité U_i de chaque alternative A_i. Celle-ci est obtenue par la somme des valeurs normalisées r_{ij} de chaque attribut multipliées par leurs poids respectifs w_j. Nous

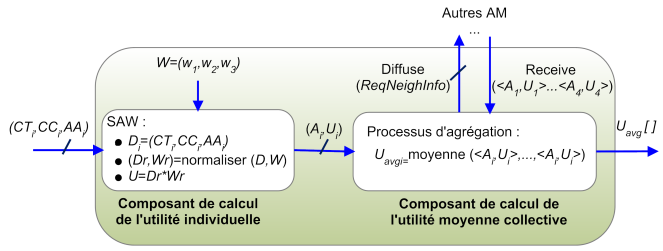

FIGURE 4. Architecture du moteur de prise de décision

considérons deux critères négatifs de coût que sont le temps de convergence (CT) et le coût de communication (CC), et un critère positif de gain qu'est la précision des agrégats (AA). Le vecteur des poids de ces critères est fixé par l'opérateur du réseau selon qu'il désire privilégier l'approche la moins coûteuse, celle offrant la meilleure performance ou celle permettant un compromis coût et performance. Ainsi, la valeur locale d'utilité U_i d'une approche d'agrégation i est calculée avec l'équation 1, dans laquelle w_1, w_2 et w_3 représentent les poids fixés par l'utilisateur pour, respectivement, CT, CC et AA.

$$U_i = w_1 * CT_i + w_2 * CC_i + w_3 * AA_i \quad (1)$$

Le processus d'agrégation collectif calcule l'utilité moyenne de chaque approche. Nous utilisons le protocole SV1 qui a montré au cours de nos simulations qu'il était suffisant pour voir converger l'ensemble des AM vers le même choix de protocole. Pour cela, chaque AM interroge ses voisins directs pour collecter l'utilité de chaque protocole. Ensuite, la valeur moyenne d'utilité de chaque protocole est calculée par l'équation 2, où k est le nombre de voisins. Le protocole ayant l'utilité moyenne la plus grande est le protocole candidat.

$$U_{avg} = \frac{1}{k} \sum_{j=1}^{k} U_i \quad (2)$$

D. Contrôleur d'exécution

Le contrôleur d'exécution compare l'utilité du protocole d'agrégation courant avec l'utilité du protocole candidat, puis il décide s'il est plus intéressant de conserver la stratégie courante ou d'en changer pour garantir une meilleure performance et/ou un moindre coût. Ce contrôleur d'exécution est un automate à états finis ayant deux états A_{curr} et $A_{max} \in \{$arbre, rumeur, SV1, SV2$\}$ comme le montre la figure 5.

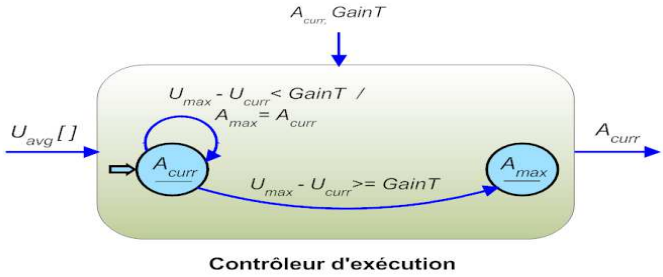

FIGURE 5. Architecture du contrôleur d'exécution des actions planifiées

Le premier état A_{curr} est l'état initial et correspond à l'exécution du protocole courant tandis que A_{max} est un état final provoquant l'exécution du protocole candidat. La transition depuis A_{curr} vers A_{max} s'effectue $U_{max} - U_{curr} \geq GainT$. Cette opération interrompt alors l'exécution du protocole courant et lance le protocole candidat. Dans les autres cas, l'AM continue d'utiliser la même stratégie d'agrégation en attendant l'exécution d'une nouvelle itération de l'algorithme d'adaptation.

IV. VALIDATION DE SAAM

Afin de valider SAAM, nous avons mis en œuvre un *framework* qui permet d'utiliser, soit SAAM pour sélectionner dynamiquement d'une manière autonome la meilleure approche d'agrégation à utiliser, soit l'une des approches d'agrégation qui y sont implémentées et que l'administrateur humain choisit d'une manière statique.

A. Architecture de l'environnement de test

Le *framework* que nous proposons intègre les blocs fonctionnels de SAAM ainsi que les outils sur lesquels celui-ci se fonde. Nous y avons intégré l'implémentation des quatres approches d'agrégation : arbre, rumeur, SV1 et SV2. Dans le cas de la rumeur et de la vue située, nous définissons par nous-mêmes les relations de voisinage, le processus de maintenance et celui de la communication des agrégats. En revanche, dans le cas de l'arbre, nous utilisons la hiérarchie de la DHT Pastry et le système de multicast applicatif Scribe, pour diffuser les agrégats de la racine sur les autres AM. Pour cela, nous utilisons FreePastry [1]. Nous profitons du simulateur intégré de FreePastry pour la création de nœuds virtuels et l'échange de messages. FreePastry est séparable en deux couches : une couche de DHT que nous utilisons dans le protocole fondé sur un arbre et une couche *overlay* générique que nous utilisons dans le cas de la vue située et de la rumeur, pour faire des topologies non structurées. Concernant l'implémentation des contrôleurs flous, nous utilisons la librairie jFuzzyLogic [2].

Les paramètres de simulation issus de la literature que nous utilisons sont donnés dans le tableau III.

Concernant les préférences de l'utilisateur, nous considérons entre autres que le vecteur de poids W contient trois valeurs correspondant, respectivement, à CT, CC et AA. Nous exécutons SAAM en considérant que l'approche d'agrégation privilégiée est celle donnant la précision des agrégats la plus élevée ($W_{accuracy} = (0.1, 0.1, 0.8)$). SAAM décide également de changer d'approche d'agrégation lorsque l'écart entre l'utilité de l'approche candidate et celle de l'approche courante soit supérieur à un seuil $GainT$ fixé à 1%.

B. Résultats d'évaluation

Nous proposons d'évaluer SAAM en vérifiant la validité de son modèle flou, en mesurant le coût relatif à son processus d'adaptation, puis en testant sa performance et sa capacité à choisir l'approche d'agrégation optimale.

1. FreePastry project, http://freepastry.org.
2. Pablo Cingolani *et al.*, jFuzzyLogic, http://jfuzzylogic.sourceforge.ne

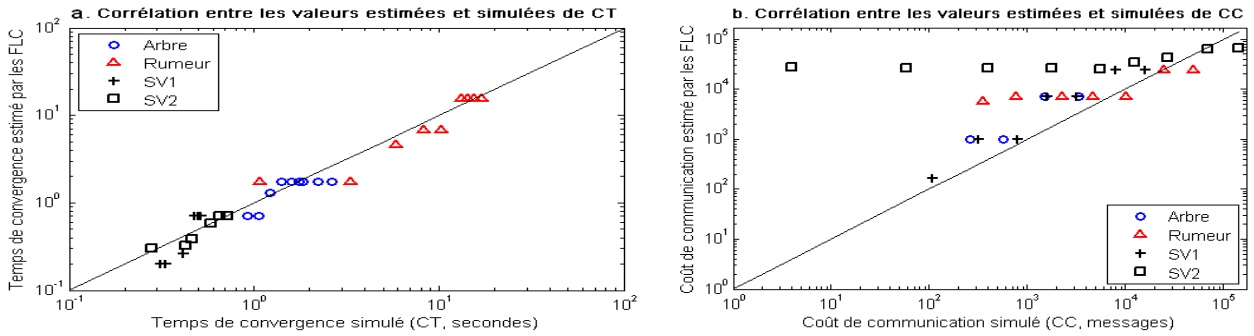

FIGURE 6. Comparaison des temps de convergence et du coût de communication obtenus par le contrôleur flou et par simulation

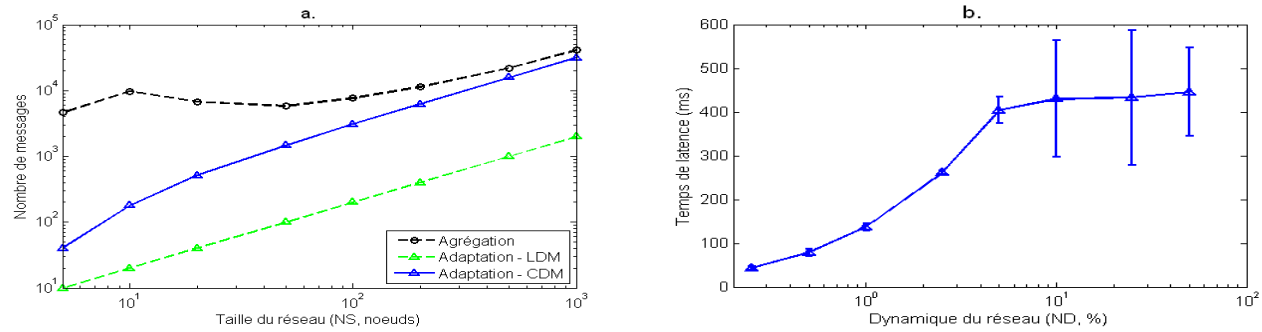

FIGURE 7. Coût de communication (messages) et temps de latence dans SAAM

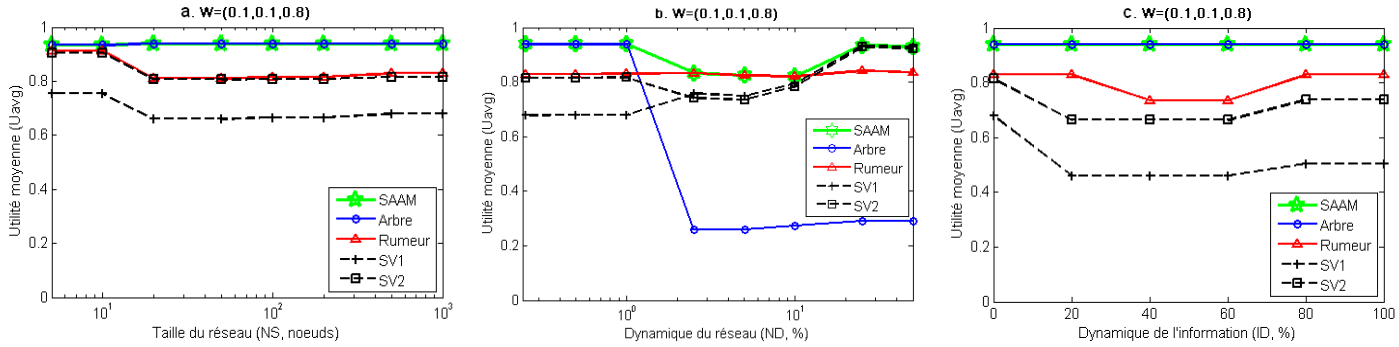

FIGURE 8. Maximisation de l'utilité en fonction de la taille du réseau, sa dynamique et celle des informations de gestion

Afin d'évaluer la précision des résultats de simulation obtenus à travers de multiples exécutions de chaque approche, nous calculons pour chacune des valeurs des courbes un intervalle de confiance à 95%. Nous représentons ces intervalles de confiance sur les courbes par des barres définissant la marge d'erreur possible.

1) Validité du modèle flou: Nous évaluons la validité du modèle flou utilisé par l'estimateur de coût et de performance car ce composant est l'un des plus importants de SAAM. Si les valeurs estimées par les contrôleurs flous ne sont pas précises, des décisions erronées sur le choix de la meilleure approche à utiliser pourraient être prises. Pour cela, nous mesurons la précision des métriques de coût et de performance estimées directement par les contrôleurs flous, puis nous les comparons aux valeurs obtenues par simulation dans [10]. En faisant varier le nombre de nœuds dans un réseau statique, nous obtenons, en sortie du contrôleur flou de coût, le résultat illustré dans la figure 6.

La figure 6.a montre la corrélation entre les valeurs estimées et simulées du temps de convergence. Le résultat direct estimé par le contrôleur de coût pour le temps de convergence correspond à celui obtenu par simulation, que ce soit au niveau de sa distribution ou au niveau de ses valeurs numériques.

La figure 6.b montre la corrélation entre les valeurs estimées et simulées pour le coût de communication. Nous observons une forte corrélation entre les résultats des deux types de mesure, sauf dans le cas de SV2 où le coût de communication est surestimé par le FLC lorsque le réseau est de petite taille. Dans le cas particulier de SV2, l'ordre des coûts est toujours respecté entre les approches. Ainsi, cela n'affecte pas SAAM dans sa sélection de la meilleure approche à utiliser.

Globalement, les valeurs estimées par les contrôleurs flous sont proches de celles que nous avons obtenues par simulation dans [10] et ce, pour les quatre approches d'agrégation étudiées. Cette cohérence des résultats est vérifiée, que ce soit pour le temps de convergence ou le coût de communication. Cela montre la précision des valeurs estimées par les contrôleurs flous et, par la même occasion, la validité des

TABLE III
PARAMÈTRES DE CONFIGURATION

	Paramètre	Valeur
Topologie	Modèle de topologie du réseau	Euclidien
	Freq. Maintenance topologie	60 secondes
	Degré de voisinage	8
	Nombre maximal de nœuds	1000
Agrégation	Fréq. changement des informations	1 seconde
	Fonction d'agrégation	Moyenne
	Cycle d'agrégation	1 minute
	Cycle de la rumeur	600 millisecondes
Adaptation	Cycle d'adaptation	10 minutes
	Approche par défaut	SV1
	Ordre (réseau statique)	SV1→SV2→Arbre→Rumeur
	Ordre (réseau dynamique)	SV1→SV2→Rumeur→Arbre
Dynamique	Taux d'arrivée des nœuds	Poisson : $\lambda \in [0.0041, 0.41]$
	Durée de vie des nœuds	exponentielle : $\mu \in [8.33 \times 10^{-6}, 8.33 \times 10^{-4}]$
	Niv. dynamique de l'information	$\alpha \in [0, 1]$
Préf.	Vecteur de poids	$W_{accuracy}$ = (0.1, 0.1, 0.8)

règles d'inférence, des sous-ensembles flous et des méthodes sur lesquels les contrôleurs se fondent.

2) Coût d'adaptation: SAAM est conçu pour optimiser le coût et la performance de l'agrégation, mais son processus d'adaptation engendre un coût supplémentaire lié à son fonctionnement. Nous évaluons ce coût en termes de messages d'adaptation et de temps de latence.

a) Coût de communication: C'est le nombre de messages d'adaptation nécessaires au fonctionnement de SAAM. Ce coût de communication inclut : (1) les messages de synchronisation, utilisés par SAAM lorsque de nouveaux nœuds arrivent dans le système et récupèrent les informations d'initialisation de l'agrégation à partir de leur entourage, et (2) les messages de collaboration, utilisés par SAAM pour calculer l'utilité moyenne de chaque approche d'agrégation en collaborant avec d'autres AM. Les résultats relatifs à ce critère sont représentés dans la figure 7.a qui compare le nombre de messages d'adaptation obtenu lors d'une prise de décision collaborative (CDM) à celui obtenu dans le cas d'une prise de décision locale (LDM) sans collaboration, puis au nombre de messages d'agrégation engendrés par SAAM.

Nous observons une évolution linéaire du nombre de messages d'adaptation et d'agrégation en fonction de la taille du réseau. Le nombre de messages d'adaptation de CDM est plus grand que le nombre de messages d'adaptation de LDM et plus petit que le nombre de messages d'agrégation. Le nombre de messages d'agrégation est de 2 à 100 fois plus grand que le nombre de messages d'adaptation. Le nombre de messages de LDM est très petit car il n'y a pas de collaboration. En revanche, dans le cas de CDM, le nombre de messages d'adaptation est plus grand car SAAM en a besoin pour exécuter ses opérations de collaboration et de synchronisation.

En résumé, le nombre de messages d'adaptation dont SAAM a besoin est raisonnable en comparaison avec le nombre de messages utilisés par SAAM pour l'agrégation d'une seule information de gestion avec une fréquence d'adaptation assez élevée.

b) Temps de latence: Les informations de gestion doivent être idéalement collectées en temps réel pour permettre une prise de décision rapide et efficace. Ainsi, il est impératif de minimiser le temps de latence et de traitement des AM. L'AM d'un nouveau nœud doit d'abord collecter des informations d'initialisation (type d'approche utilisée) à partir de son entourage. Pendant cette période, les informations sur l'état du réseau ne sont pas à jour. Elles peuvent conduire à des décisions erronées. Il est donc important dans notre cas d'estimer le temps d'attente de chaque AM avant l'agrégation. En mesurant cette métrique dans un environnement dynamique, nous obtenons le résultat illustré dans la figure 7.b.

Nous observons dans cette figure une évolution linéaire du temps moyen de latence en fonction du niveau de dynamique du réseau. Nous remarquons également que plus la dynamique est grande, plus l'erreur est importante. Le temps moyen de latence ne dépasse pas 450 millisecondes. Cette valeur est presque négligeable en la comparant, par exemple, à la durée d'un cycle d'adaptation qui est de 10 minutes. Ce temps de latence est proportionnel au taux d'arrivée des nœuds car les nouveaux nœuds doivent tout d'abord contacter leurs voisins pour se synchroniser avec eux.

En résumé, le temps moyen d'attente d'un AM engendré par la synchronisation dans SAAM est acceptable comparé au temps de convergence ou à la durée d'un cycle d'adaptation qui sont plus grands.

3) Maximisation de l'utilité: Cette mesure permet de comparer la performance globale de SAAM à celle des autres approches d'agrégation sur lesquelles il se fonde. Nous vérifions dans quelle mesure SAAM choisit, sous contrainte de seuil, toujours l'approche d'agrégation candidate dont l'utilité moyenne est maximale. Pour cela, nous comparons l'utilité moyenne de SAAM à celles obtenues en utilisant les autres approches d'agrégation. Nous étudions le comportement de SAAM en évaluant l'impact de trois critères sur son utilité : la taille du réseau (NS), la dynamique du réseau (ND) et, finalement, la dynamique des informations (ID). Les résultats de simulation obtenus dans le cas de $W_{accuracy}$ sont représentés, respectivement, dans les figures 8.a, 8.b et 8.c.

a) Impact de la taille du réseau: Nous considérons pour ce critère un réseau statique où les informations sont fixées d'une manière aléatoire et uniforme sur chaque AM.

Dans la figure 8.a, nous remarquons que l'utilité moyenne des approches étudiées varie très peu en fonction de la taille du réseau. Nous observons également ici que l'utilité moyenne de l'approche fondée sur la rumeur est égale à celle de SV2 mais elle reste toujours inférieure à celle de l'arbre qui fournit la meilleure utilité quelle que soit la taille de son réseau. Ainsi, SAAM privilégie en permanence l'approche fondée sur un arbre car elle offre la meilleure performance et un faible coût dans un contexte statique. Ainsi, SAAM suit toujours l'approche d'agrégation maximisant son utilité moyenne, quelle que soit la taille du réseau.

b) Impact de la dynamique du réseau: Nous considérons pour ce critère que les informations de gestion sont statiques,

puis nous faisons varier les taux d'arrivée et de départ des nœuds dans le réseau.

Dans la figure 8.b, nous constatons que SAAM utilise l'arbre lorsque le réseau a une très faible dynamique. Cependant, lorsque le réseau est moyennement dynamique, c'est l'approche fondée sur la rumeur qui est utilisée car elle fournit la meilleure utilité. L'utilité de l'arbre d'agrégation chute brusquement dès que le réseau devient dynamique car il est très sensible aux arrivées et départs de nœuds, contrairement aux autres approches qui supportent mieux l'attrition. SAAM utilise SV1 lorsqu'il est fortement dynamique car son utilité moyenne augmente lorsque le réseau est plus dynamique. Cette approche résiste mieux à l'attrition de nœuds dans ce contexte. Ainsi, SAAM suit toujours l'approche d'agrégation maximisant son utilité moyenne, quel que soit le niveau de dynamique de son réseau.

c) Impact de la dynamique de l'information: Nous considérons ici un réseau statique de 1000 nœuds dans lequel nous faisons varier le niveau de dynamique de l'information.

Dans la figure 8.c, nous observons une utilité moyenne de l'arbre est constante et toujours supérieure à celles des autres approches. Dans le cas de la rumeur, son utilité dépasse clairement celle de la vue située mais cela n'est pas suffisant pour qu'elle soit choisie par SAAM, car elle est plus affectée que l'arbre par la dynamique des informations. Ainsi, SAAM suit toujours l'approche d'agrégation maximisant son utilité moyenne, quel que soit le niveau de dynamique des informations de gestion.

V. Conclusion et perspectives

Dans cet article, nous avons proposé le mécanisme d'agrégation auto-adaptatif SAAM qui choisit la meilleure approche d'agrégation parmi les trois classes d'approches standard qui sont un arbre d'agrégation, la diffusion épidémique et une vue située à un saut et à deux sauts. Pour effectuer ce choix, SAAM utilise deux contrôleurs de logique floue qui estiment le coût et la performance de chaque approche d'agrégation en fonction d'informations sur l'état opérationnel du réseau. Ensuite, en utilisant la méthode de prise de décision multi-attribut (SAW), SAAM calcule localement une valeur d'utilité pour chaque approche, puis échange ce résultat avec les autres AM en vue d'estimer collectivement une utilité moyenne par approche. L'approche choisie est alors celle qui maximise l'utilité et dépasse le seuil de gain défini par l'utilisateur. Les résultats de simulations obtenus ont montré que notre approche était valide car (1) la vue du réseau restituée par les contrôleurs flous est suffisamment précise, (2) la fonction d'agrégation choisie par SAAM est celle qui maximise son utilité et (3) le surcoût lié à cette auto-adaptation reste raisonnable en regard du coût lié à l'agrégation.

Les directions de travaux futurs sont de trois ordres. La première consiste à déployer SAAM dans un environnement expérimental pour consolider les résultats obtenus par simulation, notamment concernant la surveillance du réseau qui induit elle-même une latence et imprécision que nous n'avons pas considérées ici. Ensuite, une étude de la stabilité serait nécessaire pour prouver que l'ensemble des nœuds converge vers le même état et que, pour un état du réseau donné, le processus n'oscille pas entre différents protocoles d'agrégation. Enfin, nous envisageons l'utilisation de cette architecture dans un contexte de résilience dans lequel, au sein d'un environnement hostile (présence d'attaquants) ou incertain, notre approche pourrait permettre de maintenir les fonctions de gestion et contrôle dans un état opérationnel et ainsi contribuer à la résilience du réseau.

Références

[1] D. F. Macedo, A. L. dos Santos, G. Pujolle, and J. M. S. Nogueira, "MANKOP : A Knowledge Plane for wireless ad hoc networks," in *NOMS*, 2008, pp. 706–709.

[2] M. Wawrzoniak, L. Peterson, and T. Roscoe, "Sophia : an Information Plane for Networked Systems," *ACM SIGCOMM Computer Communication Review*, vol. vol.34, no. 1, pp. 15–20, 2004.

[3] R. V. Renesse, K. P. Birman, and W. Vogels, "Astrolabe : a robust and scalable technology for distributed system monitoring, management, and data mining," *ACM Transactions on Computer Systems*, vol. 21, no. 2, pp. 164–206, 2003.

[4] J. Leitao, J. Pereira, and L. Rodrigues, "Large-Scale Peer-to-Peer Autonomic Monitoring," in *DANMS*, 2008, pp. 1–5.

[5] Y. Maurel, "CEYLAN : Un canevas pour la création de gestionnaires autonomiques extensibles et dynamiques," Ph.D. dissertation, Université de Grenoble, December 2010.

[6] Y. Maurel, P. Lalanda, and A. Diaconescu, "Towards introspectable, adaptable and extensible autonomic managers," in *ACM/IEEE/IFIP CNSM*, Paris, France, 2011.

[7] B. Simmons and H. Lutfiyya, "Achieving high-level directives using strategy-trees," in *IEEE MACE*, 2009, pp. 44–57.

[8] A. Moui and T. Desprats, "Towards self-adaptive monitoring framework for integrated management," in *IFIP AIMS PhD Workshop*, 2011, pp. 160–163.

[9] A. Moui, T. Desprats, E. Lavinal, and M. Sibilla, "Managing polling adaptability in a CIM/WBEM infrastructure," in *DMTF SVM*, 2010, pp. 1–6.

[10] R. Makhloufi, G. Doyen, G. Bonnet, and D. Gaïti, "Impact of Dynamics on Situated and Global Aggregation Schemes," in *IFIP AIMS*, 2011, pp. 148–159.

[11] M. Dam and R. Stadler, "A generic protocol for network state aggregation," in *Radiovetenskap och Kommunikation*, 2005.

[12] R. Makhloufi, G. Bonnet, G. Doyen, and D. Gaïti, "Towards a P2P-based Deployment of Network Management Information," in *IFIP AIMS*, 2010, pp. 26–37.

[13] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems*, vol. vol.23, no. 3, pp. 219–252, 2005.

[14] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani, "Estimating aggregates on a Peer-to-Peer network," Tech. Rep., 2003.

[15] K. M. Passino and S. Yurkovich, *Fuzzy Control*. Addison Wesley Longman, 1998.

[16] R. Makhloufi, G. Doyen, G. Bonnet, and D. Gaïti, "Towards Self-Adaptive Management Frameworks : the Case of Aggregated Information Monitoring," in *ACM/IEEE/IFIP CNSM 2011*.

[17] ——, "SAAM : a Self-Adaptive Aggregation Mechanism for Autonomous Management Systems," in *IEEE/IFIP NOMS*, 2012.

[18] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[19] W. V. Leekwijck and E. E. Kerre, "Defuzzification : criteria and classification," *Fuzzy Sets and Systems*, vol. 108, no. 2, pp. 159–178, 1999.

[20] C.-L. Hwang and K. Yoon, *Multiple Attribute Decision Making*. Springer-Verlag, 1981.

[21] W. Zhang, "Handover decision using fuzzy MADM in heterogeneous networks," in *IEEE WCNC*, vol. 2, 2004, pp. 653–658.