Agents intelligents et gestion de réseaux ATM[1]

Pierre Conti, Morsy Cheikhrouhou

Corporate Communications Department
Institut Eurécom
BP 193 – 06904 Sophia-Antipolis Cédex – France
[conti,cheikhro]@eurecom.fr

Résumé

Les systèmes de gestion de réseaux manquent de flexibilité, et souffrent d'une complexité croissante. Le paradigme des agents intelligents est supposé pouvoir résoudre ces problèmes, mais les nombreuses controverses qui existent autour de la définition de l'agent intelligent suffisent à démontrer le manque de maturité de ce domaine.
Est-il néanmoins déjà possible d'utiliser quelques principes admis comme étant parties intégrantes de l'agent intelligent pour proposer une solution à des problèmes de gestion de réseaux ? C'est le sujet de l'étude de cas que nous présentons ici, dont l'objectif principal est de tester les possibilités de ces technologies émergeantes en étudiant une solution à base d'agents intelligents pour résoudre un problème relatif à la gestion de configuration. La configuration des réseaux ATM, et plus particulièrement la création d'un PVC, est connue pour être une procédure longue et complexe. Nous proposons ici, en indiquant la démarche que nous avons suivie, une solution à base d'agents intelligents prouvant l'intérêt de cette technologie.

Mots clefs : Gestion de réseaux, Agents intelligents, ATM, PVC

1 Introduction

Avec la croissance très rapide du nombre de réseaux, sont apparues les faiblesses des systèmes de gestion traditionnellement centralisés. Ceux-ci ne peuvent solutionner les problèmes liés à la complexité des configurations, et sont inadaptés à leurs évolutions continuelles. Les entreprises spécialisées dans la fourniture de systèmes de gestion de réseaux (en anglais NMS : Network Management System) cherchent des solutions pour pouvoir gérer des réseaux de toutes tailles, et diminuer la grande expertise indispensable aux ingénieurs réseaux pour maîtriser leurs outils. Grâce aux nombreuses études en cours, il est de plus en plus admis que ces solutions devront passer :
_ par la distribution des tâches de gestion pour éviter les goulots d'étranglement qui se créent autour des systèmes centralisés.
_ par la distribution de l'intelligence, pour pouvoir diminuer la complexité de l'analyse des informations remontant du réseau, et accélérer la réactivité du système de gestion.

[1] Cette recherche a été financée par SwissCom en partenariat avec l'Institut Eurécom dans le cadre du projet DIANA.

_ par la modularité des composants, pour pouvoir s'adapter facilement à l'évolution de ceux-ci.
Depuis quelques temps la tendance est à l'utilisation de technologies d'objets distribués comme CORBA [1] permettant d'augmenter le découpage des systèmes, et à l'utilisation d'architectures hiérarchiques [2] pour diminuer le coût et la charge des machines de gestion.
Ces solutions sont pourtant loin d'être suffisantes. On peut le constater notamment lorsque de nouvelles technologies comme ATM sont introduites dans les réseaux à gérer, où l'on constate que le travail de l'administrateur augmente proportionnellement à l'incapacité du système de gestion à administrer ces systèmes.
Face à ce constat, et si nous excluons les multitudes de solutions spécifiques que proposent maintenant les constructeurs, le concept qui semble le plus prometteur pour résoudre l'ensemble de ces problèmes est celui des Agents Intelligents [3].

Afin de mieux cerner ce concept et étudier la plus-value que peuvent présenter les agents intelligents, nous proposons dans cet article de traiter sous forme d'une étude de cas un des problèmes auxquels se trouvent actuellement confrontés les gestionnaires de réseaux ATM, en développant une solution à base d'agents intelligents.

Dans un premier temps nous décrivons le problème que nous allons traiter, en l'occurrence la gestion automatique de configuration de PVC (Permanent Virtual Connections) en rappelant brièvement le travail que représente la configuration d'un PVC pour l'administrateur de réseaux.
Puis nous présentons succinctement quelques principes de fonctionnement des agents que nous allons utiliser, afin de comprendre comment a été élaborée la solution à notre problème.
Dans la troisième partie (chapite 4) nous expliquons la méthode employée pour développer cette application ; et dans la quatrième (chapitre 5) l'utilisation de cette dernière avec les agents.
Enfin nous concluons sur les extensions possibles de cette solution et les travaux à mener pour pouvoir généraliser les développements à base d'agents.

2 Configuration de PVCs et scénario

La recherche sur l'aide que peuvent apporter les agents intelligents au paradigme ATM, comme pour la création de PVC que nous avons choisi de traiter dans cette étude de cas, est un sujet d'actualité dans le domaine de la gestion de réseaux, si l'on se réfère aux nombreuses publications qui y sont consacrées [4][5][6][7].

Historiquement le paradigme ATM a été sélectionné par l'ITU (Internationnal Telecommunications Union) pour pouvoir répondre à la grande variété de demandes spécifiques, en terme de consommation de ressources, de la multitude d'applications utilisant les réseaux. En dehors de la grande flexibilité qu'apporte ATM en supportant tout type d'application sur un réseau large bande, un des grands avantages des réseaux ATM est de garantir une qualité de service (QoS) sur une connexion fournie à la

demande, comme par exemple une bande passante constante , ou un délai maximum de transmission.

Actuellement très peu d'équipements permettent une utilisation complète et dynamique des services ATM. Aussi, pour que des applications nécessitant une qualité de service particulière puissent fonctionner correctement, il faut au préalable établir un PVC entre les sites devant communiquer en précisant les caractéristiques de cette QoS. Mais si la configuration d'une connexion SVC (Switched Virtual Connection) se fait automatiquement sur l'ensemble des équipements présents sur le chemin entre les deux systèmes à connecter, grâce au protocole de signalisation, il n'en est pas de même pour les PVCs. Pour établir la connexion permanente (PVC) de bout en bout, l'administrateur doit configurer manuellement l'ensemble des commutateurs sur le chemin entre les deux machines à connecter. Parmi les opérations à accomplir, il doit vérifier les VPs (Virtual Paths) utilisables sur chaque commutateur, ainsi que leur disponibilité en ressources (exemple : bande passante), et cela en fonction de la qualité de service exigée pour la connexion demandée . Les caractéristiques de cette qualité de service doivent être spécifiées sur chaque commutateur par un contrat de trafic appelé UPC (Usage Parameter Control) qui sera par la suite attaché au PVC en création.
De plus, la complexité et la durée de la configuration d'un PVC augmentent avec l'hétérogénéité plus ou moins inévitable des systèmes.
Du fait même de leur capacité à traiter des problèmes complexes dans un environnement distribué, les agents intelligents sont apparus comme une solution intéressante pour aider un administrateur de réseaux à configurer les PVCs.
Si certaines expérimentations ont été menées en utilisant des agents mobiles pourvus d'un mécanisme de raisonnement [4] leur permettant de configurer automatiquement les commutateurs sur le chemin du PVC à établir, nous proposons ici une approche utilisant des agents statiques que nous pensons plus économiques en terme de consommation de bande passante, et à même de fonctionner plus rapidement.
Pour cette étude de cas nous nous sommes placés dans le contexte d'un utilisateur ayant souscrit un abonnement auprès d'un fournisseur de service, et pouvant choisir de créer automatiquement une connexion PVC en spécifiant l'application qu'il souhaite utiliser ainsi que l'adresse du site distant avec lequel il veut communiquer. Les applications sont attachées à des QoS prédéterminés, qui sont déjà configurés sur les commutateurs et reconnus par les agents intelligents. Afin de mieux appréhender la méthode utilisée pour réaliser notre solution à base d'agents, nous décrivons dans le chapitre suivant quelques points importants de l'architecture des agents que nous allons utiliser.

3 Les agents DIANA

Si la notion d'agents intelligents n'est pas encore clairement définie, ceux-ci sont déjà présents dans de nombreux domaines et semble présenter un intérêt commercial non négligeable si l'on remarque le nombre croissant d'applications se réclamant à base d'agents intelligents. Si nous pouvons également trouver certaines applications dans le domaine de la gestion de réseaux [8], actuellement seuls des agents logiciels spécifiques ont été construits. Cela provient principalement de la difficulté, voir même de l'impossibilité d'après certains spécialistes [9] de créer des agents génériques.

Néanmoins notre approche consiste à viser cette généricité tout au moins en l'étendant au domaine de la gestion de réseaux, avec pour objectif de proposer à terme une architecture de NMS entièrement basée sur une technologie agents intelligents.

L'architecture des agents intelligents que nous développons utilise pour satisfaire à cette exigence de généricité, le concept de modules de compétences(skill modules). Associé aux rôles de l'agent que nous détaillerons au chapitre 4.1, le module de compétences est un élément clef du fonctionnement de l'agent. Contenant la partie opérationnelle des applications de gestion de réseaux développées, il est manipulé par le mécanisme délibératif de l'agent. Lorsqu'il doit accomplir une tâche, c'est à dire exécuter une série d'opérations l'agent, grâce aux modules de compétences, a la possibilité de demander à d'autres agents de lui fournir le ou les modules de compétences nécessaires à l'exécution de cette tâche (*transfert de compétences* Figure 1), ou de leur demander d'exécuter eux-mêmes celle-ci (*délégation*).
Cette décision peut-être dictée par sa disponibilité, calculée en fonction du seuil maximal de temps CPU qui lui est autorisé, ou en fonction d'autres informations influençant son comportement.

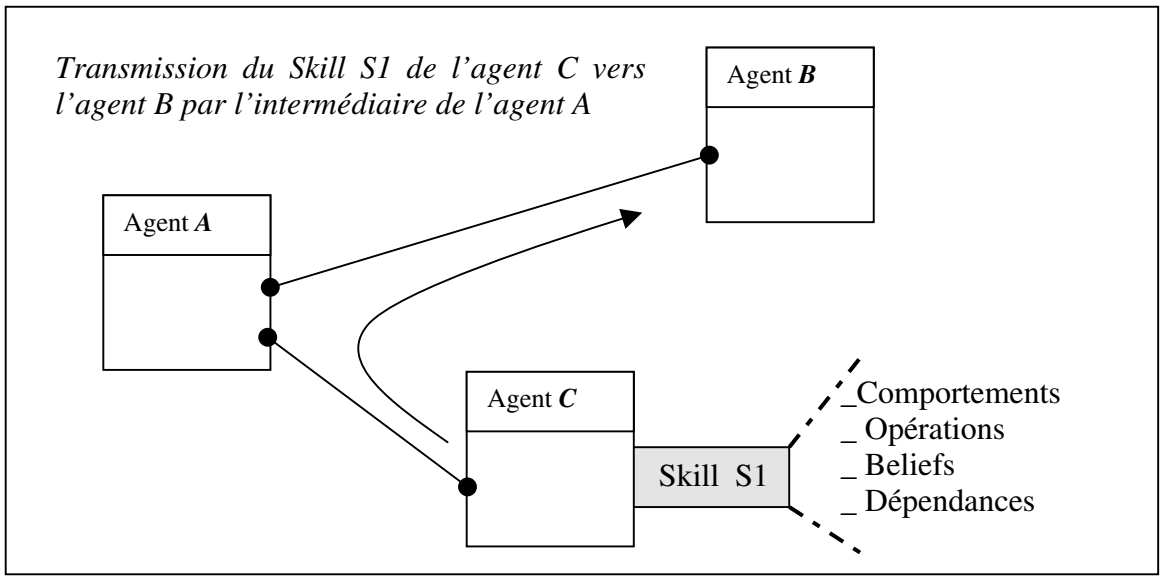

Figure 1

En dehors du transfert physique du module qui consiste pour l'agent à copier dans son environnement le module qui lui est envoyé par un autre agent situé sur un autre système (Figure 1), le transfert de compétences effectif a lieu lors du chargement en mémoire du module pendant lequel l'agent prend connaissance :

1. des opérations que ce module peut effectuer
2. des comportements (behaviours) qui y sont associés
3. des états (*belief*s) qui peuvent être générés par l'exécution de ces opérations
4. des relations de dépendance des skills

Cette faculté d'apprentissage statique, par opposition à l'apprentissage dynamique étudié et développé par les spécialistes de l'Intelligence Artificielle (IA), a pour principaux avantages :

1. de permettre le développement des compétences indépendamment de l'agent
2. de pouvoir affecter de nouveaux rôles aux agents en cours de fonctionnement
3. d'optimiser les performances d'exécution grâce aux opérations codées que contiennent les skills.

Comme nous pouvons le déduire de cette introduction sur les skills, les agents DIANA présentent des comportements de base :

- Sociaux tout d'abord, qui leur permettent de s'identifier, de savoir qui détient tel ou tel skill, de s'échanger des informations sur leurs états et plus généralement leurs connaissances.
- Délibératifs ensuite, leur permettant d'utiliser leurs connaissances des skills pour décider des actions à entreprendre en fonction d'événements affectant leurs propres *belief*s, ou les *beliefs* d'autres agents auxquels ils sont abonnés

Dans les sections suivantes nous précisons les notions attachées aux skills qui vont nous permettre de traiter notre étude de cas.

3.1 Le belief

C'est l'élément d'information de base qu'utilise l'agent en interne et dans les communications inter-agents. Dans l'architecture DIANA le *belief* est une information non différenciée, qui peut représenter un état (*fact*) actuel ou à atteindre. Cette information est produite par l'exécution d'une opération interne à l'agent, ou par un autre agent.
Différents composants internes de l'agent interviennent en utilisant des mécanismes de pattern-matching pour rechercher les entités intéressées par chaque *belief*, qui sont alors notifiées de l'existence de cette information. En plus des services de base (création, mise à jour, suppression, lecture) nécessaires à la manipulation des *belief*s, l'agent fournit un service d'abonnement sur les *belief*s et notifie les abonnés, par exemple un groupe d'agents, lorsqu'un événement survient sur le *belief* correspondant à leur abonnement.

3.2 L'opération

Lors du chargement d'un skill, l'agent prend connaissance des informations relatives aux opérations :

1. leur nom et leurs arguments
2. leurs pré-conditions
3. les *belief*s qui leur sont nécessaires en cours d'exécution
4. les opérations dépendantes qu'elles pourraient déclencher
5. les *belief*s produits en conséquence de leur exécution.

L'agent utilise ces informations pour planifier lorsque c'est nécessaire une suite d'opérations à exécuter pour atteindre un but. La description et l'utilisation des opérations par l'agent seront revues aux chapitre 4.2.

3.3 Le comportement

Par opposition aux opérations dont l'algorithme est codé et non redéfinissable par l'agent, le comportement est une connaissance interprétée et modifiable par l'agent , qui décrit un rôle ou un élément de rôle en relation avec le skill qui le contient.
Par exemple pour le skill relation clientèle:

```
If  ! ( any agent in  gbelief(« AgentList * »)
Such that gbelief(« sendTo $agent MasterPVC state actif » != «  »
))
 then  loadSkill MasterPVC
```

exprime le comportement correspondant à la décision de prise en charge du rôle de responsable de création de PVCs si aucun agent connu ne l'assume.
Il est à noter que la version de l'agent utilisée pour cette étude de cas ne prenait pas en compte les éléments de comportement, aussi ceux-ci ont été codés sous forme d'opérations de haut niveau.

4 Conception de l'étude de cas

Dans cette section nous décrivons la démarche que nous avons suivie pour développer l'application de création de PVC. L'architecture interne des agents DIANA étant en constante évolution, aucune méthodologie de développement n'a été spécifiquement étudiée.
Cependant le concept de skills utilisé par les agents DIANA incite logiquement à recourir à une modélisation de type organisationnel pour décrire et concevoir les applications..

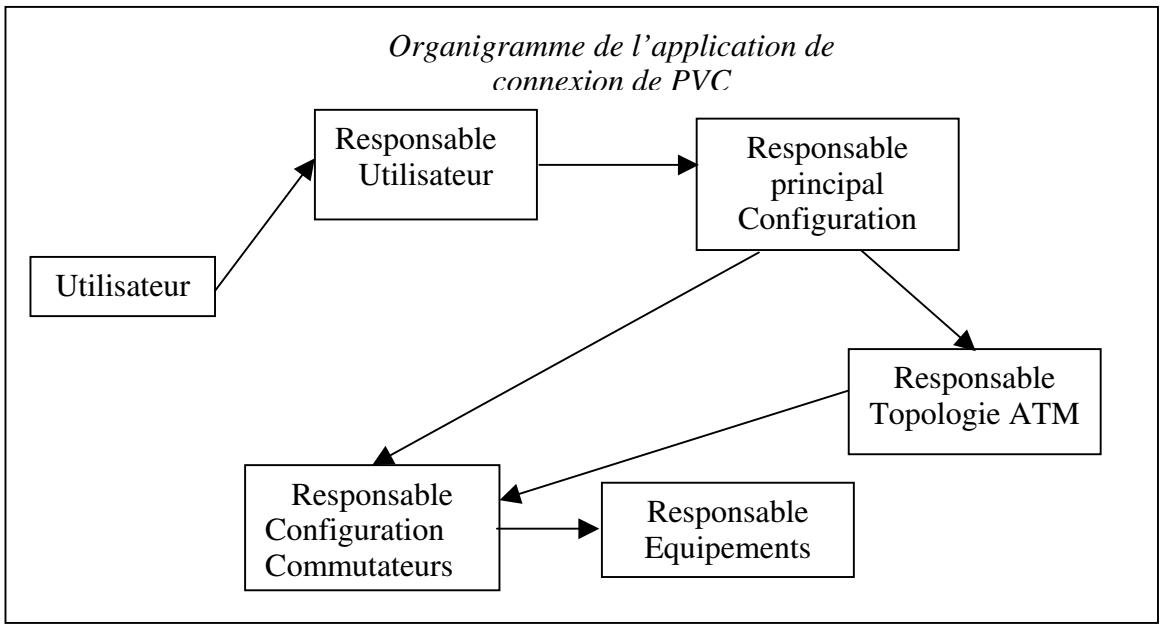

Figure 2 : Organisation des rôles

Cette modélisation a en effet pour avantage de permettre la description des fonctions (titres), de leurs rôles, des services, et des relations inter-fonctions, et se rapproche par cela de la structure de fonctionnement que l'on peut donner à un groupe d'agents
Le concepteur peut alors voir l'application comme un ensemble d'unités ou de départements, ayant chacun un ou des rôles et fournissant des services aux autres, avec pour objectif final de satisfaire l' utilisateur.

Comme le montre l'organigramme utilisé pour la réalisation de cette application (

Figure 2), plusieurs simplifications ont été choisies, comme par exemple l'absence de fournisseur (opérateur ou autre fournisseur de service), et il n'a été pris en compte qu'une seule unité englobant l'ensemble des services.
Une fois l'organisation précisée la deuxième étape consiste à décrire le rôle des différentes fonctions.

4.1 Les rôles

Un rôle est associé à des connaissances sur les opérations et les informations à utiliser ou utilisables, et la manière de les utiliser. Le processus d'identification des rôles dans la conception d'un système à base d'agents permet de déduire les opérations à coder, les informations à échanger entre les opérations d'un même rôle, et entre les opérations de rôles différents.

Rôles de l'utilisateur (SkUserPVC):
1) Demander l'établissement d'une connexion, la suppression d'une connexion auprès du responsable utilisateur du fournisseur de service
2) Présenter une interface utilisateur simple pour sélectionner la station à connecter et l'application devant utiliser le PVC.

Rôles du responsable utilisateur[2] :
Récupérer les demandes utilisateurs, les transmettre au responsable principal de configuration, et transmettre aux utilisateurs le résultat de leurs requêtes.

Rôles du responsable de la topologie ATM (SkToplogyATM):
1) Fournir les chemins possibles entre une source et une destination à tout demandeur

Rôles du responsable principal de configuration (SkMasterPVC):
1) Demander une connexion : vérifier qu'il existe un chemin pour établir une connexion entre le site du demandeur et le site du destinataire.
2) S'il existe un chemin, demander une réservation de connexion sur tous les éléments du chemin (agents en charge). Demander la mise en place de la configuration si toutes les réservations on été acceptées, et sauvegarder les références de la connexion. Reporter au responsable utilisateur les résultats de la demande.
3) Supprimer une connexion : vérifier que la connexion est référencée, et transmettre à tous les responsables sur le chemin les demandes d'annulation. Supprimer la référence de la connexion.

[2] Ce rôle ayant peu d'intérêt fonctionnel à été intégré à celui du skill MasterPVC

Rôles du responsable de configuration de commutateur (SkSlavePVC):

1) Réserver une connexion : vérifier qu'il existe un VP remplissant les conditions de l'UPC, ou qu'un tel VP peut être créé. Renvoyer le résultat au responsable de bout en bout.
2) Mettre en place la connexion : envoyer aux switchs voisins sur le chemin, une proposition d'une liste de couples VPI/VCI, et à la réception des listes voisines sélectionner la plus basse commune. Demander au responsable local de configuration la création des VPI/VCI retenus
3) Transmettre la demande d'annulation au responsable de la configuration locale.

Rôles du responsable d'équipement (SkForeATM,...):

1) Réserver une connexion : renseigner le responsable sur la configuration du switch
2) Créer une connexion : configurer le switch en fonction des informations qui ont été données.
3) Annuler une connexion : supprimer sur le switch les références de la connexion à annuler.

Pour être traduit en skills, chaque rôle doit pouvoir se décomposer complètement sous la forme :

1. Comportements
2. Opérations
3. Beliefs

C'est pourquoi, à partir de ces descriptions initiales qui ont servi pour spécifier les skills et les opérations devant être codés, un certain nombre de cycles « rôles -> skills -> rôles » on été effectués pour préciser les informations devant être traitées par les opérations, et pour intégrer le comportement sous forme d'opérations.

4.2 Les opérations

Les opérations correspondent aux capacités (*capabilities*) de traitement d'un skill, et sont comme nous venons de le voir déduites de la spécification des rôles. Pour des raisons pratiques, nous ne donnons ci-dessous (Figure 3) que la spécification de l'opération createPVC associée au skill SkMasterPVC.

Durant la phase de chargement, l'agent range dans ses différentes mémoires les éléments des opérations. Tout d'abord le nom et les paramètres ou attributs de l'opération qui seront utilisés pour comparer les beliefs qu'il doit traiter, et lui permettre ainsi d'identifier les opérations à lancer.

Puis les beliefs prérequis qu'il devra chercher à créer si l'opération doit être exécutée. Dans le cas de la création de PVC, le seul belief prérequis est le nom de l'agent et du host qui servira au Skill MasterPVC à créer un identifiant unique à chaque PVC .

Puis sont donnés les beliefs créés par l'exécution de l'opération. Ils sont utilisés par l'agent lorsqu'il doit verifier la possibilité d'atteindre un but corrspondant à la création de beliefs. Il essaie alors de chaîner les opérations en fonction des valeurs des attributs et en cas de réussite, active les opérations correspondantes. Les goals créés ne sont

actuellement pas utilisés par l'agent, et donnés uniquement à titre indicatif. Par contre lorsque l'agent activera l'opération, il abonnera automatiquement le skill sur *les used beliefs* qui sont les dernières informations de la spécification d'une opération. Grâce à cela, le skill sera notifié des informations qui l'intéressent pendant toute la durée de l'exécution de l'opération.

```
 ((SkMasterPVC createPVC :source * :dest * :upcNr * :userName $U :reqId $R)
//prerequisite beliefs
 ((me :name * :host *))
 //created beliefs
((SkMasterPVC :result * :pvcId * :reqId $R)
(SkMasterPVC :pvcId * :source * :dest * :userName $U :upcNr * :status *)
(SkMasterPVC :result * :reqId $R :reason *))
//created goals
((SkTopology findSP :source * :dest *)
(SkSlavePVC reservePVC :iport * oport * :upc * :resId *)
(SkSlavePVC createPVC :resId * :pvcId *)
(SkSlavePVC cancelRes :resId *)
(sendTo * SkSlavePVC reservePVC :iport * oport * :upc * :resId *)
(sendTo * SkSlavePVC createPVC :resId * :pvcId *)
(sendTo * SkSlavePVC cancelRes :resId *))
//used beliefs
((SkTopology :nodeList *)
(SkSlavePVC :desc 'reservationDone' :resId *)
(SkSlavePVC :desc 'reservationError' :resId * :reason *)
(SkSlavePVC :desc 'creationDone' :resId *)
(SkSlavePVC :desc 'creationError' :resId * :reason *)
(recvFrom * SkSlavePVC :desc 'reservationDone' :resId *)
(recvFrom * SkSlavePVC :desc 'reservationError' :resId * :reason *)
(recvFrom * SkSlavePVC :desc 'creationDone' :resId *)
(recvFrom * SkSlavePVC :desc 'creationError' :resId * :reason *))
```

Figure 3 : Spécification de l'opération createPVC

5 Déploiement de l'application

L'objectif de cette section est de montrer la facilité avec laquelle les agents peuvent collaborer et s'échanger les modules de compétences afin de mettre en place ce nouveau service d'établissement de PVCs. Nous extrapolons ci-dessous les résultats obtenus à partir du déploiement réel de notre étude de cas sur deux commutateurs FORE ATM, à un ensemble simulé de cinq commutateurs.

N'ayant pas encore la possibilité d'utiliser des commutateurs intégrant des machines virtuelles Java, nous avons lancé nos agents sur des systèmes connectés aux différents

commutateurs en affectant un agent par commutateur. La partie instrumentation de nos agents n'ayant pas encore été développée au moment de l'expérimentation, une couche de protocole SNMP a été intégrée au skill SkForeATM pour permettre aux agents de dialoguer avec les équipements.

5.1 Conditions de départ

Nous nous plaçons dans le contexte d'un système à base d'agents en fonctionnement à qui on va rajouter un nouveau service de création de PVC.

1. L'utilisateur a à sa disposition l'agent représenté par (A) sur la Figure 4
2. Les agents sont tous actifs
3. Les agents S1 à S5 ne connaissent que les agents directement voisins excepté S3
4. Les skills développés sont tous placés dans l'environnement de l'agent S3, qui est le seul à avoir la liste complète des autres agents.
5. Tous les switchs ATM sont des FORE

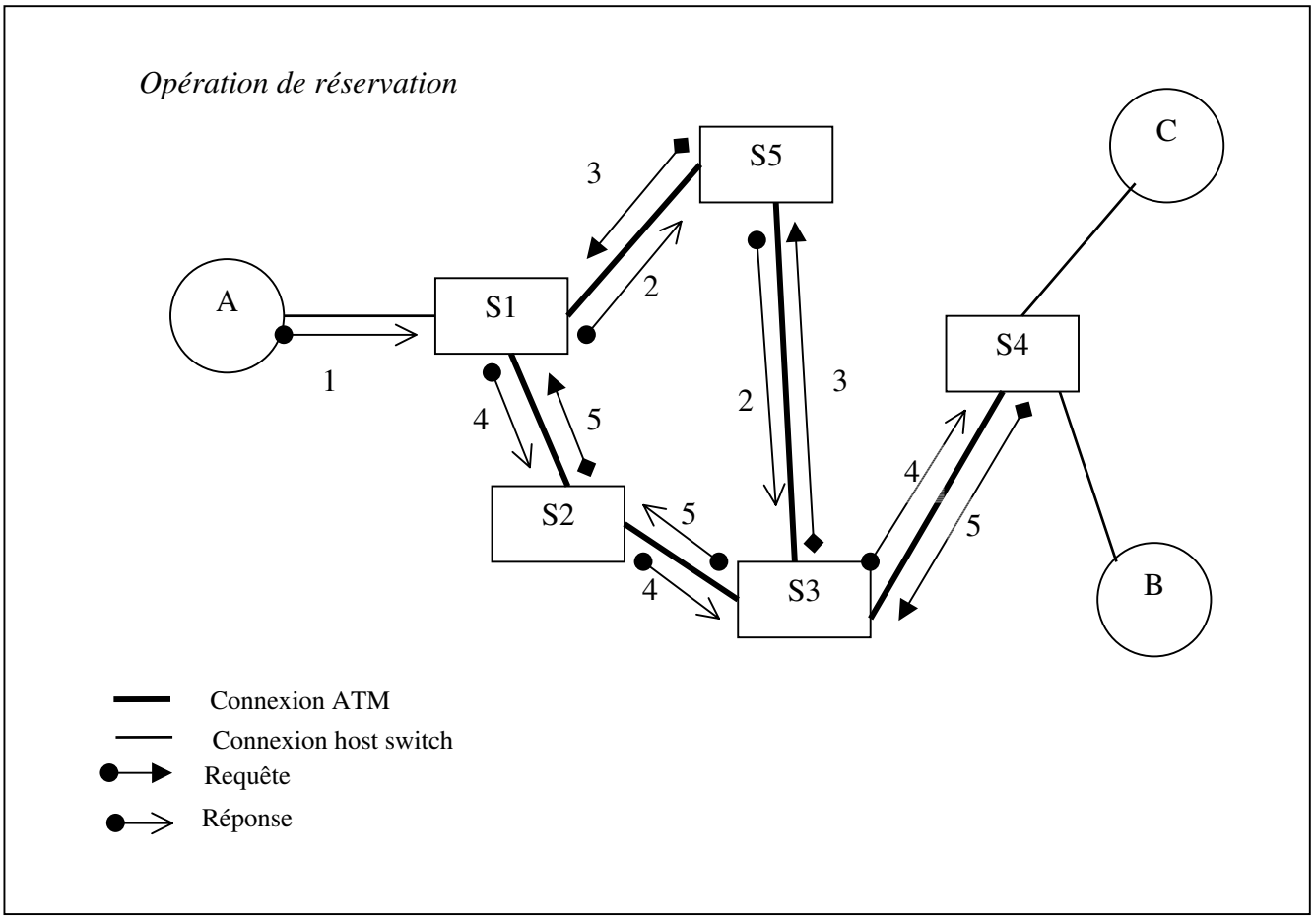

Figure 4 : Déroulement d'une opération de réservation de PVC

5.2 Chargement des Skills et transfert de compétences

A partir de l'interface utilisateur de l'agent S3, l'administrateur affecte un switch par agent en créant les beliefs suivants:

"sendTo S1 me :agent S1 :switch S1 ", ..., "sendTo S5 me :agent S5 :switch S5"

Puis de la même manière, il demande à chaque agent switch de prendre le rôle de SlavePVC :

"sendTo S1 loadSkill SkSlavePVC", ..., "sendTo S5 loadSkill SkSlavePVC"

En envoyant ces requêtes, l'administrateur laisse aux agents le soin de rechercher qui a la compétence SkSlavePVC, et de récupérer celle-ci.

A la fin de cette opération, les agents responsables des switchs ont tous la compétence SkSlavePVC, mais également SkForeATM qu'ils ont recherchée et chargée après avoir pris en compte qu'ils avaient à gérer des switchs Fore.

Conclusion

Nous pouvons observer des processus de :

- Collaboration: S1,S2,S3,S4,S5 travaillent ensemble pour mettre à niveau leurs compétences
- Délégation: S1 demande à S2 et S5 de traiter la recherche des compétences
- Transfert de compétences: mise à disposition des compétences de S3

5.3 Etablissement d'un PVC

A partir de l'interface standard de son agent (A), l'utilisateur doit demander le chargement de la compétence SkCustomerPVCInterface, qui est l'interface développée pour permettre de choisir le type d'application nécessitant le PVC et le site distant à connecter.

L'agent A demande alors au seul agent qu'il connait, c'est à dire S1, de rechercher pour lui cette compétence, et celle-ci est chargée après avoir été trouvée dans l'environnement de S3.

Une fois ces compétences chargées par l'agent, l'utilisateur peut choisir sa destination (le site B dans notre exemple (Figure 4)) et une application (comme de la Visio conférence).

Lorsque la requête est validée, l'agent client A vérifie en premier lieu si l'agent S1 a bien la compétence nécessaire pour traiter la demande, et dans la négative, il lui demande de récupérer cette compétence. Une fois la demande de création de PVC reçue (1), l'agent S1 demande au skill SkTopologie qu'il a préalablement chargé les chemins possibles pour ce PVC .

La Figure 4 nous indique que deux chemins sont possibles :

1) S1,S5,S3,S4
2) S1,S2,S3,S4

L'agent S1 envoie tout d'abord une demande de réservation (2) à S5, S3 et S4 mais S3 répond par la négative, S1 annule donc la réservation et fait une demande (4) sur le deuxième chemin, qui aboutit (5).

L'agent peut alors confirmer la création du PVC et renvoyer une réponse positive à l'agent A.

Conclusion
Tout comme précédemment, les agents ont montré une faculté de collaboration, travaillant ensemble pour permettre l'établissement de ce PVC, et de délegation, l'agent S1 laissant le soin de la configuration des commutateurs aux agents responsables.

6 Conclusions

Grâce aux travaux réalisés, la configuration dynamique de PVCs ne présente plus de problèmes avec les agents DIANA, et un utilisateur aussi bien qu'un administrateur peut maintenant en quelques secondes établir un PVC sur un réseau ATM. Nous avons pu montrer que les possibilités de délégation statique et dynamique [10], la capacité de transférer des compétences, donc de répartir dynamiquement le traitement des informations, sont maintenant accessibles grâce aux technologies agents intelligents. D'autres études de cas menées en parallèle nous ont permis de mettre en évidence la robustesse potentielle des applications à base d'agents intelligents[11].

Malgré tout, l'absence de comparaison avec une solution purement centralisée, et l'obligation d'utiliser des agents externes aux systèmes à gérer ne permettent pas de tirer des conclusions sur le gain en performance, ni sur l'économie de ressources supposée être apportée par les agents.

Grâce à cette étude de cas nous avons pu dégager quelques points importants pour la suite de notre recherche.

6.1 Les points forts

L'architecture DIANA
Cette étude de cas nous a permis de valider les principes de l'architecture des agents DIANA que nous avons commencé à développer. Le concept de modules de compétences nous a permis de concevoir et de programmer cette application distribuée plus facilement que nous n'aurions pu le faire avec un langage déclaratif.

Le développement d'applications
L'approche organisationnelle a été testée avec succès et semble bien adaptée au développement particulier que requièrent les agents DIANA. Une fois que le fonctionnement des agents a été intégré, le développement d'un module de compétences est assez rapide pour un développeur averti, malgré l'absence d'outils spécifiques. De plus, la flexibilité de l'application finale permet aisément de modifier ou d'ajouter de nouveaux modules de compétences (ex : skills pour gérer les switchs d'autres constructeurs).
L'utilisation des agents
Pourvus d'une interface de communication relativement simple, les agents DIANA peuvent être préconfigurés avant lancement, ou bien interactivement comme cela a été

démontré dans le chaiptre 5, grâce à une interface utilisateur qu'il peuvent charger ou décharger à la demande. Cette interface permet de communiquer avec les agents et de les contrôler facilement.

6.2 Les travaux en cours et futurs

Lors de la conception nous avons constaté qu'il était important d'aller plus loin dans notre recherche sur la description et l'utilisation des comportements dynamiques des agents. C'est un problème bien connu des spécialistes de systèmes multi-agents, mais qu'il est impératif de traiter pour pouvoir garantir un comportement cohérent des agents. Nous avons aussi été confronté au problème de l'évaluation et de la coordination des opérations inter-agents, qui empêche un agent de savoir à l'avance si un but peut être atteint lorsque certaines opérations sont effectuées par ses collègues.
D'un autre côté, nous travaillons pour intégrer dans l'architecture des composants permettant d'optimiser le monitoring des éléments de réseaux (instrumentation) et de programmer les réflexes que devrait avoir l'agent pour gérer les fautes.

7 Remerciements

Nous tenons à remercier Ernesto Ruggiano de SwissCom et Karima Boudaoud pour leur participation à cette étude de cas ainsi que Jacques Labetoulle pour ses précieux conseils.

8 Références

[1] K. Iseda, T. Chujo. 'Corba-Based Network Operation System Architecture'. NOMS'98, New Orleans, Février 1998.
[2] M.-A. Mountzia. 'Intelligent Agents in Integrated Network and Systems Management'. EUNICE'96 Summer School, Lausanne, sep, 1996
[3] S. Albayrak, F. J. Garijo, M. Plu . 'Préface'. Intelligent Agents for Telecommunication Applications, Paris, Juillet 1998.
[4] B. Pagurek, Y. Li, A. Bieszczad, G. Susilo. 'Network configuration management in heterogeneous ATM environments'. IATA'98, Paris, Juillet 1998.
[5] T. Magedanz, K. Rothermel. 'Intelligent agents : An emerging technology for next generation telecommunication ?'. INFOCOM'96, USA Mars 1996
[6] M. Baldi, G. P. Picco. 'Evaluating the tradeoffs of mobile code design paradigms in network management applications'. ICSE'97, Koyto, 1997
[7] Y. Iraqi, A. Ghlamallah, A. Mehaoua.'Une architecture multi-agents pour le contrôle de la qualité deu servive vidéo dans les réseaux ATM'. GRES'97, Rennes, Septembre 1997
[8] M.Cheikhrouhou, P.Conti, J.Labetoulle. 'Intelligent Agents in Network Management, A State of the Art'. In 'Networking and Information Systems' 1998, Vol. 1,N 1
[9] M.Wooldridge, N Jennings. "Pitfalls of Agent-Oriented Development".Agents-98, Minneapolis, 1998
[10] M.-A. Mountzia. 'Delegation of functionality : Aspects and Requirements on Management Architectures'. DSOM'96, L'Aquila, Italy, Octobre 1996.
[11] K. Marcus, P. Conti, M. Cheikhrouhou. 'Intelligent Agents for Network Management : Fault Detection Experiment'. Submitted to IM'99.