Le projet RESEAU FUTE : systèmes multi-agents et gestion de réseaux

Babak Esfandiari (LIRMM)
Joël Quinqueton (INRIA-LIRMM)
161 rue Ada F-34392 Montpellier Cedex 5
Jean-Luc Koning (LIFIA-ISAR)
Yves Demazeau (LIFIA-CNRS)
46 avenue Félix Viallet F-38031 Grenoble Cedex

Résumé : Si l'utilisation de l'Intelligence Artificielle Distribuée dans le domaine de la gestion de réseaux a souvent démontré son efficacité, l'étude des concepts et normes de gestion de réseaux s'est également révélée très intéressante dans la compréhension des problèmes de communication entre entités autonomes hétérogènes.

Celles-ci ne représentant pas nécessairement les données de la même manière, l'ISO et le CCITT, les deux organismes principaux de normalisation en matières de réseaux et de télécommunication ont défini un modèle orienté objet de description de données (GDMO), ainsi qu'un langage de requête (CMISE) s'appuyant sur ce modèle afin de permettre à une plateforme de gestion de réseau de consulter les informations contenues dans les noeuds du réseau.

Nous nous efforçons dans cet article de montrer que la généricité du prototype la rend adéquate pour la conception de systèmes multi-agents.

Mots-clés : systèmes multi-agents, gestion de réseaux, protocoles d'interaction, GDMO, CMISE.

Remerciements : Ce travail a été réalisé dans le cadre du contrat CNET-LIRMM *Réseau Futé* et du groupe de travail MARCIA du PRC-GDR *Intelligence Artificielle*.

1 Introduction

Le paradigme multi-agents s'appuie sur le concept de distribution de l'intelligence dans des agents formant une société dans laquelle chacun a une certaine autonomie. Un agent est une entité physique ou abstraite qui est capable d'agir sur elle-meme et sur son environnement, qui ne dispose que d'une représentation partielle de cet environnement, qui peut communiquer avec d'autres agents, qui poursuit un objectif individuel, et dont le comportement est la conséquence de ses observations, de ses connaissances, de ses compétences et des interactions qu'il peut avoir avec les autres agents de l'environnement [9].

Un agent cognitif est un agent capable de raisonner à partir de représentations de son environnement, des autres agents, de ses propres connaissances et de celles des autres. Un agent est intentionnel quand il possède des buts explicites et construit explicitement des plans lui permettant d'accomplir ces buts. Un agent est rationnel quand il peut justifier ses décisions (expliquer les résultats de son raisonnement en fonction de ses méthodes). Enfin un agent est adaptatif s'il possède la capacité (a) d'exercer un controle sur ses activités de perception, de communication, d'action et de raisonnement en réponse aux contraintes de l'environnement, et (b) d'accroitre ses connaissances par interaction avec les autres agents.

L'objectif visé par le groupe de travail MARCIA, lors de sa constitution initiale, est de construire un modèle de la structure interne d'un tel agent puis un langage de spécification d'agents construits sur ce modèle. Ce langage de spécification devra caractériser le comportement de l'agent, par exemple les différentes formes de raisonnement qu'il peut pratiquer, l'état de ses connaissances, les contraintes sur son fonctionnement qui proviennent de l'environnement, les modes de communication avec d'autres agents, ces autres agents pouvant etre un utilisateur humain. Pour une part de ces spécifications, on envisagera la définition de règles ou de méthodes permettant de les opérationnaliser, c'est-à-dire de conférer à l'agent les propriétés spécifiées.

Notons que si une partie de ce langage de spécification (et des méthodes associées) est accessible et manipulable par l'agent, il devient alors possible que l'agent s'auto-spécifie et donc modifie son comportement tout en pouvant le caractériser. Cela pourra constituer un apport important en ce qui concerne l'activité de controle d'un agent. Par effet de bord, on s'attend à ce que les efforts de clarification sous-jacents aux travaux conduits pour atteindre cet objectif permettent de dégager des thèmes à explorer plus avant.

Dans cet article nous nous consacrerons plus à l'aspect communications entre agents en montrant que les protocoles définis pour la gestion des réseaux répondent en partie à ces objectifs. L'énorme quantité d'informations à gérer, le caractère décentralisé du problème, la nécessité de tolérance aux pannes et le besoin d'un point de vue multi-expert dû à l'hétérogénéité des réseaux rendent le point de vue multi-agent intéressant dans l'introduction d'intelligence artificielle dans la gestion de réseaux. Beaucoup de projets ont été menés dans ce domaine [10].

Le projet RESEAU FUTE est centré sur l'utilisation de systèmes multi-agents pour la supervision de réseau de télécommunications. Un des objectifs du projet est d'évaluer dans ce type d'architecture l'intérêt de l'apprentissage automatique pour la détection de situations d'engorgement ou de panne, et ceci en introduisant le protocole multi-agent d'apprentissage MOSCA [16, 17, 15].

A cette fin, une étude des concepts et normes de gestion de réseau s'est révélée nécessaire. Or ces normes se sont penchées de façon approfondie sur les problèmes de modélisation d'informations dans un contexte hétérogène ainsi que la définition de protocoles de communication de haut niveau entre entités du réseau. Nous pensons que les résultats obtenus peuvent être extrêmement utiles pour la conception de systèmes multi-agents génériques en présentant un point de vue original et une bonne source d'inspiration.

Cet article débutera donc par une présentation succincte de quelques concepts et normes de gestion de réseaux. Nous verrons ensuite comment la prise en compte de ces normes pour la conception d'une simulation de plateforme de gestion de réseau nous a naturellement conduit à un système multi-agent générique. Nous étudierons enfin la possibilité d'enrichir le modèle en définissant des protocoles d'interaction afin de faciliter la coopération entre nos agents.

2 Présentation rapide de quelques normes de gestion de réseau

L'ISO a défini un formalisme orienté objet de description : G.D.M.O. (Guideline for the Definition of Managed Objects). GDMO permet de décrire les informations de gestion en terme d'objets gérés. On peut définir un objet géré comme une abstraction sur une ou plusieurs ressources physiques ou logiques du système géré.

La communication de gestion entre un gestionnaire et un agent s'effectue grâce aux services CMISE (Common Management Information Service Element). Ces éléments de service offrent au processus gestionnaire de réaliser sur un ou plusieurs objets gérés des opérations de gestion (création et destruction d'objets, consultation et modification d'attributs et déclenchement d'actions) et de rapporter des notifications.

2.1 La norme GDMO (Guideline for the Definition of Managed Objects)

GDMO est un formalisme orienté objet de description des ressources d'un réseau. Chaque ressource est considérée comme étant un *objet géré*. Les objets de même nature (i.e. ayant les mêmes propriétés) sont des instances d'une même classe d'objets. Ces classes sont regroupées dans un arbre d'héritage.

Une classe d'objets est une *template*, c'est-à-dire une structure prédéfinie, représentée principalement par ses :

- attributs

- actions : opérations effectuées sur les attributs
- notifications : messages envoyés par un objet pour signaler un événement
- une description informelle de son comportement

Afin de rendre possible des requêtes sur un ensemble d'objets, les noms des objets sont organisés hiérarchiquement dans un arbre dit de nommage. Ainsi un objet est nommé de façon unique. L'arbre est défini par la relation de contenance, qui exprime le fait qu'un objet géré peut contenir d'autres objets, de la même classe ou d'une classe differente. Elle permet de modéliser des relations physiques (assemblage, composant...) ou des relations organisationnelles (répertoire, fichier, enregistrement, champ...) entre des ressources représentées par les objets.

L'ensemble conceptuel des objets d'un système géré constitue, par définition, la Base d'Informations de Gestion (en Anglais la MIB : *Management Information Basis*).

2.2 La norme CMISE (Common Management Information Service Element)

CMISE permet le dialogue entre deux entités du réseau en fournissant un ensemble de primitives qui sont utilisées pour créer et envoyer des requêtes ou des alarmes. Les principales primitives sont :

- `m-get` : consulte la valeur de l'attribut d'un objet
- `m-set` : modifie la valeur de l'attribut d'un objet
- `m-create` : crée un objet avec ses valeurs par défaut
- `m-delete` : supprime un objet
- `m-action` : déclenche une action sur l'objet designé
- `m-event-report` : émet un rapport d'événement (création d'un objet, depassement d'un seuil...)

Les primitives de requête, en plus de la désignation de l'objet, peuvent avoir les attributs suivants, qui permettent la sélection d'un ensemble d'objets grâce à l'arbre de nommage :

- Le champ d'action (de communication) : il détermine la profondeur du sous-arbre de nommage sur lequel s'applique la requête.
- Le filtre : il permet de sélectionner un sous-ensemble du sous-arbre précédent en définissant des critères sur les attributs.

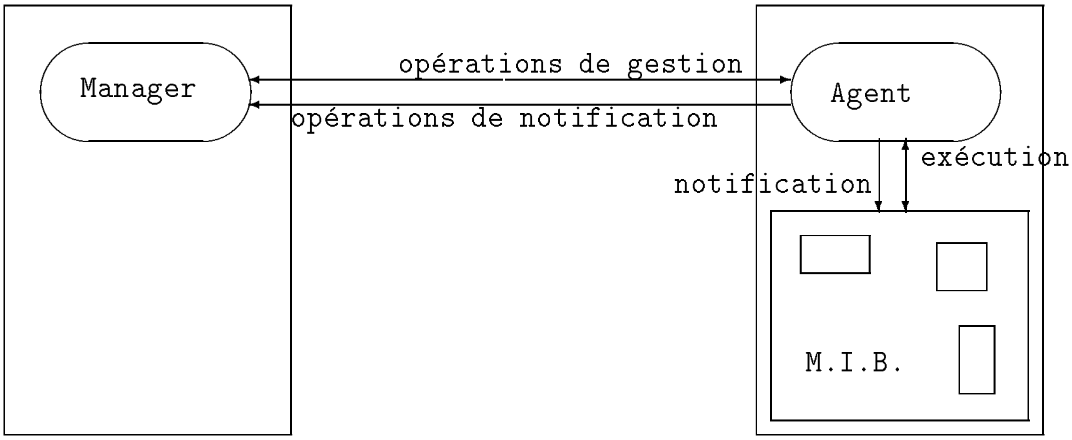

Figure 1: Le modèle Manager-Agent

Pour le type de l'interaction, nous pouvons adopter les primitives proposées par Gaspar [11], qui consiste à distinguer quatre possibilités : présentation, requete, réponse, et information, la première d'entre elles permettant à un agent d'entrer dans une société et de se présenter aux autres, ce qui garantit une certaine ouverture du système. Les trois autres possibilités sont triviales.

2.3 Le modèle Manager-Agent

En gestion de réseaux, les deux entités de base sont appelées le *manager* et l'*agent*. Elles sont représentées sur la figure 1. Le manager envoie des requêtes et reçoit les réponses et les notifications. L'agent est l'entité symétrique, qui reçoit les requêtes et envoie les réponses et les notifications.

Notons qu'une même entité peut jouer alternativement les deux rôles, ce qui rend la communication manager-manager possible. Pour plus de détail sur les normes de gestion de réseau, consulter notamment [14].

2.4 Un modèle générique ?

GDMO permet de modéliser tout type de connaissance (et non uniquement des données concernant les réseaux) et d'homogénéiser ainsi des connaissances entre agents hétérogènes. Une application peut adopter dès le depart ce formalisme, et si elle est déjà réalisée, elle peut utiliser GDMO comme interface externe, pouvant ainsi communiquer à l'aide de primitives CMISE (c'est d'ailleurs ainsi qu'elle est utilisée dans la plupart des plateformes de gestion de réseaux actuelles). De ce point de vue, CMISE est très proche de langages tels que KQML [12] ou MAI2L [20].

KQML est un langage de communication entre agents développé par le DARPA Knowledge Sharing Effort. Un agent est vu de l'extérieur comme gérant une base de données, et KQML offre des primitives (*tell, insert, reply, ask-if,...*) qui permettent de manipuler ces données au cours d'un dialogue. Un message KQML est constitué d'une primitive de communication et de mots-clés qui précisent entre autre les noms de l'expéditeur et du destinataire, le langage interne de l'agent et l'ontologie (qui représente le point de vue choisi pour décrire les données). Le Knowledge Sharing Effort propose d'ailleurs un langage interne, KIF (Knowledge Interchange Format [13]), qui permet une représentation particulière des données et des opérations logiques sur ces données.

En fait, GDMO et CMISE, grâce notamment à l'arbre de nommage et au filtre, permettent pratiquement les mêmes manipulations logiques sur les données que KQML associé à KIF. Bref, CMISE étant antérieur à KQML, il peut être considéré comme un des premiers langages de communication génériques entre agents logiciels !

3 Vers un cadre unifié pour l'échange de connaissances

Nous allons maintenant montrer en quoi CMISE fournit une base pour un cadre unifié d'échange de connaissances. Présentons d'abord les modèles classiques de communication de systèmes multi-agent, en mettant en évidence les différentes couches.

3.1 Le Langage d'Interaction

Comme cela est indiqué dans [4], si la communication est simplement décrite en termes d'émission et de réception de messages, chaque agent doit etre capable d'inférer l'intention de l'émetteur lorsqu'il énonce le message. Si les messages ne sont pas structurés, cette inférence peut etre très difficile, voire impossible. C'est pourquoi les messages doivent etre attachés à des restrictions formelles, et structurés de façon à en faciliter l'interprétation.

Cela peut etre effectué par exemple en employant des types de messages, de sorte que l'intention de l'émetteur soit immédiatement reconnue à partir du message luimeme. Cela nous amène naturellement à faire la distinction entre le *Langage de Communication* et le *Langage de Représentation des Connaissances*.

Le premier exprime principalement le message d'un point de vue strictement Systèmes Distribués, alors que le second inclut à la fois la connaissance du domaine multi-agent (*Langage Multi-Agent*, qui véhicule, tant que faire se peut, l'intention de l'émetteur), mais aussi le *Langage du Domaine d'Application* (par exemple, ici, le langage du domaine des Télécommunications).

Notre *Langage d'Interaction* [7] définit le vocabulaire commun et sa sémantique. Il est utilisé pour construire des échanges d'informations, appelées *interactions*, entre les agents. Les interactions peuvent etre des échanges d'actions, de plans, de buts, ou d'hypothèses correspondant aux types des informations que les agents manipulent. Une interaction est construite selon la syntaxe suivante:

`<interaction> ::= <communication> <multi-agent> <application>`

3.2 Le Langage de Communication

Le langage de communication, décrit par `<communication>`, consiste en différents champs utilisés par la couche de communication du système.

`<communication> ::= <from> <to> <id> <via> <mode>`

Ce sont l'émetteur `<from>` et le récepteur `<to>` du message (le contenu de ce champ pouvant etre soit un identificateur d'agent, soit *broadcast*), l'identificateur du message `<id>`, le type de medium de communication utilisé `<via>` (dans certains systèmes, plusieurs types de medium peuvent etre utilisés, par exemple l'un à très haut débit pour les importants volumes de données, et un autre pour les simples échanges de messages par mécanisme de boite aux lettres), et le `<mode>` de communication (synchrone ou asynchrone).

Cette couche du système est très largement classique dans les langages de télécommunication. Elle permet de spécifier le canal de communication.

3.3 Le Langage Multi-Agent

Il est évident, pour un niveau de compréhension donné, que le degré de complexité d'un protocole d'interaction, et donc du langage multi-agent, est inversement dépendant du degré de développement du langage du domaine d'application qui est utilisé par les agents pour interagir dans la société. Afin d'utiliser un langage du domaine d'application simple, du premier ordre par exemple, pour un domaine comme celui des Télécommunications, nous nous focalisons ici sur un langage multi-agents assez complexe, qui sera associé à des protocoles d'interaction qui peuvent etre des plus simples aux plus complexes.

Notre langage multi-agents, décrit par le champ `<multi-agent>`, regroupe toute l'information relative au domaine multi-agent. Les trois types d'information sont le type, la nature, et la force de l'interaction.

`<multi-agent> ::= <type> <nature> <force>`

Le `<type>` de l'interaction est celui proposé par Gaspar [11], que nous avons déjà décrit dans le cadre du langage CMISE.

La `<nature>` définit à quelle couche de controle de l'agent récepteur s'adresse le contenu de l'interaction, et prévoit trois possibilités : couche de décision (cf. buts), couche de commande (cf. actions), couche d'observations (cf. faits ou hypothèses).

La `<force>` définit la priorité de cette information, tout en dénotant partie de l'intention de l'émetteur. Les possibilités sont empruntées de la théorie des actes de langage, allant du *commandement*, de priorité maximale, à la simple *information*, de priorité minimale. Cette échelle peut etre adaptée, et nous avons adopté comme base l'ensemble des étiquettes illocutoires défini par Campbell et d'Inverno [5], duquel nous avons extrait un sous-ensemble de tons adapté aux types d'interactions auxquels nous nous intéressons. Ils permettent d'exprimer les intentions suivantes : recherche d'informations, information, avertissement, conseil, marchandage, persuasion, commandement, et expression.

Notons, et cela est particulièrement important, que nous pouvons garantir la réception et le traitement de tout message émis à en restreignant les possibilités du langage de communication. De fait, il suffit de mettre la `<force>` du message à sa priorité maximale pour que, eu égard aux positions relatives des agents dans la société, l'émetteur dominant le récepteur, il soit admis que l'interaction soit interprétée par le récepteur comme un ordre à satisfaire.

3.4 Les Protocoles d'Interaction

L'utilisation d'un langage multi-agents permet à chaque agent d'extraire explicitement de la sémantique du message de l'information qui serve à contrôler l'échange d'information et par là même au contrôle de la société d'agents dans son ensemble.

Le fait de découpler l'intention de l'expéditeur du message lui-même est un premier pas dans ce sens mais n'est pas suffisant. En effet, les agents doivent aussi savoir comment réagir à un message, ou bien ce qu'ils doivent attendre après avoir envoyé un message.

C'est cet ensemble de spécifications auquel il s'agit de répondre dans un cadre unique, permettant ainsi de structurer les interactions entre agents, que nous appelons le protocole d'interactions.

Chaque type de message d'un langage multi-agent est en fait associé à un protocole de base donné, qui est choisi par le récepteur du message en fonction du type de message, comme dans le travail de Burmeister [3]. Ce que nous entendons ici par "protocole d'interaction" est ce qu'on définit pour restreindre les interactions entre agents en fonction des problèmes à résoudre.

Il existe un certain nombre de protocoles classiques dans la littérature d'Intelligence Artificielle Distribuée. Citons par exemple celui des *Réseaux Contractuels* [19], ou bien le protocole dit *hiérarchique* [8].

Plus récemment, sont apparus des protocoles plus spécifiques, pour l'introduction de nouveaux agents [1] ou encore la négociation [6]. Ce dernier, basé sur des modèles provenant de l'étude des situations de lutte pour la survie dans les sociétés humaines. Ce protocole prend en considération les situations de conflits où le contrôle est distribué et où aucune solution globale exacte n'existe.

De meme, dans le protocole multi-agent d'apprentissage MOSCA [16], une situation typique d'apprentissage est modélisée (figure 2), et les différents types de messages que peuvent s'envoyer les agents correspondent aux interactions entre les différents éléments intervenant dans un processus d'apprentissage automatique

Pour [7], un protocole d'interaction est un graphe d'état prédéterminé dans lequel l'agent évolue en fonction des types de messages reçus. Ceci conditionne son comportement, en particulier les types de messages qu'il doit à son tour émettre dans une situation donnée. On peut, grâce à ce type de graphes, modéliser des protocoles connus tels que celui des Réseaux Contractuels [19].

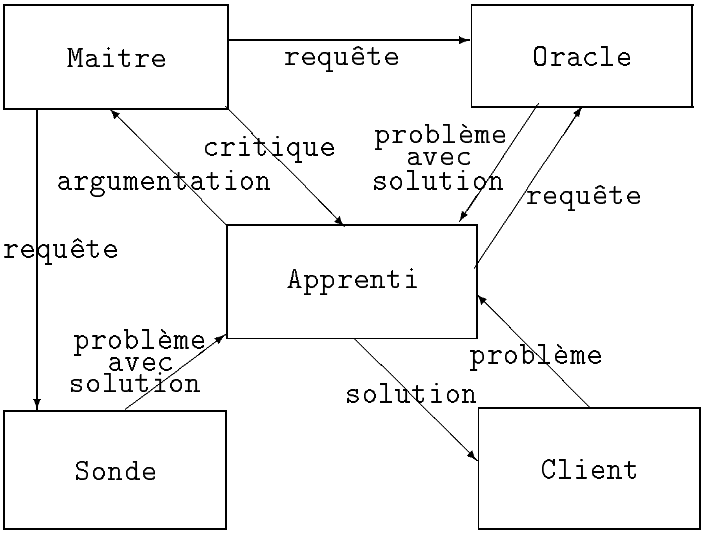

Figure 2: Le protocole le plus volatile de l'histoire de l'IA : la pentade MOSCA

Ce tour d'horizon des protocoles d'échange de connaissances nous permet de jeter les bases d'une maquette générique, qui s'efforcera de les expérimenter dans le domaine de la supervision de réseau.

La section suivante montre comment notre simulation de plateforme de gestion de réseau pourrait être utilisée comme un système multi-agent générique.

4 Vers un prototype de système multi-agents

Notre simulation d'une plateforme de supervision de réseau respectant les normes décrites précédemment semble mener à la conception d'entités autonomes capables d'interagir avec leur environnement voire de coopérer.

4.1 Architecture et implémentation de la plateforme

Un manager (notre agent) est un programme Smalltalk qui communique avec d'autres programmes Smalltalk (qui peuvent se dérouler sur des machines distantes) via le protocole TCP/IP. Smalltalk comporte l'avantage d'être orienté objet, ce qui facilite grandement la simulation de GDMO.

Les grands principes de fonctionnement de notre plateforme sont résumés dans les schémas suivants (figures 3, 4 et 5), qui détaillent les *protocoles d'interaction* de base entre le *gestionnaire de communications* et les autres agents. Les flèches représentent le flux des données.

La plateforme peut être divisée en quatre parties distinctes :

1. Le gestionnaire de communication,

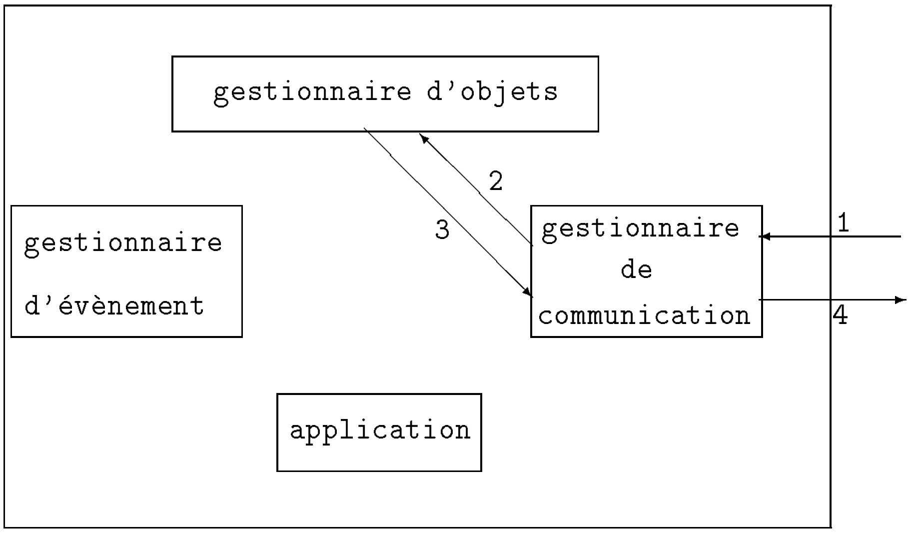

Figure 3: Traitement d'une requete

2. Le gestionnaire d'événements,

3. Le gestionnaire d'objets,

4. L'application de gestion.

Le gestionnaire de communication reçoit et envoie les messages vers les autres agents grâce au protocole TCP/IP.

Le gestionnaire d'objets met à jour les données et interprète les requêtes (figure 3).

Le gestionnaire d'événements détermine si les modifications amenées aux objets doivent donner lieu à un envoi de notification (figure 4).

L'application de gestion, qui est ici le superviseur de réseau, mais qui peut etre tout type d'application, qui décide du comportement de l'agent selon les événements reçus. Elle peut entre autre composer des requêtes et les envoyer aux autres agents par l'intermédiaire des gestionnaires de communication (figure 5).

Cette partie est la seule de la plateforme à être entièrement dépendante du problème à résoudre ; elle peut d'ailleurs ne constituer qu'une interface avec le superviseur humain.

Plus généralement, les agents communiquent à l'aide d'un langage d'interaction associé à des protocoles d'interaction qui sont utilisés pour controler le flux et le séquencement des messages.

En fait, le langage d'interaction de CMISE peut etre considéré comme un langage d'interaction assez générique, comme nous allons le voir dans la prochaine section.

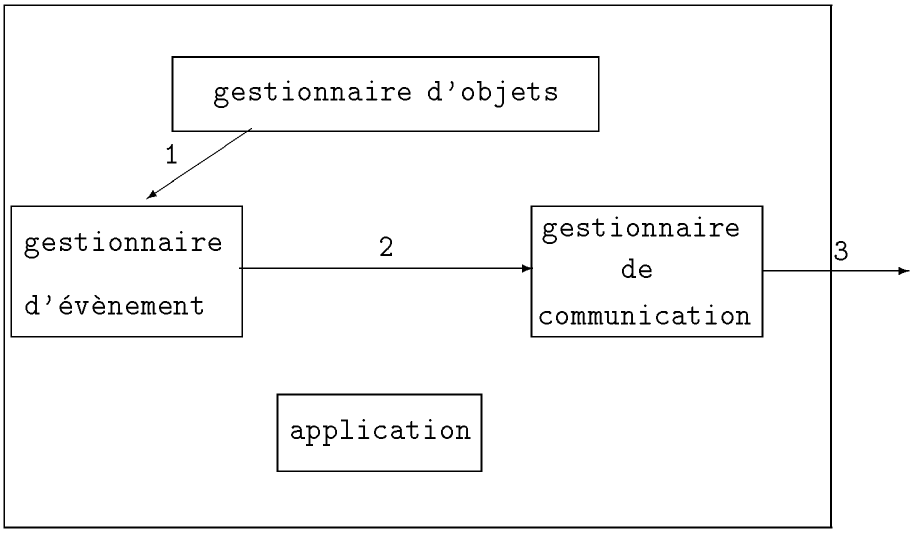

Figure 4: Envoi d'une notification

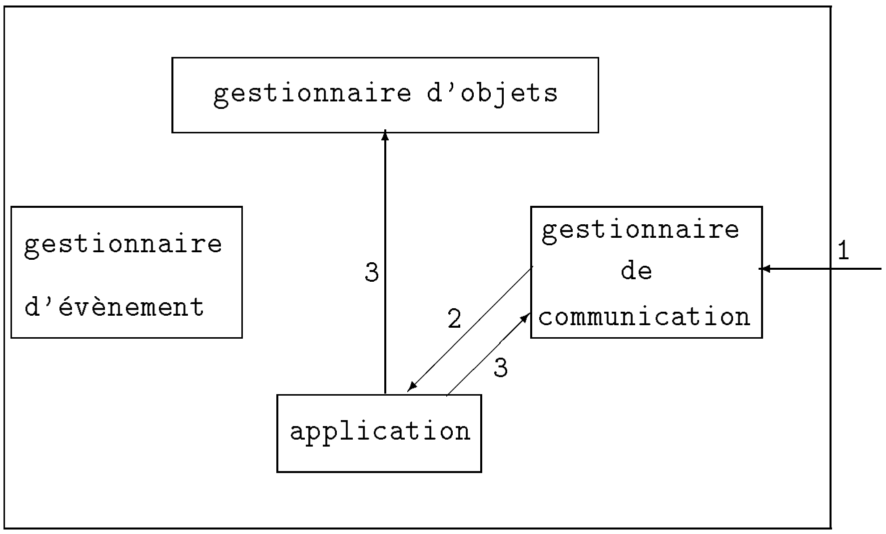

Figure 5: Traitement d'une notification

4.2 Vers un paradigme général

Le rapprochement de cette plateforme avec des systèmes tels qu'AGENT0, le langage orienté agent de Shoham [18] est assez aisé. Bien sûr, notre plateforme ne comporte pas explicitement des notions telles que l'engagement ou la condition mentale (ce n'était pas son objectif).

En revanche, outre des similarités dans les primitives de communication, on constate une ressemblance quant au fonctionnement interne de l'agent : conditions de mise à jour de données, d'émission d'événements, ou de déclenchement d'une action. Ce fonctionnement, tel qu'il est décrit dans la section précédente, nous conduit naturellement à adopter une démarche méthodologique systématique qui peut être résumée de la manière suivante :

1. Créer les classes d'objets et leurs instances initiales. Ces classes ont toutes pour ancêtre commun la classe GDMOObject qui factorise leur comportement par défaut.

2. Déterminer pour chaque classe les conditions d'émission d'un événement.

3. Programmer la partie "application", c'est-à-dire le comportement à adopter à la réception d'un événement.

Nous pensons que ce mode de fonctionnement de la plateforme, associée à la méthodologie décrite ci-dessus décrit un paradigme assez simple et général. Il est prévu d'essayer d'intégrer ce modèle dans la plateforme Actalk [2] afin d'en tester l'efficacité.

5 Conclusion

Nous avons essayé dans cette présentation de montrer la généricité des protocoles de communication définis pour les réseaux de télécommunication.

Les normes de gestion de réseau permettent donc la mise en communication d'agents hétérogènes (de conception différente) par une description universelle des objets et de leur format et par la définition d'un jeu simple et néanmoins suffisant de primitives de communication.

L'utilisation de ces normes et la reproduction du fonctionnement d'une plateforme de gestion de réseau constituent donc un moyen simple de mise en place d'un système multi-agent générique (quoique minimaliste !). La mise en place de protocoles d'interaction comme surcouche de ce système pourrait faciliter la coopération et limiter le nombre de messages échangés.

Nous n'avons bien évidemment pas présenté ici tous les aspects des normes de gestion de réseau. Beaucoup d'autres points paraissent intéressants à exploiter, dont la négociation du contexte où il s'agit d'attribuer les rôles d'agent et de manager entre deux entités du réseau.

References

[1] S. Berthet, Yves Demazeau, and Olivier Boissier. Knowing each other better. In *11th International Workshop on Distributed Artificial Intelligence*, Glenn Arbor, Février 1992. IWDAI'92.

[2] Jean Pierre Briot. From objects to actors: Study of a limited symbiosis in smalltalk-80. Research Report RXF 88-58, LITP, Univ Paris 6, 1988.

[3] B. Burmeister, A. Haddadi, and K. Sundermeyer. Generic configurable cooperation protocols for multi-agent systems. In *MAAMAW'93 proceedings*, Neuchatel, CH, Août 1993.

[4] B. Burmeister and K. Sundermeyer. Cooperative problem solving guided by intentions and perception. In Werner and Y. Demazeau, editors, *Decentralized AI III*. Elsevier Science, 1992.

[5] J. Campbell and M. D'Inverno. Knowledge interchange protocol. In *Decentralized AI I*. Elsevier Science, 1990.

[6] M. Chang and C. Woo. Sanp: a communication level protocol for negotiations. In Werner and Yves Demazeau, editors, *Decentralized AI III*. Elsevier Science, 1992.

[7] Yves Demazeau, Olivier Boissier, and Jean Luc Koning. Using interaction protocols to control vision systems. In *International Conference on Systems, Man and Cybernetics*, San Antonio, Octobre 1994. IEEE.

[8] E. Durfee and T. Montgomery. A hierarchical protocol for coordinating multiagent behavior. In *Proceedings of the AAAI'90 Conference*. AAAI, 1990.

[9] J. Ferber. *Objets et agents : une étude des structures de représentation et de communication en IA*. Etat, Univ. de Paris VI, Juin 1989.

[10] D. Gaiti. *L'utilisation des techniques de l'intelligence artificielle pour la gestion des réseaux*. Thèse d'université, Université Paris 6, 1991.

[11] G. Gaspar. Communication and belief changes in a society of agents: Towards a formal model of autonomous agent. In Y. Demazeau and J.P. Muller, editors, *Decentralized AI II*. Elsevier Science, 1991.

[12] M. R. Genesereth and S. P. Ketchpel. Software agents. *ACM*, Juillet 1994.

[13] M.L. Ginsberg. Knowledge interchange format : the kif of the death. Technical report, Stanford University, 1992.

[14] Lecerf and Chaumel. *Les normes de gestion de réseau à l'ISO*. Masson, 1992.

[15] J. Quinqueton and P. Reitz. Mosca: a multi-agent learning framework. In *Proceedings of ECML'93 poster session*, 1993.

[16] J. Quinqueton, P. Reitz, and J. Sallantin. Les schémas mentaux, un cadre conceptuel pour l'apprentissage à partir d'exemples. In J.P. Laurent, editor, *Reconnaissance des Formes et Intelligence Artificielle*, Lyon, Novembre 1991. AFCET, INRIA and ARC.

[17] Philippe Reitz. *Contribution à l'étude des environnements d'apprentissages: conceptualisation, spécificcations et prototypage.* Thèse d'université, Université Montpellier II — Sciences et Techniques du Languedoc, LIRMM, 161 rue Ada, F-34392 Montpellier Cedex 5, Février 1992.

[18] Yoav Shoham. Agent oriented programming. Research Report STAN-CS-90-1335, Stanford University, Octobre 1990.

[19] R. Smith. The contract net protocol: High level communication and control in distributed problem solver. *IEEE Transactions on Computers*, 29, 1980.

[20] D. Steiner, A. Burt, M. Kolb, and Ch. Lerin. The conceptual framework of mai2l. In Hedira and Sprumont, editors, *Proceedings of the 5th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Neuchatel, Août 1993.