Master
Web Intelligence

Systèmes Multi-agents

BROKERING

Laurent Vercouter
Laurent.Vercouter@emse.fr
SMA/ENS Mines Saint-Etienne
8 Octobre 2003

Plan

- Le problème de connexion
- Architectures pour les connexions
- Interaction dédiées
- Description d'agent & Matchmaking

Motivations initiales

Contract net protocol [Davis & Smith, 83]

Objectif: fournir un cadre spécifiant le contrôle et les communications pour une résolution distribuée d'un problème

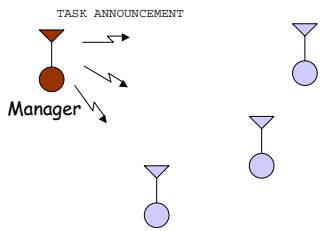
1- Appel d'offre d'un agent « manager »

2- Propositions d'agents « candidats »

3- Sélection du partenaire

Le problème de connexion

Connection problem : comment assurer la pertinence des interactions entre agents ?



Le problème de connexion

- Concevoir le **cadre de résolution** du CP
- Fournir les **outils et techniques** appropriés
- Définir les **rôles** de chaque entité
- **Etendre les capacités** des entités concernées

Cadre de résolution

- Objectif de la résolution
  - ✓ **inter-opération**
  - ✓ **échanges d'information**
  - ✓ coordination
- Contexte
  - ✓ **système ouvert** ou fermé
  - ✓ **agents coopératifs** ou compétitifs
  - ✓ **croyances sur les autres certaines** ou incertaines

Outils & techniques

- Modèle de représentation des agents
  - ✓ capacités et besoins
  - ✓ informations et ressources
  - ✓ plans et intentions
- Protocoles pour la résolution
  - ✓ présentation de représentations
  - ✓ recherche de partenaires
  - ✓ recommandation d'agents
- Services associés
  - ✓ sélection, observation, apprentissage, ...

Rôles des entités

- Composition du SMA
  - ✓ agents dédiés à certains services
- Isolation et attribution
  - ✓ croyances sur les agents du SMA
  - ✓ capacités d'échanges et d'acquisition des croyances
  - ✓ obligations et permissions d'accès et de transmission
- Définition d'une organisation
  - ✓ Rôles et liens organisationnels des agents dédiés
  - ✓ Droits d'accès des agents « normaux »

Extension de capacités

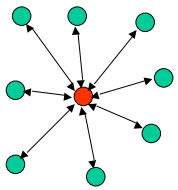
- Affiner la résolution du problème de connexion
  - ✓ évaluation des compétences
  - ✓ détection d'agents malicieux
- Ajout de dimensions au SMA
  - ✓ garantie d'anonymat
  - ✓ gestion de l'ouverture
  - ✓ gestion et traduction d'ontologies différentes

Plan

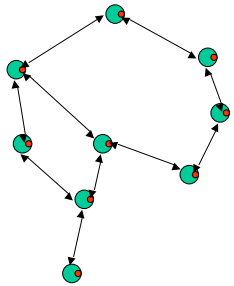
- Le problème de connexion
- Architectures pour les connexions
- Interaction dédiées
- Description d'agent & Matchmaking

Architectures pour la connexion

Centralisée



Décentralisée

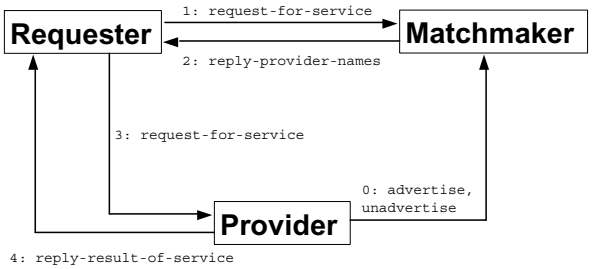


Middle-agent

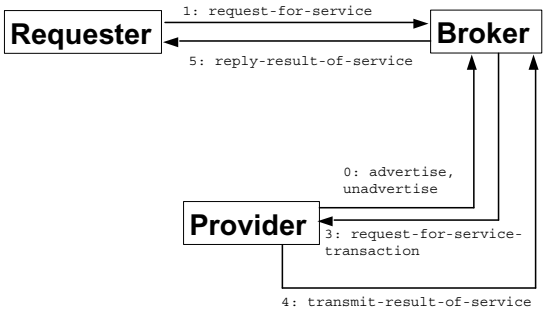
Définition : Un middle-agent est un agent fourni par l'infrastructure multi-agent dont la fonction est de localiser les agents en fonction de leurs capacités

- **Caractéristiques** :
  - ✓ centralise les représentations des agents
  - ✓ résout le problème de connexion pour le compte des agents
- deux **exemples** :
  - ✓ matchmaker
  - ✓ broker

Matchmaker



Broker



Rôles du middle-agent

Requirements known by	***Capabilities known by***		
	provider	**provider + middle-agt**	**provider + middle-agt + requester**
requester	(broadcaster)	front agent	matchmaker / yellow-pages
requester + middle-agt	anonymizer	broker	recommender
requester + middle-agt + provider	blackboard	introducer / bodyguard	arbitrator

Middle-agent

- Le middle-agent est souvent assimilé à un service global fourni par une plate-forme ou une infrastructure multi-agent.
- Le service de brokering peut être assuré par plusieurs middle-agents pour :
  - ✓ décomposer ses fonctions
  - ✓ répartir la charge entre plusieurs agents
  - ✓ fournir des services plus élaborés

plate-forme FIPA

- AMS = pages blanches
- DF = pages jaunes

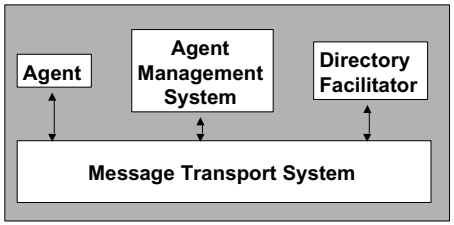
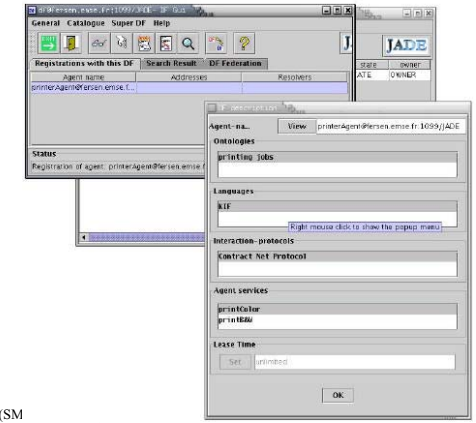


plate-forme FIPA

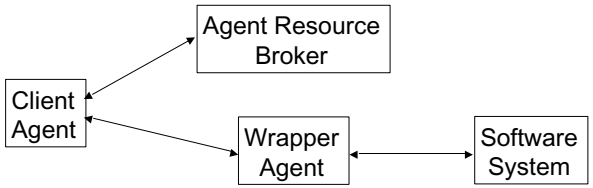
- **Agent Management System**
  - ✓ Correspondance entre un identifiant et une adresse
- **Directory Facilitator** représente les agents par:
  - ✓ son **identifiant**
  - ✓ les **services** qu'il peut effectuer
  - ✓ les **protocoles** qu'il peut suivre
  - ✓ les **ontologies** qu'il connaît
  - ✓ les **langages** qu'il comprend

plate-forme FIPA, ex: JADE



FIPA Software Brokering

L'intégration de logiciels non-agents pose aussi le problème de connexion.
La FIPA propose de le résoudre par l'utilisation de middle-agents spécifiques.

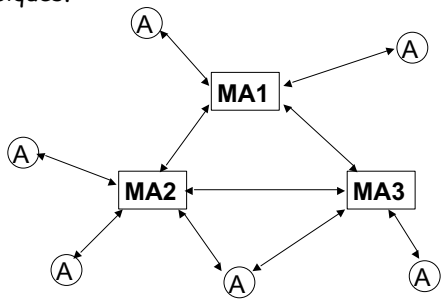


FIPA Software Brokering

- **Wrapper agent**
  - ✓ encapsule le logiciel
  - ✓ présente le logiciel comme un service de type `fipa-wrapper`
- **Agent Resource Broker**
  - ✓ maintient une représentation des services `fipa-wrapper`
  - ✓ a la capacité de connecter une liste de propriétés requises à une catégorie de services `fipa-wrapper`
  - ✓ est enregistré auprès du DF avec un service `fipa-arb`
- **Client Agent**
  - ✓ demande au DF qui peut fournir un service `fipa-arb`
  - ✓ demande au ARB qui peut fournir un service `fipa-wrapper` respectant certaines propriétés

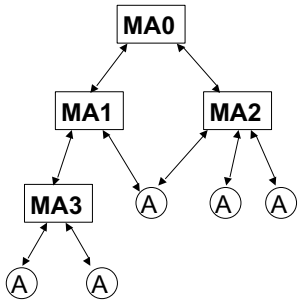
Combinaison de middle-agents

Les agents sont regroupés selon des critères fonctionnels ou physiques.



Hiérarchie de middle-agents

Chaque middle-agent est responsable d'une catégorie de services ou de descripteurs.



- Plus un middle-agent est bas, plus ces catégories sont spécifiques.
- Ex: Les catégories du MA3 sont inclues dans celles du MA1.
- Le MA0 est un middle-agent « général ».

Plan

- Le problème de connexion
- Architectures pour les connexions
- Interaction dédiées
- Description d'agent & Matchmaking

Interactions dédiées

Les connaissances du middle-agent sont nécessaires à la résolution du problème de connexion.

Mais les agents du SMA doivent y accéder en **interagissant** avec le middle-agent.

- **Performatifs** dédiés aux brokering
- **Protocoles** dédiés au brokering

Performatifs KQML

[Labrou & Finin, 97]

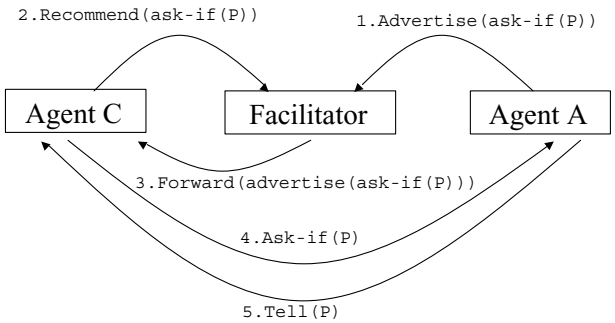
Le langage KQML spécifie un ensemble de performatif pour interagir avec un middle-agent tenant un rôle :

- de matchmaker
- de broker assurant l'anonymat
- de broker n'assurant pas l'anonymat

Performatifs recommend

- Le middle-agent joue un rôle de matchmaker
- Le contenu du message est un acte de langage
- L'émetteur souhaite connaître l'adresse d'agents pouvant satisfaire l'acte spécifié dans le contenu
- Connaître un tel agent : `recommend-one`
- Connaître tous ces agents : `recommend-all`

Performatifs recommend



Performatif recommend-one

```
(recommend-one
      :sender           <word>
      :receiver         <word>
      :in-reply-to      <word>
      :language         <word>
      :ontology         <word>
      :content          (performative_name
                         :sender    <word>
                         :language  <word>
                         :ontology  <word>
                         :content   <expression>))
```


Ex: recommend-one

```
(recommend-one
      :sender           C
      :receiver         facilitator
      :in-reply-to      id3
      :language         KQML
      :ontology         kqml-ontology
      :content          (ask-if
                         :sender    C
                         :language  Prolog
                         :ontology  foo
                         :content   "bar(X,Y)"))
```

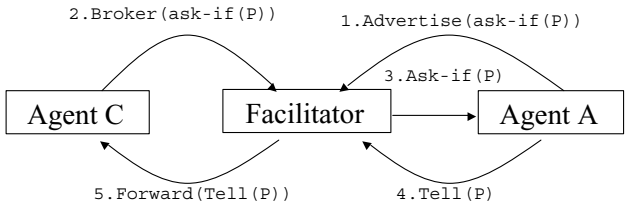

Ex: recommend-one

```
(forward
   :from          A
   :to            C
   :sender        A
   :receiver      facilitator
   :content       (advertise
                    :sender   A
                    :receiver C
                    :content  (ask-if
                                :sender   C
                                :receiver A
                                :language Prolog
                                :ontology foo
                                :content  "bar(X,Y)")))
```


Performatifs broker

- Le middle-agent joue un rôle de broker (avec anonymat)
- Le contenu du message est un acte de langage
- L'émetteur souhaite recevoir une réponse à l'acte spécifié dans le contenu
- Recevoir une réponse : `broker-one`
- Recevoir toutes les réponses : `broker-all`

Performatifs broker



Performatif broker-one

```
(broker-one
      :sender          <word>
      :receiver        <word>
      :in-reply-to     <word>
      :language        <word>
      :ontology        <word>
      :content         (performative_name
                           :sender      <word>
                           :language    <word>
                           :ontology    <word>
                           :content     <expression>))
```


Ex: broker-one

```
(broker-one
 :sender          C
 :receiver        facilitator
 :in-reply-to     id3
 :language        KQML
 :ontology        kqml-ontology
 :content         (ask-if
  :sender   C
  :language Prolog
  :ontology foo
  :content  "bar(X,Y)"))
```


Ex: broker-one

```
(ask-if
  :sender   facilitator
  :receiver A
  :language Prolog
  :ontology foo
  :content  "bar(X,Y)"))
```

```
(tell
  :sender   A
  :receiver facilitator
  :language Prolog
  :ontology foo
  :content  "bar(X,Y)"))
```

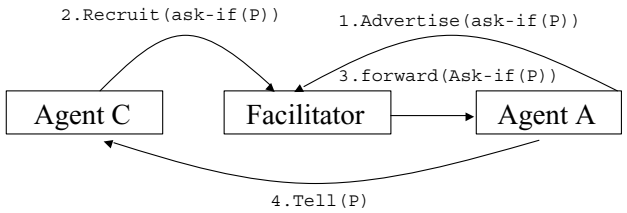
Ex: broker-one

```
(forward
   :from          C
   :sender        facilitator
   :receiver      C
   :in-reply-to   id3
   :language      KQML
   :ontology      kqml-ontology
   :content       (tell
                     :receiver C
                     :language Prolog
                     :ontology foo
                     :content  "bar(X,Y)"))
```


Performatifs recruit

- Le middle-agent joue un rôle de broker (sans anonymat)
- Le contenu du message est un acte de langage
- L'émetteur souhaite recevoir une réponse à l'acte spécifié dans le contenu provenant directement des agents y répondant
- Recevoir une réponse : `recruit-one`
- Recevoir toutes les réponses : `recruit-all`

Performatifs recruit



Performatif recruit-one

```
(recruit-one
   :sender        <word>
   :receiver      <word>
   :in-reply-to   <word>
   :language      <word>
   :ontology      <word>
   :content       (performative_name
                     :sender   <word>
                     :language <word>
                     :ontology <word>
                     :content  <expression>))
```

Ex: recruit-one

```
(recruit-one
   :sender        C
   :receiver      facilitator
   :in-reply-to   id3
   :language      KQML
   :ontology      kqml-ontology
   :content       (ask-if
                     :sender        C
                     :language      Prolog
                     :ontology      foo
                     :content       "bar(X,Y)"))
```


Ex: recruit-one

```
(forward
   :from          C
   :to            A
   :sender        facilitator
   :receiver      A
   :content       (ask-if
                     :sender        C
                     :receiver      A
                     :language      Prolog
                     :ontology      foo
                     :content       "bar(X,Y)"))
```


Ex: recruit-one

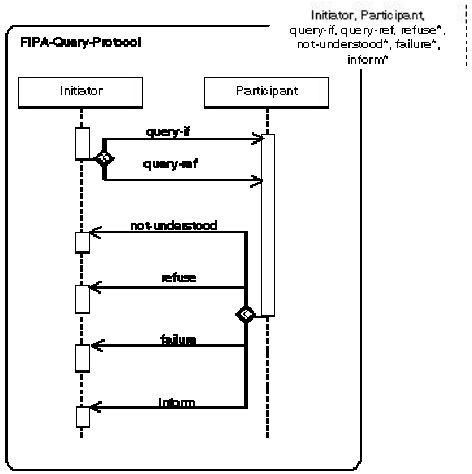
```
(tell
   :sender        A
   :receiver      C
   :language      Prolog
   :ontology      foo
   :content       "bar(X,Y)"))
```

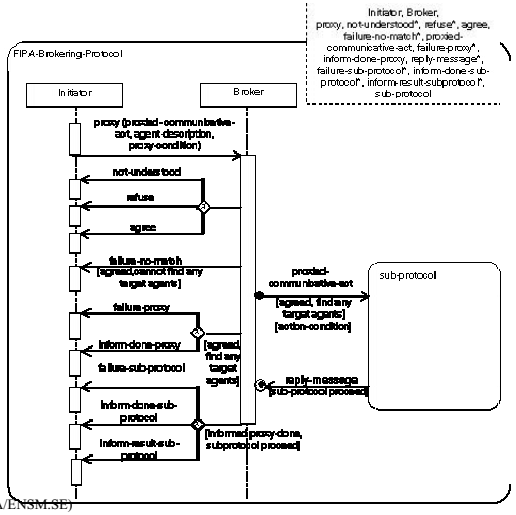

Protocoles FIPA

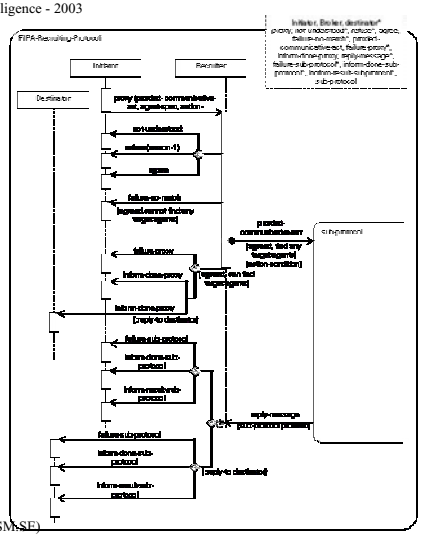
La FIPA a spécifié plusieurs protocoles contraignant les échanges de messages pour le brokering.

- protocole **Query** pour interagir avec un matchmaker
- protocoles **Brokering** et **Recruiting** pour interagir avec un broker

! l'anonymat peut être conservé dans les protocoles brokering et recruiting. Dans ce cas, la différence est que le protocole recruit peut préciser que l'agent recevant la réponse est **un autre agent que l'inititiateur** du protocole.







Ouverture avec KQML & FIPA

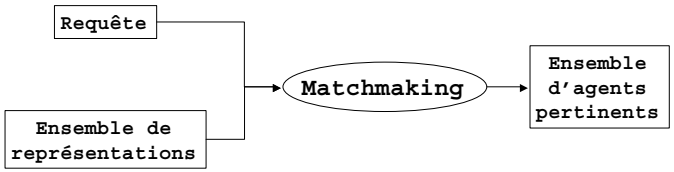
- KQML
  - ✓ Un agent signale au facilitateur qu'il met un service à disposition des autres par le performatif `advertise`
  - ✓ Un agent annule cette information par le performatif `unadvertise`
- FIPA
  - ✓ Les agents utilisent le DF comme un service bas niveau
  - ✓ fonctions `register`, `deregister`, `modify`

Plan

- Le problème de connexion
- Architectures pour les connexions
- Interaction dédiées
- Description d'agent & Matchmaking

Matchmaking

Définition [Ogston & Vassiliadis, 01] : *Le matchmaking est le processus permettant à un agent ayant une requête donnée de trouver les agents proposant un ou plusieurs services qui y répondent.*



Catégories de matchmaking

Le matchmaking intervient pour détecter des connexions :

- **ponctuelles**
  - ✓ requêtes immédiates et consommables
  - ✓ coopérations non prévisibles
  - ✓ ex : recherche d'information, SMA homogènes, ...
- **persistantes**
  - ✓ types de besoins identifiés et activables
  - ✓ coopérations régulières
  - ✓ ex : chaîne de production, groupement d'entreprises, conception de système, ...

Connexions ponctuelles

- **Représentation d'agent**

$$R_i = (s_i^1, \ldots, s_i^n)$$

- **Complémentarité**

$$complementary(s,r) \rightarrow \text{▯}$$

où s est un service et r une requête

- **Matching**

$$match(r) \rightarrow \{(A_i,\{(s_i^k,x_i^k), (s_i^l,x_i^l), \ldots\}), (A_j,\{(s_j^m,x_j^m), \ldots\}), \ldots\}$$

où A_i est un agent et x_i^k est l'estimation de complémentarité entre la requête r et le service s_i^k

Ex:FIPA 2000

- Représentation d'un service :
  - un **type**;
  - les **protocoles**, **ontologies** et **langages** supportés;
  - l'**agent** proposant le service;
  - une liste de **propriétés**.
- *complementary(s,r)* → {0,1}
- *match(r)* → $\{(A_i,\{s_i^k, s_i^l, ...\}), (A_j,\{s_j^m, ...\}), ...\}$

Ex: Larks [Sycara *et al., 01]*

- Représentation d'un service :
  - un **contexte** sous la forme de mots-clés;
  - les **types de données** manipulées;
  - ses **entrées/sorties** assorties de contraintes;
  - des **explications** sur les concepts ambigus;
  - une **description textuelle** libre.
- *complementary(s,r)* →
- *match(r)* → $\{(A_i,\{(s_i^k,x_i^k),...\}), ...\}$

Connexions persistantes

- **Représentation d'agent**

$$R_i = (c_i^1, ..., c_i^n, b_i^1, ..., b_i^m)$$

où c_i^k est une compétence et b_i^l un besoin

- **Complémentarité**

complementary(c,b) →

- **Matching**

match(b) → $\{(A_i,\{(c_i^k,x_i^k), (c_i^l,x_i^l), ...\}), (A_j,\{(c_j^m,x_j^m), ...\}), ...\}$

où A_i est un agent et x_i^k est l'estimation de complémentarité entre le besoin *b* et la compétence c_i^k

Description externe [Sichman, 95], [Vercouter,00]

- **Représentation d'agent**

$$R_i = (COMP_i, BES_i)$$

- **Compétences:** $COMP_i = (G_i, P_i, A_i, R_i)$
  - G_i est un ensemble de buts
  - P_i est un ensemble de plans
  - A_i est un ensemble d'actions
  - R_i est un ensemble de ressources

Description externe [Sichman, 95], [Vercouter,00]

- **Besoins:** $BES_i = (PN_i , TN_i , RN_i)$
  - ✓ PN_i est un ensemble de besoins en plan

    $pn \in PNi \supset \exists g \in Gi / g = goal(pn)$
  - ✓ TN_i est un ensemble de besoins en tâche

    $tn \in TNi \supset \exists p \in Pi , \exists t \in body(p) / t = task(tn)$
  - ✓ RN_i est un ensemble de besoins en ressources

    $rn \in RNi \supset \exists a \in Ai , \exists r \in uses(a) / r = resource(rn)$

Conclusion

- Il existe des techniques de brokering adapté à différents contextes
- Spécifications précises de performatifs et de protocoles
- La gestion de l'ouverture est réalisée conjointement à la résolution du problème de connexion
- Et elles ne passent pas nécessairement par un middle-agent !

Bibliographie

[Bayardo *et al.*, 98] INFOSLEUTH: Agent-based semantic integration of information in open and dynamic environments, *Readings in agents*: 205-216, 1998

[Davis & Smith, 83] Negociation as a metaphor for distributed problem solving, *Artificial Intelligence*, 20 (1): 63-109, 1983

[FIPA] http://www.fipa.org

[Ogston & Vassiliadis, 01] Local distributed matchmaking, *Technical report*, 2001

[Sichman, 95] Du raisonnement social chez les agents, *thèse de doctorat*, INP Grenoble, 1995

[Sycara, 01] Multi-agent infrastructure, agent discovery, middle-agents for web services and interoperation, *Multi-agent systems and applications*: 17-49, 2001

Bibliographie

[Sycara *et al.*, 01] LARKS: Dynamic matchmaking among heterogeneous software agents in cyberspace, *Journal on Autonomous agents an multi-agent systems*, 2001

[Vercouter, 00] Conception et mise en œuvre de systèmes multi-agents ouverts et distribués, *thèse de doctorat*, ENSM.SE, UJM, 2000