Systèmes Multi-Agents

Badr Benmammar

Systèmes Multi-Agents

Badr Benmammar

bbm@badr-benmammar.com

Définition d'un SMA

- Un système multi-agents est un ensemble organisé d'agents.

- Il est constitué d'une ou plusieurs organisations qui structurent les règles de cohabitation et de travail collectif entre agents.

  - Dans un même système, un agent peut appartenir à plusieurs organisations.

Communication entre agents

- Un agent doit être capable de communiquer avec les autres agents.
- Les agents doivent avoir des capacités à manipuler un langage commun.
- 2 types de communication :
  - Communication indirecte : Partage d'informations
    - via l'environnement,
  - Communication directe :
    - envoi de messages,

Communication entre agents

- L'agent peut participer à un dialogue en étant passif ou actif.
  - Un agent passif doit accepter les questions des autres agents et répondre à leur questions.
  - Un agent actif doit proposer et envoyer des interrogations.

- Dans un dialogue les agents alternent des rôles actifs et passifs, et échangent des séries de messages en respectant des protocoles biens précis, ce sont les protocoles de coordination, de coopération et de négociation.

Communication entre agents

La coopération

La coordination

La négociation

Les protocoles de coordination

- Les protocoles de coordination aident les agents à gérer leurs engagements.
- Les protocoles de coordination lui permettent de gérer ces engagements dans le cas où les circonstances dans lesquelles ils ont été élaborés, évoluent.
- Ils définissent aussi sous quelles conditions les engagements peuvent être revus et quelles sont alors les actions à prendre.

Les protocoles de coopération

- La coopération entre les agents consiste à décomposer les tâches en sous-tâches puis à les répartir entre les différents agents, il existe plusieurs décompositions possibles, le processus de décomposition doit donc tenir compte **des ressources disponibles** et **des compétences** des agents.

La négociation

- La négociation intervient lorsque des agents interagissent pour prendre des décisions communes, alors qu'ils poursuivent des buts différents.
- Les deux principales voies sur la négociation sont :
  - Les langages de négociation
  - Le processus de négociation

La négociation

- Les langages de négociation : il s'agit d'étudier les primitives de communication pour la négociation, leur sémantique et leur usage dans les protocoles.

- Le processus de négociation : il s'agit de proposer des modèles généraux de comportements des agents en situation de négociation.

La négociation

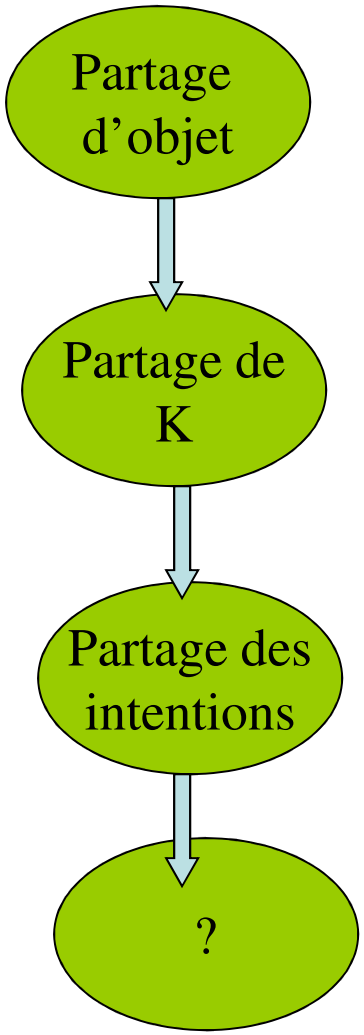
- Deux techniques de négociation :
  - La négociation centrée sur l'environnement : adapter le contexte ou l'environnement à la négociation.
  - La négociation centrée sur l'agent : adapter le comportement de l'agent compte-tenu des propriétés du contexte donné.

Communication entre agents

- La communication inter-agent est fondamentale à la réalisation du paradigme agent, tout comme le développement du langage humain était la clé du développement de l'intelligence humaine et des sociétés.

- Pour échanger les informations et les connaissances, les agents utilisent des **ACL (Agent Communication Language).**

Évolution ACL



-> Partage des objets, des appels de procédure et de SDD (CORBA, RPC, …)

-> Partage des connaissances (faits, règles, contraintes, procédures, …)
KIF, KQML, FIPA, ….

-> Partage des intentions (croyances, buts, intentions) => Niveau intentionnel
Théorie BDI

-> Que peut-on partager d'autre ?
Expériences, Stratégies…

Le modèle BDI et la communication

- Communication ⇔
  - Révéler à l'autre l'état de nos croyances, désires et intentions.
  - Essayer d'influencer l'état des croyances, désires et intentions de l'autre.
  - Un agent a des croyances sur le monde (son environnement), sur les croyances des autres agents et sur les croyances qu'ont les autres agents sur lui …

Langages de communications entre agents

- Tout langage multi agent est représenté par une structure de donnée comprenant les champs :
  - **Emetteur**
  - **Récepteur**
  - **Langage utilisé :** langage dans lequel le vrai message est rédigé
  - **Contenu du message :** le vrai message qui fait l'objet de la communication
  - **L'ontologie :** le vocabulaire dans un domaine donné pour que les agents puissent se comprendre
    - Ensemble de définitions concernant le message

Langages de communications entre agents

- De nombreux langages de communications entre agents (ACL) se sont développés.
  - KQML (93, 97)
  - FIPA-ACL (97, 99, 2000)

KQML (Knowledge Query and Manipulation Language)

- KQML a été conçu comme étant à la fois un format de message et un protocole de transfert de messages venant aider les agents intelligent au partage et échange de données de haut niveau tout en étant indépendant des machines.
- KQML est un langage qui se présente sur 3 couches :

La couche contenu
La couche communication
La couche message

KQML

La couche « Contenu »

- Il s'agit du contenu réel du message, il peut être écrit dans le langage de représentation du programme de l'agent.

  - KQML peut transporter des messages écrits dans n'importe quel langage de représentation (ex : PROLOG, KIF, LISP, C, KQML (lui même), XML ...)

KQML

La couche « Communication »

- Dans cette couche on retrouve des informations d'un niveau un peu plus bas permettant le bon fonctionnement de la communication telles que l'identité de l'émetteur ou celle du récepteur du message, ainsi qu'un identificateur unique pour le message.

KQML

- **La couche « Message » :**
  - Cette couche constitue le cœur du langage.
  - Elle incorpore des argument décrivant le contenu du message telles que le langage utilisé, l'ontologie…
  - Ces arguments permettent à KQML d'analyser, acheminer et délivrer les messages même si leur contenu lui est opaque et inaccessible.

KQML : la syntaxe

(KQML-performative
Niveau message
    :language<texte>
    :ontology<texte>

Niveau communication
    :sender<texte>
    :receiver<texte>

Niveau contenu
    :content<expression>

KQML : les performatives

36 performatives répartis en 3 catégories :

- Les 18 performatives de discours : servent à échanger des connaissances et des informations présentes dans la base de connaissance de l'agent
  - (ask-if, ask-one, tell, describe, stream-all …)
- Les 11 performatives d'interconnexion : aide à la mise en relation des agents entre eux
  - (register, unregister, broadcast …)
- Les 7 performatives d'exception : servent à changer le déroulement normal des échanges
  - (error, sorry, standby …)

Quelques performatives

E : l'agent émetteur
R : l'agent récepteur
C : le contenu du message
BVC : la base virtuelle de connaissances (connaissances attribuées par chaque agent aux autres agents)

- ask-one : E veut que seulement R réponde à sa question C
- ask-if : E veut savoir si la réponse à la question précisée en C se trouve dans la BVC de R
- tell : E affirme au R que C est dans la BVC de E
- broadcast : E veut que R transmette à son tour la performative à toutes ses connexions
- error : E considère le message précédent de R comme mal formé
- sorry : R ne peut pas fournir plus d'information
- http://www.exso.com/courses/cs101c/kqml/node6.html

Exemple de KQML

- l' agent A veut connaître toutes les personnes définies comme étant des hommes.

```
(ask-all
    :sender A
    :receiver B
    :language PROLOG
    :ontology phylosophie
    :content
    homme (=x)
    :reply_with question1
)
```

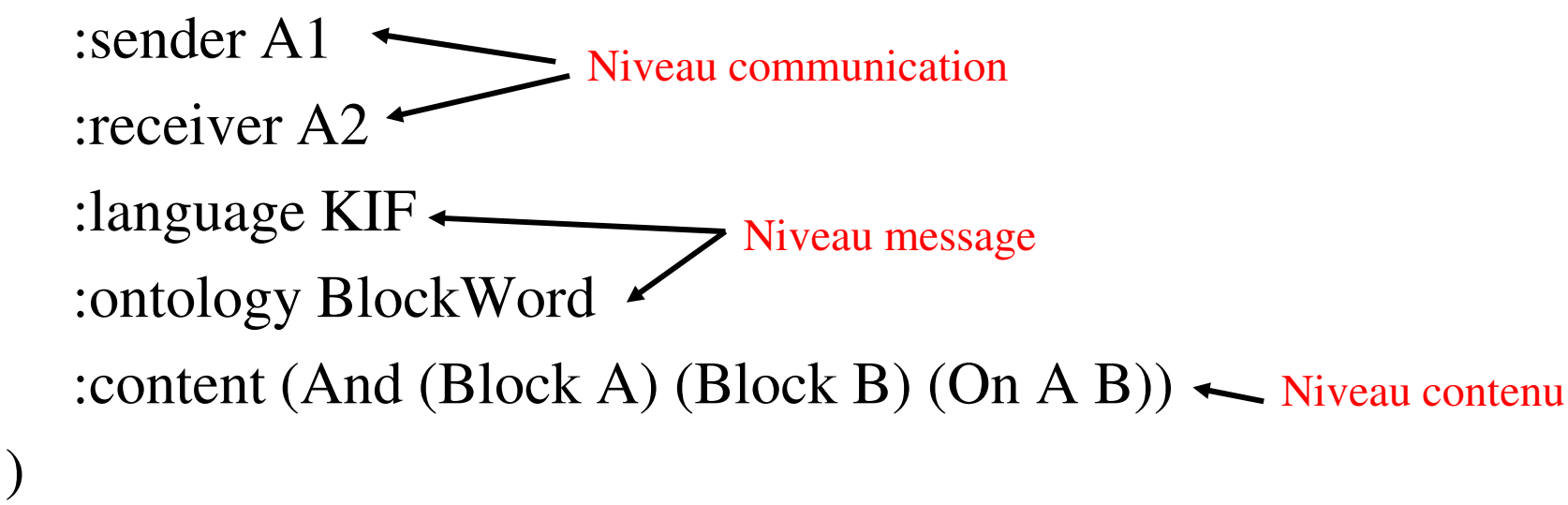
Exemple de KQML

- l' agent B répond que Socrate est un homme.

```
(tell
    :sender B
    :receiver A
    :language PROLOG
    :ontology phylosophie
    :content
    homme(Socrate)
    :in_reply_to question1
)
```

Exemple de KQML

- l'agent A1 veut informer l'agent A2 que le bloc A est sur le bloc B



Exemple imbriqué

- A1 demande à A3 de transférer le message précédent à A2

```
(forward
:from A1
:to A2
:sender A1
:receiver A3
:language KQML
:ontology kqml-ontology
:content (tell        :sender A1
                      :receiver A2
                      :language KIF
                      :ontology BlockWord
                      :content (And (Block A) (Block B) (On A B))))
```

Exemple de KQML

L'agent A1 demande à l'agent A2 le prix d'un portable Nokia et A2 lui répond.

```
(ask-one
      :sender A1
      :receiver A2
      :content (val (prix Nokia-6100))
      :language KIF
      :ontology portables
)
(tell
      :sender A2
      :receiver A1
      :content (=(prix Nokia-6100) (scalar 290 Euro))
      :language KIF
      :ontology portables
)
```

Exemple de KQML

L'agent A1 demande à l'agent A2 toutes les informations concernant le portable Nokia 6100, et A2 lui répond par plusieurs messages qui se terminent avec un '*eos*':

```
(stream-about
      :sender A1
      :receiver A2
      :reply-with N6100
      :language KIF
      : ontology portables
      :content Nokia-6100
)
(tell
      :sender A2
      :receiver A1
      :in-reply-to N6100
      :content (=(automomy Nokia-6100)
      (scalar 48 hours))
      :language KIF
      :ontology portables
)
```

```
(tell
            :sender A2
            :receiver A1
            :in-reply-to N6100
            :content   (=(prix Nokia-6100)
            (scalar 290 Euro))
            :language KIF
            :ontology portables
)
(eos
            :sender A2
            :receiver A1
            :in-reply-to N6100
)
```

Exemple de KQML

Dans une application en Java, l'agent A veut connaître la moyenne de 4 nombres

```
(ask-one
      :sender A
      :receiver B
      :language Java
      :ontology mathématique
      :content
      double m = moyenne_B (n1, n2, n3, n4)
      :reply_with question3
)
```

Plates-formes des systèmes multi-agents && KQML

- Une plate-forme de développement des systèmes multi-agents est une infrastructure de logiciels utilisée comme environnement pour le déploiement et l'exécution d'un ensemble d'agents.
- **AgentBuilder**
  - http://www.agentbuilder.com
  - AgentBuilder est entièrement programmé en Java.
  - Les agents construits en utilisant AgentBuilder communiquent en utilisant KQML.
  - Disponible pour Windows 98/ME/NT/2000/XP, Solaris et Linux
- **JAT : Java(tm) Agent Template**
  - http://www-cdr.stanford.edu/ABE/JavaAgent.html
  - La communication est basée sur KQML.
- **Java Intelligent Agent Library**
  - http://www.bitpix.com/business/main/bitpix.htm
  - La communication est basée sur KQML.
  - La librairie supporte aussi les agents mobiles.

La norme FIPA

- La FIPA (Foundation for Intelligent Physical Agents) est une organisation à but non lucratif fondée en 1996 dont l'objectif est de produire des standards pour l'interopération d'agents logiciels hétérogènes.

- **FIPA ACL :** Syntaxe similaire à celle de KQML.

FIPA-ACL

- sender : l'émetteur du message
- receiver : le destinataire du message
- reply-to : participant à l'acte de communication
- content : le contenu du message (l'information transportée par la performative)
- language : le langage dans lequel le contenu est représenté

FIPA-ACL

- ontology : le nom de l'ontologie utilisé pour donner un sens aux termes utilisés dans le *content*
- conversation-id : identificateur de la conversation
- reply-with : identificateur unique du message, en vue d'une référence ultérieure
- in-reply-to : référence à un message auquel l'agent est entrain de répondre (précisé par l'attribut reply-with de l'émetteur)
- reply-by : impose un délai pour la réponse

Catégorie de performatives FIPA

- Information
  - query_if, query_ref, sunscribe, inform, inform_if, inform_ref, confirm, disconfirm, not_understood
- Gestion des erreurs
  - not-understood, failure
- Négociation
  - cfp (Call for proposal), propose, accept_proposal, reject_proposal
- ……
- ……
  - http://jmvidal.cse.sc.edu/talks/agentcommunication/performatives.xml

FIPA-ACL: Les 20 performatives

performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

Exemple

- L'agent A veut informer l'agent B du temps qu'il fera demain, selon ses prévisions :

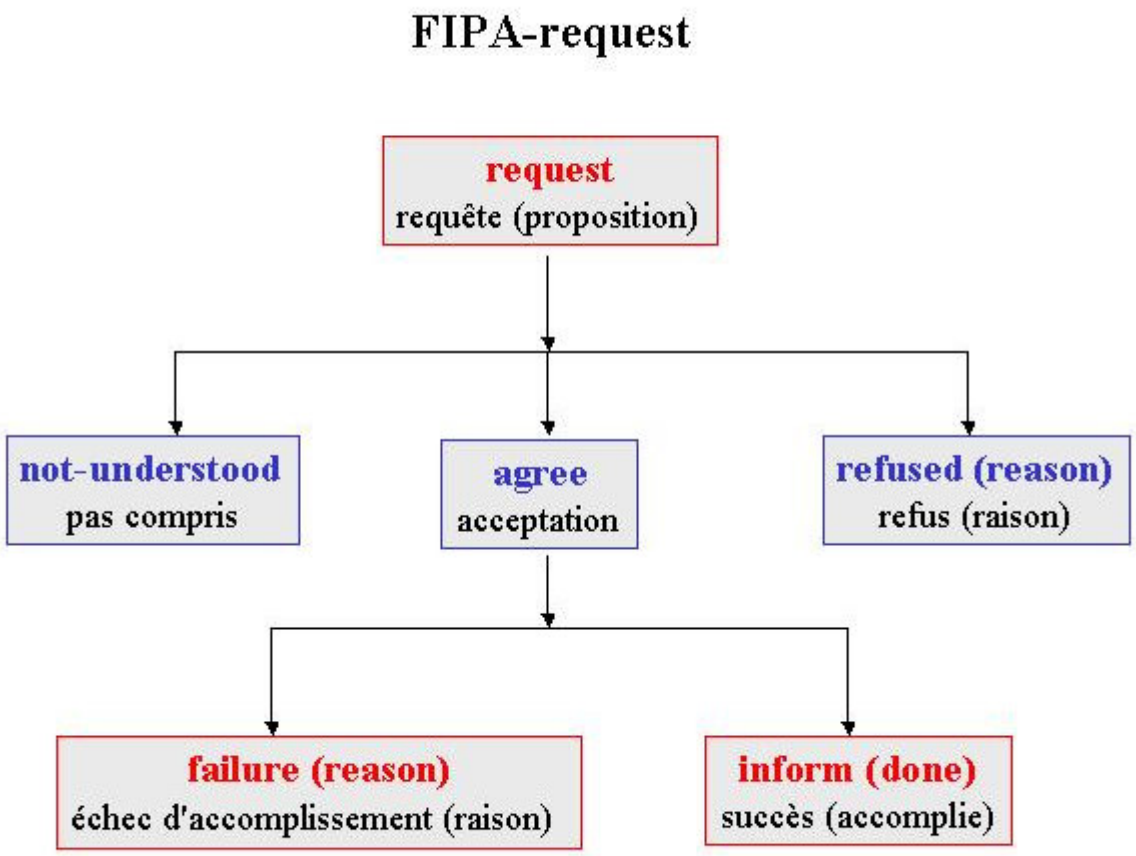
```
(inform
    :sender   A
    :receiver B
    :content temps (demain, pleuvoir)
    :language Prolog
 )
```

Les protocoles d'interaction FIPA

- Sémantique : une description de protocole représente un schéma d'interaction, ensemble de messages échangés entre différents agents

- Protocoles :
  - fipa-request: le récepteur est demandé d'exécuter une action
  - fipa-query : le récepteur est demandé d'exécuter un acte informatif

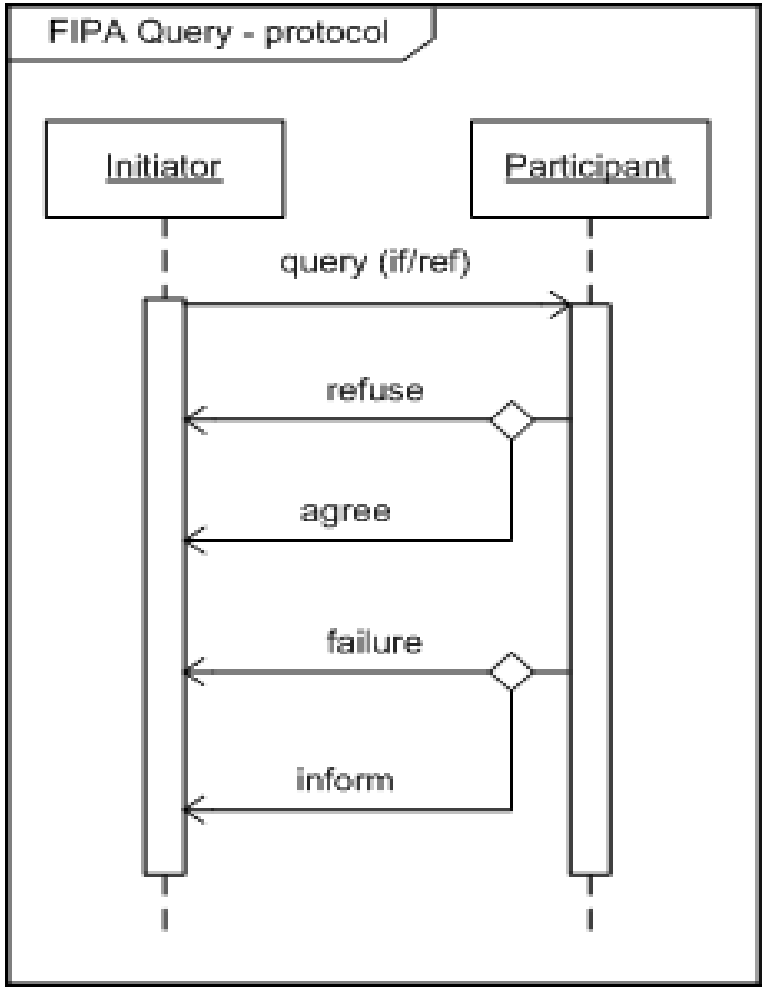
FIPA-Request



FIPA-Request

- Avec FIPA-request, un agent sollicite un autre agent pour exécuter des actions et l'agent récepteur retourne soit une réponse favorable à l'exécution d'actions, soit une réponse défavorable expliquée par telle ou telle raison. Supposons que l'agent ***i*** aie besoin de l'agent ***j*** pour exécuter l'action « action ».
  - L'agent ***i*** envoie « request » à l'agent ***j***.
    - Si l'agent ***j*** accepte la requête, il retourne « agree ». Ensuite, quand ***j*** a fini d'exécuter « action », il en informe ***i*** en utilisant « inform ».
    - Si l'agent ***j*** accepte mais rencontre un problème durant le traitement de « action », il retourne « failure » et les raisons de l'échec.
    - Si l'agent ***j*** n'accepte pas la requête de l'agent ***i***, ***j*** retourne « refuse » et les raisons de ce refus.

FIPA-Query


FIPA Query - protocol
Initiator
Participant
query (if/ref)
refuse
agree
failure
inform

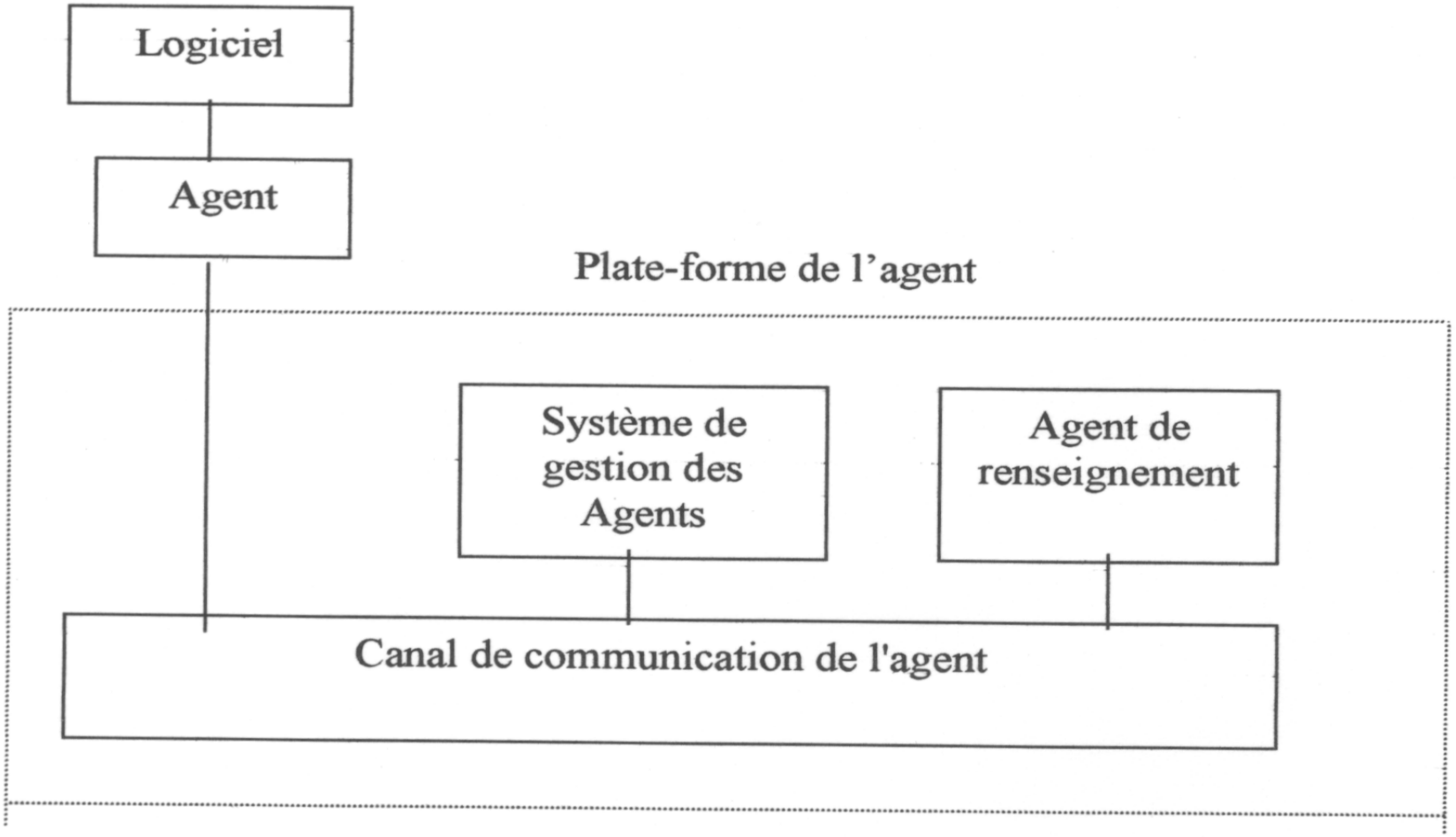
FIPA-Query

- FIPA-Query signifie que l'agent émetteur sollicite l'agent récepteur pour exécuter un des types d'un performatif «inform», c'est-à-dire pour répondre à la demande.
- Supposons que l'agent ***i*** fasse une demande à l'agent ***j***.
  - L'agent ***i*** envoie un performatif «query» à l'agent ***j***.
    - Si l'agent ***j*** peut répondre à la demande, il l'informe en utilisant le performatif « inform ».
    - Si l'agent ***j*** a essayé de répondre à la demande mais qu'il ne le peut pas, il retourne «failure» et les raisons de cette impossibilité.
    - Si l'agent ***j*** refuse de répondre à la demande, il retourne «refuse» et les raisons de ce refus.

Modèle de référence de l'agent

- Identifier les rôles de quelques agents clés nécessaires pour la gestion de la plate-forme.
  - **L'agent de renseignement** (**D**irectory **F**acilitator)
  - **L'agent de gestion du système** (**A**gent **M**anagement **S**ystem)
  - **Le canal de communication de l'agent** (**A**gent **C**ommunication **C**hannel)

Modèle de référence de l'agent



Modèle de référence de l'agent

- **L'agent de renseignement** (**D**irectory **F**acilitator)
  - Son rôle est de fournir le service "pages jaunes" aux autres agents.
  - Il enregistre les descriptions des agents ainsi que les services qu'ils offrent.
  - Il doit fournir l'information la plus actuelle sur les agents enregistrés dans son répertoire à tous les agents autorisés.

Modèle de référence de l'agent

- **L'agent de renseignement** (**D**irectory **F**acilitator)
  - L'agent de renseignement peut restreindre l'accès à l'information et vérifier toutes les permissions d'accès pour les agents qui tentent de l'informer des changements d'état de l'agent.
  - Les agents peuvent enregistrer leurs services auprès d'un DF ou demander à DF de découvrir les services offerts par d'autres agents.
  - Un DF au moins doit être présent sur chaque plate-forme d'agents.
  - Une plate-forme d'agents peut supporter plusieurs DF.

Modèle de référence de l'agent

- **L'agent de gestion du système** (**A**gent **M**anagement **S**ystem)
  - Cet agent est responsable de gérer les activités d'une plate-forme. Ses responsabilités consistent à créer des agents, supprimer des agents, décider si un agent peut dynamiquement s'inscrire sur la plate-forme et enfin superviser la migration d'agents de plate-forme en plate-forme.
  - Il contrôle l'accès ainsi que l'utilisation du canal de communication des agents.
  - Un seul système de gestion réside sur une plate-forme.

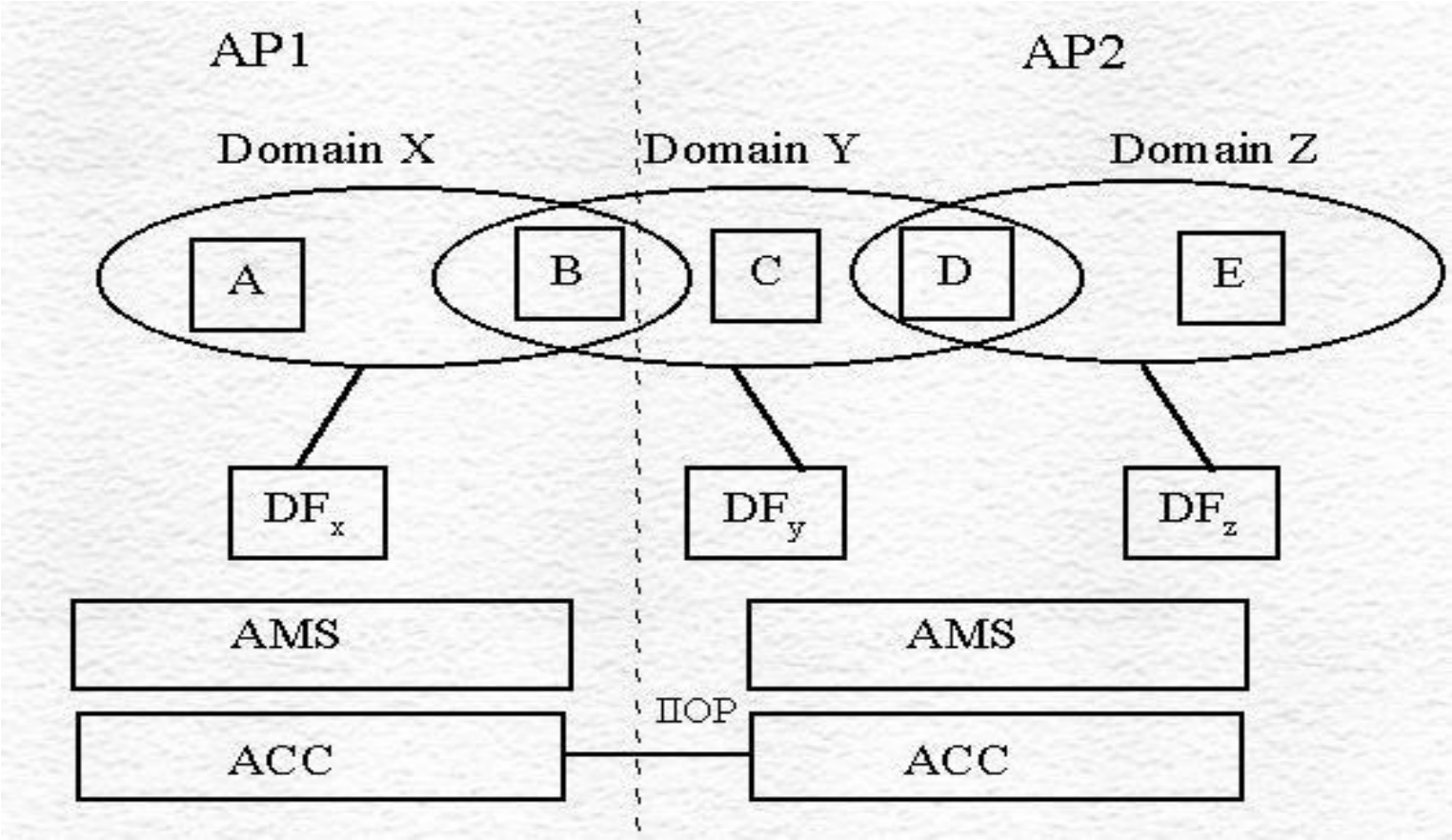
Modèle de référence de l'agent

- **Le canal de communication de l'agent** (**A**gent **C**ommunication **C**hannel)
  - Fournir le chemin pour un échange entre un agent et d'autres agents.
  - Il achemine des messages entre agents situés dans la plate-forme mais également à d'autres agents résidant sur d'autres plates-formes.
  - L'ACC est le moyen de communication par défaut qui connecte tous les agents sur une plate-forme et entre plates-formes.

Le domaine de l'agent

- Le domaine de l'agent est un groupement d'agents définis par appartenance à un répertoire supervisé par le DF.
- Le répertoire liste tous les agents dans le domaine et est utilisé pour faire connaître les services et les capacités disponibles.
- Un agent peut être présent dans un ou plusieurs domaines.
- Un agent doit s'inscrire auprès d'un DF afin d'être présent dans un domaine.

La plate-forme de l'agent (AP)

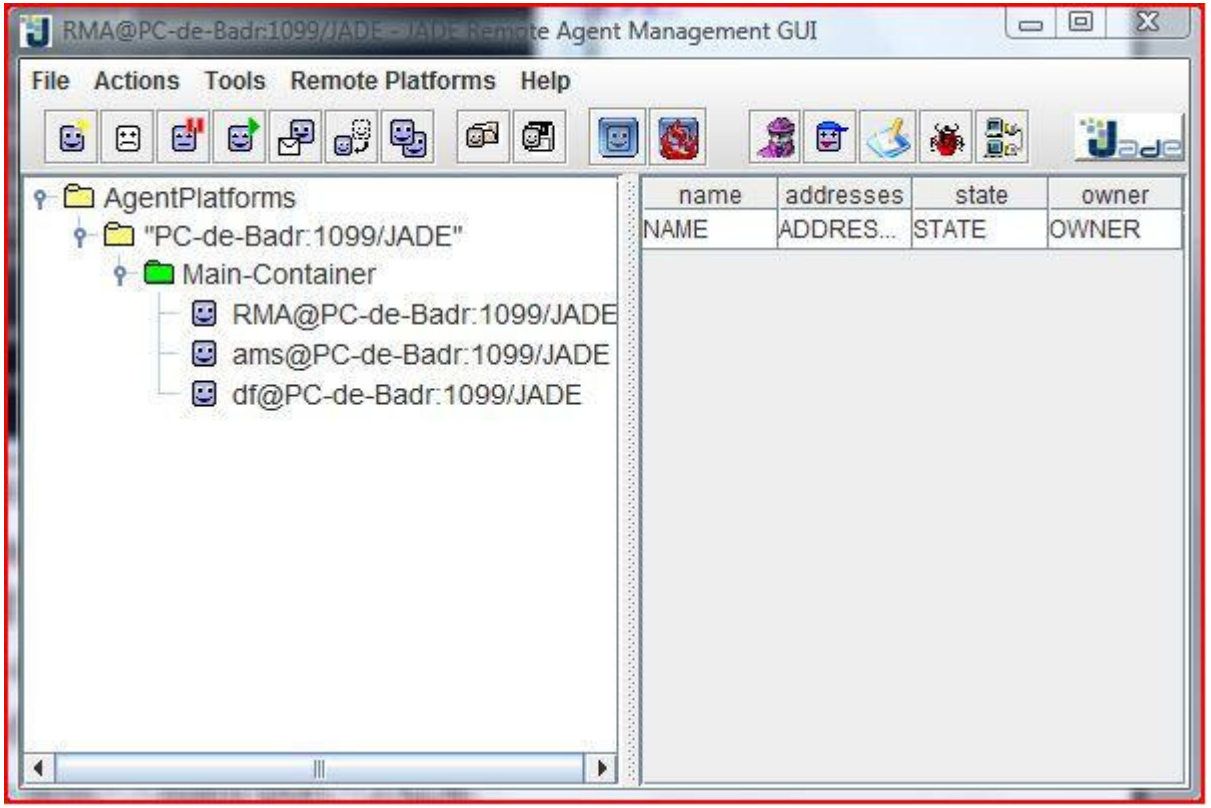


Plates-formes des systèmes multi-agents && FIPA

- **JADE : http://jade.tilab.com/**
  - Langage utilisé : Java.
  - FIPA ACL pour la communication.

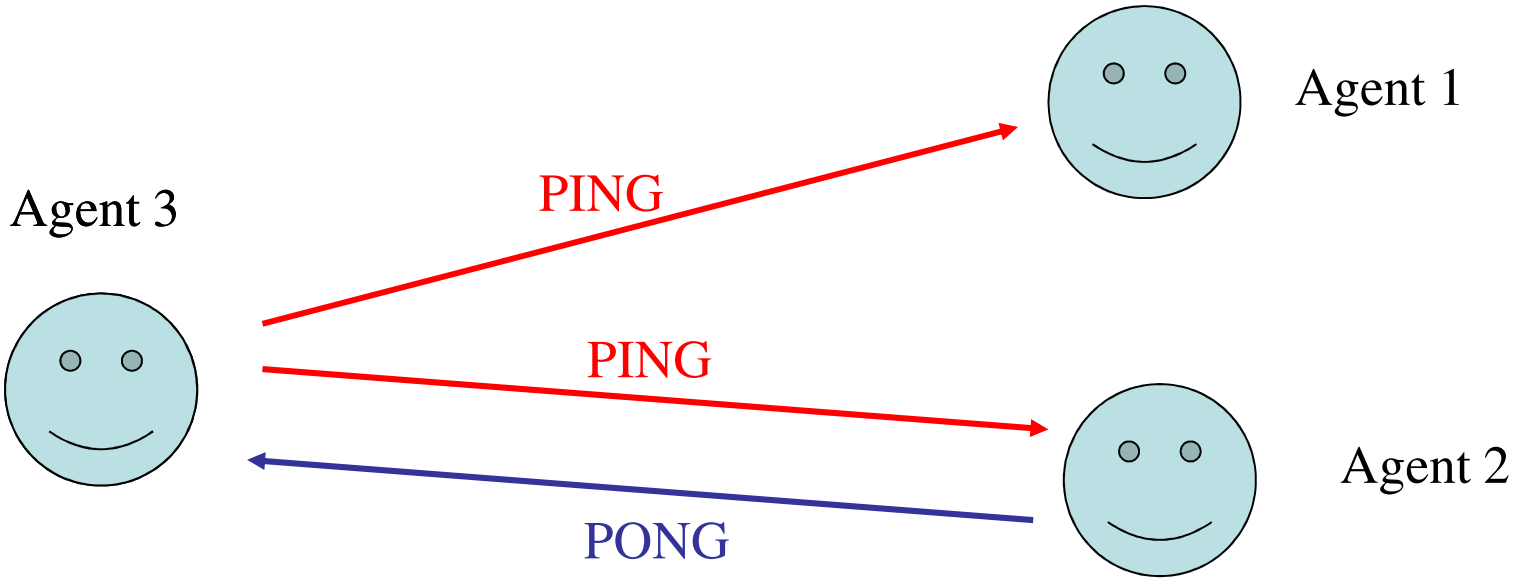
Jade GUI

- Contrôler les agents
  - Créer
  - Tuer
  - Suspendre
  - …
- Démarrer les autres outils



Exemple avec JADE

- Agent 3 envoie un message PING aux deux Agents Agent1 et Agent2.
- En plus de la confirmation de la réception du PING de la part des deux Agents, Agent2 envoie un Pong à Agent3.



```
import jade.core.Agent;
import jade.core.AID;
import jade.core.behaviours.*;


import jade.lang.acl.*;


public class Sender extends Agent
{
    protected void setup()
    {
    // First set-up answering behaviour

        addBehaviour(new CyclicBehaviour(this)
        {
            public void action() {
                ACLMessage msg= receive();
                if (msg!=null)
                    System.out.println(myAgent.getLocalName() + " <-" +
                    msg.getContent() + " from "
                     + msg.getSender().getName() );
                block();
            }
        });

    // Send messages to "a1" and "a2"

        ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
        msg.setContent("Ping");
        for (int i = 1; i<=2; i++)
            msg.addReceiver( new AID("Agent" + i, AID.ISLOCALNAME));
        send(msg);
    }
}
```

Agent 3

```
import jade.core.Agent;
import jade.core.behaviours.*;
import jade.lang.acl.*;


public class Receiver extends Agent
{

    protected void setup()
    {
        addBehaviour(new CyclicBehaviour(this)
        {
            public void action() {
               ACLMessage msg= receive();
               if (msg!=null)
                   System.out.println(myAgent.getLocalName() + " <- " +
                      msg.getContent() + " from "
                    +  msg.getSender().getName());
               block();
            }
        });
    }
    }
```

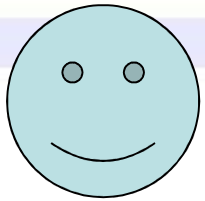
Agent 1

```
import jade.core.Agent;
import jade.core.behaviours.*;
import jade.lang.acl.*;

public class Pong extends Agent
{
    protected void setup()
    {
        addBehaviour(new CyclicBehaviour(this)
        {
            public void action()
            {
               ACLMessage msg = receive();
               if (msg!=null) {
                   System.out.println(myAgent.getLocalName() + " <- " +
                      msg.getContent() + " from "
                    +  msg.getSender().getName());

                   ACLMessage reply = msg.createReply();
                   reply.setPerformative( ACLMessage.INFORM );
                   reply.setContent(" Pong");
                   send(reply);
               }
               block();
            }
        });
    }
}
```

Agent 2



Exemple avec JADE

- java jade.Boot Agent3 :Sender Agent1 :Receiver Agent2 :Pong

Agent1 <- Ping from Agent3@PC-de-Badr:1099/JADE
Agent2 <- Ping from Agent3@PC-de-Badr:1099/JADE
Agent3 <- Pong from Agent2@PC-de-Badr:1099/JADE

- Lancer un agent sans l'interface :
  - java jade.Boot <nom de l'agent>:<classe de l'agent>
- Le nom d'un agent est de la forme :
  - <nom-agent>@<nom-plate-forme>
- La plate-forme par défaut est :
  - <main-host>:<main-port>/JADE
- Pour plus d'informations sur les numéros de port:
- http://www.frameip.com/liste-des-ports-tcp-udp/

Critères pour l'étude comparative de plates-formes SMA

- **Caractéristiques générales**
  - Nom de la plate-forme :
  - Auteurs :
  - Etat de la plate-forme :
    - Maquette -- Prototype -- Produit fini
    - Logiciel libre -- Produit Commercial
  - Nombre d'applications développées
    - des petits exemples – une application réelle – plusieurs applications réelles
  - Type d'application privilégié :
    - Simulation -- Résolution de problème
  - Configuration logicielle et matérielle
  - Langages utilisés
  - Systèmes d'exploitation

Critères pour l'étude comparative de plates-formes SMA

- **Modèles multi-agents disponibles**
  - Interaction/Communication
    - Modèle de communication : direct -- indirect
    - Langage de communication : signaux -- actes de langage -- autres
    - Protocoles d'interaction :
  - Caractéristiques extrinsèques du système multi-agents
    - Nombre d'agents (ordre de grandeur)
      - agent -- <10 --- aucune limite
    - Nature des agents
      - homogènes -- hétérogènes
    - Modèle de coopération
      - Négociation, délégation de tâches, planification multi-agents, …
    - Type de contrôle
      - Centralisé – hiérarchique -- distribué

Critères pour l'étude comparative de plates-formes SMA

- **Caractéristiques physiques du système multi-agents**
  - Agent
    - Implémentation : déclaratif – procédural
    - Mobilité : stationnaire – mobile
  - Organisation
    - Distribué - centralisé
    - Passif (ex : base de données) - dynamique

Critères pour l'étude comparative de plates-formes SMA

- **Environnement de développement**
  - Méthodologie d'analyse
  - Disponibilité d'un langage de description d'agents
  - Nécessité de connaître le langage de programmation dans lequel la plate-forme est écrite
  - Existence de bibliothèques
  - Documentation disponible
  - Tutoriel

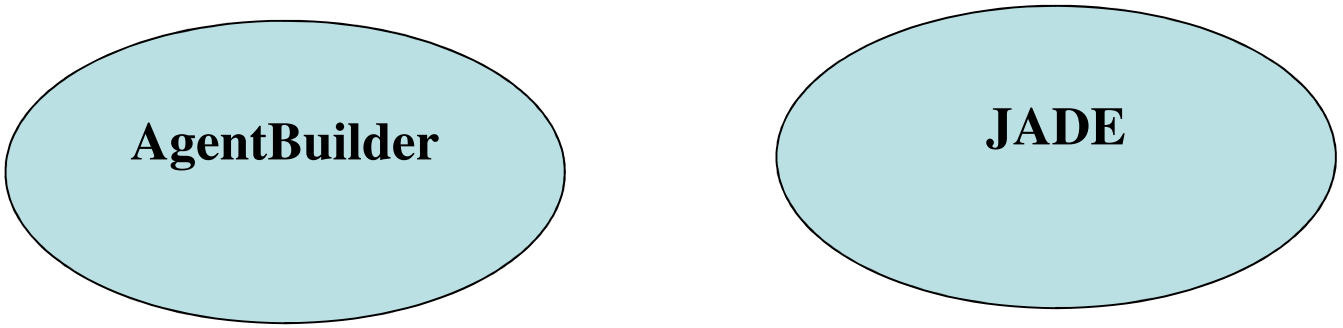
Critères pour l'étude comparative de plates-formes SMA

- **Environnement d'exécution**
  - Outils disponibles
    - Outils d'administration, configuration, lancement
    - Outils de monitoring
    - Outils de mise au point
  - Documentations disponibles
    - Aide en ligne
    - Manuels d'installation
    - Manuels d'utilisation
    - Manuels de maintenance
  - Support logistique
    - Fréquence des nouvelles versions
    - Support technique disponible

Comparaison entre plates-formes multi-agents

KQML

FIPA ACL



Exercice KQML

- Le but de cet exercice est de faire communiquer des agents implémentés en JAVA en utilisant le langage KQML, les interactions entre les agents simulent un jeu de loto.
- Le SMA contient 3 agents :
- **Agent1** : aide l'utilisateur à remplir une grille de 6 nombres compris entre 1 et 49.
- **Agent2 :** il fait le tirage des 6 nombres et traite la grille de l'utilisateur.
- **Agent3 :** envoie un commentaire à l'Agent1 en tenant compte des informations reçus de l'Agent2.
- La simulation de ce jeu se déroule comme suit :
- Agent1 envoie la grille saisie par l'utilisateur à Agent2 ainsi qu'une demande afin de connaître le nombre de nombres communs entre cette grille et le tirage réalisé par Agent2.
- Après la réception de la réponse envoyée par Agent2, Agent1 envoie une deuxième grille à Agent2 en lui demandant de simuler des tirages tant que l'utilisateur n'a pas les 6 bons numéros et de calculer le nombre de tirages effectués.
- Agent2 demande à Agent3 de transférer le nombre de tirages effectués à Agent1.
- Agent3 profite de cette information et envoie un commentaire à Agent1.
- Le code de Agent1 contient la fonction **grilleUtilisateur.**
- Le code de Agent2 contient 3 fonctions **tirage, nbCorrect** et **simulationJeu.**
- Le code de Agent3 contient la fonction **Réponseutilisateur.**

Exercice KQML

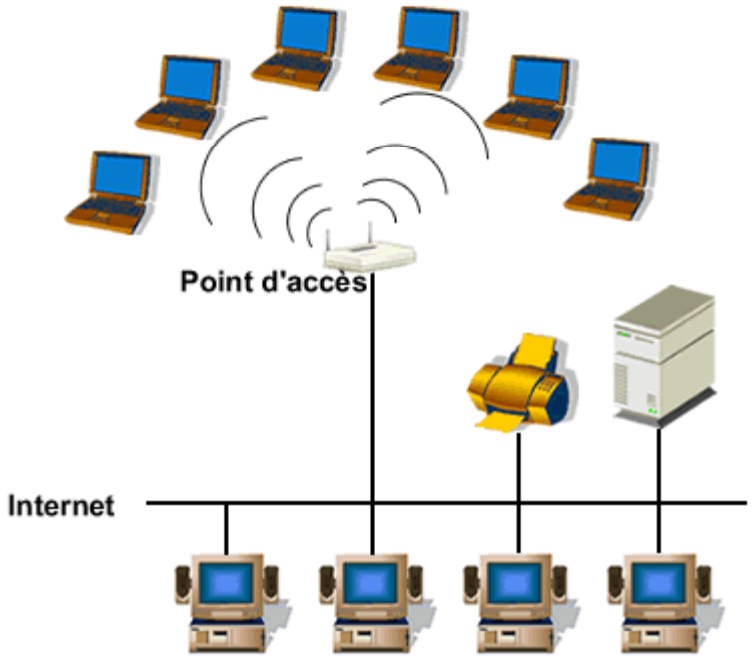
- 1. Ecrire la fonction **grilleUtilisateur** qui retourne un tableau de 6 nombres compris entre 1 et 49. Les nombres du tableau devront entre saisie par l'utilisateur. Un même nombre ne peut apparaître qu'une seule fois dans le tableau. Bien évidemment, il faut vérifier que l'utilisateur entre des nombres corrects.
- 2. Ecrire la fonction **tirage** qui retourne un tableau de 6 nombres compris entre 1 et 49. Un même nombre ne peut apparaître qu'une seule fois dans le tableau.
- 3. Ecrire la fonction **nbCorrect** qui prend en paramètre 2 tableaux de 6 nombres et qui retourne le nombre de nombres communs aux deux tableaux.
- 4. Ecrire la fonction **simulationJeu** qui prend comme paramètre la grille saisie par l'utilisateur afin de simuler des tirages tant que l'utilisateur n'a pas les 6 bons numéros et retourner le nombre de tirages effectués.
- 5. Ecrire la fonction **Réponseutilisateur** qui a pour paramètre le nombre de tirages effectués et qui doit retourner le commentaire « bon joueur » ou « mauvais joueur ».
- 6. Ecrire en langage KQML les interactions dans le système.

Communication des agents dans un environnement sans fil

Le contexte : Les réseaux sans fil

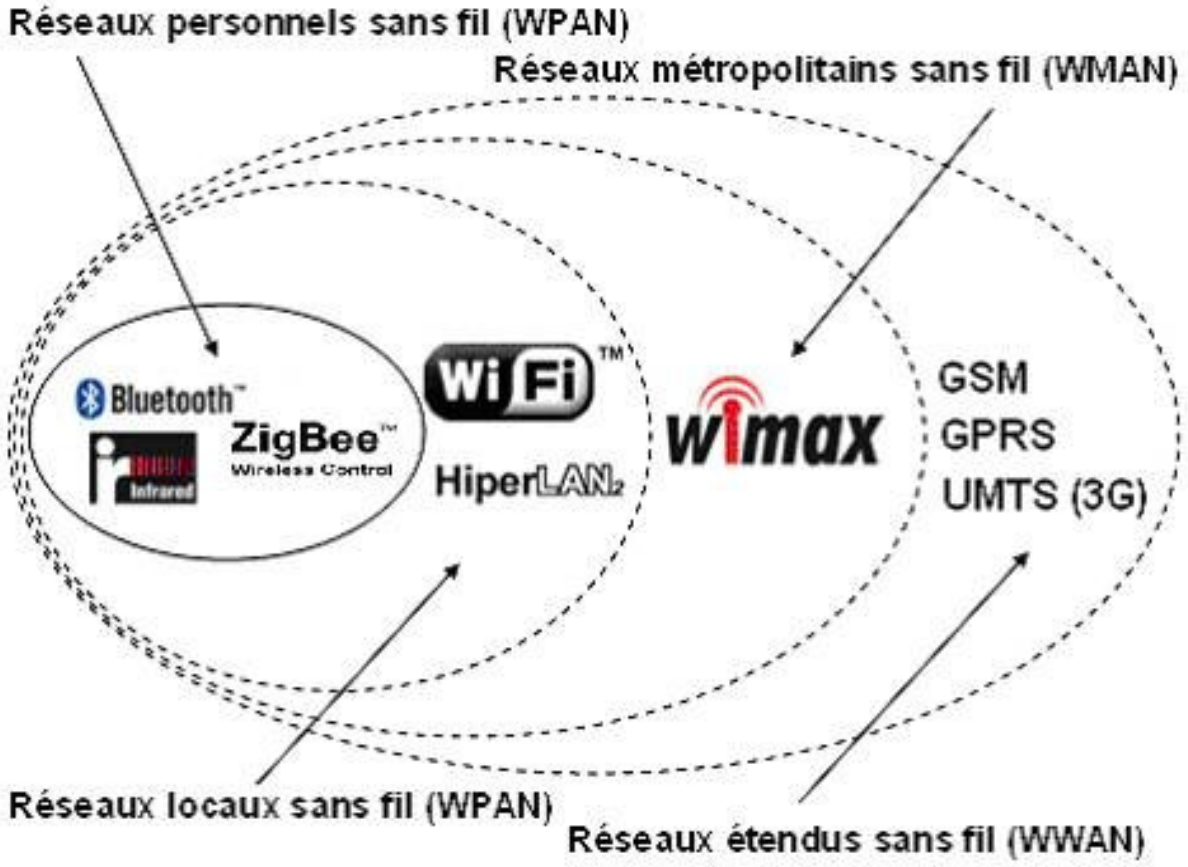
Qu’est-ce qu’un réseau sans fil

- Un **réseau sans fil** (*wireless network*) est, comme son nom l'indique, un réseau dans lequel au moins deux terminaux peuvent communiquer sans liaison filaire.



Les catégories de réseaux sans fil

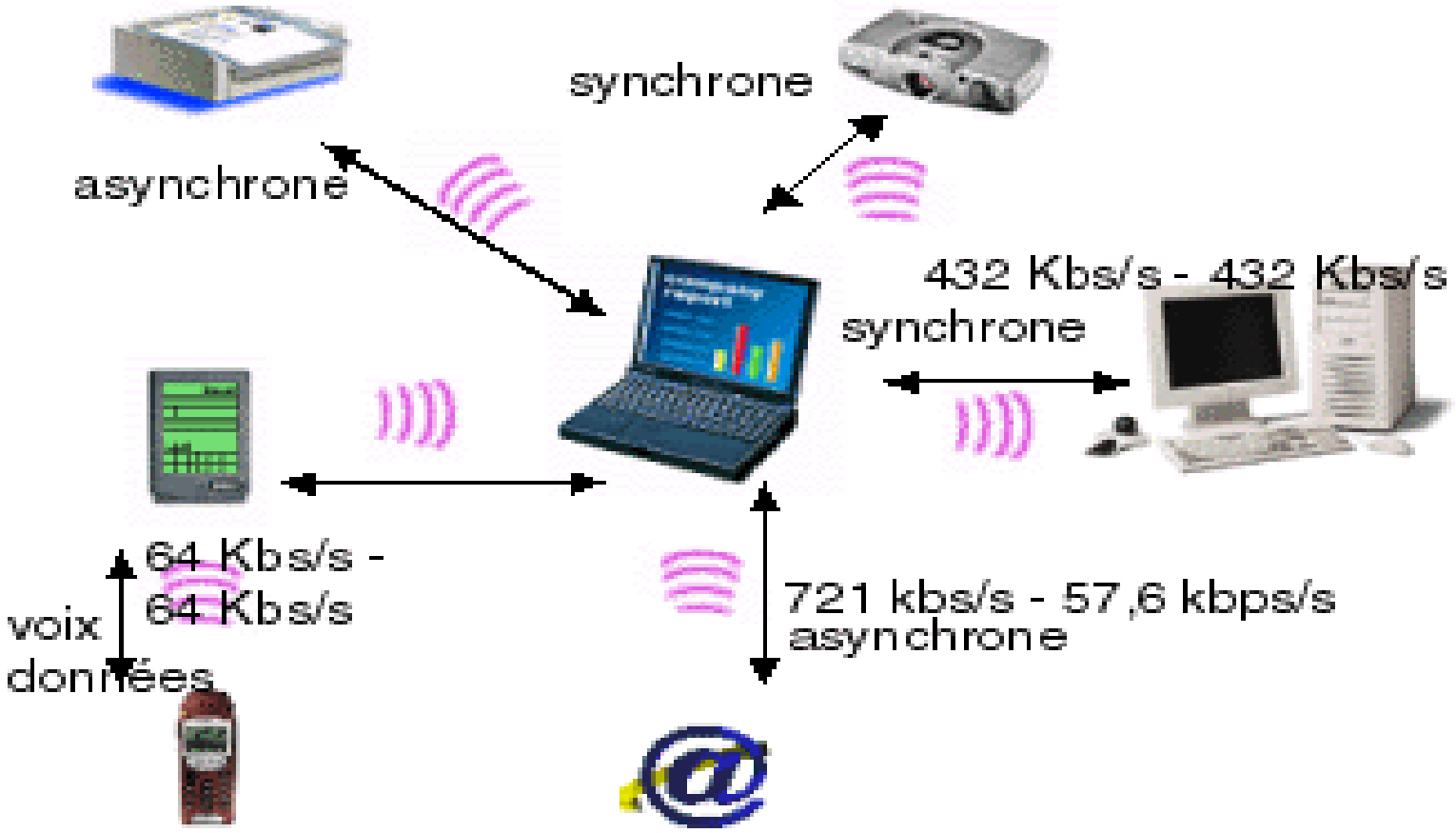
- On distingue habituellement plusieurs catégories de réseaux sans fil, selon le périmètre géographique offrant une connectivité (appelé *zone de couverture*) :



Les réseau sans fil

- WPAN : **Bluetooth,**
  - Wireless Personal Area Network
- WLAN : **Le Wifi,**
  - Wireless Local Area Network
- WMAN : **WiMAX,**
  - Wireless Metropolitan Area Network
- WWAN : **GSM, GPRS, UMTS**
  - Wireless Wide Area Network

Bluetooth

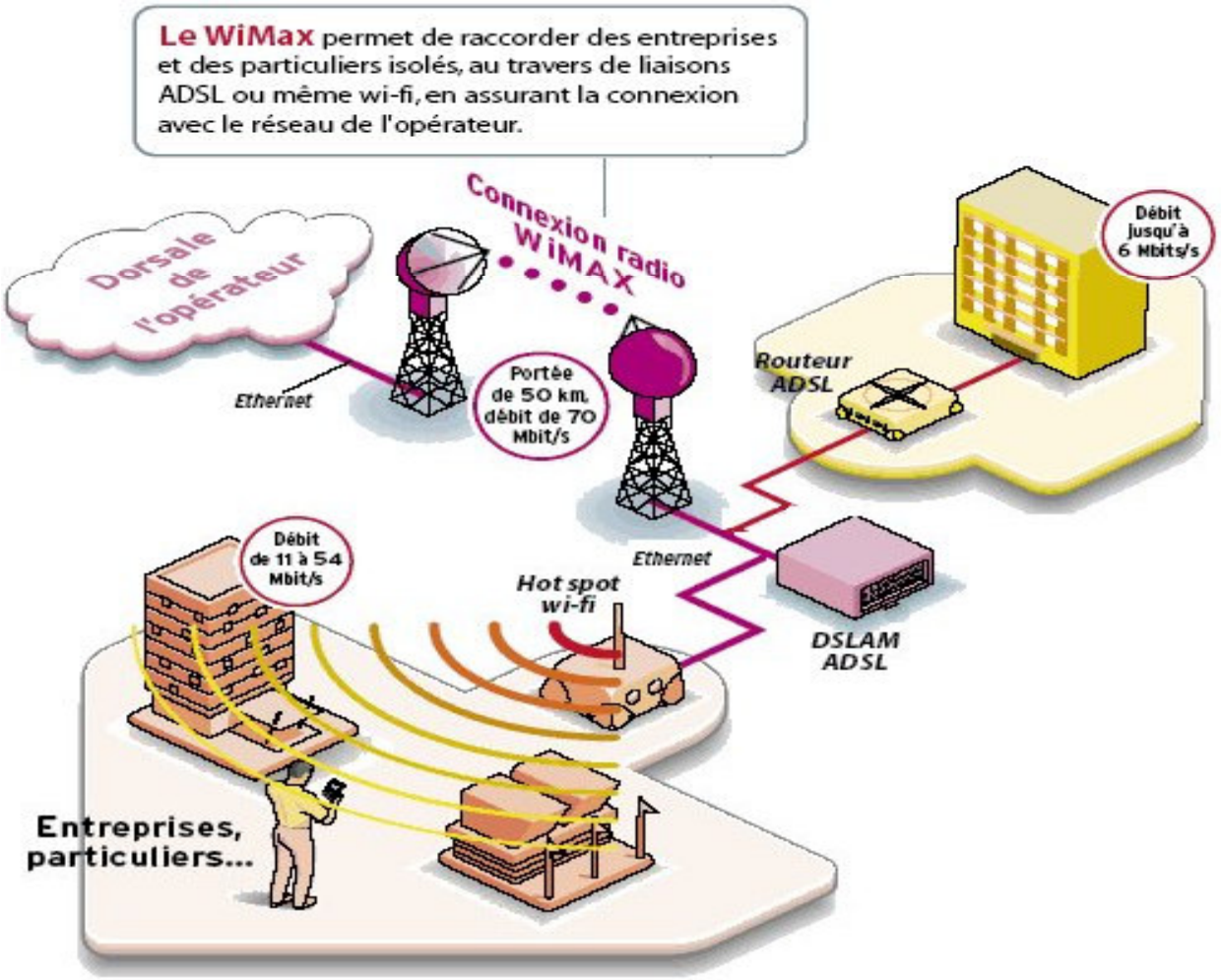

synchrone
asynchrone
432 Kbs/s - 432 Kbs/s
synchrone
64 Kbs/s -
64 Kbs/s
voix
données
721 kbs/s - 57,6 kbps/s
asynchrone

Le WiFi

- Le WiFi permet la mise en place des transmissions dans les endroits où la pose de câble est difficile, voir impossible.

WiMAX

- La norme de réseau métropolitain sans fil la plus connue est le **WiMAX**, permettant d'obtenir des débits de l'ordre de 70 Mbit/s sur un rayon de plusieurs kilomètres.

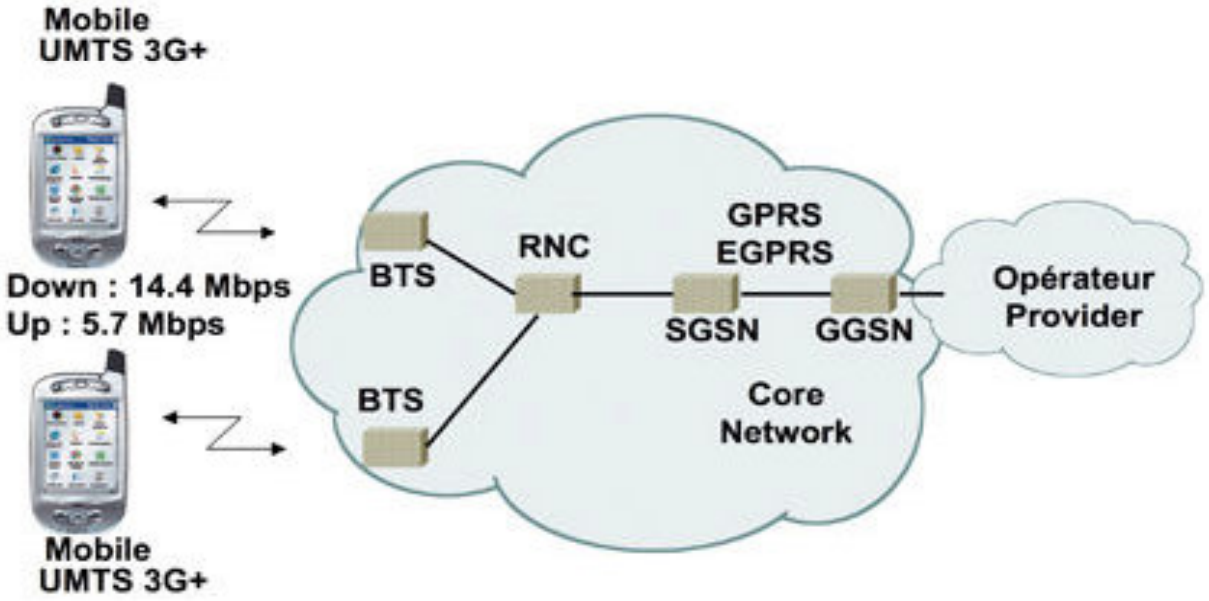


Réseaux étendus sans fil (WWAN)

- Les principales technologies sont les suivantes :
  - GSM (Global System for Mobile Communication ou en français Groupe Spécial Mobile)
  - GPRS (General Packet Radio Service)
  - UMTS (Universal Mobile Telecommunication System)

UMTS

- L'UMTS signifie système universel de télécommunications mobiles, c'est-à-dire un réseau mobile capable d'offrir des services multimédias, partout et à tout moment.

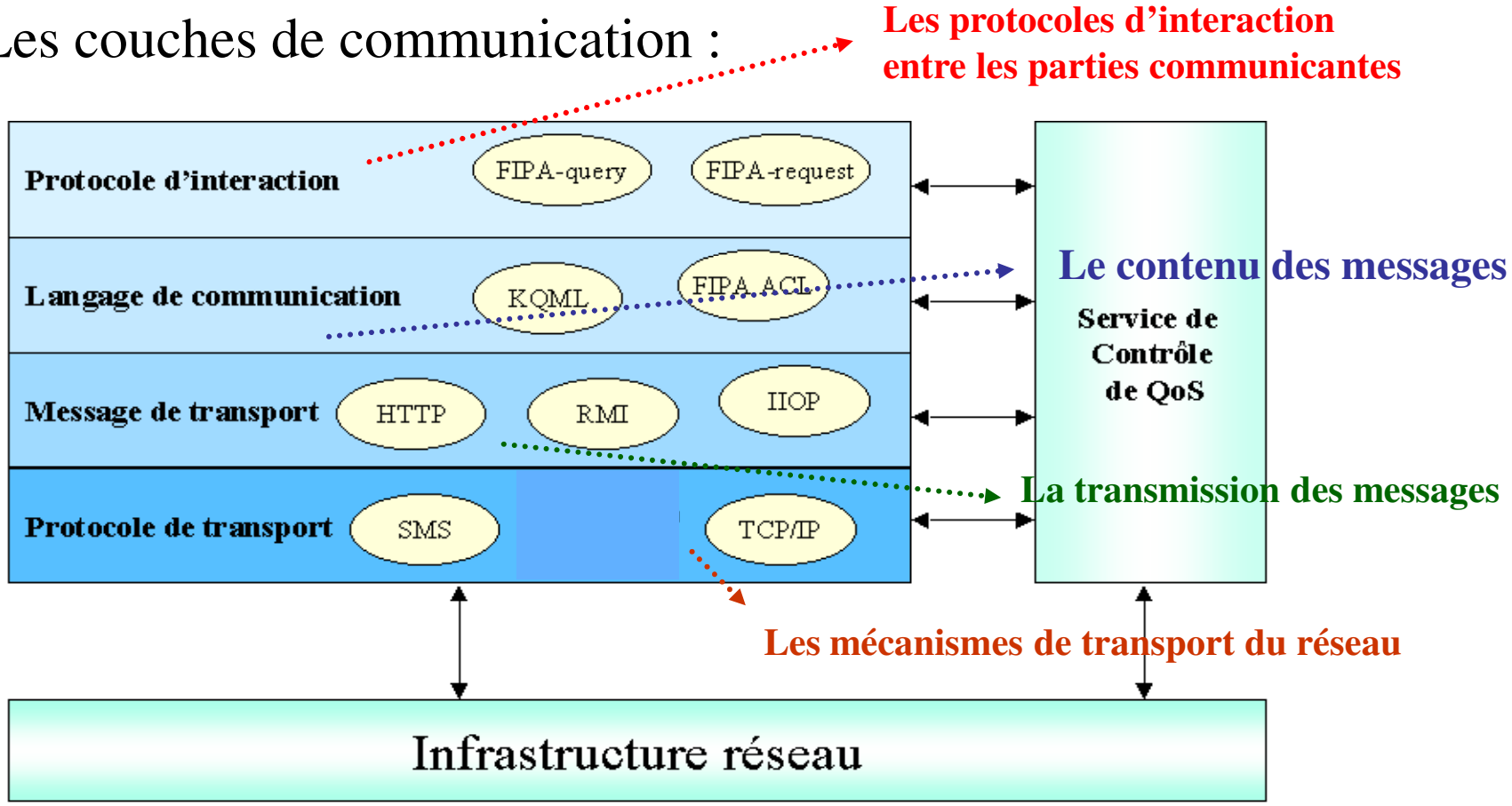


Les couches de communication

- La communication entre agents dans un environnement sans fil peut être divisée en quatre couches :
  - **La couche d'Interaction**
  - **La couche de Langage de Communication**
  - **La couche de Message**
  - **La couche de Transport**

Communication des agents dans un environnement sans fil

- La communication entre agents dans un environnement sans fil peut s'effectuer à différents niveaux et de différentes façons.
- Les couches de communication :



Mes publications dans ce domaine

Articles de revues internationales ou chapitres de livres avec comité de lecture et de sélection (5)

- **B. Benmammar,** Z. Jrad and F. Krief. "QoS management in mobile IP networks using a terminal assistant". **Wiley/ACM International Journal of Network Management**, Wiley InterScience Edition, ISSN (printed): 1055-7148, ISSN (electronic): 1099-1190, Volume 19, Issue 1, Date: January/February 2009, Pages: 1-24.
- **B. Benmammar** and Z. Jrad. "Intelligent Interfaces and Mobile Communications" **(Book Chapter)**, Communicating embedded systems: Network applications, Wiley-ISTE Edition, January 2010, Pages: 271-303, ISBN: 978-1-84821-144-5.
- **B. Benmammar**. "Agents and 3rd and 4th mobile generation" **(Book Chapter)**, Autonomic Networks, Wiley-ISTE Edition, December 2007, Pages: 233-264, ISBN: 978-1-84821-002-8.
- **B. Benmammar** et Zeina Jrad. "Les interfaces intelligentes et les communications mobiles" **(Chapitre de livre)**. **Traité IC2, série réseaux et télécoms,** les Systèmes embarqués communicants: mobilité, sécurité, autonomie. Édition Hermès Science, septembre 2008, Pages: 259-287. ISBN 2746218739, 9782746218734.
- **B. Benmammar**. "Agents et mobiles de troisième et quatrième générations" **(Chapitre de livre)**. **Traité IC2, série réseaux et télécoms,** Intelligence dans les réseaux. Édition Hermès Science, juillet 2005, Pages: 253-286. ISBN 2746211343, 9782746211346.

Mes publications dans ce domaine

Articles d'actes de conférences internationales avec comité de lecture et de sélection (5)

- **Badr Benmammar**, Nader Mbarek and Francine Krief. "Signaling Environment for Service Level Negotiation and QoS management in Mobile IP Networks". Proceedings of the 14th International Conference on Telecommunication Systems - Modeling and Analysis (ICTSM2006). October 5-8, 2006. Penn State University - Berks, Reading, PA. ISBN: 0-9716253-2-8; S. 332-335.
- Nancy Samaan, **Badr Benmammar**, F. Krief, A. Karmouch. "Prediction-based Advanced Resource Reservation in a Mobile Environment". Proceedings of the 18th **IEEE** Annual Canadian Conference on Electrical and Computer Engineering, CCECE05, May 1-4, 2005, Saskatoon Inn, Saskatoon, Saskatchewan Canada, Print ISBN: 0-7803-8885-2, Digital Object Identifier : 10.1109/CCECE.2005.1557244.
- Zeina Jrad, **Badr Benmammar**, Joseph Corréa, Francine Krief, Nader Mbarek. "A User Assistant for QoS Negotiation in a Dynamic Environment Using Agent Technology". Proceedings of the Second **IEEE and IFIP** International Conference on Wireless and Optical Communications Networks WOCN 2005. March 6 - 8, 2005, Hyatt Regency Hotel, Dubai, United Arab Emirates UAE, Print ISBN: 0-7803-9019-9, Digital Object Identifier: 10.1109/WOCN.2005.1436032.
- **B. Benmammar** and F. Krief. "Agents for Wireless Environments". Proceedings of the International Conference on Telecommunication Systems, Modeling and Analysis. ICTSM'2004. **IFIP** WG 7.3. Monterey, USA. Jully 2004. ISBN: 0971625352.
- Z. Jrad, F. Krief and **B. Benmammar**. "An Intelligent User Interface for the Dynamic Negotiation of QoS". Proceedings of the 10th **IEEE** International Conference on Telecommunications. ICT'2003. Papeete, Tahiti. February 2003, Print ISBN: 0-7803-7661-7.

Mes publications dans ce domaine

Articles d'actes de conférences nationales avec comité de lecture et de sélection (2)

- **B. Benmammar** et F. Krief. "Gestion dynamique du handover horizontal et vertical basée sur le profil de mobilité de l'utilisateur". Dans les actes du Colloque GRES 2005 : Gestion de REseaux et de Services, Du 28 Février au 3 Mars à LUCHON, France.
- **B. Benmammar** et F. Krief. "La Technologie Agent et les Réseaux Sans Fil". Dans les actes du 17 ème Congrès Des Nouvelles Architectures pour les Communications. DNAC'2003. Paris, France. Octobre 2003.

Rapports de Recherche (3)

- **Badr Benmammar**, Zeina Jrad, Francine Krief, Nader Mbarek. "Dynamique de l'environnement : Scénarios, simulations et maquette". IP-SIG/LIV/4.2. Contrat RNRT IPSIG. 2005.
- **Badr Benmammar**, Joseph Corréa, Zeina Jrad, Francine Krief, Nader Mbarek. "Dynamique de l'environnement". IP-SIG/LIV/4. Contrat RNRT IP-SIG. 2004.
- **Badr Benmammar**, Nguyen Thi Mai Trang, Guy Pujolle, Vedat Yilmaz. "Définition d'un SLA/SLS". IP-SIG/LIV/1.Contrat RNRT IP-SIG. 2003.

Mémoires (2)

- **B. Benmammar**. "La gestion dynamique de la qualité de service dans les réseaux IP mobiles". Rapport de thèse en Informatique. Laboratoire Bordelais de Recherche en Informatique. Université Bordeaux 1. Mai 2006. ***Num. national de thèse : 2006BOR13160***.
- **Badr Benmammar**. "Négociation dynamique de niveau de service entre un utilisateur final et un fournisseur de service". Mémoire de Master Recherche en Intelligence Artificielle et Optimisation Combinatoire. Laboratoire d'Informatique de Paris Nord - Université Paris 13. Septembre 2002.